Laplace Approximated Gaussian Process State-Space Models

Abstract

Gaussian process state-space models describe time series data in a probabilistic and non-parametric manner using a Gaussian process transition function. As inference is intractable, recent methods use variational inference and either rely on simplifying independence assumptions on the approximate posterior or learn the temporal states iteratively. The latter hampers optimization since the posterior over the presence can only be learned once the posterior governing the past has converged. We present a novel inference scheme that applies stochastic variational inference for the Gaussian process posterior and the Laplace approximation on the temporal states. This approach respects the conditional dependencies in the model and, through the Laplace approximation, treats the temporal states jointly, thereby avoiding their sequential learning. Our method is computationally efficient and leads to better calibrated predictions compared to state-of-the art alternatives on synthetic data and on a range of benchmark datasets.

1 INTRODUCTION

The world we live in is full of uncertain developments: Will the car behind me overtake in the next five seconds? Will the sun be shining this weekend? How will the stock market evolve in the upcoming months? While we can make more precise predictions by gathering more information, we cannot remove all effects of stochasticity from the world. This especially holds true for sequential data for which we expect that the uncertainty grows over time: While we can still reasonably estimate tomorrow's weather, our estimate will be much less precise if our task is to predict the weather in two weeks' time.

Gaussian processes [Rasmussen and Williams, 2006] provide a well established framework for dealing with uncertainty. They define a distribution over functions $f \sim p(f)$ and can be used as building blocks in state-space models for modeling complex temporal dependencies. This model class, aptly called Gaussian process state-space models [Frigola, 2015], can be used to model observations $y_t \in \mathbb{R}^{d_y}$ from a time series, where $t = 1, \ldots, T$ is the time index. It assumes that each observation y_t is emitted by a latent state $x_t \in \mathbb{R}^{d_x}$, $y_t \sim p(y_t|x_t)$, and that the latent states have a Markovian structure, i.e., $x_{t+1} \sim p(x_{t+1}|x_t, f)$. The noise of this so-called transition model and the function uncertainty of the Gaussian process are propagated over time, resulting in complex and non-Gaussian behavior.

Since the first work on Gaussian process state space models [Wang et al., 2005], much work has been devoted to deriving efficient and accurate inference schemes. The arguably most expressive model at this point is by Ialongo et al. [2019] which assumes a flexible, parametric Markov-structured Gaussian posterior over the temporal states, $x_t \sim q(x_t|x_{t-1}, f)$, and allows for dependencies with the Gaussian process posterior. However, its empirical performance often does not match its theoretical expressiveness: In the original publication the authors report many cases in which this model is outperformed by an easier alternative that simply sets $q(x_t|x_{t-1}, f)$ to the prior transition [Doerr et al., 2018]. The gap between the empirical and theoretical performance of Ialongo et al. [2019] can most likely be attributed to the hard learning problem created by employing a flexible, parametric $q(x_t|x_{t-1}, f)$: These additional parameters have to be estimated, and, moreover, one can only start optimizing the parameters governing the temporal state x_t once the parameters governing x_{t-1} begin converging since the inference scheme is built on sequential sampling of the temporal states.

In this work we address these issues by presenting a novel inference algorithm for Gaussian process state-space models. Our approach applies stochastic variational inference over the Gaussian process posterior [Hensman et al., 2013] and, conditioned on it, the Laplace approximation [see e.g.

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

MacKay, 2003] over the temporal states, thereby allowing for complex dependencies. Inference is performed via a double loop algorithm in which we optimize over the Gaussian process posterior in the outer loop and over temporal states in the inner loop. The resulting approximate posterior over the temporal states, $q(x_t|x_{t-1}, f)$, has a Markov Gaussian form and is found by a joint optimization over all temporal latent states. The latter addresses the issue of Ialongo et al. [2019], in which the parameters governing x_t can only be learned once the estimate of the previous state, x_{t-1}, is meaningful. Our experiments confirm the benefits of this novel inference scheme which provides higher quality uncertainty estimates than its state-of-the-art alternatives.

Our method is computationally efficient since we (i) can compute cheap gradients through the Laplace approximation by using the inverse function theorem [see e.g. Krantz and Parks, 2002], (ii) exploit the Markovian structure of our model [see e.g. Bell, 2000], (iii) approximate the Gaussian process posterior using inducing points [Quinonero-Candela and Rasmussen, 2005] and (iv) apply minibatching.

The remainder of this paper is structured as follows. In Sec. 2, we provide background, while we introduce our new method in Sec. 3. We relate it to existing work in Sec. 4, report experimental results in Sec. 5, and conclude in Sec. 6.

2 BACKGROUND

In this section we provide background on the Laplace approximation for parametric state-space models and on variational inference for Gaussian process state-space models (see also Fig. 4 in Appx. G for qualitative differences between the two approximation techniques).

2.1 STATE-SPACE MODELS AND THE LAPLACE APPROXIMATION

State-space models [see e.g. Särkkä, 2013] offer a principled way to model time series, i.e., noisy observations $Y_T = \{y_t\}_{t=1}^T$ from a dynamical system, where $y_t \in \mathbb{R}^{d_y}$. In order to disentangle the dynamics from the observational noise, state-space models use latent states $X_{T_0} = \{x_t\}_{t=0}^T$ with $x_t \in \mathbb{R}^{d_x}$ that are then assumed to form a Markov sequence.[1] Such a model is completely described by the *initial distribution* $p_\theta(x_0)$, a *transition model* $p_\theta(x_t|x_{t-1})$ and the *emission model* $p_\theta(y_t|x_t)$, resulting in

$$p_\theta(Y_T, X_{T_0}) = p_\theta(x_0) \prod_{t=1}^{T} p_\theta(y_t|x_t) p_\theta(x_t|x_{t-1}), \quad (1)$$

where generally the transition and emission model, and the initial distribution depend on model parameters θ that we wish to infer. Except for linear state-space models [Kalman, 1960], computing the marginal likelihood of the observations Y_T under our model (also called evidence),

$$L(\theta) = p_\theta(Y_T) = \int p_\theta(Y_T, X_{T_0}) dX_{T_0}, \quad (2)$$

is not analytically possible and we have to resort to approximations. In the following, we introduce the Laplace approximation [see e.g. MacKay, 2003, Skaug and Fournier, 2006] which can be used to obtain an approximate maximum likelihood estimate of the parameters: Our goal is to find the setting of the parameters θ^* that maximizes the evidence, i.e., $\theta^* = \operatorname{argmax}_\theta \; L(\theta)$. Defining $g(X_{T_0}, \theta) = \log p_\theta(Y_T, X_{T_0})$ (note that Y_T is constant and hence not considered as a variable), we denote its maximizer[2] with respect to (wrt.) the latent states as $\hat{X}_{T_0} = \operatorname{argmax}_{X_{T_0}} g(X_{T_0}, \theta)$. Performing a second order Taylor approximation of $g(X_{T_0}, \theta)$ around $\hat{X}_{T_0}$, plugging this back into Eq. (2), and then evaluating the resulting Gaussian integral yields (see Appx. B for a detailed derivation)

$$L(\theta) \approx \widetilde{p}_\theta(Y_T) \propto p_\theta(Y_T, \hat{X}_{T_0}) \det\left(H(\theta)\right)^{-\frac{1}{2}}, \quad (3)$$

$$H(\theta) = -\frac{\partial^2}{\partial X_{T_0}^2} g(X_{T_0}, \theta)\Big|_{X_{T_0} = \hat{X}_{T_0}}. \quad (4)$$

Here and in the following we use $\widetilde{p}$ to denote a distribution, where the Laplace approximation has been applied to X_{T_0} (and which therefore depends on $\hat{X}_{T_0}$). The expression in Eq. (3) can then be optimized numerically wrt. θ in order to estimate the parameters θ^* [Skaug and Fournier, 2006]. Note that this can be done efficiently since the Hessian H in Eq. (4) is sparse and structured (see Sec. 3.3). The same methodology can also be applied efficiently to other latent variable models with a sparse or structured Hessian [e.g. Rue et al., 2009, Kristensen et al., 2016].

2.2 VARIATIONAL INFERENCE FOR GAUSSIAN PROCESS STATE-SPACE MODELS

Gaussian process state-space models [GPSSMs, see e.g. Frigola, 2015] are non-parametric extensions of the state-space models in Eq. (1). They employ a Gaussian process (GP) to model the possibly unknown dynamical transitions of the system. Next, we briefly describe GPs, how they can be used in state-space models, and how inference can be performed.

[1] We denote with capital letter with a capital index (Y_T, X_{T_0}) the collection of the respective lower case variables, and the index signifies the length of the collection ($Y_T = \{y_t\}_{t=1}^T$). We use the index T_0 to denote the inclusion of x_0 in the collection X_T.

[2] This is in general a local optimum as the optimization problem is non-convex. However, if the observations are dense, the locally linear approximation to the dynamics made implicitly by the Laplace approximation is a reasonable assumption [see e.g. Eleftheriadis et al., 2017] leading to well-identifiable optima.

Gaussian processes A zero-mean GP $f \sim GP(0, k(\cdot,\cdot))$ is a distribution over functions and is fully specified by a positive-definite, symmetric kernel $k(\cdot,\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$. For every finite set of input points $X_M = \{x_m\}_{m=1}^M$, $x_m \in \mathbb{R}^{d_x}$, the outputs $F_M = \{f(x_m)\}_{m=1}^M$ are distributed according to a Gaussian distribution $p_\theta(F_M) = \mathcal{N}(F_M|0, K_{MM})$, where $K_{MM} = \{k(x_m, x_{m'})\}_{m,m'=1}^M$. Predictions for a new input point x_t can be made by using the joint Gaussianity of $f_t \equiv f(x_t)$ with the F_M, and formulas for Gaussian conditionals resulting in $p(f_t|x_t, F_M) = \mathcal{N}(f_t|\mu(x_t, F_M), \Sigma(x_t))$, with mean and covariance

$$\mu(x_t, F_M) = K_{tM} K_{MM}^{-1} F_M, \tag{5}$$

$$\Sigma(x_t) = k_{tt} - K_{tM} K_{MM}^{-1} K_{tM}^\top. \tag{6}$$

Here $k_{tt} = k(x_t, x_t)$, and $K_{tM} = \{k(x_t, x_m)\}_{m=1}^M$. For a detailed introduction see Rasmussen and Williams [2006].

Gaussian process state-space models In a GPSSM, a GP prior is placed on the (mean of the) transition model that learns the mapping from a latent state x_{t-1} to the next, x_t. Placing iid. Gaussian noise (with variance Q) on the transitions leads to

$$p_\theta(x_t|x_{t-1}, f_{t-1}) = \mathcal{N}(x_t|x_{t-1} + f_{t-1}, Q). \tag{7}$$

The resulting joint model is then given by

$$p_\theta(Y_T, X_{T_0}, F_T) = p_\theta(x_0) p_\theta(F_T|X_{T_0}) \times \prod_{t=1}^T p_\theta(y_t|x_t) p_\theta(x_t|x_{t-1}, f_{t-1}), \tag{8}$$

where $F_T = \{f(x_t)\}_{t=0}^{T-1}$. The emission model and the initial distribution remain unspecified since the methodology introduced in the following does not depend on them.

Variational inference Recent inference methods for GPSSMs [e.g. Doerr et al., 2018, Ialongo et al., 2019] rely on variational inference (VI). This approximate inference method works by choosing a parametric family of distributions $q_\psi(X_{T_0}, F_T)$ and optimizing the setting of the parameters ψ such that q is closest to the true posterior $p_\theta(X_{T_0}, F_T|Y_T)$ as measured by the Kullback-Leibler (KL) divergence $KL(q \parallel p)$. It can be shown that minimizing this KL divergence is equivalent to maximizing the so-called evidence lower bound (ELBO),

$$\mathcal{L}_{\text{VI}}(\psi, \theta) = \mathbb{E}_{q_\psi(X_{T_0}, F_T)} \log \frac{p_\theta(X_{T_0}, F_T, Y_T)}{q_\psi(X_{T_0}, F_T)}, \tag{9}$$

wrt. ψ [for more details see Blei et al., 2016]. Furthermore, since the ELBO is a lower bound to the model evidence $\log p_\theta(Y_T)$ [cf. Eq. (2)], we can simultaneously learn the optimal model parameters θ^* by maximizing Eq. (9) wrt. θ. These model parameters include for instance the hyperparameters of the GP (e.g. kernel length scales).

