Learning-based Robust Control: Guaranteeing Stability while Improving Performance

Felix Berkenkamp and Angela P. Schoellig

***Abstract*— To control dynamic systems, modern control theory relies on accurate mathematical models that describe the system behavior. Machine learning methods have proven to be an effective method to compensate for initial errors in these models and to achieve high-performance maneuvers by adapting the system model and control online. However, these methods usually do not guarantee stability during the learning process. On the other hand, the control community has traditionally accounted for model uncertainties by designing robust controllers. Robust controllers use a mathematical description of the uncertainty in the dynamic system derived prior to operation and guarantee robust stability for all uncertainties. Unlike machine learning methods, robust control does not improve the control performance by adapting the model online. This paper combines machine learning and robust control theory for the first time with the goal of improving control performance while guaranteeing stability. Data gathered during operation is used to reduce the uncertainty in the model and to learn systematic errors. Specifically, a nonlinear, nonparametric model of the unknown dynamics is learned with a Gaussian Process. This model is used for the computation of a linear robust controller, which guarantees stability around an operating point for all uncertainties. As a result, the robust controller improves its performance online while guaranteeing robust stability. A simulation example illustrates the performance improvements due to the learning-based robust controller.**

NOTE

This is a workshop paper. Please consider reading the conference paper instead: [1].

I. INTRODUCTION

Mathematical models are an important prerequisite when designing model-based controllers for dynamic systems using, for example, techniques such as Linear Quadratic Regulators [2] or Model Predictive Control [3]. The performance of the resulting controllers depends directly on the accuracy of the model. While established methods for system modeling and identification exist [4], [5], the resulting models are always an approximation of the real-system behavior. The goal of this paper is to use data gathered during operation to improve the model and, thus, the control performance while giving stability guarantees during the learning process (see Fig. 1).

In order to overcome the limitations of approximate models, machine learning techniques have been used before. These methods leverage online measurements to improve the control performance. One such example is iterative learning control (ILC), which achieves high-performance tracking by learning feed-forward inputs through a repeated execution of the task [6]. Other approaches improve the system model directly: an example are Neural Networks (NNs) [7]. Recently, the focus of the community has shifted to Gaussian Processes (GP) [8]. Learned GP models have, for example, been successfully used for dynamic programming [9] and to improve trajectory tracking of a robot traversing unknown, rough terrain [10]. An interesting property of GPs is that they provide uncertainty estimates for their predictions, which have been used in [11] to minimize the expected squared error in a model predictive control setting. While all these methods achieve impressive control performance after learning, they do not guarantee stability during the learning process. The reason for this is that these methods focus on updating the estimated model of the dynamics without explicitly considering the mismatch between the learned model and the real dynamics of the system. This mismatch has been studied in the adaptive control community, for example in [12], but the resulting approaches do not model uncertainties explicitly.

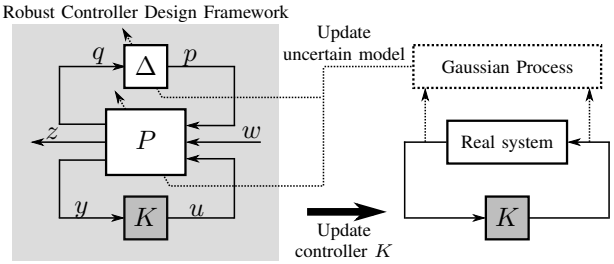

Fig. 1. Learning-based robust controller design. Using input-output data of the real system, the plant model P is updated and the uncertainty estimate Δ is gradually reduced. As a result, the performance of the robust controller K improves.

In the control community, model errors have been explicitly considered in the field of robust control established in the 1980s [13]. Robust control focuses on linear systems, incorporates an *a priori* estimate of the model uncertainty in the controller design and guarantees the stability of the system for all modeled uncertainties [14]. However, this uncertainty estimate is not updated during the operation of the system. The uncertainty specification provides stability guarantees but may decrease controller performance. The latter is due to the fact that the robust controller minimizes its performance objective for all possible system dynamics that lie in the uncertainty specification.

Since machine learning methods provide excellent perfor-

Felix Berkenkamp is with ETH Zurich, Zurich, Switzerland. Email: befelix@ethz.ch

Angela P. Schoellig is with the University of Toronto Institute for Aerospace Studies (UTIAS), Toronto, Canada. Email: schoellig@utias.utoronto.ca

mance after learning and robust control provides stability guarantees by explicitly taking model uncertainties into account, it is of interest to combine the two. The goal is a robust controller that learns from data gathered during operation in order to improve its performance while guaranteeing stability for all uncertainties during learning. Guaranteeing stability during learning has recently been stated as an open problem in robotics [15].

