Networked Online Learning for Control of Safety-Critical Resource-Constrained Systems based on Gaussian Processes

Armin Lederer, Mingmin Zhang, Samuel Tesfazgi and Sandra Hirche

***Abstract*— Safety-critical technical systems operating in unknown environments require the ability to quickly adapt their behavior, which can be achieved in control by inferring a model online from the data stream generated during operation. Gaussian process-based learning is particularly well suited for safety-critical applications as it ensures bounded prediction errors. While there exist computationally efficient approximations for online inference, these approaches lack guarantees for the prediction error and have high memory requirements, and are therefore not applicable to safety-critical systems with tight memory constraints. In this work, we propose a novel networked online learning approach based on Gaussian process regression, which addresses the issue of limited local resources by employing remote data management in the cloud. Our approach formally guarantees a bounded tracking error with high probability, which is exploited to identify the most relevant data to achieve a certain control performance. We further propose an effective data transmission scheme between the local system and the cloud taking bandwidth limitations and time delay of the transmission channel into account. The effectiveness of the proposed method is successfully demonstrated in a simulation.**

I. INTRODUCTION

Technical systems are required to operate increasingly autonomously in uncertain environments. For ensuring safety and high performance, these systems need to be able to infer models from observed data online, such that they can quickly adapt to new situations. This is particularly important in applications such as the safe control of autonomous underwater vehicles [1], unmanned aerial vehicles [2] and wearable robots [3], where uncertainty arising from humans in the control loop and changing environments can prevent the derivation of accurate models prior to system operation.

Gaussian process (GP) regression is a supervised machine learning method, which is commonly employed in highly nonlinear, safety-critical applications due to its high expressiveness and probabilistically bounded prediction errors [4]. Even though it admits closed-form updates allowing online learning and thereby an iterative adaptation of inferred models, it exhibits a quadratic update complexity in the number of training samples. Therefore, it becomes too slow for processing streaming data generated during system operation in real-time, since controllers often run at sampling

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 and by the Horizon 2020 research and innovation programme of the European Union under grant agreement number 871767 of the project ReHyb. Published under © IEEE 2022.

The authors are with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany [armin.lederer, mingmin.zhang, samuel.tesfazgi, hirche]@tum.de

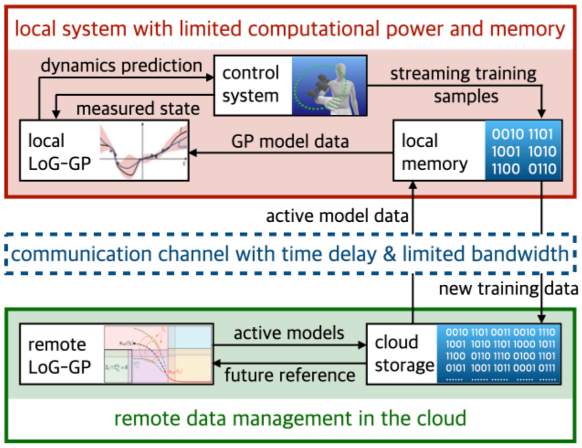


Fig. 1. Overview of the proposed networked online learning architecture: The LoG-GP predicts the unknown dynamics, e.g., of a wearable robot, for a measured state. For computing these predictions, it can only access GP model data in the local memory. Measurements of the system are continuously stored in the local memory and regularly sent to the cloud, where necessary models for a future reference trajectory are determined using a sampling based approach and corresponding data is sent to the local memory.

rates in the magnitude of 10^2 Hz to 10^3 Hz and consequently measurements quickly accumulate to large data sets, which render exact inference computationally intractable [5]. In order to reduce the complexity of GPs, several approximations for online learning have been developed, which include inducing point methods [6], variational inference approaches [7] and finite feature approximations [8]. While these approaches can yield computation times low enough for online learning in control, beneficial safety-relevant theoretical properties of exact GPs such as uniform error bounds [9] do not directly extend to them, and thus, they cannot be used in safety-critical applications. In addition, those approaches exhibit a linear or even higher order polynomial memory complexity, which prohibits their application in resource-constrained technical systems such as drones, autonomous underwater vehicles or wearable robots with limited memory for storing data. In summary, there is a significant gap between the principle potential of GPs and their realistic application in safety-critical systems.

This paper addresses the problem of online learning control for safety-critical systems with limited computational and memory resources. We exploit the fact that our envisioned applications are able to communicate with external infrastructure including clouds with potentially unlimited data storage. Since realistic communication network restrictions such as time delays and limited bandwidth prevent the full externalization of the online model inference, we propose to exploit networked cloud computing only partially for determining and storing GP models. Our approach, which is illustrated in Fig. 1, transmits data to the remote compute system, where a tree structure with localized GP models at its leaf nodes is iteratively build up using a method developed by the authors, which is called locally growing random tree of GPs (LoG-GP)[1] [10]. A sampling-based reachability analysis is executed in the cloud to determine the localized GP models which need to be communicated back to the resource-constrained system. Thereby, predictions and model updates can be computed locally without any delays. We ensure the timely availability of required data on the local system using an effective transmission scheme. This scheme provides insight on fundamental trade-offs between the bandwidth, time delays, local memory and achievable tracking error. By employing a smart aggregation method for the predictions of the local GP models, prediction error bounds of GPs are inherited. Therefore, the safety guarantees derived for control with exact GP models are maintained despite computational and memory limitations. The effectiveness of the developed networked online learning method is demonstrated in close-to-real simulations of a robotic exoskeleton.

The remainder of this paper is structured as follows: Section II formally describes the considered problem, followed by the proposed networked online learning method based on GPs in Section III. In Section IV, a tracking error bound for an online learning feedback linearizing control law is exemplarily derived, such that the effectiveness of the networked online learning approach can be demonstrated in Section V, before the paper is concluded in Section VI.

II. PROBLEM DESCRIPTION

Since accurate models for many systems such as autonomous underwater vehicles and wearable robots are often not available in practice, we consider the problem of inferring a dynamics model online from measurements generated during operation, such that the tracking performance of model-based control can be improved. Formally, we model these systems with differential equations of the form[2]

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

where $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^{d_x}$ denotes the state, $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{d_u}$ is the control input, and $\boldsymbol{f} : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{d_x}$ is the unknown dynamics function. We consider the task of tracking a bounded, continuously differentiable reference trajectory $\boldsymbol{x}_{\mathrm{ref}} : \mathbb{R}_{0,+} \to \mathbb{X}$ with the system state $\boldsymbol{x}(t)$. For this purpose, we employ a model-based control law $\boldsymbol{\pi}_{\hat{f}} : \mathbb{X} \to \mathbb{U}$, where $\hat{\boldsymbol{f}} : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{d}$ is a model of the unknown function $\boldsymbol{f}(\cdot)$. The tracking performance of such a control law typically depends strongly on the accuracy of the model $\hat{\boldsymbol{f}}(\cdot)$, such that we employ the following assumption on the model-based control law $\boldsymbol{\pi}_{\hat{f}}(\cdot)$, which is satisfied by many control techniques such as feedback linearization [11], backstepping [12] and adaptive control [13].

Assumption 1: The tracking error $\boldsymbol{e}(t) = \boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)$ is ultimately bounded with monotonously increasing ultimate bound $\vartheta : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$, i.e., for every $c \in \mathbb{R}_+$, there exists a time $T = T(c, \vartheta)$, such that it holds that

$$\|\boldsymbol{e}(0)\| \leq c \qquad \Rightarrow \qquad \|\boldsymbol{e}(t)\| \leq \vartheta(\kappa_t),\ \forall t \geq T, \tag{2}$$

where $\kappa_t = \max_{t' \in [0,t]} \|\boldsymbol{f}(\boldsymbol{x}(t')) - \hat{\boldsymbol{f}}(\boldsymbol{x}(t'))\|$.

