Domain-aware Gaussian process state-space models

Mishra, Anurodh; Rajan, Raj Thilak

DOI
10.1016/j.sigpro.2025.110003

Publication date
2026

Document Version
Final published version

Published in
Signal Processing

Signal Processing 238 (2026) 110003

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Short communication

Domain-aware Gaussian process state-space models☆

Anurodh Mishra *, Raj Thilak Rajan

Signal Processing Systems, Faculty of EEMCS, Delft University of Technology, Mekelweg 4, Delft, 2628CD, The Netherlands

ARTICLE INFO

Keywords:
Gaussian processes
Non-parametric learning
Dynamical systems
State-space models

ABSTRACT

Gaussian process state-space models are a widely used modeling paradigm for learning and estimation in dynamical systems. Reduced-rank Gaussian process state-space models combine spectral characterization of dynamical systems with Hilbert space methods to enable learning, which scale linearly with the length of the time series. However, the current state of the art algorithms struggle to deal efficiently with the dimensionality of the state-space itself. In this work, we propose a novel algorithm, referred to as Domain-Aware reduced-rank Gaussian Process State-Space Model (DA-GPSSM), which exploits the relationship between state dimensions to model only necessary dynamics resulting in reduced computational cost, by potentially orders of magnitude in comparison to the state-of-the-art. The proposed approach grants modeling flexibility while maintaining comparable performance and thus increasing the applicability of these models. We present implications of the proposed approach and discuss applications where DA-GPSSM can be beneficial. Finally, we conduct simulations to demonstrate the performance and reduced computational cost of our proposed method, compared to the state-of-the-art learning method, and propose future research directions.

1. Introduction

Gaussian Process regression can be used to learn nonlinear state transition and measurement models using only measurement data. Previous works, on learning state transition and measurements models using Gaussian processes have shown to lead to a better performance as compared to traditional nonlinear filters such as extended Kalman filters (EKF) which require the true nonlinear model to be known in advance [1]. Gaussian process state-space models (GPSSMs) combine a Bayesian framework for representing dynamical systems using state-space models (SSMs) with the flexibility of modeling the nonlinearities present within the system using versatile covariance functions provided by Gaussian processes. The flexibility afforded by the use of GPs to model the state-transition function allows for the estimation of a wider scope of nonlinear state-transition functions by using appropriate covariance functions. Other approaches such as Recurrent Neural Networks (RNNs) have been used previously to model dynamical systems but require the latent state, akin to states of an SSM, to be deterministic. This is particularly unreliable in the presence of process noise. Furthermore, to add stochasticity to the hidden states in RNN, latent variables have been introduced. However, the proposed approaches are complicated and better suited to high-dimensional measurement sensors such as cameras. Alternatively, GPSSMs provide a fully probabilistic approach that not only allow for latent states to be stochastic, but also enable multiple sensor modalities to be fused in a coherent way, exploiting the uncertainty information available for each modality.

Learning in GPSSMs conventionally requires operations involving the inversion of the covariance matrix and thus scale poorly with the number of datapoints with time complexity $\mathcal{O}(T^3)$ where T is the length of the timeseries. Hence GPs and consequently GPSSMs suffer from the curse of dimensionality [2]. To address this challenge, two types of approaches are considered in literature: *input-based methods* [3–5] and *spectrum-based methods* [6–8]. In input-based methods, e.g. [3], a subset of the data is used to form an approximate covariance matrix, which reduces the computational complexity to $\mathcal{O}(N^2T)$ where $N < T$ is the length of the selected subset obtained using eigendecomposition. Principled methods to choose the data subset, not necessarily from the training set, have been proposed in [4,5]. In contrast, the spectrum-based methods approximate the covariance function using basis function expansion in the spectral domain. In particular, Reduced-Rank Gaussian Processes, based on the Hilbert space approximation of the covariance function proposed in [8], which allows for the full spectrum governing the dynamical process to be utilized for a given rank M.

☆ This work is partially funded by the European Commission Key Digital Technologies Joint Undertaking - Research and Innovation (HORIZON-KDT-JU-2023-2-RIA), under grant agreement No 101139996, the ShapeFuture project - "Shaping the Future of EU Electronic Components and Systems for Automotive Applications".

* Corresponding author.
E-mail addresses: a.mishra@tudelft.nl (A. Mishra), r.t.rajan@tudelft.nl (R.T. Rajan).

https://doi.org/10.1016/j.sigpro.2025.110003
Received 6 September 2024; Received in revised form 22 February 2025; Accepted 15 March 2025
Available online 10 May 2025

This reduces the computational complexity from cubic to linear in time. However, this approach scales poorly in computational complexity when the dimension of the state-space (D) is large i.e., $\mathcal{O}(M^{3D})$. The extension of the Hilbert space approach in [8] to state-space models is referred to as Reduced-Rank Gaussian Process State-Space Models (RR-GPSSMs) [9], and suffers from similar computational issues associated with the dimension of the state-space. In [10], the authors propose deep hidden layers, similar to deep neural networks, to limit the exponential increase in the number of parameters to be learnt. However, in absence of further analysis, the spectral interpretation of the original process using the Hilbert space approximation is lost in such an approach. A similar work on deep layers of GPSSM is presented in [11] based on Fourier random features [6]. However, for high-dimensional state-space, the number of spectral points required in this approach are significantly large [8]. Another approach is given in [12–14], where variational approximation of the covariance function is proposed based on inducing point methods in [5]. For scaling the inducing input based variational approach to large datasets, the authors in [13] propose a variational approximation that allows for the reparameterization trick from [15] to be used to efficiently optimize the evidence lower bound. However, the approach is still computationally expensive. Additionally, these approaches do not directly treat the dimensionality of the underlying state-space model. In addition, there have been other works recently, based on Gaussian processes and state-space models e.g., [16–18], for the learning of ordinary differential equation in a regression setting but they do not deal with the dimensionality of the state space.