The main challenge in combining the two approaches lies in identifying the unknown plant dynamics and the corresponding uncertainty in a way that can be used in a robust controller design framework. Classical offline system identification methods focus on fitting linear models to observed data using either time or frequency domain methods [4], [5]. While these approaches can be extended to include uncertainty estimates [16], [17], fitting a linear model to the unknown nonlinear dynamics can lead to errors. In [18] a way to avert these errors and quantify the resulting uncertainty was shown by applying periodic inputs to the system, but this is generally not possible in an online setting. Thus, to achieve bias-free online system identification one must turn to nonlinear, nonparametric identification methods [19]. GPs are of particular interest, since they provide uncertainty estimates [8]. Combining robustness with nonlinear, nonparametric system identification was explored in [20], where a linear controller for the nominal plant was used to show robustness assuming that the learned dynamics can be considered as a bounded disturbance. While this approach guaranteed robustness, it did not adapt the *a priori* model uncertainty and neglected the fact that learned dynamics generally cannot be described via bounded disturbances.

This paper uses GPs for online learning of the system dynamics. The learned GP model and its uncertainty estimates are linearized using properties of GPs that have not been leveraged in a robust control framework before. The linearization point itself is considered to be uncertain and identified as well. Robust controller design methods are applied to the obtained uncertain model resulting in a convex optimization problem solvable online [21].

The remainder of this paper is structured as follows: in Sec. II the system model is presented and the robust control problem is stated. GPs are introduced in Sec. III and their application to online model learning is shown in Sec. IV. In Sec. V a robust control method is presented that is applicable to the GP model of Sec. IV. A discussion of the presented approach is provided Sec. VI. Finally, a simulation example is presented in Sec. VII and conclusions are drawn in Sec. VIII.

II. Problem Statement

This section introduces the system model and the robust control problem considered in this paper.

Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$ denote the system states, inputs and measured outputs, respectively. The discrete system dynamics are separated into two components, a *known* function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ (derived, for example, from first principles) and an *unknown* function $\mathbf{g}(\mathbf{x}, \mathbf{u})$. The function $\mathbf{g}$ represents unknown, deterministic dynamics that are not captured by the *a priori* model $\mathbf{f}$. Both functions are assumed to be continuously differentiable, $\mathbf{f}, \mathbf{g} \in C^1$. Measurements are made via a known matrix $\mathbf{C} \in \mathbb{R}^{p \times n}$ and are corrupted by zero-mean Gaussian noise, $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_n)$. The corresponding discrete system dynamics are given by

$$\begin{aligned} \mathbf{x}_{k+1} &= \underbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}_{\textit{a priori} \text{ model}} + \underbrace{\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k)}_{\text{unknown model}} \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \boldsymbol{\omega}_k, \end{aligned} \tag{1}$$

where k refers to the k th time step.

The goal of this paper is to find an estimate of $\mathbf{g}(\mathbf{x}, \mathbf{u})$ and an associated confidence interval of this estimate from measurement data. This information is used to design a robust controller that stabilizes the system despite the uncertainty in the estimate. As the estimate of $\mathbf{g}(\mathbf{x}, \mathbf{u})$ becomes more accurate, the robust controller performance gradually improves. In this paper, we model the unknown dynamics, $\mathbf{g}(\mathbf{x}, \mathbf{u})$, as a GP (see Sec. III) and consider linear robust control around a specific operating point $\mathbf{x}_s$ with a corresponding steady-state input $\mathbf{u}_s$, where

$$\mathbf{x_s} = \mathbf{f}(\mathbf{x}_s, \mathbf{u}_s) + \mathbf{g}(\mathbf{x}_s, \mathbf{u}_s). \tag{2}$$

Both $\mathbf{x}_s$ and $\mathbf{u}_s$ are initially unknown and are also estimated from measurement data, see Sec. IV-A.

Throughout this paper, we assume that $\mathbf{C}$ is full-rank such that noisy full-state information is available. While it is possible to extend the obtained results to partial-state information, this is outside of the scope of this paper. Pointers to relevant publications for the extension to partial-state information are provided in the relevant sections below.

III. Gaussian Process

In this section, we introduce the basic properties of GPs. GPs are used to model the unknown dynamics, $\mathbf{g}(\mathbf{x}, \mathbf{u})$, in (1). More details on how GPs are used in this context are given in Sec. IV.

GPs are a popular choice for nonparametric regression in machine learning, where the goal is to find an approximation of a nonlinear map, $g(\mathbf{a}) : \mathbb{R}^{\dim(\mathbf{a})} \mapsto \mathbb{R}$, from an input vector $\mathbf{a}$ to the function value $g(\mathbf{a})$. This is accomplished by assuming that the function values $g(\mathbf{a})$ are random variables and that any finite number of these random variables have a joint Gaussian distribution depending on the values of $\mathbf{a}$ [8].