For notational simplicity, we assume no knowledge of $\boldsymbol{f}(\cdot)$ before system operation, but considering a prior model $\hat{\boldsymbol{f}}_0(\cdot)$ is straightforward [11]. In order to infer a model $\hat{\boldsymbol{f}}(\cdot)$ online, we require periodical measurements of the system.

Assumption 2: Data pairs $(\boldsymbol{x}_n, \boldsymbol{y}_n = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{\pi}_{\hat{\boldsymbol{f}}}(\boldsymbol{x}))) + \boldsymbol{\epsilon}_n)$, where $\boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \sigma_y^2 \boldsymbol{I}_{d_x})$ are i.i.d. Gaussian random variables with variance $\sigma_y^2 \in \mathbb{R}_+$, are sampled at time instances $t^{(n)} = n\tau$ with sampling time $\tau \in \mathbb{R}_+$. The data is aggregated in a time-varying training set $\mathbb{D}_t = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N(t) = \lfloor \frac{t}{\tau} \rfloor}$.

Assumption 2 admits training targets $\boldsymbol{y}$ perturbed by Gaussian noise, which is a frequently found assumption in literature, see, e.g., [11], [12], [13]. It also requires noise-free state measurements for training, which however, is commonly assumed in many employed control schemes such as feedback linearization and sliding mode control [14].

Since Assumption 2 ensures a continuous data stream, model updates of $\hat{\boldsymbol{f}}(\cdot)$ must be computed fast enough to avoid that data is generated at higher rates than it can be processed. Hence, the average update time T_{up} of $\hat{\boldsymbol{f}}(\cdot)$ must satisfy the computational constraint

$$T_{\mathrm{up}} \leq \tau. \tag{3}$$

Additionally, the continuous stream of data leads to a steadily growing size of the data set $\mathbb{D}_t$. Therefore, the amount of generated data will eventually reach the memory limitations, which are unavoidable on all real-world systems. Formally, this can be modelled via the memory constraint

$$|\mathbb{D}_t^{\mathrm{loc}}| \leq \bar{M}, \tag{4}$$

where $\mathbb{D}_t^{\mathrm{loc}}$ denotes the data set stored in the memory of the technical system and $\bar{M} \in \mathbb{N}$ represents the memory limitations. Since this restriction can crucially limit the achievable control performance [15], we consider that data can be transferred to a cloud via a network connection, effectively extending the overall memory capacity. The available memory in the cloud is usually significantly larger than on the local system, such that we assume it to be infinite for simplicity. However, the data transfer between the cloud and the local system takes non-negligible time in practice due to effects such as network delays $T_d \in \mathbb{R}_+$ and finite bandwidth $B \in \mathbb{R}_+$. Therefore, data sent to the cloud cannot be immediately accessed by the local system, but the time T_{access} between requesting data $\mathbb{D}$ and using it has to satisfy the network constraint

$$T_{\mathrm{access}} \geq \frac{|\mathbb{D}|}{B} + T_d. \tag{5}$$

Despite these restrictions, the model-based control law $\boldsymbol{\pi}_{\hat{\boldsymbol{f}}}(\cdot)$ using the model $\hat{\boldsymbol{f}}(\cdot)$ learned from the streaming data $\mathbb{D}_t$

[1]Open-source software packages for several programming languages are available at `https://gitlab.lrz.de/online-GPs/LoG-GPs`.

[2]Notation: Lower/upper case bold symbols denote vectors/matrices, $\mathbb{R}_+$/$\mathbb{R}_{0,+}$ all real positive/non-negative numbers, $\boldsymbol{I}_n$ the $n \times n$ identity matrix, $\|\cdot\|$ the Euclidean norm, $|\mathbb{D}|$ the cardinality of a set $\mathbb{D}$, and $\lceil\cdot\rceil$/$\lfloor\cdot\rfloor$ the ceil/floor operator.

should achieve a high tracking control performance. Therefore, we consider the problem of developing a networked online learning method for inferring a highly accurate model $\hat{\boldsymbol{f}}(\cdot)$ of the unknown dynamics $\boldsymbol{f}(\cdot)$ under computational, memory and network constraints.

III. NETWORKED ONLINE LEARNING BASED ON GAUSSIAN PROCESSES

Since the time delay T_d prevents externalizing the online learning, we propose the networked online learning approach outlined in Fig. 1, which performs inference locally, but transfers unnecessary data to the cloud. The approach is based on GP regression [4] due to its strong theoretical foundation as introduced in Section III-A. For enabling online learning with GPs, we employ LoG-GPs firstly proposed in our earlier work [10], which inherit the probabilistic prediction error guarantees of exact GPs while having merely logarithmically increasing update and prediction complexities as outlined in Section III-B. In order to transmit data to the cloud without performance loss, we exploit the modular structure of LoG-GPs and determine the region, in which system states can potentially be in a given time interval, using a sampling-based approach in Section III-C. By developing a data transmission scheme in Section III-D, we ensure that necessary data is always locally available despite transmission bandwidth limitations and network delays. For notational simplicity, the proposed method is presented for scalar functions $f(\cdot)$, but can be employed for the vector-valued dynamics in (1) by applying it to each dimension individually.

A. Gaussian Process Regression

A Gaussian process is an infinite collection of random variables, any finite subset of which follows a joint Gaussian distribution [4]. The GP is usually denoted as $\mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, where $m : \mathbb{R}^d \to \mathbb{R}$ is a prior mean incorporating a priori knowledge such as approximate models, and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{0,+}$ is a covariance function reflecting information such as periodicity. Since we assume no prior knowledge, the prior mean $m(\cdot)$ is set to 0 in the sequel. Analogously, we employ the probably most common choice for the covariance function: the squared exponential kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\sum_{i=1}^d (x_i - x_i')^2/(2l_i^2))$, where $\sigma_f \in \mathbb{R}_+$ denotes the signal standard deviation, and $l_i \in \mathbb{R}_+$, $i = 1, \dots, d$ are length scales [4].

Given a prior GP $\mathcal{GP}(0, k(\cdot,\cdot))$, regression is performed by conditioning on the training data $\mathbb{D}_t$ as introduced in Assumption 2. The resulting posterior distribution is again Gaussian with mean and variance given by

$$\mu(\boldsymbol{x}) = \boldsymbol{k}^T(\boldsymbol{x}) \left(\boldsymbol{K} + \sigma_y^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \tag{6}$$

$$\sigma^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^T(\boldsymbol{x}) \left(\boldsymbol{K} + \sigma_y^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}(\boldsymbol{x}), \tag{7}$$

where the elements of $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{k}(\boldsymbol{x}) \in \mathbb{R}^N$ are defined through $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $k_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$, respectively, and we concatenate training targets $\boldsymbol{y} = [y_1 \ \cdots \ y_N]^T$.

