Scalable gradient matching based on state space Gaussian Processes

Futoshi Futami FUTOSHI.FUTAMI.UK@HCO.NTT.CO.JP
Communication Science Laboratories, NTT, KYOTO, JAPAN

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

In many scientific fields, various phenomena are modeled by ordinary differential equations (ODEs). Parameters in ODEs are generally unknown and hard to measure directly. Since analytical solutions for ODEs can rarely be obtained, statistical methods are often used to infer parameters from experimental observations. Among many existing methods, Gaussian process-based gradient matching has been explored extensively. However, the existing method cannot be scaled to a massive dataset. Given N data points, existing algorithms show $\mathcal{O}(N^3)$ computational cost. In this paper, we propose a novel algorithm using the state space reformulation of Gaussian processes. More specifically, we reformulate Gaussian process gradient matching as a special state-space model problem, then approximate its posterior distribution by a novel Rao-Blackwellization filtering, which enjoys $\mathcal{O}(N)$ computational cost. Moreover, our algorithm is expressed as closed forms, it is 1000 times more faster than existing methods measured in wall clock time.

Keywords: Gaussian process, state space model, gradient matching

1. Introduction

In scientific fields ranging from physics to biology, many natural phenomena are often described by models based on ordinary differential equations (ODEs) (Butcher, 2016). However, the parameters of ODEs are difficult to measure directly. Thus, parameters need to be estimated from observations, which are given as time-series data.

Unfortunately, many interesting phenomena are described by non-linear ODEs, which do not have closed-form solutions. This problem has traditionally been bypassed via computationally expensive numerical integration of ODEs (Tarantola, 2005). Due to numerical integration, using standard maximum likelihood estimation results in computationally expensive since every time we update parameters, we need to repeatedly integrate ODEs numerically.

To circumvent the problem of high computation costs in maximum likelihood estimation, gradient matching methods have been proposed (Calderhead et al., 2009). These methods have many variants, and we briefly explain their intuitive principle following Calderhead et al. (2009). The principle of gradient matching is composed of the *data interpolation* (DI) and *parameter adaptation* (PA) steps. In the DI step, we develop a smooth interpolation of the underlying dynamics from noisy observations based on statistical methods. In the PA step, we first compute the time derivative based on the smoothed interpolation and

compared it with the time derivative obtained from ODEs. Then, the parameters are estimated by adjusting or minimizing the mismatch between them.

Currently, Gaussian process (GP) based methods (Calderhead et al., 2009) and many extensions (Dondelinger et al., 2013; Macdonald et al., 2015; Gorbach et al., 2017; Wenk et al., 2018) is widely used because of the flexibility of GP. The key idea of GP-based methods is that when we use a time differentiable kernel, a time derivative of GP is also GP. Thus, if we construct GP prior based on observations, this also provides GP over a time derivatives of the system. Thanks to this property, the GP-based DI step provides us with a natural way to conduct the PA step. However, existing GP-based gradient matching suffers from high computation costs when we apply it to very large data.

A drawback of the existing methods comes from both the DI and PA steps. In the DI step, GP requires a massive computational cost, which is $\mathcal{O}(N^3)$ for N observations. There are many methods to reduce the computation cost, such as a sparse GP and low-rank approximation of the kernel. These methods choose typical M data points or optimize separate M inducing points and reduce the computation cost to $\mathcal{O}(M^2N)$. However, when the underlying process is highly nonlinear, it is unclear how to choose such points and how much bias and variance they will cause to the parameter estimation.

Even if we can construct smoothed interpolation, the computation cost of the inference in the PA step is also demanding. A typical inference step of GP-based gradient matching relies on Markov Chain Monte Carlo (MCMC) (Calderhead et al., 2009). In existing methods, the dimension of the posterior distribution is $\mathcal{O}(N)$, which can be high-dimensional. Thus, the mixing speed of MCMC becomes very slow, and we need a vast number of trials to obtain reasonable results. Also, we need the $\mathcal{O}(N^3)$ or $\mathcal{O}(M^2N)$ computation cost to evaluate the proposal distribution. Thus, the MCMC based PA step is difficult to be applied to a massive dataset.

Gorbach et al. (2017) proposed a scalable method in terms of the number of states in ODEs. However, their method suffers from $\mathcal{O}(N^3)$ computational cost when performing natural parameter evaluation at each step during the evidence lower bound optimization.

This lack of scalability for the number of observations in existing work is problematic when we have to treat very long observations. For example, to simulate nonlinear dynamics exactly, we use small time steps to reduce the discretization error. This often results in a vast amount of data points. Unfortunately, existing methods cannot be applied to such a problem and need heuristic sub-sampling to reduce the number of data points. To overcome these problems of the existing methods, we propose a scalable gradient matching method based on the state-space representation of GPs.

It is known that GPs can be reformulated as state-space models for time series data (Solin et al., 2016). Once we reformulate GP as a state-space model, we can take advantage of efficient algorithms for state-space models whose computation cost is $\mathcal{O}(N)$. Thus, the state space formulation can drastically reduce the computation costs. Then we derive a novel algorithm of gradient matching based on the combination of Rao-Blackwellization particle filtering. Our algorithm enjoys small computation costs with high accuracy.

2. Background

In this section, we first introduce the existing GP-based gradient matching method. Next, we briefly review the state space reformulation of GPs.

2.1. ODEs and notations

First, we introduce the notations of ODEs. Consider a K-dimensional continuous dynamical system whose dynamics is described by a set of K states, $x(t) = [x_1(t) \cdots x_K(t)]^\top$ with K set of ODEs:

$$\dot{x}_k(t) = \frac{dx(t)}{dt} = f_k(x(t), \theta_k, t), \tag{1}$$

where $\theta_k \in \mathbb{R}^d$ is a parameter vector of ODEs. We express the whole parameters as θ for simplicity. For simplicity, we assume that each state is one dimensional, $x_k(t) \in \mathbb{R}$. We are interested in non-linear dynamics, that is, f is non-linear and no closed-form solution is available. We assume that we only observe the noisy state y:

$$p_k(y_k|x_k) = N(y_k|x_k, \sigma_k), \tag{2}$$

where N denotes the standard normal distribution. When we observe ODEs at N time points, we express it as

$$y(t_n) = x(t_n) + \epsilon(t_n), \tag{3}$$

and assume that $t_1 < \cdots < t_N$ holds. For convenience, we express the observed discrete states as the matrix,

$$X = [x(t_1) \cdots x(t_N)] = [x_1 \cdots x_K]^\top \in \mathbb{R}^{K \times N}, \tag{4}$$

$$Y = [y(t_1) \cdots y(t_N)] = [y_1 \cdots y_K]^\top \in \mathbb{R}^{K \times N}, \tag{5}$$

where $x_k = [x_k(t_1) \cdots x_k(t_N)] \in \mathbb{R}^N$ denotes the k-th state observations. In the same way, we define $\dot{X}$ as the matrix of K states and N observation points. We also use $y_{:t}$ to express all the observations until time t. We also express $x_{k,n} := x_k(t_n)$, and $x_n := [x_1(t_n) \cdots x_K(t_n)]^\top$ for notational simplicity.

Our goal is to infer the state X and the parameter θ given the observations Y. Since we want to infer the uncertainty of the estimated parameters and states rather than point estimates, we will infer them in a Bayesian way. Thus, our goal is to derive the posterior distribution about (θ, X) conditioned on the observation Y. With this notation, we can express Eq.(3) as

$$p(Y|X) = \prod_k \prod_n N(y_k(t_n)|x_k(t_n), \sigma_k). \tag{6}$$

This is used as a likelihood function.