For the nonparametric regression we need to define a prior for the mean of $g(\mathbf{a})$ and for the covariance between any two function values, $g(\mathbf{a}_i)$ and $g(\mathbf{a}_j)$, also known as the kernel. The mean is assumed to be zero, since the GP is used to learn the dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1) for which no prior knowledge is available. There are several possibilities for the covariance function. While the learning-based robust control approach presented in this paper works for any kernel with continuous first derivative, we focus on the often used squared-exponential function in the following discussion. For a squared-exponential function, points that are close to each other (in terms of their scaled, squared distance) have similar

function values. The covariance between two data points $g(\mathbf{a}_i)$ and $g(\mathbf{a}_j)$ is given by

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{M}^{-2}(\mathbf{a}_i - \mathbf{a}_j)\right) + \delta_{ij}\sigma_n^2, \tag{3}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The covariance is parameterized by three hyperparameters: the measurement noise, σ_n^2, the process variation, σ_f^2, and length scales, $\mathbf{l} \in \mathbb{R}^{\dim(\mathbf{a})}$, in the matrix $\mathbf{M} = \mathrm{diag}(\mathbf{l})$ corresponding to the rate of change of the function g with respect to $\mathbf{a}$. These hyperparameters are learned from observed data by solving a log maximum likelihood problem using gradient ascent methods [8].

A. Prediction

The introduced framework can be used to predict the function value of $g(\mathbf{a}^*)$ at an arbitrary input, $\mathbf{a}^*$, based on a set of N past observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{g}(\mathbf{a}_i)\}_{i=1}^N$. We assume that observations are noisy measurements of the true function value, $g(\mathbf{a})$: $\hat{g}(\mathbf{a}) = g(\mathbf{a}) + \omega_n$ with $\omega_n \sim \mathcal{N}(0, \sigma_n^2)$. The joint probability distribution of the function value $g(\mathbf{a}^*)$ and the observed data is given by

$$\begin{bmatrix} \hat{\mathbf{g}} \\ g(\mathbf{a}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}_N, \begin{bmatrix} \mathbf{K} & \mathbf{k}^{\mathrm{T}}(\mathbf{a}^*) \\ \mathbf{k}(\mathbf{a}^*) & k(\mathbf{a}^*, \mathbf{a}^*) \end{bmatrix}\right), \tag{4}$$

where $\hat{\mathbf{g}} = \left[\hat{g}(\mathbf{a}_1), \ldots, \hat{g}(\mathbf{a}_N)\right]^{\mathrm{T}}$ is the vector of observed function values and $\mathbf{0}_N \in \mathbb{R}^N$ is a vector of zeros. The covariance matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ has entries $\mathbf{K}_{ij} = k(\mathbf{a}_i, \mathbf{a}_j)$, and $\mathbf{k}(\mathbf{a}^*) = \begin{bmatrix} k(\mathbf{a}^*, \mathbf{a}_1) & \ldots & k(\mathbf{a}^*, \mathbf{a}_N) \end{bmatrix}$ contains the covariances between the new input $\mathbf{a}^*$ and the observed data points. Using standard properties of joint Gaussian distributions the prediction of $g(\mathbf{a}^*)$ conditioned on the data set $\mathcal{D}$ is given by a $g(\mathbf{a}^*)|\mathcal{D} \sim \mathcal{N}(\mu(\mathbf{a}^*), \sigma^2(\mathbf{a}^*))$ [8] with

$$\mu(\mathbf{a}^*) = \mathbf{k}(\mathbf{a}^*)\mathbf{K}^{-1}\hat{\mathbf{g}}, \tag{5}$$

$$\sigma^2(\mathbf{a}^*) = k(\mathbf{a}^*, \mathbf{a}^*) - \mathbf{k}(\mathbf{a}^*)\mathbf{K}^{-1}\mathbf{k}^{\mathrm{T}}(\mathbf{a}^*). \tag{6}$$

Notice that the prediction depends on the inverse of the $N \times N$ matrix $\mathbf{K}$, which is an expensive $\mathcal{O}(N^3)$ computation. However, the inverse must be computed only once for a given data set in order to make predictions.

So far we considered a scalar function g. The extension to a vector-valued function, as required for approximating $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1), can be done by extending (4) resulting in a matrix $\mathbf{K}$ of size $nN \times nN$. Such a matrix is usually too big to be inverted efficiently. To maintain computational feasibility, independent GPs are trained for each dimension of $\mathbf{g}(\mathbf{x}, \mathbf{u})$ neglecting any additional information that could be gained from the correlation between the outputs introduced, for example, by the measurement noise.

