Bridging Reinforcement Learning and Iterative Learning Control: Autonomous Reference Tracking for Unknown, Nonlinear Dynamics

Michael Meindl, Dustin Lehmann, and Thomas Seel

***Abstract*— This work addresses the problem of reference tracking in autonomously learning agents with unknown, nonlinear dynamics. Existing solutions require model information or extensive parameter tuning, and have rarely been validated in real-world experiments.**

We propose a learning control scheme that learns to approximate the unknown dynamics by a Gaussian Process (GP), which is used to optimize and apply a feedforward control input on each trial. Unlike existing approaches, the proposed method neither requires knowledge of the system states and their dynamics nor knowledge of an effective feedback control structure. All algorithm parameters are chosen automatically, i.e. the learning method works plug and play. The proposed method is validated in extensive simulations and real-world experiments. In contrast to most existing work, we study learning dynamics for more than one motion task as well as the robustness of performance across a large range of learning parameters. The method's plug and play applicability is demonstrated by experiments with a balancing robot, in which the proposed method rapidly learns to track the desired output. Due to its model-agnostic and plug and play properties, the proposed method is expected to have high potential for application to a large class of reference tracking problems in systems with unknown, nonlinear dynamics.

***Index Terms*— Autnomous systems, Gaussian processes, Iterative learning control, Nonlinear systems, Reinforcement learning, Robot learning**

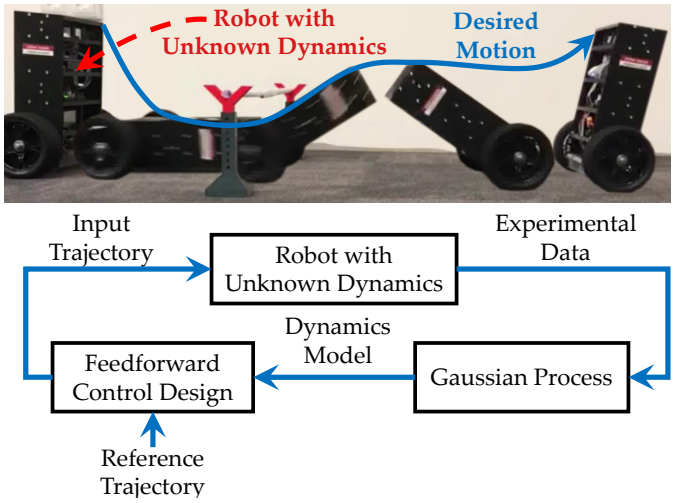

Fig. 1. A robot with unknown dynamics is meant to track a reference trajectory leading to a desired, highly dynamic motion (top). On each iteration, the proposed learning method determines, based on experimental data, a Gaussian Process model, which is in turn used to design and apply a feedforward control input (bottom).

I. Introduction

Recent developments in robotic technology remarkably contribute to the quality of human live: Hazardous tasks on rescue missions are handled by mobile robots that rifle through wreckage to locate people in need of help [1]. Advances in medical robotics strive for minimizing complications during surgery [2]. And the combination of exoskeletons and control algorithms aims for a future in which people struck by disability can walk again [3]. The way to such accomplishments is paved by control techniques that enable robots to precisely perform agile and dynamic motions.

For example, model predictive control can achieve accurate motion if a precise model of the dynamics is available [4]–[6]. Requirements regarding the model's precision can be relaxed by robust or adaptive control techniques if the uncertainties comply with preset assumptions [7]–[9]. Under similar conditions, Iterative Learning Control (ILC) can overcome model uncertainties and unknown disturbances by learning from errors of previous trials [10], [11]. However, all of these control approaches require system-specific prior knowledge to craft a suited model, controller, or learning configuration. Autonomy requires a methodology that self-reliantly learns a solution to the control problem without requiring any system-specific prior knowledge. In particular, Reinforcement Learning (RL) techniques have been employed to solve complex motion tasks without requiring any prior information. However, RL solutions typically suffer from two major drawbacks: First, most of the results were obtained in simulated environments [12]–[14]. Second, the few real-world applications typically required multiple hours of learning, and the resulting controllers can be prone to failure [15]–

This manuscript has been submitted to review on 19th of July 2021.

Michael Meindl is with the Embedded Mechatronics Laboratory at Hochschule Karlsruhe, Germany, and the Control Systems Group at Technische Universität Berlin, Germany (e-mail:michael.meindl@hs-karlsruhe.de).

Dustin Lehmann is with the Control Systems Group at Technische Universität Berlin, Germany (e-mail:lehmann@control.tu-berlin.de)

Thomas Seel is with the Departement Artificial Intelligence in Biomedical Engineering at Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany (e-mail: thomas.seel@fau.de}).

[18]. A major breakthrough with respect to robustness and data-efficiency was achieved by hybrid techniques that learn parameter-free models, namely Gaussian Processes (GP), but also employ system-specific information such as knowledge of a state vector and an effective state feedback structure [19]. In the prominent example of *PILCO* [20], experimental data are used to approximate the unknown dynamics by a GP, which is used to determine the optimal parameters of a state feedback controller. By this approach, an inverted pendulum on a cart could be swung up and stabilized in 12 seconds of system interaction [20]. However, in the context of autonomous motion learning, GP-based learning methods still suffer from two drawbacks: First, previously proposed methods only solve set-point stabilization tasks, which do not enable robots to perform challenging, dynamic maneuvers. These require reference tracking. Second, GP-based learning techniques still require system-specific prior knowledge such as the configuration of cost functions, a state vector that fully describes the system dynamics, and a control structure that is effective with respect to the problem at hand. Hence, the methods are not suited for plug & play learning of highly dynamic robotic motions.

The present contribution proposes a GP-based learning method for autonomously solving highly dynamic reference tracking tasks in systems with unknown, nonlinear dynamics. The proposed method autonomously determines all of its necessary parameters such that plug & play application becomes feasible. The method's capability to rapidly learn solutions to various reference tracking tasks while not requiring any system-specific prior knowledge is validated by extensive simulations and real-world experiments using a two-wheeled inverted pendulum robot.

A. Related Work

Learning for control has been considered in a large body of literature that can be categorized by (i) the considered control problem respectively control strategy, (ii) necessary system-specific prior knowledge, and (iii) speed of learning. Reinforcement Learning (RL) techniques typically do not require any model and only few learning parameters such as step sizes or weights in cost functions. Furthermore, general RL approaches such as genetic algorithms [21] or policy gradient approaches [22] can be applied to arbitrary control problems with unknown, nonlinear dynamics, but in turn require comparatively long periods of learning [20]. The speed of learning can be significantly increased if the technique is targeted towards a specific control problem and strategy such as stabilization by state feedback control, see e.g. [23], [24]. A particularly data-efficient approach are so called model-based techniques that model the unknown, nonlinear dynamics by a GP, which is then used to design a state feedback controller [25]. Some successful applications to real-world examples are the control of a single inverted pendulum [20], double inverted pendulum [26], and robotic manipulator [27]. The concept of GP-based learning control has been further investigated in a variety of contributions. Stability of feedback-controlled GPs has been analyzed [28], [29], the problem of computational and data requirements has been investigated [30], [31], and solutions for safely improving an existing feedback controller have been proposed [32]–[34]. While all of these works consider the challenging problem of efficiently learning control solutions for unknown, respectively uncertain, nonlinear dynamics, they have focused on the problem of set-point stabilization of systems, for which an effective feedback control structure is known. If the control tasks consists in performing a highly-dynamic motion, the achievable performance of time-domain feedback control is inherently limited by phenomena such as unknown delays, measurement noise, or non-minimum phase dynamics. To overcome the performance limitations of feedback control, a feedforward control component is required, see Fig. 2.

In contrast to GP-based learning techniques, Iterative Learning Control (ILC) has focused on reference tracking tasks solved by feedforward control [35], [36]. Model-based techniques like norm-optimal or $\mathcal{H}_\infty$ ILC automatically determine the learning parameters, but require a model of the linear plant dynamics [37]–[39]. Model-free approaches like PD-ILC do not require a model but learning parameters that are typically tuned in experiment [40]. The concepts of PD-type [41] and norm-optimal [42] ILC have been extended to the case of nonlinear dynamics, but assume the dynamics to be known. To relax requirements with respect to available model information, recent research has focused on so called data-driven ILC (DD-ILC) [43], which does not require a model of the plant. In the case of nonlinear, unknown dynamics, DD-ILC methods typically employ dynamic linearization of the plant dynamics and estimate the gradient of said linearization [44]–[46]. Alternatively, neural networks (NN) have been employed in DD-ILC to model the unknown dynamics [47], [48]. While DD-ILC methods can solve reference tracking tasks without requiring a plant model, some system-specific prior knowledge is required as, e.g., the signs of the dynamic linearization [44]–[46] or the layout of a suited neural network [47], [48].

In summary, we conclude that reference tasks in systems with unknown, nonlinear dynamics can be solved by DD-ILC methods, which, however, require system-specific prior knowledge such that autonomous plug & play application is generally not possible. In contrast, set-point stabilization problems can be solved by GP-based learning that assume comparatively little system-specific prior knowledge. However, in the context of reference tracking tasks, GP-based learning methods suffer from the inherent limitations of feedback control. To the best of our knowledge, there exists no learning method that autonomously solves reference tracking tasks for unknown, nonlinear systems, employs feedforward control to overcome the limitations of feedback control, and does not require system-specific prior knowledge such that autonomous plug & play application is enabled.