In this work, we focus on RR-GPSSM which is based on the Hilbert space approximation of the covariance function proposed in [8], and implemented in [9] in the context of state-space models. In contrast to other approximations, the Hilbert space approach in [8] allows for the full spectrum governing the dynamical process to be utilized for a given rank M. This reduces the computational complexity from cubic to linear in time. However, the approach scales poorly when the dimension of the state-space, D, is large with computational complexity of $\mathcal{O}(M^{3D})$. This has been a particular bottleneck of RR-GPSSMs [9]. In this paper, we aim to reduce the computational load associated with Hilbert spaced based RR-GPSSMs, allowing for a wider range of high-dimensional state-space models to be learnt. The key contributions made in this work are briefly listed below:

1. We model the interactions between the state dimensions, using selection matrices based on the binary relationships between the dimensions, leading to reduced computational complexity of $\mathcal{O}(M^{3K})$, where $K \leq D$.
2. We propose Domain-Aware Gaussian Process State-Space Models (DA-GPSSMs), extending the work of [9], to learn the required parameters of the modified model.
3. We discuss the implications of the proposed modification in terms of modeling capability and computational gains and discuss scenarios where the proposed DA-GPSSM offers exponential computational gains.
4. We provide simulations to show the effectiveness of our model when compared to the implementation in [9].

The layout of the paper is as follows. In Section 2, we introduce RR-GPSSM model and briefly discuss the learning algorithm. In Section 3, we propose DA-GPSSM and discuss in detail the implications of the modeling assumptions used in the proposed approach. Simulations showcasing the performance of the proposed approach are provided in Section 4, and the results are compared to traditional as well as the state-of-the-art algorithms.

2. Gaussian process state-space models

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [2]. Given a dataset $\{\mathbf{X}, \mathbf{y}\}$, with input $\mathbf{X} \in \mathbb{R}^{N\times D}$ and output $\mathbf{y} \in \mathbb{R}^{N}$, the underlying process which generates this data, is modeled by defined by a mean function $\mu(\mathbf{x}) \in \mathbb{R}$ and a covariance matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$. Each element of the covariance matrix is determined by a scalar kernel function which is parameterized by the hyperparameters $\boldsymbol{\theta}$. For example, consider the squared-exponential kernel function given by $k(\mathbf{x}, \mathbf{x}') = \sigma_{\text{SE}}^2 \exp -\dfrac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\, l_{\text{SE}}^2}$, where the hyperparameters are given as $\boldsymbol{\theta} = \begin{bmatrix}\sigma_{\text{SE}}^2 & l_{\text{SE}}\end{bmatrix}^T$, with σ_{SE}^2 and l_{SE} being the kernel variance and lengthscale, respectively. Learning in GPs then amount to learning the hyperparameters $\boldsymbol{\theta}$, allowing for prediction at a new data point $\mathbf{x}_* \in \mathbb{R}^D$.

Let $\mathcal{D} \triangleq \{1, \dots, D\}$, then for $d \in \mathcal{D}$ a standard GPSSM is given by

$$f_d(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \mathbf{K}(\mathbf{x}, \mathbf{x}')), \tag{1a}$$

$$\mathbf{x}_0 \sim p(\mathbf{x}_0), \tag{1b}$$

$$\mathbf{x}_t \mid \mathbf{x}_{t-1} \sim \mathcal{N}(\mathbf{x}_t \mid \mathbf{f}(\mathbf{x}_{t-1}), \mathbf{Q}), \tag{1c}$$

$$\mathbf{y}_t \mid \mathbf{x}_t \sim p(\mathbf{y}_t \mid \mathbf{x}_t), \tag{1d}$$

where $\mathbf{x}_t \in \mathbb{R}^D$ is the state vector at time $t \in \mathbb{R}$ and $\mathbf{Q} \in \mathbb{R}^{D\times D}$ models the process noise of the dynamics. The state-transition function is denoted by the mapping $\mathbf{f} : \mathbb{R}^D \to \mathbb{R}^D$ and the output $\mathbf{y}_t \in \mathbb{R}^P$ follows a measurement model with a known probability distribution function (pdf) $p(\mathbf{y} \mid \mathbf{x}_t)$. In GPSSMs, the unknown vector-valued state transition function $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}), & \dots, & f_D(\mathbf{x})\end{bmatrix}^T$ is assumed to be modeled by an independent Gaussian process for each dimension as given in (1). For learning in GPSSMs, a number of methods have been proposed in literature [19,20]. However, these methods are computationally impractical for large datasets due to the inversion of the covariance matrix which has a computational complexity of $\mathcal{O}(T^3)$, where T is the length of the time-series.

We now describe RR-GPSSM, which aims to reduce the cubic dependence of computational complexity on time, using spectral methods. As proposed in [8], a valid covariance function of a Gaussian process admits an infinite dimensional basis function expansion, as laid out by the *Wiener–Khintchine* theorem. Based on the work in [9], for a scalar x, the infinite dimensional basis function expansion can be approximated by an M-rank truncation, i.e.

$$k(x_i, x_j) \approx \sum_{m=1}^{M} S_\theta(\lambda^{(m)})\, \phi^{(m)}(x_i)\, \phi^{(m)}(x_j), \tag{2}$$

where $\lambda^{(m)}$ and $\phi^{(m)}(x)$ are the eigenvalues and eigenfunctions corresponding to the covariance function $k(\cdot,\cdot)$. This allows for the *Kosambi–Karhunen–Loéve* (KKL) expansion of the function $f(x)$, given by

$$f(x) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x})) \leftrightarrow f(x) \approx \sum_{m=1}^{M} a^{(m)}\, \phi^{(m)}(x), \tag{3}$$

with $a^{(m)} \sim \mathcal{N}(0, S_\theta(\lambda^{(m)}))$. The GPSSM itself can be written as a linear state-space model with basis functions, i.e.,

$$x_{t+1} = \begin{bmatrix} a^{(1)}, & \dots, & a^{(M)}\end{bmatrix} \begin{bmatrix} \phi^{(1)}(x_t) \\ \vdots \\ \phi^{(M)}(x_t)\end{bmatrix} + \epsilon_t, \tag{4}$$

where ϵ_t is the zero-mean noise variable plaguing the system. For the multi-dimensional case of $\mathbf{x}_t \in \mathbb{R}^D$ [8], a truncation of order M_d for $d \in D$, is typically performed for each dimension and the iterant of the kernel expansion runs through all possible combinations of truncated eigenvalues and eigenfunctions for each dimensions, i.e.,

$$k(\mathbf{x}_i, \mathbf{x}_j) \approx \sum_{m_1,\dots,m_D=1}^{M^D} S_\theta(\lambda^{(m_1,\dots,m_D)})\, \phi^{(m_1,\dots,m_D)}(\mathbf{x}_i)\, \phi^{(m_1,\dots,m_D)}(\mathbf{x}_j), \tag{5}$$

where each iterant in the index $\{m_1, \dots, m_D\}$ varies from 1 to M_d, i.e. $m_i \in \{1, \dots, M_d\}$. The form of the eigenvalue and eigenfunction in the basis expansion depends on the choice of the covariance function.