B. Derivatives

An extension to the general GP framework, which is important for Sec. IV, is that the derivative of a GP is a GP as well [8]. This follows from the fact that the derivative is a linear operator. From (4), we get

$$\begin{bmatrix} \hat{\mathbf{g}} \\ \frac{\partial g(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}_N, \begin{bmatrix} \mathbf{K} & \frac{\partial \mathbf{k}^{\mathrm{T}}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \\ \frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} & \frac{\partial^2 k(\mathbf{a},\mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} \end{bmatrix}\right), \tag{7}$$

where $h(\mathbf{a})|_{\mathbf{a}^*}$ means $h(\mathbf{a})$ evaluated at $\mathbf{a} = \mathbf{a}^*$. The derivatives of the squared-exponential function are given by

$$\frac{\partial \mathbf{k}_{(:,i)}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} = M^{-2}(\mathbf{a}_i - \mathbf{a}^*)k(\mathbf{a}^*, \mathbf{a}_i), \tag{8}$$

$$\frac{\partial^2 k(\mathbf{a}, \mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} = \sigma_f^2 \mathbf{M}^{-2}, \tag{9}$$

where $\mathbf{k}_{(:,i)}$ refers to the ith column of $\mathbf{k}$. In the literature, this property has mainly been used to include derivative observations; however, following the same reasoning as in the previous section predictions of the derivatives are given by $\frac{\partial g(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} |\mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}'(\mathbf{a}^*), \boldsymbol{\Sigma}'(\mathbf{a}^*))$ with

$$\boldsymbol{\mu}'(\mathbf{a}^*) = \left(\frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \mathbf{K}^{-1}\right)^{\mathrm{T}} \hat{\mathbf{g}}, \tag{10}$$

$$\boldsymbol{\Sigma}'(\mathbf{a}^*) = \frac{\partial^2 k(\mathbf{a}, \mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} - \frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \mathbf{K}^{-1} \frac{\partial \mathbf{k}^{\mathrm{T}}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*}. \tag{11}$$

From (11) it can be seen that $\sigma_f^2 \mathbf{M}^{-2}$ is the prior variance of the first derivative, which confirms the interpretation of the length-scale hyperparameters as corresponding to the rate of change of g. It is important to notice that (10) is just the derivative of the predicted mean in (5), but this does not hold for the variance.

IV. Model Update

In this section, we show how the nonlinear GPs introduced in Sec. III are used for the online identification of a linear model of the unknown dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1) around a linearization point. This model is used in Sec. V to update the robust controller, see Fig. 1.

We use separate GPs to identify each dimension of the unknown dynamics. The input $\mathbf{a}_k = (\mathbf{x}_k, \mathbf{u}_k)$ consists of the current state and input while the training target of the GPs is the model error $\hat{\mathbf{g}}(\mathbf{a}_k) = \mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. As a result, it is possible to make predictions about the function value and first derivative of the unknown dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$ using (5)-(6) and (10)-(11), respectively. As mentioned in Sec. II, this requires the full state $\mathbf{x}$ to be known. Extensions to partial-state measurements can be found in [22], [23].

A. Operating Point

The goal of the control problem considered in this paper is to stabilize the system around a fixed desired operating point. Since the *a priori* operating point and associated steady-state input of the known dynamics $\mathbf{f}$ with $\bar{\mathbf{x}}_s = \mathbf{f}(\bar{\mathbf{x}}_s, \bar{\mathbf{u}}_s)$ are not necessarily equal to (2), we regularly update the operating point as the unknown dynamics are learned.

Given the prior operating point $\bar{\mathbf{x}}_s$, an updated steady-state operating point that satisfies (2) with the mean estimate for $\mathbf{g}$ from (5) is found via the optimization problem

$$\begin{aligned} &\min_{\mathbf{x}_s,\mathbf{u}_s} \|\bar{\mathbf{x}}_s - \mathbf{x}_s\|_{\mathbf{Q}} \\ &\text{subject to} \\ &\quad \mathbf{x}_s = \mathbf{f}(\mathbf{x}_s,\mathbf{u}_s) + \boldsymbol{\mu}(\mathbf{x}_s,\mathbf{u}_s), \end{aligned} \tag{12}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the learned mean consisting of the individual predictions for each state according to (5) and $\|\mathbf{x}\|_{\mathbf{Q}} = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}$ refers to the norm with a weighting matrix $\mathbf{Q}$ that accounts for the prior uncertainty in the different states as well as for scaling. This nonlinear, constrained optimization problem can be solved quickly using $\bar{\mathbf{x}}_s$ as an initial guess. Notice that if $\mathbf{x}_s$ is known a simpler optimization problem can be formulated, which is solvable via gradient descent:

$$\min_{\mathbf{u}_s} \|\mathbf{x}_s - \mathbf{f}(\mathbf{x}_s,\mathbf{u}_s) - \boldsymbol{\mu}(\mathbf{x}_s,\mathbf{u}_s)\|_{\mathbf{Q}}. \tag{13}$$

B. Linearization

In the previous section, an operating point including steady-state input was found. The second step of the model update (Fig. 1) linearizes system (1) is linearized around this operating point and obtains corresponding elementwise confidence intervals. This information is used in Sec. V to update the robust controller.

