Online Bayesian Inference and Learning of Gaussian-Process State-SpaceModels

Berntorp, Karl

TR2021-026 April 04, 2021

Abstract

This paper addresses the recursive joint inference (state estimation) and learning (system identification) problem for nonlinear systems admitting a state-space formulation. We model the system as a Gaussian-process state-space model (GP-SSM) and leverage a recently developed reduced-rank formulation of GP-SSMs to enable efficient, online learning. The unknown dynamical system is expressed as a basis-function expansion, where a connection to the GP makes it possible to systematically assign priors to the basis-function weights. The approach is formulated within the sequential Monte-Carlo framework, wherein each particle retains its own weights of the basis functions, which are updated recursively as measurements arrive. We report competitive results when compared to a state-of-the art offline Bayesian learning method. We also apply the method to a case study concerning tire-friction estimation. The results indicate that our method can accurately learn the tire friction using automotive-grade sensors in an online setting, and quickly detect sudden changes of the road surface.

Automatica

Online Bayesian Inference and Learning of Gaussian-Process State-Space Models

Karl Berntorp [a]

[a] *Mitsubishi Electric Research Laboratories (MERL), 02139 Cambridge, MA, USA. Email:* `karl.o.berntorp@ieee.org`

Abstract

This paper addresses the recursive joint inference (state estimation) and learning (system identification) problem for nonlinear systems admitting a state-space formulation. We model the system as a Gaussian-process state-space model (GP-SSM) and leverage a recently developed reduced-rank formulation of GP-SSMs to enable efficient, online learning. The unknown dynamical system is expressed as a basis-function expansion, where a connection to the GP makes it possible to systematically assign priors to the basis-function weights. The approach is formulated within the sequential Monte-Carlo framework, wherein each particle retains its own weights of the basis functions, which are updated recursively as measurements arrive. We report competitive results when compared to a state-of-the art offline Bayesian learning method. We also apply the method to a case study concerning tire-friction estimation. The results indicate that our method can accurately learn the tire friction using automotive-grade sensors in an online setting, and quickly detect sudden changes of the road surface.

Key words: System identification, Nonlinear models, Bayesian learning, Gaussian processes, Monte Carlo methods.

1 Introduction

System identification [20] concerns learning models of dynamical systems from data, oftentimes with an a priori assumed model structure to facilitate learning. Gaussian processes (GPs) by now have a long history in nonlinear function learning, with numerous successful applications [31]. Unlike some of the more traditional system-identification methods, GPs are nonparametric modeling tools, which imply flexibility in the ability to model general nonlinear functions without a priori enforcing an explicit parametric structure. In recent years there have been several contributions enabling GPs for learning in nonlinear dynamical systems (e.g., [1, 21, 29]).

Modeling the state-transition $\boldsymbol{f}$ and observation function $\boldsymbol{h}$ as GPs, leading to GP state-space models (GP-SSMs) [15, 23, 37], has shown to be an efficient modeling approach to learn systems from uncertain data. GPs make it possible to reason about uncertainty in the learning process, both uncertainty in the data used in the learning and uncertainty of the learned model. For example, as noted in [22], GP-based models can propagate uncertainty information such that the estimates do not become overly confident in regions in the state space where data are limited, while avoiding the common issue of overfitting to data [35].

Since system identification needs data, most methods are offline methods; that is, they process a sufficiently long batch of data to learn the system that is observed. While this approach is the sensible one for systems where the dynamics is constant over time, or at least very slowly time varying, many systems have nonstationary uncertainties such that recursive approaches are most appropriate. Two examples are in automotive [6] and target tracking [26]. While there has been much effort in online parametric learning, for example, in learning slowly time varying parameters for both linear [24] and nonlinear [27] systems, the research on recursive learning using GPs for nonparametric uncertainties is limited, although there have been some efforts, for example, using sparse representations of the GP [8].

To address this, we consider learning of GP-SSMs in an online setting. GP-SSMs encompass dynamical systems that can be written in the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{1a}$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k, \tag{1b}$$

where the (latent) state $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ at each time step k is observed through the measurement $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$. The nonlinear functions $\boldsymbol{f} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $\boldsymbol{h} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are assumed to be realizations from GPs. The process noise $\boldsymbol{w}_k$ and measurement noise $\boldsymbol{e}_k$ are Gaussian distributed with covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$ according to $\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$ and $\boldsymbol{e}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$, respectively.

Learning in GP-SSMs amounts to estimating the posterior distributions of $\boldsymbol{f}$, $\boldsymbol{h}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, and the hyperparameters associated with the GPs, and is hard for several reasons. First, the state is only implicitly observed through the measurement model (1b), and the quality of the state estimates affects the model identification, and vice versa, which necessitates including state inference in the learning process. Second, since the SSM (1) is nonlinear and at least partially unknown, the state inference problem is nonlinear and cannot be solved with analytic methods. In the offline (batch) setting, this can be solved by iterative procedures, such as particle Markov-chain Monte-Carlo (PMCMC) [14,35], where state estimation over the data set and subsequent model updates are iterated until convergence. However, in the online setting the inference and learning problem must be addressed simultaneously, at each time step.

For these reasons, we develop a particle-filter (PF, [11]) based approach for jointly estimating *recursively* the state trajectory and the SSM (1) interpreted as a GP-SSM. [1] To enable online learning with GPs, we leverage a recently developed reduced-rank model formulation [33, 35] of the GP-SSM, in which connections between GPs and a finite basis-function expansion of the unknown system dynamics is made by introducing priors on the basis-function coefficients. We adapt the work in [35], which treated offline learning in a PMCMC setting, to the online setting, by tailoring a PF to GP-SSMs. A key enabler for the approach is that each particle carries its own set of statistics associated with the system dynamics. By adopting the interpretation of a GP as a basis-function expansion [35], with a proper implementation of the PF and suitable approximations, learning of GP-SSMs can indeed be done online.

Notation: Throughout, for a vector $\boldsymbol{x}$, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ indicates that $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ and x_n denotes the nth component of $\boldsymbol{x}$. Matrices are indicated in capital bold font as $\boldsymbol{X}$, and the element on row i and column j of $\boldsymbol{X}$ is denoted with X_{ij}. The determinant of $\boldsymbol{X}$ is $|\boldsymbol{X}|$, the trace of $\boldsymbol{X}$ is $\mathrm{Tr}(\boldsymbol{X})$, and the Cholesky decomposition of $\boldsymbol{X}$ is $\mathrm{chol}(\boldsymbol{X})$. We use the notation $\boldsymbol{\theta}_{k|k}$ to denote the estimate of a set of parameters $\boldsymbol{\theta}$ at time step k given $\{\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k}\}$, and $\boldsymbol{\theta}_{k|k-1}$ denotes the one-step prediction of $\boldsymbol{\theta}_{k-1|k-1}$. With $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{0:k})$, we mean the posterior density function of the state trajectory $\boldsymbol{x}_{0:k}$ from time step 0 to time step k given the measurement sequence $\boldsymbol{y}_{0:k} := \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_k\}$, and $\boldsymbol{x}^i_{0:k}$ is the ith realization of $\boldsymbol{x}_{0:k}$. The notation $f \sim \mathcal{GP}(0, \kappa(x, x'))$ means that the function $f(x)$ is a realization from a GP prior with a given covariance function $\kappa(x, x')$ subject to some hyperparameters $\boldsymbol{\vartheta}$, and $\mathcal{IW}(\nu, \boldsymbol{\Lambda})$ is the inverse-Wishart distribution with degree of freedom ν and scale matrix $\boldsymbol{\Lambda}$. We write $\mathcal{T}(\nu, \boldsymbol{M}, \boldsymbol{\Lambda})$ to denote the multivariate Student-t distribution, and $\mathcal{MN}(\boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{V})$, $\mathcal{MT}(\nu, \boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{V})$ are the Matrix-Normal and Matrix-variate Student-t distribution, respectively, with mean $\boldsymbol{M}$, right covariance (scale) $\boldsymbol{Q}$, and left covariance (scale) $\boldsymbol{V}$ Finally, the compund distribution $\mathcal{MNIW}(\boldsymbol{A}, \boldsymbol{Q}|\boldsymbol{M}, \boldsymbol{V}, \boldsymbol{\Lambda}, \nu) = \mathcal{MN}(A|\boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{V})\mathcal{IW}(\boldsymbol{Q}|\nu, \boldsymbol{\Lambda})$.

Outline: Sec. 2 connects the current paper to previous, related research. Sec. 3 outlines the modeling framework and the problem formulation based on the reduced-rank GP-SSM. In Sec. 4 we present the proposed method. Sec. 5 presents a numerical evaluation of the proposed method. Application of the algorithm to the problem of real-time tire-friction estimation for autonomous vehicles is presented in Sec. 6, with both synthetic and experimental data. Finally, Sec. 7 concludes the paper.

2 Related Work

Interesting work related to GP-based system identification includes impulse response estimation [29], nonlinear ordinary differential equations [21], and force modeling [1]. Modeling the system as a GP-SSM has been done in several previous papers—for example, [13], where a sparse GP representation is used for learning, and [22], in which a variational inference approach is taken. An early example related to basis-function expansions is [16], which uses radial basis functions and extended Kalman filtering in an expectation-maximization (EM) algorithm. A more recent method, more closely aligned with the approach in this paper, is presented in [36], in which both an offline and an online method are detailed, similar to us leveraging particle filtering for the state inference and parameter learning. However, a major difference compared to our method is that [36] uses a Gaussian kernel basis-function expansion whose weights are learned using an MCMC approach, based on the maximum-likelihood state estimate from the PF. Our approach also differs to that taken in [8], which uses a sparse approximation of the GP instead of a basis-function expansion.

Approaches based on PMCMC methods [2] related to the work in this paper have been developed in a series of papers. The work in [14] is fully Bayesian and introduces PMCMC with the GP-SSM framework, and [13] extends this to a variational inference scheme using inducing points and sparse GPs. Another fully Bayesian learning approach was presented in [35], where the reduced-rank GP approximation introduced in [33] was used in

[1] This work extends previous preliminary work in [4,5]. The current version contains a detailed explanation of the algorithm developments, an extension to online adaptation of the GP hyperparameters, a significantly extended evaluation, and an experimental section relating to tire-friction estimation.

combination with a tailored PMCMC scheme to enable efficient nonparametric fully Bayesian learning. Moreover, [35] also includes an alternative EM cheme based on the same framework. The present paper adapts the reduced-rank formulation for learning of GP-SSMs to the online setting, by providing a recursive implementation using a tailored marginalized PF.

