Adaptive Low-Pass Filtering using Sliding Window Gaussian Processes

Alejandro J. Ordóñez-Conejo[1*], Armin Lederer[2*] and Sandra Hirche[2]

Abstract— **When signals are measured through physical sensors, they are perturbed by noise. To reduce noise, low-pass filters are commonly employed in order to attenuate high frequency components in the incoming signal, regardless if they come from noise or the actual signal. Therefore, low-pass filters must be carefully tuned in order to avoid significant deterioration of the signal. This tuning requires prior knowledge about the signal, which is often not available in real-world applications. In order to overcome this limitation, we propose an adaptive low-pass filter based on Gaussian process regression. By considering a constant window of previous observations, updates and predictions fast enough for real-world filtering applications can be realized. Moreover, the online optimization of hyperparameters leads to an adaptation of the low-pass behavior, such that no prior tuning is necessary. We show that the estimation error of the proposed method is uniformly bounded, and demonstrate the flexibility and efficiency of the approach in several simulations.**

I. INTRODUCTION

Control systems interact with the physical world by providing signals to actuators based on measurements received from the plant. Since the physical quantities are measured using sensors, they are perturbed by noise resulting from various sources such as limited resolution of sensors, electromagnetic interference from the environment and quantization in digital to analog conversion. In order to reduce this measurement noise in signals, various forms of filters can be found in almost all real-world control systems.

These filters must trade-off noise attenuation and signal deterioration, since both goals cannot be achieved simultaneously [1]. As physical systems often operate at comparatively low frequencies, this trade-off is usually achieved using low-pass filters, which strongly attenuate input signals above a cutoff frequency. They can be realized in digital systems based on linear system theory, which leads to approaches with infinite impulse response, e.g., Butterworth or Chebyshev filters, or with finite impulse response, e.g., moving average and Savitzky-Golay filters. As these filters attenuate input signals above the cutoff frequency regardless of their source, i.e., both the actual signal and the noise, they must be carefully tuned to avoid a signal deterioration. However, this requires prior knowledge about the signal and the noise distribution, which is often not available, in particular in learning systems. For example, when applying model-free reinforcement learning algorithms to obtain optimal control laws for physical human-robot interaction, no system model of the human behavior is known, such that little information about system trajectories is available in advance.

When data can be processed off-line, such a lack of information has been overcome in many control applications by inferring models with Gaussian process regression [2], which is a non-parametric supervised machine learning technique. Due to its Bayesian background, Gaussian processes naturally allow for an implicit bias-variance trade-off, and can efficiently learn from sparse data. Moreover, Gaussian processes also provide a measure for their predictive uncertainty, such that information about the reliability of Gaussian process models is available. This allows the derivation of prediction error bounds, which are based either on the theory of reproducing kernel Hilbert spaces [3] or on Bayes theory [4]. These advantageous properties have led to a wide spread usage of Gaussian processes in control, see, e.g., [5].

Despite their common application in control, Gaussian processes have been rarely considered in filtering problems so far. Probably most notable is their application as forward model in Kalman filters [6], [7], but they have also been employed for attenuating noise in measured signals [8], [9]. In the latter example, a Gaussian process is fitted to obtain a mapping between the measurement time and the signal state, which is exploited in a secondary step to identify the parameters of the signal generating differential equation using gradient matching. Thereby, the parameters of the differential equation can be obtained in a data efficient way.

While such smoothing approaches are computationally efficient for parameter estimation, they are far too slow to be directly applicable in filtering problems with continuous streams of data due to the cubic complexity of Gaussian process regression. This limitation of Gaussian processes has been addressed by many authors recently. For example, the approaches in [10], [11] aim to construct global approximations, which achieve update and prediction rates suitable for learning in control loops. However, this comes at the price of high memory requirements or significantly weaker theoretical guarantees. A different approach for enabling online learning with Gaussian processes relies on the restriction to a training set with constant number of samples, which gets updated after observing new samples [12]. By simply removing the oldest data in the set, the updates can be efficiently realized with constant complexity [13]. Moreover, these methods have been demonstrated to be able to continuously

This work was supported by the European Research Council (ERC) Consolidator Grant "Safe data-driven control for human-centric systems (CO-MAN)" under grant agreement number 864686. A. L. gratefully acknowledges financial support from the German Academic Scholarship Foundation. Published under © 2022 IEEE with DOI 10.23919/ECC55457.2022.9838285.

[1]Alejandro J. Ordóñez-Conejo is with Costa Rica Institute of Technology, 30101, Cartago, Costa Rica. ajoseoc@gmail.com

[2]Armin Lederer and Sandra Hirche are with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany. [armin.lederer, hirche]@tum.de

[*]These authors contributed equally.

adapt to unknown system dynamics in experiments [14].

Due to these promising results, we follow the idea of employing varying data sets with constant sizes for updating Gaussian processes online. Based on this idea, we develop an efficient algorithm called sliding window Gaussian process, which achieves high update and prediction rates. By employing this algorithm to learn a map between measurement times and signal values, it can be straightforwardly applied for filtering arbitrary signals. Using a novel approach based on identities from linear algebra, which does not rely on reproducing kernel Hilbert space theory and Bayesian properties, we prove that the filtered signal exhibits a bounded error under weak assumptions. The low-pass behavior of the proposed filter is investigated in simulations, in which we demonstrate its capability of adapting to input signals using the online optimization of hyperparameters of the Gaussian process. Finally, a comparison with commonly used filters clearly demonstrates an advantageous signal recovery over a wide range of frequencies.

The remainder of this work is structured as follows. In Section II, the considered problem is formally described. The proposed low-pass filter using sliding window Gaussian processes is introduced and proven to provide bounded estimation errors in Section III. In Section IV, the low-pass behavior of the sliding window Gaussian process is compared to commonly used filters. Section V concludes this paper.

II. Problem Description

We consider a nonlinear system[1]

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^d$ denotes the state, $\boldsymbol{u} \in \mathbb{U} \subseteq \mathbb{R}^p$ is the control input and $\boldsymbol{f} : \mathbb{X} \times \mathbb{U} \times \mathbb{R}_{0,+} \to \mathbb{R}^d$ is the nonlinear dynamic. In order to exclude systems, whose solutions have finite escape times, we make the following assumption.

Assumption 1: The function $\boldsymbol{f}(\cdot,\cdot,\cdot)$ is bounded for all $\boldsymbol{x} \in \mathbb{X}$, $\boldsymbol{u} \in \mathbb{U}$, $t \in \mathbb{R}_{0,+}$.

This assumption is satisfied, e.g., for time-invariant, continuous functions $f(\cdot,\cdot,\cdot)$ on compact domains $\mathbb{X}$ and $\mathbb{U}$, but also holds for many frequently found systems in practice such as, e.g., Euler-Lagrange systems with bounded control inputs. Therefore, it is not restrictive in general.