The dynamics (1) are linearized around the operating point $\mathbf{a}^* = (\mathbf{x}_s, \mathbf{u}_s)$ using the properties of GPs presented in Sec. III-B. Denoting the derivative of the i^{th} component in $\mathbf{g}$ with respect to all states and inputs in $\mathbf{a}$ by $\frac{\partial \mathbf{g}_i(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \sim \mathcal{N}(\boldsymbol{\mu}_i'(\mathbf{a}^*), \boldsymbol{\Sigma}_i'(\mathbf{a}^*))$, we obtain

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} + \begin{bmatrix}\boldsymbol{\mu}_1'(\mathbf{a}^*) \\ \vdots \\ \boldsymbol{\mu}_n'(\mathbf{a}^*)\end{bmatrix} \tag{14}$$

and the uncertainty is represented by

$$\begin{bmatrix}\mathbf{A}_u & \mathbf{B}_u\end{bmatrix} = 2\begin{bmatrix}\mathrm{diag}(\boldsymbol{\Sigma}'^{1/2}_1(\mathbf{a}^*))^{\mathrm{T}} \\ \vdots \\ \mathrm{diag}(\boldsymbol{\Sigma}'^{1/2}_n(\mathbf{a}^*))^{\mathrm{T}}\end{bmatrix}, \tag{15}$$

where $\mathbf{a} = (\mathbf{x}, \mathbf{u})$ and each element of the matrices $\mathbf{A}_u$ and $\mathbf{B}_u$ in (15) is equal to the 2σ (95%) confidence interval for the corresponding element in $\mathbf{A}$ and $\mathbf{B}$. Consequently, the linearized dynamics are given by

$$\mathbf{x}_{k+1} = (\mathbf{A} + \mathbf{A}_u \circ \boldsymbol{\Delta}_x)\mathbf{x}_k + (\mathbf{B} + \mathbf{B}_u \circ \boldsymbol{\Delta}_u)\mathbf{u}_k, \tag{16}$$

where $\circ$ is the elementwise matrix product and $\boldsymbol{\Delta}_{(x,u)}$ represents an interval uncertainty with matrix entries in $[-1, 1]$; that is, the state matrix is $\mathbf{A} + \mathbf{A}_u$ where every element in $\mathbf{A}_u$ is independently scaled by a factor in $[-1, 1]$.

V. Robust Control

We derive a robust controller of the form $\mathbf{u}_k = \mathbf{K}\mathbf{x}_k$ for the linear model in (16), which stabilizes the system despite the uncertainties introduced by $\boldsymbol{\Delta}_{(x,u)}$. In the general framework of robust control (see Fig. 1), the objective is to minimize an error signal $\mathbf{z}$ caused by a disturbance $\mathbf{w}$ for all possible uncertainties introduced via the uncertain signal $\mathbf{p}$, cf. [14]:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{B}_w\mathbf{w}_k + \mathbf{B}_p\mathbf{p}_k \tag{17}$$

$$\mathbf{z}_k = \mathbf{C}_z\mathbf{x}_k + \mathbf{D}_{zu}\mathbf{u}_k + \mathbf{D}_{zw}\mathbf{w}_k \tag{18}$$

$$\mathbf{q}_k = \mathbf{C}_q\mathbf{x}_k + \mathbf{D}_q\mathbf{u}_k \tag{19}$$

$$\mathbf{p}_k = \boldsymbol{\Delta}\mathbf{q}_k, \tag{20}$$

where $\boldsymbol{\Delta} = \mathrm{diag}(\delta_1, \ldots, \delta_r)$ with $|\delta_i| \leq 1, \quad i = 1, \ldots, r$, together with $\mathbf{C}_q, \mathbf{D}_q$, and $\mathbf{B}_p$ represents the uncertainty in the system (15) and $(\mathbf{A}, \mathbf{B})$ the nominal system (14). The uncertainty (15) can be represented by choosing

$$\mathbf{C}_q = \begin{bmatrix}\mathrm{diag}(\mathbf{A}_{u,(1,:)}) \\ \vdots \\ \mathrm{diag}(\mathbf{A}_{u,(n,:)}) \\ \mathbf{0}_{nm\times n}\end{bmatrix}, \quad \mathbf{D}_q = \begin{bmatrix}\mathbf{0}_{n^2\times m} \\ \mathrm{diag}(\mathbf{B}_{u,(1,:)}) \\ \vdots \\ \mathrm{diag}(\mathbf{B}_{u,(n,:)})\end{bmatrix}, \tag{21}$$

where $\mathbf{A}_{u,(i,:)}$ refers to the i^{th} row of $\mathbf{A}_u$. Thus the required uncertainty can be represented by picking $\mathbf{B}_p = \begin{bmatrix}\mathbf{I}_n \otimes \mathbf{1}_{n\times 1} & \mathbf{I}_n \otimes \mathbf{1}_{m\times 1}\end{bmatrix}$ with $\otimes$ denoting the Kronecker product, $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ the identity matrix and $\mathbf{0}_{i\times j}, \mathbf{1}_{1\times j} \in \mathbb{R}^{i\times j}$ vectors with respectively zeros and ones as the entries.