For example, the squared exponential covariance function for the multidimensional case admits the following eigenfunctions and eigenvalues given by

$$\lambda^{(m_1,\dots,m_D)} = \sum_{d=1}^{D} \left(\frac{\pi\, m_d}{2\, L_d} \right)^2, \tag{6a}$$

$$\phi^{(m_1,\dots,m_D)}(\mathbf{x}) = \prod_{d=1}^{D} \frac{1}{\sqrt{L_d}} \sin\left(\frac{\pi\, m_d\, (x_d + L_d)}{2\, L_d} \right), \tag{6b}$$

where L_d gives the boundary conditions for each dimension d. Consequently, the summation in (5) involves $\overline{M} \triangleq \prod_{d=1}^{D} M_d$ terms combining the eigenvalues and eigenfunctions over all D dimensions. Similar to (4), the *KKL* expansion for the multi-dimensional case results in

$$\mathbf{x}_{t+1} = \underbrace{\begin{bmatrix} a_1^{(1)} & \dots & a_1^{(\overline{M})} \\ \vdots & \ddots & \vdots \\ a_D^{(1)} & \dots & a_D^{(\overline{M})} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \phi^{(1)}(\mathbf{x}_t) \\ \vdots \\ \phi^{(\overline{M})}(\mathbf{x}_t) \end{bmatrix}}_{\boldsymbol{\phi}(\mathbf{x}_t)} + \boldsymbol{\epsilon}_t \tag{7}$$

where $\mathbf{A} \in \mathbb{R}^{D \times \overline{M}}$, $\boldsymbol{\phi}(\mathbf{x}_t) \in \mathbb{R}^{\overline{M}}$ and $\mathbb{E}[\boldsymbol{\epsilon}_t\, \boldsymbol{\epsilon}_t^T] = \mathbf{Q}$ is the covariance matrix, akin to the process noise in standard state-space models. As detailed in [9], learning in (7) is carried out using a block Gibbs sampler for the state trajectory $\mathbf{x}_{0:T}$ and the parameters $\mathbf{A}$, $\mathbf{Q}$ and θ. Sampling of the state trajectory is then carried out using a conditional particle filter with ancestral sampling (CPF-AS [21]) as laid out in Algorithm 1. Please note that we use $\mathbf{x}_t^{(i)}$ to denote the state vector for the ith particle in Algorithm 1.

Algorithm 1 CPF-AS [21]

Input: Reference trajectory, $\bar{\mathbf{x}}_{0:T}$, prior $p(\mathbf{x}_0)$ and N particles.
Sample $\mathbf{x}_0^{(i)}$ with $p(\mathbf{x}_0)$ for $i \in \{1, \dots, N-1\}$.
Set the last particle $\mathbf{x}_0^{(N)} = \bar{\mathbf{x}}_0$.
Set weights $\mathbf{w}_0 = \mathbf{1}/N$.
for $t = 2$ to T **do**
    Draw ancestor $a_t^{(i)} \propto \mathbf{w}_{t-1}^{(i)}$ for $i \in \{1, \dots, N-1\}$.
    Draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{a_t^{(i)}})$ for $i \in \{1, \dots, N-1\}$.
    Set $\mathbf{x}_t^{(N)} = \bar{\mathbf{x}}_t$.
    Draw $a_t^{(N)} \propto \mathbf{w}_{t-1}^{(i)}\, p(\bar{\mathbf{x}}_t \mid \mathbf{x}_{t-1}^{(i)})$.
    Set $\mathbf{x}_{1:t}^{(i)} = \left[\mathbf{x}_{1:t-1}^{(a_t^{(i)})},\ \mathbf{x}_t^{(i)} \right]$ for $i \in \{1, \dots, N\}$.
    Set $\mathbf{w}_t^{(i)} = p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ for $i \in \{1, \dots, N\}$ and normalize the weights.
end for
Sample $k \propto \mathbf{w}_T$ and set $\bar{\mathbf{x}}_{0:T} = \mathbf{x}_{0:T}^{(k)}$.
Output: Trajectory $\mathbf{x}_{0:T}$.

3. Domain-aware GPSSM

For the linear state-space model in (7), matrix $\mathbf{A}$ assumes a Matrix-Normal distribution $\mathbf{A} \sim \mathcal{MN}(\mathbf{0}, \mathbf{Q}, \mathbf{V})$. As a consequence of the *KKL* theorem, the eigenvalues and hence the hyperparameters of the covariance kernel function form the diagonal of the column covariance matrix $\mathbf{V} = \text{diag}\left(\left[S_\theta^{-1}(\lambda_1), \dots, S_\theta^{-1}(\lambda_{\overline{M}}) \right] \right)$. Note that the eigenfunctions, $\boldsymbol{\phi}^{(m)}(\mathbf{x})$, only depend on the class of chosen covariance function, which are known, and the kernel hyperparameters enter the state space through the matrix $\mathbf{A}$. Since the kernel hyperparameters, θ, enter the state-space equation in (7) through matrix $\mathbf{A}$, learning in RR-GPSSM involves learning the parameters for $\mathbf{A}$ and process noise parameter $\mathbf{Q}$ using the algorithm proposed in [9]. This allows for the computational complexity to be linear in time $\mathcal{O}(T)$ and is well suited for Kalman filtering approaches for large datasets. However, they scale poorly if the dimension of the state-space is high. Assuming the same truncation parameter for each dimension, i.e. $M_d = M$, $\forall d \in D$, the total number of eigenvalues and eigen functions to be considered is $\overline{M} = M^D$, thus limiting the usage of the method to lower dimensional state-space models. We now leverage domain-specific knowledge about the dynamical system being modeled to reduce the computational complexity and propose a modification to the learning algorithm in [9] for efficient computation. Observe from (7), that the transition function for each state, $f_d(.)$, depends upon all state dimensions from the previous time-step $\mathbf{x}_{t-1}$ for maximum expressivity, which may not be required for most dynamical systems.