For recursive joint state inference and learning, particularly related approaches are those based on particle filtering. Notable contributions include [34], which is based on augmentation of the state with the parameters to be estimated, and [10], which considers static parameter estimation using the marginalized PF. To overcome the path degeneracy of estimating static quantities in PFs, [27] considers the role of exponential forgetting in an adaptive marginalized PF. This work has later been extended to dependent process and measurement noise in [6]. These recursive joint inference and learning methods all consider parametric uncertainties and do not introduce GPs in the estimation process. However, since we rely on a reduced-rank formulation, the proposed learning scheme in the end estimates weights of basis functions, effectively making it parametric for any realistic implementation. Hence, the methodology and underlying idea of the method is similar to, for example, [6,27].

3 Modeling and Problem Formulation

We rely on GPs to encode prior information for learning of the SSM. We focus on learning of $\boldsymbol{f}$ and $\boldsymbol{Q}$, and treat $\boldsymbol{h}$ and $\boldsymbol{R}$ as known. There are two main reasons for having a known observation model. First, introducing too many unknowns might have implications on observability of the system, and simultaneous learning of the state-transition function and measurement function can be a daunting task. Second, $\boldsymbol{h}$ usually corresponds to a sensor model that typically is known, at least up to some parametric uncertainty that can be determined a priori. Note, however, that interchanging $\boldsymbol{f}$ with $\boldsymbol{h}$ (i.e., instead assuming a known $\boldsymbol{f}$) is conceptually straightforward. Similarly, we ignore any known inputs $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ but those can also be added to the learning process, which we illustrate with a numerical example in Sec. 5.

3.1 Modeling the State-Transition Function

We express $\boldsymbol{f} = [f_1 \ \cdots \ f_{n_x}]^\top$ by a basis-function expansion according to

$$\hat{f}_i(\boldsymbol{x}) = \sum_{j=1}^{M} \gamma_{ij}\phi_j(\boldsymbol{x}) \tag{2}$$

for each $i = 1, 2, \ldots, n_x$, for a known set of basis functions, where the weights γ_{ij} are to be determined. To enable tractability in the learning problem and not introduce too much flexibility, we assume that the order n_x of $\boldsymbol{f}$ is known and that the number of basis functions M is set a priori. By employing a basis-function expansion (2), we can formulate a joint inference and learning approach that is linear in the parameters (i.e., the weights γ_{ij}). Together with the reduced-rank formulation of a GP-SSM, this will enable computationally tractable online inference and learning.

Remark 1 *Setting M small will limit the expressiveness of the basis-function expansion and therefore restrict the set of functions $\boldsymbol{f}$ we can model. Since we use GPs for formulating priors on $\boldsymbol{f}$, a viable option is to set M as large as the implementation permits, as also noted in [35], and let the estimator determine which γ_{ij} to use.*

3.1.1 Setting the GP-Priors

While there are a number of choices for how to choose basis functions, to make connections to GPs, we choose the eigenfunctions with associated eigenvalues to the Laplace operator, which for $n_x = 1$ defined on a closed interval $[-L, L] \in \mathbb{R}$ equal

$$\phi_j(x) = \frac{1}{\sqrt{L}} \sin\left(\frac{\pi j(x+L)}{2L}\right), \tag{3a}$$

$$\lambda_j = \left(\frac{\pi j}{2L}\right)^2. \tag{3b}$$

In the multi-dimensional case ($n_x > 1$), the corresponding eigenfunctions and eigenvalues defined on the domain $[-L_1, L_1] \times \cdots \times [-L_{n_x}, L_{n_x}] \in \mathbb{R}^{n_x}$ are

$$\phi_{j_1,\ldots,j_{n_x}} = \prod_{n=1}^{n_x} \frac{1}{\sqrt{L_n}} \sin\left(\frac{\pi j_n(x_n+L_n)}{2L_n}\right), \tag{4a}$$

$$\lambda_{j_1,\ldots,j_{n_x}} = \sum_{n=1}^{n_x} \left(\frac{\pi j_n}{2L_n}\right)^2. \tag{4b}$$

Note that for $n_x = 1$, (3) equals (4). From (4a), it is clear that the number of weights to learn nominally increases exponentially with the state dimension as $M = m^{n_x}$, where m is the number of parameters for $n_x = 1$. While this is a common unwanted feature for many basis-function expansions, it is possible to alleviate. In [35], two possibilities are pointed out. First, to assume independence between the different dimensions. Second, to choose another set of basis function. More detailed treatment of the dimensionality aspects is beyond the scope of this paper, but could be interesting future research.

For isotropic covariance functions κ, that is, for covariance functions that are a function of $r = |\boldsymbol{x} - \boldsymbol{x}'|$, [33] provides a connection between GPs and a basis-function expansion of a function f with basis functions (3a). Isotropic covariance functions can be equivalently represented in terms of the spectral density $S(\omega)$ of κ. In particular, the connection between the basis-function expansion (2) with basis functions (4a) can be written as

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(0, \kappa(\boldsymbol{x}, \boldsymbol{x}')) \Leftrightarrow \boldsymbol{f}(\boldsymbol{x}) \approx \sum_{j=1}^{M} \boldsymbol{\gamma}_j \phi_j(\boldsymbol{x}), \quad (5)$$

with $\boldsymbol{\gamma}_j = [\gamma_{1j} \ \cdots \ \gamma_{n_x j}]^\top$ and

$$\gamma_{ij} \sim \mathcal{N}(0, \mathcal{S}(\lambda_j)). \quad (6)$$

We use the squared exponential covariance function,

$$\kappa(r) = \sigma^2 \exp\left(-\frac{r^2}{2\ell^2}\right), \quad (7)$$

with hyperparameters $\boldsymbol{\vartheta} = \{\sigma, \ell\}$, where for simplicity we assume the same hyperparameters for each dimension. The covariance (7) has the spectral density

$$\mathcal{S}(\omega) = \sigma^2 \sqrt{2\pi\ell^2} \exp -\frac{\pi^2 \ell^2 \omega^2}{2}. \quad (8)$$

Hence, with the connection (5), the spectral density can be used to assign suitable priors on the weights in the basis-function expansion (3a), and effectively enabling online learning of GP-SSMs. For instance, the degree of smoothness of the prior determines how likely the function $\boldsymbol{f}$ is to rapidly change or not. In general, if a different L_n is used for the different dimensions, so are the associated hyperparameters $\boldsymbol{\vartheta}$.

Remark 2 *The squared exponential covariance function* (7) *is infinitely differentiable, implying that the corresponding GP is smooth. While the squared exponential is probably the most widely used covariance function, for some physical systems the smoothness makes it unrealistic [31]. In this case, the Matérn class of covariance functions might be more appropriate. Note that the proposed framework is not requiring the squared exponential, but is used since it works sufficiently well for our purposes.*

Remark 3 *The basis-function expansion with the covariance function* (7) *depends on ℓ, L_n, and σ, all of which impact the behavior of the weights as well as the basis functions themselves. These parameters can often be chosen a priori using expert knowledge of the system to be determined, and we give a practical example of this in Sec. 6. In some cases, such expert knowledge might not be available. If prior data are available, the parameters can be estimated offline using, e.g., the method in [35]. The choice of L_n directly affects the basis functions and is also connected to the GP hyperparameters through* (4b) *entering the weight prior* (6). *Hence, online learning of all these parameters as well as the weights, which are also connected, and the state itself, will be difficult both from an algorithmic standpoint and due to observability/identifiability. However, if determining L_n a priori, the GP hyperparameters can be adjusted. We propose such an adaptation scheme in Sec. 4.3.*

3.2 The Reduced-Rank GP-SSM Summary

With the basis-function expansion (2) and basis functions (4), a reduced-rank GP-SSM (1a) is

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1m} \\ \vdots & & \vdots \\ \gamma_{n_x 1} & \cdots & \gamma_{n_x m} \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \phi_1(\boldsymbol{x}_k) \\ \vdots \\ \phi_M(\boldsymbol{x}_k) \end{bmatrix}}_{\boldsymbol{\varphi}(\boldsymbol{x}_k)} + \boldsymbol{w}_k. \quad (9)$$

In the limit, (9) converges to (1) [35].

Theorem 1 *The reduced-rank GP-SSM* (9) *convergences in distribution to* (1) *when the size of the domain and M tend to infinity.*

Remark 4 *Theorem 1 is for $M \to \infty$, but for a reasonably large M, the approximation effects are negligible. For the examples in this paper, about 10 basis functions in each dimension gives negligible performance reductions, while maintaining reasonable computational load.*

For convenience, we express the prior on the coefficients γ_{ij} in (6) at time step $k = 0$ as a zero-mean Matrix-normal ($\mathcal{MN}$) distribution [9] over $\boldsymbol{A}$,

$$\boldsymbol{A} \sim \mathcal{MN}(\boldsymbol{0}, \boldsymbol{Q}, \boldsymbol{V}), \quad (10)$$

with right covariance $\boldsymbol{Q}$ and left covariance $\boldsymbol{V}$ with diagonal elements $\mathcal{S}(\lambda_j)$, which incorporates the prior (6). The density of the $\mathcal{MN}$ distribution is

$$\mathcal{MN}(\boldsymbol{A}|\boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{V}) = \frac{1}{(2\pi)^{n_x M/2} |\boldsymbol{V}|^{n_x/2} |\boldsymbol{Q}|^{M/2}} \cdot \exp\left(-\frac{1}{2} \mathrm{tr}\left((\boldsymbol{A} - \boldsymbol{M})^{\mathrm{T}} \boldsymbol{Q}^{-1} (\boldsymbol{A} - \boldsymbol{M}) \boldsymbol{V}^{-1}\right)\right). \quad (11)$$

We put an inverse-Wishart ($\mathcal{IW}$) prior on $\boldsymbol{Q}$ according to $\boldsymbol{Q} \sim \mathcal{IW}(\nu_0, \boldsymbol{\Lambda}_0)$, with the density given by

$$\mathcal{IW}(\boldsymbol{Q}|\nu, \boldsymbol{\Lambda}) = \frac{|\boldsymbol{\Lambda}|^{\nu/2} |\boldsymbol{Q}|^{-(n_x + \nu + 1)/2}}{2^{\nu n_x/2} \Gamma_{n_x}(\nu/2)} \cdot \exp\left(-\frac{1}{2} \mathrm{tr}(\boldsymbol{Q}^{-1} \boldsymbol{\Lambda})\right), \quad (12)$$

where $\Gamma_{n_x}(\cdot)$ is the multivariate gamma function. Assuming the covariance prior to be $\mathcal{IW}$ distributed is common in covariance estimation due to its beneficial properties [6, 27, 35].