The different variational approximations to GPSSMs all have in common that they use sparse GPs [Snelson and Ghahramani, 2005, Titsias, 2009, Hensman et al., 2013] to model the GP part of the approximate posterior. The main idea of sparse GPs is to use a set of M pseudo data points $\{X_M, F_M\}$ to summarize the information in the latent GP evaluations F_T. Here, the $X_M = \{x_m\}_{m=1}^M$ with $x_m \in \mathbb{R}^{d_x}$ are the inducing inputs that are placed in the same space as the latent states x_t, and the $F_M = \{f(x_m)\}_{m=1}^M$ are the so-called inducing outputs that share a joint Gaussian distribution, $p_\theta(F_T, F_M)$, with the F_T. The model by Doerr et al. [2018] that we will use in this work employs the fully independent training conditional (FITC) approximation [Snelson and Ghahramani, 2005] that assumes independence of the latent GP evaluations given the inducing outputs, i.e., $p_\theta(F_T|X_{T_0}, F_M) \approx \prod_{t=0}^{T-1} p_\theta(f_t|x_t, F_M)$. Notably, Doerr et al. [2018] also use this approximation in the augmented prior [cf. Eq. (8)], leading to $p_\theta(Y_T, X_{T_0}, F_M) = p_\theta(F_M) p_\theta(Y_T, X_{T_0}|F_M)$ with

$$p_\theta(Y_T, X_{T_0}|F_M) = p_\theta(x_0) \prod_{t=1}^T p_\theta(y_t|x_t) p_\theta(x_t|x_{t-1}, F_M). \tag{10}$$

Here we have already marginalized the F_T from the prior model which can be done because of the FITC assumption:

$$\begin{aligned} &p_\theta(x_t|x_{t-1}, F_M) \\ &= \int p_\theta(x_t|x_{t-1}, f_{t-1}) p_\theta(f_{t-1}|x_{t-1}, F_M) df_{t-1} \\ &= \mathcal{N}(x_t|x_{t-1} + \mu(x_{t-1}, F_M), Q + \Sigma(x_{t-1})), \end{aligned} \tag{11}$$

where we used Eqs. (5)- (7) and standard formulas for Gaussian integrals. While the FITC assumption might seem quite restrictive and is criticized by Ialongo et al. [2019], we show in Appx. A that recent methods [including Ialongo et al., 2019] implicitly use this approximation as well, i.e., the methods could have used the approximation and would have arrived at the same optimization objective. This substantially weakens the criticism of Ialongo et al. [2019] on the choice of Doerr et al. [2018] on the FITC prior and motivates us to use it as well in our method. We discuss differences between these and other works in Sec. 4.

3 METHOD

In this section we propose a new inference method for the model used in Doerr et al. [2018], i.e. Eq. (10). Instead of relying solely on VI as previous works have done, we employ it in combination with the Laplace approximation. This allows us to treat the local latent variables, i.e. the temporal states X_{T_0}, and the global latent variables, i.e. the inducing outputs F_M, differently. Note that naively applying the Laplace approximation to this model, i.e., without making a distinction between the different sets of latent variables, (i) does not lead to an efficient algorithm since the resulting

Hessian would not have an exploitable sparsity structure (see Sec. 3.3) and (ii) would assume a linear relationship between the two latent variable classes.

We start by deriving our optimization objective in Sec. 3.1 and then, in Sec. 3.2, discuss a caveat that we encounter when trying to naively optimize this objective. Afterwards, we show how the sparsity of the Hessian in the Laplace approximation can be exploited for computational savings (Sec. 3.3) before summarizing the algorithm in Sec. 3.4.

3.1 COMBINING VARIATIONAL INFERENCE AND THE LAPLACE APPROXIMATION

Similarly as the previous methods, we wish to find an approximation to the log marginal likelihood $\log p_\theta(Y_T)$. We start from its definition given the model under consideration,

$$\log p_\theta(Y_T) = \log \int p_\theta(Y_T|F_M)p_\theta(F_M)dF_M, \tag{12}$$

where $p_\theta(Y_T|F_M) = \int p_\theta(Y_T, X_{T_0}|F_M)dX_{T_0}$. Using the Laplace approximation (Sec. 2.1) on this integral leads to

$$\log p_\theta(Y_T) \approx \log \int \widetilde{p}_\theta(Y_T|F_M)p_\theta(F_M)dF_M. \tag{13}$$

Here, $\widetilde{p}_\theta(Y_T|F_M)$ is given by [cf. Eq. (3)]

$$\widetilde{p}_\theta(Y_T|F_M) \propto p_\theta(Y_T, \hat{X}_{T_0}|F_M) \det\left(H(\theta, F_M)\right)^{-\frac{1}{2}}, \tag{14}$$

where $\hat{X}_{T_0}$ is a mode of the log-density

$$g_{\text{GP}}(X_{T_0}, \theta, F_M) = \log p_\theta(Y_T, X_{T_0}|F_M), \tag{15}$$

that can be evaluated using Eq. (10), and H is the corresponding Hessian (cf. Eq. (4)). We proceed by using the VI methodology to lower bound the expression in Eq. (13): First, we multiply the term within the integral by $q_\psi(F_M)/q_\psi(F_M)$, where $q_\psi(F_M)$ is the variational distribution, an arbitrary distribution over the F_M parameterized by ψ. Then, we use Jensen's inequality to push the logarithm inside of the resulting integral (thereby lower bounding the expression), and use the definition of the KL-divergence to arrive at our optimization objective,

$$\begin{aligned}\mathcal{L}(\theta, \psi) = &\int q_\psi(F_M) \log \widetilde{p}_\theta(Y_T|F_M)dF_M \\ &- KL(q_\psi(F_M) \parallel p_\theta(F_M)),\end{aligned} \tag{16}$$

which obeys $\log p_\theta(Y_T) \gtrapprox \mathcal{L}(\theta, \psi)$. There are two things to note abound this bound: First, it is an approximate lower bound, since the Laplace approximation does not provide a valid bound but only an approximation. Second, the property that we minimize a KL-divergence between the true posterior $p_\theta(F_M|Y_T)$ and the approximate posterior $q_\psi(F_M)$ by optimizing this bound (see Sec. 2.2), also only holds approximately. In fact, the KL-divergence that is minimized by optimizing this bound is $KL\left[q_\psi(F_M) \parallel \widetilde{p}_\theta(F_M|Y_T)\right]$, i.e. $q_\psi(F_M)$ approximates the posterior after applying the Laplace approximation to the latent states X_{T_0}.

In principle, we could now go ahead, choose a parametric family for $q_\psi(F_M)$ and then evaluate our bound in Eq. (16) and use automatic differentiation to optimize the parameters θ and ψ. However, this is inefficient for two reasons: First, evaluating Eq. (14) involves an optimization to obtain $\hat{X}_{T_0}$ and automatic differentiation through this optimization is inefficient. Second, we need the determinant of the Hessian $H(\theta, F_M)$ in Eq. (14) which scales cubically in its size. In the following two subsections we provide solutions to both of these efficiency problems.

3.2 IMPLICIT FUNCTION THEOREM

Next, we turn to an important dependence in our construction: The mode $\hat{X}_{T_0}$ depends on the setting of the model parameters θ. Since our optimization objective $\mathcal{L}(\theta, \psi)$ in Eq. (16) involves the mode $\hat{X}_{T_0}$ of the log-density $g_{\text{GP}}(X_{T_0}, \theta, F_M)$ [Eq. (15)], we require the derivative $\frac{\partial \hat{X}_{T_0}}{\partial \theta}$ in order to compute $\frac{\partial \mathcal{L}}{\partial \theta}$.[3] Being used to automatic differentiation we would usually leave this calculation to our favorite framework, since obtaining the mode $\hat{X}_{T_0}$ is nothing but a (rather long) sequence of summations and multiplications which automatic differentiation can deal with. Nevertheless, since several optimization steps are needed to compute the mode, backpropagating through optimization would lead to an enormous memory footprint and long execution times. Instead, we can calculate the derivative of the mode wrt. θ solely with the value of $\hat{X}_{T_0}$, i.e. independent of the steps taken to get there, with the help of the implicit function theorem [IFT, see e.g. Krantz and Parks [2002]].[4] In Appx. C, we show that the required derivative can be obtained as

$$\frac{\partial \hat{X}_{T_0}(\theta)}{\partial \theta} = H^{-1}(\theta, F_M)\frac{\partial h(\hat{X}_{T_0}, \theta, F_M)}{\partial \theta}, \tag{17}$$

which we derive using the IFT and where h [Eq. (42) in Appx. C] is the Jacobian of the function g_{GP} [Eq. (15)] [see also Skaug and Fournier, 2006]. Both terms on the right hand side of Eq. (17) can be obtained using automatic differentiation and require only the value of $\hat{X}_{T_0}$ such that the complete computational graph of how it has been obtained is no longer required. Note that Eq. (17) exchanges potentially costly automatic differentiation computations with a Hessian solve. Naively, this would incur memory and time costs scaling quadratically and cubically in the size of the latent state, respectively. Therefore, this does only lead to

[3] Through the F_M, there is also a dependence on ψ which must be treated in the same way. We omit the equivalent derivation for the sake of clarity and conciseness.

[4] The website http://implicit-layers-tutorial.org/ provides an introduction to the IFT geared to the Machine Learning practitioner.

an efficient algorithm by also exploiting the structure of the Hessian, as we will discuss next.

3.3 STRUCTURE OF THE HESSIAN

Taking a closer look at the definition of the Hessian $H(\theta, F_M)$ appearing in Eqs. (14) and (17), $H(\theta, F_M) = -\frac{\partial^2 g_{\text{GP}}(X_{T_0}, \theta, F_M)}{\partial X_{T_0}^2}$, we realize that it is given as the second partial derivatives of a sum of $T+1$ terms since g_{GP} [Eq. (15)] is defined as the log-density of the distribution $p_\theta(Y_T, X_{T_0}|F_M)$ that consists of a product of $T+1$ terms [Eq. (10)]. Due to the Markovian structure of our model, all second partial derivatives wrt. latent states being more than one time step t apart vanish, i.e., $H_{tt'}(\theta, F_M) = -\frac{\partial^2 g_{\text{GP}}(X_{T_0}, \theta, F_M)}{\partial x_t \partial x_{t'}} = 0$ for $t' \notin \{t-1, t, t+1\}$. This results in a block-tridiagonal structure of the Hessian:

$$H = \begin{pmatrix} A_0 & B_1 & 0 & \cdots & 0 \\ B_1^\top & A_1 & B_2 & \ddots & \vdots \\ 0 & B_2^\top & A_2 & B_3 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & B_T^\top & A_T \end{pmatrix}, \tag{18}$$

where $A_t = -\frac{\partial^2 g_{\text{GP}}}{\partial x_t \partial x_t}, B_t = -\frac{\partial^2 g_{\text{GP}}}{\partial x_t \partial x_{t-1}} \in \mathbb{R}^{d_x \times d_x}$ such that $H \in \mathbb{R}^{d_x(T+1) \times d_x(T+1)}$. This structure can also be found in similar models [see e.g. Bell, 2000] and the recent work by Durrande et al. [2019] considers the efficient implementation of computations for similar structures (banded matrices) into automatic differentiation frameworks.

The structure in Eq. (18) reveals another interesting aspect of our algorithm: The first step of the Laplace approximation consists of making a multivariate Gaussian approximation to the posterior $p_\theta(X_{T_0}|Y_T, F_M)$, where $\hat{X}_{T_0}$ is the mean and $H(\theta, F_M)$ is the precision matrix (see Appx. B, especially Eq. (40)). Therefore, the structure of H tells us something about the underlying (implicit) structural assumption that we have used to approximate $p_\theta(X_{T_0}|Y_T, F_M)$. Exploiting the structure in Eq. (18) and using standard formulas for Gaussian conditionals, we can rewrite the approximate posterior to find a linear Markov Gaussian model,

$$\mathcal{N}\left(X_{T_0}|\hat{X}_{T_0}, H^{-1}\right) = \prod_{t=0}^{T} \mathcal{N}\left(x_t|a_t + b_t x_{t-1}, c_t\right), \tag{19}$$

where the coefficients a_t, b_t, and c_t depend on $\hat{X}_{t-1:T}$ and the blocks $A_{t:T}$ and $B_{t:T}$ of the Hessian H (here we used the shorthand $A_{t:T} = \{A_{t'}\}_{t'=t}^T$). Such a linear Markov Gaussian model is also used by Eleftheriadis et al. [2017], where the authors use parameters a_t, b_t, and c_t that have to be optimized during inference while the conditional dependence on the F_M is not taken into account.

There are three aspects of the algorithm for which we can achieve considerable computational savings when we take the structure of the Hessian into account: i) obtaining the blocks A_t and B_t [Eq. (18)] of the Hessian while not calculating the unnecessary zero blocks, ii) calculating the determinant of the Hessian required for Eq. (14), and finally iii) performing the Hessian solve in Eq. (17). Problem i), while not hard, requires a technical solution which we detail in Appx. D.1. Problems ii) and iii) can both be solved by following e.g. Koulaei and Toutounian [2007] in noting that the Hessian H in Eq. (18) allows a factorization that can be exploited as we show in Appx. D.2. Implementing these improvements leads to a reduction of the memory footprint of the algorithm from $\mathcal{O}(T^2 d_x^2)$ to $\mathcal{O}(T d_x^2)$ [mainly through i)] and a reduction of the theoretical runtime from $\mathcal{O}(T^3 d_x^3)$ to $\mathcal{O}(T d_x^3)$ through ii) and iii).