B. Locally Growing Random Tree of Gaussian Processes

Since the update complexity of Gaussian process regression scales quadratically with the number of training samples, we employ the recently proposed approach of locally growing random trees of GPs [10], which preserves beneficial properties of exact GP inference such as the existence of uniform prediction error bounds. LoG-GPs rely on the idea of iteratively constructing a tree, whose leaf nodes contain locally active GP models. In detail, the construction starts with a single GP model, which is updated with incoming streaming data until a prescribed threshold of training samples $\bar{N}$ is reached. When the GP model contains $\bar{N}$ training samples in its data set $\mathbb{D}_0$, the data set is split into 2 subsets $\mathbb{D}_i$, $i = 1, 2$, by assigning data in $\mathbb{D}_0$ to a subset $\mathbb{D}_i$ via sampling from a Lipschitz continuous probability function $p^0 : \mathbb{R}^d \to [0, 1]$. Thereby, a tree with 2 leaf nodes is generated, which contain all the data, such that individual GP models can be efficiently computed using (6) and (7). New streaming data obtained after the splitting can be assigned to the subsets $\mathbb{D}_i$ by sampling from $p^0(\cdot)$ again until either of the subsets $\mathbb{D}_i$ reaches the capacity limit $\bar{N}$. Then, a new probability function $p^i : \mathbb{R}^d \to [0, 1]$ is defined to distribute the data to new subsets, thereby extending the tree of GPs by a new layer. By repeating this procedure every time a subset $\mathbb{D}_i$ reaches $\bar{N}$ training samples, a tree of GP models is iteratively constructed with a computational complexity of $\mathcal{O}_p(\log(N))$ allowing updates with rates up to 1kHz [10], which is fast enough to satisfy the computational constraint (3) in many systems.

For computing predictions with LoG-GPs, we simply multiply the probabilities $p^i(\boldsymbol{x})$ along a path to a leaf node l to obtain the weight $\omega_l(\boldsymbol{x})$. Then, a generalized product of experts aggregation scheme [5] can be employed to obtain the approximate GP prediction

$$\tilde{\mu}(\boldsymbol{x}) = \sum_{l\in\mathbb{L}} \frac{\omega_l(\boldsymbol{x})\tilde{\sigma}^2(\boldsymbol{x})}{\sigma_l^2(\boldsymbol{x})} \mu_l(\boldsymbol{x}), \quad \tilde{\sigma}^{-2}(\boldsymbol{x}) = \sum_{l\in\mathbb{L}} \frac{\omega_l(\boldsymbol{x})}{\sigma_l^2(\boldsymbol{x})}, \tag{8}$$

where $\mathbb{L}$ denotes the set of leaf nodes of the tree of GP models. By defining the probability functions such that only a single child node has a positive probability in most of the input domain $\mathbb{R}^d$, most of the weights $\omega_l(\boldsymbol{x})$ become 0. Since the definition of the aggregated mean $\tilde{\mu}(\cdot)$ implies that the local GP predictions $\mu_l(\boldsymbol{x})$ and $\sigma_l^2(\boldsymbol{x})$ must only be computed if $\omega_l(\boldsymbol{x}) > 0$, models with $\omega_l(\boldsymbol{x}) = 0$ can be considered locally inactive at $\boldsymbol{x}$ and therefore, aggregated predictions can be efficiently computed in $\mathcal{O}_p(\log^2(N))$ complexity. Moreover, this construction of the aggregated prediction $\tilde{\mu}(\cdot)$ ensures that uniform error bounds are directly inherited from exact GP regression.

Lemma 1 ([10]): Assume the function $f : \mathbb{R}^d \to \mathbb{R}$ is a sample from a Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ with a L_k-Lipschitz kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Then, the aggregated mean function (8) of a LoG-GP trained with data satisfying Assumption 2 guarantees a probabilistically, uniformly bounded prediction error on a compact domain $\Omega \subset \mathbb{R}^d$, i.e., for $\delta \in (0, 1)$ and $\rho \in \mathbb{R}_+$, we have

$$P(|f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})| \leq \eta(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega) \geq 1 - \delta, \tag{9}$$

where

$$\eta(\boldsymbol{x}) = \sqrt{\beta(\delta, \rho)} \sum_{l\in\mathbb{L}} \frac{\omega_l(\boldsymbol{x})\tilde{\sigma}^2(\boldsymbol{x})}{\sigma_l(\boldsymbol{x})} + \gamma(\rho) \tag{10}$$

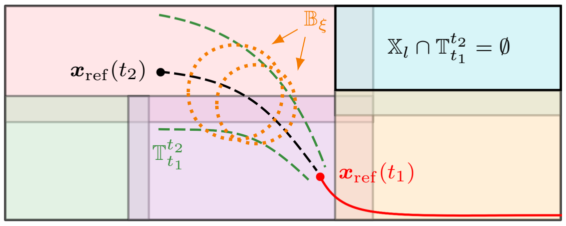

$\mathbb{B}_\xi$
$\mathbb{X}_l \cap \mathbb{T}_{t_1}^{t_2} = \emptyset$
$\boldsymbol{x}_{\text{ref}}(t_2)$
$\mathbb{T}_{t_1}^{t_2}$
$\boldsymbol{x}_{\text{ref}}(t_1)$

Fig. 2. A local model $l \in \mathbb{L}$ is inactive if its active region $\mathbb{X}_l$ does not intersect with the tube $\mathbb{T}_{t_1}^{t_2}$ induced by the tracking error bound $\vartheta(\kappa_t)$ as illustrated for the region in the top right. The set of active models $\hat{\mathbb{A}}$ is found by over-approximating the tube $\mathbb{T}_{t_1}^{t_2}$ with balls $\mathbb{B}_\xi$, from which random samples $\boldsymbol{x}^{(i)}$ are drawn to determine the active models $\mathbb{A}_{\boldsymbol{x}^{(i)}}$ at these states.

$$\beta(\delta,\rho) = 2\log\left(d^{\frac{d}{2}}|\mathbb{L}| \max_{\boldsymbol{x},\boldsymbol{x}'\in\mathbb{R}^d} \|\boldsymbol{x}-\boldsymbol{x}'\|_\infty^d\right) - 2\log\left(\delta 2^d \rho^d\right) \quad (11)$$

$$\gamma(\rho) = \sum_{l\in\mathbb{L}} \frac{\omega_l \tilde{\sigma}^2(\boldsymbol{x})}{\sigma_l^2(\boldsymbol{x})}\left(L_{\mu_l}\rho + \sqrt{\beta(\rho)}L_{\sigma_l}\tau\right) + L_f\rho, \quad (12)$$

and L_f, L_{μ_l}, L_{σ_l} are Lipschitz constants of $f(\cdot)$, $\mu_l(\cdot)$, $\sigma_l(\cdot)$.

This result relies on a well-calibrated prior GP, which is a rather unrestrictive assumption in practice since the sample spaces of GPs are very expressive for many frequently used kernels, e.g., the space of continuous functions for GPs with a squared exponential kernel [9]. Since the posterior variances $\sigma_l^2(\boldsymbol{x})$ are guaranteed to converge to 0 for dense data [16], this result ensures that arbitrarily accurate models can be obtained. Therefore, LoG-GPs are accompanied by strong theoretical guarantees for their prediction accuracy as required in safety-critical applications.

C. Sampling-Based Identification of Active Models

Since Lemma 1 ensures bounded prediction errors when using (8) to learn a model $\hat{\boldsymbol{f}}(\cdot)$ of the unknown dynamics $\boldsymbol{f}(\cdot)$, we can determine the system states $\boldsymbol{x}$ which can be potentially reached within a fixed time interval using the tracking error bound $\vartheta(\kappa_t)$ introduced in Assumption 1. Therefore, we can obtain the models, which need to be available in the local memory, by finding all individual GP models which are active for states $\boldsymbol{x}$ in the potentially reachable set.