Note that if there is prior knowledge about the physical system underlying (1), it can be incorporated into (9). Two common examples of prior knowledge are that only parts of the system are unknown, and that $\boldsymbol{f}$ is approximately known (e.g., from an initialization process). For both cases, (9) can be modified to account for this.

3.3 Problem Formulation

We consider approximate joint state inference and learning of the GP-SSM (1a) with known measurement equation (1b) (the extension to unknown $\boldsymbol{h}$ is similar to the setting with unknown $\boldsymbol{f}$). Instead of targeting (1a), we learn (9). In a fully Bayesian setting, this implies estimating the posterior distributions of $\boldsymbol{x}$, $\boldsymbol{A}$, and $\boldsymbol{Q}$, at each time step k. Since we are aiming for online applications, we are interested in approximating the marginal (filtering) distributions.

The involved distributions cannot be resolved analytically, and we therefore approach the estimation of $\boldsymbol{x}$, $\boldsymbol{A}$, and $\boldsymbol{Q}$ in a sequential Monte-Carlo (SMC) setting, where we tailor a PF for approximating $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ and $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{0:k})$, where $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{Q}\}$.

As is the case for most filtering problems, we assume $p(\boldsymbol{x}_0)$ to be known. We assign a suitable prior $p(\boldsymbol{\theta}_0)$ on $\boldsymbol{\theta}$ that is dependent on the hyperparameters $\boldsymbol{\vartheta}$ affecting the spectral density (e.g., (8) for the squared exponential kernel). Initially, we assume the hyperparameters $\boldsymbol{\vartheta}$ of said prior to be known. However, in Sec. 4.3 we address different ways to relax this assumption.

Remark 5 *In practice, the priors $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta}_0)$ can be deduced based on expert knowledge of the system at hand, and we give a practical example of this in Sec. 6. Alternatively, since the method proposed in this paper is an online extension of the method proposed in [35], that method can be used to initialize some of the priors and hyperparameters assumed known in this paper.*

4 Online Bayesian Inference and Learning

In this section, we focus on approximating the joint posterior density $p(\boldsymbol{x}_{0:k+1}, \boldsymbol{\theta}_k|\boldsymbol{y}_{0:k})$, [2] from which marginal densities $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ and $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{0:k})$ can be computed. We decompose the joint posterior as

$$p(\boldsymbol{x}_{0:k+1}, \boldsymbol{\theta}_k|\boldsymbol{y}_{0:k}) = p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k})p(\boldsymbol{x}_{0:k+1}|\boldsymbol{y}_{0:k}). \tag{13}$$

We will go through the main steps in the proposed algorithm for approximating (13). First, we explain how to recursively update the parameters $\boldsymbol{\theta}_k$ and compute $p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k})$ given the trajectory $\{\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k}\}$. Utilizing the priors (11), (12), this step can be done analytically. Second, we describe how to approximate $p(\boldsymbol{x}_{0:k+1}|\boldsymbol{y}_{0:k})$ using a tailored PF, which utilizes that the analytic expression for $p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k})$ enables marginalizing out $\boldsymbol{\theta}_k$ in the prediction step.

[2] Because $\boldsymbol{\theta}_k$ depends on $\boldsymbol{x}_{k+1}$, this modified joint posterior is more appropriate than $p(\boldsymbol{x}_{0:k}, \boldsymbol{\theta}_k|\boldsymbol{y}_{0:k})$.

4.1 Estimating the Parameter Posterior

The distribution of $\boldsymbol{\theta}_k$ is computed conditioned on the realization of the state and measurement trajectories for each particle. For a specific realization $\boldsymbol{x}_{0:k+1}$, the posterior density of $\boldsymbol{\theta}_k$ can be written according to

$$p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k}) = p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}), \tag{14}$$

since (1b) is not dependent on $\boldsymbol{\theta}_k$. Using Bayes' rule, (14) can be decomposed into a likelihood and prior as

$$p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}) \propto p(\boldsymbol{x}_{k+1}|\boldsymbol{\theta}_k, \boldsymbol{x}_{0:k})p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k}). \tag{15}$$

The first term on the right-hand side of (15) is the transition density of the reduced-rank model (9),

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{\theta}_k, \boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_{k+1}|\boldsymbol{A}_k\boldsymbol{\varphi}(\boldsymbol{x}_k), \boldsymbol{Q}_k). \tag{16}$$

To get a recursive expression for updating (15), we leverage that the $\mathcal{MNIW}$ distribution is a conjugate prior for the considered model.

Theorem 2 *Suppose that the initial prior $p(\boldsymbol{\theta}_0)$ is distributed according to $p(\boldsymbol{\theta}_0) = \mathcal{MNIW}(\boldsymbol{\theta}_0|\boldsymbol{0}, \boldsymbol{V}, \boldsymbol{\Lambda}_0, \nu_0)$. Define $\boldsymbol{\Xi} := \boldsymbol{\Sigma}_{k|k} + \boldsymbol{V}^{-1}$. Then, for realizations $\boldsymbol{x}_{0:k+1}$ and $\boldsymbol{y}_{0:k}$, (14) can be computed at each time step k as*

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k}) = \mathcal{MNIW}(&\boldsymbol{\theta}_k|\boldsymbol{\Psi}_{k|k}\boldsymbol{\Xi}^{-1}, \\ &\boldsymbol{\Xi}^{-1}, \boldsymbol{\Lambda}_0 + \boldsymbol{\Phi}_{k|k} - \boldsymbol{\Psi}_{k|k}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k}^\top, \nu_{k|k}), \end{aligned} \tag{17}$$

where $\boldsymbol{\Psi}_{k|k}$, $\boldsymbol{\Phi}_{k|k}$, $\boldsymbol{\Sigma}_{k|k}$, and $\nu_{k|k}$ can be recursively updated as

$$\boldsymbol{\Phi}_{k|k} = \boldsymbol{\Phi}_{k|k-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^\top, \tag{18a}$$
$$\boldsymbol{\Psi}_{k|k} = \boldsymbol{\Psi}_{k|k-1} + \boldsymbol{x}_{k+1}\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top, \tag{18b}$$
$$\boldsymbol{\Sigma}_{k|k} = \boldsymbol{\Sigma}_{k|k-1} + \boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top, \tag{18c}$$
$$\nu_{k|k} = \nu_{k|k-1} + 1. \tag{18d}$$

PROOF. See Appendix.

For constant parameters, the time-update step is $(*)_{k|k-1} = (*)_{k-1|k-1}$ for all quantities in (18). However, for PFs it can be problematic to estimate static parameters. Using the principle of exponential forgetting [3], for slowly time-varying parameters an approach is to write the time-update step of the predictive statistics as

$$\boldsymbol{\Phi}_{k|k-1} = \lambda \boldsymbol{\Phi}_{k-1|k-1}, \tag{19a}$$
$$\boldsymbol{\Psi}_{k|k-1} = \lambda \boldsymbol{\Psi}_{k-1|k-1}, \tag{19b}$$
$$\boldsymbol{\Sigma}_{k|k-1} = \lambda \boldsymbol{\Sigma}_{k-1|k-1}, \tag{19c}$$
$$\nu_{k|k-1} = \lambda \nu_{k-1|k-1}. \tag{19d}$$

The forgetting factor $0 \leq \lambda \leq 1$ helps in the estimation of dynamic variables by forgetting older data as new data are accumulated in time. As a rule of thumb, for a forgetting factor λ the update of the statistics will rely on the last $1/(1-\lambda)$ time instants.

To find the posterior distribution of $\boldsymbol{\theta}_k$, we marginalize out the state trajectory according to

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{0:k}) &= \int p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k}) p(\boldsymbol{x}_{0:k+1}|\boldsymbol{y}_{0:k})\, \mathrm{d}\boldsymbol{x}_{0:k+1} \\ &\approx \sum_{i=1}^{N} q_k^i p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}^i, \boldsymbol{y}_{0:k}), \end{aligned} \tag{20}$$

where $p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}^i, \boldsymbol{y}_{0:k})$ is given by (17) and q_k^i is the particle weight, see Sec. 4.2.

Remark 6 *The scalar real-valued number $\lambda \in [0,1]$ provides exponential forgetting in $p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}, \boldsymbol{y}_{0:k})$ that allows the algorithm to adapt to (slowly time-varying) changes in $\boldsymbol{\theta}$ over time. It also mitigates path degeneracy since for $\lambda < 1$, mixing is introduced in the model [27], which explains why the PF works for marginal distributions [11]. For $\lambda = 1$ we recover the case of static parameters, however, with increased risk of path degeneracy in a practical implementation for a finite N.*

Remark 7 *In this paper we assume that λ is known, which is consistent with previous approaches in the literature (e.g., [27, 32]). One reason for assuming λ known is that in practice it can often be determined from expert knowledge, as we will discuss in Sec. 6. Another reason for assuming a known λ is more practical. In an online algorithm, introducing too many variables and parameters to estimate impacts the observability/identifiability of the system, and also inevitably increases complexity of the estimation algorithm.*

4.2 Particle Filtering for State Inference

SMC methods, such as PFs, constitute a class of techniques that estimates the posterior distribution in SSMs, and SMCs have recently proved useful in learning of SSMs [19]. PFs approximate the posterior density $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{0:k})$ by a set of N weighted trajectories,

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{0:k}) \approx \sum_{i=1}^{N} q_k^i \delta_{\boldsymbol{x}_{0:k}^i}(\boldsymbol{x}_{0:k}), \tag{21}$$

where q_k^i is the importance weight of the ith state trajectory $\boldsymbol{x}_{0:k}^i$ and $\delta(\cdot)$ is the Dirac delta mass. The trajectories $\boldsymbol{x}_{0:k}^i$ are sampled recursively from a tractable, user-designed *proposal distribution* $\pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k})$, and the weights $\{q_k^i\}_{i=1}^N$ are updated as

$$q_k^i \propto q_{k-1}^i \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}^i, \boldsymbol{y}_{0:k-1}) p(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{0:k-1})}{\pi(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{0:k})}. \tag{22}$$