B. Contributions

In this contribution, a GP-based ILC scheme is proposed that autonomously solves reference tracking tasks in systems with unknown, nonlinear dynamics. The proposed method

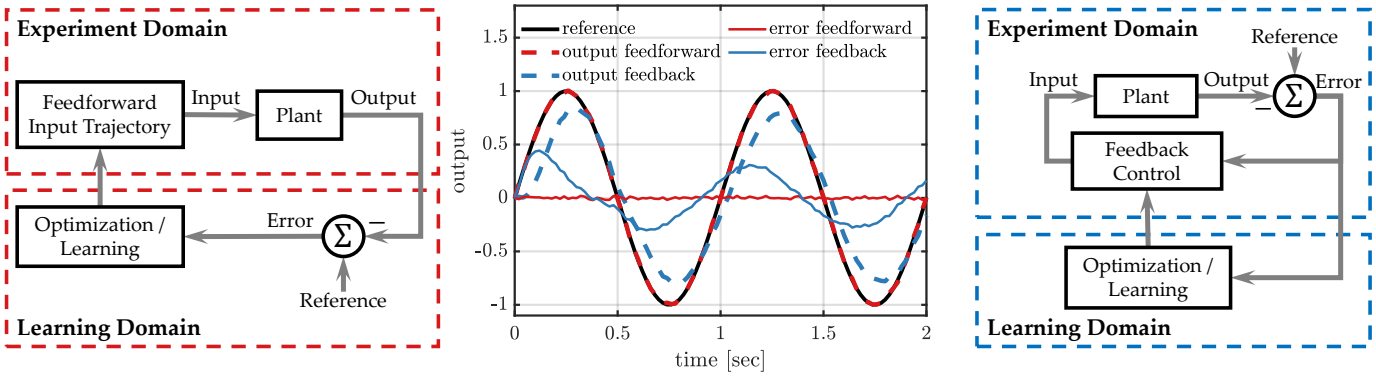

Fig. 2. Comparison of feedforward (left) and feedback learning control (right) for reference tracking with a system affected by input delay and measurement noise: Feedforward, unlike feedback control, achieves almost perfect tracking (middle). Hence, the learning method proposed in this work employs feedforward control.

includes a procedure to autonomously determine necessary parameters and enable plug & play application. Since the proposed method employs a GP to model the input/output dynamics, only the output variable, instead of an entire state vector, has to be known and measured. To overcome the inherent limitations of feedback control, the proposed method employs feedforward control.

The proposed method is first validated by extensive simulations of a two-wheeled inverted pendulum robot (TWIPR), in which precise tracking is achieved after a small number of trials. Unlike existing approaches, the proposed method is not only verified for a single, well-chosen parameter configuration but for a wide range of parameter combinations such that robustness with respect to the autonomously determined parameters is ensured. In contrast to a variety of contributions, in which validation was restricted to simulated environments, the proposed method's capability of solving real-world reference tracking tasks in a plug & play manor is validated by experiments on a TWIPR, see Fig. 1.

C. Notation

Let $\mathbb{N}_{\geq 0}$ and $\mathbb{N}$ denote the set of nonnegative, respectively positive, integers. Let $\mathbb{R}$ denote the set of real numbers. Vectors are in lower-case letters and bold type, e.g., $\mathbf{v}$. Matrices are in upper-case letters and bold type, e.g., $\mathbf{A}$. The i^{th} component of a vector $\mathbf{v}$ is denoted by $[\mathbf{v}]_i$. The element in row i and column j of matrix $\mathbf{A}$ is denoted by $[\mathbf{A}]_{ij}$. Let $\|\mathbf{v}\|$ denote a norm of vector $\mathbf{v}$, and $\|\mathbf{A}\|$ the corresponding, induced matrix norm of matrix $\mathbf{A}$. A specific example is the infinity norm, denoted by $\|\cdot\|_\infty$.

II. Problem Formulation

Consider an autonomous system that can repeatedly attempt a reference tracking task, as, e.g., a robot trying to perform a desired maneuver. We assume that the system's output, e.g., a joint angle or position, can be influenced by an input signal, e.g., a motor torque, and that the relation of these variables is deterministic, causal, and time-invariant. However, we do *not* assume that a model of the dynamics is available and we *do* assume the general case of nonlinear dynamics.

Formally, consider a discrete-time, single-input, single-output, repetitive system with a finite trial duration of $N \in \mathbb{N}$ samples, and, on trial $j \in \mathbb{N}_{\geq 0}$ and sample $n \in [1, N]$, output variable $y_j(n) \in \mathbb{R}$, respectively input variable $u_j(n) \in \mathbb{R}$. The samples are collected in the so called output trajectory $\mathbf{y}_j \in \mathbb{R}^N$, respectively input trajectory $\mathbf{u}_j \in \mathbb{R}^N$, i.e., $\forall j \in \mathbb{N}_{\geq 0}$,

$$\mathbf{y}_j := \begin{bmatrix} y_j(1) & y_j(2) & \dots & y_j(N) \end{bmatrix}^T \tag{1}$$

$$\mathbf{u}_j := \begin{bmatrix} u_j(1) & u_j(2) & \dots & u_j(N) \end{bmatrix}^T . \tag{2}$$

Without loss of generality, the dynamics can be written in the lifted form

$$\forall j \in \mathbb{N}_{\geq 0}, \qquad \mathbf{y}_j = \boldsymbol{p}(\mathbf{u}_j) , \tag{3}$$

where $\boldsymbol{p}$ is the unknown dynamics function. The task consists of updating the input $\mathbf{u}_j$ from trial to trial such that the output $\mathbf{y}_j$ converges to the desired reference trajectory $\mathbf{r} \in \mathbb{R}^N$. Tracking performance is measured by the error trajectory

$$\forall j \in \mathbb{N}_{\geq 0}, \qquad \mathbf{e}_j := \mathbf{r} - \mathbf{y}_j \tag{4}$$

and root-mean-squared error (RMSE)

$$\forall j \in \mathbb{N}, \qquad e_j^{\text{RMS}} := \sqrt{\sum_{i=1}^{N} \frac{[\mathbf{e}]_i^2}{N}} . \tag{5}$$

The problem considered in this work consists in developing a learning method that updates the input trajectory on each trial such that the RMSE decreases. Learning performance is judged based on the progression of the RMSE through trials, and the RMSE shall decline quickly and monotonically. The learning method must not require any a priori model information on the plant dynamics. To support plug & play application, the method must autonomously determine necessary parameters. Furthermore, the method must provide a fair degree of robustness with respect o autonomously determined parameters.

III. Proposed Learning Method

We address the proposed problem by an iterative learning scheme, in which each iteration consists of three steps. First, a

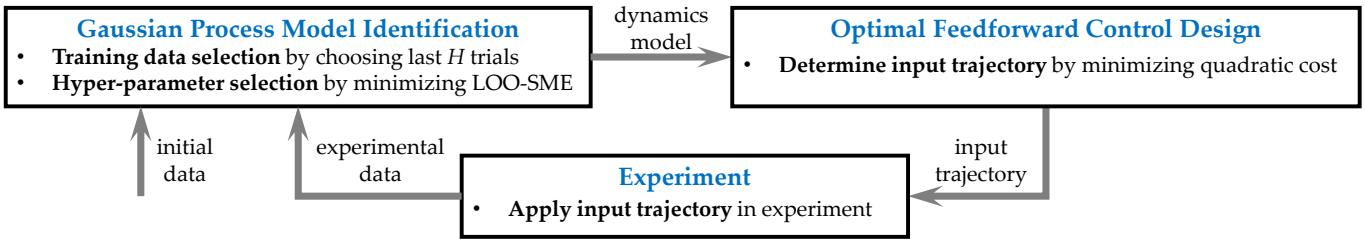

Fig. 3. Overview of the proposed learning method: First a Gaussian Process (GP) model is identified, which in turn is used to determine an input trajectory via optimization. The resulting input trajectory is applied in experiment yielding new data to refine the GP model.

parameter-free model of the plant dynamics is identified using the experimental data of previous trials, see Section III-A. To accommodate for possibly nonlinear dynamics, a generic GP model is employed, which predicts the output trajectory for a given input trajectory. Second, the updated input trajectory is determined by solving an optimal feedfoward control problem based on the GP model, see Section III-B. Third, the updated input trajectory is applied to the plant and resulting data is in turn used to refine the GP model. The structure of the proposed learning scheme is depicted in Fig. 3. To enable plug & play application, the proposed method autonomously determines necessary parameters, see Section III-C.

A. Gaussian Process Model

We propose a Gaussian Process (GP) model, formally a function $\boldsymbol{m} : \mathbb{R}^N \mapsto \mathbb{R}^N$, that predicts the plant's output trajectory $\hat{\mathbf{y}} \in \mathbb{R}^N$ based on an input trajectory $\mathbf{u} \in \mathbb{R}^N$, where the trial index is omitted for sake of notational simplicity.