2.2. Gradient matching

To circumvent the numerical integration of ODEs explained in Section 1, Calderhead et al. (2009) proposed building GP prior on each latent state x_k, of which covariance is defined by a differentiable kernel $k_\phi(t_i, t_j)$, where ϕ is a hyperparameter of the kernel. We express the corresponding Gram matrix by C_ϕ, where $(C_\phi)_{(i,j)} = k_\phi(t_i, t_j)$. We put the Gaussian process prior on x_k with mean zero and covariance C_{ϕ_k},

$$p_k(x_k|\phi_k) = N(x_k|0, C_{\phi_k}). \tag{7}$$

Then we construct K number of GPs, each corresponds to each ODEs for x_k.

Since a GP is closed under the linear operation and the differentiation is a linear operation, the derivative of a GP is also a GP. Thus, we can derive the distribution over derivatives conditioned on the states at the observation:

$$\begin{aligned} &p(\dot{x}_k|x_k, \phi_k) = N(\dot{x}_k|D_k x_k, B_k), \\ &D_k = C'_{\phi_k} C_{\phi_k}^{-1}, \\ &B_k = C''_{\phi_k} - C'_{\phi_k} C_{\phi_k}^{-1} C'^{\top}_{\phi_k}, \end{aligned} \tag{8}$$

where

$$(C'_{\phi_k})_{i,j} = k_{\phi_k}\left(\frac{d}{dt}x(t)\Big|_{t=t_i}, x(t_j)\right) = \frac{d}{dt}k_{\phi_k}\left(x(t), x(t_j)\right)\Big|_{t=t_i}, \tag{9}$$

$$(C''_{\phi_k})_{i,j} = \frac{d}{dt}\frac{d}{dt'}k_{\phi_k}\left(x(t), x(t')\right)\Big|_{t=t_i, t'=t_j}. \tag{10}$$

Then, we incorporate the mechanism of ODE into a joint distribution. We model the relationship between the output of an ODE whose input is x and $\dot{x}$ as the Gaussian distribution,

$$p(\dot{x}_{k,t}|x_t, \theta, \gamma_k) = N(\dot{x}_{k,t}|f_k(x_t, \theta_k), \gamma_k). \tag{11}$$

Following the product of experts approach, we connect the two distributions Eqs.(8) and (11) for $\dot{x}$,

$$p(\dot{X}|X, \theta, \gamma, \phi) \propto \prod_{k,n} p(\dot{x}_k|x_k, \phi) p(\dot{x}_{k,t_n}|x_{t_n}, \theta_k, \gamma_k), \tag{12}$$

where $\dot{X}$ is defined is the same way as X. The intuition of this approach is that the resulting density only assigns high probability if both experts assign high probabilities, that means, both the ODE model and the observed data agree well (Wenk et al., 2018). We show the graphical model of this expert model in Fig 1.

Then by introducing the prior distribution $p(\theta), p(\gamma), p(\phi)$, we obtain the joint distribution for $(X, \dot{X}, \theta, \gamma, \phi)$,

$$p(X, \dot{X}, \theta, \gamma, \phi) \propto \prod_{k,n} N(x_k|0, C_{\phi_k}) N(\dot{x}_{k,t_n}|f_{\theta_k,k}(x_n), \gamma_k) N(\dot{x}_k|D_k x_k, B_k) p(\theta_k) p(\phi_k) p(\gamma_k), \tag{13}$$

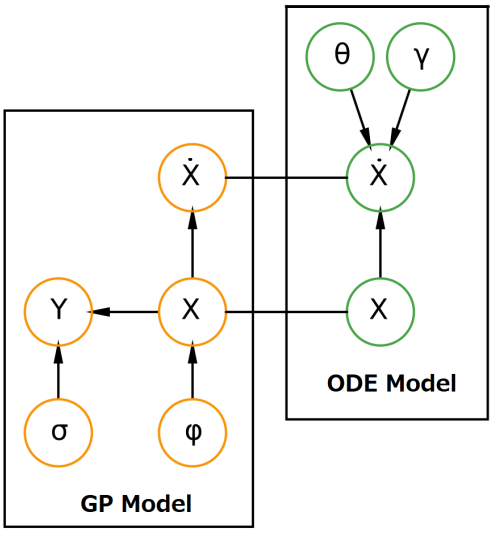

Figure 1: Graphical model of gradient matching

where we defined $f_{\theta_k,k}(x_n) := f_k(x_{t_n}, \theta_k)$ for simplicity. We can analytically integrate out about $\dot{x}$. Finally, we incorporate the likelihood function of Eq.(6). Then we obtain the joint distribution:

$$p(Y, X, \theta, \gamma, \phi, \sigma) \propto \prod_k N(y_k|x_k, \sigma_k) N(\mathbf{f}_k(X, \theta_k)|D_k x_k, B_k + \gamma_k) \times \\ N(x_k|0, C_{\phi_k}) p(\theta_k) p(\phi_k) p(\gamma_k) p(\sigma_k), \tag{14}$$

where $\mathbf{f}_k(X, \theta_k) := [f_k(x_{t_1}), \ldots, f_k(x_{t_N})]^\top \in \mathbb{R}^N$. Thus, we can obtain the posterior distribution from this joint distribution. There are two ways to obtain the posterior distribution: one is to use the sampling method such as MCMC, and the other is variational inference.

Since we simultaneously treat all the data points and dimensions of the random variables of the posterior, we call this existing GP-based gradient matching the batch GP method.

2.3. State-space Gaussian process

When we treat one-dimensional data that has a natural ordering, we can exploit that data structure. Time series data is a nice candidate for this. Surprisingly, we can reformulate one-dimensional time series GPs to corresponding state-space models (SSM) (Sarkka et al., 2013; Solin et al., 2016, 2018; Nickisch et al., 2018). A GP prior on one dimensional times data $x_k(t)$ (Here k is the k-th dimension of $x(t)$ following the notation in the previous sections) is equivalent to following SSM:

$$\tilde{\mathbf{x}}_k(t_n) \sim N(\tilde{\mathbf{x}}_k(t_n)|A_{k,n-1}\tilde{\mathbf{x}}_k(t_{n-1}), Q_{k,n-1}), \quad y_k(t_n) \sim p(y_k(t_n)|h^\top \tilde{\mathbf{x}}_k(t_n)), \tag{15}$$

where $\tilde{\mathbf{x}}_k(t_n) \in \mathbb{R}^m$ is the state of the SSM, and $A \in \mathbb{R}^{m\times m}$, $Q \in \mathbb{R}^{m\times m}$, $h \in \mathbb{R}^{m\times 1}$ are transition, noise, and measurement matrices, respectively. As we explained below, m and

these matrices are completely determined by a choice of a kernel function of the original GP.

The relation between observed data $x_k(t)$ and $\tilde{\mathbf{x}}_k(t)$ is

$$\tilde{\mathbf{x}}_k(t) = [x_k(t), \dot{x}_k(t), \ddot{x}_k(t), \dots]^\top \in \mathbb{R}^m, \tag{16}$$

and thus, $\tilde{\mathbf{x}}_k(t)$ represents high order time derivative information. Here, m, the dimension of the SSM, is determined by the choice of the kernel matrix of the original GP.

As for transition and noice matrices, when we use a stationary kernel that is, $k(t,t') = k(t-t')$, following conditions are satisfied: $A_n = \exp(F\Delta t_n)$ with $\Delta t_n = t_{n+1} - t_n$, and $Q_n = P_\infty - A_n P_\infty A_n^\top$ where P_∞ is the solution of the Lyapunov equation, that is, $\dot{P}_\infty = FP_\infty - P_\infty F^\top + LQ_cL^\top = 0$.