Since we cannot continuously measure the true state in real-world systems, we assume access to noisy measurements

$$\boldsymbol{y}^{(n)} = \boldsymbol{x}(t^{(n)}) + \boldsymbol{\epsilon}^{(n)} \tag{2}$$

at times $t^{(n)} \in \mathbb{R}_{0,+}$, $n \in \mathbb{N}$, where $\boldsymbol{y}^{(n)}$ is the measured state at time $t^{(n)}$ disturbed by some noise $\boldsymbol{\epsilon}^{(n)} \in \mathbb{W} \subseteq \mathbb{R}^d$. For obtaining these samples, we consider a time-triggered scheme as formalized in the following.

Assumption 2: Data is sampled at regularly spaced time instances $t^{(n)} = n\tau$ with sampling time $\tau \in \mathbb{R}_+$ and aggregated into a data set $\{(t^{(n)}, \boldsymbol{y}^{(n)})\}_{n=0}^{\lfloor \frac{t}{\tau} \rfloor}$.

[1]Notation: Lower/upper case bold symbols denote vectors/matrices, $\mathbb{R}_+$/$\mathbb{N}_+$ all real/integer positive numbers, $\boldsymbol{I}_n$ the $n \times n$ identity matrix, $\boldsymbol{1}_n = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T \in \mathbb{R}^n$ the unit vector, $\|\cdot\|$ the Euclidean norm, and $\lfloor \cdot \rfloor$ the floor operator. Scalar operators applied to vectors and matrices denote an element-wise operation.

Time-triggered sampling is probably the most common approach when designing control methods for continuous-time systems. Hence, this assumption is not restrictive in practice.

Based on the obtained measurements aggregated in the data set, we consider the problem of designing a filter for noise attenuation, which provides an estimate $\hat{\boldsymbol{x}}(t)$ of the noise free state $\boldsymbol{x}(t)$. While we cannot expect to achieve identity between $\hat{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$, we want to guarantee a small estimation error as defined in the following.

Definition 1: The estimation error between the filtered state $\hat{\boldsymbol{x}}(t)$ and the noise free state $\boldsymbol{x}(t)$ is *uniformly bounded* by a function $c : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$ if it satisfies

$$\|\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)\| \leq c(t) \qquad \forall t \in \mathbb{R}_{0,+}. \tag{3}$$

The goal is to derive a filter with guaranteed uniform error bound $c(\cdot)$, without posing additional assumptions on the dynamics $\boldsymbol{f}(\cdot,\cdot,\cdot)$ or the noise distribution. In order to achieve this goal, we derive a non-parametric online learning method based on Gaussian process regression and employ it for filtering the noisy data stream as explained in the following.

Remark 1: Full state measurements of $\boldsymbol{x}$ are only assumed for notational simplicity. We can equivalently consider the problem of noise attenuation for output measurements $\boldsymbol{y}^{(n)} = \boldsymbol{C}\boldsymbol{x}(t^{(n)}) + \boldsymbol{\epsilon}^{(n)}$, where $\boldsymbol{C} \in \mathbb{R}^{q \times d}$, $q \in \mathbb{N}$.

III. Adaptive Noise Attenuation with Sliding Window Gaussian Processes

Since the proposed adaptive filter, for noise attenuation with guaranteed estimation errors, is based on Gaussian process regression, we briefly explain this non-parametric supervised machine learning method in Section III-A. In order to make Gaussian processes applicable in filtering problems, in Section III-B, we propose a fast approximation for online learning, which relies only on the most recent data. Finally, we apply this method as an adaptive low-pass filter and derive novel estimation error bounds in Section III-C.

A. Gaussian Process Regression

Gaussian process (GP) regression is a supervised machine learning method, which can be used to infer scalar functions $g : \mathbb{R}^q \to \mathbb{R}$ with inputs $\boldsymbol{z} \in \mathbb{R}^q$ [2]. For example, in trajectory smoothing applications it is used to learn the relationship between time t and a component x_i, $i = 1, \dots, d$ of the state, such that $z = t$ and $g(z) = x_i(t)$ [9]. Since multi-target regression can directly be achieved by individually employing GP regression to each target dimension, we consider scalar systems (1) with $d = 1$ in the sequel without loss of generality. Note that the extension to multiple targets with known correlation structure is also straightforward [15].

Formally, GPs are an infinite collection of random variables that follow joint Gaussian distributions [2], defined by a prior mean function $m : \mathbb{R}^q \to \mathbb{R}$ and a kernel $k : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}_{0,+}$. The prior mean $m(\cdot)$ can be used to include approximate, parametric models in the regression. Since we do not assume any knowledge of this kind, we set $m(\boldsymbol{z}) = 0$ for all $\boldsymbol{z} \in \mathbb{R}^q$ subsequently without loss of generality. The covariance function $k(\cdot,\cdot)$ is used to encode abstract prior knowledge such as smoothness and periodicity. When no such information is known in advance, the probably most frequently used covariance function is the squared exponential kernel

$$k(\boldsymbol{z},\boldsymbol{z}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{q}\frac{(z_i - z_i')^2}{2l_i^2}\right), \tag{4}$$

where $\sigma_f \in \mathbb{R}_+$ denotes the signal standard deviation and $\boldsymbol{l} = \begin{bmatrix} l_1 & \cdots & l_q \end{bmatrix}^T$ are the length scales. These parameters form the hyperparameter vector $\boldsymbol{\theta} = \begin{bmatrix} \sigma_f & \boldsymbol{l}^T & \sigma_{\text{on}} \end{bmatrix}$ together with the assumed standard deviation σ_{on} of the observation noise $\boldsymbol{\epsilon}$. Given a data set $\mathbb{D}_N = \{(\boldsymbol{z}^{(n)}, y^{(n)})\}_{n=1}^N$ with scalar measurements $y^{(n)} = g(\boldsymbol{z}) + \epsilon$, the hyperparameters are typically determined by minimizing the negative log-likelihood

$$\begin{aligned} -\log p(\boldsymbol{y}_N|\boldsymbol{Z}_N,\boldsymbol{\theta}) = &\frac{1}{2}\boldsymbol{y}_N^T(\boldsymbol{K}_N + \sigma_{\text{on}}^2\boldsymbol{I}_N)^{-1}\boldsymbol{y}_N \\ &+\frac{1}{2}\log(\det(\boldsymbol{K}_N + \sigma_{\text{on}}^2\boldsymbol{I}_N)) + \frac{n}{2}\log(2\pi), \end{aligned} \tag{5}$$

where $\boldsymbol{Z}_N = [\boldsymbol{z}^{(1)} \ \cdots \ \boldsymbol{z}^{(N)}]$, $\boldsymbol{y}_N = [y^{(1)} \ \cdots \ y^{(N)}]^T$ are the concatenated training samples, and $\boldsymbol{K}_N$ is the covariance matrix, whose elements are defined via $k_{N,i,j} = k(\boldsymbol{z}^{(i)}, \boldsymbol{z}^{(j)})$, $i,j = 1,\dots,N$.