Let $f(\mathbf{v}) : \mathbb{R}^D \mapsto \mathbb{R}$ denote the unknown target function that that depends on the regression vector $\mathbf{v} \in \mathbb{R}^D$. Predictions are based on $K \in \mathbb{N}$ observations $z_k \in \mathbb{R}$ stemming from

$$\forall k \in [1, K], \quad z_k = f(\mathbf{v}_k) + w_k \quad | \; w_k \sim \mathcal{N}(0, \sigma_w^2) \,. \tag{6}$$

The K observation pairs $(z_k, \mathbf{v}_k)$ are collected in the observation training vector $\bar{\mathbf{z}} \in \mathbb{R}^K$ and regression training matrix $\bar{\mathbf{V}} \in \mathbb{R}^{D \times K}$, i.e.,

$$\bar{\mathbf{z}} := \begin{bmatrix} z_1 & z_2 & \dots & z_K \end{bmatrix}^T , \tag{7}$$

$$\bar{\mathbf{V}} := \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_K \end{bmatrix} . \tag{8}$$

The kernel function of two regression vectors $\mathbf{v} \in \mathbb{R}^D$ and $\hat{\mathbf{v}} \in \mathbb{R}^D$ is denoted by $k_{\mathbf{v}\hat{\mathbf{v}}} \in \mathbb{R}$. The kernel matrix of two regression matrices $\mathbf{V} \in \mathbb{R}^{D \times K}$, $\hat{\mathbf{V}} \in \mathbb{R}^{D \times \hat{K}}$, which are assembled according to (8), is denoted by $\mathbf{K}_{\mathbf{V}\hat{\mathbf{V}}} \in \mathbb{R}^{K \times \hat{K}}$ and has entries $\left[\mathbf{K}_{\mathbf{V}\hat{\mathbf{V}}}\right]_{ij} = k_{\mathbf{v}_i \hat{\mathbf{v}}_j}$.

Given $F \in \mathbb{N}$ test regression vectors assembled in the regression matrix $\mathbf{V} \in \mathbb{R}^{D \times F}$, the predicted mean $\boldsymbol{\mu} \in \mathbb{R}^F$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{F \times F}$ are given by

$$\boldsymbol{\mu} = \mathbf{K}_{\mathbf{V}\bar{\mathbf{V}}} \left[\mathbf{K}_{\bar{\mathbf{V}}\bar{\mathbf{V}}} + \sigma_w^2 \mathbf{I}\right]^{-1} \bar{\mathbf{z}} \tag{9}$$

$$\boldsymbol{\Sigma} = \mathbf{K}_{\mathbf{V}\mathbf{V}} - \mathbf{K}_{\mathbf{V}\bar{\mathbf{V}}} \left[\mathbf{K}_{\bar{\mathbf{V}}\bar{\mathbf{V}}} + \sigma_w^2 \mathbf{I}\right]^{-1} \mathbf{K}_{\bar{\mathbf{V}}\mathbf{V}} \,. \tag{10}$$

The general GP framework can be employed in three ways to model the unknown dynamics (3), where the model characteristics are determined by the definition of observation variable z, regression vector $\mathbf{v}$, and kernel function k. First, we exploit the dynamics' time-invariance by employing a single GP for predicting each output sample. Hence, the observation variable and regression vector are time dependent, i.e., $\forall n \in [1, N]$, z_n, $\mathbf{v}_n$. Secondly, the model can be chosen to be of type finite impulse response (FIR), infinite impulse response (IIR), or state space (SS). A FIR model results when the regression vector consists of the current and all previous input samples, i.e.,

$$\forall n \in [1, N], \quad \mathbf{v}_n := [u(n) \; \dots \; u(1) \; \dots \; 0 \; \dots \; 0]^T \,. \tag{11}$$

An IIR model results when the regression vector consists of the current input and the $P \in \mathbb{N}$ previous output samples, i.e.,

$$\forall n \in [1, N], \quad \mathbf{v}_n := [u(n) \; y(n-1) \; \dots \; y(n-P) \; 0 \; \dots \; 0]^T \,. \tag{12}$$

A SS model results when the regression vector consists of the current input and the previous state sample, i.e.,

$$\forall n \in [1, N], \quad \mathbf{v}_n := [u(n) \; \mathbf{x}^T(n-1)]^T \,. \tag{13}$$

The SS model requires multiple GPs with each predicting the progression of a single state variable [49], which not only increases computational complexity, but also requires measurement of the full state vector. Furthermore, IIR and SS models require roll-out predictions meaning that previous predictions are required for predicting the current sample [49], which also increases model complexity. In contrast, the FIR model only requires a single batch prediction according to (9). We, hence, employ a FIR model and the regression vector is defined as in (11).

We further choose difference-predictions, i.e.,

$$\forall n \in [1, N], \quad z_n := y(n) - y(n-1) \quad | \; y(-1) = 0 \,, \tag{14}$$

which, compared to absolute predictions, increase the model's capability of extrapolation, see [49].

As kernel function, we employ a squared-exponential kernel (SEK)

$$k(\mathbf{v}, \tilde{\mathbf{v}}) = \exp\left(-\frac{1}{2l^2}(\mathbf{v} - \tilde{\mathbf{v}})^T(\mathbf{v} - \tilde{\mathbf{v}})\right) , \tag{15}$$

where $l \in \mathbb{R}$ is a so called length scale.

Remark 1. SEKs allow a GP to model arbitrary target functions. In the context of dynamic systems, a SEK leads to a nonlinear, time-invariant (NTI) model. Using a squared kernel instead, as e.g.

$$k_{\mathbf{v}\tilde{\mathbf{v}}} := \mathbf{v}^T \tilde{\mathbf{v}} \,, \tag{16}$$

results in a linear, time-invariant (LTI) model. If the plant dynamics are linear, one may employ a squared kernel to decrease computational complexity in comparison to a NTI model.

To predict an output trajectory $\hat{\mathbf{y}}$ for an arbitrary input trajectory $\mathbf{u}$, the latter is used to determine N regression vectors $\mathbf{v}_n$, $n \in [1, N]$, according to (11), which are assembled in a regression matrix $\mathbf{V}$ according to (8). The predicted mean vector $\boldsymbol{\mu}$ follows from (9). By (14), $\boldsymbol{\mu}$ contains difference predictions such that the components of $\hat{\mathbf{y}}$ follow from the cumulative sum of $\boldsymbol{\mu}$, i.e.

$$\forall n \in [1, N], \qquad [\hat{\mathbf{y}}]_n = \sum_{i=1}^{n} [\boldsymbol{\mu}]_i \,. \tag{17}$$

Mean and covariance predictions require the measurement variance σ_w^2 and length-scale l, which are so called hyper-parameters. Typically, hyper-parameters are determined based on training data, and numerous approaches have been detailed in the literature [50]. We propose selecting hyper-parameters by minimizing the so called leave-one-out squared-mean-error (LOO-SME). For each of the $k \in [1, K]$ available observations z_k, the remaining observation pairs are used to predict z_k. The LOO-SME follows from summing the squared difference between the K leave-one-out predictions and respective observations z_k. Formally, the k^{th} LOO-prediction $\hat{\mu}_k$ is given by

$$\hat{\mu}_k = z_k - \frac{\left[\left(\mathbf{K}_{\bar{\mathbf{V}}\bar{\mathbf{V}}} + \sigma_w^2 \mathbf{I}\right)^{-1} \bar{\mathbf{z}}\right]_k}{\left[\left(\mathbf{K}_{\bar{\mathbf{V}}\bar{\mathbf{V}}} + \sigma_w^2 \mathbf{I}\right)^{-1}\right]_{kk}} \tag{18}$$

leading to the LOO-SME e_{LOO}

$$e_{\text{LOO}} = \sum_{k=1}^{K} (z_k - \hat{\mu}_k)^2 \tag{19}$$

and the hyper-parameters $\boldsymbol{\theta} := \begin{bmatrix} \sigma_w^2 & l \end{bmatrix}^T$ follow from

$$\boldsymbol{\theta} = \underset{\tilde{\boldsymbol{\theta}}}{\operatorname{argmin}} \; e_{\text{LOO}} \,. \tag{20}$$

The optimization problem (20) can be solved efficiently, because analytic expressions of the gradients are available, see [50].

Remark 2. Determining hyper-parameters by LOO-SME minimization is a rather uncommon choice as the variance of the predictions is not taken into account [50]. However, comparison of LOO-SME minimization with state of the art methods such as evidence maximization, see [50], has shown that the former leads to more reliable performance of the learning scheme proposed in this work. We assume that this is due to the proposed learning scheme solely relying on the GP's mean prediction and, hence, hyper-parameters should be chosen such that the accuracy of mean predictions is maximized.

GP predictions are known to become computational expensive with increasing amounts of training data [51]. To overcome this limitation, various data selection approaches have been proposed to reduce training data to a tractable amount, see, e.g., [51], [52]. In the present work, we simply propose limiting the training data to the last $H \in \mathbb{N}$ trials.

B. Optimal Feedforward Control

After the GP model has been identified, it is used to determine an input trajectory that leads to a smaller difference between reference and output trajectory than the input trajectories of previous trials. We propose an optimal control design, where the input is chosen to minimize a quadratic cost criterion. The latter not only considers the predicted tracking error, but also the change of the input trajectory to avoid model inversion and, hence, increase robustness with respect to the uncertainty of the current trial's model. Formally, the cost criterion is given by

$$\forall j \in \mathbb{N}_{\geq 0}, \quad J(\mathbf{u}_{j+1}) = q \left\| \mathbf{r} - \hat{\mathbf{y}}(\mathbf{u}_{j+1}) \right\|_2^2 + s \left\| \mathbf{u}_{j+1} - \mathbf{u}_j \right\|_2^2 \,, \tag{21}$$

where $q, s \in \mathbb{R}_{>0}$ are scalar weights. On each trial, the updated input trajectory $\mathbf{u}_{j+1}$ is chosen to minimize the cost criterion, i.e.,

$$\forall j \in \mathbb{N}_{\geq 0}, \quad \mathbf{u}_{j+1} = \underset{\tilde{\mathbf{u}}}{\operatorname{argmin}} \; J(\tilde{\mathbf{u}}) \,. \tag{22}$$

The optimization problem (22) can be solved efficiently since analytic expressions of the cost's gradient with respect to the input variable can be obtained [20].