3.4 ALGORITHM

In order to evaluate and optimize our optimization objective $\mathcal{L}(\theta, \psi)$ in Eq. (16), we need to choose a parametric family for $q_\psi(F_M)$. We follow the literature [e.g. Ialongo et al., 2019] and take a Gaussian distribution $q_\psi(F_M) = \mathcal{N}(F_M|m, S)$, allowing an analytical evaluation of the KL-term in Eq. (16). The first term on the right hand side of Eq. (16) is analytically intractable so we resort to sampling,

$$\int q_\psi(F_M) \log \widetilde{p}_\theta(Y_T|F_M) dF_M \approx \sum_{n=1}^{N} \log \widetilde{p}_\theta(Y_T|F_M^{(n)}), \tag{20}$$

with $F_M^{(n)} \sim q_\psi(F_M)$, and we use reparameterized samples [e.g. Kingma and Welling, 2014] to be able to compute derivatives wrt. ψ. The resulting basic algorithm to evaluate and optimize $\mathcal{L}(\theta, \psi)$ is summarized in Alg. 1 in Appx. E.

There are three extensions of this algorithm that are typically required for applying GPSSMs in practice (e.g. in Sec. 5.2), i) minibatches, ii) multi-dimensional latent states and iii) control inputs: Many time series are too long to be handled in one batch such that using i) minibatches helps obtaining a computationally tractable algorithm. For many real-world problems a one-dimensional latent state is not expressive enough [Frigola, 2015] and we require ii) multi-dimensional latent states x_t. Lastly, many datasets come with an additional time series $u_t \in \mathbb{R}^{d_u}$ of iii) external inputs that control the behavior of the system. We discuss the implementation of theses features into Alg. 1 in Appx. E.2.

4 RELATED WORK

There are two lines of work that directly relate to our approach: The first is on inference techniques for GPSSMs, the second on optimizing model parameters for latent variable models using the Laplace approximation.

Inference for GPSSMs The idea of using GPs to model transitions in state-space models goes back to Wang et al.

[2005], where learning was performed by finding a maximum a posteriori estimate of the latent variables. The first Bayesian treatment of the latent states and the transition function in GPSSMs can be found in Frigola et al. [2013] using Markov Chain Monte Carlo. Due to the computational complexity of this approach, later works focused on variational approximations, beginning with Frigola et al. [2014]: They used a sparse GP in the flavor of Titsias [2009], introducing inducing outputs F_M to deal with the transition function and using an independence assumption between the F_M and the latent states X_{T_0} in the variational posterior. This allowed them to find optimal variational distributions $q(F_M)$ and $q(X_{T_0})$ using variational calculus, where the latter is analytically intractable. This leads, similarly as in our work, to a double-loop algorithm, where in the inner loop the distribution $q(X_{T_0})$ is approximated and in the outer loop $q(F_M)$ has to be obtained. The authors of Frigola et al. [2014] opt for a particle filtering method in the inner loop (note that the Laplace approximation would have also been possible here), but in contrast to our work do not take the conditional dependencies between the F_M and the X_{T_0} into account. Furthermore, this assumption allows for alternating updates for the parameters of F_M and X_{T_0} without having the need to differentiate through the latent states X_{T_0}.

Eleftheriadis et al. [2017] improved upon this method by using a doubly stochastic variational inference scheme that allows for the first time, and similarly as our approach, for minibatches. They opt for a parametric Gaussian distribution $q(F_M)$, a linear Markov Gaussian model for $q(X_{T_0})$ and additionally employ a recognition model to amortize the inference of the many parameters of such an approach. It is worth noting that our Laplace approximation (implicitly) also leads to a linear Markov Gaussian model as an approximation to the posterior $p(X_{T_0}|Y_T, F_M)$. However, our approach respects the conditional dependence of the temporal states on the inducing outputs F_M, and does not require any additional parameters to be learned since the means and variances of the Markov Gaussian model are obtained from the mode $\hat{X}_{T_0}$ and the Hessian H.

Recently, variational methods have also incorporated the dependence between the F_M and the X_{T_0} in their approximations: First Doerr et al. [2018], who have simply used the prior for $q(X_{T_0}|F_M)$ and then Ialongo et al. [2019], who employ a parametric non-linear Markov Gaussian model for the same term. A further subtle difference between these two approaches is that Doerr et al. [2018] use the FITC approximation from Snelson and Ghahramani [2005] in prior and approximate posterior, while Ialongo et al. [2019] do not use it and criticize its implications. However, it was already noted by Frigola et al. [2014], that the optimization objective for their model would not change if they applied the FITC approximation.We have confirmed in Appx. A, that this is also the case for the model of Ialongo et al. [2019], thus making this subtle difference even smaller.

Interestingly, even though the model of Doerr et al. [2018] is a special case of Ialongo et al. [2019], the latter work reports many cases in which the easier model still outperforms the more powerful one. This can potentially be attributed to the harder learning problem obtained through the additional introduction of the parameters of the Gauss-Markov model for $q(X_{T_0}|F_M)$ which only allows to learn the parameters of $q(x_t|x_{t-1}, F_M)$ after the preceding state x_{t-1} has reached a meaningful state due to the sequential nature of the algorithm. In contrast to these two methods, we do not employ a variational distribution $q(X_{T_0}|F_M)$ but rather approximately marginalize $X_{T_0}|Y_T, F_M$ through the Laplace approximation.

Laplace approximation The other line of related work consists of approaches using the Laplace approximation for parameter estimation in latent variable models. One influential approach is Rue et al. [2009] that uses the Laplace approximation (twice) to perform approximate Bayesian inference of the model parameters in latent Gaussian models and pays close attention to the sparsity of the Hessian that is induced by different models. The work of Skaug and Fournier [2006] combines the Laplace approximation with automatic differentiation methods to arrive at an algorithm that can be used to approximate the marginal likelihood of (non-) Gaussian latent state models and therefore for maximum likelihood parameter estimation. The software package described in Kristensen et al. [2016] provides a recent implementation of the ideas in Skaug and Fournier [2006] with an additional focus on efficient automatic differentiation exploiting the sparsity of the Hessian. While closely connected to our work, these methods do not cover our approach since they jointly treat all latent variables, $\{F_M, X_{T_0}\}$, which would not lead to a Hessian whose structure can be exploited. Furthermore, a Laplace approximation over all latent variables would also lead to a less expressive approximation: This would entail a joint Gaussianity assumption of F_M and X_{T_0}, while our current approach approximates both sets of variables with Gaussian distributions but allows for potentially complex and non-Gaussian interactions.

5 EXPERIMENTS

We first test the ability of our inference scheme to deal with different noise sources on the toy dataset called *kink* and then assess the performance of our method on a range of real-world benchmark datasets. Both experiments confirm that our new approach results in better calibrated predictions when compared with the Variationally Coupled Dynamics and Trajectories (VCDT) method from Ialongo et al. [2019] and the Probabilistic Recurrent State-Space Model (PRSSM) method from Doerr et al. [2018].

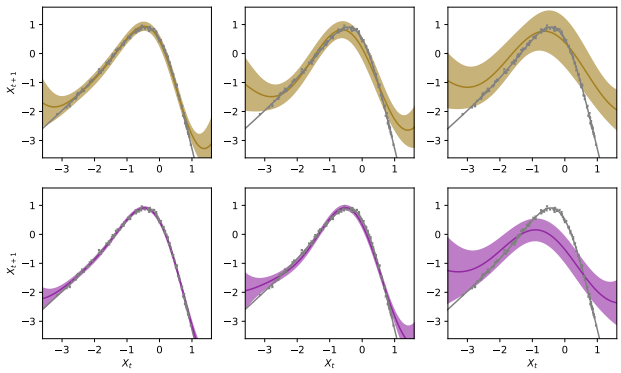

Figure 1: Sparse GP fits (mean $\pm$ 2σ confidence interval) to the *kink* transition function for our model (top) and VCDT (bottom) for varying emission noise ($\sigma_y^2 \in \{0.008, 0.08, 0.8\}$, left to right). Each plot shows the kink function (solid gray) and one sequence of $T = 120$ noisy latents x_t drawn from that function (tiny gray crosses).

5.1 KINK

The *kink* function $f_k(x) = 0.8 + (x + 0.2)[1 - 5/(1 + e^{-2x})]$ (Ialongo et al. [2019], see also Figs. 1 and 5 in Appx. G) can be used as a challenging transition function to probe state-space models: It tests the ability to model the non-linear transition function and, by injecting additional noise, also the ability of the inference scheme to deal with different levels of emission noise. We generate data according to

$$x_t \sim \mathcal{N}\left(x_t | f_k(x_{t-1}), \sigma_x^2\right), \quad y_t \sim \mathcal{N}\left(y_t | x_t, \sigma_y^2\right),$$

for $t = 1, \ldots, T$, where we fix $T = 120$, $x_0 = 0.5$, $\sigma_x = 0.05$ and vary $\sigma_y^2 \in \{0.008, 0.08, 0.8\}$. Here, $\sigma_y^2 = 0.8$ corresponds to the setting of Ialongo et al. [2019] for which they empirically demonstrated that the inference scheme of Doerr et al. [2018] is not able to cope with the transition noise σ_x and fails to learn the underlying dynamics. We follow Ialongo et al. [2019] and fix the emission model to the groundtruth, $p(y_t|x_t) = \mathcal{N}\left(y_t | x_t, \sigma_y^2\right)$. In addition, we choose a zero-mean sparse GP transition model with trainable Gaussian noise Q [cf. Eq. (11)], $p(x_t|x_{t-1}, F_M) = \mathcal{N}\left(x_t | \mu(x_{t-1}, F_M), Q + \Sigma(x_{t-1})\right)$ and a fixed initial distribution $p(x_0) = \mathcal{N}\left(x_0 | -0.5, 1.5\right)$, all with one dimensional latent states x_t. We defer further details, the description of the setup of VCDT [Ialongo et al., 2019], and the initialization and training routines to Appx. F.1.

We present the resulting fits of the sparse GPs to the kink transition function and the noisy latent transitions in Fig. 1. We find that the GP in our model is well able to locate the kink in the *kink* transition function and finds better approximations with increasing signal to noise ratio (decreasing σ_y^2). The latter is also true for the VCDT method while the former does not hold for the largest $\sigma_y^2 = 0.8$. The small confidence intervals of the VCDT method sometimes result in very good fits ($\sigma_y^2 = 0.008$), while other times leading to overconfident predictions (slightly for $\sigma_y^2 = 0.08$, evidently for $\sigma_y^2 = 0.8$), an observation that we have also made for the benchmark datasets in Sec. 5.2. In contrast, our method provides higher variance estimates resulting in very few data points not lying within the confidence intervals.

Table 1: Comparison of our method with VCDT [Ialongo et al., 2019] on the *kink* data set. Shown are mean and standard errors over ten repetitions of the log-density (higher is better) of the *kink* function varying the emission noise variance σ_y^2. See Appx. F.1 for more details.

MODEL	$\sigma_y^2 = 0.008$	$\sigma_y^2 = 0.08$	$\sigma_y^2 = 0.8$
LAPLACE	1.35(0.04)	**0.36(0.08)**	**-1.08(0.15)**
VCDT	1.53(0.31)	-1.10(0.72)	-4.16(1.97)

Next, we repeat the experiment 10 times, each time generating a new random dataset. Our observations also hold quantitatively as demonstrated in Tab. 1. In accordance with Fig. 1, we find that VCDT performs slightly, but not significantly, better than our method for $\sigma_y^2 = 0.008$, whereas we significantly outperform VCDT on the noisier data sets.

The controllable environment of the kink dataset also allows us to test if the theoretical speedups from Sec. 3.3 can be observed in practice. For this, we use the same setup as above, changing only $\sigma_y^2 = 0.01$ to provide an easy learning task and vary $T \in \{2^4, 2^5, 2^6, 2^7, 2^8\}$. We then compare the average runtime per iteration over the first 1000 iterations of our efficient implementation with a naive implementation, where all elements of the Hessian are calculated and the full Hessian is used to compute its determinant and the required Hessian solves, i.e., ignoring the improvements proposed in Sec. 3.3. The results of this comparison are depicted in Fig. 3 and clearly show that our efficient implementation scales linearly with the length of the time series T. For the naive implementation we observe a quadratic scaling with T even though the theoretical scaling is $\mathcal{O}(T^3)$ (see Sec. 3.3). We attribute this to the huge cost of calculating the elements of the Hessian with automatic differentiation whose number scales quadratically for the naive version and linearly for our implementation. We hypothesize that the cubic scaling only sets in for very high values of T which might become important if one wants to study long term effects requiring minibatches of increased size.