If the proposal is chosen as the predictive density,

$$\pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{0:k-1}), \tag{23}$$

the weight update (22) simplifies to

$$q_k^i \propto q_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_k^i). \tag{24}$$

The PF algorithm iterates between prediction and weight update, combined with a resampling step that removes particles with low weights and replaces them with more likely particles. We assume a known measurement model, and the weight update (24) therefore uses the standard Gaussian likelihood. However, since the transition density depends on $\boldsymbol{\theta}$, the prediction step is nonstandard. We predict the state trajectory by sampling from the predictive density (23). From marginalization of the unknown quantity $\boldsymbol{\theta}$, (23) can be expanded for each particle i as

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k-1}) = \int & p(\boldsymbol{x}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{x}_{k-1}) \\ & p(\boldsymbol{\theta}_{k-1}|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k-1}) \mathrm{d}\boldsymbol{\theta}_{k-1}. \end{aligned} \tag{25}$$

By assumption, the second density in the integrand of (25) is $\mathcal{MNIW}$ distributed. The integrand can be written as the hierarchical model

$$\begin{aligned} \boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{A}_{k-1}, \boldsymbol{Q}_{k-1} &\sim \mathcal{N}(\boldsymbol{A}_{k-1}\boldsymbol{\varphi}(\boldsymbol{x}_{k-1}), \boldsymbol{Q}_{k-1}), \\ \boldsymbol{A}_{k-1}, \boldsymbol{Q}_{k-1} &\sim \mathcal{MNIW}(\boldsymbol{M}^*, \boldsymbol{\Sigma}^*, \boldsymbol{\Lambda}^*, \nu_{k|k-1}), \end{aligned} \tag{26}$$

where

$$\begin{aligned} \boldsymbol{M}^* &= \boldsymbol{\Psi}_{k|k-1}(\boldsymbol{\Sigma}_{k|k-1} + \boldsymbol{V})^{-1}, \\ \boldsymbol{\Sigma}^* &= (\boldsymbol{\Sigma}_{k|k-1} + \boldsymbol{V})^{-1}, \\ \boldsymbol{\Lambda}^* &= \boldsymbol{\Lambda}_0 + \boldsymbol{\Phi}_{k|k-1} - \boldsymbol{\Psi}_{k|k-1}(\boldsymbol{\Sigma}^*)^{-1}\boldsymbol{\Psi}_{k|k-1}^\top. \end{aligned} \tag{27}$$

The predictive distribution of an $\mathcal{MNIW}$ distribution is a matrix-variate Student-t ($\mathcal{MT}$) distribution [9, 38],

$$\begin{aligned} \mathcal{MT}(\boldsymbol{A}|\nu, \boldsymbol{M}, \boldsymbol{\Lambda}, \boldsymbol{\Sigma}) &= \frac{\Gamma_M(\frac{\nu+n_x+M-1}{2})}{(2\nu\pi)^{Mn_x/2}\Gamma_M(\frac{\nu+n_x-1}{2})} \\ &\cdot \frac{|\boldsymbol{I}_M + \boldsymbol{\Lambda}^{-1}(\boldsymbol{A}-\boldsymbol{M})\boldsymbol{\Sigma}^{-1}(\boldsymbol{A}-\boldsymbol{M})^\top|^{-\frac{\nu+n_x+M-1}{2}}}{|\boldsymbol{\Lambda}|^{-M/2}|\boldsymbol{\Sigma}|^{-n_x/2}}. \end{aligned} \tag{28}$$

Hence, using the hierarchical structure (26), we can generate states $\boldsymbol{x}_k^i$ by first sampling $\boldsymbol{A}_{k-1}^i \in \mathbb{R}^{n_x \times M}$ as

$$\boldsymbol{A}_{k-1}^i \sim \mathcal{MT}(\nu_{k|k-1}^i, \boldsymbol{M}^*, \boldsymbol{\Lambda}^*, \boldsymbol{\Sigma}^*), \tag{29}$$

and then obtain $\boldsymbol{x}_k^i$ as $\boldsymbol{x}_k^i = \boldsymbol{A}_{k-1}^i \boldsymbol{\varphi}(\boldsymbol{x}_{k-1}^i)$. Theorem 3 states that instead of sampling $\boldsymbol{A} \in \mathbb{R}^{n_x \times M}$ from an $\mathcal{MT}$ distribution, we can generate $\boldsymbol{x}_k$ by sampling a vector of dimension $n_x \times 1$ from a multivariate Student-t ($\mathcal{T}$) distribution, which is computationally simpler than sampling a matrix.

Theorem 3 *Assume that the integrand of* (25) *can be written on the form* (26)*. Then, the predictive density* (23) *can be written as*

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k-1}) = \mathcal{T}(\nu_{k|k-1} - n_x + 1, \bar{\boldsymbol{M}}, \bar{\boldsymbol{\Lambda}}), \tag{30}$$

where $\bar{\boldsymbol{M}} = \boldsymbol{\Psi}_{k|k-1}(\boldsymbol{\Sigma}_{k|k-1} + \boldsymbol{V})^{-1}\boldsymbol{\varphi}(\boldsymbol{x}_{k-1})$ *and*

$$\bar{\boldsymbol{\Lambda}} = \operatorname{chol}(\boldsymbol{\Lambda}^*)$$
$$\cdot \operatorname{Tr}\big(\operatorname{chol}(\boldsymbol{\Sigma}^*)\boldsymbol{\varphi}(\boldsymbol{x}_{k-1})\boldsymbol{\varphi}(\boldsymbol{x}_{k-1})^\top \operatorname{chol}(\boldsymbol{\Sigma}^*)^\top\big)\operatorname{chol}(\boldsymbol{\Lambda}^*)^\top,$$

and where $\boldsymbol{M}^*$*,* $\boldsymbol{\Sigma}^*$*, and* $\boldsymbol{\Lambda}^*$ *are defined as in* (27)*.*

PROOF. See Appendix.

In the implementation, we generate samples $\bar{\boldsymbol{w}}_{k-1}^i \sim \mathcal{T}(\nu_{k|k-1} - n_x + 1, \boldsymbol{0}, \bar{\boldsymbol{\Lambda}}^i)$ and determine $\boldsymbol{x}_k^i$ from (9) using $\boldsymbol{A}^i = \bar{\boldsymbol{M}}^i$ according to $\boldsymbol{x}_k^i = \bar{\boldsymbol{M}}\boldsymbol{\varphi}(\boldsymbol{x}_{k-1}^i) + \bar{\boldsymbol{w}}_{k-1}$. From the lemma on transformations of variables in densities applied to (9) [30],

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k-1}) = p(\boldsymbol{w}_{k-1}|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{0:k-1}),$$

this generates samples consistent with (30).

Remark 8 *The generation of samples* (30) *is done based on the assumption of unknown noise covariance. If it is known, the sampling is instead done according to a Gaussian distribution.*

4.3 Determining the GP Hyperparameters

In this section we discuss how to adjust the GP hyperparameters. For simplicity, we assume one set of range and hyperparameters for all dimensions. As discussed in Remark 3, determining online all parameters associated with the GP is prohibitive. Sometimes the parameters can be determined from expert knowledge; for instance, σ can be chosen such that the initial uncertainty covers the possible range of uncertain functions, L can be chosen based on knowledge of the possible values of $\boldsymbol{x}$, and ℓ can be determined based on prior knowledge of the shape of the nonlinearity to be learned. We apply this reasoning to a real-world example in Sec. 6.

If there is no expert knowledge but there are data available a priori, it is possible to set the parameters based on offline learning methods—for example, it is possible to run the fully Bayesian PMCMC method in [35], which uses the same basis function expansion, to initialize the parameters. One approach is then to collect data, run the method in [35], and then execute the proposed online method based on the offline estimated parameters. Another possibility is to occasionally update the parameters online using a data batch, and, for example, execute the method in [35] or a maximum aposteriori method over the data batch and then reinitialize the estimator. While fully recursive adaptation of the basis-function range L may be infeasible, it is possible to adapt the GP hyperparameters $\boldsymbol{\vartheta}$ (i.e., the length ℓ and the scale σ) online when L has been set. Intuitively, the GP hyperparameters adjust to the prior choice L, as long as is covers the feasible space.

Let the GP hyperparameters evolve according to a random walk and assume independence between the different dimensions,

$$\boldsymbol{\vartheta}_k = \boldsymbol{\vartheta}_{k-1} + w_{\vartheta,k-1}, \tag{31}$$

where $w_{\vartheta,k} \sim \mathcal{N}(0, \boldsymbol{Q}_\vartheta)$, $\boldsymbol{Q}_\vartheta$ is a diagonal matrix. Instead of having $\boldsymbol{V}$ in (10) fixed, we now let it be updated according to the sampled GP hyperparameters. The additional steps involved when including GP hyperparameter estimation amount to; (i) sample $\{\boldsymbol{\vartheta}^i\}_{i=1}^N$ from the prior (31); (ii) update the $\mathcal{MN}$ left covariance prior $\{\boldsymbol{V}_k^i\}_{i=1}^N$ in (10) according to (8). The updated $\{\boldsymbol{V}_k^i\}_{i=1}^N$ affect the update of $\boldsymbol{\theta}$ by Theorem 2 and the generation of states by (29). The additional steps for the GP hyperparameter updates imply an estimation of the joint posterior density $p(\boldsymbol{x}_{0:k+1}, \boldsymbol{\vartheta}_{0:k}, \theta_k|\boldsymbol{y}_{0:k})$. However, in the end we are interested in the marginal densities $p(\boldsymbol{x}_k|\boldsymbol{y}_{0:k})$ and $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{0:k})$. The standard PF implementation marginalizes out $\boldsymbol{x}_{0:k-1}$ by discarding it, which leads to an $\mathcal{O}(N)$ implementation. This approximation relies on sufficient mixing properties in the dynamic model to avoid the depletion problem, essentially meaning that errors in the state are forgotten as time progresses, which is the reason why the $\mathcal{O}(N)$ implementation of the PF works in many realistic scenarios. In addition, the use of exponential forgetting by introducing λ helps in this regard [27]. However, when adapting the GP hyperparameters the model used to generate the state samples will differ at each time step, which increases the risk of particle depletion. Especially in high signal-to-noise ratios, it may be important to account for different paths, which leads to the modified weight update

$$q_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) \sum_{j=1}^{N} q_{k-1}^j p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^j, \boldsymbol{A}_{k-1}^j, \boldsymbol{Q}_{k-1}^j). \quad (32)$$

Eq. (32) results in an $\mathcal{O}(N^2)$ method, but it is possible to implement (32) with $\mathcal{O}(N \log N)$ complexity and even linearly (e.g., using accept-reject sampling [18]).