C. Autonomous Parameterization

Ideally, autonomous learning methods should require neither a priori model information nor manual tuning of parameters. In contrary to previous contributions, the proposed method automatically determines necessary parameters by the procedure outlined in this section, and, as a result, plug & play application is enabled, see Fig. 4.

First, we consider the choice of the initial data set $\mathcal{I}$ that is used to determine the first GP model and consists of $I \in \mathbb{N}$ trajectory pairs $(\mathbf{y}_i, \mathbf{u}_i)$, i.e.,

$$\mathcal{I} := \{(\mathbf{y}_i, \mathbf{u}_i) \mid i \in [1, I]\} \,. \tag{23}$$

For this purpose, we first determine the largest significant frequency f_O of the reference trajectory. The frequency f_O is used to design a zero-phase low-pass filter $\boldsymbol{f}_{\text{LP}}$. The low-pass filter $\boldsymbol{f}_{\text{LP}}$ is applied to a zero mean normal distribution with covariance $\sigma_{\text{I}}^2 \mathbf{I}$, and the initial input trajectories are drawn from the resulting distribution, i.e.,

$$\forall i \in [1, I], \quad \mathbf{u}_i \sim \boldsymbol{f}_{\text{LP}} \left(\mathcal{N}(\mathbf{0}, \sigma_{\text{I}}^2 \mathbf{I}) \right) \,. \tag{24}$$

The input variance σ_{I}^2 is iteratively increased until an input trajectory drawn according to (24) leads to an output trajectory, whose maximum roughly equals the maximum of the reference, i.e.,

$$\mathbf{u} \sim \boldsymbol{f}_{\text{LP}} \left(\mathcal{N}(\mathbf{0}, \sigma_{\text{I}}^2 \mathbf{I}) \right) \implies \left\| \boldsymbol{p}(\mathbf{u}) \right\|_\infty \approx \left\| \mathbf{r} \right\|_\infty \,. \tag{25}$$

The number of initial trials is a robust parameter and is set to one, $I = 1$. The following simulations are going to show that larger values of I increase the learning method's robustness.

Once the parameters f_O, σ_{I}^2, and I have been determined, the initial trials are performed, and the weights q and s are

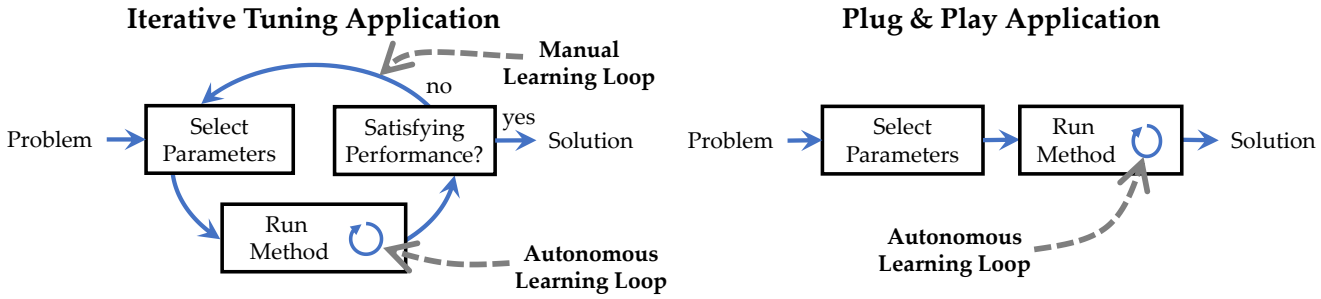

Fig. 4. Learning methods typically require some learning parameters. *left*: If no procedure for determining the parameters is provided, iterative experimental tuning is required. *right*: If a procedure for determining reliable parameters is available, plug & play application without iterative tuning is possible.

chosen based on the experimental data. The scalar q, which weights the error trajectory, is chosen as unity, i.e.,

$$q = 1\,. \tag{26}$$

The scalar s, which weighs the change in input variable, is chosen as the average squared ratio of output to input maxima over the I initial trials, i.e.,

$$s = \frac{1}{I}\sum_{i=1}^{I}\frac{\|\mathbf{y}_i\|_\infty^2}{\|\mathbf{u}_i\|_\infty^2}\,. \tag{27}$$

The procedure described in this section automatically determines all the necessary parameters without requiring any a priori information on the plant. The following simulations are going to demonstrate that the automatically determined parameters lead to the desired learning performance and that the method provides a fair degree of robustness with respect to the automatically determined parameters.

IV. Validation by Simulation

In this section, the proposed learning method is validated by simulation of a two-wheeled inverted pendulum robot (TWIPR) that is meant to perform challenging maneuvers, see Fig. 5. The TWIPR and automatic determination of learning parameters are presented in Section IV-A. Afterwards, the learning performance for three representative references is investigated in Section IV-B, and the proposed method's robustness with respect to learning parameters is verified in Section IV-C. Lastly, the effect of the weight s on the learning characteristics is studied in Section IV-D.

A. The Learning Problem

Consider the TWIPR and three desired maneuvers depicted in Fig. 5. The corresponding pitch angle reference trajectories are denoted by $\mathbf{r}_1 \in \mathbb{R}^{25}$, $\mathbf{r}_2 \in \mathbb{R}^{50}$, and $\mathbf{r}_3 \in \mathbb{R}^{71}$ and formal definitions are given in Appendix II. The robot consists of a pendulum body housing main electronics including a microcomputer, inertial measurement units, motors and accumulator. Wheels are mounted onto the motors such that the robot can drive while balancing its chassis. A model of the TWIPR's nonlinear dynamics is not available. Only an approximate, linear model of the dynamics at the upright equilibrium has been obtained, which merely suffices to design a stabilizing feedback controller, see Appendix III. Due to the imprecise model, the feedback controller can not track the references precisely, and we instead employ the proposed learning method.

Instead of a state vector, the learning method only requires knowledge of the output variable, which is given by the pitch angle, i.e.,

$$\forall n \in \mathbb{N}_{\geq 0}, \quad y(n) := \Theta(n)\,. \tag{28}$$

The input variable is given by the motor torque, $\forall n \in [1, N]$, $u_L(n) \in \mathbb{R}$.

Application of the proposed learning method requires learning parameters that are automatically determined by the procedure outlined in Section III-C. We aim at tracking pitch trajectories with a maximum of approximately 75 degrees and and spectral content roughly below 5 Hertz, i.e.,

$$\|\mathbf{r}\|_\infty \approx 75\ ^\circ \qquad f_\mathrm{O} \approx 5\ \mathrm{Hz}\,. \tag{29}$$

Based on the frequency f_O, a forward-backward, second order Butterworth filter $\boldsymbol{f}_\mathrm{LP}$ is designed, which is used for drawing initial input trajectories, see (24). To determine the input variance σ_I^2, three test input trajectories are applied to the unknown, nonlinear dynamics [53], which are implemented as a black-box simulation model. The input trajectories are drawn according to (24) with respective variances

$$\sigma_\mathrm{I}^2 \in \{1, 25, 225\}\,. \tag{30}$$

As detailed in Fig. 6, the parameterization selects $\sigma_\mathrm{I}^2 = 25$ because the resulting output trajectory has the some order of magnitude as the reference.

Next, the weights s and q of the cost function are determined. According to (26), $q = 1$ is selected. To determine the weight s, one initial trial with the previously determined input variance is performed. As detailed in Fig. 7, the value of of s directly results from (27) and the experimental data, i.e.,

$$s = \frac{\|\mathbf{y}_I\|_\infty^2}{\|\mathbf{u}_I\|_\infty^2} \approx \frac{0.4^2}{4^2} = 10^{-2}\,. \tag{31}$$

To demonstrate the data-efficiency of the proposed learning method, the training data are limited to the last five trials, i.e., $H = 5$.

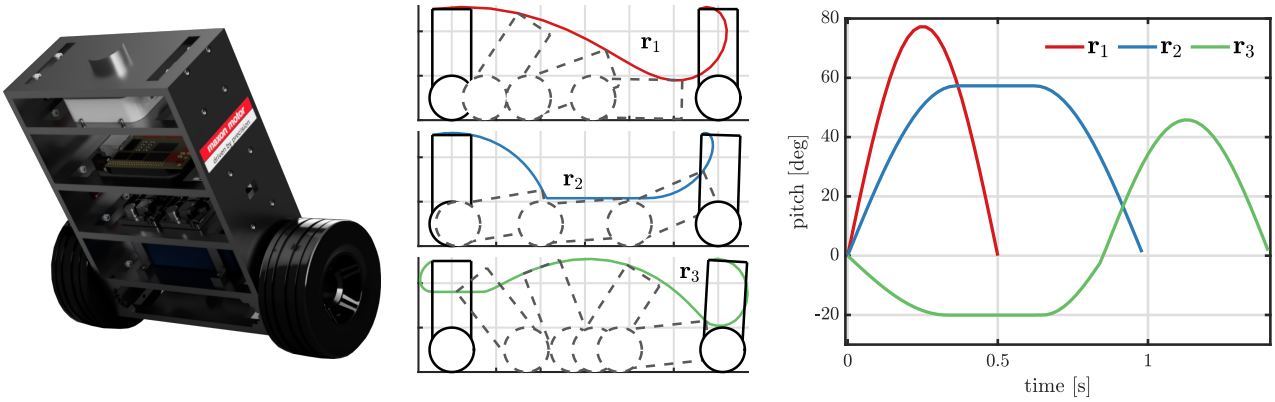

Fig. 5. The learning problem: A TWIPR (left) is meant to perform three challenging maneuvers (middle). The corresponding pitch angle references (right) differ in length, amplitude, and frequencies.