For simplicity, we focus on using Matérn($\nu = 3/2$) kernel:

$$k(t,t') = l^2(1 + \sqrt{3}|t-t'|/l')\exp(-\sqrt{3}|t-t'|/l'), \tag{17}$$

which is a common choice in time series GPs and we use this kernel function in this paper. Here l and l' are hyperparameters of Matérn($\nu = 3/2$) kernel. Then, the dimension of SSM is determined as $m = 2$ and moreover:

$$F = \begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix}, \quad P_\infty = \begin{bmatrix} l^2 & 0 \\ 0 & \lambda^2 l^2 \end{bmatrix}, \tag{18}$$

where $\lambda = \sqrt{3}/l'$ and , $h = [1,0]^\top$. Thus, using Matérn($\nu = 3/2$) kernel, SSM incorporates $[x_k(t), \dot{x}_k(t)]$ for its linear dynamics.

3. Proposed method

In this section, we propose a gradient matching algorithm based on the state-space reformulation of GPs.

3.1. State space gradient matching

To reduce the large computational cost of GPs in gradient matching, we propose to transform GPs in gradient matching into SSM as we had seen in Section 2.3. This reformulation enables us to use sample efficient and online setting algorithms in SSM. Since Eq.(15) is a linear state-space model and a Gaussian likelihood is used for gradient matching, we can use the linear Bayesian filter (Särkkä, 2013) to estimate the posterior distribution of the underlying dynamics. This algorithm enjoys $\mathcal{O}(N)$ computational cost, which is smaller than the computational cost of original GP, $\mathcal{O}(N^3)$.

First, we reformulate the joint likelihood of gradient matching in Eq. (14) into SSM as follows under Matérn($\nu = 3/2$) kernel:

$$p(Y, X, \dot{X}, \theta, \gamma, \phi) \propto \prod_{k,n} N([x_{k,n}, \dot{x}_{k,n}]^\top | A_{k,n-1}[x_{k,n-1}, \dot{x}_{k,n-1}]^\top, Q_{k,n-1}) \\ \times N(\dot{x}_{k,n} | f_{\theta_k,k}(x_n), \gamma_k) N(y_{k,n} | x_{k,n}, \sigma_k) p(\theta_k) p(\phi_k) p(\gamma_k) p(\sigma_k) \tag{19}$$

and $\phi = [l, l']$. We call our approach gradient matching state-space model (GMSSM).

Algorithm 1 Augment state (Naive particle filtering)

Draw S particles from the prior $p(x_0, \dot{x}_0, \theta_0)$ and set the all weights $w_0^i = 1/S$.
for $t = 1$ **to** T **do**
1)Update the particles by Eq.(20)
also $\theta_{t+1} = \theta_t + \epsilon$, ϵ is small noise.
2)Update weights by Eq.(21)
3)If the effective particle number Eq.(22) is low, perform resampling.
end for

Algorithm 2 Resampling in the particle filtering

1)Interpret the each weight $w_T^{(i)}$ as the discrete probability which we obtain $x_T^{(i)}$.
2)Draw S samples from the discrete probability distribution in step 1
3)Set all the weights as $1/S$

Next, we need to estimate the posterior distribution $p(X, \theta|Y)$ from the obtained joint distribution. For SSM, a widely used approximation is particle filtering shown in Alg. 1. However, we found that naive particle filtering has significant problems in our model (see nection 3.3 for details). To solve problems in naive particle filtering, in Section 3.4, we will introduce our filtering algorithm shown in Alg. 3.

3.2. Naive particle filtering

In this section, we briefly explain the naive particle filtering shown in Alg. 1. In particle filtering, we approximate the posterior distribution by weighted samples. For example, each particle in Eq.(19) is composed of the tuple of weights and state-space $(w_n^{(l)}, \{x_{k,n}^{(l)}, \dot{x}_{k,n}^{(l)}\}_{k=1}^K)$, which means the l-th particle for k-th dimension of the state at time t_n. Particle filtering is composed of two steps: one is updates of particles by proposal distributions and the other is updates of weights of each particle. In this work, we use the transition matrix $A_{k,n}$ for the particle updates,

$$[x_{k,n+1}^{(l)}, \dot{x}_{k,n+1}^{(l)}]^\top \sim A_{k,n}[x_{k,n}^{(l)}, \dot{x}_{k,n}^{(l)}]^\top + \epsilon, \tag{20}$$

where $\epsilon \sim N(0, Q_{k,n})$. To infer θ, a naive approach is using the augmented space method, with which we extend the state space from $(x_{k,n}^{(l)}, \dot{x}_{k,n}^{(l)})$ to $(x_{k,n}^{(l)}, \dot{x}_{k,n}^{(l)}, \theta_{k,n}^{(l)})$. Here we assume that a dynamics of θ is defined as $\theta_{k,n+1}^{(l)} = \theta_{k,n}$. In practice, we add a small Gaussian noise for regularity (Kitagawa, 1998). The benefit of this augmented method is that we can avoid the iterative filtering steps, which are required in the maximum likelihood approach, such as the EM algorithm.

We update weights $w_n^{(l)}$ in proportion to the likelihood function. In our setting, we multiply the likelihood of the observation and the ODE condition:

$$w_n^{(l)} \propto w_{n-1}^{(l)} \prod_k N(\dot{x}_{k,n}^{(l)}|f_{k,\theta_{k,n}^{(l)}}(x_n^{(l)}), \gamma_k) N(y_{k,n})|x_{k,n}^{(l)}, \sigma_k). \tag{21}$$

When weights become small, the effective particle number defined as

$$n_{\text{eff}}^{n} = \left(\sum_{l}^{S} \left(w_{n}^{(l)} \right)^{2} \right)^{-1} \tag{22}$$

become small. This leads to inefficiency. A common approach to eliminate this problem is resampling shown in Alg.2. The resampling algorithm means we draw new S samples from the discrete distribution composed from the weighted particles which are obtained by the sequential monte carlo until now.

3.3. Problems of naive particle filtering in Alg. 1

We found that Alg. 1 works poorly in our setting. The drawback comes from two points. The first drawback is the property of the augmented space for θ. Practically, Alg. 1 suffers from poor exploration in the parameter space since the dynamics of θ is a Brownian motion, which is completely random. Thus, to get reasonable performances, we need extensive hyperparameter tuning. The second drawback comes from a lack of information about $\dot{x}$ in the observation. This missing information results in too large uncertainty of the posterior distribution of parameters. To solve these problems, we develop an approximation based on variational inference and particle filtering.

3.4. Rao-Blackwellization filtering

Ideally, in Bayesian inference, problems which we discussed in Section 3.3 should be solved by marginalizing out $\dot{x}$ and θ from the joint distribution at each time steps. Then we use the distribution of x where $\dot{x}$ and θ are marginalized out for updating weights of particle x. Then estimate $\dot{x}$ and θ by using the conditinal distribution $p(\dot{x}|x)$ and $p(\theta|x)$. This ideal strategy is called Rao-Blackwellization (RB) filtering (Kantas et al., 2015). We apply such a marginalized particle filtering as we explained in the below and our algorithm is summarized in Alg. 3.

3.4.1. STEP I): INITIAL STATE CALCULATION

First, we focus on the initial state $t = 1$. Then the posterior distribution is proportional to the joint distribution given as

$$p(y_1, x_1, \dot{x}_1, \theta) \propto \prod_{k} N(\dot{x}_{k,1}|f_{\theta_k,k}(x_1), \gamma_k) N(y_{k,1}|x_{k,1}, \sigma_k) p(x_1) p(\dot{x}_1) p(\theta_k). \tag{23}$$

Here, we omit $p(\phi), p(\sigma)$ and $p(\gamma)$ for simplicity. Then we consider to approximate $x_{k,1}$ by RB filtering which needs to integrate out $\dot{x}_{k,1}$ and θ. When we assume that $p(x_1), p(\dot{x}_1)$, and $p(\theta_k)$ are the Gaussian distributions. We assume the linear condition for parameters in ODE, that is, we can write the ODEs:

$$f_{\theta_k,k}(x) := \sum_{d'_k=1}^{d} c_{d',k}(x)\theta_{d',k} + c_{0,k}(x) \tag{24}$$

and we define $\mathbf{c}_k(x) := [c_{1,k}(x), \ldots, c_{d,k}(x)]^\top$. This assumption is not strong since it is wide enough to cover many interesting nonlinear dynamics (Butcher, 2016) since nonlinear dependency on the state x is available via $\mathbf{c}_k(x)$.