We assume that each state dimension in $\mathbf{x}_t^{(d)} \in \mathbb{R}$, $d \in D$ depends only on a subset $K_d < D$ of the state vector. Here, we use the superscript $(\cdot)^{(d)}$ to refer to the dth dimension of $\mathbf{x}_t$ unless otherwise stated. We define a selection matrix $\mathbf{S}_d \in \mathbb{R}^{K_d \times D}$, such that $\mathbf{x}_t^{(d)}$ depends only on $\mathbf{S}_d\, \mathbf{x}_{t-1}$, and subsequently (7) can be rewritten as

$$\mathbf{x}_t^{(d)} = \mathbf{A}_d\, \boldsymbol{\phi}(\mathbf{S}_d\, \mathbf{x}_{t-1}) + \mathbf{w}_{t-1}^{(d)} \quad \forall d \in D, \tag{8}$$

where $\mathbf{A}_d \in \mathbb{R}^{1 \times M_d}$, $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{M_d}$ with $M_d = M^{K_d}$ and $\mathbf{w}_{t-1}^{(d)} \sim \mathcal{N}(0, \mathbf{Q}_d)$. Here, the eigenfunction $\boldsymbol{\phi}_t(\cdot)$ is a variadic function taking inputs of different dimensions. We make the following observations.

- The dependencies between the state vectors are modeled through the set of selection matrices, $S \triangleq \{\mathbf{S}_1, \dots, \mathbf{S}_D\}$.
- For all $d \in D$, each state dimension $\mathbf{x}_t^{(d)}$ is modeled with separate kernel hyperparameter denoted by θ_d with its own lengthscale and variance in case of a squared-exponential kernel.
- $\mathbf{A}_d$ follows a multivariate normal distribution, i.e. $\mathbf{A}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_d)$ where $\mathbf{V}_d = \text{diag}\left(\left[S_{\theta_d}^{-1}(\lambda_{d,1}), \dots, S_{\theta_d}^{-1}(\lambda_{d,M_d}) \right] \right)$ and $\lambda_{d,\cdot}$ are the eigenvalues in (6).
- The process covariance matrix $\mathbf{Q} = \text{diag}\left(Q_1, \dots, Q_D\right)$ is a diagonal matrix, elements of which follow an inverse-gamma distribution, i.e. $Q_d \sim \mathcal{IG}(\alpha_d, \beta_d)$ where $\alpha_d \in \mathbb{R}$ determines the shape parameter of the distribution and $\beta_d \in \mathbb{R}$ gives the scale parameter for each dimension.

To learn the trajectory $\mathbf{x}_{0:T}$, the matrix $\mathbf{A}$ and the hyperparameters θ_d, sampling approaches such as Particle Gibbs with Ancestral Sampling (PGAS) Markov kernel [22] and Regularized Auxiliary Particle Chain Filter (RAPCF) [23] can be used. In this work, we use the PGAS kernel, similar to the approach in [24], to define the following statistics

$$\begin{aligned} \boldsymbol{\Phi}_d &= \sum_{t=1}^{T} \mathbf{x}_t^{(d)} \left(\mathbf{x}_t^{(d)} \right)^T, \quad \boldsymbol{\Psi}_d = \sum_{t=1}^{T} \mathbf{x}_t^{(d)}\, \boldsymbol{\phi}^T(\mathbf{S}_d\, \mathbf{x}_{t-1}), \\ \boldsymbol{\Sigma}_d &= \sum_{t=1}^{T} \boldsymbol{\phi}(\mathbf{S}_d\, \mathbf{x}_{t-1})\, \boldsymbol{\phi}^T(\mathbf{S}_d\, \mathbf{x}_{t-1}), \end{aligned} \tag{9}$$

resulting in the following closed form posterior distributions [9]

$$p(\mathbf{A}_d \mid Q_d,\, \mathbf{x}_{1:T}) = \mathcal{MN}\left(\mathbf{A}_d \mid \boldsymbol{\Psi}_d \left(\boldsymbol{\Sigma}_d + \mathbf{V}_d \right)^{-1},\, Q_d,\, \left(\boldsymbol{\Sigma}_d + \mathbf{V}_d \right)^{-1} \right) \tag{10a}$$

$$p(Q_d \mid \mathbf{x}_{1:T}) = \mathcal{IW}\left(Q_d \mid T + \alpha_d,\, \beta_d + \boldsymbol{\Psi}_d \left(\boldsymbol{\Sigma}_d + \mathbf{V}_d \right)^{-1} \boldsymbol{\Psi}_d^T \right). \tag{10b}$$

The posterior distribution for the kernel hyperparameters θ_d for each dimension of the state-space can be factored as

$$p(\theta_d \mid \mathbf{x}_{1:T},\, Q_d,\, \mathbf{A}_d) \propto p(\theta_d)\, p(Q_d \mid \mathbf{x}_{1:T})\, p(\mathbf{A}_d \mid Q_d,\, \mathbf{x}_{1:T}) \tag{11}$$

One can now sample from the above distributions using an MCMC algorithm such as the Metropolis–Hastings sampler. Other more complex MCMC sampling methods such as slice sampling [25] can be used as well. Learning in RR-GPSSM, as given in (7), is summarized in Algorithm 2.

3.1. Selection matrices S

The set of selection matrices S can be constructed from logical matrices that define a binary relation over the set of state dimensions,

Algorithm 2 PGAS for Domain-Aware GPSSM

Input: Measurements over time $\mathbf{y}_{0:T}$, where T is the length of the time series, selection matrices $\mathcal{S}$ constructed as described in Section 3.1 and priors $p(\mathbf{A}, \mathbf{Q})$ and $p(\boldsymbol{\theta})$.
Initialize $\mathbf{x}_{1:T}[0]$ and $\mathbf{A}_d[0]$, $\mathbf{Q}_d[0]$, $\boldsymbol{\theta}_d[0]$ for each dimension $d \in \mathcal{D}$ and total number of iterations N.
for n in 1 to N **do**
  Sample $\mathbf{x}_{1:T}[n+1] | \mathbf{A}_d[n], \mathbf{Q}_d[n], \boldsymbol{\theta}_d[n], \mathbf{S}_d$ for $d \in \mathcal{D}$ using CPF-AS
  for $d \in \mathcal{D}$ **do**
    Sample $\mathbf{Q}_d[n+1] | \mathbf{A}_d[n], \mathbf{x}_{1:T}[n+1]$ using the known posterior in (10b)
    Sample $\mathbf{A}_d[n+1] | \mathbf{Q}_d[n+1], \mathbf{x}_{1:T}[n+1]$ using the known posterior in (10a)
    Sample $\boldsymbol{\theta}_d[n+1] | \mathbf{A}_d[n+1], \mathbf{Q}_d[n+1], \mathbf{x}_{1:T}[n+1]$ using Metropolis–Hastings sampler.
  end for
  Update $n \leftarrow n+1$
end for
Output: Trajectory $\mathbf{x}_{0:T}$, matrix coefficients $\mathbf{A}_d$, process covariance $\mathbf{Q}_d$ and hyperparameters $\boldsymbol{\theta}_d$, $\forall d \in \mathcal{D}$.