In detail, this set $\mathbb{A}$ of potentially active models during a time window $\mathbb{W} = [t_1, t_2]$, $t_1, t_2 \in \mathbb{R}$, $t_2 > t_1$, is defined through the intersections between active regions $\mathbb{X}_l = \{\boldsymbol{x} : \omega_l(\boldsymbol{x}) > 0\}$ of local models $l \in \mathbb{L}$ and the tube $\mathbb{T}_{t_1}^{t_2} = \{\boldsymbol{x} \in \mathbb{R}^d : \exists t \in \mathbb{W}, \boldsymbol{x} \in \mathbb{B}_{\vartheta(\kappa_t)}\}$ based on balls $\mathbb{B}_{\vartheta(\kappa_t)} = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\| \leq \vartheta(\kappa_t)\}$ with radius given by Assumption 1, i.e., $\mathbb{A} = \{l \in \mathbb{L} : \mathbb{X}_l \cap \mathbb{T}_{t_1}^{t_2} \neq \emptyset\}$ as illustrated in Fig. 2. Since the computation of the intersections $\mathbb{X}_l \cap \mathbb{T}_{t_1}^{t_2}$ requires an explicit representation of the active regions $\mathbb{X}_l$ of local models $l \in \mathbb{L}$, which is not provided by LoG-GPs, the definition of $\mathbb{A}$ cannot be directly used in practice. We follow a different idea exploiting the implicit representation of the active regions $\mathbb{X}_l$ via the weights $\omega_l(\cdot)$, which allows to directly compute the set of active models $\mathbb{A}_{\boldsymbol{x}} = \{l \in \mathbb{L} : \omega_l(\boldsymbol{x}) > 0\}$ for a given state $\boldsymbol{x}$. Therefore, we can alternatively represent the set of potentially active models during the time window $\mathbb{W}$ via $\mathbb{A} = \bigcup_{t\in\mathbb{W}} \bigcup_{\boldsymbol{x}\in\mathbb{B}_{\vartheta(\kappa_t)}} \mathbb{A}_{\boldsymbol{x}}$. By approximating the unions over uncountable sets via discretization and random sampling as outlined in Algorithm 1, we can over-approximate the set $\mathbb{A}$ via $\hat{\mathbb{A}}$ and obtain

Algorithm 1: Determining Active Models

```
Function ActiveModels($N_s$, $t_1$, $t_2$, $\Delta t$):
    $\hat{\mathbb{A}} \leftarrow \emptyset$
    compute $\xi$ using (14)
    for $j = 0 : \lceil \frac{t_2 - t_1}{\Delta t} \rceil$ do
        for $i = 1 : N_s$ do
            Determine active models $\mathbb{A}_{\boldsymbol{x}^{(i)}}$ for input $\boldsymbol{x}^{(i)} \sim \mathcal{U}(\mathcal{B}_\xi)$
            $\hat{\mathbb{A}} \leftarrow \hat{\mathbb{A}} \cup \mathbb{A}_{\boldsymbol{x}^{(i)}}$
    return $\hat{\mathbb{A}}$
```

$$\hat{\mu}(\boldsymbol{x}) = \sum_{l\in\hat{\mathbb{A}}} \frac{\omega_l(\boldsymbol{x})\hat{\sigma}^2(\boldsymbol{x})}{\sigma_l^2(\boldsymbol{x})}\mu_l(\boldsymbol{x}), \quad \hat{\sigma}^{-2}(\boldsymbol{x}) = \sum_{l\in\hat{\mathbb{A}}} \frac{\omega_l(\boldsymbol{x})}{\sigma_l^2(\boldsymbol{x})}. \quad (13)$$

If sufficiently many samples are used, this approximation yields identical predictions as shown in the following result.

Theorem 1: Consider a dynamical system (1) and assume Assumptions 1 and 2 hold. Choose $\|\boldsymbol{e}(t_1)\| \leq \vartheta(\kappa_{t_1})$ and

$$\xi = 2\zeta + L_{\boldsymbol{x}_{\text{ref}}}\frac{\Delta t}{2} + \vartheta(\kappa_{t_1 + j\Delta t}) \quad (14)$$

for constants $\zeta, \Delta t \in \mathbb{R}_+$. Then, with probability of at least

$$1 - \|\mathbb{L}\| \left\lceil \frac{t_2 - t_1}{\Delta t} \right\rceil \left(1 - \frac{\min\{r_{\min}^{d_x}, \zeta^{d_x}\}}{\xi^{d_x}}\right)^{N_s}, \quad (15)$$

where $r_{\min}$ denotes the radius of the largest ball contained in the smallest active region of a leaf node $l \in \mathbb{L}$, the predictions $\hat{\mu}(\boldsymbol{x}(t))$ and $\tilde{\mu}(\boldsymbol{x}(t))$ are identical for all $t \in \mathbb{W}$.

Proof: See Appendix A. ■

Since this theorem ensures that (8) and (13) are identical with probability greater than (15), it ensures that using $\hat{\mu}(\cdot)$ as model in a control law $\boldsymbol{\pi}_{\hat{\boldsymbol{f}}}(\cdot)$ yields no reduction in control performance with high probability. Therefore, it allows us to determine irrelevant data for a time interval $\mathbb{W}$, which we exploit in the following section for transmitting data to the cloud, thereby reducing the local memory occupation.

D. Transmission Scheme

Due to the non-negligible time required for a data transfer, the transmission to and from the cloud must be carefully scheduled in order to ensure that the necessary data is always available locally. For simplicity, we consider that data is transmitted at regularly spaced time instances $j\Delta T$, $j = \mathbb{N}$, such that each time interval $\mathbb{W}_j = [(j-1)\Delta T, j\Delta T]$ has a length of $\Delta T \in \mathbb{R}_+$. During each time interval $\mathbb{W}_j$, we propose the transmission scheme illustrated in Fig. 3, where the idea is that the memory is divided into two parts. During each interval $\mathbb{W}_j$, half of the memory is used for updating the local data set with data from the cloud, while the other half contains the data set $\mathbb{D}_j$ necessary for computing the mean predictions $\hat{\mu}(\cdot)$ during time interval $\mathbb{W}_j$ according to the potentially active models $\hat{\mathbb{A}}_j$. For updating the local memory, the data set $\mathbb{D}_{j-1}$ from the previous interval $\mathbb{W}_{j-1}$, which contains newly measured training samples as well as data from the cloud, is sent to the cloud. Once this transmission has been completed, the cloud contains the complete data set $\mathbb{D}_{(j-1)\Delta T}$ obtained until time $(j-1)\Delta T$, such that Algorithm 1 can be employed to determine the possibly active models $\hat{\mathbb{A}}_{j+1}$ for the next time interval $\mathbb{W}_{j+1}$

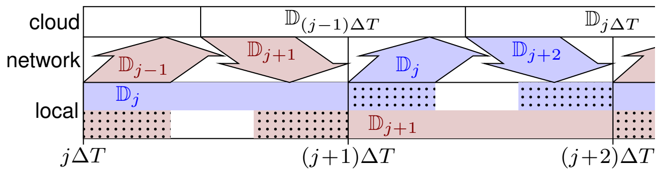


Fig. 3. During each interval $\mathbb{W}_j = [j\Delta T, (j+1)\Delta T]$, the previously necessary data $\mathbb{D}_{j-1}$ is sent to the cloud and the data $\mathbb{D}_{j+1}$ for the next interval $\mathbb{W}_{j+1}$ is fetched. While these data sets occupy memory during the interval $\mathbb{W}_j$, parts of $\mathbb{D}_{j-1}$ and $\mathbb{D}_{j+1}$ are in transmission and not available on the local system. Therefore, these data sets cannot be used for prediction, which is highlighted through the dotted pattern. The data in the cloud is updated with incoming transmissions, such that it contains the complete data set $\mathbb{D}_{(j-1)\Delta T}$ up to the end of previous interval $j-1$.

in the cloud. The corresponding data set $\mathbb{D}_{j+1}$ is sent to the local memory, such that it is available for $t \geq (j+1)\Delta T$.