Now all the distributions in the joint distribution are Gaussian and parameters in ODE are linear dependency, we can integrate out $\dot{x}_{k,1}$ and θ in $p(y_1, x_{k,1}, \dot{x}_{k,1}, \theta)$. Assume that $p(\dot{x}_{k,1}) = N(\dot{x}_{k,1}|\mu_{\dot{x}_{k,1}}, \sigma_{\dot{x}_{k,1}})$ and $p(\theta_k) = N(\theta_k|\mu_{\theta_{k,1}}, \sigma_{\theta_{k,1}})$. Then, we get the conditional posterior analytically

$$
\begin{aligned}
&p(\dot{x}_{k,1}|x_{k,1}, y_1, \theta) = N(\dot{x}_{k,1}|\mu_{\dot{x}_{k,1}}(x_1), \sigma_{\dot{x}_{k,1}}(x_1)), \\
&\mu_{\dot{x}_{k,1}}(x_1) := ((\sigma_{\dot{x}_{k,1}})^{-1} + \gamma_k^{-1})^{-1}((\sigma_{\dot{x}_{k,1}})^{-1}\mu_{\dot{x}_{k,1}} + \gamma_k^{-1} f_{k,\theta}(x_1)), \\
&\sigma_{\dot{x}_{k,1}}(x_1) := ((\sigma_{\dot{x}_{k,1}})^{-1} + \gamma_k^{-1})^{-1}),
\end{aligned}
\tag{25}
$$

and

$$
\begin{aligned}
&p(\theta_k|x_{k,1}, y_1) = N(\theta_k|\mu_{\theta_{k,1}}(x_1), \sigma_{\theta_{k,1}}(x_1)), \\
&\mu_{\theta_{k,1}}(x_1) := (\sigma_{\dot{x}_{k,1}}(x_1) + \gamma_k)^{-1}(\mu_{\dot{x}_{k,1}}(x_1) - c_{k,0}(x_1))\sigma_{\theta_{k,1}}(x_1)\mathbf{c}_k(x_1) + \sigma_{\theta_{k,1}}(x_1)\mu_{\theta_{k,1}}, \\
&\sigma_{\theta_{k,1}}(x_1) := \left(\sigma_{\theta_{k,1}}^{-1} + (\sigma_{\dot{x}_{k,1}} + \gamma_k)^{-1}\mathbf{c}_k(x_1)\mathbf{c}_k(x_1)^\top\right)^{-1}.
\end{aligned}
\tag{26}
$$

Since we cannot obtain explicite form for $p(x_{k,1}|y_1)$, we approximate it by particle filtering. First, we draw S samples from the prior distribution $p(x_1)$. We then set the weights $w_0^{(l)} = 1/S$ for all $l = 1, \ldots S$. We update weights by using the marginalized likelihood:

$$
\begin{aligned}
&w_1^{(l)} \propto L^{(1)}(x_1^{(l)}) w_0^{(l)} \prod_k N(y_{k,1}|x_{k,1}, \sigma_k), \\
&L^{(1)}(x_1) := \prod_k \frac{e^{-\frac{1}{2}(\sigma_{\dot{x}_{k,1}} + \gamma_k + \mathbf{c}_k(x_1)^\top \sigma_{\theta_{k,1}} \mathbf{c}_k(x_1))^{-1}(\mu_{\dot{x}_{k,1}} - (\mathbf{c}_k(x_1)\mu_{\theta_k} + c_{k,0}(x_1)))^2}}{\sqrt{2\pi(\sigma_{\dot{x}_{k,1}} + \gamma_k + \mathbf{c}_k(x_1)^\top \sigma_{\theta_{k,1}} \mathbf{c}_k(x_1))}}.
\end{aligned}
\tag{27}
$$

Then, we get the approximation of the posterior:

$$
q(x_1|y_1) \approx \sum_{l=1}^{S} w_1^{(l)} \delta(x_1 - x_1^{(l)}). \tag{28}
$$

Using this, we calculate posterior samples of $\dot{x}_1$ and θ by exact conditional distributions Eqs. (25), (26):

$$
\dot{x}_{k,1}^{(l)} = \mu_{\dot{x}_{k,1}}(x_1^{(l)}) + \sqrt{\sigma_{\dot{x}_{k,1}}(x_1^{(l)})}\epsilon, \quad \epsilon \sim N(0,1), \tag{29}
$$

$$
\theta_{k,1}^{(l)} = \mu_{\theta_{k,1}}(x_1^{(l)}) + \sqrt{\sigma_{\theta_{k,1}}(x_1^{(l)})}\epsilon, \quad \epsilon \sim N(0,1). \tag{30}
$$

Here, we express samples of $\theta_k^{(l)}$ as $\theta_{k,1}^{(l)}$ to express that these θ incorporate the observation at time $t = 1$. In this way, we marginalize $\dot{x}$ and θ, which does not relate to observed data y directly. As we will see in experiments, this marginalization significantly improves the naive particle filtering. We repeat a similar calculation after $t = 1$ as we explain in the next section.

3.4.2. STEP II): STATE-SPACE TRANSITION

Next, we consider at $t = n$ and our goal is to derive the posterior distribution expressed as

$$p(x_{k,n}, \dot{x}_{k,n}, \theta_k | \dot{x}_{:n-1}, x_{:n-1}, y_{:n-1}) \propto N(\dot{x}_{k,n} | f_{\theta,k}(x_n), \gamma_k) N(y_{k,n} | x_{k,n}, \sigma_k) \\ \times N([x_{k,n}, \dot{x}_{k,n}]^\top | A_{k,n-1}[x_{k,n-1}, \dot{x}_{k,n-1}]^\top, Q_{k,n-1}) q(\theta_k | x_{:n-1}, y_{:n-1}). \tag{31}$$

Here, $q(\theta | x_{:n-1}, y_{:n-1})$ is the posterior distribution of parameters until timen $t = n-1$. If $q(\theta | x_{:n-1}, y_{:n-1})$ is the Gaussian distribution, then since all the related distribution are the Gaussian distributions, we can proceed the calculation almost in the same way as we had done in Section 3.4.1. However, as calculated in Eq. (30), $q(\theta | x_{k,:n-1}, y_{:n-1})$ is an empirical distribution and not the Gaussian distribution. Thus, we approximate this as Gaussian distribution by calculating weighted mean and variance:

$$\mu_{k,n-1} = \frac{1}{w^l_{n-1}} \sum_l w^l_{n-1} \theta^{(l)}_{k,n-1}, \tag{32}$$

$$\Sigma_{k,n-1} = \frac{1}{w^l_{n-1}} \sum_l w^l_{n-1} (\theta^{(l)}_{k,n-1} - \mu_{k,n-1})^\top (\theta^{(l)}_{k,n-1} - \mu_{k,n-1}) \tag{33}$$

and set this as $q(\theta_k | x_{:n-1}, y_{:n-1}) = N(\theta_k | \mu_{k,n-1}, \Sigma_{k,n-1})$. Using the Gaussian distribution not an empirical distribution significantly improve the efficiency of the algorithm. Then, we can analytically calculate the conditional distributions in the same way as Section 3.4.1. We can proceed RB filtering in the following steps; First, we update $x^{(l)}_{k,n-1}$ to $x^{(l)}_{k,n}$ using SSM:

$$x^{(l)}_{k,n} = (A_{k,n-1})_{11} x^{(l)}_{k,n-1} + (A_{k,n-1})_{12} \dot{x}^{(l)}_{k,n-1} + \sqrt{(Q_{k,n-1})_{11}} \epsilon, \quad \epsilon \sim N(0,1). \tag{34}$$