After the hyperparameters have been determined, the posterior distribution at an arbitrary test point $\boldsymbol{z}$ can be analytically computed under the assumption of Gaussian distributed noise $\boldsymbol{\epsilon}$. This posterior is again a Gaussian distribution with mean and variance defined as

$$\mu_N(\boldsymbol{z}) = \boldsymbol{k}_N^T(\boldsymbol{z})\boldsymbol{\alpha}_N \tag{6}$$
$$\sigma_N^2(\boldsymbol{z}) = k(\boldsymbol{z},\boldsymbol{z}) - \boldsymbol{k}_N^T(\boldsymbol{z})\boldsymbol{A}_N^{-1}\boldsymbol{k}_N(\boldsymbol{z}), \tag{7}$$

where

$$\boldsymbol{A}_N = \boldsymbol{K}_N + \sigma_{\text{on}}^2\boldsymbol{I}_N \tag{8}$$
$$\boldsymbol{\alpha}_N = \boldsymbol{A}_N^{-1}\boldsymbol{y}_N \tag{9}$$

and the kernel vector $\boldsymbol{k}_N(\boldsymbol{z})$ is defined elementwise via $k_{N,i}(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}^{(i)})$, $i = 1,\dots,N$.

B. Online Learning with Sliding Window Gaussian Process

Even though exact Gaussian process regression allows closed-form expressions for the posterior mean and variance functions, it crucially suffers from a cubically growing computational complexity in the number of training samples N. In order to overcome this limitation, we propose sliding window Gaussian processes (SW-GP) as outlined in Alg. 1, which rely on the idea of handling continuous data streams by employing only the most recent training samples for training a GP. In particular, after observing N training samples, we restrict the training data of a GP to the subset

$$\mathbb{W}^{(N)} = \{(\boldsymbol{z}^{(n)}, y^{(n)})\}_{n=N+1-\bar{N}}^{N}, \tag{10}$$

where $\bar{N} \in \mathbb{N}_+$ is a constant determining the number of samples in the set $\mathbb{W}^{(N)}$. By using these windows of data $\mathbb{W}^{(N)}$ instead of the full data set $\mathbb{D}_N$, the computation of the posterior mean and variance using

$$\mu_{\mathbb{W}^{(N)}}(\boldsymbol{z}) = \boldsymbol{k}_{\mathbb{W}^{(N)}}^T(\boldsymbol{z})\boldsymbol{\alpha}_{\mathbb{W}^{(N)}} \tag{11}$$
$$\sigma_{\mathbb{W}^{(N)}}^2(\boldsymbol{z}) = k(\boldsymbol{z},\boldsymbol{z}) - \boldsymbol{k}_{\mathbb{W}^{(N)}}^T(\boldsymbol{z})\boldsymbol{A}_{\mathbb{W}^{(N)}}^{-1}\boldsymbol{k}_{\mathbb{W}^{(N)}}(\boldsymbol{z}) \tag{12}$$

has a constant computational complexity, where we use indexes $\mathbb{W}^{(N)}$ to emphasize that $\mathbb{W}^{(N)}$ is used for computation instead of $\mathbb{D}_N$, e.g., $\boldsymbol{k}_{\mathbb{W}^{(N)}}(\boldsymbol{z})$ is a vector with elements $k_{\mathbb{W}^{(N)},i}(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}^{(N-\bar{N}+i)})$, $i = 1,\dots \bar{N}$. Since (11) and (12) reduce to simple scalar products and matrix-vector multiplications given $\boldsymbol{A}_{\mathbb{W}^{(N)}}^{-1}$ and $\boldsymbol{\alpha}_{\mathbb{W}^{(N)}}$, they allow a fast computation of predictions in practice.

Algorithm 1: Iterative learning with SW-GPs

```
1 Function update($\mathbb{W}^{(N)}$, $\boldsymbol{\theta}^{(N)}$, $\boldsymbol{\Delta}^{(N)}$, $\boldsymbol{\nabla}L^{(N)}$, $\boldsymbol{z}^{(N+1)}$, $y^{(N+1)}$):
2     $\mathbb{W}^{(N+1)} \leftarrow \mathbb{W}^{(N)} \cup \{(\boldsymbol{z}^{(N+1)}, y^{(N+1)})\}$
3     if $|\mathbb{W}^{(N+1)}| > \bar{N}$ then
4         $\mathbb{W}^{(N+1)} \leftarrow \mathbb{W}^{(N+1)} \setminus \{(\boldsymbol{z}^{(N+1-\bar{N})}, y^{(N+1-\bar{N})})\}$
5     $\boldsymbol{\theta}^{(N+1)} \leftarrow \boldsymbol{\theta}^{(N)} + \mathrm{sign}(\boldsymbol{\nabla}L^{(N)})\boldsymbol{\Delta}^{(N)}$
6     $\boldsymbol{A}_{\mathbb{W}^{(N+1)}}, \boldsymbol{\alpha}_{\mathbb{W}^{(N+1)}} \leftarrow$ getA($\mathbb{W}^{(N+1)}$, $\boldsymbol{\theta}^{(N+1)}$)   % (8), (9)
7     $\boldsymbol{\nabla}L^{(N+1)} \leftarrow$ gradient($\boldsymbol{A}_{\mathbb{W}^{(N+1)}}$, $\boldsymbol{\alpha}_{\mathbb{W}^{(N+1)}}$, $\boldsymbol{\theta}^{(N)}$) % (15)
8     $\boldsymbol{\Delta}^{(N+1)} \leftarrow$ stepUpdate($\boldsymbol{\nabla}L^{(N+1)}$, $\boldsymbol{\nabla}L^{(N)}$, $\boldsymbol{\Delta}^{(N)}$)% (16)
9 return $\mathbb{W}^{(N+1)}$, $\boldsymbol{\theta}^{(N+1)}$, $\boldsymbol{\Delta}^{(N+1)}$, $\boldsymbol{\nabla}L^{(N+1)}$, $\boldsymbol{\alpha}_{\mathbb{W}^{(N+1)}}$, $\boldsymbol{A}_{\mathbb{W}^{(N+1)}}^{-1}$
```

Therefore, we focus on the online adaptation of the hyperparameters $\boldsymbol{\theta}^{(N)}$ to new training pairs $(\boldsymbol{z}^{(N+1)}, y^{(N+1)})$ in the following, which we realize by minimizing the negative log-likelihood (5), i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ -\log p(\boldsymbol{y}_{\mathbb{W}^{(N)}}|\boldsymbol{Z}_{\mathbb{W}^{(N)}}, \boldsymbol{\theta}). \tag{13}$$

