Exploiting Differential Flatness for Robust Learning-Based Tracking Control using Gaussian Processes

Melissa Greeff and Angela P. Schoellig

***Abstract*— Learning-based control has shown to outperform conventional model-based techniques in the presence of model uncertainties and systematic disturbances. However, most state-of-the-art learning-based nonlinear trajectory tracking controllers still lack any formal guarantees. In this paper, we exploit the property of differential flatness to design an online, robust learning-based controller to achieve both high tracking performance and probabilistically guarantee a uniform ultimate bound on the tracking error. A common control approach for differentially flat systems is to try to linearize the system by using a feedback (FB) linearization controller designed based on a nominal system model. Performance and safety are limited by the mismatch between the nominal model and the actual system. Our proposed approach uses a nonparametric Gaussian Process (GP) to both improve FB linearization and quantify, probabilistically, the uncertainty in our FB linearization. We use this probabilistic bound in a robust linear quadratic regulator (LQR) framework. Through simulation, we highlight that our proposed approach significantly outperforms alternative learning-based strategies that use differential flatness.**

I. INTRODUCTION

The heightened interest in using learning-based control to achieve high-accuracy tracking has, in part, been driven by advanced robotic applications where accurate models are required but difficult to derive. In practice, many current learning-based control techniques achieve good tracking performance by learning a fairly accurate nonlinear dynamics model. However, the adoption of these techniques has been limited to applications that are not safety-critical as these techniques fail to provide any rigorous analysis of safety such as constraint satisfaction or stability and convergence. The need to provide guarantees while using a data-driven learned model is still a key challenge for robotics.

The limitation of many learning-based approaches, for example, standard Neural Networks (NNs), is that they are ill-suited to quantify any mismatch between the learned model and the real dynamics. For this reason, nonparametric approaches, such as Gaussian Processes (GPs), have gained popularity within the control community as they can provide uncertainty estimates for their predictions [1], [2]. The question then is: *how can we efficiently use this uncertainty measure in the control loop?* This depends on the assumed dynamic structure of the actual system and the selected part of the dynamics to be learned.

The authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS) and the Vector Institute for Artificial Intelligence, Toronto. Email: melissa.greeff@mail.utoronto.ca, schoellig@utias.utoronto.ca

This work was supported by Drone Delivery Canada, Defence Research and Development Canada, and Natural Sciences and Engineering Research Council of Canada.

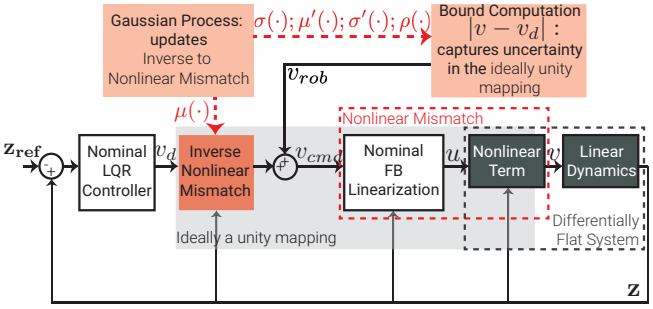


Fig. 1. Our proposed architecture exploiting differential flatness for robust learning-based tracking control has three key components: 1) *Updating the Inverse Nonlinear Mismatch*: a GP learns the model error resulting from using a nominal feedback (FB) linearization; 2) *Bound Computation*: the properties of GPs are exploited to estimate a probabilistic bound on how well we linearize the system; 3) *Robust LQR*: this bound is combined with a nominal LQR to guarantee (probabilistically) an ultimate bound on the tracking error.

GPs can be used to learn the forward nonlinear system dynamics. For example, one common approach is to use this learned GP model in a nonlinear model predictive control framework where the uncertainty estimate from the GP is used to tighten constraints [3]. However, this approach provides no stability analysis of the controlled system. Another approach linearizes the learned nonlinear model about an operating point, which, combined with the uncertainty estimate from the GP is used in a linear robust control framework [4]. However, this robust learning-based controller is limited to stabilization tasks.

In this paper, we impose two structural properties on the system dynamics. Firstly, we assume the system is control-affine. Secondly, we assume the system dynamics exhibit a property known as differential flatness [5]. Many first-principle models of physical systems, for example, quadrotors, cranes and cars with trailers, exhibit this property [6]. Intuitively, differential flatness allows us to separate the nonlinear model into a linear dynamics component and a nonlinear term, see Fig. 1. This property is commonly used in feedback (FB) linearization controllers which attempt to cancel the nonlinear term such that outer-loop linear controllers, for example, linear quadratic regulators (LQR), can be designed based on the linear dynamics [7].

Related work on FB linearization has used learning-based strategies in one of two ways:

Strategy 1: The first strategy is to simply update a nominal FB linearization controller with a data-driven learned model. In [8], reinforcement learning was used for FB linearization by learning the inverse model of the nonlinear term of the differentially flat system. In [9], a GP was used to learn

the forward model of the nonlinear term, such that, by transferring knowledge of the control-affine structure into the kernel function, it could be used to update a FB linearization controller. Approaches in this category have not provided guarantees of stability or tracking convergence as they do not quantify how well the learned FB linearization controller cancels the nonlinear term and, consequently, how well it linearizes the system.

Strategy 2: The second strategy is to use a data-driven model to quantify how well a nominal FB linearization controller linearizes the system. We call the model error between the nominal FB linearization controller and nonlinear term the *Nonlinear Mismatch*. In [10], a GP is used to learn a forward model of the *Nonlinear Mismatch* for Lagrangian systems. The learned prediction and uncertainty model is used to generate a bound on how well the nominal FB linearization controller linearizes the system. The bound is then used in a linear, robust outer-loop controller. While this strategy provides tracking guarantees, it is conservative as it does not update or improve the FB linearization.