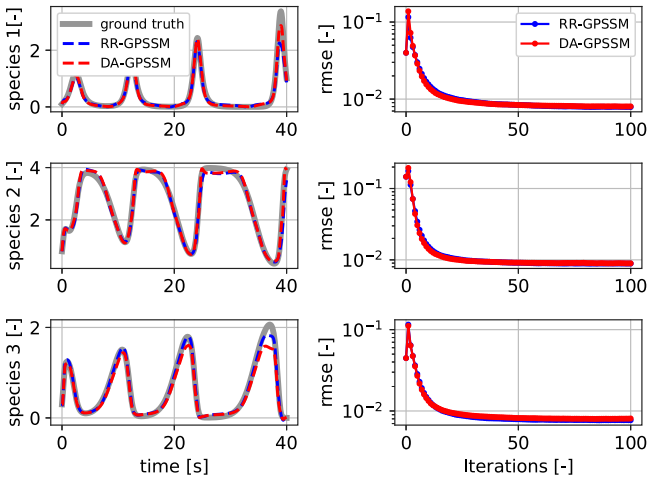

Fig. 1. Mean predicted trajectory plot and rmse for trajectory over iterations for LV Model in (12). Performance is similar with lower complexity for the proposed approach.

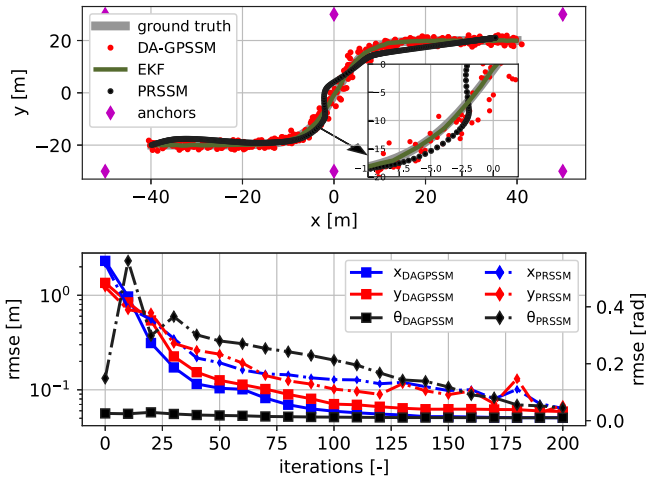

Fig. 2. Comparison w.r.t. state of the art methods: (a) Trajectory tracking of a mobile robot and (b) root-mean-square-error (rmse) of system states in (13).

$\mathcal{D}$. Given a binary relationship $\mathcal{R}$ over the set of indices $\mathcal{D}$, i.e. $\mathcal{R} \subseteq \mathcal{D} \times \mathcal{D}$, the elements of $\mathbf{S}$ is given by

$$[\mathbf{S}]_{ij} = \begin{cases} 1 & (i,j) \subset \mathcal{R} \\ 0 & (i,j) \not\subset \mathcal{R} \end{cases}$$

Denoting the dth row of logical matrix $\mathbf{S}$ as $\mathbf{S}_{d,:}$, the selection matrix for dimension $d \in \mathcal{D}$, can be constructed as $\mathbf{S}_d = \vartheta\{\mathrm{diag}(\mathbf{S}_{d,:})\}$, where the operator $\vartheta\{\cdot\}$ removes all zero rows from a given matrix, e.g. $\vartheta\left\{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right\} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. This allows us to model the transition function for each dimension individually, resulting in separate covariance function for each dimension with its own hyperparameters $\boldsymbol{\theta}_d$. However, for the implementation in [9], the hyperparameters are shared between all output dimensions. Thus, the proposed approach is more flexible. The choice of selection matrices in $\mathcal{S}$ depends on the domain knowledge and is predicated on the assumption that the state transition function for each state does not depend on the all the other states. The simulations in Section 4 illustrate this point in detail.

3.2. Computational and storage analysis

We now present an analysis on the computational and storage aspects of the proposed DA-GPSSM. Let $K \triangleq \max\left\{\mathrm{rank}(\mathbf{S}_d) \mid \mathbf{S}_d \in \mathcal{S}\right\}$ be the maximum rank of the selection matrices in $\mathcal{S}$. If we assume the truncation parameter to be the same for all dimensions, i.e. $M_d = M$, $\forall d$, the number of eigenfunctions to be considered is given by $\widetilde{M} \triangleq M^K$, where $K \leq D$. Thus, the computational time for the algorithm scales with K in the proposed approach. In [9], sampling the hyperparameters $\mathbf{Q}$ and $\mathbf{A}$ depends upon the size of covariance matrices in (7) and is carried out using Cholesky decomposition with computational complexity of $\mathcal{O}(\overline{M}^3)$ while the same operation for Algorithm 2 is $\mathcal{O}(\widetilde{M}^3)$, thus resulting in significant computational gains for $K < D$. The additional hyperparameters related to $\boldsymbol{\theta}_d$ do not have any significant affect on the computational complexity which is dominated by the learning of matrix $\mathbf{A}$. For kernel hyperparameters, $\boldsymbol{\theta}$, Metropolis–Hastings sampler is used for which the computational complexity depends upon on the complexity of the target distribution and the number of iterations required to achieve stationarity. For storage, again the limiting parameters is the matrix $\mathbf{A}$ for [9] and $\mathbf{A}_d$ for our approach. Thus, our approach also reduces the storage requirement from $\mathcal{O}(\overline{M}^2)$ in [9] to $\mathcal{O}(\widetilde{M}^2)$ when $K < D$.

Table 1
Number of coefficients to be learnt for matrix $\mathbf{A}$ in (7) and $\mathbf{A}_d$ in (8), with truncation order $M = 6$.