Our numerical optimization scheme employed for this purpose is based on RPROP [16], which has been demonstrated to outperform other elaborate methods while maintaining relatively low computational complexity [17]. In contrast to existing work, we merely perform one update iteration for each data set $\mathbb{W}^{(N)}$ to limit the number of computations executed for each new data pair. This results in the adaptation rule

$$\boldsymbol{\theta}^{(N+1)} = \boldsymbol{\theta}^{(N)} + \mathrm{sign}(\boldsymbol{\nabla}L^{(N)})\boldsymbol{\Delta}^{(N)}, \tag{14}$$

where the gradient $\boldsymbol{\nabla}L^{(N)}$ of the negative log-likelihood (5) can be efficiently computed using

$$\boldsymbol{\nabla}_i L^{(N)} = -\frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}_{\mathbb{W}^{(N)}}\boldsymbol{\alpha}_{\mathbb{W}^{(N)}}^T - \boldsymbol{A}_{\mathbb{W}^{(N)}}^{-1})\frac{\partial \boldsymbol{A}_{\mathbb{W}^{(N)}}}{\partial \theta_i}\right) \tag{15}$$

for $\frac{\partial \boldsymbol{A}_{\mathbb{W}^{(N)}}}{\partial \theta_i}$ determined using the partial derivatives of the kernel $k(\cdot,\cdot)$ due to (8). The step size is defined recursively for each dimension $i = 1,\dots,d$ through

$$\Delta_i^{(N+1)} = \begin{cases} \eta^+\Delta_i^{(N)}, & (\boldsymbol{\nabla}_i L^{(N+1)})(\boldsymbol{\nabla}_i L^{(N)}) > 0 \\ \eta^-\Delta_i^{(N)}, & (\boldsymbol{\nabla}_i L^{(N+1)})(\boldsymbol{\nabla}_i L^{(N)}) < 0 \\ \Delta_i^{(N)}, & (\boldsymbol{\nabla}_i L^{(N+1)})(\boldsymbol{\nabla}_i L^{(N)}) = 0 \end{cases} \tag{16}$$

for the update factors $\eta^+, \eta^- \in \mathbb{R}^+$ with $\eta^+ > 1$ and $\eta^- < 1$ [16]. The updates of the step size $\boldsymbol{\Delta}$ follow the simple idea to take larger steps when the descent direction does not change between two iterations as indicated by a positive value of $(\boldsymbol{\nabla}_i L^{(N+1)})(\boldsymbol{\nabla}_i L^{(N)})$. When this product becomes negative, the sign of the derivative has changed between iterations, such that the optimization went past a local minimum. Therefore, the step size $\boldsymbol{\Delta}$ is reduced in this case, leading to simple and intuitive updates of the hyperparameters.

Remark 2: We employ the gradient $\boldsymbol{\nabla}_i L^{(N)}$ from the previous iteration for computing the hyperparameter update (14), which does not depend on the new data pair $(\boldsymbol{z}^{(N+1)}, y^{(N+1)})$. Therefore, we can re-use the values from the previous update iteration, such that we merely have to compute $\boldsymbol{A}_{\mathbb{W}^{(N)}}$ and $\boldsymbol{\alpha}_{\mathbb{W}^{(N)}}$ once per update iteration, which significantly reduces computation time.

C. Low-Pass Filtering with Gaussian Processes

In order to apply the presented sliding window Gaussian process approach for filtering a signal $\boldsymbol{y}^{(N)}$, we first note that the time-triggered sampling as introduced in Assumption 2 establishes the relationship

$$N(t) = \left\lfloor \frac{t}{\tau} \right\rfloor + 1, \tag{17}$$

such that we obtain a data window $\mathbb{W}^{(N(t))} = \{(z^{(n)} = t^{(n)}, y^{(n)} = x(t^{(n)}) + \epsilon^{(n)})\}_{n=N(t)+1-\bar{N}}^{N(t)}$, which changes over time. Given this set, we can compute a SW-GP mean function $\mu_{\mathbb{W}^{(N(t))}}(t)$ as defined in (11). For notational simplicity, we use indices $\bar{N}$ to denote the values computed using $\mathbb{W}^{(N(t))}$ in the following, e.g., $\mu_{\bar{N}}(t) = \mu_{\mathbb{W}^{(N(t))}}(t)$. Using this notation, we obtain the filtered signal via

$$\hat{x}(t) = \mu_{\bar{N}}(t). \tag{18}$$

For showing that this filter guarantees a uniformly bounded estimation error, we first derive a closed-form expression for the regression error, which allows an intuitive separation of error sources. For this purpose, we define the subsets of $\mathbb{W}^{(N(t))}$ containing only the first i training pairs as $\mathbb{W}_i^{(N(t))} = \{(z^{(n)}, y^{(n)})\}_{n=N(t)+1-\bar{N}}^{N(t)+i-\bar{N}}$. Furthermore, we use indices i to denote the values computed using the training set $\mathbb{W}_i^{(N(t))}$, e.g., $\mu_i(t) = \mu_{\mathbb{W}_i^{(N(t))}}(t)$. Based on this notation, we introduce the following auxiliary result.

Lemma 1: The inverse of a kernel matrix

$$\boldsymbol{A}_{\bar{N}} = \begin{bmatrix} \boldsymbol{A}_{\bar{N}-1} & \boldsymbol{k}_{\bar{N}-1}(t^{(\bar{N})}) \\ \boldsymbol{k}_{\bar{N}-1}^T(t^{(\bar{N})}) & k(t^{(\bar{N})}, t^{(\bar{N})}) + \sigma_{\text{on}}^2, \end{bmatrix} \tag{19}$$

can be computed by

$$\boldsymbol{A}_{\bar{N}}^{-1} = \begin{bmatrix} \boldsymbol{A}_{\bar{N}-1}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\sigma_{\bar{N}-1}^2(t^{(\bar{N})}) + \sigma_{\text{on}}^2} \begin{bmatrix} \boldsymbol{a} \\ -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a}^T & -1 \end{bmatrix} \tag{20}$$

with $\boldsymbol{a} = \boldsymbol{A}_{\bar{N}-1}^{-1} \boldsymbol{k}_{\bar{N}-1}(t^{(\bar{N})})$.

Proof: Due to the definition of $\boldsymbol{A}_{\bar{N}}$, the block inversion formula yields

$$\boldsymbol{A}_{\bar{N}}^{-1} = \begin{bmatrix} \boldsymbol{A}_{\bar{N}-1}^{-1} + \frac{\boldsymbol{a}\boldsymbol{a}^T}{\sigma_{\bar{N}-1}^2(t^{(\bar{N})}) + \sigma_{\text{on}}^2} & -\frac{\boldsymbol{a}}{\sigma_{\bar{N}-1}^2(t^{(\bar{N})}) + \sigma_{\text{on}}^2} \\ -\frac{\boldsymbol{a}^T}{\sigma_{\bar{N}-1}^2(t^{(\bar{N})}) + \sigma_{\text{on}}^2} & \frac{1}{\sigma_{\bar{N}-1}^2(t^{(\bar{N})}) + \sigma_{\text{on}}^2} \end{bmatrix}.$$

Rearranging the terms leads to the result. ■

Based on this result, we can derive the following theorem, which we state for scalar systems with $d = 1$ for simplicity, but generalization to arbitrary dimension d is straightforward.