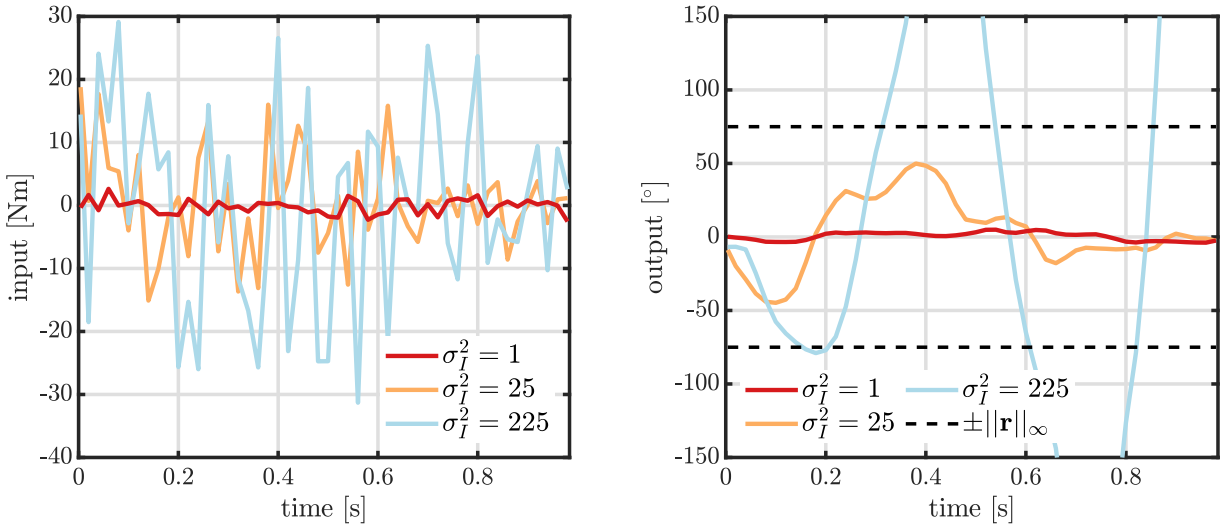

Fig. 6. Determination of the input variance $\boldsymbol{\sigma_I^2}$: Three different values are used to draw random input trajectories that are applied to the plant. The input variance $\boldsymbol{\sigma_I^2 = 1}$ hardly excites the system. In contrary, the input variance $\boldsymbol{\sigma_I^2 = 225}$ leads to an output trajectory that significantly exceeds the reference's maximum. The input variance $\boldsymbol{\sigma_I^2 = 25}$ is selected, because the corresponding output trajectory has the same order of magnitude as the reference.

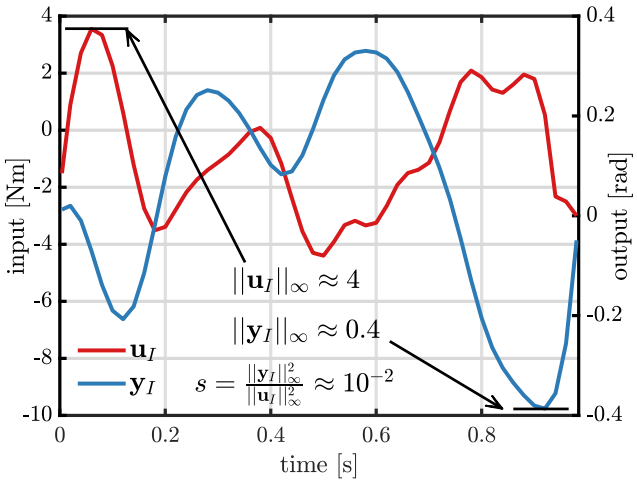

Fig. 7. Determination of the weight s: Based on the maxima of input and output trajectory in an initial trial, the weight s is chosen according to (27).

B. Learning Performance

First, learning performance for the desired references $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$ is investigated. The parameters are chosen according to the previous section and only one initial trial $I = 1$ is used. In Fig. 8, progressions of the output trajectories and error norms over the trials are depicted. For all three references, the proposed method achieves precise tracking after roughly 15 trials. The respective RMSEs rapidly decline over the first trials and converge to a small value close to zero. In case of the references $\mathbf{r}_2$ and $\mathbf{r}_3$, the RMSE decreases monotonically. The simulations demonstrate that, by using the automatically determined parameters, the proposed method rapidly learns to track three different reference trajectories without requiring any system-specific prior knowledge.

C. Robustness Analysis

The previous simulations have validated the method's capability of achieving satisfying tracking performance when using the automatically determined parameters. However, a

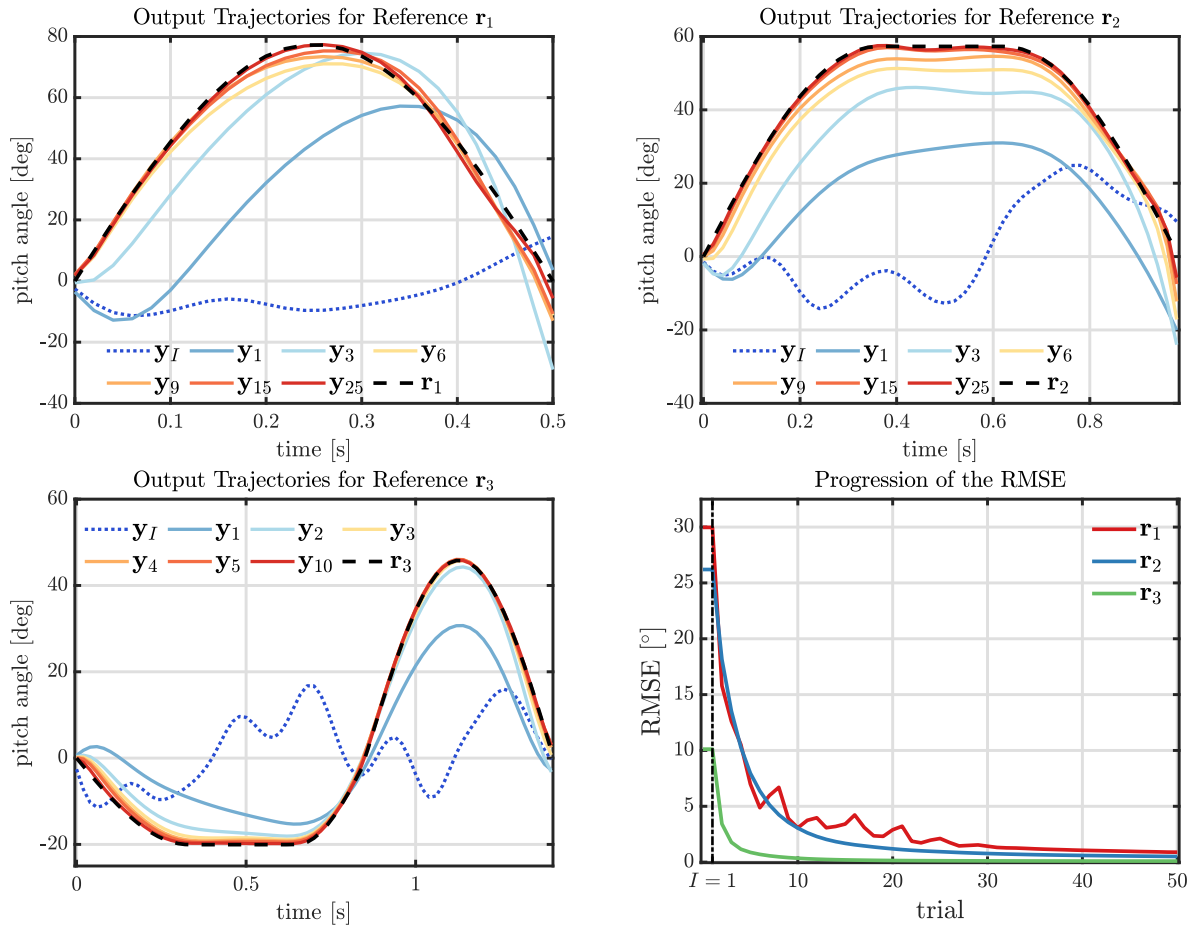

Fig. 8. The proposed learning method is employed to track the three desired references. Despite varying lengths, amplitudes and frequencies of the references, satisfying tracking performance is achieved within 10-15 trials. The RMSE is monotonically declining for two of the references and converges to a small value close to zero in all three scenarios.