The matrices $\mathbf{B}_w, \mathbf{C}_z, \mathbf{D}_{zu}$ and $\mathbf{D}_{zw}$ define the control objective to be minimized by the robust controller. Possible choices can be found in [14]. A popular measure for the error signal $\mathbf{z}$ is the infinity norm ($\mathcal{H}_\infty$ control), which corresponds to minimizing the worst case gain of the transfer function T_{zw} from the disturbance $\mathbf{w}$ to the error $\mathbf{z}$ over all uncertainties. The corresponding optimization problem is $\min_{\mathbf{K}} \max_{\boldsymbol{\Delta}} \|T_{zw}\|_\infty$ where $T_{zw} = (\mathbf{C}+\mathbf{D}_{zu}\mathbf{K})(z\mathbf{I}-(\mathbf{A}+\mathbf{B}\mathbf{K}+\mathbf{B}_p\boldsymbol{\Delta}(\mathbf{C}_q+\mathbf{D}_q\mathbf{K})))^{-1}\mathbf{B}_w + \mathbf{D}_{zw}$ depends on the controller $\mathbf{K}$ and $\boldsymbol{\Delta}$.

Methods for solving this kind of problem have been proposed in [14]. In this paper, we use a method that requires finding a single Lyapunov function that guarantees stability for all possible $\boldsymbol{\Delta}$. While this method can be more conservative than others, it obtains the discrete-time controller by solving a convex optimization problem in terms of linear matrix inequalities (LMIs). Efficient solvers for LMIs exist [24], making these techniques applicable to online applications as considered in this paper. In fact, in [25] this method was applied to find a model predictive controller, which required solving an LMI at every time step.

The choice of control method is independent of the system identification approach shown in the previous sections. In this paper, the control law is based on [21], where an $\mathcal{H}_\infty$ state-feedback controller was found under the assumption that $\mathbf{p}$ is bounded by $\mathbf{p}^{\mathrm{T}}\mathbf{p} \leq \alpha(\mathbf{x}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \mathbf{u}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{u})$ for a given constant α and matrices $\mathbf{F}$ and $\mathbf{H}$. We modify the uncertainty representation in [21] to fit our uncertainty definition in (17)-(20). The proof is similar to [21] and omitted here, but can be found in [26]. Denoting symmetric matrix elements by $\bullet$ and with $\boldsymbol{\Lambda} = \mathrm{diag}(\tau_1, \ldots, \tau_r)$ and $\beta = \alpha^{-2}$, the resulting optimization problem to find a controller such that $\|T_{zw}\|_\infty^2 \leq \gamma$ is given by (22).

For $\alpha = 1$, the resulting closed-loop system is robustly stable for all uncertainties. Whenever this optimization problem is infeasible, a line search maximizing α with $0 \leq \alpha < 1$ leads to the best possible controller given the assumptions in [21]. From (22), we obtain $\mathbf{Q}$ and $\mathbf{R}$, and the resulting robust controller is given by $\mathbf{u}_k = \mathbf{K}\mathbf{x}_k$ with $\mathbf{K} = \mathbf{R}\mathbf{C}\mathbf{Q}^{-1}\mathbf{C}^{-1}$. This controller is re-calculated regularly based on the most up-to-date model obtained from Sec. IV. Results in [21] can be extended to the case of partial-state information.

$$\min_{\mathbf{Q}=\mathbf{Q}^{\mathrm{T}},\mathbf{R},\mathbf{\Lambda},\gamma,\beta=1} \gamma \tag{22}$$

subject to

$$\begin{bmatrix} -\mathbf{Q} & \bullet & \bullet & \bullet & \bullet & \bullet \\ \mathbf{0} & -\gamma\mathbf{I} & \bullet & \bullet & \bullet & \bullet \\ \mathbf{0} & \mathbf{0} & -\mathbf{\Lambda} & \bullet & \bullet & \bullet \\ \mathbf{A}\mathbf{Q}+\mathbf{B}\mathbf{R}\mathbf{C} & \mathbf{B}_w & \mathbf{B}_p\mathbf{\Lambda} & -\mathbf{Q} & \bullet & \bullet \\ \mathbf{C}_q\mathbf{Q}+\mathbf{D}_q\mathbf{R}\mathbf{C} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\beta\mathbf{\Lambda} & \bullet \\ \mathbf{C}_z\mathbf{Q}+\mathbf{D}_{zu}\mathbf{R}\mathbf{C} & \mathbf{D}_{zw} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \end{bmatrix} \leq 0.$$

VI. Discussion

To initialize the learning-based robust controller, the hyperparameters of the kernel function (3) must be specified and provide an initial guess for the uncertainty. There are two options to do this, the first of which is typical for robust control: we assume that we know the prior uncertainty in our system and derive a robust controller that is robustly stable against all uncertainties. After operating the system for some time the hyperparameters are calculated using the observed data. As we operate and observe the system our knowledge of the system improves and the prior controller is replaced as soon as the online calculated robust controller outperforms the *a priori* one; that is, the γ value is smaller. The second option is typical for robotics applications, where either an expert controls the system or the system is allowed to fail while identifying the hyperparameters. This allows the GP to gather data prior to applying our learning-based framework. Whichever method is chosen, in the long term additional operation data provides more information about the unknown dynamics and the controller improves.