Theorem 1: Consider a scalar filtering problem for a data stream satisfying Assumption 2. Then, the estimation error between the SW-GP prediction $\hat{x}(t) = \mu_{\bar{N}}(t)$ and the true state $x(t)$ is given by

$$\hat{x}(t) - x(t) = \Delta(t) - \nu_0^{\bar{N}} \tilde{x}^{(1)} + \sum_{n=1}^{\bar{N}-1} \nu_n^{\bar{N}} \Delta_n + \sum_{n=1}^{\bar{N}} \nu_n^{\bar{N}} \eta_n \tilde{\epsilon}^{(n)}, \tag{21}$$

where

$$\eta_n = \frac{\sigma_{n-1}^2(\tilde{t}^{(n)})}{\sigma_{n-1}^2(\tilde{t}^{(n)}) + \sigma_{\text{on}}^2} \tag{22}$$

$$\nu_n^{\bar{N}} = \prod_{i=n}^{\bar{N}-1} - \frac{\sigma_{\text{on}}^2}{\sigma_i^2(\tilde{t}^{(n)}) + \sigma_{\text{on}}^2} \tag{23}$$

$$\Delta_n = \tilde{x}^{(n+1)} - \tilde{x}^{(n)} + \mu_n(\tilde{t}^{(n)}) - \mu_n(\tilde{t}^{(n+1)}) \tag{24}$$

$$\Delta(t) = x(t) - \tilde{x}^{(\bar{N})} + \mu_{\bar{N}}(\tilde{t}^{(\bar{N})}) - \mu_{\bar{N}}(t) \tag{25}$$

for $\tilde{t}^{(n)} = t^{(N+n-\bar{N})}$, $\tilde{x}^{(n)} = x^{(N+n-\bar{N})}$ and $\tilde{\epsilon}^{(n)} = \epsilon^{(N+n-\bar{N})}$.

Proof: We prove this theorem by induction, showing that for any $j = 1, \ldots, \bar{N}$ it holds that the error $e_j = \tilde{x}^{(j)} - \mu_j(\tilde{t}^{(j)})$ satisfies

$$e_j = \sum_{n=1}^{j} \eta_n \nu_n^j \tilde{\epsilon}^{(n)} + \sum_{n=1}^{j-1} \nu_n^j \Delta_n + \nu_0^j \tilde{x}^{(1)}. \tag{26}$$

For the induction basis $j = 1$, we have

$$\mu_1(\tilde{t}^{(1)}) = \frac{\tilde{y}^{(1)} k(\tilde{t}^{(1)}, \tilde{t}^{(1)})}{k(\tilde{t}^{(1)}, \tilde{t}^{(1)}) + \sigma_{\text{on}}^2},$$

where $\tilde{y}^{(1)} = y^{(N+1-\bar{N})}$. Since $\sigma_0^2(\tilde{t}^{(1)}) = k(\tilde{t}^{(1)}, \tilde{t}^{(1)})$, it is straightforward to see that

$$e_1 = \frac{\sigma_0^2(\tilde{t}^{(1)})}{\sigma_0^2(\tilde{t}^{(1)}) + \sigma_{\text{on}}^2} \tilde{\epsilon}^{(1)} + \frac{\sigma_{\text{on}}^2}{\sigma_0^2(\tilde{t}^{(1)}) + \sigma_{\text{on}}^2} \tilde{x}^{(1)},$$

which proves the induction basis. For showing the induction step, note that $\boldsymbol{y}_j^T \boldsymbol{A}_j^{-1} \left(\boldsymbol{k}_j(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2 \boldsymbol{1}_j \right) = \tilde{y}^{(j)}$. Moreover, due to Lemma 1, we have

$$\boldsymbol{A}_j^{-1} \boldsymbol{1}_j = \frac{1}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2} \begin{bmatrix} \boldsymbol{A}_{j-1}^{-1} \boldsymbol{k}_{j-1}(\tilde{t}^{(j)}) \\ -1 \end{bmatrix},$$

such that we obtain

$$e_j = \tilde{\epsilon}^{(j)} - \frac{\sigma_{\text{on}}^2 (y^{(j)} - \mu_{j-1}(\tilde{t}^{(j)}))}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2}.$$

Decomposing this into noise and signal components yields

$$e_j = \eta_j \tilde{\epsilon}^{(j)} - \frac{\sigma_{\text{on}}^2 (\tilde{x}^{(j)} - \mu_{j-1}(\tilde{t}^{(j)}))}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2}.$$

Using the definition of Δ_n in (24), the second term on the right-hand side results in

$$\frac{\sigma_{\text{on}}^2 (\tilde{x}^{(j)} - \mu_{j-1}(\tilde{t}^{(j)}))}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2} = \frac{\sigma_{\text{on}}^2 \Delta_{j-1}}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2} + \frac{\sigma_{\text{on}}^2 e_{j-1}}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2}.$$

Therefore, the induction assumption (26) yields

$$e_j = \eta_j \tilde{\epsilon}^{(j)} - \frac{\sigma_{\text{on}}^2 \sum\limits_{n=1}^{j-1} \eta_n \nu_n^{j-1} \tilde{\epsilon}^{(n)}}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2} - \frac{\sigma_{\text{on}}^2 \left(\Delta_{j-1} + \sum\limits_{n=1}^{j-2} \nu_n^{j-1} \Delta_n + \nu_0^{j-1} \tilde{x}^{(1)} \right)}{\sigma_{j-1}^2(\tilde{t}^{(j)}) + \sigma_{\text{on}}^2},$$

which is identical to (26) and therefore, concludes the induction. Finally, it remains to bound the error resulting from predicting into the future, i.e., when not predicting at the training sample $\tilde{t}^{(N)}$, but at an arbitrary point $t > \tilde{t}^{(N)}$. Since the time between these two points is bounded by τ due to the time-triggered sampling scheme, it follows that

$$x(t) - \mu_{\bar{N}}(t) = \tilde{x}^{(\bar{N})} - \mu_{\bar{N}}(\tilde{t}^{(\bar{N})}) + \Delta(t) \tag{27}$$