5.2 SYSTEM IDENTIFICATION

We compare the performance of our method against PRSSM [Doerr et al., 2018] and VCDT [Ialongo et al., 2019] on five time series system identification benchmark datasets. Those consist of one dimensional time series of various lengths between 296 and 1024 data points and an equally long time

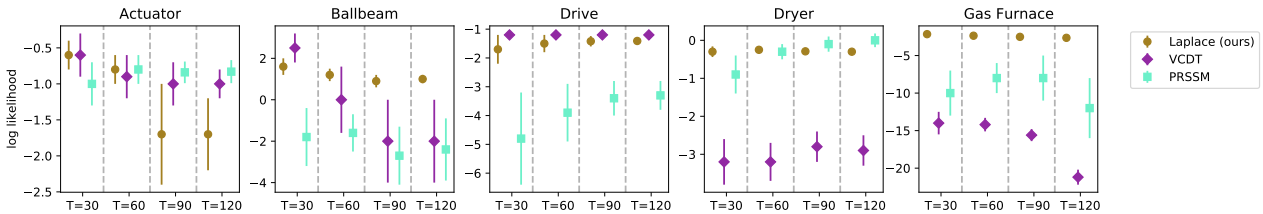

Figure 2: Comparison of average predictive log-likelihoods (higher is better) and their standard errors for ten repetitions on five different benchmark datasets. We evaluate our model (Laplace), VCDT [Ialongo et al., 2019] and PRSSM [Doerr et al., 2018] on predictions for $T \in \{30, 60, 90, 120\}$ steps in the future. See Appx. F.2 for more details.

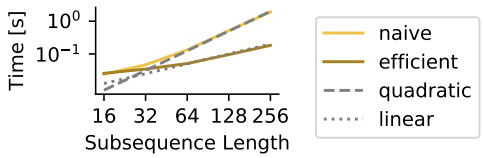

Figure 3: Timing comparison between a naive Laplace GPSSM and our proposed efficient implementation. The log-log plot shows the average time per training iteration as a function of the subsequence length T.

series of one dimensional control inputs (see the appendix of Doerr et al. [2018] for more information about the datasets). For these more complicated tasks we choose a two dimensional latent state x_t and a residual transition model with a sparse GP as in Eq. (11) with (diagonal) trainable Gaussian noise Q. We furthermore keep the initial distribution uninformative, $p(x_0) = \mathcal{N}(x_0|0, 1)$, but choose a slightly more expressive emission model [following Ialongo et al., 2019] $p(y_t|x_t) = \mathcal{N}(y_t|Cx_t + b, \Omega)$, where we fix $C = [1, 0]^\top$ and introduce trainable parameters b and Ω. Note that an even more expressive emission model does not lead to more expressivity of the composite model, only to more non-identifiabilities [Frigola, 2015]. For the PRSSM and VCDT methods, we use the original models detailed in Doerr et al. [2018] and Ialongo et al. [2019], respectively. For each dataset, we create ten different training tasks by varying the starting index of the training sequence, while keeping the length of the training sequence fixed to one half of the whole time series. Whereas Doerr et al. [2018] only compared the long-term predictions and Ialongo et al. [2019] only the short-term predictions, we evaluate both regimes by recording the predictive performance for varying time horizons in $T \in \{30, 60, 90, 120\}$. For more information about data splits, model configurations as well as training and prediction routines for each method, see Appx. F.2.

We plot the resulting test log-likelihoods in Fig. 2 (for a tabular comparison, see Tab. 2 in Appx. G). Our results clearly demonstrate that our method is a valuable addition to the set of inference methods for GPSSMs: For short-term predictions ($T = 30, 60$), our methods yields excellent results over all datasets, while VCDT shows deteriorated behavior on *Dryer* and *Gas Furnace*, and PRSSM on *Ballbeam*, *Drive* and *Gas Furnace*. For long-term predictions ($T = 90, 120$), our methods significantly outperforms its competitors on *Ballbeam* and *Gas Furnace*, while it underperforms on *Actuator*. We further report the root-mean-square errors (RMSE) in Tab. 3 in Appx. G. It is evident that a deteriorated log-likelihood value does not necessarily result in a large RMSE-value or vice versa. Instead, a drop in log-likelihood values is more likely caused by overconfident predictions which we can also witness in our exemplary plots in Fig. 6 in Appx. G.

In sum, we observed in our experiments that our method is able to learn the underlying dynamics for a variety of different tasks. It is well known in the Bayesian community that using too compact variational families can result in narrow uncertainty estimates [e.g. Turner and Sahani, 2011]. Our experiments indicate that combining variational inference with the Laplace approximation favors less compact predictive distributions that lead to better calibrated predictions.

6 SUMMARY

In this paper, we have developed a new inference method for Gaussian process state-space models (GPSSMs) that combines a Laplace approximation over the temporal states x_t with variational inference over the inducing outputs of the Gaussian process (GP) part of the model. Our approach learns a joint approximate posterior over the inducing outputs and the temporal states, refraining from sequentially learning the latter. We empirically find that our inference scheme is rewarded by better calibrated predictions compared to state-of-the-art methods.

While we only focused on the application of our inference scheme to GPSSMs, it can generally be applied to all models with two distinct sets of latent variables, e.g., the Bayesian treatment of model parameters in latent Gaussian models Rue et al. [2009] or of hyperparameters in (sparse) GP models [e.g. Hensman et al., 2013]. We further deem exchanging our variational inference engine with Hamiltonian Monte Carlo [Margossian et al., 2020] as an interesting avenue of future work on GPSSMs since recent research has shown promising results for a fully Bayesian treatment over GP hyperparameters and inducing locations [Rossi et al., 2021].

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and others. Tensorflow: a system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*, 2016.

Vincent Adam, Paul E. Chang, Mohammad Emtiyaz Khan, and Arno Solin. Dual Parameterization of Sparse Variational Gaussian Processes. In *Advances in Neural Information Processing Systems*, 2021.

Christopher Aicher, Yi-An Ma, Nicholas J. Foti, and Emily B. Fox. Stochastic Gradient MCMC for State Space Models. *SIAM Journal on Mathematics of Data Science*, 1(3):555–587, 2019.

Bradley M Bell. The marginal likelihood for parameters in a discrete Gauss-Markov process. *IEEE Transactions on Signal Processing*, 48(3):870–873, 2000.

David Blei, Alp Kucukelbir, and Jon McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2016.

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic Recurrent State-Space Models. In *International Conference on Machine Learning*, 2018.

Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis, and James Hensman. Banded Matrix Operators for Gaussian Markov Models in the Automatic Differentiation Era. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of Gaussian Process State Space Models. In *Advances in Neural Information Processing Systems*, 2017.

R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

Roger Frigola. *Bayesian time series learning with Gaussian processes*. PhD thesis, University of Cambridge, 2015.

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 *Advances in Neural Information Processing Systems*, 2013.

Roger Frigola, Yutian Chen, and Carl E. Rasmussen. Variational Gaussian Process State-Space Models. In *Advances in Neural Information Processing Systems*, 2014.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

Alessandro Davide Ialongo, Mark van der Wilk, James Hensman, and Carl Edward Rasmussen. Overcoming Mean-Field Approximations in Recurrent Gaussian Process Models. In *International Conference on Machine Learning*, 2019.

Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

Diederik Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

M. H. Koulaei and F. Toutounian. On computing of block ILU preconditioner for block tridiagonal systems. *Journal of Computational and Applied Mathematics*, 202(2):248–257, 2007.

Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

Kasper Kristensen, Anders Nielsen, Casper W. Berg, Hans Skaug, and Bradley M. Bell. TMB : Automatic Differentiation and Laplace Approximation. *Journal of Statistical Software*, 70(5):1–21, 2016.

Krista Longi, Jakob Lindinger, Olaf Duennbier, Melih Kandemir, Arto Klami, and Barbara Rakitsch. Traversing Time with Multi-Resolution Gaussian Process State-Space Models. *arXiv preprint arXiv:2112.03230*, 2021.

David MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

Charles Margossian, Aki Vehtari, Daniel Simpson, and Raj Agrawal. Hamiltonian Monte Carlo using an adjoint-differentiated Laplace approximation: Bayesian inference for latent Gaussian models and beyond. In *Advances in Neural Information Processing Systems*, 2020.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40): 1–6, 2017.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive computation and machine learning. MIT Press, 2006.

Simone Rossi, Markus Heinonen, Edwin V. Bonilla, Zheyang Shen, and Maurizio Filippone. Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009.

Hugh Salimbeni, Stefanos Eleftheriadis, and James Hensman. Natural Gradients in Practice: Non-Conjugate Variational Inference in Gaussian Process Models. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Davod Khojasteh Salkuyeh. Comments on "A note on a three-term recurrence for a tridiagonal matrix". *Applied Mathematics and Computation*, 176(2):442–444, 2006.

Hans J. Skaug and David A. Fournier. Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. *Computational Statistics & Data Analysis*, 51(2):699–709, 2006.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2005.

Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

Michalis Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics*, 2009.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In *Bayesian Time series models*, pages 109–130. Cambridge University Press, 2011.

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A Framework for Interdomain and Multioutput Gaussian Processes. *arXiv preprint arXiv:2003.01115*, 2020.

Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian Process Dynamical Models. In *Advances in Neural Information Processing Systems*, 2005.

A GPSSMS AND THE FITC APPROXIMATION

In this section we formalize and prove a remark on the properties of variational approximations for GPSSMs that have been used in the literature so far. More precisely, we show that the ELBO of Ialongo et al. [2019] remains unchanged when the FITC approximation [Snelson and Ghahramani, 2005], $p(F_T \mid F_M) \approx \prod_{t=0}^{T-1} p(f_t \mid F_M)$ is used in both the prior and the approximate posterior. This remark is a direct extension of the equivalent remark of Frigola et al. [2014] about the same property of their model, which can be found in their appendix.

Before we begin, we first reiterate some previous results: The ELBO of the VCDT method from Ialongo et al. [2019] is given as

$$\begin{aligned}\mathcal{L}_{\text{VCDT}} = &\sum_{t=1}^{T} \mathbb{E}_{q(x_t)} \left[\log p(y_t \mid x_t)\right] - KL\left[q(x_0) \parallel p(x_0)\right] - KL\left[q(F_M) \parallel p(F_M)\right] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q(x_{t-1}, f_{t-1}, F_M)} \left[KL\left[q(x_t \mid F_M, x_{t-1}) \parallel p(x_t \mid f_{t-1}, x_{t-1})\right]\right],\end{aligned} \tag{21}$$

where the posterior marginals are given by

$$q(x_t) = \int \left[\prod_{t'=1}^{t} q(x_{t'} \mid F_M, x_{t'-1})\right] q(x_0) q(F_M) dF_M dx_{0:t-1}, \tag{22}$$

$$q(x_{t-1}, f_{t-1}, F_M) = \int \left[\prod_{t'=1}^{t-1} q(x_{t'} \mid F_M, x_{t'-1})\right] q(x_0) q(F_M) p(f_{t-1}|F_M) dx_{0:t-2}, \tag{23}$$

where we use the shorthand $dx_{i:j} = \prod_{t'=i}^{j} dx_{t'}$. This can be obtained by using the prior and approximate posterior,

$$p(Y_T, X_{T_0}, F_T, F_M) = p(x_0) p(F_T \mid F_M) p(F_M) \prod_{t=1}^{T} p(y_t \mid x_t) p(x_t \mid f_{t-1}, x_{t-1}), \tag{24}$$

$$q(X_{T_0}, F_T, F_M) = q(x_0) p(F_T \mid F_M) q(F_M) \prod_{t=1}^{T} q(x_t \mid F_M, x_{t-1}), \tag{25}$$

respectively. To clarify some of the above points, we first of all note that the expectation value is defined as $\mathbb{E}_{p(a,b)}\left[g(a,b)\right] = \int p(a,b) g(a,b) da db$, and the KL-divergence as $KL\left[q(a) \parallel p(a)\right] = \int q(a) \log \frac{q(a)}{p(a)} da$. The ELBO in Eq. (21) is the same as in Ialongo et al. [2019], where we simply collected all the terms that are somewhat spread over their paper and use our notation (introduced in Sec. 2). The marginals in Eqs. (22) and (23) are not written out in as much detail in Ialongo et al. [2019], but can be simply obtained from plugging in Eqs. (24) and (25) in the general formula for the ELBO in Eq. (9) and collecting the remaining terms. This will be almost equivalent to the very similar derivation that we perform as part of the proof below. Finally, the exact definitions of the distributions q and p appearing in Eqs. (21) - (25) [given in Ialongo et al., 2019] are not very important for our current undertaking, only their conditional (in)dependencies.

After these preliminaries we are ready to formally state our observation about the FITC approximation:

Remark 1. *The ELBO in Eq.* (21) *remains unchanged when the FITC approximation,* $p(F_T \mid F_M) \approx \prod_{t=0}^{T-1} p(f_t \mid F_M)$*, is used in Eqs.* (24) *and* (25)*.*

Proof of Remark 1. The proof is conceptually simple: It relies on altering Eqs. (24) and (25) with the FITC approximation and then calculating the ELBO by using Eq. (9) and showing that this coincides with Eqs. (21) - (23). The concepts needed in the proof are also very simple: We only require the definition of expectation value and KL-divergences from above, rules for the logarithm and the property for arbitrary probability densities $q(a,b)$ that

$$\int q(a,b) g(b) da db = \int q(b) g(b) \left(\int q(a \mid b) da\right) db = \int q(b) g(b) db, \tag{26}$$

due to the normalization property of probability distributions.