It is straightforward to see that this transmission scheme can ensure the satisfaction of the network constraint (5) for a fixed data set $\mathbb{D}_j$, if $T_{\text{access}} = \Delta T/2$ is sufficiently large. However, due to the online generation of data during system operation, it generally cannot be ensured that the data sets $\mathbb{D}_j$ have a bounded size, such that the fixed time ΔT might eventually not be sufficient to finish the transmission within the time interval $\mathbb{W}_j$. Therefore, the real-time learning with data generated online during system operation has to be stopped eventually at some interval $\mathbb{W}_\iota$, $\iota \in \mathbb{N}$ in order to upper bound the size of all sets $\mathbb{D}_j$. This leads to the data transfer scheme outlined in Algorithm 2 for the cloud and in Algorithm 3 for the local system, for which it is straightforward to prove the satisfaction of the network constraint (5) as shown in the following result.

Lemma 2: Choose $\Delta T \geq \frac{\bar{M}}{B} + 2T_d$ and $\iota \in \mathbb{N}$ such that the memory constraint (4) is satisfied. Then, Algorithms 2 and 3 ensure the satisfaction of the network constraint (5).

Proof: See Appendix B. ■

In order to apply this lemma in a real-world system, it remains to develop an approach for enforcing the memory constraint (4) by choosing a suitable value of ι. In practice, this value can be selected online using heuristics such that learning can be stopped, e.g., when the number of active models exceeds a threshold. Moreover, when the reference $\boldsymbol{x}_{\text{ref}}$ is periodic, we can determine ι based on the data sets from previous periods, as shown in the following theorem.

Theorem 2: Assume the reference trajectory is periodic with period $T_p = q\Delta T$ for $\Delta T \geq \frac{\bar{M}}{B} + 2T_d$ and $q \in \mathbb{N}$. Let

$$\iota = q + \min_{|\mathbb{D}_j| > \frac{\bar{M}-2\bar{m}}{2}} j, \quad \bar{m} = \max_{j\in\mathbb{N}} |\mathbb{D}_{(j+q)}| - |\hat{\mathbb{D}}_j| \leq \left\lceil \frac{T_p}{\tau} \right\rceil. \tag{16}$$

Then, Algorithms 2 and 3 ensure the satisfaction of the memory constraint (4) and network constraint (5).

Proof: See Appendix C. ■

This theorem allows to determine online when to stop adding new training samples to the LoG-GP by checking if $|\mathbb{D}_j| > \frac{\bar{M}}{2} - \bar{m}$, which can be performed with low complexity and can be directly implemented. Moreover, it provides valuable insight into the interrelations between achievable tracking accuracy, memory constraint $\bar{M}$, time delay T_d and limited bandwidth B. In order to see this, note that the data set size $|\mathbb{D}_j|$ usually grows almost linearly with the interval length ΔT. Since an increase in bandwidth B admits smaller ΔT, learning can continue up to higher values of ι in general. Therefore, a higher data density can be achieved, which in turn yields a lower GP variance [16] guaranteeing a smaller tracking error. In contrast, an increase in local memory $\bar{M}$ admits larger data set sizes $|\mathbb{D}_j|$, but in turn requires longer intervals ΔT, such that the achievable data density and consequently the tracking accuracy are barely affected. Finally, a reduction of the delay T_d allows smaller values of ΔT and thereby also leads to an improvement in achievable control performance. Therefore, available bandwidth B for data transmission and time delay T_d are crucial for the achievable tracking accuracy when using the networked online learning control law, while finite local memory $\bar{M}$ only has secondary relevance to enable implementation of the transmission scheme using Algorithm 2 and 3. This insight can be beneficially used for the design of autonomous systems in practice, since it allows a reduction of local memory when sufficient bandwidth for data transmission is available.

Algorithm 2: Data Transfer Scheme: Cloud

```
Function UpdateLoop($\Delta T$, $\tau$, $\Delta t$):
    for $n = 1, \ldots, \infty$ do
        if $n\tau \geq j\Delta T$ then
            $j \leftarrow j+1$
            $\mathbb{D}_{j-1} \leftarrow$ Receive ()
            $\hat{\mathbb{A}}_{j+1} \leftarrow$ ActiveModels($N_s, j\Delta T, (j+1)\Delta T, \Delta t$)
            Transmit ($\mathbb{D}_{j+1}$)
```

Algorithm 3: Data Transfer Scheme: Local System

```
Function UpdateLoop($\Delta T$, $\iota$, $\tau$):
    for $n = 1, \ldots, \infty$ do
        if $n\tau \leq \iota\Delta T$ then
            $\mathbb{D}_t^{\text{loc}} \leftarrow \mathbb{D}_t^{\text{loc}} \cup (\boldsymbol{x}^{(n)}, y^{(n)})$
        if $n\tau \geq j\Delta T$ then
            $j \leftarrow j+1$
            Transmit ($\mathbb{D}_{j-1}$)
            Delete ($\mathbb{D}_{j-1}$)
            $\mathbb{D}_{j+1} \leftarrow$ Receive ()
```

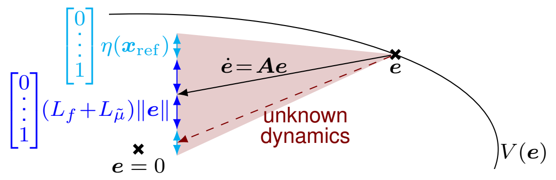
IV. Efficient Tracking Error Bounds

In order to demonstrate the applicability of the proposed networked online learning approach, we exemplarily derive a tracking error bound $\vartheta(\kappa_t)$ for a feedback linearizing control law, which can be applied to a wide range of practically relevant systems such as robotic manipulators, unmanned aerial and autonomous underwater vehicles. For the derivation of $\vartheta(\kappa_t)$ we employ Lyapunov stability theory, such that computing $\vartheta(\kappa_t)$ effectively reduces to determining the regions of the state space with decreasing Lyapunov function along system trajectories. Due to the prediction error bound $\eta(\cdot)$, this decrease condition can be efficiently decoupled for feedback linearizing control laws as illustrated in Fig. 4. Thereby, we obtain a straightforwardly implementable tracking error bound, which can be directly used in Algorithms 2 and 3.

In more detail, we consider feedback linearizable systems

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = x_3, \quad \ldots \quad \dot{x}_{d_x} = f(\boldsymbol{x}) + g(\boldsymbol{x})u, \tag{17}$$

where a scalar control input u as well as scalar functions



$\begin{bmatrix}0\\ \vdots\\ 1\end{bmatrix}\eta(\boldsymbol{x}_{\mathrm{ref}})$
$\begin{bmatrix}0\\ \vdots\\ 1\end{bmatrix}(L_f+L_{\tilde{\mu}})\|\boldsymbol{e}\|$
$\dot{\boldsymbol{e}}=\boldsymbol{A}\boldsymbol{e}$
$\boldsymbol{e}$
unknown dynamics
$\boldsymbol{e}=0$
$V(\boldsymbol{e})$

Fig. 4. Ultimate boundedness is analyzed using Lyapunov theory with Lyapunov function $V(\boldsymbol{e}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e}$. For exact feedback linearization, the closed-loop dynamics are linear with $\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e}$ and ensure a decreasing Lyapunov function along system trajectories. Due to model errors, which can be expressed through the model error bound $\eta(\boldsymbol{x}_{\mathrm{ref}})$ at the reference $\boldsymbol{x}_{\mathrm{ref}}$ and the error $(L_f+L_{\tilde{\mu}})\|\boldsymbol{e}\|$ of linearization around the reference, the uncertainty in the dynamics can be bounded. Thereby, the region with decreasing Lyapunov function can be efficiently determined.

$f:\mathbb{X}\to\mathbb{R}$ and $g:\mathbb{X}\to\mathbb{R}$ are assumed only for simplicity of exposition, while all derived results straightforwardly extend to multi-input systems in the canonical form. Similar to previous work [11], we assume that $f(\cdot)$ is an unknown function, while $g(\cdot)$ is known. The knowledge of $g(\cdot)$ is merely used to streamline the presentation, but all results can be extended to unknown functions $g(\cdot)$ following the approach in [15]. In order to ensure global controllability of (1), the following assumption is needed.