Therefore, we obtain

$$x(t) - \mu_{\bar{N}}(t) = \Delta(t) - \nu_0^{\bar{N}} \tilde{x}^{(1)} + \sum_{n=1}^{\bar{N}} \nu_n^{\bar{N}} \eta_n \tilde{\epsilon}^{(n)} + \sum_{n=1}^{\bar{N}-1} \nu_n^{\bar{N}} \Delta_n$$

by substituting (26) in (27), which concludes the proof. ■

This result allows a straightforward and intuitive interpretation of the estimation error (21), which can be essentially decoupled into two sources, namely the signal attenuation and the unfiltered noise components. Signal attenuation corresponds to the error, which would result from the SW-GP filter in the absence of any noise due to the amplitude and variation of the signal $x(\cdot)$, and it corresponds to the first three terms in (21). The first summand $\Delta(t)$ results from predicting the state at a future time instant. If the signal is only predicted at sampling times $t = n\tau$, $n \in \mathbb{N}$, it is trivial to see that $\Delta(n\tau) = 0$. The term $\nu_0^{\bar{N}} \tilde{x}^{(1)}$ corresponds to the attenuation of the signal amplitude due to the prior distribution with mean $m(t) = 0$, while $\sum_{n=1}^{\bar{N}-1} \nu_n^{\bar{N}} \Delta_n$ is the error resulting from variation in the signal. While the latter would be approximately equal to 0 for constant signals $x(\cdot)$, the former is non-zero for non-zero signals. Both these terms crucially depend on the parameters $\nu_n^{\bar{N}}$, $n = 1, \ldots, \bar{N}$, which allow intuitive insight into the estimation process. It can be straightforwardly observed that $\nu_n^{\bar{N}} \approx 1$ if $\sigma_{\text{on}} \gg \sigma_n(\tilde{t}^{(n+1)})$ for all i. This implies that the signal and mean variations Δ_n are almost summed up, which can yield a high estimation error. However, this behavior is to be expected since large observation noise standard deviations σ_{on} mean that the signal is heavily smoothed, thereby, strongly attenuating high frequency signals. In contrast, small values $\sigma_{\text{on}} \approx \sigma_n(\tilde{t}^{(n+1)})$ are beneficial for signal reconstruction since they result in $|\nu_n^{\bar{N}}|$ approximately proportional to $\frac{1}{2^{\bar{N}-n}}$. Therefore, variation in the signal $x(\cdot)$ several time steps before the estimation barely has any effect on the reconstruction error. While a small value of σ_{on} is beneficial for low signal attenuation, it also barely attenuates the noise. This becomes clear in the last term in (21), which represents the unfiltered noise components. For $\sigma_{\text{on}} \approx \sigma_n(\tilde{t}^{(n+1)})$, it follows that $\eta_n \approx 1$, such that noise is almost directly passed through to the estimated state $\hat{x}(t)$. In contrast, high observation noise standard deviations $\sigma_{\text{on}} \gg \sigma_n(\tilde{t}^{(n+1)})$ lead to $\eta_n \approx 0$, such that noise is almost completely removed from the signal.

This clearly demonstrates the conflict between noise attenuation and signal recovery in SW-GP filters, which is a well-known problem in systems theory and filter design [1]. This issue is commonly resolved by defining frequency ranges, where either of those two behaviors is dominant, leading to, e.g., low-pass filters. In SW-GP filters, we pursue a similar approach by exploiting the hyperparameters $\boldsymbol{\theta}$, which allow us to directly modify the observation noise standard deviation σ_{on}, but also $\sigma_n(\tilde{t}^{(n)})$ through, e.g., the length scales $\boldsymbol{l}$ when using a squared exponential kernel (4). Thereby, the behavior can be tuned for noise attenuation or low signal disturbance, depending on the measurements $\boldsymbol{y}^{(n)}$. However, in contrast to traditional filtering techniques, this approach has the advantage that hyperparameters can be automatically tuned using log-likelihood maximization. In fact, it is even possible to adapt the hyperparameters online, such that adaptive filters can be straightforwardly implemented using Alg. 1.

While Theorem 1 allows a straightforward interpretation, it cannot be calculated in practice as the noise realizations $\tilde{\epsilon}^{(n)}$, $n = 1, \ldots, \bar{N}$ are generally unknown. However, as long as (probabilistic) bounds for the noise realizations can be derived, it allows us to directly bound the estimation error $x(t) - \hat{x}(t)$. This is exemplarily shown for bounded noise distributions in the following.

Corollary 1: Consider a scalar filtering problem for a data stream satisfying Assumption 2. Assume a Lipschitz continuous kernel is employed for computing the sliding window Gaussian process mean (11) and assume that the noise is bounded by a constant $\bar{\epsilon} \in \mathbb{R}_{0,+}$, i.e., $|\epsilon^{(i)}| \leq \bar{\epsilon}$ for all $i \in \mathbb{N}$. Then, the estimation error between $\hat{x}(t) = \mu_{\bar{N}}(t)$ and $x(t)$ is uniformly bounded with upper bound

$$c(t) = |\nu_0^{\bar{N}} \tilde{x}^{(1)}| + \sum_{n=1}^{\bar{N}} |\nu_n^{\bar{N}}| (\eta_n \bar{\epsilon} + (L_{\mu_n} + L_x)\tau), \tag{28}$$

where L_{μ_n} and L_x denote the Lipschitz constants of the SW-GP mean functions and the signal, respectively.

Proof: Lipschitz continuity of kernel used in the sliding window GP implies Lipschitz continuity of the mean $\mu_n(\cdot)$ for all $n = 1, \ldots, \bar{N}$ [4] and boundedness of $f(\cdot, \cdot, \cdot)$ implies Lipschitz continuity of $x(\cdot)$. Therefore, we obtain

$$\Delta_n \leq (L_{\mu_n} + L_x)\tau \tag{29}$$

$$\Delta(t) \leq (L_{\mu_n} + L_x)\tau. \tag{30}$$

The remainder of the proof follows from Theorem 1. ■

Remark 3: If the estimate $\hat{\boldsymbol{x}}(t) = \mu_{\bar{N}}(t)$ is determined only at sampling times $t = N\tau$, we can write it as a linear combination of previous measurements $\boldsymbol{y}^{(n)}$, $n = N - \bar{N} + 1, \ldots, N$, e.g., for $d = 1$ we have $\hat{x}(N\tau) = \boldsymbol{c}^T \boldsymbol{y}_{\bar{N}}$, where $\boldsymbol{c}^T = \boldsymbol{k}_{\bar{N}}(N\tau)(\boldsymbol{K}_{\bar{N}} + \sigma_{\text{on}}^2 \boldsymbol{I}_{\bar{N}})^{-1}$. Therefore, the SW-GP filter is a linear finite impulse response filter, whose filter coefficients are adapted to the input signal. Note that for stationary kernels $k(\cdot, \cdot)$ [2] and fixed hyperparameters, the filter coefficients are even constant, such that moving average and Sawitzky-Golay type filters [1] can be derived.