We start by defining the alternative VCDT-FITC model by plugging in the FITC approximation $p(F_T \mid F_M) \approx \prod_{t=0}^{T-1} p(f_t \mid F_M)$ in Eqs. (24) and (25):

$$p_{\text{FITC}}(Y_T, X_{T_0}, F_T, F_M) = p(x_0)p(F_M)\prod_{t=1}^{T} p(f_{t-1} \mid F_M)p(y_t \mid x_t)p(x_t \mid f_{t-1}, x_{t-1}), \tag{27}$$

$$q_{\text{FITC}}(X_{T_0}, F_T, F_M) = q(x_0)q(F_M)\prod_{t=1}^{T} q(x_t \mid F_M, x_{t-1})p(f_{t-1} \mid F_M). \tag{28}$$

Plugging these equations in Eq. (9) yields the formula for the ELBO of this model:

$$\mathcal{L}_{\text{VCDT-FITC}} = \mathbb{E}_{q_{\text{FITC}}(X_{T_0}, F_T, F_M)}\left[\log \frac{p_{\text{FITC}}(Y_T, X_{T_0}, F_T, F_M)}{q_{\text{FITC}}(X_{T_0}, F_T, F_M)}\right] \tag{29}$$

Evaluating the terms inside the logarithm according to Eqs. (27) and (28) yields

$$\mathcal{L}_{\text{VCDT-FITC}} = \mathbb{E}_{q_{\text{FITC}}(X_{T_0}, F_T, F_M)}\left[\sum_{t=1}^{T}\log p(y_t \mid x_t) + \log\frac{p(x_0)}{q(x_0)} + \log\frac{p(F_M)}{q(F_M)} + \sum_{t=1}^{T}\log\frac{p(x_t \mid f_{t-1}, x_{t-1})}{q(x_t \mid F_M, x_{t-1})}\right]. \tag{30}$$

Note that here the identical FITC terms, $\prod_{t=0}^{T-1} p(f_t \mid F_M)$ in the prior and approximate posterior have canceled each other out inside the logarithm, but this also happens for the terms $p(F_T \mid F_M)$ in the original VCDT formulation [Eqs. (24) and (25)].

Using the definition of the expectation value, then the property in Eq. (26) and then the definition of the KL-divergence, we identify two of the KL-terms in Eq. (21):

$$\begin{aligned}\mathcal{L}_{\text{VCDT-FITC}} = \mathbb{E}_{q_{\text{FITC}}(X_{T_0}, F_T, F_M)}&\left[\sum_{t=1}^{T}\log p(y_t \mid x_t) + \sum_{t=1}^{T}\log\frac{p(x_t \mid f_{t-1}, x_{t-1})}{q(x_t \mid F_M, x_{t-1})}\right] \\ &- KL\left[q(x_0) \parallel p(x_0)\right] - KL\left[q(F_M) \parallel p(F_M)\right].\end{aligned} \tag{31}$$

In the following we carefully treat the remaining two terms inside the expectation: First,

$$\begin{aligned}&\mathbb{E}_{q_{\text{FITC}}(X_{T_0}, F_T, F_M)}\left[\sum_{t=1}^{T}\log p(y_t \mid x_t)\right] \\ &= \sum_{t=1}^{T}\int q(x_0)q(F_M)\left[\prod_{t'=1}^{T} q(x_{t'} \mid F_M, x_{t'-1})\cancel{p(f_{t'-1} \mid F_M)}\right]\log p(y_t \mid x_t)dF_M dx_{0:T}\cancel{df_{0:T-1}} \\ &= \sum_{t=1}^{T}\int\left(\int q(x_0)q(F_M)\left[\prod_{t'=1}^{t} q(x_{t'} \mid F_M, x_{t'-1})\right]dF_M dx_{0:t-1}\right)\log p(y_t \mid x_t)dx_t \\ &= \sum_{t=1}^{T}\mathbb{E}_{q(x_t)}\left[\log p(y_t \mid x_t)\right],\end{aligned} \tag{32}$$

which we recognize as the first term in Eq. (21). Here, we have used Eq. (26) to get rid of the integrals over all $f_{t'}$ and over all $x_{t'}$ for $t' > t$ in the second step and we have identified $q(x_t)$ from Eq. (22) in the last step.

We note that for the term in Eq. (32) as well as for the KL-terms in Eq. (31), the difference between the FITC and the non-FITC approximate posterior does not play a role since the terms within the expectation value do not depend on any f_t.

This is different for the remaining term in Eq. (31), which has a dependence on f_t:

$$
\begin{aligned}
&\mathbb{E}_{q_{\text{FITC}}(X_{T_0},F_T,F_M)}\left[\sum_{t=1}^{T}\log\frac{p(x_t \mid f_{t-1},x_{t-1})}{q(x_t \mid F_M,x_{t-1})}\right]\\
=&\sum_{t=1}^{T}\int\left[q(x_0)q(F_M)\left(\prod_{t'=1}^{T}p(f_{t'-1}\mid F_M)q(x_{t'}\mid F_M,x_{t'-1})\right)\log\frac{p(x_t \mid f_{t-1},x_{t-1})}{q(x_t \mid F_M,x_{t-1})}dF_M dx_{0:T}df_{0:T-1}\right]\\
=&\sum_{t=1}^{T}\int\left[q(x_0)q(F_M)p(f_{t-1}\mid F_M)\prod_{t'=1}^{t-1}q(x_{t'}\mid F_M,x_{t'-1})dx_{0:t-2}\right.\\
&\qquad\left.\times\left(\int q(x_t\mid F_M,x_{t-1})\log\frac{p(x_t \mid f_{t-1},x_{t-1})}{q(x_t \mid F_M,x_{t-1})}dx_t\right)dF_M df_{t-1}dx_{t-1}\right]\\
=&-\sum_{t=1}^{T}\mathbb{E}_{q(x_{t-1},f_{t-1},F_M)}\left[KL\left[q(x_t\mid F_M,x_{t-1})\parallel p(x_t\mid f_{t-1},x_{t-1})\right]\right].
\end{aligned}
\tag{33}
$$

We recognize Eq. (33) as the last term in Eq. (21). Here, we have used Eq. (26) to get rid of the integrals over all $f_{t'}$ for $t' \neq t-1$ and over all $x_{t'}$ for $t' > t$ in the second step and additionally did some reordering of the terms. In the last step, we have identified $q(x_{t-1}, f_{t-1}, F_M)$ from Eq. (23) and a (negative) KL-divergence. It is interesting to note that at this point the non-FITC variational approximation still would not make a difference: Since every summand within the expectation value in the first line of Eq. (33) only depends on a single f_t, the switching of summation and expectation makes both versions, $p(F_T \mid F_M)$ and $\prod_{t=0}^{T-1} p(f_t \mid F_M)$, reduce to $p(f_{t-1} \mid F_M)$ [using Eq. (26)] in the end.

Together, Eqs. (29) - (33) show that the ELBO of the VCDT-FITC model matches that of the original VCDT model in Eq. (21), therefore completing the proof. □

We furthermore note that since Eleftheriadis et al. [2017] use the same conditional (in)dependencies in their variational family as Frigola et al. [2014], the remark in the appendix of the latter work holds also true for the former. In summary, all practically viable variational treatments of GPSSMs either explicitly use the FITC approximation in both prior and approximate posterior [Doerr et al., 2018] or implicitly do so [Frigola et al., 2014, Eleftheriadis et al., 2017, Ialongo et al., 2019], i.e., the latter methods could have used the approximation and would have arrived at the same optimization objective.[5] This somewhat weakens the criticism of Ialongo et al. [2019] on the choice of Doerr et al. [2018] to use a FITC prior and motivates us to do the same in our work.

B DERIVATION OF THE LAPLACE APPROXIMATION

In this section we provide a more extensive derivation for the Laplace approximation in our notation, see e.g. MacKay [2003], Skaug and Fournier [2006], Kristensen et al. [2016] for derivations in other notations or contexts. As explained in Sec. 2.1, we use the Laplace approximation to approximate the marginal likelihood [Eq. (2)],

$$
p_\theta(Y_T) = \int p_\theta(Y_T, X_{T_0})dX_{T_0}. \tag{34}
$$

In some more detail, we do this with the following steps: We first introduce an exponential and a logarithm that cancel each other out,

$$
p_\theta(Y_T) = \int \exp\left[\log p_\theta(Y_T, X_{T_0})\right] dX_{T_0}. \tag{35}
$$

Then we find a mode $\hat{X}_{T_0}$ of $\log p_\theta(Y_T, X_{T_0})$,

$$
\hat{X}_{T_0} = \operatorname{argmax}_{X_{T_0}} \log p_\theta(Y_T, X_{T_0}), \tag{36}
$$

and perform a second order Taylor expansion of $\log p_\theta(Y_T, X_{T_0})$ around this mode leading to

$$
p_\theta(Y_T) \approx \int \exp\left[\log p_\theta(Y_T, \hat{X}_{T_0}) - \frac{1}{2}\left(\hat{X}_{T_0} - X_{T_0}\right)^\top H\left(\hat{X}_{T_0} - X_{T_0}\right)\right] dX_{T_0}. \tag{37}
$$

[5] This does not hold for the method described as "non-factorised non-linear" in Ialongo et al. [2019], but the $\mathcal{O}(T^3)$ scaling makes this and similar methods unattractive to use in practice.

Here H is the negative Hessian of $\log p_\theta(Y_T, X_{T_0})$ as in Eq. (4) and the first order term vanishes since we are expanding around a mode, where, by definition, the gradient vanishes. The first term in the exponential in Eq. (37) is constant in X_{T_0} and can be pulled out of the integral:

$$p_\theta(Y_T) \approx p_\theta(Y_T, \hat{X}_{T_0}) \int \exp\left[-\frac{1}{2}\left(\hat{X}_{T_0} - X_{T_0}\right)^\top H \left(\hat{X}_{T_0} - X_{T_0}\right)\right] dX_{T_0}. \tag{38}$$

Remaining in the integral, we recognize the exponential of a negative quadratic form, i.e., an unnormalized multivariate Gaussian, which generally has the density [see e.g. MacKay, 2003],

$$\mathcal{N}(x|\mu, \Sigma) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mu - x)^\top \Sigma^{-1} (\mu - x)\right], \tag{39}$$

with mean μ and covariance matrix Σ. We therefore recognize $\hat{X}_{T_0}$ as the mean and H as the inverse of the covariance matrix, i.e., the precision matrix, in Eq. (38), leading to

$$p_\theta(Y_T) \approx p_\theta(Y_T, \hat{X}_{T_0}) \int \det(2\pi H^{-1})^{\frac{1}{2}} \mathcal{N}\left(X_{T_0}|\hat{X}_{T_0}, H^{-1}\right) dX_{T_0}. \tag{40}$$

This integral is very easy to solve since the factor with the determinant does not depend on X_{T_0}, leaving us with an integral over a normalized probability density which, by definition, evaluates to 1. Therefore

$$\begin{aligned} p_\theta(Y_T) &\approx p_\theta(Y_T, \hat{X}_{T_0}) \det(2\pi H^{-1})^{\frac{1}{2}} \\ &= \sqrt{2\pi}^{d_x(T+1)} p_\theta(Y_T, \hat{X}_{T_0}) \det(H)^{-\frac{1}{2}}, \end{aligned} \tag{41}$$

where we used standard rules for determinants, using that H is of dimension $d_x(T+1) \times d_x(T+1)$. This is the formula that can be found in Eq. (3).

C USING THE IMPLICIT FUNCTION THEOREM

In order to obtain a formula for $\frac{\partial \hat{X}_{T_0}(\theta)}{\partial \theta}$ we start by defining a function h, that is the Jacobian of the function g_{GP} in Eq. (15):

$$h(x, \theta, F_M) = -\left.\frac{\partial \log p_\theta(Y_T, X_{T_0} \mid F_M)}{\partial X_{T_0}}\right|_{X_{T_0}=x}. \tag{42}$$

By definition, plugging in the mode $\hat{X}_{T_0}$ for the fixed value $\hat{\theta}$ of the parameters θ that has been used when obtaining $\hat{X}_{T_0}$, yields a vanishing Jacobian, $h(\hat{X}_{T_0}(\hat{\theta}), \hat{\theta}, F_M) = 0$. Here, we made the dependence of the mode on the parameter setting $\hat{\theta}$ explicit. Under mild differentiability assumptions, the IFT, roughly speaking, guarantees that $\hat{X}_{T_0}(\theta)$ is in fact a function of θ that is *implicitly defined* through a vanishing Jacobian in the vicinity of $\hat{\theta}$, i.e., through the equation

$$h(\hat{X}_{T_0}(\theta), \theta, F_M) = 0, \tag{43}$$

and that is also differentiable at $\hat{\theta}$ and in its vicinity. Therefore, we can calculate the total derivative wrt. θ on both sides of Eq. (43), yielding

$$\frac{\partial h(\hat{X}_{T_0}, \theta, F_M)}{\partial \theta} + \left.\frac{\partial h(x, \theta, F_M)}{\partial x}\right|_{x=\hat{X}_{T_0}} \frac{\partial \hat{X}_{T_0}(\theta)}{\partial \theta} = 0, \tag{44}$$

where we used the chain rule. Recognizing

$$\left.\frac{\partial h(x, \theta, F_M)}{\partial x}\right|_{x=\hat{X}_{T_0}} = H(\theta, F_M),$$

i.e., the Hessian H [cf. Eq. (4)] of the function g_{GP} in Eq. (15), we can solve Eq. (44) for the required derivative, obtaining

$$\frac{\partial \hat{X}_{T_0}(\theta)}{\partial \theta} = H^{-1}(\theta, F_M) \frac{\partial h(\hat{X}_{T_0}, \theta, F_M)}{\partial \theta}. \tag{45}$$

This is Eq. (17) appearing in Sec. 3.2.