Use case	RR-GPSSM [9]	DA-GPSSM
Lotka–Volterra model	648	288
Trajectory tracking	23 328	468

4. Simulations

In this section, we illustrate the performance of the proposed method. In Section 4.1, we illustrate the difference in performance

between our approach and the state of the art in [9] on the *Lotka–Volterra* (LV) model. In Section 4.2, we showcase the computational gain using the proposed approach in a localization application for a 2D robot with anchor measurements. The performance for the two cases is evaluated as the root-mean-square error defined as

$$\text{rmse}(\mathbf{x}) = \frac{1}{T}\left(\sqrt{\frac{1}{N_{\text{exp}}}\sum_{n=1}^{N_{\text{exp}}}\left\|\mathbf{x}_{0:T}[n] - \bar{\mathbf{x}}_{0:T}\right\|_2^2}\right),$$

where N_{exp} is the number of Monte-Carlo runs and $\bar{\mathbf{x}}_{0:T}$ is the true trajectory. The code for the simulations in this paper can be found on [26]. For the simulations carried out in this paper, we use $N_{\text{exp}} = 100$ Monte-Carlo runs to calculate the RMSE. The number of particles used for the CPF-AS in Algorithm 1 is $N = 5$ and the proposal distribution for the Metropolis–Hastings sampler in step 8 of Algorithm 2 is taken to be a Gaussian distribution with mean equal to value in the previous iteration and the diagonal covariance matrix.

4.1. Lotka–Volterra model

We consider the *Lotka–Volterra* (LV) model, generally used for simulating the population dynamics in a multi-species ecosystem, to compare the performance of the proposed approach w.r.t. the state of the art (SOTA) implementation in [9]. The state-transition model for such as system with D species is given by [27]

$$\mathbf{x}_t = \mathbf{F}(\mathbf{x}_{t-1}) + \mathbf{w}_t \tag{12a}$$

$$\mathbf{F}(\mathbf{x}_t) = \left(\mathbf{I}_D + \text{diag}\,(\boldsymbol{\alpha})\right)\ \mathbf{x}_t + \mathbf{x}_t \circ (\mathbf{G}\,\mathbf{x}_t) \tag{12b}$$

$$\mathbf{y}_t = \mathbf{x}_t + e_t \tag{12c}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^D$, $\mathbf{G} \in \mathbb{R}^{D\times D}$ and $\circ$ is the Hadamard product. The process noise $\mathbf{w} \sim \mathcal{N}(0, \sigma^2\,\mathbf{I}_D)$. For each dimension in $\mathbf{x}_t$, the dependency on the previous time step depends upon the structure of $\boldsymbol{\alpha}$ and $\mathbf{G}$, where $\boldsymbol{\alpha}$ and $\mathbf{G}$ [27] are

$$\boldsymbol{\alpha} = \begin{bmatrix}3 & 4 & 7.2\end{bmatrix}^T \text{ and } \mathbf{G} = \begin{bmatrix}-0.5 & -1 & 0\\ 0 & -1 & -2\\ -2.6 & -1.6 & -3\end{bmatrix}.$$

The process and measurement noise are assumed to additive Gaussian and are given by $w_t \sim \mathcal{N}(0, \sigma_x^2\,\mathbf{I}_D)$ and $\mathbf{e}_t \sim \mathcal{N}(\mathbf{0}, \sigma_y^2\,\mathbf{I}_D)$, respectively. The variances are chosen as $\sigma_x^2 = \sigma_y^2 = 0.01$. We model the state-transition function using a squared exponential kernel with number of basis functions $M = 6$ per dimension, giving a total of $M^D = 216$ basis functions. This model is chosen because the dimensionality of the data is relatively manageable to showcase the modeling differences from Section 4.3. Given the structure of the matrix $\mathbf{G}$, it should be noted that $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ do not depend on all three dimensions of the previous time step $\mathbf{x}_{t-1}$ and $\mathbf{x}_t^{(3)}$ does. Representing the relation between state dimensions using the logical matrix $\mathbf{S}$, we have $\mathbf{S} = \begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$. Thus, using the process described in Section 3.1, the set of selection matrices can be constructed as

$$\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\} = \left\{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\right\}.$$

Here $K = \text{rank}(\mathbf{S}_3) = 3$, leading to some computational gains, as summarized in Table 1. Fig. 1 shows the predicted estimates of the population of different species and the trajectory rmse while learning using $N_{\text{exp}} = 100$ Monte-Carlo runs. The performance of the two methods is similar. The error in the estimate towards the end of the simulation can be ascribed to lower samples available for learning in the corresponding region of the state-space. The number of coefficients to be learnt in SOTA is $D \times M^D = 648$ and $2M^2 + M^3 = 288$ for the proposed approach leading to marginal computational gains.

4.2. Trajectory tracking of mobile robots

To showcase the computational gains, we consider the problem of trajectory estimation for a mobile robot. The state vector is $\mathbf{x}_t = \begin{bmatrix}x_t & y_t & \theta_t\end{bmatrix}^T$ denoting the spatial location (x, y) and the orientation θ of the robot. The inputs to the state-space model are speed $v_t \in \mathbb{R}$ and angular velocity $\omega_t \in \mathbb{R}$. The state dynamics for a mobile robot is given by

$$x_t = x_{t-1} + v_{t-1}\ \cos(\theta_{t-1})\ \Delta t, \tag{13a}$$

$$y_t = y_{t-1} + v_{t-1}\ \sin(\theta_{t-1})\ \Delta t, \tag{13b}$$

$$\theta_t = \theta_{t-1} + \omega_{t-1}\ \Delta t, \tag{13c}$$

Thus, the state-space model for the system can be written as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathbf{w}_t, \tag{14}$$

where $\mathbf{u}_t := \begin{bmatrix}v_t & \omega_t\end{bmatrix}^T$ denotes the inputs and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the process noise, where $\mathbf{Q} = \text{diag}\,(0.01, 0.01, 0.0003)$. We further assume that there are $P = 6$ fixed anchors measuring distance to the mobile robot, given as $\mathbf{y}_t \in \mathbb{R}^P$. The noise on the distance measurements is assumed to additive white Gaussian noise, $\mathcal{N}(0, \sigma_s^2\,\mathbf{I}_P)$ with $\sigma_s = 0.1$ m [28]. Since the distance measurements do not give any information about the orientation of the robot, we assume the orientation of the robot is directly measured with additive Gaussian noise $\mathcal{N}(0, \sigma_\theta^2)$ with $\sigma = 0.0001$ rad. Thus, the measurement model is given by