Next, we update weights:

$$w^{(l)}_n \propto L^{(n)}(x^{(l)}_n) w^{(l)}_1 \prod_k N(y_{k,n} | x_{k,n}, \sigma_k), \tag{35}$$

$$L^{(n)}(x_n) := \prod_k \frac{e^{-\frac{1}{2}(\sigma^1_k + \gamma_k + \mathbf{c}_k(x_n)^\top \Sigma_{k,n-1} \mathbf{c}_k(x_n))^{-1} (\mu^1_k - (\mathbf{c}_k(x_n) \mu_{k,n-1} + c_{k,0}(x_n)))^2}}{\sqrt{2\pi(\sigma^1_k + \gamma_k + \mathbf{c}_k(x_n)^\top \Sigma_{\theta_k} \mathbf{c}_k(x_n))}},$$

where μ^1_k and σ^1_k is defined below. Then, we calculate the conditional distribution

$$\dot{x}^{(l)}_{k,n} = \mu_{\dot{x}_{k,n}}(x^{(l)}_n) + \sqrt{\sigma_{\dot{x}_{k,n}}(x^{(l)}_n)} \epsilon, \quad \epsilon \sim N(0,1), \tag{36}$$
$$\mu_{\dot{x}_{k,n}}(x_n) := ((\sigma^1_k)^{-1} + \gamma_k^{-1})^{-1} ((\sigma^1_k)^{-1} \mu^1_k + \gamma_k^{-1} f_{k,\theta}(x_n)),$$
$$\sigma_{\dot{x}_{k,n}}(x_n) := ((\sigma^1_k)^{-1} + \gamma_k^{-1})^{-1}),$$
$$\mu^1_k := (Q_{k,n-1})_{21} (Q_{k,n-1})^{-1}_{11} (x_{k,n} - \mu_a) + \mu_b,$$
$$\sigma^1_k := (Q_{k,n-1})_{22} - (Q_{k,n-1})_{21} (Q_{k,n-1})^{-1}_{11} (Q_{k,n-1})_{12}),$$
$$\mu_a := (A_{k,n-1})_{11} x_{k,n-1} + (A_{k,n-1})_{12} \dot{x}_{k,n-1},$$
$$\mu_b := (A_{k,n-1})_{21} x_{k,n-1} + (A_{k,n-1})_{22} \dot{x}_{k,n-1}$$

Algorithm 3 RB filtering (Proposed)

Draw S particles from the prior $p(x_1)$ and set the all weights $w_0^{(l)} = 1/S$.
Calculate $w_1^{(l)}$ by Eq. (27).
Calculate $\theta_{k,1}^{(l)}$ and $\dot{x}_{k,1}^{(l)}$ by Eq. (29), (30).
for $t = t_2$ **to** t_N **do**
1)Calculate $x_{k,n}^{(l)}$ Eq. (34).
2)Update weights by Eq. (35).
3)Calculate $\theta_{k,n}^{(l)}$ and $\dot{x}_{k,n}^{(l)}$ by Eqs (36),(37)
4)If the effective particle number Eq. (22) is low, perform resampling.
end for
(Option) Run backward smoothing

and

$$\theta_{k,n}^{(l)} = \mu_{\theta_{k,n}}(x_n^{(l)}) + \sqrt{\sigma_{\theta_{k,n}}(x_n^{(l)})}\epsilon, \quad \epsilon \sim N(0,1), \tag{37}$$
$$\mu_{\theta_{k,n}}(x_n) := (\sigma_k^1 + \gamma_k)^{-1}(\mu_k^1 - c_{k,0}(x_n))\sigma_{\theta_{k,n}}(x_n)\mathbf{c}_k(x_n) + \sigma_{\theta_{k,n}}(x_n)\mu_{k,n-1},$$
$$\sigma_{\theta_{k,n}}(x_n) := \left(\Sigma_{k,n-1}^{-1} + (\sigma_k^1 + \gamma_k)^{-1}\mathbf{c}_k(x_n)\mathbf{c}_k(x_n)^\top\right)^{-1}.$$

Now we get the approximation for $t = n$. We repeat this update until $t = N$. Furthermore, this algorithm is the forward filtering. Thus inference of the parameter of θ at an early time is not so accurate. This means that the posterior for x_t and $\dot{x}_t$ for small t strongly depends on our prior $p(x)$ and $p(\dot{x})$. Thus, if we are interested in infer x_t and $\dot{x}_t$ for small t, we should do backward filtering (smoothing). This is easily done since we reverse the calculation of the conditional distribution and update weights for x in reverse order. See Appendix A.2 for the backward filtering (smoothing). Moreover, we use the Gaussian distribution for $p(x_1)$ and use y_1 as a mean of the prior distribution. In summary, our algorithm is shown in Alg. 3.

3.5. Hyperparameter tuning

In the previous section, we did not discuss hyperparameters, ϕ, σ, and γ. As for ϕ, which is the hyperparameter of the kernel function, following Wenk et al. (2018), we estimate it by running the GP regression before the gradient matching algorithm and maximizing the marginal likelihood. In our setting, we maximize the marginal likelihood of SSM, in which the ODE condition term is not included in the joint likelihood. Thus, we can calculate the marginal likelihood analytically, which requires $\mathcal{O}(N)$ computational cost. Estimating ϕ separately from other parameters shows the state of the art performance and numerical stability than the joint optimization with gradient matching (Dondelinger et al., 2013).

As for σ and γ, if their prior distributions are the Gaussian distribution, then we can easily integrate out them from the joint distribution and get the conditional distribution in the same way as θ.

4. Numerical experiments

We compared our method with the state-of-the-art method, FGPGM (Wenk et al., 2018). FGPGM is MCMC based approach, and we discard the first 1000 samples as burn-in and stop MCMC when obtained 1000 samples. About the wall clock time of MCMC, we measured the time from the beginning of the burn-in to finish sampling. Note that the time of maximizing the marginal likelihood of GP regression is not included for both our algorithm and FGPGM. All the detailed experimental settings are shown in Appendix B.

4.1. Lotka Volterra

The fist application of our algorithm is the Lotka Volterra system (Lotka, 1932),

$$\begin{aligned} \dot{x_1} &= \theta_1 x_1 - \theta_2 x_1 x_2, \\ \dot{x_2} &= -\theta_3 x_2 + \theta_x x_1 x_2. \end{aligned} \tag{38}$$

Thus, $K = 2$, and there are 4 parameters. This is used to study predator-prey interactions and shows high non-linearlity. We follow the standard setting FGPFM (Wenk et al., 2018), that is, true parameters are set as $[\theta_1 = 2, \theta_2 = 1, \theta_3 = 4, \theta_4 = 1]$. We generate the true trajectory with the time interval 0.01 and add the observation noise with $\sigma = 0.1$ and generate 400 data points.