D EXPLOITING STRUCTURE AND SPARSITY OF THE HESSIAN

In the following we provide some technical details on how we obtain the blocks of the Hessian in Eq. (18) in Sec. D.1 before we show in Sec. D.2 how these blocks can be used to efficiently calculate the determinant of the Hessian and performing a Hessian solve needed for Eqs. (14) and (17), respectively.

D.1 EFFICIENTLY OBTAINING THE NON-ZERO BLOCKS OF THE HESSIAN

Reverse mode automatic differentiation frameworks such as TensorFlow [Abadi et al., 2016] implement efficient vector-Jacobian products. As we can think of the Hessian as the Jacobian of the Jacobian of the function g_{GP} in Eq. (15), we can naively obtain the $d_x(T+1)$ columns of the Hessian in Eq. (18) by considering the vector-Hessian products $e_j^\top H$ with all $d_x(T+1)$ unit vectors $e_j = \{\delta_{jj'}\}_{j'=0}^{d_x T} \in \mathbb{R}^{d_x(T+1)}$, where δ_{ij} is the Kronecker delta. This would require $\mathcal{O}(T^2 d_x^2)$ storage and computation time. As in Sec. 3.3, this is wasteful since many unnecessary zeros are being calculated. Therefore, in order to tackle the problem of obtaining only the non-zero blocks of the Hessian, we propose instead to use only vector-Hessian products with the three block-vectors $\widetilde{e}_0$, $\widetilde{e}_1$, and $\widetilde{e}_2$ defined by $\widetilde{e}_k = \{\delta_{k,k'\%3}\mathbb{I}_{d_x}\}_{k'=0}^{T} \in \mathbb{R}^{d_x(T+1)\times d_x}$. Here $\%$ denotes the modulo operation and $\mathbb{I}_{d_x}$ is the identity matrix of size $d_x \times d_x$. As an example, $\widetilde{e}_0 = (\mathbb{I}_{d_x}, 0, 0, \mathbb{I}_{d_x}, \cdots)^\top$ such that

$$\widetilde{e}_0^\top H = (A_0, B_1, B_3^\top, A_3, B_4, B_6^\top, \cdots)^\top,$$

which can be implemented as d_x vector-Hessian products. Similarly $\widetilde{e}_1^\top H = (B_1^\top, A_1, B_2, \cdots)^\top$ and $\widetilde{e}_2^\top H = (0, B_2^\top, A_2, B_3 \cdots)^\top$ such that we can obtain all non-zero elements of the Hessian with only $3d_x$ vector-Hessian products, reducing the memory and time requirements to $\mathcal{O}(T d_x^2)$. After some reshaping and transposing this provides us with the quantities $\{A_t\}_{t=0}^T$ and $\{B_t\}_{t=1}^T$ which are needed for the further steps in our sparse algorithm in the following section.

D.2 EFFICIENTLY CALCULATING THE HESSIAN DETERMINANT AND PERFORMING HESSIAN SOLVES

We follow e.g. Koulaei and Toutounian [2007] in noting that the Hessian H in Eq. (18) allows the factorization

$$H = (\Lambda + B^\top)\Lambda^{-1}(\Lambda + B), \tag{46}$$

where B is the strictly upper-triangular part of H (consisting only of the different B_t blocks) and Λ is the block diagonal matrix of recursively defined blocks

$$\Lambda_0 = A_0, \qquad \Lambda_t = A_t - B_t^\top \Lambda_{t-1}^{-1} B_t, \quad t = 1, \ldots, T. \tag{47}$$

The factorization in Eq. (46) allows us to calculate the determinant of the Hessian as

$$\det H = \prod_{t=0}^{T} \det \Lambda_t, \tag{48}$$

using $\det(\Lambda + B^\top) = \det(\Lambda + B) = \det \Lambda$ (since Λ is block diagonal and B and $B^\top$ are strictly upper and lower triangular, respectively), and $\det(CD) = \det C \det D$ [see also Salkuyeh, 2006]. The Hessian solve in Eq. (17) can be done by exploiting Eq. (46) as well, yielding

$$H^{-1} = (\Lambda + B)^{-1}\Lambda(\Lambda + B^\top)^{-1}. \tag{49}$$

Therefore a solve with a block banded lower triangular matrix $(\Lambda + B^\top)$, then a matrix multiplication with a block diagonal matrix (Λ) followed by a solve with a block banded upper triangular matrix $(\Lambda + B)$ are equivalent to a solve with H. Since all of these operations can be performed in $\mathcal{O}(T d_x^3)$ steps, which is true as well for the calculation of the blocks of Λ in Eq. (47) and the determinant in Eq. (48), we have achieved the desired theoretical speed-ups.

We note that further speed-ups are possible, such as considering the inverse subset algorithm to calculate the derivative of the logarithm of the determinant of the Hessian [see e.g. Kristensen et al., 2016, Durrande et al., 2019]. Furthermore, several inherently sequential parts of the code could be written in C++ as is done in Durrande et al. [2019].

Algorithm 1 ELBO for GPSSM

Given time series Y_T
Choose number of iteration and samples I and N, latent state dimension d_x
Initialize model parameters θ, variational parameters $\psi = \{m, S\}$
for $i = 1 \dots I$ **do** ▷ Optimization steps
  for $n = 1 \dots N$ **do** ▷ Reparameterized samples
    Sample $F_M^{(n)} \sim q_\psi(F_M)$
    Find $\hat{X}_{T_0}^{(n)}$ by maximizing $g_{\text{GP}}(X_{T_0}, \theta, F_M^{(n)})$ ▷ Eq. (15)
    Obtain non-zero elements of H (A_t, B_t) ▷ Appx. D.1
    Evaluate $\det H$ ▷ Eqs. (47) and (48)
    Evaluate $\widetilde{p}_\theta(Y_T \mid F_M^{(n)})$ ▷ Eq. (14)
  end for
  Evaluate $\mathcal{L}(\theta, \psi)$ ▷ Eqs. (16) and (20)
  Obtain gradients $\frac{\partial \mathcal{L}}{\partial \theta}$, $\frac{\partial \mathcal{L}}{\partial \psi}$, using custom $\frac{\partial \hat{X}_{T_0}^{(n)}}{\partial \theta}$, $\frac{\partial \hat{X}_{T_0}^{(n)}}{\partial \psi}$ ▷ Eqs. (17) and (49)
  Update θ and ψ
end for

E ALGORITHM AND IMPLEMENTATION DETAILS

In Alg. 1, we provide the basic algorithm to evaluate and optimize the optimization objective $\mathcal{L}(\theta, \psi)$ in Eq. (20). Then, in the following we provide some details on our implementation of Alg. 1. First general details in Sec. E.1 and then some practical details that are required to make the algorithm work on practical problems in Sec. E.2.

E.1 GENERAL IMPLEMENTATION DETAILS

We build our implementation on TensorFlow [Abadi et al., 2016] and use the multioutput GP implementation provided by GPflow [van der Wilk et al., 2020]. For finding the mode in Eq. (36) required for the Laplace approximation, we use the BFGS algorithm [see e.g. Fletcher, 1987] provided by TensorFlow-Probability. For the implementation of the custom derivative coming from the Implicit Function Theorem in Eq. (17), we use the *custom_gradient* function provided by TensorFlow.

E.2 PRACTICAL IMPLEMENTATION DETAILS

In the following we first tackle the practical details of the three extensions of Alg. 1 mentioned in Sec. 3.4: Minibatches, multi-dimensional latent states and control inputs. At the end of this section, we also discuss some practical details related to the optimization required to get the mode $\hat{X}_{T_0}$ for the Laplace approximation.

Minibatches Time series are typically too long to be handled in one batch such that using minibatches helps obtaining a computationally tractable algorithm. The first term in our optimization objective [Eq. (20)] can be written as a sum over the observations y_t [cf. Eqs. (14) and (10)]. We can approximate this using minibatches (or rather subsequences) of length T_b, $Y_{T_b} = \{y_t\}_{t=t_0}^{t_0+T_b}$ starting at some (random) time index t_0. Note that this is an approximation that ignores the temporal structure of the data and ignores effects coming from observation and transitions before and after the minibatch which is discussed in more detail in Aicher et al. [2019]. Nevertheless, minibatching nicely integrates with the sampling step in Eq. (20), where we can draw a new minibatch for every sample from $q_\psi(F_M)$, resulting in a new approximation,

$$\int q_\psi(F_M) \log \widetilde{p}_\theta(Y_T \mid F_M) dF_M \approx \frac{T}{T_b} \sum_{n=1}^{N} \log \widetilde{p}_\theta(Y_{T_b}^{(n)} \mid F_M^{(n)}), \quad F_M^{(n)} \sim q_\psi(F_M). \tag{50}$$

On a more technical note, we find that when using minibatches, we do not need a buffer to reduce the effect of the initial distribution as suggested in Aicher et al. [2019] and e.g. used in Longi et al. [2021]. This might be the case since we usually normalize the data and use an uninformative initial distribution such as $p(x_0) = \mathcal{N}(x_0|0, 1)$, such that the Laplace approximation can flexibly find a value for $\hat{x}_0$ (or $\hat{x}_{t_0-1}$ for the minibatches) that has high mass under the initial distribution.

Multi-dimensional latent states For challenging problems a one-dimensional latent state is typically not expressive enough [Frigola, 2015] and we require multi-dimensional latent states x_t. This is problematic in the transition model [Eq. (11)], where the mean and the covariance of the GP appear, both of which are one-dimensional. We follow the literature [e.g. Doerr et al., 2018, Ialongo et al., 2019] and choose independent (sparse) GPs for each dimension of the latent state resulting in

$$p_\theta(x_t \mid x_{t-1}, F_M^{d_x}) = \prod_{d=1}^{d_x} \mathcal{N}\left(x_t^{(d)} | x_{t-1}^{(d)} + \mu^{(d)}(x_{t-1}, F_M^{(d)}), Q_d + \Sigma^{(d)}(x_{t-1})\right),$$

where $F_M^{d_x} = \{F_M^{(d)}\}_{d=1}^{d_x}$ is the collection of all inducing outputs, $x_t^{(d)}$ is the d-th dimension of the latent state and $\mu^{(d)}$ and $\Sigma^{(d)}$ are the mean and covariance of the sparse GP responsible for the d-th dimension, respectively [cf. Eqs. (5) and (6)]. In order to reduce non-identifiabilities and to simplify training, we choose a diagonal transition noise variance, $Q = \text{diag}(\{Q_d\}_{d=1}^{d_x})$, the same inducing points X_M and kernel hyperparameters for all GPs, thus only leaving the variational distributions $q_{\psi_d}^{(d)}(F_M^{(d)})$ as the distinguishing property of the different dimensions of the transition function.

Control inputs Lastly, time series prediction problems often come with an additional time series $U_T = \{u_t\}_{t=1}^T, u_t \in \mathbb{R}^{d_u}$ of control inputs that are applied at time index t and change the behavior of the system. In GPSSMs these are modeled as additional (constant) inputs to the GP which are simply concatenated with the latent states x_t at the same time index. This leads only to two small changes in the algorithm: The kernel of the GPs have to accept input pairs coming from $\mathbb{R}^{d_x+d_u}$ and the inducing points X_M also have to live in that higher dimensional space.

Optimizing the latent states In order to obtain the mode $\hat{X}_{T_0}$ required for the Laplace approximation, we have to perform an optimization in the inner loop of our algorithm (Alg. 1). This requires some fine-tuning to get a numerically stable algorithm: First, since this is a non-stochastic problem, we use an approximate second-order optimizer for fast convergence and choose the BFGS optimizer [see e.g. Fletcher, 1987]. Second, while it is generally no longer considered as state of the art to use the Adam optimizer on all parameters including the variational ones for optimizing models involving sparse GPs [see e.g. Salimbeni et al., 2018, Adam et al., 2021], we also empirically found that this approach leads to the BFGS optimizer struggling to find good optima (or even not converging at all). We therefore opted for the following two strategies: we use the BFGS optimizer to find the mode in the inner loop and then i) the Natural gradient optimizer [Salimbeni et al., 2018] on the variational parameter ψ and the Adam optimizer on all other model parameters or ii) using a diagonal covariance for the variational distribution $q_\psi(F_M)$ and then optimizing all parameters with the Adam optimizer.

We consider the optimization for finding the mode $\hat{X}_{T_0}^{(n)}$ (see Alg. 1) as failed if (i) the maximum number of iterations has been reached without converging within the tolerance, (ii) the determinant of the Hessian [required for Eq. (14)] is negative, or iii) some of the optimal latent states $\hat{x}_t$ converge to positions far outside of the region where there are inducing points X_M (which is justified when normalizing the data as we do). In all cases we simply ignore the contributions of these terms to the optimization objective by removing the n-th term from the sum in Eq. (20). Typically this leads to the removal of well below 1% of the optimization runs.