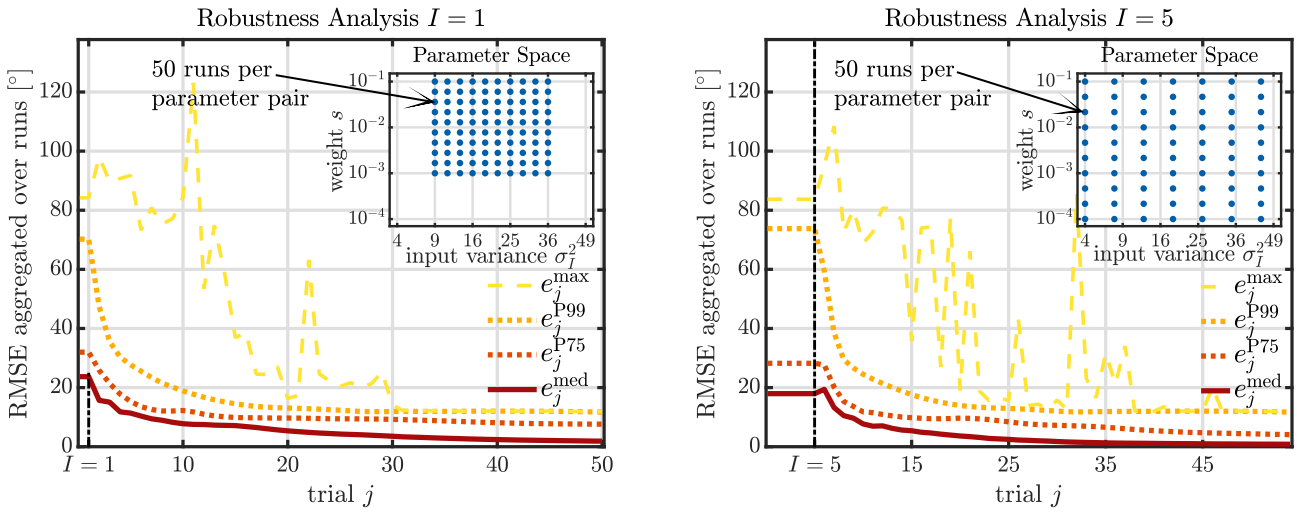

Fig. 9. The proposed learning method is run for a total of 5000 different combinations of parameters and initial data. The RMSE's maximum over all runs converges to a value significantly lower than the initial. Hence, robust learning is guaranteed for a large parameter space.

learning method should, ideally, not only achieve satisfying performance for a single parameter configuration, but for a wide parameter space. Hence, the proposed method's robustness with respect to the automatically determined parameters is validated in the following study, where we aim at tracking reference $\mathbf{r}_1$. We consider two different scenarios, namely, the greedy case of one initial trial, $I = 1$, and the conservative case of five initial trials, $I = 5$. Recall the two parameters s and σ_I^2, which are the weight in the optimal control problem and the initial input variance.

In the case of $I = 1$, the weight s is chosen from the set $\mathcal{S}_1$ that consists of ten logarithmically spaced values and the initial input variance σ_I^2 is chosen from the set $\mathcal{V}_1$ that consists of ten quadratically spaced values, i.e., for $I = 1$,

$$\mathcal{S}_1 = \{10^{-1}, \dots, 10^{-3}\} \quad |\mathcal{S}_1| = 10\,, \tag{32}$$
$$\mathcal{V}_1 = \{3^2, \dots, 6^2\} \quad |\mathcal{V}_1| = 10\,, \tag{33}$$
$$\left(s, \sigma_\mathrm{I}^2\right) \in \mathcal{P}_1 := \mathcal{S}_1 \times \mathcal{V}_1 \quad |\mathcal{P}_1| = 100\,. \tag{34}$$

For each of the 100 parameter pairs in $\mathcal{P}_1$, 50 *runs* are performed. A *run* r consists of choosing a parameter pair $(s, \sigma_\mathrm{I}^2)_k$ from $\mathcal{P}_I$, producing I initial input trajectories, and executing the proposed learning method for an additional 50 trials such that a progression of the RMSE throughout trials is obtained, which we denote by

$$e_{j,k,r}^{\mathrm{RMS}}\,, \tag{35}$$

where $j \in [0, I + N]$ is the trial index, $k \in [1, 100]$ is the parameter index, and $r \in [1, 50]$ is the run index.

The same procedure is applied in the case of $I = 5$, but the parameters are chosen from larger sets, i.e., for $I = 5$,

$$\mathcal{S}_5 = \{10^{-1}, \dots, 10^{-4}\} \quad |\mathcal{S}_5| = 10\,, \tag{36}$$
$$\mathcal{V}_5 = \{3^2, \dots, 7^2\} \quad |\mathcal{V}_5| = 10\,, \tag{37}$$
$$\left(s, \sigma_\mathrm{I}^2\right) \in \mathcal{P}_1 := \mathcal{S}_1 \times \mathcal{V}_5 \quad |\mathcal{P}_1| = 100\,. \tag{38}$$

To evaluate performance, the maximum $e_j^{\max} \in \mathbb{R}$, 99th percentile $e_j^{\mathrm{P99}} \in \mathbb{R}$, 75th percentile $e_j^{\mathrm{P75}} \in \mathbb{R}$, and median $e_j^{\mathrm{med}} \in \mathbb{R}$ of the RMSE over parameters and runs are considered. Formally,

$$\forall j \in \mathbb{N}_{\geq 0}, \quad e_j^{\max} := \max_{k \in [1,100], r \in [1,50]} \left(e_{j,k,r}^{\mathrm{RMSE}}\right)\,, \tag{39}$$

and e_j^{P99}, e_j^{P75}, e_j^{med} are defined accordingly. Results depicted in Fig. 9 show that, for both $I = 1$ and $I = 5$, the maximum of the RMSE converges to a value that is a roughly ten times smaller than the initial value such that the method's robustness is validated. The RMSE's 99^{th} percentile is monotonically decreasing, which implies that, besides single outliers, the method achieves the desired form of convergence as defined in Section II. Furthermore, the RMSE's median declines below a value of five degrees within 25 trials meaning that satisfying tracking performance is achieved. Lastly, it should be noted that a wider parameter space could be considered in the case of $I = 5$ meaning that, by increasing the amount of initial data, the robustness of the method can be further increased.

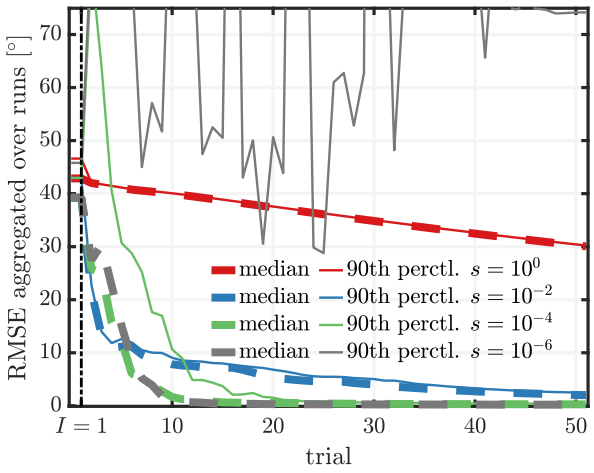

Fig. 10. Investigation of the effect of weight s on the learning characteristics: Large values of s lead to slow learning with small performance variance. Increasing the value leads to faster learning but also a larger variance in performance. Excessively small values of s may lead to a RMSE that diverges for some initial data.

D. Effects of Weights

The previous analysis has shown that the method rapidly learns to track a desired reference while also being robust with respect to the automatically determined parameters. Next to the use case of automatic plug & play application, the method can also be tuned to meet the needs of a specific application. Hence, we next investigate how the choice of weight s affects learning characteristics, namely the rate of convergence and robustness with respect to initial data. For this purpose, we consider the weights

$$s \in \{10^0, 10^{-2}, 10^{-4}, 10^{-6}\}\,. \tag{40}$$

The remaining learning parameters are chosen as one initial trial $I = 1$ and an initial input variance $\sigma_\mathrm{I}^2 = 25$. For each weight, 50 runs with differing initial data are performed and performance is judged based on the RMSE's 90^{th} percentile and median over the 50 runs.

Results depicted in Fig. 10 show that for a comparatively large value of $s = 10^0$, the RMSE monotonically declines at a slow pace. Furthermore, there is hardly any difference between median and 90^{th} percentile performance. Decreasing the value to $s = 10^{-2}$ leads to a significant increase in speed of convergence. Speed of median convergence can be further increased by lowering the value of the weight to $s = 10^{-4}$, which, however, comes at the price of larger 90^{th} percentile RMSEs, which imply an increase in performance variance. If the weight is lowered to an even smaller value, $s = 10^{-6}$, median performance is not further increased, but the 90^{th} percentile RMSE does no longer converge meaning that learning fails in a significant portion of runs. In summary, the study indicates that the weight s may be used to tune learning behavior, whereby comparatively large values of s lead to slow learning that is robust with respect to initial data. Decreasing the value of s can increase the speed of learning, but may come at the price of sensitivity with respect to initial data.

V. Validation by Experiment

To demonstrate the plug & play applicability of the proposed learning method, it is applied to a real-world TWIPR, which has been previously used to validate learning control methods [54]. The robot is meant to dive beneath an obstacle as depicted in Fig. 1 with the corresponding reference trajectory $\mathbf{r} \in \mathbb{R}^{75}$, $\forall n \in [1, 75]$,

$$[\mathbf{r}]_n = \begin{cases} 80\sin(\pi T n) & n \leq 25 \\ 80 & 25 < n \leq 50 \\ 80\sin(\pi T(n-25)) & 50 < n \end{cases} \quad [^\circ]. \tag{41}$$

First, the proposed method determines the learning parameters yielding $I = 1$, $\sigma_\mathrm{f}^2 = 2$, and $s = 0.1$. The initial input trajectory is drawn according to (24) an applied to the TWIPR. The corresponding output trajectory significantly differs from the reference with a RMSE of roughly 75°, see Fig. 11. From here onwards, the method iteratively determines a GP model, updates the input trajectory, and performs an experimental trial. Once learning begins, the RMSE rapidly declines, the RMSE drops below 20° on the fourth trial, and a RMSE of less than 10° is reached on the eighth trial. Sufficiently precise tracking precision for diving beneath the obstacle is achieved on the seventh trial and a RMSE close to zero is achieved on the tenth trial. Note, that the RMSE slightly increases on some of the trials, which is likely due to the initial conditions varying from trial to trial.