Remark 4: There is generally no straightforward way to determine the hyperparameters of the SW-GP without data just from system knowledge. Since the online hyperparameter optimization provides a principled approach for the fast online adaptation of the filtering behavior, this is not an issue.

IV. Numerical Evaluation

In order to demonstrate the low-pass behavior and adaptation capabilities of SW-GPs, we execute three different numerical experiments. In Section IV-A, we

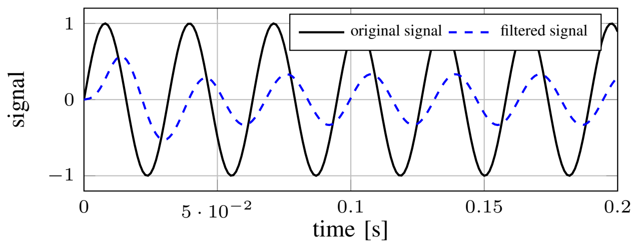

Fig. 1. When filtering a sinusoidal signal with the SW-GP, the resulting output signal is sinusoidal with reduced amplitude and small phase shift.

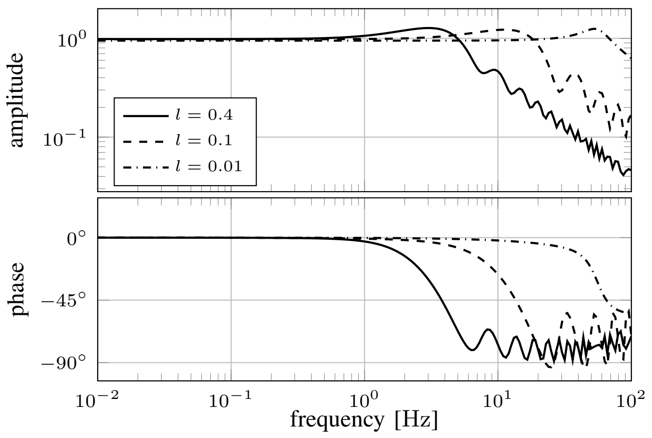

Fig. 2. The SW-GP filter shows a low pass behavior for sinusoidal input signals with increasing cutoff frequency for smaller length scales l.

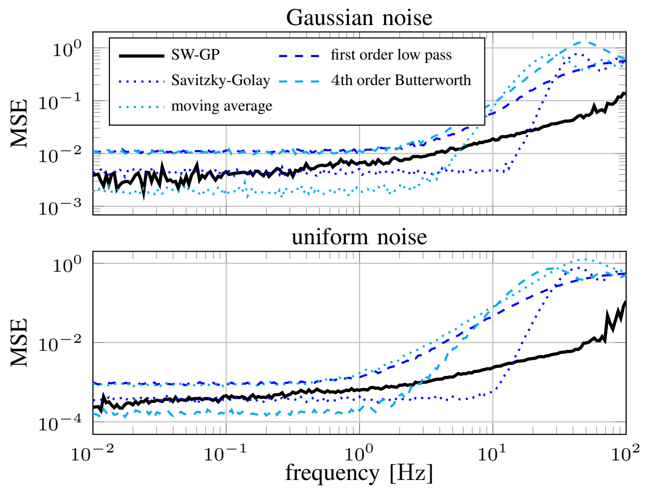

Fig. 3. The estimation error for SW-GPs grows continuously with the frequency for signals perturbed by Gaussian noise (top) and uniform noise (bottom). In contrast, classical filter approaches exhibit a low error for low frequencies, but after the cut-off frequency, no signal can be recovered.

TABLE I

AVERAGE COMPUTATION TIMES WITH STANDARD DEVIATIONS IN BRACKETS FOR DIFFERENT FILTERS

	SW-GP	Savitzky-Golay	moving average	first order low pass	4th order Butterworth
comp. time [ms]	3.4 (5.0)	0.478 (2.1)	0.039 (0.089)	0.002 (0.036)	0.040 (0.078)

illustrate the dependency of the SW-GP filter's low-pass behavior using its frequency response. In Section IV-B, we demonstrate the adaptivity of the SW-GP due to the online hyperparameter optimization, which allows noise attenuation over far wider frequency ranges than existing filters. Finally, we employ the SW-GP filter in a robot control simulation to illustrate its practical applicability in Section IV-C.

A. Frequency Response Analysis

For demonstrating the low-pass behavior of SW-GP filters and its dependency on the hyperparameters, we use a squared exponential kernel (4) with fixed hyperparameters $\sigma_f = 1$, $\sigma_{\text{on}} = \sqrt{0.1}$ and use a window size $\bar{N} = 200$. Moreover, we exemplarily choose different length scales $l = 0.1$, $l = 0.4$ and $l = 0.01$ to illustrate their impact on the cutoff frequency. We apply it to sinusoidal input signals $x(t) = \sin(2\pi f t)$ with frequencies in the range of 10^{-2} Hz to 10^2 Hz. The signals are sampled with a rate of $1kHz$, and the SW-GP filter is updated and evaluated at the same rate. As exemplarily illustrated in Fig. 1 for an input signal with $f = 10^{1.5} \approx 31.62$ Hz, the filtered signal is clearly sinusoidal again which is a direct consequence of Remark 3.

Therefore, we analyze the frequency response of the SW-GP filter using a Bode plot, which is depicted in Fig. 2. It can be clearly seen that the SW-GP exhibits a low-pass behavior, with the signal amplified near its cut-off frequency and non-continuous decay. Furthermore, it can be observed that the cutoff frequency strongly depends on the length scale, which leads to, e.g., a cutoff at approximately 20 Hz for $l = 0.1$. In fact, we empirically found that the cut-off frequency f_c of the SW-GP filter can be approximately computed from its hyperparameters via $f_c \approx \frac{2}{l}$ for large enough windows $\bar{N}$, where l is the length scale of the squared exponential kernel (4). Since the proposed hyperparameter optimization scheme in Section III-B can tune the hyperparameters online, this relationship allows to automatically adapt the cut-off frequency to the observed signal. This beneficial feature is investigated in detail in the following subsection.

B. Adaptive Noise Attenuation

In order to evaluate the adaptive attenuation of noise of the SW-GP filter over a range of input signal frequencies, we employ the sinusoidal input signals from Section IV-A, but perturb them by noise. We run the SW-GP filter for each frequency for 1 s and average over 20 repetitions of the experiment. The hyperparameters of the SW-GP are initialized as $\sigma_f = 1$, $l = 0.1$ and $\sigma_n = \sqrt{0.1}$ and adapted using $\eta^+ = 1.2$ and $\eta^- = 0.5$ in all simulations. The performance of the SW-GP filter is evaluated in terms of the mean squared error (MSE) and compared to a first order low-pass, a fourth order Butterworth, a moving average, and a third order Sawitzki-Golay filter. All finite impules response filters consider a window of data with $\bar{N} = 200$.