4.4 Algorithm Summary

Algorithm 1 summarizes the proposed method, where we have introduced $\Xi = \{\boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{\Sigma}, \boldsymbol{\Lambda}, \nu\}$. In the proposed method, each particle i retains its own set of Ξ^i.

Algorithm 1 Pseudo-code of proposed algorithm

```
    Initialize: Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$, $\{q_{-1}^i\}_{i=1}^N = 1/N$,
    $\Xi_{i=1}^N = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \boldsymbol{\Lambda}_0, \nu_0\}$, $\{\boldsymbol{\vartheta}^i\}_{i=1}^N = \{\sigma_0^i, \ell_0^i\}_{i=1}^N$.
 1: for $k = 0, 1, \ldots$ do
 2:     for $i \in \{1, \ldots, N\}$ do
 3:         if GP hyperparameter estimation then
 4:             Update weight $\bar{q}_k^i$ using (24) or (32).
 5:         else
 6:             Update weight $\bar{q}_k^i$ using (24).
 7:         end if
 8:         Determine $\Xi_{k|k-1}^i$ using (19).
 9:     end for
10:     Normalize weights as $q_k^i = \bar{q}_k^i / (\sum_{i=1}^N \bar{q}_k^i)$.
11:     Compute $N_{\text{eff}} = 1/(\sum_{i=1}^N (q_k^i)^2)$.
12:     if $N_{\text{eff}} \leq N_{\text{thr}}$ then
13:         Resample particles and copy the corresponding statistics. Set $\{q_k^i\}_{i=1}^N = 1/N$.
14:     end if
15:     for $i \in \{1, \ldots, N\}$ do
16:         if GP hyperparameter estimation then
17:             Sample $\boldsymbol{\vartheta}_k$ from (31).
18:             Update $\boldsymbol{V}_k^i$.
19:         end if
20:         Sample $\boldsymbol{x}_{k+1}^i$ from (30).
21:         Determine $\Xi_{k|k}^i$ using (18).
22:     end for
23: end for
```

5 Numerical Evaluation

In this section we consider the system

$$\begin{aligned} x_{k+1} &= \tanh(2x_k) + w_k, & w_k &\sim \mathcal{N}(0, 0.1), & \text{(33a)} \\ y_k &= x_k + e_k, & e_k &\sim \mathcal{N}(0, 0.1), & \text{(33b)} \end{aligned}$$

where the objective is to learn $f(x_k) = \tanh(2x_k)$ and $Q_w = 0.1$, as well as estimating x_k. We initialize with $\nu_0 = 1$, $\Lambda_0 = 2$, (i.e., with prior $Q_w \sim \mathcal{IW}(2, 1)$). We set $L = 4$ in (4) and $M = 16$ basis functions, and the forgetting factor to $\lambda = 0.97$. We compare with the fully Bayesian PMCMC method reported in [35], which our method extends to the online case. The method uses a particle Gibbs Markov kernel for finding the state smoothing distribution within an MCMC procedure based on the whole sequence of measurements. We perform a Monte-Carlo study and compare the state root mean-square error (RMSE) of Algorithm 1 with and without GP hyperparameter estimation. We initialize $\sigma_0 = 1$, $\ell_0 = 0.1$ in (8). Note that these values are kept fixed for Algorithm 1 without GP hyperparameter estimation. For Algorithm 1 with GP hyperparameter estimation, we set $Q_\vartheta = \text{diag}(1, 0.3)$.

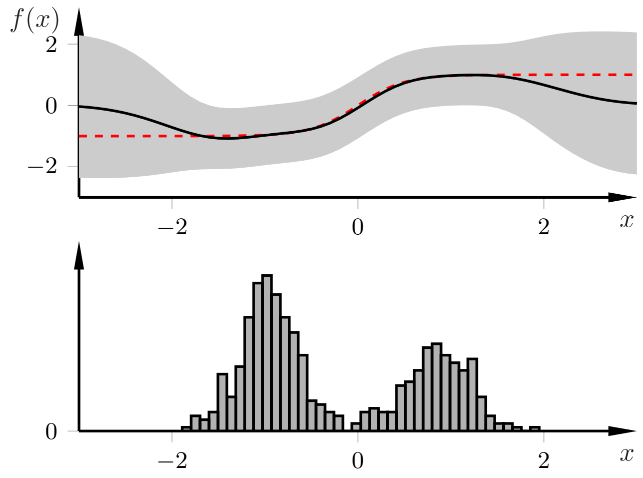

Fig. 1. The posterior estimates of the Bayesian PMCMC method (see [35]) for $T = 500$ data points and 5000 MCMC iterations for the example in Sec. 5. The true function is in dashed red, the estimate in black, and the Bayesian learning, the 3σ confidence is shown in gray. The bars in the lowest plot show the distribution of data in the state space.

The posterior estimates for the method in [35] for $T = 500$ data points and $K = 5000$ MCMC iterations are shown in Fig. 1. Fig. 2 displays the results for Algorithm 1 using GP hyperparameter estimation for $k = 50$, $k = 300$, and $k = 500$ time steps, respectively. Algorithm 1 provides quite similar results as in Fig. 1 as the number of data points increases. The role of the forgetting factor, which increases uncertainty in regions that havent been explored recently, can be seen when comparing the second and third plots for $x \approx 0$.

Fig. 3 shows the GP hyperparameter and covariance estimates averaged over 500 Monte-Carlo runs. As time progresses, the ℓ and Q_w estimates using Algorithm 1 converge very close to the estimates of the PMCMC method, but the σ estimate is slightly biased. Note that the two methods are distinctly different. Algorithm 1 uses the method outlined in Sec. 4.3 whereas [35] employs a Metropolis-Hastings step within the MCMC procedure.

Fig. 4 compares the state RMSE values of Algorithm 1 with and without GP hyperparameter estimation for

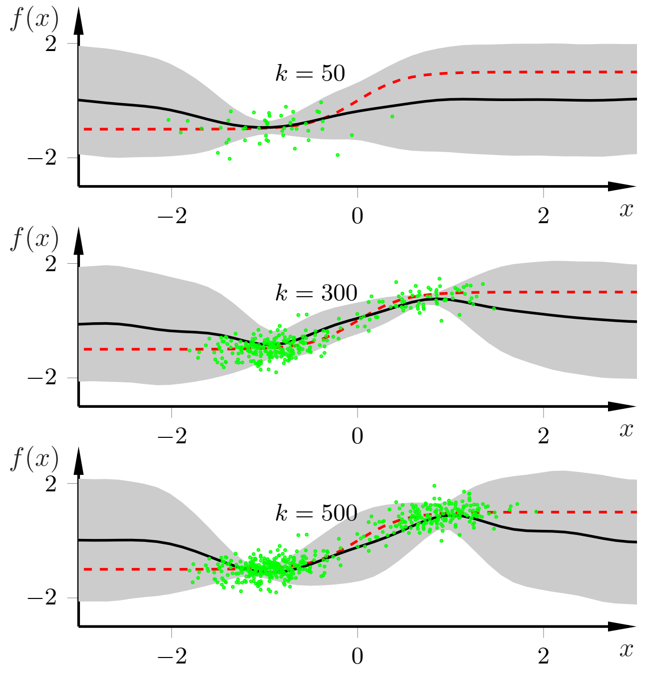

Fig. 2. The posterior estimates of Algorithm 1, for the example in Sec. 5. True function in dashed red, estimated function in black, and 3σ confidence in gray. The green dots are the measurements until time step k.

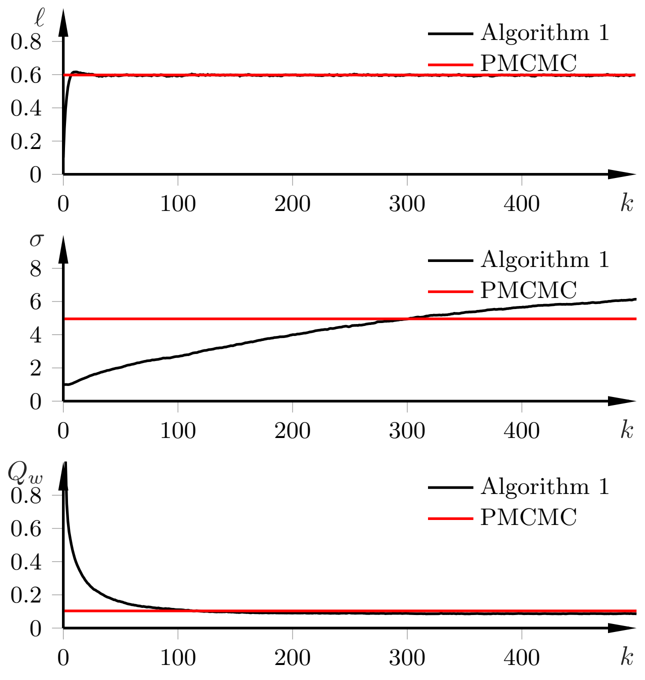

Fig. 3. The GP hyperparameter estimates for the example in Sec. 5 averaged over 500 Monte-Carlo runs. The PMCMC method estimates are taken as the average of the realizations of the MCMC iterations after the burn-in period.

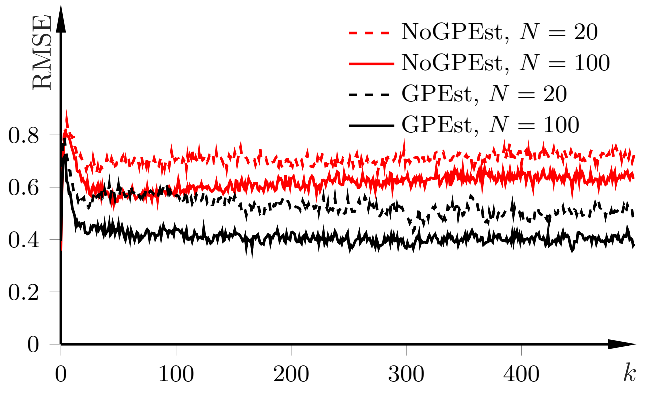

Fig. 4. Comparison of state RMSE for 500 Monte-Carlo runs with and without GP hyperparameter estimation.