F EXPERIMENTAL DETAILS

F.1 KINK

Training data For each of the ten different seeds, we generate a (different, noise dependent) sequence of latent states X_T according to the description in Sec. 5.1. Then we add (seed dependent) zero-mean iid. Gaussian noise with emission noise variances $\sigma_y^2 \in \{0.008, 0.08, 0.8\}$ to each of the sequences of latent states, thus generating Y_T, our training data, resulting in 30 time series in total. See also Fig. 5 for a visualization of the data.

VCDT model For the experiment in Sec. 5.1, the VCDT model from Ialongo et al. [2019] uses the same emission model, $p(y_t \mid x_t) = \mathcal{N}\left(y_t | x_t, \sigma_y^2\right)$ that is fixed to the groundtruth. The initial distribution is chosen slightly more flexible, $p(x_0) = \mathcal{N}\left(x_0 | m_0, \sigma_0\right)$, with learnable parameters m_0 and σ_0. This is important for this model as the sequential sampling in the inference requires a well tuned starting point. Another difference lies in the GP contribution to the transition function. Whereas we use a sparse GP with the FITC approximation, resulting in Eq. (11), Ialongo et al. [2019] use a proper prior on the GP and inducing outputs, resulting in the standard GP transition model $p(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t | f_{t-1}, Q\right)$, where $f_{t-1} = f(x_{t-1})$. In this experiment a zero-mean GP prior is placed on the function f, as was done in the original work. See also Appx. A for a discussion on the role of the FITC prior.

VCDT training For the training of the VCDT model we took the code provided in Ialongo et al. [2019] and contacted the authors for the full experimental details. Taking their suggestions into account and after additional fine-tuning, we fix the inducing points to 50 points in the range of $[-4, 2]$ and take an RBF kernel for the GP, where we initialize the trainable variance and lengthscale to 2. We leave all other model parameters at their standard initialization values except for the Cholesky factor of the covariance matrix of the variational distribution and the transition noise which we scale by a factor of 0.1 and 0.01, respectively. For training we set the number of samples to be drawn from the posterior to 100 and take the Adam optimizer [Kingma and Ba, 2015] using a learning rate of 10^{-3} for 10^4 iterations.

Laplace training We use the same kernel and inducing point settings as for the VCDT model. We initialize the means of the variational distribution to zero-mean Gaussian noise with standard deviation 10^{-3} and the Cholesky factor of its covariance matrix to 10^{-5} times that of the prior Cholesky factor. The transition noise variance is initialized to 0.0025. For the training we use $N = 10$ samples from the variational posterior and a minibatch size of $T_b = 30$. We optimize the parameters using a combination of the Adam optimizer [Kingma and Ba, 2015] and the Natural gradient optimizer [Salimbeni et al., 2018] provided by GPflow [Matthews et al., 2017]: The Natural gradient optimizer is used on the variational parameters with an initial learning rate of 0.004 and the Adam optimizer on all other parameters with an initial learning rate of 0.0008. We train for 7000 iterations where we scale both learning rates with a factor of 0.99 every 500 steps. For the first 1000 iterations we scale the KL-divergence in our optimization objective [Eq. (16)] by a factor which we initialize at 0 and linearly increase to 1 over these iterations. We found this to help learning the sparse GP initially, which in turn improved the optimization behavior for finding the mode required for the Laplace approximation. For finding this mode, we initialize X_{T_0} to the observations Y_T adding zero-mean Gaussian noise with standard deviation 10^{-3}. We set the parameters of the *bfgs_minimize* function provided by TensorFlow-Probability to 500 for *max_iterations* and to 10^{-8} for *tolerance*, counting runs as not converged if this (gradient norm) tolerance is not achieved within the 500 iterations.

Evaluation For the evaluation of the (train) log-likelihoods (resulting in Tab. 1) we proceed as follows: We take $J = 70$ linearly spaced points $\{x_j\}_{j=1}^J$ between the minimum and maximum latent state of the ground truth time series X_{T_0} (different for every seed but not for different σ_y^2), since this is the region in which the GP should be able to reasonably learn the ground truth. For each of the trained sparse GPs we obtain (marginal) predictions for each of these test points x_j, i.e., mean and variance predictions $\mu(x_j)$, and $\Sigma(x_j)$, respectively using Eqs. (5) and (6). These are then compared with the groundtruth, i.e., the *kink* function evaluated at these locations, $f_k(x_j)$, via the average log-likelihood LL,

$$\mathrm{LL} = \frac{1}{J} \sum_{j=1}^{J} \log \mathcal{N}\left(f_k(x_j) | \mu(x_j), \Sigma(x_j)\right).$$

F.2 SYSTEM IDENTIFICATION

Training data We prepare the data as follows: First, we normalize the data (observations and control inputs) to have zero mean and unit variance. Then we prepare the ten splits of the data: These are deterministic and depend on the total length of the time series T_{tot}, the length of the training sequence T_{train} (which we always choose to be $T_{\text{tot}}/2$) and the length of the test sequence T_{test} on which we want to predict, which we choose to be 120. For the first data split, the training sequence starts with the first data point in the time series, while for the last data split the test sequence ends with the last data point (meaning, the training sequence starts with data point $T_{\text{tot}} - T_{\text{tot}}/2 - T_{\text{test}} = T_{\text{tot}}/2 - 120$). All other 8 data splits are chosen such that the inital points of the training sequence are linearly spaced between the extreme cases described above, thus maximizing the diversity between the training tasks.

Laplace training We use an RBF kernel with automatic relevance determination and initialize its lengthscales to $\sqrt{3}$ and its variance to 0.5^2. We choose $M = 100$ trainable inducing points and initialize them with a Sobol sequence in the hypercube of dimension $d_x + d_c = 3$ centered around the origin with side lengths equal to the largest value $|y_t|$ in the (normalized) training sequence. We initialize the means of the variational distributions to zero-mean Gaussian noise with standard deviation 5×10^{-3} and choose diagonal covariance matrices which we initialize to 10^{-6} times the identity matrix. The transition noise variance Q is initialized to 0.01 and the trainable parameters of the emission model to $b = 0$ and $\Omega = 0.005$. For the training we use $N = 10$ samples from the variational posterior and a minibatch size of $T_b = 30$. As a pretraining step that we found helpful to get the sparse GP and the inner optimization to "interact nicely with each other", we train all parameters for 50 iterations with the Adam optimizer and a learning rate of 0.003. During this pretraining we scale the KL-divergence in our optimization objective [Eq. (16)] by a factor which we initialize at 0 and linearly increase to 1, while exponentially scaling the parameter Q down to 0.001. Then, we optimize all parameters using the Adam optimizer

with an initial (dataset dependent) learning rate of $0.2/T_{\text{tot}}$. We train for 20000 iterations where we scale the learning rate with a factor of 0.95 every 500 steps. For finding the mode in the inner optimization, we initialize the first dimension of X_{T_0} to the observations Y_T adding zero-mean Gaussian noise with standard deviation 10^{-3}, and we initialize the second dimension to zero-mean Gaussian noise with standard deviation 10^{-1}. We set the parameters of the *bfgs_minimize* function provided by TensorFlow-Probability to 500 for *max_iterations* and to 2×10^{-7} for *f_relative_tolerance*, counting runs as not converged if this tolerance (on the relative change of the objective value) is not achieved within the 500 iterations.

VCDT model For the VCDT model, we use the one employed in Ialongo et al. [2019], i.e., a four-dimensional latent state x_t with initial distribution $p(x_0) = \mathcal{N}(x_0|m_0, \sigma_0)$, with learnable parameters m_0 and σ_0, and the GP transition model from Eq. (7), using a fixed linear mean function (residual). The emission model is the same that we also use.

VCDT training For the training of the VCDT model we took the code provided in Ialongo et al. [2019] and, after contact with the authors, employed similar settings as in Doerr et al. [2018] for the following hyperparameters: We use an RBF kernel with automatic relevance determination and initialize its lengthscales to 2 and its variance to 0.5^2. We choose $M = 100$ trainable inducing points of dimensionality $d_x + d_c = 5$ and initialize them uniformly in the range $[-2, 2]$. We initialize the means of the variational distributions to zero-mean Gaussian noise with standard deviation 0.05 and initialize the covariance matrices to 10^{-4} times the identity matrix. The transition noise variance Q is initialized to 0.002^2, Ω to 10^{-4}, and σ_0 to 10^{-2}. For the training we use 100 latent state samples and the Adam optimizer with an initial learning rate of 0.01 running for 20000 iterations, where the learning rate is scaled by a factor of 0.98 every 500 iterations. For all other settings, we employed the default values provided in the code of Ialongo et al. [2019].

PRSSM model and training For the PRSSM model we take the exact model used in Doerr et al. [2018] and use the provided code for training, leaving the recommended settings unchanged. The only difference is the new data split in training and test sequence as explained above.

Evaluation For the evaluation all models are only provided with the test control inputs for $T_{\text{pred}} = 120$ time points after the end of the training series and with those have to predict the future observations $Y_{T_{\text{pred}}}$. This is done by getting the latent state at the end of the training sequence that then becomes x_0 for the test sequence. Starting from this, each trained model can then recursively sample future observations from the prior model [Eq. (8), with trained parameters θ]. We store the means μ_t and standard deviations σ_t over multiple recursive sampling runs at each time point $t = t_0, \ldots, t_0 + T_{\text{pred}}$, where t_0 is the initial time index of the test sequence. The way how x_0 is obtained and the recursive sampling is performed is slightly different for each model due to the different inference schemes. For PRSSM we simply follow the procedure outlined in the code. For VCDT we take the x_0 as the last state obtained in the last iteration of training, together with the sample from $q(F_M)$ that has been used to obtain those. See Ialongo et al. [2019] for more information about the inference scheme. There are multiple, 100, of such x_0 and $q(F_M)$ pairs as the inference scheme requires multiple latent state samples per iteration. The prediction coming from recursively sampling from the model with these values are then averaged over to obtain μ_t and σ_t. For our method we obtain the x_0 from the converged model as follows: We take the subsequence of Y_T of length T_b that ends at the last observation of the training sequence and obtain N samples from $q(F_M)$. For each of these samples we run the inner loop of Alg. 1 thus providing us with N optimal latent state sequences $\hat{X}_{T_0}^{(n)}$ from which we take the last state as an estimate for x_0 for the prediction task. We use the same N samples from $q(F_M)$, sample $S = 100$ times for each N from the prior model, thus providing an average over SN samples, yielding μ_t and σ_t. The log-likelihood is then evaluated as

$$\text{LL} = \frac{1}{T_{\text{test}}} \sum_{t=t_0}^{t_0+T_{\text{test}}} \log \mathcal{N}(y_t|\mu_t, \sigma_t)$$

and the RMSE as

$$\text{RMSE}^2 = \frac{1}{T_{\text{test}}} \sum_{t=t_0}^{t_0+T_{\text{test}}} (y_t - \mu_t)^2,$$

for $T_{\text{test}} \in \{30, 60, 90, 120\}$, respectively. Both these quantities are calculated on the unnormalized data, i.e., after reversing the normalization for the predictions.

G ADDITIONAL EXPERIMENTS

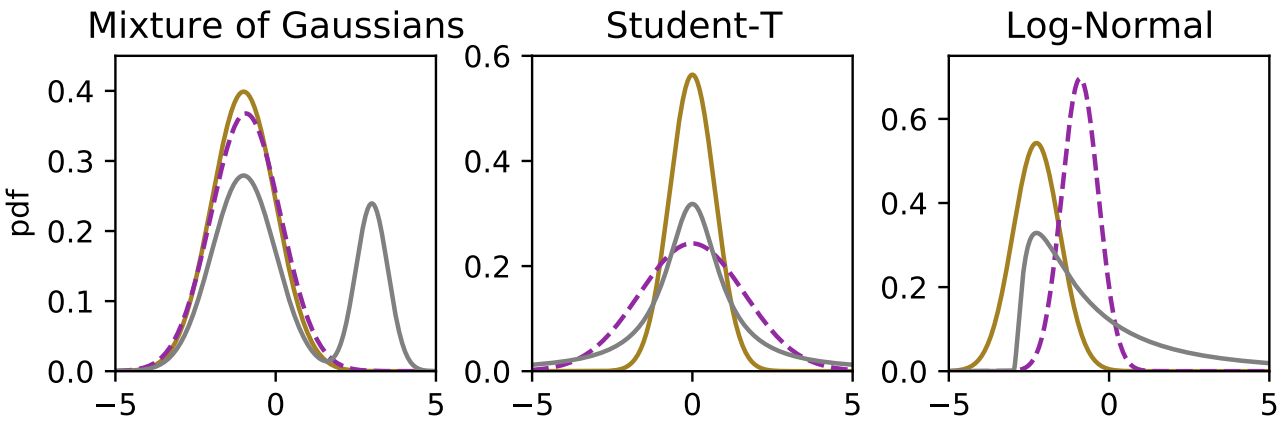

Figure 4: Qualitative difference between the Laplace approximation (brown, solid) and variational inference (magenta, dashed). The ground-truth distribution is depicted in gray. Both methods fit a Gaussian distribution to the true distribution. The Laplace approximation fits the mean to the mode of the distribution and has the same curvature as the true function at the mode. Variational inference minimizes the KL-divergence between the approximate and the true distribution. The resulting approximation is mode-seeking and avoids placing mass in regions that have zero mass under the true distribution.