In summary, the experiments validate that the proposed method enables a real-world robot with unknown, nonlinear dynamics to learn a challenging maneuver. Not only did learning require a small number of trials (≈ 10) but the method could also be applied in a plug & play manor without iterative tuning of parameters.

VI. Conclusion

In this work, a GP-based learning control scheme has been proposed that autonomously solves reference tracking tasks in systems with unknown, nonlinear dynamics. On each iteration, the unknown dynamics are approximated by a Gaussian Process (GP), which is then used to determine and apply an optimal feedforward control input. The method is completely plug & play, since all required algorithms parameters are determined automatically and manual tuning is avoided. The effectiveness and efficiency of the method were demonstrated by simulations and experiments using the example of a two-wheeled inverted pendulum robot that rapidly learns to perform several challenging maneuvers without any manual tuning or system-specific prior knowledge.

In contrast to previous GP-based learning control approaches, the proposed method overcomes the inherent limitations of time-domain feedback control; it neither assumes knowledge of an effective feedback control structure, nor does it assume the entire state vector to be known. Instead, the proposed method directly adjusts the input based on the measured output. It is therefore as model-agnostic and independent of system-specific prior knowledge as pure reinforcement learning schemes.

While reinforcement learning approaches typically require hundreds of trials for convergence and are therefore unsuitable

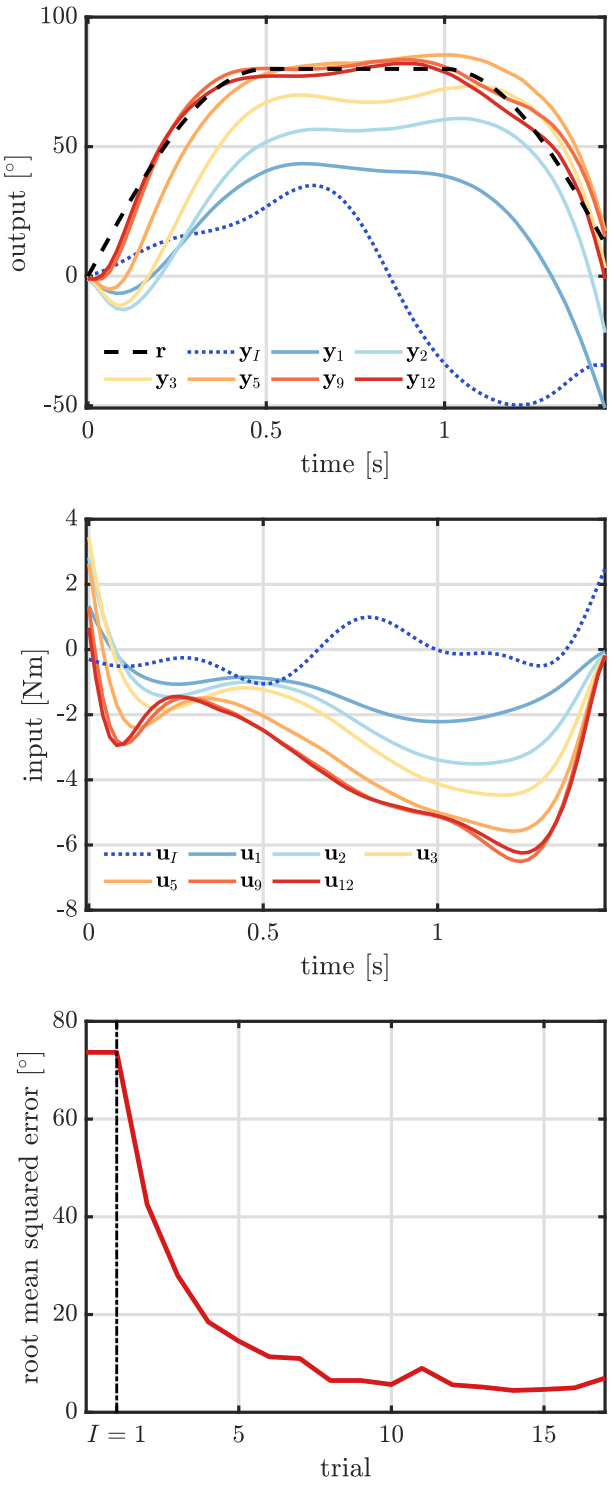

Fig. 11. Experimental results of the TWIPR learning to dive beneath an obstacle. Starting from an initial RMSE of roughly **75°**, the tracking error rapidly declines over the the following trials and sufficiently precise tracking for diving beneath the obstacle is achieved on the seventh trial.

for experimental validation, the proposed learning control method solves reference tracking problems in a small two-digit number of trials and was successfully validated in real-world experiments.

While the vast majority of previous contributions either validate methods only in simulations or provide only a single results for one carefully chosen parameterization and one specific motion, the present validation has proven effectiveness of the proposed method for several different motions and a large range of algorithm parameterizations. We thereby demonstrated robustness with respect to the automatically determined parameters, and we further investigated the effect of the learning weights on the trade-off between speed of learning and robustness.

We believe that the proposed method is highly suitable for use in kinematic systems that must perform challenging, highly dynamic maneuvers. Beyond the use case of rigid robotics, we expect the proposed method to have a major impact on the development of soft robotics, exoskeletons, and neuroprosthetics, and will therefore contribute to the evolution of autonomous robotic systems that rapidly learn to perform complex, dynamic motions under unknown conditions.

Future work will be concerned with extending the proposed approach to multi-input/multi-output systems and applying the method to other real-world applications. Moreover, combined approaches for simultaneous learning of feedfoward and feedback control will be devised and studied.

REFERENCES

[1] R. Murphy, "Activities of the rescue robots at the world trade center from 11-21 september 2001," *IEEE Robotics & Automation Magazine*, vol. 11, pp. 50–61, Sept. 2004.

[2] C. Coulson, R. Taylor, A. Reid, M. Griffiths, D. Proops, and P. Brett, "An autonomous surgical robot for drilling a cochleostomy: preliminary porcine trial," *Clinical Otolaryngology*, vol. 33, pp. 343–347, Aug. 2008.

[3] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. W. Grizzle, "Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking," *IEEE Control Systems Magazine*, vol. 38, no. 6, pp. 61–87, 2018.

[4] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast online trajectory optimization for the bipedal robot cassie," in *Robotics: Science and Systems XIV*, Robotics: Science and Systems Foundation, June 2018.

[5] M. Hehn and R. D'Andrea, "Real-time trajectory generation for interception maneuvers with quadrocopters," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4979–4984, IEEE, 2012.

[6] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, Nov. 2014.

[7] W. Dong and K. D. Kuhnert, "Robust adaptive control of nonholonomic mobile robot with parameter and nonparameter uncertainties," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 261–266, 2005.

[8] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive Control of Quadrotor UAVs :," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1400–1406, 2013.

[9] I. Golovin and S. Palis, "Robust control for active damping of elastic gantry crane vibrations," *Mechanical Systems and Signal Processing*, vol. 121, pp. 264–278, Apr. 2019.

[10] F. L. Muller, A. Schoellig, and R. D'Andrea, "Iterative learning of feed-forward corrections for high-performance tracking," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3276–3281, 2012.

[11] T. Seel, C. Werner, J. Raisch, and T. Schauer, "Iterative learning control of a drop foot neuroprosthesis—generating physiological foot motion in paretic gait by automatic feedback control," *Control Engineering Practice*, vol. 48, pp. 87–97, 2016.

[12] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller, "Deepmind control suite," *CoRR*, vol. abs/1801.00690, 2018.

[13] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[14] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[15] E. Schuitema, *Reinforcement Learning on autonomous humanoid robots*. PhD thesis, Delft University of Technology, 2012.

[16] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.

[17] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning* (A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 651–673, PMLR, 29–31 Oct 2018.

[18] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[19] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*, vol. 9. KIT Scientific Publishing, 2010.

[20] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, no. July, pp. 465–472, 2011.

[21] D. E. Moriarty and R. Miikkulainen, "Efficient Reinforcement Learning through Symbiotic Evolution," 2007.

[22] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2006.

[23] F. L. Lewis and D. Vrabie, "Adaptive dynamic programming for feedback control," *Proceedings of 2009 7th Asian Control Conference, ASCC 2009*, pp. 1402–1409, 2009.

[24] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 1, pp. 14–25, 2011.

[25] M. P. Deisenroth, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1–142, 2011.

[26] M. Hesse, J. Timmermann, E. Hüllermeier, and A. Trächtler, "A Reinforcement Learning Strategy for the Swing-Up of the Double Pendulum on a Cart," *Procedia Manufacturing*, vol. 24, pp. 15–20, 2018.

[27] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," *Robotics: Science and Systems*, vol. 7, pp. 57–64, 2012.

[28] J. Vinogradska, B. Bischo, D. Nguyen-Tuong, and J. Peters, "Stability of controllers for Gaussian process dynamics," *Journal of Machine Learning Research*, vol. 18, pp. 1–37, 2017.

[29] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, H. Schmidt, A. Romer, and J. Peters, "Stability of controllers for Gaussian process forward models," *33rd International Conference on Machine Learning, ICML 2016*, vol. 2, pp. 819–828, 2016.

[30] D. Nguyen-Tuong and J. Peters, "Local Gaussian process regression for real-time model-based robot control," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 380–385, 2008.

[31] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized active learning of Gaussian process state space models," *arXiv*, vol. 120, pp. 1–10, 2020.

[32] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," *2015 European Control Conference, ECC 2015*, pp. 2496–2501, 2015.

[33] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *arXiv*, vol. 28, no. 6, pp. 2736–2743, 2017.