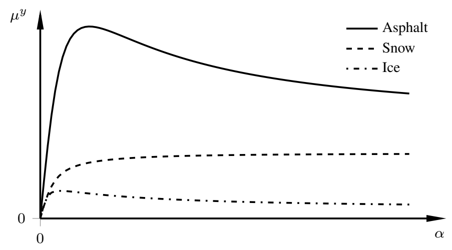

Fig. 5. Illustrations of lateral tire friction μ^y as a function of slip angle α for surfaces corresponding to asphalt, loose snow, and ice.

$N = 20$ and $N = 100$ particles. Clearly, the adjustment of the GP hyperparameters increases accuracy.

6 Application to Tire-Friction Estimation

In this section, we apply Algorithm 1 to real-time estimation of the friction between the tire of a vehicle and the road using both synthetic and real data. The friction dependence between tire and road is highly nonlinear and varies heavily between different surfaces, see Fig. 5 for an illustration. Knowledge of the tire friction is important for real-time vehicle control [7]. A difficulty when addressing the tire-friction estimation problem using automotive-grade sensors is that the sensors are low grade, only provide indirect measurements of the friction, and do not even measure the vehicle state, which is nonlinearly dependent on the tire friction and must therefore be known or estimated for learning the tire friction.

6.1 Modeling

The vehicle model relating the vehicle state to the tire friction is the well-known *single-track model*,

$$\dot{v}^Y + v^X\dot{\psi} = \frac{1}{m}(F_f^z\mu_f^y\cos(\delta) + F_r^z\mu_r^y + F_f^z\mu_f^x\sin(\delta)), \tag{34a}$$

$$I_{zz}\ddot{\psi} = l_fF_f^z\mu_f^y\cos(\delta) - l_rF_r^z\mu_r^y + l_fF_f^z\mu_f^x\sin(\delta), \tag{34b}$$

where μ^y is the lateral tire-friction function and the subscripts f, r stand for front and rear, respectively, m is the vehicle mass, I_{zz} is the vehicle inertia about the vertical axis, and δ is the front-wheel steering angle. By denoting the wheel base with $l = l_f + l_r$, the normal force F^z resting on each front/rear wheel is $F_f^z = mgl_r/l$, $F_r^z = mgl_f/l$. The tire-friction components μ_i^y, $i \in \{f, r\}$ are modeled as static functions of the slip quantities,

$$\mu_i^y = f_i^y(\alpha_i(\boldsymbol{x})),\ i \in \{f, r\}, \tag{35}$$

α_i is the slip angle, $\alpha_i = -\arctan(v_{y,i}/v_{x,i})$, where $v_{x,i}$ and $v_{y,i}$ are the longitudinal and lateral wheel velocities for wheel i with respect to an inertial system, expressed in the coordinate system of the wheel. The wheel velocities can be computed from a transformation of the longitudinal and lateral vehicle velocities. The lateral velocity is estimated in the proposed method, whereas the longitudinal velocity v^X is determined from the measured wheel-speeds $\{\omega_i\}_{i=1}^4$. For brevity, we define the vector $\boldsymbol{\alpha} = [\alpha_f\ \alpha_r]^{\mathrm{T}}$ and write (35) as $\boldsymbol{\mu} = [f_f^y\ f_r^y]^\top$, and model the friction vector as a realization from

$$\boldsymbol{\mu} \sim \mathcal{GP}(\mathbf{0}, \kappa(\boldsymbol{\alpha}, \boldsymbol{\alpha}')). \tag{36}$$

Discretizing the system with sampling period T_s and using $\boldsymbol{u} = [\delta\ v^X]^{\mathrm{T}}$ as the known input vector, the vehicle model (34)–(36) can be written as $\boldsymbol{x}_{k+1} = \boldsymbol{a}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{G}(\boldsymbol{x}_k, \boldsymbol{u}_k)\boldsymbol{\mu}(\boldsymbol{\alpha}_k)$, where $\boldsymbol{a}(\cdot)$ and $\boldsymbol{G}(\cdot)$ are the (known) parts of the vehicle model, and $\boldsymbol{\mu}(\cdot)$ is the unknown tire-friction function. After a coordinate transformation, the model is in accordance with (9), where we model the tire friction $\boldsymbol{\mu}$ using the reduced-rank formulation.

The measurement model is based on a setup commonly available in production cars, namely the lateral acceleration a_m^Y and the yaw rate $\dot{\psi}_m$, forming the measurement vector $\boldsymbol{y} = [a_m^Y\ \dot{\psi}_m]^\top$. We model the measurement noise $\boldsymbol{e}_k$ as zero-mean Gaussian distributed noise with known covariance $\boldsymbol{R}$ according to $\boldsymbol{e}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$, and the resulting measurement model is on the form (1b).

6.2 Prelimiaries

We use 10 basis functions each for the front and rear tire. From expert knowledge we know that (i) the friction between the front and rear wheel can be estimated independently, (ii) the tire friction is antisymmetric, which implies that we only use odd basis functions—that is, $j = 2, 4, \ldots$ in (3). This gives $M = 10$ basis functions in total, whereas without using the expert knowledge $M = 100$ basis functions would have been needed. The sampling period is $T_s = 40$ms. The number of particles is $N = 200$. In (8), we set $s_f = 50$, which allows the initial uncertainty to cover the different ranges of possible friction values. We set $L = 30\pi/180$, since α usually is restricted to $-15 \lessapprox \alpha \lessapprox 15$ deg, and $\ell = 2\pi/180$, to align with the shape of a typical friction curve. We set the estimator to use roughly the last second of measurements for learning, which gives a forgetting factor of $\lambda = 0.95$. This forgetting factor is a trade off between being able to quickly estimate sudden surface changes and not being overly sensitive to bad data points due to the large noise in the automotive-grade sensors.

6.3 Simulation Results

For the simulation results, we simulate a vehicle executing a number of step-steer maneuvers at constant velocity. The vehicle parameters are that of a mid-size SUV and we use the well-known Pacejka tire model [28] to model the tire forces (c.f. Fig. 5),

$$\mu_i = \bar{\mu}_i \sin(C_i \arctan(B_i(1 - E_i)\alpha_i + E_i \arctan(B_i\alpha_i))), \tag{37}$$

for $i \in \{f, r\}$, where $\bar{\mu}$, B, C, and E are the peak, stiffness, shape, and curvature factor, respectively. The simulation lasts for 25s and starts off on dry asphalt. After 14.5s, there is an abrupt change in surface from dry asphalt to snow.

Fig. 6 shows eight snapshots of the estimates during different time steps for the rear tire. In the first two rows, the vehicle drives on dry asphalt and the estimates are very close to the true friction function. At $t = 14.5$s, there is an abrupt change in surface from dry asphalt to snow. The estimator reacts to the new measurements that indicate a surface shift, and again converges close to the true friction function.

6.4 Experiments

For the experimental study, we have used a mid-size SUV to gather data and collected several data sets using the same vehicle on a snow-covered track, all data sets roughly 250s long, and the maneuvers are such that parts of the nonlinear region of the tire-force curve is excited at certain times. The data contains both variation in velocities and periods of straight driving and cornering. The parameters of the vehicle model have been extracted from data sheets and bench testing.

We do not have access to ground truth of the real tire-friction function. Instead, as a performance indication, we use the offline Bayesian learning PMCMC method in [35], also evaluated against in Sec. 5, to get a high-performance estimate. This method is the offline version of the method proposed in this paper and comes with strong convergence guarantees, and is therefore a good benchmark for our method.

6.4.1 Results

Fig. 7 shows the experimental results of our method (black solid) for $t = 20, 30, 45$s, respectively, for the front

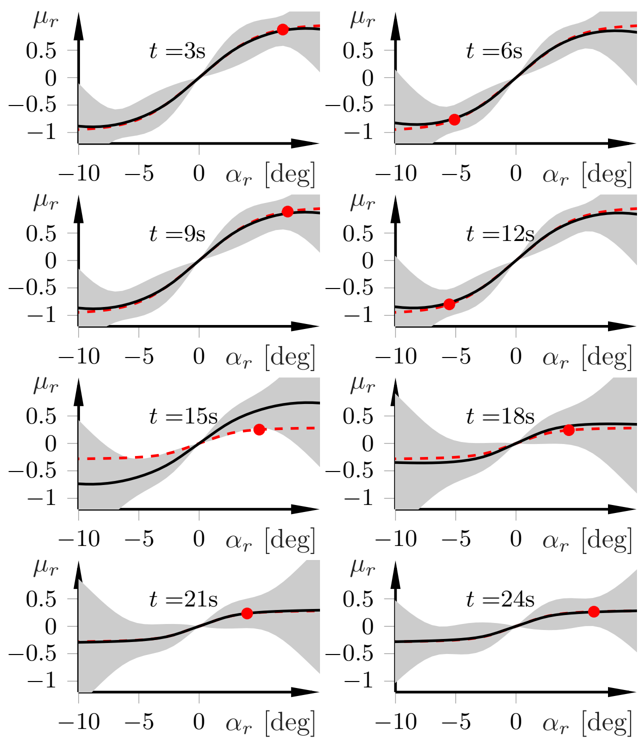

Fig. 6. The posterior estimates (black solid), estimated 3σ (gray) of our proposed method for the rear tire, true friction curve in red dashed, and current α_r value as the red circle. Knowledge of α_r is not used in learning, it is estimated online. Surface change from dry asphalt is at $t = 14.5$s.

(first row) and rear (second row) tire. The third row in Fig. 7 shows the excitation level of the underlying data for the range of slip angles of the front wheel. The slip angles shown are computed using the state estimates from the PF. We stress that the data in the lowest row are a result from Algorithm 1. For learning, we only employ the onboard automotive-grade wheel-speed sensors for computing the forward velocity, the steering angle, the yaw rate, and lateral acceleration measurements. Due to the zero-mean prior of the function coefficients in (6), the estimates do not suffer from overfitting issues outside of the available data range. Instead, they smoothly converge to the zero-mean prior.