The resulting MSE curves for Gaussian noise with variance $\sigma_n^2 = 0.1$ and uniform noise with bounds $\pm 0.5\sqrt{0.1}$ are depicted in Fig. 3. It can be observed, that the SW-GP filter outperforms the infinite impulse response filters over the whole frequency range. While it yields slightly higher errors than the finite impulse response filter approaches for low frequencies, the SW-GP filter is capable of adapting

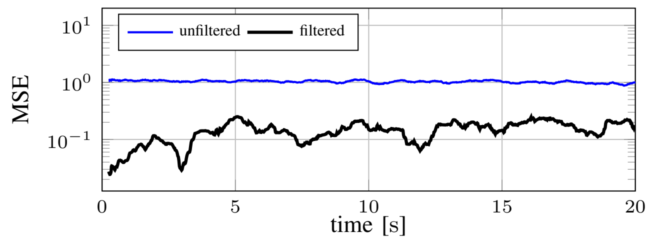

Fig. 4. The SW-GP strongly attenuates the measurement noise, such that the control error induced by noisy measurements is significantly reduced.

to the signal frequency, such that it attenuates the signal significantly less in higher frequency domains. Overall, this leads to a MSE curve which slowly grows, while classical filters are limited to a prescribed frequency band. Even though this comes at the cost of higher computation times as shown in Table I, SW-GP filters are still fast enough for many real-world systems, which often exhibit sampling rates in the range of 10Hz to 1000Hz, and provide a principled approach for the automatic online adaptation of the filtering behavior.

C. Low-Pass Filtering for Robot Control

We demonstrate the practical applicability of the proposed SW-GP filter including its automatic tuning capabilities in a simulation of a planar robotic manipulator with two degrees of freedom [18], which is controlled using a PD control law $\boldsymbol{u} = -k_p(\boldsymbol{q} - \boldsymbol{q}_{\mathrm{ref}}) - k_d(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{ref}})$ with state $\boldsymbol{x} = [\boldsymbol{q}^T \ \dot{\boldsymbol{q}}^T]^T$ and reference $\boldsymbol{x}_{\mathrm{ref}} = [\boldsymbol{q}_{\mathrm{ref}}^T \ \dot{\boldsymbol{q}}_{\mathrm{ref}}^T]^T$. The gains are set to $k_p = 100$, $k_d = 10$ and the control law is run at a frequency of $1\,\mathrm{kHz}$ using zero order hold. The reference signal is chosen to be sinusoidal with increasing frequency, which is realized through the function $\boldsymbol{q}_{\mathrm{ref}} = [\sin(0.1t^2) \ \ 0.5\sin(0.1t^2)]^T$. We assume access to measurements of the joint angles $\boldsymbol{q}$, which are perturbed by Gaussian noise with noise standard deviation $\sigma_{\mathrm{on}} = 0.1$. Angular velocities are obtained through numerical differentiation with finite differences. The SW-GP is initialized as in Section IV-B, but we consider only windows with $\bar{N} = 50$. In our simulations, this results in average update and prediction times of $1.7\,\mathrm{ms}$ and $0.09\,\mathrm{ms}$ with standard deviations of $267\,\mu\mathrm{s}$ and $11\,\mu\mathrm{s}$, respectively. Therefore, updates of the SW-GP are performed at $333\,\mathrm{Hz}$, while predictions are computed at $1\,\mathrm{kHz}$, i.e., the SW-GP is updated every third measurement.

The resulting mean squared error between the state $\boldsymbol{x}$ of the system controlled by noiseless and filtered signal averaged over 20 simulation runs is illustrated in Fig. 4. In comparison with a controller employing the unfiltered signal, the improvement due to the SW-GP filter can clearly be seen. While the error using the filtered signal slightly increases at the beginning, this is in accordance with the results from Section IV-B as the reference trajectory $\boldsymbol{q}_{\mathrm{ref}}$ exhibits a smaller frequency for small t. However, the overall increase of the MSE over time is minor. This clearly demonstrates the applicability of SW-GP filters in more complex systems such as control architectures, where the filtered signal has a direct effect on the evolution of the signal itself.

V. Conclusion

In this paper, we propose Sliding Window Gaussian process filters, which allow adaptive low-pass filtering of signals. By training a Gaussian process on a subset of previous measurements, updates and predictions can be computed with low computational complexity, such that Gaussian processes become applicable in filtering applications. Moreover, an efficient online hyperparameter optimization scheme leads to an automatic adaptation to the input signal, such that no manual parameter tuning is necessary. Finally, we prove that the proposed method guarantees a bounded estimation error using simple linear algebra identities, and demonstrate its flexibility and efficiency in several simulations.

References

[1] S. Orfanidis, *Introduction to Signal Processing*. Prentice-Hall, 1995.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006.

[3] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[4] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for gaussian process regression with application to safe control," in *Advances in Neural Information Processing Systems*, 2019, pp. 659–669.

[5] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer International Publishing, 2016.

[6] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic moment-based Gaussian process filtering," in *International Conference on Machine Learning*, 2009, pp. 1–8.

[7] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, pp. 75–90, 2009.

[8] B. Calderhead, M. Girolami, and N. D. Lawrence, "Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes," in *Advances in Neural Information Processing Systems*, 2009, pp. 217–224.

[9] P. Wenk, G. Abbati, M. A. Osborne, B. Schölkopf, A. Krause, and S. Bauer, "ODIN: ODE-informed regression for parameter and state inference in time-continuous dynamical systems," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 6364–6371.

[10] A. Gijsberts and G. Metta, "Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[11] A. Lederer, A. Odonez Conejo, K. Maier, W. Xiao, J. Umlauft, and S. Hirche, "Gaussian Process-Based Real-Time Learning for Safety Critical Applications," in *International Conference on Machine Learning*, 2021, pp. 6055–6064.

[12] D. Petelin and J. Kocijan, "Control System with Evolving Gaussian Process Models," *IEEE Workshop on Evolving and Adaptive Intelligent Systems*, pp. 178–184, 2011.

[13] F. Meier and S. Schaal, "Drifting Gaussian processes with varying neighborhood sizes for online model learning," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 264–269.

[14] F. Meier, D. Kappler, N. Ratliff, and S. Schaal, "Towards robust online inverse dynamics learning," in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 4034–4039.

[15] A. Lederer, A. Capone, T. Beckers, J. Umlauft, and S. Hirche, "The impact of data on the stability of learning-based control," in *Conference on Learning for Dynamics and Control*, vol. 144, 2021, pp. 623–635.

[16] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *IEEE International Conference on Neural Networks*, 1993, pp. 586–591.

[17] M. Blum and M. Riedmiller, "Optimization of Gaussian process hyperparameters using RPROP," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013, pp. 339–344.

[18] R. M. Murray, Z. Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.