As for all system identification methods the system needs to be excited sufficiently. If this is not the case, the hyperparameters may not reflect the true uncertainty in the system.

The proposed approach learns a linear model around a desired operating point. The controller only stabilizes the system in the vicinity of the operating point. For global stability, one would have to turn to nonlinear robust control. However, since we use a nonlinear method for the model update the approach presented in this paper is extendable to robust tracking of nonlinear systems or gain scheduling.

Lastly, if the functional form of $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1) is known, other methods such as (extended) Kalman Filters are better suited for learning, since they use this additional knowledge about the system. However, if $\mathbf{g}(\mathbf{x}, \mathbf{u})$ is unknown the proposed method has an advantage over assuming $\mathbf{g}(\mathbf{x}, \mathbf{u})$ to be linear (as, for example, done in a Kalman filter): the hyperparameters define a region in which the system behaves linearly around the operating point, which enables GPs to learn only from data points within this linear domain. Linear methods would try to fit a linear model to the entire state space.

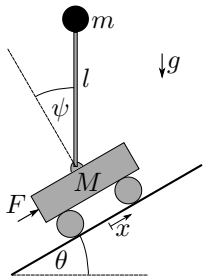

Fig. 2. Simulation example: an inverted pendulum on a tilted slope.

VII. Example

In this section, we demonstrate our approach for an inverted cart-pendulum system on a slope (Fig. 2). The position of the cart is given by x, the pendulum angle is ψ , and F is a force that serves as the control input. The frictionless surface is tilted by an angle θ. The cart has a mass M. The massless pendulum is of length l with a mass m attached at its end. The nonlinear, non-minimum-phase equations of motion governing the behavior of the system are given by

$$\begin{aligned} \ddot{x} &= \big(F - (m+M)g\sin(\theta) - mg\cos(\psi)\sin(\psi-\theta) \\ &\quad + ml\dot{\psi}^2\sin(\psi)\big)/\big(M + m\sin^2(\psi)\big), \\ \ddot{\psi} &= \big((m+M)g\sin(\psi-\theta) - \cos\psi\,(F-(m+M)g\sin\theta) \\ &\quad - ml\dot{\psi}^2\sin\psi\cos\psi\big)/\big(l\,\big(M + m\sin^2\psi\big)\,\big). \end{aligned} \tag{23}$$

Depending on the inclination of the slope, the steady-state input and equilibrium point of the system change. Defining the system state as $\mathbf{x} = (x, \dot{x}, \psi, \dot{\psi})$ and the input $\mathbf{u} = F$, the equilibrium point and corresponding steady-state input are given by $\mathbf{x}_s = (0, 0, \theta, 0)$ and $\mathbf{u}_s = (m+M)g\sin(\theta)$, respectively, and explicitly depend on θ. The *a priori* model (23) is discretized with a sampling time of $T_s = 0.05$s to arrive at the required model representation in (1).

To illustrate the proposed learning-based robust control approach, we assume that all modeled system parameters have an error and that the real slope has an inclination of $30°$, while the model assumes $\theta = 0°$. The modeled and real parameters are shown in Tab. I. As a result, the initial operating point and steady-state input are incorrect.

We start with a robust controller that stabilizes the system (cf. Sec. VI). The system is excited using an input signal drawn from a uniform distribution. This is not necessary, but ensures that a broad spectrum of states is explored quickly. From the first 50 input-output samples the hyperparameters are learned. With these hyperparameters, the robust

TABLE I
Modeled and Real Parameters (Fig. 2).

	real	modeled	error
M [kg]	1.5	1.4	0.1
m [kg]	0.175	0.16	0.015
l [m]	0.28	0.26	0.02
θ [deg]	30°	0°	30°

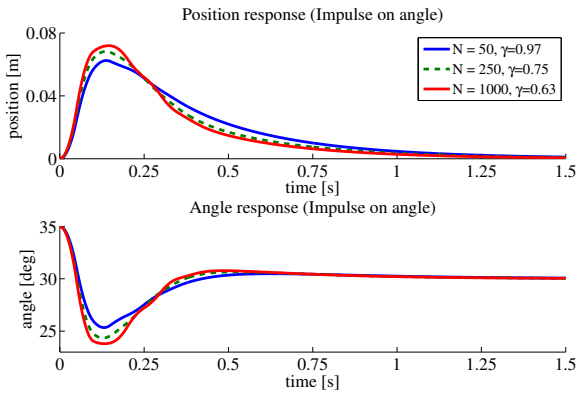

Fig. 3. System response of the learning-based robust controller to an impulse on the angle. The performance of the controller increases (that is, γ decreases) as the Gaussian Process learns the system dynamics from an increasing number of input-output samples N. After 1000 samples (red line) the performance index is 0.63 and the controller reacts more aggressively.