Assumption 3: The function $g(\cdot)$ is positive, i.e., $g(\boldsymbol{x})>0$.

This assumption is a standard condition when designing control laws for systems in the canonical form [14, Definition 13.1], and ensures the non-singularity of $g(\cdot)$. It is naturally satisfied by many systems such as Euler-Lagrange systems, where $g(\cdot)$ corresponds to the positive inertia. Therefore, this assumption is not restrictive in practice.

Additionally, we assume that the unknown function is well-behaved, which is formalized in the following.

Assumption 4: The function $f(\cdot)$ is L_f-Lipschitz.

This assumption globally ensures a unique solution for the system (1) [14], such that it can be commonly found in control. Since it is satisfied by many systems such as Euler-Lagrange dynamics in practice, it is not restrictive.

In order to allow the accurate tracking of the reference trajectory with system (17), we consider reference trajectories

$$\boldsymbol{x}_{\mathrm{ref}}(t)=\begin{bmatrix}x_{\mathrm{ref}}(t) & \dot{x}_{\mathrm{ref}}(t) & \cdots & \frac{\mathrm{d}^{d_x-1}}{\mathrm{d}t^{d_x-1}}x_{\mathrm{ref}}(t)\end{bmatrix}^T, \quad (18)$$

where $x_{\mathrm{ref}}:\mathbb{R}\to\mathbb{R}$ is d_x times continuously differentiable. For tracking this trajectory, we employ the control law

$$u=\pi_{\mathrm{FL}}(\boldsymbol{x})=g^{-1}(\boldsymbol{x})\Big(-\tilde{\mu}(\boldsymbol{x})+\nu+\frac{\mathrm{d}^{d_x}}{\mathrm{d}t^{d_x}}x_{\mathrm{ref}}(t)\Big), \quad (19)$$

where the mean $\tilde{\mu}(\boldsymbol{x})$ defined in (8) is used as model. The input ν to the approximately linearized system is given by the linear feedback law $\nu=-k_c\begin{bmatrix}\lambda_1 & \cdots & \lambda_{d_x-1} & 1\end{bmatrix}\boldsymbol{e}$, where $k_c\in\mathbb{R}_+$ is the control gain and $\lambda_1,\ldots,\lambda_{d_x-1}\in\mathbb{R}$ are coefficients such that for $s\in\mathbb{C}$, the polynomial $s^{d_x-1}+\lambda_{d_x-1}s^{d_x-2}+\ldots+\lambda_1$ is Hurwitz. Due to these choices, the error dynamics can be compactly expressed by

$$\dot{\boldsymbol{e}}=\underbrace{\begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ -\lambda_1 k_c & -\begin{bmatrix}\lambda_2 k_c & \cdots & k_c\end{bmatrix}\end{bmatrix}}_{\boldsymbol{A}}\boldsymbol{e}+(f(\boldsymbol{x})-\tilde{\mu}(\boldsymbol{x}))\begin{bmatrix}0\\ \vdots\\ 1\end{bmatrix}. \quad (20)$$

The matrix $\boldsymbol{A}$ defines a stable dynamical system because of the Hurwitz coefficients λ_i, which is independent of the online learning. Therefore, the second summand in (20) can be considered a disturbance depending on the online learning, which allows us to straightforwardly analyze the ultimate boundedness of this system using Lyapunov theory.

Theorem 3: Consider a dynamical system (1), where $f(\cdot)$ is a sample from a Gaussian process with L_k-Lipschitz kernel $k(\cdot,\cdot)$. Moreover, assume Assumptions 2, 3 and 4 hold, and choose a positive definite, symmetric matrix $\boldsymbol{Q}\in\mathbb{R}^{d\times d}$ and a control gain k_c such that

$$\|\boldsymbol{p}_d(k_c)\|<\frac{\lambda_{\min}(\boldsymbol{Q})}{2(L_f+L_{\tilde{\mu}})}, \quad (21)$$

where $\boldsymbol{P}=[\boldsymbol{p}_1(k_c)\ \cdots\ \boldsymbol{p}_d(k_c)]$ is the solution to the Lyapunov equation $\boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A}=-\boldsymbol{Q}$ and $L_{\tilde{\mu}}$ denotes the Lipschitz constant of the LoG-GP mean $\tilde{\mu}(\cdot)$. Then, the online learning feedback linearizing control law (19) guarantees an ultimately bounded tracking error

$$\vartheta(\kappa_t)=\frac{2\|\boldsymbol{p}_d(k_c)\|\sqrt{\lambda_{\max}(\boldsymbol{P})}\max_{t'\in[0,t]}\eta(\boldsymbol{x}_{\mathrm{ref}}(t'))}{(\lambda_{\min}(\boldsymbol{Q})-2\|\boldsymbol{p}_d(k_c)\|(L_f+L_{\tilde{\mu}}))\sqrt{\lambda_{\min}(\boldsymbol{P})}}. \quad (22)$$

Proof: See Appendix D. ∎

This theorem has the advantage over previously derived results in similar settings [9], [11] that the ultimate bound $\vartheta(\kappa_t)$ in Theorem 3 depends only on the standard deviation along the reference $\boldsymbol{x}_{\mathrm{ref}}$. Thereby, the ultimate bound (22) can be efficiently computed, whereas the results in previous works provide only an implicit representation of the ultimate bound due to the dependency of the GP error bound on the state $\boldsymbol{x}$. This advantage resulting from the linearization around the reference $\boldsymbol{x}_{\mathrm{ref}}$ in (26) comes at the cost of the additional constraint (21) for the control gain k_c compared to existing approaches [9], [11]. Even though this constraint makes the application of Theorem 3 more restrictive, this weakness is strongly outweighed by the benefit of the explicit tracking error bound for determining the active models online.

Remark 1: Due to the definition of $\boldsymbol{A}$ in (20), it can be straightforwardly checked that it is always possible to ensure the satisfaction of (21) for a fixed matrix $\boldsymbol{Q}$ by choosing a sufficiently large gain k_c. Therefore, condition (21) effectively imposes a lower bound for the control gains k_c admitting ultimate tracking error bounds (22).

V. EVALUATION IN EXOSKELETON CONTROL

In order to evaluate the applicability of the proposed networked online learning approach for resource constrained systems[3], we employ it for the control of an upper-limb human-exoskeleton assisting a user in tracking a reference trajectory, which is simulated in Julia [17], a modern programming language for accelerating physics simulations. Since the exoskeleton is intended to be used in a portable manner, this scenario resembles an example for a wearable robotic system with memory and computational constraints. These constraints are particularly challenging for the control of the exoskeleton as human user data is required in practice to infer models allowing for personalized assistance.

[3]Open-source code conceptually demonstrating the proposed method is available at `https://gitlab.lrz.de/online-GPs/cloud-GPs`.

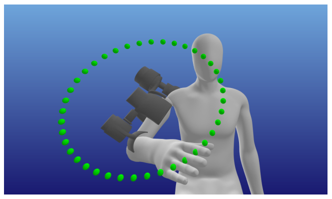


Fig. 5. Visualization of the upper-limb human-exoskeleton simulation and trajectory tracking task. The green circles depict discrete points along the elliptic reference trajectory, which must be followed with the hand.