The results for the PMCMC approach are shown in red dotted. Note that the results for the PMCMC approach have been genererated using the whole data set (250s). As data are acquired, the results of Algorithm 1 become increasingly similar to those of the PMCMC approach. The fit between the estimated Pacejka tire models for each estimator is also good for the region where data have been collected, which is an indication of how the uncertainty estimate from the method can be useful in determining how to trust the available estimates. For instance, at $t = 20$ and $t = 30$s, mostly straight driving has occured and there are few data points exciting anything but the region close to the origin. As a result, the estimates for Algorithm 1 are only accurate around the origin. At $t = 45$ s, when data have been gathered for a larger region, the estimates of our approach very closely resemble those of the PMCMC approach. Hence, the method is able to produce similar results as the substantially more computation-heavy PMCMC approach.

To validate further the accuracy of the learned tire-friction functions, Fig. 8 shows the measured yaw rate (red dashed, upper) and lateral acceleration (red dashed, lower), together with the predicted quantities (black solid) when simulating the system using the estimated tire-friction function for a portion of one data set (different from the data set used for learning). Note that it is a pure simulation of the vehicle model that has been used to generate the trajectories, with the average estimate of the tire friction (i.e., the black solid lines in the two upper left-most plots in Fig. 7) to predict the forces. The resulting tire models give accurate prediction capabilities when comparing to the measured quantities, which is an indication that the tire-friction estimates using the proposed method give valid results.

7 Conclusions

This work builds upon recent work on learning and parameter estimation for systems that can be described by state-space models. This paper focused on a fully Bayesian approach incorporating GPs into a state-space formulation for online inference and learning. Our work relies on the PF framework, and we extended the use case of the PF to estimate online both complex functions describing the dynamics, as well as the state. The proposed algorithm relies on a reduced-rank formulation of GP-SSMs, together with marginalization and conjugate priors. The algorithm is applicable to general nonlinear systems, and a comparison with a state-of-the-art offline learning approach indicates that the online method can, at least for certain problems, give similar performance. The experimental case study involving the difficult problem of tire-friction estimation shows that the method can indeed be used for real-world applications.

In learning involving GPs, tuning the hyperparameters can turn out to be a difficult problem by itself. While estimating all involved tuning parameters online is intractable, we presented an extension of our method to tune the GP hyperparameters, which reduces the need for prior tuning and improves estimation accuracy. However, inevitably, some tuning is necessary, and it is future research how to systematically introduce estimation of, for example, the forgetting factor in the PF framework.

Another question is how to integrate the online learning with its uncertainty estimates into safe control strategies, where the estimated distribution is used to assess safety in a systematic manner. We have started to explore this related to adaptive model-predictive control (MPC, [7]), and the potential to combine with recent

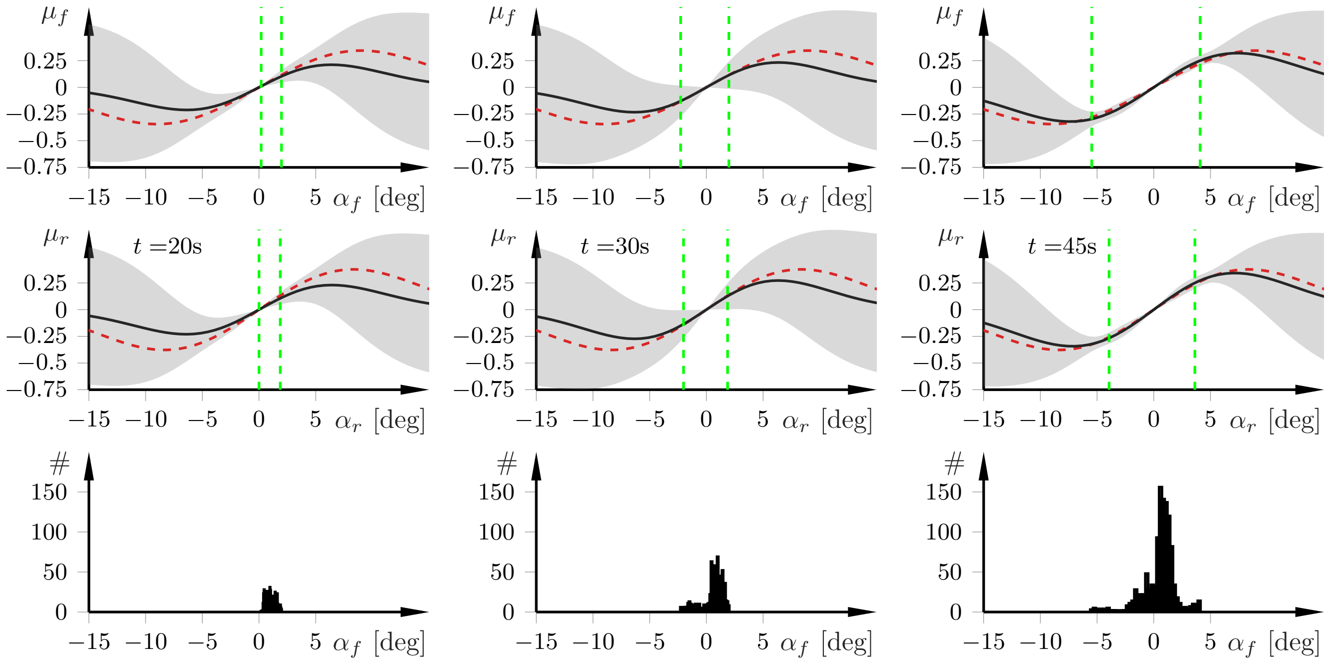

Fig. 7. The posterior estimates (black solid) and estimated 3σ confidence (gray) of our proposed method after $t = 20$s, $t = 30$s, and $t = 45$s (left to right), for the tire-friction experiments. The offline Bayesian PMCMC learning method in [35] is shown in red dashed and the lowest row shows the excitation level for the (estimated) front slip angle up until the different time instants. The green vertical lines indicate the excitation level and correspond to the boundaries of the excitation range.

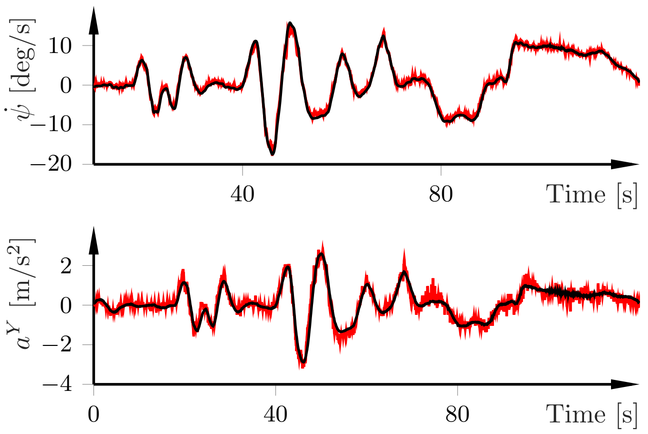

Fig. 8. Predicted (black solid) and measured (red dashed) yaw rate and lateral acceleration using the estimated tire-friction function).

developments in stochastic MPC is a potentially fruitful research direction [12].

References

[1] Mauricio A Alvarez, David Luengo, and Neil D Lawrence. Linear latent force models using Gaussian processes. *IEEE Trans Pattern Anal Mach Intell*, 35(11):2693–2705, 2013.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *J. Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] Karl Johan Åström and Björn Wittenmark. *Adaptive Control (2 rev. Dover ed.)*. Dover Publications, 2008.

[4] K. Berntorp. Bayesian inference and learning of Gaussian-process state-space models. In *Eur. Control Conf.*, Naples, Italy, June 2019.

[5] Karl Berntorp. Online Bayesian tire-friction learning by Gaussian-process state-space models. In *IFAC World Congress*, Berlin, Germany, July 2020.

[6] Karl Berntorp and Stefano Di Cairano. Tire-stiffness and vehicle-state estimation based on noise-adaptive particle filtering. *IEEE Trans. Control Syst. Technol.*, 27(3):1100–1114, 2018.

[7] Karl Berntorp, Rien Quirynen, Tominaga Uno, and Stefano Di Cairano. Trajectory tracking for autonomous vehicles on varying road surfaces by friction-adaptive nonlinear model predictive control. *Veh. Syst. Dyn.*, 58(5):705–725, 2019.

[8] Hildo Bijl, Thomas B. Schön, Jan-Willem van Wingerden, and Michel Verhaegen. System identification through online sparse Gaussian process regression with input noise. *IFAC Journal of Systems and Control"*, 2:1–11, 2017.

[9] A Philip Dawid. Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, 68(1):265–274, 1981.

[10] Petar M Djurić and Joaquin Miguez. Sequential particle filtering in the presence of additive Gaussian noise with unknown parameters. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, May 2002.

[11] Arnaud Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *Handbook of Nonlinear Filtering*. Oxford University Press, 2009.

[12] Xuhui Feng, Stefano Di Cairano, and Rien Quirynen. Inexact adjoint-based SQP algorithm for real-time stochastic nonlinear MPC. In *IFAC World Congress*, Berlin, Germany, July 2020.

[13] Roger Frigola, Yutian Chen, and Carl Edward Rasmussen. Variational Gaussian process state-space models. In *Advances in Neural Information Processing Systems*, Montreal, Canada, December 2014.

[14] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Adv. Neural Information Processing Systems*, Lake Tahoe, NV, December 2013.

[15] Roger Frigola-Alcade. *Bayesian time series learning with Gaussian processes.* phdthesis, Univ. Cambridge, 2015.

[16] Zoubin Ghahramani and Sam T. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems 11*, pages 431–437. 1998.

[17] A.K. Gupta and D.K. Nagar. *Matrix Variate Distributions.* Taylor & Francis, 1999.

[18] Fredrik Gustafsson. On marginal particle filters with linear complexity. In *Computational Adv. Multi-Sensor Adaptive Process.*, 2013.

[19] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[20] Lennart Ljung. *System identification: theory for the user.* Prentice-hall, second edition, 1999.

[21] Benn Macdonald, Catherine Higham, and Dirk Husmeier. Controversy in mechanistic modelling with Gaussian processes. In *Int. Conf. Machine Learning*, Lille, France, July 2015.

[22] César Lincoln C Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A Barreto, and Neil D Lawrence. Recurrent Gaussian processes. *arXiv preprint arXiv:1511.06644*, 2015.

[23] Andrew McHutchon. *Nonlinear modelling and control using Gaussian processes.* phdthesis, Univ. Cambridge, 2014.