$$\mathbf{y}_t = \begin{bmatrix}\mathbf{s}_t\\ \theta_t\end{bmatrix} + \mathbf{e}_t,\ \mathbf{s}_t \triangleq \left[\left\|\mathbf{x}_t - \mathbf{a}_1\right\|_2,\ \ldots,\ \left\|\mathbf{x}_t - \mathbf{a}_P\right\|_2\right]^T,$$

where $\mathbf{a}_p,\ p \in \{1, \ldots, P\}$ are the known anchor positions and $\mathbf{e}_t \sim \mathcal{N}(0, \mathbf{R})$ is the measurement noise such that $\mathbf{R} = \text{blkdiag}(\sigma_s^2\,\mathbf{I}_P, \sigma_\theta^2)$. For this simulation, the state $\mathbf{x}_t$ is appended with the inputs $\mathbf{u}_t$, resulting in $D = 5$. To learn the state transition function and the resulting trajectory, we model the state-transition function using a squared exponential kernel with the truncation parameter $M = 6$ for each dimension. For the implementation in [9], the total number of coefficients to be learnt is $3M^D = 23328$. The domain knowledge related to the nonholonomic motion of the robot can be coded in the selection matrices as explained in Section 3.1, i.e.

$$\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\} = \left\{\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}, \begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix},\right.$$
$$\left.\times \begin{bmatrix}0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1\end{bmatrix}\right\}$$

Thus, exploiting the proposed approach, the number of coefficient to be learnt can be reduced to $(2\,M^3 + M^2) = 468$. Here, $K = 3 < D$ resulting significant computational gains as summarized in Table 1. Due to hardware limitations, the algorithm from [9] could not be implemented for this case. However, the performance and comparison is expected to be similar to LV example in Section 4.1. We compare the performance of the proposed method to the GP-based approach using PRSSMs [13] and include traditional extended Kalman filter (EKF) for reference. For the PRSSM simulation, we have used the open source code available online [29], where we have used $N_z = 50$ inducing points and a learning rate of 0.4 for the Adam optimizer. We have performed $N_{\text{exp}} = 400$ trajectory simulations each with 400 iterations to calculate the root mean square error. However, it should be noted that the implementation of EKF requires the nonlinear transition function as well as the noise parameters to be know a priori, while both DA-GPSSM and PRSSM learn the model as well as perform the trajectory tracking task simultaneously. Additionally, EKF performance is severely affected by the degree of nonlinearity in the system, which is not a limiting factor in the other two cases. Fig. 2 compares the results of the

learning process. For DA-GPSSM we have used $N_{\text{exp}} = 10$ Monte-Carlo runs. As can be inferred from Fig. 2b, DA-GPSSM outperforms PRSSM while being computationally efficient. During the training phase, unlike DA-GPSSM, observe that PRSSM requires the inversion of the covariance matrix for each time step per iteration with a complexity of $\mathcal{O}(T\,N_z^2)$. The computational gains for the two simulation use cases are summarized in Table 1.

4.3. Discussion

Given the selection matrices from Section 3.1, observe that DA-GPSSM models the transition function for each dimension individually, resulting in separate covariance function for each dimension with individual hyperparameters θ_d. On the contrary, the hyperparameters in RR-GPSSM are shared between all output dimensions. Furthermore, in non-homogeneous spaces, similar to the trajectory tracking problem in Section 4.2 with translation and rotation spaces, the modeling assumption in RR-GPSSM of shared hyperparameters is a strong assumption, which is not the case in DA-GPSSM. In addition, RR-GPSSMs can induce additional correlations between state dimensions forcing the model to learn non-existing dynamical behavior. To illustrate this, we refer to the Lotka–Volterra simulation in Section 4.1. The state dimension $\mathbf{x}^{(2)}$ in (12) depends only on dimensions $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$. In DA-GPSSM, this can be modeled using appropriate selection matrix as shown in Section 3.1. However, in RR-GPSSM, it is modeled using all state dimensions and thus is directly affected by the dynamics of state dimension $\mathbf{x}^{(1)}$. This can also be seen by expanding the eigenfunctions for these two cases, as given in (6), i.e.

$$\phi^{(m_1,m_2,m_3)}(\mathbf{x}) = \frac{1}{\sqrt{\mathbf{L}^{(1)}\,\mathbf{L}^{(2)}\,\mathbf{L}^{(3)}}} \sin\left(\frac{\pi\, m_1\,(\mathbf{x}^{(1)}+\mathbf{L}^{(1)})}{2\,\mathbf{L}^{(1)}}\right) \times \sin\left(\frac{\pi\, m_2\,(\mathbf{x}^{(2)}+\mathbf{L}^{(2)})}{2\,\mathbf{L}^{(2)}}\right) \sin\left(\frac{\pi\, m_3\,(\mathbf{x}^{(3)}+\mathbf{L}^{(3)})}{2\,\mathbf{L}^{(3)}}\right) \tag{15a}$$

$$\phi^{(m_2,m_3)}(\mathbf{x}) = \frac{1}{\sqrt{\mathbf{L}^{(2)}\,\mathbf{L}^{(3)}}} \sin\left(\frac{\pi\, m_1\,(\mathbf{x}^{(2)}+\mathbf{L}^{(2)})}{2\,\mathbf{L}^{(2)}}\right) \times \sin\left(\frac{\pi\, m_3\,(\mathbf{x}^{(3)}+\mathbf{L}^{(3)})}{2\,\mathbf{L}^{(3)}}\right) \tag{15b}$$

Thus, DA-GPSSM provides more flexibility as a modeling paradigm in comparison with RR-GPSSM. However, in DA-GPSSM the process noise is assumed to be diagonal and thus may not readily capture colored process noises, which is not a limitation in RR-GPSSM.