Table 2: Comparison of average mean predictive log-likelihoods (higher is better) and their standard errors for ten repetitions on five different benchmark datasets. We evaluate our model (Laplace), VCDT [Ialongo et al., 2019] and PRSSM [Doerr et al., 2018] on predictions for $T \in \{30, 60, 90, 120\}$ steps in the future. See Appx. F.2 for more details and Fig. 2 for a visualization.

Dataset	Model	T=30	T=60	T=90	T=120
Actuator	Laplace	-0.6(0.2)	-0.8(0.2)	-1.7(0.7)	-1.7(0.5)
	VCDT	-0.6(0.3)	-0.9(0.3)	-1.0(0.3)	-1.0(0.2)
	PRSSM	-1.0(0.3)	-0.8(0.2)	-0.8(0.1)	-0.8(0.2)
Ballbeam	Laplace	1.6(0.4)	1.2(0.3)	0.9(0.3)	1.0(0.2)
	VCDT	2.5(0.7)	0.0(1.6)	-2.2(2.2)	-2.2(2.0)
	PRSSM	-1.8(1.4)	-1.6(0.9)	-2.7(1.4)	-2.4(1.5)
Drive	Laplace	-1.7(0.5)	-1.5(0.3)	-1.4(0.2)	-1.4(0.1)
	VCDT	-1.2(0.1)	-1.2(0.0)	-1.2(0.0)	-1.2(0.0)
	PRSSM	-4.8(1.6)	-3.9(1.0)	-3.4(0.6)	-3.3(0.5)
Dryer	Laplace	-0.3(0.1)	-0.2(0.1)	-0.3(0.1)	-0.3(0.1)
	VCDT	-3.2(0.6)	-3.2(0.5)	-2.8(0.4)	-2.9(0.4)
	PRSSM	-0.9(0.5)	-0.3(0.2)	-0.1(0.2)	-0.0(0.2)
Furnace	Laplace	-2.2(0.1)	-2.4(0.1)	-2.5(0.1)	-2.6(0.1)
	VCDT	-14.0(1.5)	-14.2(0.9)	-15.6(0.8)	-21.2(1.0)
	PRSSM	-10.1(2.9)	-8.5(2.5)	-8.4(2.6)	-11.8(3.6)

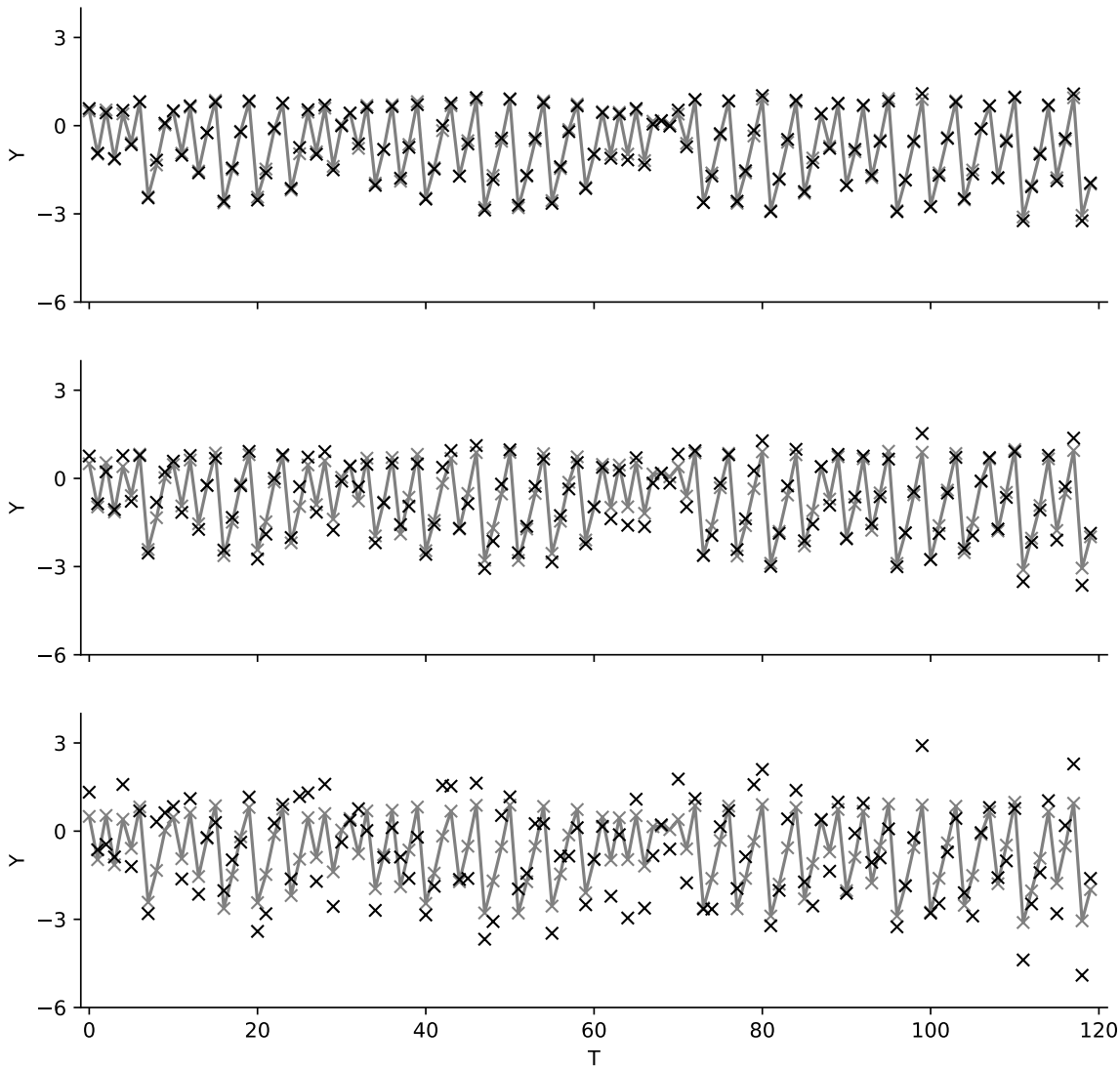

Figure 5: Visualization of the time series created by the *kink* transition function in Sec. 5.1. Shown are the latent states x_t in gray and the observations y_t in black for three different emission noise variance $\sigma_y^2 \in \{0.008, 0.08, 0.8\}$ (top to bottom). Note that the underlying latent states are the same in all plots.

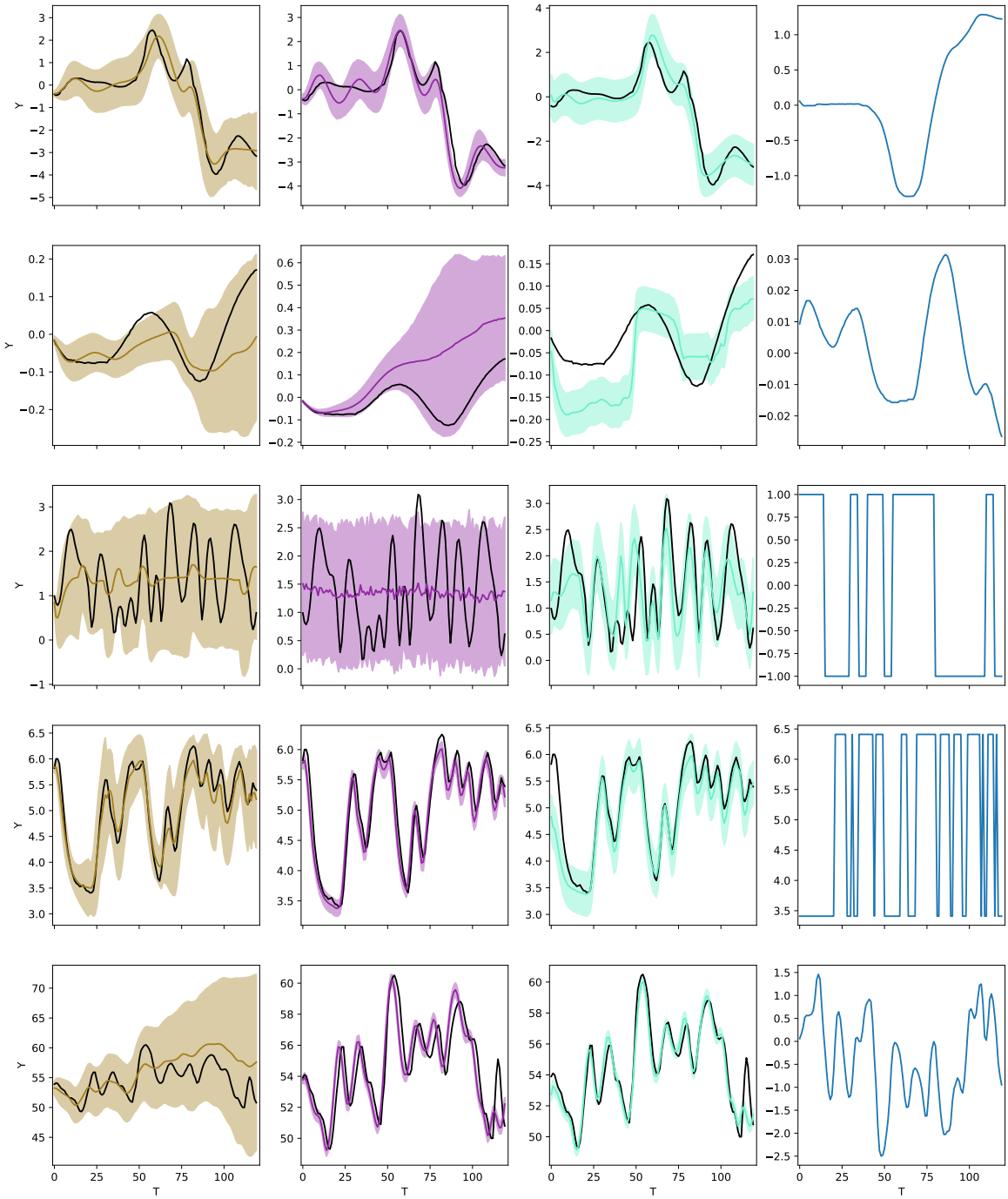

Figure 6: Exemplary figure showing the differences in predictions for the three different methods on the system identification data sets. Shown are predictions for the same seeds on the *Actuator*, *Ballbeam*, *Drive*, *Dryer*, and *Gas Furnace* datasets in the different rows (top to bottom). From left to right are the predictions for our method, VCDT [Ialongo et al., 2019], PRSSM [Doerr et al., 2018], and the control inputs (all unnormalized).

Table 3: Comparison of average RMSEs (lower is better) and their standard errors for ten repetitions on five different benchmark datasets. We evaluate our model (Laplace), VCDT [Ialongo et al., 2019] and PRSSM [Doerr et al., 2018] on predictions for $T \in \{30, 60, 90, 120\}$ steps in the future. See Appx. F.2 for more details.

DATASET	MODEL	T=30	T=60	T=90	T=120
ACTUATOR	LAPLACE	0.39(0.20)	0.42(0.18)	0.52(0.24)	0.55(0.23)
	VCDT	0.40(0.06)	0.45(0.07)	0.52(0.07)	0.54(0.07)
	PRSSM	0.53(0.09)	0.50(0.08)	0.53(0.05)	0.52(0.05)
BALLBEAM	LAPLACE	0.05(0.03)	0.07(0.05)	0.08(0.05)	0.08(0.05)
	VCDT	0.03(0.01)	0.07(0.03)	0.11(0.04)	0.14(0.04)
	PRSSM	0.06(0.01)	0.06(0.01)	0.07(0.01)	0.07(0.01)
DRIVE	LAPLACE	0.90(0.21)	0.88(0.11)	0.90(0.11)	0.94(0.20)
	VCDT	0.76(0.03)	0.77(0.01)	0.77(0.01)	0.77(0.01)
	PRSSM	0.58(0.07)	0.58(0.06)	0.56(0.04)	0.55(0.03)
DRYER	LAPLACE	0.31(0.07)	0.33(0.08)	0.39(0.24)	0.58(0.55)
	VCDT	0.22(0.01)	0.22(0.01)	0.21(0.01)	0.22(0.01)
	PRSSM	0.31(0.07)	0.26(0.06)	0.24(0.06)	0.23(0.06)
FURNACE	LAPLACE	2.15(1.05)	2.82(1.36)	3.48(1.64)	3.60(1.80)
	VCDT	1.07(0.05)	1.11(0.02)	1.15(0.01)	1.33(0.03)
	PRSSM	1.40(0.30)	1.75(0.55)	2.11(0.82)	2.34(0.86)