TABLE I
BANDWIDTH B, TIME DELAY T_d, STATE MEASUREMENT STANDARD DEVIATION $\sigma_{\boldsymbol{x}}$ AND RESULTING TIME WHEN LEARNING IS STOPPED T_s

	low bandwidth	medium bandwidth	high bandwidth	large delay	state noise
B [samples/s]	1500	3000	10000	10000	1500
T_d [s]	0.1	0.1	0.1	1.0	0.1
$\sigma_{\boldsymbol{x}}$ [rad]	—	—	—	—	0.0001
T_s [s]	44.67	60.04	—	30.81	43.13

For the simulation, we assume a rigid kinematic coupling between the human and exoskeleton arm, which allows the modelling of both as one kinematic chain consisting of four DoFs. The exoskeleton model is based on the design described in [18], whilst the model parameter for the human are chosen according to anthropometric tables [19]. Here, the reference is set to $70\,\mathrm{kg}$ and $1.75\,\mathrm{m}$. As illustrated in Fig. 5, the goal is to track an elliptic trajectory with the hand of the human by employing the learning-based feedback linearizing control law (19). Each period of the ellipse takes $T_p = 6\mathrm{s}$, the simulation runs at $1\mathrm{kHz}$, and we consider a memory constraint of $\bar{M} = 4000$ data pairs for the local memory. Streaming data for online learning is generated with noise standard deviation $\sigma_y = 0.05$ at a sampling rate of $100\mathrm{Hz}$, i.e., $\tau = 10\mathrm{ms}$. Each local GP model can contain a maximum of $\bar{N} = 100$ training points and the hyperparameters are set to $\sigma_f = 1$, $l_i = 1/l_i = 3$ for inputs corresponding to joint angles/angular velocities. Algorithm 1 is run with temporal discretization $\Delta t = 10\mathrm{ms}$ and $N_s = 1000$ random samples. Finally, the control gains are set to $k_c = 400$ and $\lambda = 1$.

In order to investigate the dependency of the tracking accuracy and memory occupation on the network bandwidth B and time delay T_d, we compare networked LoG-GP controllers under different simulation conditions as outlined in Table I. In this comparison, we also consider the case of noisy state measurements to demonstrate the robustness of the proposed method against sufficiently small noise on training inputs $\boldsymbol{x}_n$. Moreover, we employ a LoG-GP without memory constraints, i.e., $\bar{M} = \infty$, as baseline to illustrate the absence of a performance loss of the networked LoG-GP when a sufficiently high bandwidth is available. The average update time for the LoG-GP is $0.3\mathrm{ms} < \tau$ in all simulations, and the resulting curves for the evolution of the local memory occupation are depicted in Fig. 6. Since the LoG-GP has low accuracy during the first period, the tracking error bound $\vartheta(\kappa)$ is large during the first 6s, such that all data is required on the local system. After this period, the different curves exhibit the behavior discussed in Section III-D: The lower the bandwidth B, the faster the local memory consumption grows. Moreover, an increase in time delay T_d causes a

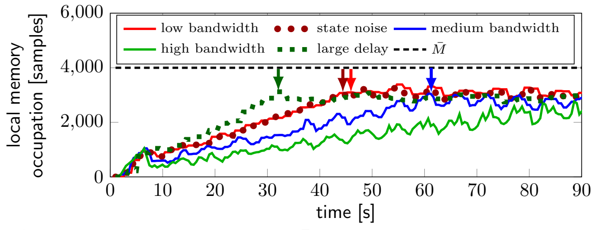


Fig. 6. The higher the bandwidth B, the longer the LoG-GP can learn before the number of training pairs in the local memory reaches the limitations. Large time delay T_d causes a significantly earlier stopping of learning, as indicated by the arrows.

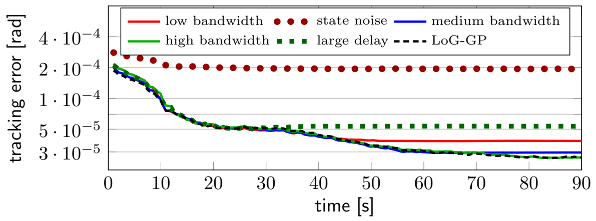


Fig. 7. When the memory limitation is reached and the learning process stops, the tracking error stagnates. Since higher bandwidths B allow learning for a longer time, larger values of B yield lower tracking errors eventually. While learning is not stopped, networked LoG-GPs ensure the same tracking accuracy as LoG-GPs without any constraints. Overall, online learning significantly improves the tracking accuracy over the baseline case without model learning, which is not depicted since it permanently exceeds $2 \cdot 10^{-2}$.

significantly faster growing memory occupation. Due to the limited local memory, this leads to an early stop in learning at the times depicted in Table I, after which the memory occupation stagnates. Note that the state measurement noise has effectively no impact on the local memory consumption.

The stagnation has an immediate effect on the evolution of the tracking error, as illustrated in Fig. 7. While the feedback linearizing control law (19) with networked LoG-GP model achieves the same improvement in tracking accuracy as with the unconstrained LoG-GP when model updates are performed, the tracking performance ceases to improve and effectively remains constant after the learning has stopped. The same behavior can be observed with noisy state measurements, but the tracking error exhibits an offset since the noise also affects the linear feedback control ν. Due to the continual learning of the networked LoG-GP with a high bandwidth connection, the corresponding evolution of the tracking error is visually identical to the curve resulting from usage of the LoG-GP without memory constraint. This clearly demonstrates that the proposed approach allows a transfer of data to the cloud without any loss in performance when sufficient transmission bandwidth is available. Moreover, even when online learning has to be stopped early, it still yields a significant improvement in tracking accuracy compared to the baseline case without model learning, where a stationary error of $\approx 2 \cdot 10^{-2}$ rad has been observed. This strongly underlines the practical advantages of online model inference for model-based control despite resource constraints.

VI. CONCLUSION

This paper presents a novel networked online learning approach for control of safety-critical systems with local resource constraints based on Gaussian process regression. By employing a tree-structured local GP approximation, relevant local models for control can be efficiently determined in a sampling-based fashion. This is exploited in the design of an effective data transmission scheme, which ensures the timely availability of data in the local computing unit. The effectiveness of the proposed networked online learning approach is demonstrated in a simulation of a robotic exoskeleton.

Appendix

A. Proof of Theorem 1

Proof: Due to Assumption 1, at each time t, the tracking error $\boldsymbol{e}$ is bounded by $\vartheta(\kappa_t)$. It is straightforward to see that $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ is Lipschitz continuous, such that

$$\|\boldsymbol{e}(t)\| \leq \xi - 2\zeta \quad \forall t \in \left[t_1 + \frac{2j-1}{2}\Delta t, t_1 + \frac{2j+1}{2}\Delta t\right] \tag{23}$$

and consequently $\mathbb{T}_{t_1}^{t_2} \subset \bigcup_{j=1}^{\lceil \frac{t_2-t_1}{\Delta t} \rceil} \mathbb{B}_{\xi-2\zeta}$. Therefore, it remains to show that the set of active models for time $t_1 + j\Delta t$ defined as $\mathbb{A}_{t_1+j\Delta t} = \bigcup_{\boldsymbol{x}\in\mathcal{B}_{\xi-2\zeta}} \mathbb{A}_{\boldsymbol{x}}$ is overapproximated by Algorithm 1. For this purpose, choose any model $l \in \mathbb{A}_{t+j\Delta t}$. Then, the intersection between the active region $\mathbb{X}_l$ of this model and the ball $\mathcal{B}_\xi$ has a volume of at least $\pi^{d_x/2}(\min\{r_{\min}, \zeta\})^{d_x}/\Gamma(\frac{d_x}{2}+1)$, where $\Gamma : \mathbb{R}_+ \to \mathbb{R}_+$ denotes Euler's gamma function. Therefore, the probability of a sample $\boldsymbol{x}^{(i)} \sim \mathcal{U}(\mathcal{B}_\xi)$ being in the active region of model l can be bounded by

$$P(\omega_l(\boldsymbol{x}^{(i)}) > 0 | l \in \mathbb{A}_{t_1+j\Delta t}) \geq \min\{r_{\min}^{d_x}, \zeta^{d_x}\}/\xi^{d_x}. \tag{24}$$