First, we compared our proposed method (Alg.3) with the naive particle filtering (Alg.1) to observe how marginalization works well. Here we only did forward filtering. For that purpose, We observe the effective particle numbers. We compared the effective particle number between the state-space model without ODE condition, naive particle filtering, and our proposed method. The state-space model without ODE condition means the joint likelihood does not include the ODE condition, $p(\dot{x}|f_\theta(x))$. Since the state-space model without ODE is equivalent to the original GP regression, its effective number of particles must be almost as same as the true number of particles. The result is shown in the upper row in Fig. 2 where 10000 particles are used. The state-space GP without ODE condition has a high effective number of particles. The naive particle filtering suffers from a small effective number of particles. Compared to it, our proposed method considerably improves the effective number of particles. Moreover, thanks to the marginalization, we can reduce the computational time. Next, we check the effectiveness of marginalizing out $\dot{x}$. Since marginalization reduces uncertainty, we expect that it leads to improving the performance of inference. The lower row in Fig. 2 shows this. When we use our method shown in the red line, the parameter converges to the true value. On the other hand, in naive particle filtering shown as the green line, the parameter does not converge when only using forward filtering.

Finally, we compared our algorithm with FGPGM in terms of parameter estimation performance and computational time. The results are shown in Table. 1. We used 500 particles for our algorithm, and we only did forward filtering since it is enough to estimate parameters. Moreover, to check the computational cost of MCMC, we subsampled 100 data points from the original 400 data points such that the interval of subsampled points is equally located in [0,4]. FGPGM(N=100) denotes this subsampled result, and FGPGM(N=400) means all 400 data points are used. The result shows that our method is competitive with FGPGM(N=400) but much faster. It is quite a natural result that the fully MCMC method FGPGM(N=400) shows the best performance, and it needs vast computation.

Table 1: parameter estimation (true $\boldsymbol{\theta} = [2, 1, 4, 1]$)

Method	Proposed	FGPGM(N=100)	FGPGM(N=400)
time(s)	4.2	8601	50375
θ_1	1.85 ± 0.12	2.03 ± 0.02	1.95 ± 0.09
θ_2	0.97 ± 0.05	0.99 ± 0.01	0.98 ± 0.01
θ_3	3.90 ± 0.35	3.74 ± 0.06	3.96 ± 0.03
θ_4	0.99 ± 0.09	0.92 ± 0.02	0.99 ± 0.01

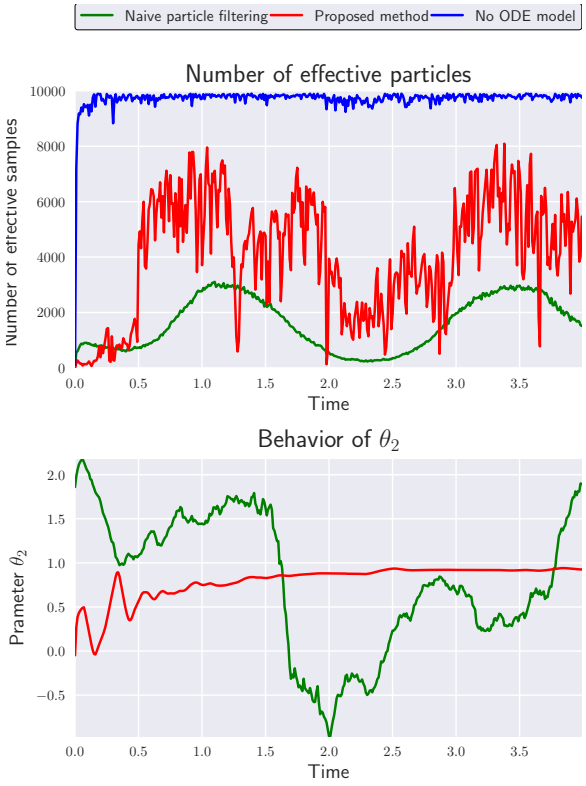

Figure 2: Effectiveness of RB technique

4.2. The Fitz-Hugh Nagumo sytem

The second example is the Fitz-Hugh Nagumo sytem:

$$\begin{aligned} \dot{x_1} &= \theta_1(x_1 - x_1^3/3 + x_2), \\ \dot{x_2} &= -\frac{1}{\theta_1}(x_1 - \theta_2 + \theta_3 x_2), \end{aligned} \tag{39}$$

which was introduced FitzHugh (1961) and Marsden et al. (1993) to model a certain cell dynamics. This is also often used as a benchmark dynamics in ODE parameters estimation framework. In our experiments, we define true parameters as $\theta_1 = 3, \theta_2 = 0.2, \theta_3 = 0.2$, $x_1 = -1, x_2 = 1$. Actually this ODE is not linear with respect to parameters. Thus, we transform the second ODE about $\dot{x}_2$ as $\dot{x}_2 = -\theta_1' x_1 + \theta_2' + \theta_3' x_2$ to apply our method.

Table 2: The performance and computational time

Method	Proposed(N=400)	FGPGM(N=100)	FGPGM(N=200)	FGPGM(N=400)
time(s)	**3.2**	1676	7717	48555
$\frac{\|\theta-\theta^*\|_2}{\|\theta^*\|_2}$	0.02 ± 0.005	0.1 ± 0.03	0.06 ± 0.03	0.04 ± 0.02

First, we compared our proposed method and FGPGM in terms of estimation performance and computational cost. The true data is generated for the time interval $t \in [0, 20]$ with the time interval $\Delta t = 0.05$ and add the observation noise with $\sigma_k = 0.1$. Thus there are $N = 400$ data points. Here we checked the parameter estimation performance by considering the L2 distance between estimated parameters and true parameters. Table. 2 shows the experimental results. In this table, we used 2000 particles for our proposed method. Same as the previous experiment, FGPGM(N=100,200) means we sub-sampled 100, 200 data points from the original 400 data points for comparisons. In our algorithm, we only performed forward filtering since using only forward filtering is enough to estimate parameter θ. Our method is extremely fast and shows competent performance with the FGPGM. About FGPGM, as we increased the number of the data points, the accuracy was improved, but we needed longer time computational time.

Next, we studied the computational cost with respect to the number of data points, the number of particles in our proposed method. We increase the data points from $N = 1000$ to $N = 10000$. We also increase the number of particles P. The result is shown in Appendix C, and we confirmed that the computational cost of our algorithm grows linear about the number of data points also linear to the number of the particles.

5. Discussion

We first discuss the relation between our proposed method and standard SSMs. In SSMs, the dynamics of the underlying process are directly modeled, and parameters, which should be estimated from data, appear in transition matrices. Then, the inference is conducted by forward filtering and backward smoothing. Then parameters are estimated by EM algorithm or augmented space methods (Särkkä, 2013). However, as we mentioned, the augmented space method suffers from computational and theoretical difficulties (Kantas et al., 2015) and the EM algorithm needs multiple filtering repetitions for the convergence.

Compared to them, our gradient matching SSM is slightly different. First, we interpolate the time series data by GP, then a true dynamics given by ODEs is incorporated into the joint likelihood in the same way as the expert model. Then ODEs are not related to SSM of GP, and thus, parameters do not appear in the transition matrix in SSM. Our proposed method estimates parameters of the dynamics by conditional distributions and particle approximation for the states. For that purpose, we integrated out about parameters θ and $\dot{x}$. This results in an efficient and computationally fast algorithm.

When focusing on the kernel function, different kernel functions result in different SSMs. Intuitively, when we increase ν of the Matérn kernel, the dimension of the state of the corresponding SSM becomes larger (Solin et al., 2016). Since each dimension of the SSM corresponds to a time derivative of the state (e.g, $[x, \dot{x}, \ddot{x}, \dots]$), we can incorporate higher-

order time derivatives in SSM by using a Matérn kernel with large ν. We can use those high-order kernels ($\nu \geq 3/2$) in our formulation. However, since we assume that the underlying ODE is a first-order ODE ($\dot{x} = f(x)$), we need to integrate out higher-order time derivative terms in SSMs. This is analytically available since all the related distributions in our SSMs are Gaussian distributions.