[24] R.K. Mehra. On the identification of variances and adaptive Kalman filtering. *IEEE Trans. Autom. Control*, 15(2):175–184, 1970.

[25] Kevin P. Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, UBC, 2007.

[26] Christopher Nemeth, Paul Fearnhead, and Lyudmila Mihaylova. Sequential Monte Carlo methods for state and parameter estimation in abruptly changing environments. *IEEE Trans. Signal Process.*, 62(5):1245–1255, 2013.

[27] Emre Özkan, Václav Šmídl, Saikat Saha, Christian Lundquist, and Fredrik Gustafsson. Marginalized adaptive particle filtering for nonlinear models with unknown time-varying noise parameters. *Automatica*, 49(6):1566–1575, 2013.

[28] Hans B. Pacejka. *Tire and Vehicle Dynamics.* Butterworth-Heinemann, Oxford, United Kingdom, 2nd edition edition, 2006.

[29] Gianluigi Pillonetto and Giuseppe De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, 2010.

[30] C. Radhakrishna Rao. *Linear Statistical Inference and its Applications.* Wiley, 2001.

[31] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, MA, USA, 2006.

[32] S. Saha and F. Gustafsson. Particle filtering with dependent noise processes. *IEEE Trans. Signal Process.*, 60(9):4497–4508, 2012.

[33] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

[34] Geir Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on signal Processing*, 50(2):281–289, 2002.

[35] Andreas Svensson and Thomas B. Schön. A flexible state-space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.

[36] Felipe Tobar, Petar M Djurić, and Danilo P Mandic. Unsupervised state-space modeling using reproducing kernels. *IEEE Transactions on Signal Processing*, 63(19):5210–5221, 2015.

[37] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Trans Pattern Anal Mach Intell*, 30(2):283–298, 2008.

[38] Shenghuo Zhu, Kai Yu, and Yihong Gong. Predictive matrix-variate t models. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, December 2008.

8 Appendix

Proof sketch of Theorem 2

For convenience, we will work with logarithms of the considered expressions. Consequently, by taking the logarithm of (16), after some manipulations we get that

$$\begin{aligned}\log\left(p(\boldsymbol{x}_{k+1}|\boldsymbol{\theta}_k,\boldsymbol{x}_k)\right) &\propto -\frac{1}{2}\log\left(|\boldsymbol{Q}|\right)\\ &- \mathrm{Tr}\big(\boldsymbol{Q}^{-1}(\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^\top - \boldsymbol{A}\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{x}_{k+1}^\top\\ &- \boldsymbol{x}_{k+1}\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top\boldsymbol{A}^\top + \boldsymbol{A}\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top\boldsymbol{A}^\top)\big). \quad (38)\end{aligned}$$

Suppose that at time step k, it holds that

$$\begin{aligned}p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k}) &= \mathcal{MNIW}(\boldsymbol{\theta}_k|\boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1},\\ \boldsymbol{\Xi}^{-1}, \boldsymbol{\Lambda}_0 &+ \boldsymbol{\Phi}_{k|k-1} - \boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top, \nu_{k|k-1}). \quad (39)\end{aligned}$$

Taking the logarithm of (39) gives that

$$\begin{aligned}\log(p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k})) &\propto -\frac{1}{2}(n_x + M + 1 + \nu_{k|k-1})\log(|\boldsymbol{Q}|)\\ &- \frac{1}{2}\mathrm{Tr}\big(\boldsymbol{Q}^{-1}(\boldsymbol{\Lambda}_0 + \boldsymbol{\Phi}_{k|k-1} - \boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top)\big)\\ &- \frac{1}{2}\mathrm{Tr}\big((\boldsymbol{A} - \boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1})^\top\boldsymbol{Q}^{-1}(\boldsymbol{A}\boldsymbol{\Xi} - \boldsymbol{\Psi}_{k|k-1})\big).\\ &\qquad (40)\end{aligned}$$

Now, assume that $\boldsymbol{x}_{k+1}$ is obtained. From (38) and (40),

$$
\begin{aligned}
&\log(p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}))\\
&\quad\propto \log\left(p(\boldsymbol{x}_{k+1}|\boldsymbol{\theta}_k,\boldsymbol{x}_k)\right)+\log(p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k})\propto\\
&-\frac{1}{2}(n_x+M+1+\nu_{k|k-1}+1)\log(|\boldsymbol{Q}|)-\frac{1}{2}\mathrm{Tr}\big(\big(\boldsymbol{Q}^{-1}(\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^\top\\
&\quad-\boldsymbol{A}\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{x}_{k+1}^\top-\boldsymbol{x}_{k+1}\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top\boldsymbol{A}^\top\\
&\quad+\boldsymbol{A}\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top\boldsymbol{A}^\top)\big)\big)\\
&\quad-\frac{1}{2}\mathrm{Tr}\big(\boldsymbol{Q}^{-1}(\boldsymbol{\Lambda}_0+\boldsymbol{\Phi}_{k|k-1}-\boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top)\big)\\
&-\frac{1}{2}\mathrm{Tr}\big((\boldsymbol{A}\boldsymbol{\Xi}\boldsymbol{A}^\top-\boldsymbol{A}\boldsymbol{\Psi}^\top-\boldsymbol{\Psi}_{k|k-1}\boldsymbol{A}^\top+\boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top)\big).
\end{aligned} \tag{41}
$$

Collecting and identifying terms in (41) lead to

$$
\begin{aligned}
&\log(p(\boldsymbol{\theta}_k|\boldsymbol{x}_{0:k+1}))\propto-\frac{1}{2}(n_x+M+1+\underbrace{\nu_{k|k-1}+1}_{\nu_{k|k}})\log(|\boldsymbol{Q}|)\\
&-\frac{1}{2}\mathrm{Tr}\Big\{\boldsymbol{Q}^{-1}\big((\boldsymbol{\Lambda}_0+\underbrace{\boldsymbol{\Phi}_{k|k-1}+\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^\top}_{\boldsymbol{\Phi}_{k|k}}-\boldsymbol{\Psi}_{k|k-1}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top)\\
&\quad+\boldsymbol{A}\big(\underbrace{\boldsymbol{\Sigma}_{k|k-1}+\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top}_{\boldsymbol{\Sigma}_{k|k}}+\boldsymbol{V}^{-1}\big)\boldsymbol{A}^\top\\
&-\boldsymbol{A}(\underbrace{\boldsymbol{\Psi}_{k|k-1}^\top+\boldsymbol{\varphi}(\boldsymbol{x}_k)\boldsymbol{x}_{k+1}^\top}_{\boldsymbol{\Psi}_{k|k}^\top})-(\underbrace{\boldsymbol{\Psi}_{k|k-1}+\boldsymbol{x}_{k+1}\boldsymbol{\varphi}(\boldsymbol{x}_k)^\top}_{\boldsymbol{\Psi}_{k|k}})\boldsymbol{A}^\top\\
&\quad+\boldsymbol{\Psi}_{k|k}\boldsymbol{\Xi}^{-1}\boldsymbol{\Psi}_{k|k-1}^\top\big)\Big\},
\end{aligned} \tag{42}
$$

which, from (14), implies (17) and (18). □

Proof sketch of Theorem 3

For ease of notation, $\boldsymbol{\varphi}:=\boldsymbol{\varphi}(\boldsymbol{x}_{k-1})$. The predictive distribution of an $\mathcal{NIW}$ distribution is a $\mathcal{T}$ distribution [25]. Hence, the idea is to find an $\mathcal{NIW}$ expression of the integrand of (25) (i.e., (26)). That the mean of the integrand is $\bar{\boldsymbol{M}}$ follows trivially from (17) and (26). We utilize two well-known properties for the $\mathcal{MN}$ distribution [17]; (i), if $\boldsymbol{X}\sim\mathcal{MN}(\bar{\boldsymbol{M}},\boldsymbol{I},\boldsymbol{I})$, $\boldsymbol{Y}=\bar{\boldsymbol{M}}+\mathrm{chol}(\boldsymbol{\Lambda}^*)\boldsymbol{X}\mathrm{chol}(\boldsymbol{\Sigma}^*)\sim\mathcal{MN}(\bar{\boldsymbol{M}},\boldsymbol{\Lambda}^*,\boldsymbol{\Sigma}^*)$; (ii), if $\boldsymbol{X}\sim\mathcal{MN}(\boldsymbol{0},\boldsymbol{I},\boldsymbol{I})$, $\mathbb{E}(\boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^\top))=\mathrm{Tr}(\boldsymbol{A}^\top)$. From $\boldsymbol{x}_k=\boldsymbol{Y}\boldsymbol{\varphi}$,

$$\mathbb{E}\big((\boldsymbol{x}_k-\bar{\boldsymbol{M}}\boldsymbol{\varphi})(\boldsymbol{x}_k-\bar{\boldsymbol{M}}\boldsymbol{\varphi})^\top\big)=\mathbb{E}(\boldsymbol{x}_k\boldsymbol{x}_k^\top). \tag{43}$$

Next, we get that

$$
\begin{aligned}
&\mathbb{E}(\boldsymbol{x}_k\boldsymbol{x}_k^\top)\\
&=\mathrm{chol}(\boldsymbol{\Lambda}^*)\mathbb{E}\big(\boldsymbol{X}\mathrm{chol}(\boldsymbol{\Sigma}^*)\boldsymbol{\varphi}\boldsymbol{\varphi}^\top\mathrm{chol}(\boldsymbol{\Sigma}^*)^\top\boldsymbol{X}^\top\big)\mathrm{chol}(\boldsymbol{\Lambda}^*)^\top\\
&=\underbrace{\mathrm{chol}(\boldsymbol{\Lambda}^*)\mathrm{Tr}(\mathrm{chol}(\boldsymbol{\Sigma}^*)\boldsymbol{\varphi}\boldsymbol{\varphi}^\top\mathrm{chol}(\boldsymbol{\Sigma}^*)^\top)\mathrm{chol}(\boldsymbol{\Lambda}^*)^\top}_{\bar{\boldsymbol{\Lambda}}},
\end{aligned} \tag{44}
$$

where we used (i) for the first equality and (ii) for the second equality. Hence, the integrand of (25) can be written as

$$\mathcal{NIW}(\boldsymbol{x}_k|\bar{\boldsymbol{M}},\bar{\boldsymbol{\Lambda}},\nu_{k|k-1}), \tag{45}$$

from which (30) follows (see, e.g., [27]). □