The probability of none of the N_s samples falling into the active region $\mathbb{X}_l$ is consequently upper bounded by $(1 - P(\omega_l(\boldsymbol{x}^{(i)}) > 0 | l \in \mathbb{A}_{t_1+j\Delta t}))^{N_s}$, such that (15) follows from the union bound over all time steps and all models $l \in \mathbb{L}$. ■

B. Proof of Lemma 2

Proof: Satisfaction of the memory constraint (4) implies that the transmission of $\mathbb{D}_j$, $j \in \mathbb{N}$, can be achieved with time $T_{\mathrm{trans}} \leq \frac{\bar{M}}{2B} + T_d$. Hence, we have $T_{\mathrm{access}} = \frac{\Delta T}{2} \geq T_{\mathrm{trans}}$, guaranteeing satisfaction of the network constraint (5). ■

C. Proof of Theorem 2

Proof: Since the cardinality of $\mathbb{D}_{j+q}$ can be bounded by $|\mathbb{D}_{j+q}| \leq |\mathbb{D}_j| + \bar{m}$, memory constraints are satisfied as long as $|\mathbb{D}_j| \leq \bar{M}/2 - \bar{m}$. Therefore, ι as defined in (16) ensures that the memory constraint (4) is satisfied, which implies the satisfaction of the network constraint (5) due to Lemma 2. ■

D. Proof of Theorem 3

Proof: In order to prove the ultimate bound, we employ the Lyapunov function $V(\boldsymbol{e}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e}$, where $\boldsymbol{P}$ is a positive definite matrix as $\boldsymbol{\lambda}$ is a Hurwitz vector. The derivative of the Lyapunov function is guaranteed to satisfy

$$\dot{V}(\boldsymbol{e}) = -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + 2\boldsymbol{e}^T\boldsymbol{p}_d(k_c)\left(f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})\right). \tag{25}$$

Due to Lipschitz continuity, we obtain

$$\begin{aligned}\dot{V}(\boldsymbol{e}) \leq& -\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{e}\|^2 + 2\|\boldsymbol{e}\|\|\boldsymbol{p}_d(k_c)\|(L_f + L_{\tilde{\mu}})\|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}\| \\ &+ 2\|\boldsymbol{e}\|\,\|\boldsymbol{p}_d(k_c)\|\,|(f(\boldsymbol{x}_{\mathrm{ref}}) - \tilde{\mu}(\boldsymbol{x}_{\mathrm{ref}}))|,\end{aligned} \tag{26}$$

where the Lipschitz constant $L_{\tilde{\mu}}$ in (21) follows directly from Lipschitz continuity of the individual mean functions resulting from the L_k-Lipschitz kernel $k(\cdot,\cdot)$ [9]. Due to Lemma 1, the error between the unknown function $f(\cdot)$ and the LoG-GP mean $\tilde{\mu}(\cdot)$ can be bounded, such that we obtain

$$\begin{aligned}\dot{V}(\boldsymbol{e}) \leq& -\left(\lambda_{\min}(\boldsymbol{Q}) - 2\|\boldsymbol{p}_d(k_c)\|\left(L_f + L_{\tilde{\mu}}\right)\right)\|\boldsymbol{e}\|^2 \\ &+ 2\|\boldsymbol{e}\|\,\|\boldsymbol{p}_d(k_c)\|\,\eta\left(\boldsymbol{x}_{\mathrm{ref}}\right).\end{aligned} \tag{27}$$

Due to (21), the Lyapunov derivative is negative for

$$\|\boldsymbol{e}\| > \frac{2\|\boldsymbol{p}_d(k_c)\|\,\eta\left(\boldsymbol{x}_{\mathrm{ref}}\right)}{\lambda_{\min}(\boldsymbol{Q}) - 2\|\boldsymbol{p}_d(k_c)\|\left(L_f + L_{\tilde{\mu}}\right)}. \tag{28}$$

Since the ultimately bounded set is given by the smallest sub-level set of $V(\cdot)$ which contains the ball defined through (28), we can directly determine it as $\{\boldsymbol{e} : V(\boldsymbol{e}) \leq \vartheta(\kappa_t)^2\lambda_{\min}(\boldsymbol{P})\}$ due to the quadratic structure of $V(\cdot)$. Over-approximating this set by a ball concludes the proof. ■

References

[1] A. Sahoo, S. K. Dwivedy, and P. Robi, "Advancements in the field of autonomous underwater vehicle," *Ocean Eng*, vol. 181, pp. 145–160, 2019.

[2] O. Andersson, M. Wzorek, and P. Doherty, "Deep learning quadcopter control via risk-aware active learning," in *AAAI Conf Artif Intell*, 2017, pp. 3812–3818.

[3] U. Martinez-Hernandez, B. Metcalfe, T. Assaf, L. Jabban, J. Male, and D. Zhang, "Wearable assistive robotics: A perspective on current challenges and future trends," *Sensors*, vol. 21, no. 20, pp. 1–19, 2021.

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

[5] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *Int Conf Mach Learn*, 2015, pp. 1481–1490.

[6] M. F. Huber, "Recursive Gaussian process: On-line regression and learning," *Pattern Recognit Lett*, vol. 45, pp. 85–91, 2014.

[7] T. Bui, C. Nguyen, and R. Turner, "Streaming sparse Gaussian process approximations," in *Adv Neural Inf Process Syst*, 2017, pp. 3300–3308.

[8] A. Gijsberts and G. Metta, "Real-time model learning using incremental sparse spectrum Gaussian process regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[9] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for Gaussian process regression with application to safe control," in *Adv Neural Inf Process Syst*, 2019, pp. 659–669.

[10] A. Lederer, A. Ordóñez Conejo, K. Maier, W. Xiao, J. Umlauft, and S. Hirche, "Gaussian process-based real-time learning for safety-critical applications," in *Int Conf Mach Learn*, 2021, pp. 6055–6064.

[11] J. Umlauft and S. Hirche, "Feedback linearization based on Gaussian processes with event-triggered online learning," *IEEE Trans Automat Contr*, vol. 65, no. 10, pp. 4154–4169, 2019.

[12] A. Capone and S. Hirche, "Backstepping for partially unknown nonlinear systems using Gaussian processes," *IEEE Control Syst Lett*, vol. 3, no. 2, pp. 416–421, 2019.

[13] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. A. Theodorou, "$\mathcal{L}_1$-$\mathcal{GP}$: $\mathcal{L}_1$ adaptive control with Bayesian learning," in *Learn Dyn & Cont*, 2020, pp. 1–15.

[14] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.

[15] A. Lederer, A. Capone, J. Umlauft, and S. Hirche, "How training data impacts performance in learning-based control," *IEEE Control Syst Lett*, vol. 5, no. 3, pp. 905–910, 2021.

[16] A. Lederer, J. Umlauft, and S. Hirche, "Posterior variance analysis of Gaussian processes with application to average learning curves," 2019. [Online]. Available: http://arxiv.org/abs/1906.01404

[17] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev*, vol. 59, no. 1, pp. 65–98, 2017.

[18] E. Trigili, S. Crea, M. Moise, A. Baldoni, M. Cempini, G. Ercolini, D. Marconi, F. Posteraro, M. Carrozza, and N. Vitiello, "Design and experimental characterization of a shoulder-elbow exoskeleton with compliant joints for post-stroke rehabilitation," *IEEE ASME Trans Mechatron*, vol. 24, no. 4, pp. 1485–1496, 2020.

[19] R. Drillis, R. Contini, and M. Bluestein, "Body segment parameters: A survey of measurement techniques," *Artif Limbs*, vol. 8, pp. 44–66, 1964.