Compared to other gradient matching algorithms, as we discussed in Section 1, our algorithm shows linear order computational complexity about the number of the data points. Moreover, we confirmed that our algorithm is remarkably fast in wall clock time. This is achieved by the novel combination of the state space model reformulation of the Gaussian process. Moreover, we developed an algorithm by analytically marginalized out $\dot{x}$ and θ, which results in efficiency, cheap computational cost, and stability. Since our algorithm is much faster than MCMC based gradient matching methods, thus we believe that our algorithm can be used combined with MCMC based methods. For example, first, we run our algorithm, which requires a small computational cost. Then, we can use the solution of our algorithm as an initial distribution or proposal distribution of MCMC based methods.

Finally, our algorithm can be used in an online setting. However, for that purpose, we need to develop the method of determining hyperparameters of a kernel function. We believe that this is an interesting direction for future work.

6. Conclusions

In this work, we proposed a novel scalable algorithm for learning parameters in ODEs. Our algorithms enjoy $\mathcal{O}(N)$ computational cost, which is a significant reduction compared to the cost of $\mathcal{O}(N^3)$ in existing methods. We reformulate the GP-based gradient matching as the state-space model-based approach, then proposed the Rao-Blackwellization filtering to approximate the posterior distributions. Our algorithm is not only scalable to the data size but also much faster than existing methods measured in wall-clock time.

References

John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Ben Calderhead, Mark Girolami, and Neil D Lawrence. Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *Advances in neural information processing systems*, 2009.

Frank Dondelinger, Dirk Husmeier, Simon Rogers, and Maurizio Filippone. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Artificial Intelligence and Statistics*, pages 216–228, 2013.

Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

Nico S Gorbach, Stefan Bauer, and Joachim M Buhmann. Scalable variational inference for dynamical systems. In *Advances in Neural Information Processing Systems*, pages 4806–4815, 2017.

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

Genshiro Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, pages 1203–1215, 1998.

Alfred J Lotka. The growth of mixed populations: two species competing for a common food supply. *Journal of the Washington Academy of Sciences*, 22(16/17):461–469, 1932.

Benn Macdonald, Catherine Higham, and Dirk Husmeier. Controversy in mechanistic modelling with Gaussian processes. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1539–1547, 2015.

SS Antman JE Marsden, L Sirovich S Wiggins, L Glass, RV Kohn, and SS Sastry. *Interdisciplinary Applied Mathematics*, volume 3. Springer, 1993.

Hannes Nickisch, Arno Solin, and Alexander Grigorevskiy. State space Gaussian processes with non-Gaussian likelihood. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

Simo Sarkka, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

Arno Solin, James Hensman, and Richard E Turner. Infinite-horizon Gaussian processes. In *Advances in Neural Information Processing Systems 31*. 2018.

Arno Solin et al. Stochastic differential equation methods for spatio-temporal Gaussian process regression. 2016.

Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico Gorbach, Andreas Krause, and Joachim M Buhmann. Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs. *arXiv preprint arXiv:1804.04378*, 2018.

Appendix A. Derivation of the Rao-Blackwellization method

A.1. Calculation of marginalization

Here, we present the exact calculation of marginalization. In this section, we consider time $t = n$ for simplicity.

Our purpose is to eliminate the effect of the $\dot{x}$ and θ by marginalization. For that purpose, we use the Gaussian distribution property. First, from the state-space model, we have

$$\begin{aligned} &p(x_{k,n}, \dot{x}_{k,n} | x_{k,n-1}, \dot{x}_{k,n-1}) \\ &= N\left(\begin{array}{c} x \\ \dot{x} \end{array} \middle| \left(\begin{array}{c} (A_{k,n-1})_{11} x_{k,n-1} + (A_{k,n-1})_{12} \dot{x}_{k,n-1} \\ (A_{k,n-1})_{21} x_{k,n-1} + (A_{k,n-1})_{22} \dot{x}_{k,n-1} \end{array} \right), \left(\begin{array}{cc} (Q_{k,n-1})_{11} & (Q_{k,n-1})_{12} \\ (Q_{k,n-1})_{21} & (Q_{k,n-1})_{22} \end{array} \right) \right). \end{aligned} \tag{40}$$

Then, using the conditional distribution of the Gaussian distribution, we have

$$p(\dot{x}_{k,n}|x_{k,n}) = N(\dot{x}_{k,n}| \, (Q_{k,n-1})_{21} \, (Q_{k,n-1})_{11}^{-1} \, (x_{k,n} - \mu_a) + \mu_b, (Q_{k,n-1})_{22} - (Q_{k,n-1})_{21} \, (Q_{k,n-1})_{11}^{-1} \, (Q_{k,n-1})_{12})$$
$$p(x_{k,n}) = N(x_{k,n}| \, (A_{k,n-1})_{11} \, x_{k,n-1} + (A_{k,n-1})_{12} \, \dot{x}_{k,n-1}, (Q_{k,n-1})_{11}).$$

For simplicity, we express

$$p(\dot{x}_{k,n}|x_{k,n}) = N(\dot{x}_{k,n}|\mu_k^1, \sigma_k^1). \tag{41}$$

Then by using the integration formula of the Gaussian distribution, we have

$$\int d\dot{x}_{k,n} N(\dot{x}_{k,n}|f_{k,\theta}(x_n), \gamma_k) N(\dot{x}_{k,n}|\mu_k^1, \sigma_k^1) = \frac{1}{\sqrt{2\pi(\sigma_k^1 + \gamma_k)}} e^{-\frac{1}{2}(\sigma_k^1+\gamma_k)^{-1}(\mu_k^1 - f_{k,\theta}(x_n))^2}. \tag{42}$$

In conclusion, the true conditional distribution that incorporate the ODE condition and the SSM model is given as

$$p(\dot{x}_{k,n}|x_{k,n}) = N(\dot{x}_{k,n}|((\sigma_k^1)^{-1} + \gamma_k^{-1})^{-1}((\sigma_k^1)^{-1}\mu_k^1 + \gamma_k^{-1} f_{k,\theta}(x_n)), ((\sigma_k^1)^{-1} + \gamma_k^{-1})^{-1})). \tag{43}$$

Next, from the marginalization about $\dot{x}$ we have

$$p(x_{k,n}, \theta_k) \propto \frac{1}{\sqrt{2\pi(\sigma_k^1 + \gamma_k)}} e^{-\frac{1}{2}(\sigma_k^1+\gamma_k)^{-1}(\mu_k^1 - f_{k,\theta}(x_n))^2} p(\theta_k). \tag{44}$$

Then, we get

$$p(x_n, \theta) \propto \prod_k \frac{1}{\sqrt{2\pi(\sigma_k^1 + \gamma_k)}} e^{-\frac{1}{2}(\sigma_k^1+\gamma_k)^{-1}(\mu_k^1 - f_{k,\theta}(x_n))^2} p(\theta_k). \tag{45}$$