[34] J. Umlauft, T. Beckers, A. Capone, A. Lederer, and S. Hirche, "Smart Forgetting for Safe Online Learning with Gaussian Processes," *Learning for Dynamics & Control*, pp. 1–10, 2020.

[35] H.-S. Ahn, Y. Chen, and K. L. Moore, "Iterative learning control: Brief survey and categorization," *IEEE Transactions on Systems, Man and*

Cybernetics, Part C (Applications and Reviews), vol. 37, pp. 1099–1121, Nov. 2007.
[36] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of Robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.
[37] A. Tayebi and M. B. Zaremba, "Robust ILC design is straightforward for uncertain LTI systems satisfying the robust performance condition," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 35, no. 1, pp. 445–450, 2002.
[38] S. Gunnarsson and M. Norrlöf, "On the design of ILC algorithms using optimization," *Automatica*, vol. 37, no. 12, pp. 2011–2016, 2001.
[39] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," *IEE Proceedings: Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996.
[40] D. A. Bristow, M. Tharayil, A. G. Alleyne, and Z. Z. Han, "A Survey of Iterative Learning Control," *Kongzhi yu Juece/Control and Decision*, vol. 20, no. 9, pp. 961–966, 2005.
[41] D. Shen, W. Zhang, and J. X. Xu, "Iterative Learning Control for discrete nonlinear systems with randomly iteration varying lengths," *Systems and Control Letters*, vol. 96, pp. 81–87, 2016.
[42] J. Lu, Z. Cao, R. Zhang, and F. Gao, "Nonlinear Monotonically Convergent Iterative Learning Control for Batch Processes," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5826–5836, 2018.
[43] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013. Data-based Control, Decision, Scheduling and Fault Diagnostics.
[44] R. Chi, Z. Hou, B. Huang, and S. Jin, "A unified data-driven design framework of optimality-based generalized iterative learning control," *Computers and Chemical Engineering*, vol. 77, pp. 10–23, 2015.
[45] R. Chi, Z. Hou, B. Huang, and S. Jin, "A unified data-driven design framework of optimality-based generalized iterative learning control," *Computers and Chemical Engineering*, vol. 77, pp. 10–23, 2015.
[46] Q. Ai, D. Ke, J. Zuo, W. Meng, Q. Liu, Z. Zhang, and S. Q. Xie, "High-Order Model-Free Adaptive Iterative Learning Control of Pneumatic Artificial Muscle with Enhanced Convergence," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 11, pp. 9548–9559, 2020.
[47] L. Ma, X. Liu, X. Kong, and K. Y. Lee, "Iterative learning model predictive control based on iterative data-driven modeling," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.
[48] Q. Yu, Z. Hou, X. Bu, and Q. Yu, "RBFNN-Based Data-Driven Predictive Iterative Learning Control for Nonaffine Nonlinear Systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1170–1182, 2020.
[49] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 465–472, 2011.
[50] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
[51] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in neural information processing systems*, pp. 1257–1264, 2006.
[52] M. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (C. M. Bishop and B. J. Frey, eds.), (Key West, FL), 2003.
[53] S. Kim and S. Kwon, "Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot," *International Journal of Control, Automation and Systems*, vol. 13, pp. 926–933, May 2015.
[54] M. Meindl, F. Molinari, J. Raisch, and T. Seel, "Overcoming output constraints in iterative learning control systems by reference adaptation," *arXiv preprint arXiv:2002.00662*, 2020.
[55] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

Appendix I
Comparison of Feedforward and Feedback Control for Reference Tracking

We consider a discrete-time, linear system of first order with sampling period $T = 0.02$ seconds, output $y \in \mathbb{R}$, and input $u \in \mathbb{R}$. The system is affected by an input delay of one sample leading to state-space dynamics

$$\forall n \in \mathbb{N}, \quad \mathbf{x}(n+1) = \begin{bmatrix} 0.5 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(n) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(n)\,, \tag{42}$$

where $\mathbf{x} \in \mathbb{R}^2$ is the state vector. The system's output is affected by measurement noise, i.e.

$$\forall n \in \mathbb{N}_{\geq 0}, \quad y(n) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(n) + w(n)\,, \tag{43}$$

where $w(n) \in \mathbb{R}$ is zero-mean measurement noise with variance 10^{-4}, i.e.

$$\forall n \in \mathbb{N}_{\geq 0}, \quad w(n) \sim \mathcal{N}(0, 10^{-4})\,. \tag{44}$$

The task consist of having the output y follow the reference $r \in \mathbb{R}$ over a finite horizon of $N = 100$ samples with

$$\forall n \in [1, N], \quad r(n) = \sin(2\pi T n)\,. \tag{45}$$

The feedforward control strategy consists of applying an input trajectory

$$\mathbf{u}_{\mathrm{FF}} := \begin{bmatrix} u_{\mathrm{FF}}(0) & u_{\mathrm{FF}}(1) & \dots & u_{\mathrm{FF}}(N-1) \end{bmatrix}^T\,. \tag{46}$$

The input values are determined by optimization such that the squared tracking error is minimized, i.e.

$$\mathbf{u}_{\mathrm{FF}} = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[r(n) - y(n) \right]^2\,. \tag{47}$$

The feedback control strategy consists of a generic, non-linear function to ensure that performance is not limited by the structure of the feedback law. In particular, the input values u_{FB} are computed as the sum of ten polynomials of tenth order, to which the current and nine previous error samples serve as inputs, i.e., $\forall n \in [1, N]$,

$$u_{\mathrm{FB}}(n) = \sum_{i=1}^{10} \sum_{j=1}^{10} k_{ij} \left[r(n) - y(n) \right]^j\,. \tag{48}$$

The set of feedback parameters $\mathcal{K} = \{k_{ij} \mid i, j \in [1, 10]\}$ is determined via optimization such that the squared tracking error is minimized, i.e.

$$\mathcal{K} = \underset{\tilde{\mathcal{K}}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[r(n) - y(n) \right]^2\,. \tag{49}$$

Appendix II
Reference Trajectories

The first reference $\mathbf{r}_1 \in \mathbb{R}^{25}$ is given by, $\forall n \in [1, 25]$,

$$[\mathbf{r}_1]_n = 75 \sin(2\pi T n)\,. \tag{50}$$

The second reference $\mathbf{r}_2 \in \mathbb{R}^{50}$ is given by, $\forall n \in [1, 50]$,

$$[\mathbf{r}_2]_n = \begin{cases} 57 \sin(1.38\pi T n) & \forall n \leq 17 \\ 57 & \forall 17 < n \leq 31 \\ 57 \sin(1.38\pi T (n - 13)) & \forall 31 < n \end{cases} \quad [^\circ]\,. \tag{51}$$

The third reference $\mathbf{r}_3 \in \mathbb{R}^{71}$ is given by, $\forall n \in [1, 71]$,

$$[\mathbf{r}_3]_n = \begin{cases} -20\sin(1.5\pi T n) & \forall n \leq 16 \\ -20 & \forall 16 < n \leq 31 \\ -20\sin(2.4\pi T(n-11)) & \forall 31 < n \leq 43 \\ 46\sin(1.8\pi T(n+13)) & \forall 43 < n \end{cases} [^\circ]\,. \tag{52}$$

Appendix III
Feedback Control of the TWIPR

First, the dynamics of the TWIPR moving along a straight line are considered. The robot dynamics have two degrees of freedom, namely, the pitch angle $\Theta \in \mathbb{R}$ and the position $s \in \mathbb{R}$. The state vector follows with

$$\mathbf{x} = \begin{bmatrix} \Theta & \dot{\Theta} & s & \dot{s} \end{bmatrix}^T . \tag{53}$$

The motor torque serves as input variable and is denoted by $u \in \mathbb{R}$. To stabilize the TWIPR in its upright equilibrium, the nonlinear dynamics are approximated by a linear, discrete-time model of the form

$$\forall n \in \mathbb{N}, \quad \mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}u(n) \tag{54}$$

has been determined using a sampling period of $T = 0.02$ seconds. The stabilizing control input $u_\mathrm{C} \in \mathbb{R}$ is computed by linear state feedback of the form

$$\forall n \in \mathbb{N}, \quad u_\mathrm{C}(n) = -\mathbf{K}\mathbf{x}(n)\,, \tag{55}$$

where the feedback matrix $\mathbf{K}$ is designed by LQR [55].

To track the desired reference maneuvers, the feedback input u_C is superposed by a learned feedforward input u_L leading to the overall input

$$\forall n \in \mathbb{N}, \quad u(n) = u_\mathrm{C}(n) + u_\mathrm{L}(n)\,. \tag{56}$$

Michael Meindl received his Master of Science degree in mechatronic engineering from HS Karlsruhe in 2020. Currently, he is a research assistant at the Embedded Mechatronics Laboratory of HS Karlsruhe while also pursuing his Ph.D. degree at TU Berlin. His research focuses on movement learning in robotic systems with the prime interest of combining methods from the fields of control theory and machine learning.

Dustin Lehmann received his Master of Science in aeronautics & astronautics at TU Berlin in 2019. Since then, he has been working as a doctoral researcher in the Science of Intelligence Excellence Cluster at TU Berlin. His research focuses on understanding collective learning in multi-agent systems and applying control theory approaches to collective learning problems.

Thomas Seel is a professor at the Department Artificial Intelligence in Biomedical Engineering at Friedrich-Alexander-Universität Erlangen-Nürnberg. He studied engineering cybernetics at OvGU Magdeburg and UC Santa Barbara and received the Ph. D. from Technische Universität Berlin in 2016. His research focus is on dynamic inference and learning in biomedical and mechatronic systems.