5. Conclusions

In this work, we introduced DA-GPSSMs and proposed modifications to the learning algorithm laid out in [9], resulting in computational gains for high-dimensional state-space dynamical models, while maintaining comparable performance. We achieve this by defining a set of selection matrices that model the dependencies between the state dimensions. Given rank M approximation of the covariance function per dimension, the proposed approach can potentially reduce the computation complexity from $\mathcal{O}(\overline{M}^3)$, $\overline{M} = M^D$ to $\mathcal{O}(\widetilde{M}^3)$, $\widetilde{M} = M^K$, $K \leq D$. Unlike the learning algorithm in [9], the proposed approach allows for various state dimensions to be modeled using separate covariance functions leading to more flexibility. We show that the performance of DA-GPSSM is better than other approaches such as PRSSM while being computationally efficient. However, the proposed approach requires the process noise covariance to be diagonal and thus cannot capture correlations between state dimensions. In our future work, we aim to alleviate the requirement for the process covariance to be diagonal, and to explore other samplers such as RAPCF for trajectory sampling and slice sampler for kernel hyperparameters and discuss the trade-offs w.r.t. existing state of the art methods.

CRediT authorship contribution statement

Anurodh Mishra: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Raj Thilak Rajan:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] J. Ko, D. Fox, GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 3471–3476, http://dx.doi.org/10.1109/IROS.2008.4651188.

[2] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, London, ISBN: 026218253X, 2006.

[3] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems, MIT Press, 2000.

[4] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, MIT Press, 2005, URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf.

[5] M. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 5, PMLR, 2009, pp. 567–574, URL: https://proceedings.mlr.press/v5/titsias09a.html.

[6] M. Lazaro-Gredilla, J. Quinonero-Candela, C.E. Rasmussen, A.R. Figueiras-Vidal, Sparse spectrum Gaussian process regression, J. Mach. Learn. Res. (2010).

[7] A. Wilson, H. Nickisch, Kernel interpolation for scalable structured Gaussian processes (KISS-GP), in: Proceedings of the 32nd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 37, PMLR, Lille, France, 2015, pp. 1775–1784, URL: https://proceedings.mlr.press/v37/wilson15.html.

[8] A. Solin, S. Särkkaä, Hilbert space methods for reduced-rank Gaussian process regression, Stat. Comput. (2020) 419–446, http://dx.doi.org/10.1007/s11222-019-09886-w.

[9] A. Svensson, A. Solin, S. Särkkä, T. Schön, Computationally efficient Bayesian learning of Gaussian process state space models, in: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 51, PMLR, Cadiz, Spain, 2016, pp. 213–221.

[10] W. Sternebrg, Identification of Gaussian Process State-Space Models (Master's thesis), Department of Computing, Imperial College London, London, 2017.

[11] Y. Liu, M. Ajirak, P.M. Djurić, Sequential estimation of Gaussian process-based deep state-space models, IEEE Trans. Signal Process. 71 (2023) 2968–2980, http://dx.doi.org/10.1109/TSP.2023.3303648.

[12] R. Frigola, Y. Chen, C.E. Rasmussen, Variational Gaussian process state-space models, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2014, URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/139f0874f2ded2e41b0393c4ac5644f7-Paper.pdf.

[13] A. Doerr, C. Daniel, M. Schiegg, D. Nguyen-Tuong, S. Schaal, M. Toussaint, S. Trimpe, Probabilistic recurrent state-space models, in: Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, PMLR, Stockholm, Sweden, 2018.

[14] X. Fan, E.V. Bonilla, T. O'Kane, S.A. Sisson, Free-form variational inference for Gaussian process state-space models, in: Proceedings of the 40th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 202, 2023, pp. 9603–9622, URL: https://proceedings.mlr.press/v202/fan23a.html.

[15] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, 2022, URL: https://arxiv.org/abs/1312.6114, arXiv:1312.6114.

[16] P. Batz, A. Ruttor, M. Opper, Approximate Bayes learning of stochastic differential equations, 2017, ArXiv e-prints. arXiv:1702.05390v1.

[17] M. Heinonen, C. Yildiz, H. Mannerström, J. Intosalmi, H. Lähdesmäki, Learning unknown ODE models with Gaussian processes, in: 35th International Conference on Machine Learning, ICML, 2018.

[18] R. Hostettler, F. Tronarp, S. Särkkä, Modeling the drift function in stochastic differential equations using reduced rank Gaussian processes, in: 8th IFAC Symposium on System Identification, SYSID, 2018.
[19] A. Damianou, M. Titsias, N. Lawrence, Variational Gaussian process dynamical systems, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2011.
[20] R. Frigola, F. Lindsten, T.B. Schön, C.E. Rasmussen, Bayesian inference and learning in Gaussian process state-space models with particle MCMC, in: Advances in GP-Bayesfilters: Bayesian Filtering, Neural Information Processing Systems, Curran Associates, Inc., 2013, URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/2dffbc474aa176b6dc957938c15d0c8b-Paper.pdf.
[21] A. Svensson, T.B. Schön, M. Kok, Nonlinear state space smoothing using the conditional particle filter, IFAC-Pap. 48 (28) (2015) 975–980, http://dx.doi.org/10.1016/j.ifacol.2015.12.257, 17th IFAC Symposium on System Identification SYSID.
[22] F. Lindsten, M.I. Jordan, T.B. Schön, Particle Gibbs with ancestor sampling, J. Mach. Learn. Res. 15 (2014).
[23] Y. Wu, J.M. Hernández-Lobato, Z. Ghahramani, Gaussian process volatility model, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2014.
[24] A. Wills, T.B. Schön, F. Lindsten, B. Ninness, Estimation of linear systems using a Gibbs sampler, IFAC Proc. Vol. 45 (16) (2012) 203–208, http://dx.doi.org/10.3182/20120711-3-BE-2027.00297, 16th IFAC Symposium on System Identification.
[25] R.M. Neal, Slice sampling, Ann. Stat. 31 (3) (2003) 705–767, http://dx.doi.org/10.1214/aos/1056562461.
[26] A. Mishra, Learning in high-demnsional dynamical systems using Gaussian process state-space model, 2024, https://github.com/asil-lab/AM_C2_domain_rrgpssm.
[27] B.H. Lee, Determination of the parameters in Lotka-Volterra equations from population measurements - algorithms and numerical experiments, SIAM Undergrad. Res. Online (2021) http://dx.doi.org/10.1137/20S1383161.
[28] A. Mishra, R.T. Rajan, Relative kinematics estimation using accelerometer measurements, in: 2022 30th European Signal Processing Conference, EUSIPCO, 2022, pp. 1856–1860, http://dx.doi.org/10.23919/EUSIPCO55093.2022.9909750.
[29] A. Doerr, PR-SSM, 2025, https://github.com/boschresearch/PR-SSM. (Accessed 10 February 2025).