We assume the linear condition for the parameter, that is, we can write the ODEs:

$$f_{k,\theta}(x) := \sum_{d'=1}^{d} c_{k,d'}(x)\theta_{d'} + c_{k,0}(x), \tag{46}$$

and we define $\mathbf{c}_k(x) := [c_{k,1}(x), \ldots, c_{k,d}(x)]^\top$ And assume that $p(\theta_k) = N(\theta_k|\mu_{\theta_k}, \Sigma_{\theta_k})$. Then from the conditional distribution formula of the Gaussian distribution, we have

$$\begin{aligned} &p(\theta_k|x_n) = N(\theta_k|\mu_{\theta_k(x_n)}, \Sigma_{\theta_k(x_n)}), \\ &\mu_{\theta_k(x_n)} := (\sigma_k^1 + \gamma_k)^{-1}(\mu_k^1 - c_{k,0}(x_n))\Sigma_{\theta_k(x_n)}\mathbf{c}_k(x_n) + \Sigma_{\theta_k(x_n)}\mu_{\theta_k}, \\ &\Sigma_{\theta_k(x_n)} := \left(\Sigma_{\theta_k}^{-1} + (\sigma_k^1 + \gamma_k)^{-1}\mathbf{c}_k(x_n)\mathbf{c}_k(x_n)^\top\right)^{-1} \end{aligned} \tag{47}$$

and we can analytically integrate about θ,

$$\begin{aligned} &\int p(x_{k,n}, \theta_k)p(\theta_k)d\theta \\ &= \frac{e^{-\frac{1}{2}(\sigma_k^1+\gamma_k+\mathbf{c}_k(x_n)^\top\Sigma_{\theta_k}\mathbf{c}_k(x_n))^{-1}(\mu_k^1-(\mathbf{c}_k(x_n)\mu_{\theta_k}+c_{k,0}(x_n)))^2}}{\sqrt{2\pi(\sigma_k^1 + \gamma_k + \mathbf{c}_k(x_n)^\top\Sigma_{\theta_k}\mathbf{c}_k(x_n))}}. \end{aligned} \tag{48}$$

Then, we obtain the update equation for the particle filter using the above marginalized distribution:

$$L^{(n)}(x_n) := \prod_k N(y_{k,n}|x_{k,n}, \sigma_k)\frac{e^{-\frac{1}{2}(\sigma_k^1+\gamma_k+\mathbf{c}_k(x_n)^\top\Sigma_{\theta_k}\mathbf{c}_k(x_n))^{-1}(\mu_k^1-(\mathbf{c}_k(x_n)\mu_{\theta_k}+c_{k,0}(x_n)))^2}}{\sqrt{2\pi(\sigma_k^1 + \gamma_k + \mathbf{c}_k(x_n)^\top\Sigma_{\theta_k}\mathbf{c}_k(x_n))}} \tag{49}$$

For $t = 1$, since we do not have the distribution which comes from the SSM. Thus, when we marginalize our $\dot{x}$, we do not include the transition distribution. Instead, we have a prior distribution for $\dot{x}$, thus we include $p(\dot{x}_{k,1})$ into the marginalization.

A.2. Backward filtering (smoothing)

We need to calculate $p(x_{k,n-1}, \dot{x}_{k,n-1}|x_{k,n}, \dot{x}_{k,n})$ from $p(x_{k,n}, \dot{x}_{k,n}|x_{k,n-1}, \dot{x}_{k,n-1})$. This is easily done since this transition distribution is the Gaussian distribution, and its transition matrix has an inverse matrix. Thus,

$$p(x_{k,n-1}, \dot{x}_{k,n-1}|x_{k,n}, \dot{x}_{k,n}) = N(x_{k,n-1}, \dot{x}_{k,n-1}|A_{k,n-1}^{-1}[x_{k,n}, \dot{x}_{k,n}]^\top, A_{k,n-1}Q_{k,n-1}A_{k,n-1}^\top). \tag{50}$$

Then we replace the mean and variance of $p(x_{k,n}, \dot{x}_{k,n}|x_{k,n-1}, \dot{x}_{k,n-1})$ in Section A.1 with this new conditional distribution. Then we can get the backward filtering (smoothing).

Appendix B. Experimental settings

In this section, we additionally describe experimental settings.

If the dynamics is stationary, it is recommended to use a prior distribution of $P(x_1, \dot{x}_1)$, which is specified by the Matérn kernel, that is $P(x_1, \dot{x}_1) \sim N([x_1, \dot{x}_1]^\top|0, P_\infty)$. However, since ODE is not the stationary dynamics, we simply use $p(x_1) = N(x_1|y_1|I)$ and we use the marginal distribution of $N([x_1, \dot{x}_1]^\top|0, P_\infty)$ for the prior of $\dot{x}$.

Fig.(3) is the example of the fitting after backward filtering. The left side is by the baseline naive particle filtering Alg.1, and the right side is our proposed filtering. We found that naive particle filtering suffers from high bias which comes from the significantly low effective number of particles in the algorithm.

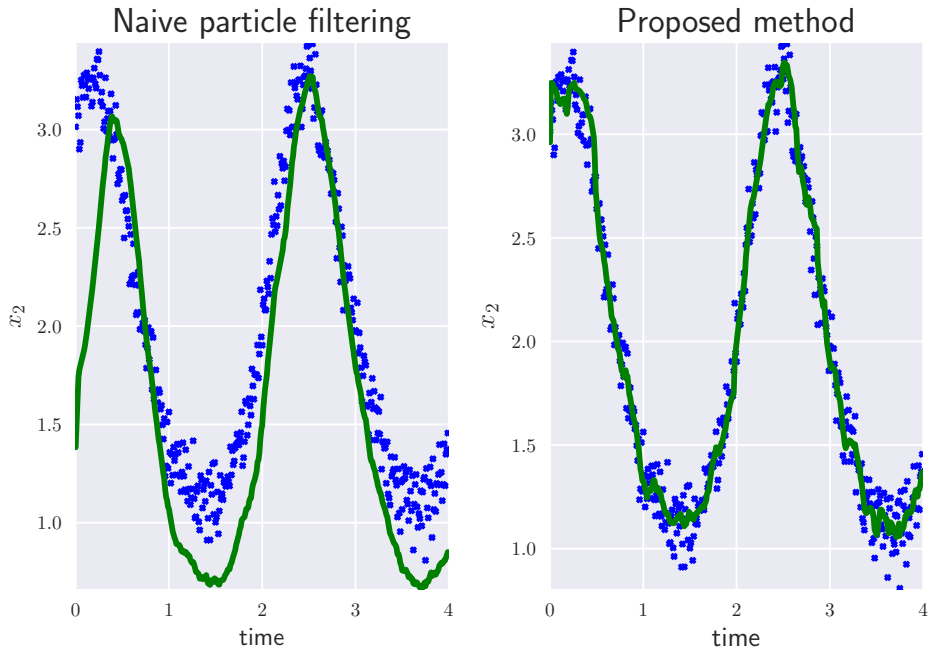

Figure 3: Comparison of the interpolation

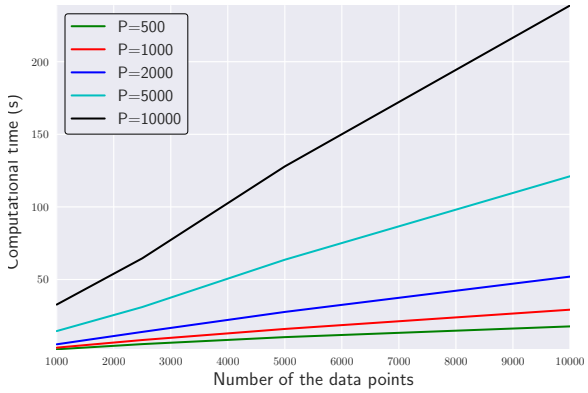

Figure 4: Computational time

Appendix C. Additional experiment

The data generation setting is the same as Section 4.2 in the main paper except for Δt in $t \in [0, 20]$. We decrease the interval time from $\Delta t = 0.02$ to $\Delta t = 0.002$. This means that the number of data points increases from $N = 1000$ to $N = 10000$. Even $N = 1000$, it is almost impossible to finish inference using FGPGM. We measured the computation time to finish our algorithm for forward filtering in our Algorithm. The result is shown in Fig.4. In the figure, P means how many particles we use in our algorithm. As we had expected, the computational cost grows linear about the number of data points also linear to the number of the particles. Thus, we confirmed that our algorithm is scalable to the number of data points and much faster than the standard existing method.