controllers are calculated after different learning periods, $N = 50, 250$ and 1000 (see Sec. IV and V). For simplicity a straight-forward control objective is chosen, which may not be appropriate for real-world examples. The matrices corresponding to the objective are:

$$\mathbf{C}_z = \begin{bmatrix} \mathbf{Q} \\ \mathbf{0}_{m\times n} \end{bmatrix}, \ \mathbf{D}_{zu} = \begin{bmatrix} \mathbf{0}_{n\times m} \\ \mathbf{R} \end{bmatrix}, \ \mathbf{B}_w = \begin{bmatrix} 0 & 0 \\ T_s & 0 \\ 0 & 0 \\ 0 & T_s \end{bmatrix} \quad (24)$$

and $\mathbf{D}_{zw} = \mathbf{0}$, where $\mathbf{Q}$ and $\mathbf{R}$ are diagonal matrices corresponding to a weighting of state and input costs. The matrix $\mathbf{B}_w$ models two disturbance forces acting on the masses M and m.

In Fig. 3 the response of the nonlinear system with the learning-based controllers to an impulse of 5° on the angle is shown.

It can be seen that the performance criterion γ decreases as we learn a more accurate model of the dynamics. For the chosen objective function, this results in a more aggressive controller with decreasing settling time as N increases. Overall the system remains robustly stable, while its performance increases. For better comparability, we assume the accurate operating point is known for Fig. 3. In normal operation, the steady-state input is identified, but small errors lead to a negligible steady-state error in the position of $\simeq 0.01\text{m}$.

VIII. Conclusion

In this paper, a method to combine online learning with robust control theory has been introduced with the goal of designing a learning controller that guarantees stability while gradually improving performance. A Gaussian Process (GP) was used to learn a nonlinear model of the unknown dynamics. Based on this model the *a priori* operating point was corrected and a linearization of the learned system model about this point was obtained including uncertainty information. Finally, a controller that is robust to the learned model uncertainties was calculated by solving a convex optimization problem. This control law is updated as better models of the system are learned, which gradually increases the control performance. The entire process was illustrated on an inverted pendulum. The control performance increased as the model improved. Ultimately, the GP framework has proven to be a powerful tool for combining nonlinear learning methods with standard robust control theory.

References

[1] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with processes," in *Proc. of the European Control Conference (ECC)*, 2015, pp. 2501–2506.

[2] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Dover Publications, 2007.

[3] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer, 2013.

[4] L. Ljung, *System identification: theory for the user*. Pearson Education, 1998.

[5] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[6] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[8] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. Cambridge: MIT Press, 2006.

[9] M. P. Deisenroth, J. Peters, and C. E. Rasmussen, "Approximate dynamic programming with Gaussian processes," in *Proc. of the American Control Conference (ACC)*, 2008, pp. 4480–4485.

[10] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2014, pp. 4029–4036.

[11] R. Murray-Smith and D. Sbarbaro, "Nonlinear adaptive control using nonparametric Gaussian process prior models," in *Proc. of the 15th IFAC World Congress on Automatic Control*, 2002.

[12] W. S. Lee, B. Anderson, R. Kosut, and I. Mareels, "On adaptive robust control and control-relevant system identification," in *Proc. of the American Control Conference (ACC)*, 1992, pp. 2834–2841.

[13] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice Hall Upper Saddle River, NJ, 1998, vol. 104.

[14] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*. Wiley New York, 2007, vol. 2.

[15] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[16] P. M. Van Den Hof and R. J. Schrama, "Identification and control - closed-loop issues," *Automatica*, vol. 31, no. 12, pp. 1751–1770, 1995.

[17] S. G. Douma and P. M. Van den Hof, "Relations between uncertainty structures in identification for robust control," *Automatica*, vol. 41, no. 3, pp. 439–457, 2005.

[18] J. Schoukens, T. Dobrowiecki, and R. Pintelon, "Parametric and nonparametric identification of linear systems in the presence of nonlinear distortions - a frequency domain approach," *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 176–190, 1998.

[19] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[20] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[21] N. Bedioui, S. Salhi, and M. Ksouri, "Robust stabilization approach and $\mathcal{H}_\infty$ performance via static output feedback for a class of nonlinear systems," *Mathematical Problems in Engineering*, 2009.

[22] J. Ko and D. Fox, "Learning GP-bayesfilters via Gaussian process latent variable models," *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, 2011.

[23] R. D. Turner, M. P. Deisenroth, and C. E. Rasmussen, "State-space inference and learning with Gaussian processes," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 868–875.

[24] S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. Society for Industrial and Applied Mathematics, 1994, vol. 15.
[25] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.
[26] F. Berkenkamp and A. Schoellig. (2014) Robust control proof. [Online]. Available: http://www.tiny.cc/robust_control