Our proposed approach combines ideas from both strategies. We use a GP to learn an inverse model of the *Nonlinear Mismatch*. As demonstrated in Fig. 1, we use our learned model to update the *Inverse Nonlinear Mismatch* which attempts to cancel the *Nonlinear Mismatch*. Considering the key idea from *Strategy 2*, we quantify how well we linearize the system. To do this we generate a probabilistic upper bound on the difference between the designed desired input and the actual input seen by the linear system dynamics. Finding this bound requires exploiting the control-affine flatness structure and several key properties of GPs. We use this bound in an additional robust term which, coupled with a nominal LQR, probabilistically guarantees stability, or more specifically, an ultimate bound on the tracking error. This paper has three key contributions:

- We present a novel approach that uses a GP to both improve the FB linearization and quantify how well we are able to linearize the system.
- We demonstrate how our quantified uncertainty can be combined with a standard robust LQR to probabilistically guarantee an ultimate bound on the tracking error.
- We show through simulations how our proposed approach results in improved tracking performance over both *Strategy 1* (only improving the FB linearization) and *Strategy 2* (only quantifying how well a nominal controller linearizes the system).

To the best of our knowledge, this is the first approach that achieves robust, online learning-based control with formal guarantees on the tracking error for generic control-affine differentially flat systems.

II. PROBLEM STATEMENT

Consider a single-input single-output (SISO), control-affine system, with state $\mathbf{x} \in \mathbb{R}^n$, and input $u \in \mathbb{R}$:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u, \tag{1}$$

where $f(\mathbf{x})$ and $g(\mathbf{x})$ are *unknown* functions.

Assumption 1: The system (1) is differentially flat in the *known* output $y = h(\mathbf{x}) \in \mathbb{R}$.

Assumption 2: We have a SISO, control-affine nominal system model:

$$\dot{\mathbf{x}} = \hat{f}(\mathbf{x}) + \hat{g}(\mathbf{x})u \tag{2}$$

that is also differentially flat in the output $y = h(\mathbf{x}) \in \mathbb{R}$. Under Assumptions 1-2, our goal is to design a control law u such that:

- we achieve high-accuracy output tracking;
- we guarantee that the overall, closed-loop system satisfies robust stability (in the sense that the tracking error is bounded despite model uncertainties).

To address this problem, we propose a learning-based control law that, by exploiting the differential flatness structure, both updates an inner-loop feedback (FB) linearization controller (red box with *Inverse Nonlinear Mismatch* in Fig. 1) and a robust linear controller. Our approach uses a GP as it allows us to quantify uncertainty. We present the proposed approach for the above SISO problem, however, a similar methodology could be applied to the multi-input, multi-output (MIMO) problem.

III. BACKGROUND

A. Uniform Ultimate Boundedness

We use Lyapunov theory to prove that using our proposed controller can probabilistically guarantee an ultimate bound on the tracking error. To do this, we make use of the following definition and theorem from [11].

Definition 1 (Uniform Ultimate Boundedness): A solution $\mathbf{e}(t) : [t_0, \infty) \to \mathbb{R}^n$ to $\dot{\mathbf{e}} = \zeta(\mathbf{e})$ with initial condition $\mathbf{e}(t_0) = \mathbf{e}_0$ is uniformly ultimately bounded (u.u.b.) with respect to a set S if there is a non-negative constant $T(\mathbf{e}_0, S)$ such that $\mathbf{e}(t) \in S \quad \forall t > t_0 + T(\mathbf{e}_0, S)$.

Theorem 1: Let $V(\mathbf{e}(t))$ be a Lyapunov function and let S be any level set of $V(\mathbf{e}(t))$. Then $\mathbf{e}(t)$ is u.u.b. with respect to S if $\dot{V} < 0$ for $\mathbf{e}(t)$ outside of S.

B. Differential Flatness

In the following Lemma 1, we highlight how differential flatness leads to both our system (1) and nominal model (2) having an identical linear dynamics component. However, their nonlinear terms, see Fig. 1, will differ. In Section IV, we exploit this structure in the learning controller design.

Definition 2 (Differential Flatness [5]): A SISO nonlinear system (1) is differentially flat in output y if there exist smooth, invertible functions such that: $\mathbf{x} = \Phi(\mathbf{z})$, $u = \Psi^{-1}(\mathbf{z}, v)$, where $\mathbf{z} = [y, \dot{y}, ..., y^{(n-1)}]^T$, $v = y^{(n)}$. Furthermore, if (1) is control-affine then $\Psi^{-1}(\mathbf{z}, v)$ is also control-affine, i.e., we can write $\Psi^{-1}(\mathbf{z}, v) = \alpha(\mathbf{z}) + \beta(\mathbf{z})v$.

Lemma 1: If our system (1) is differentially flat in output y, then it is equivalent to:

$$v = \frac{u - \alpha(\mathbf{z})}{\beta(\mathbf{z})}, \tag{3}$$

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}v, \tag{4}$$

where our linear dynamics (4) are an integrator chain of degree n and the nonlinear term is given by (3) [5], [7].

Note that since functions $f(\mathbf{x})$ and $g(\mathbf{x})$ are unknown for our system (1), the functions $\alpha(\mathbf{z})$ and $\beta(\mathbf{z})$ in our nonlinear term (3) are also unknown. Lemma 1 can also be applied to our nominal system model (2).

C. Gaussian Processes (GPs)

In this section, we highlight two key properties of GPs that are leveraged in our proposed approach in Section IV. GP regression is a nonparametric approach that is used to approximate a nonlinear map, $v_e(\mathbf{a}) : \mathbb{R}^{\dim(\mathbf{a})} \to \mathbb{R}$, from the input $\mathbf{a}$ to the function value $v_e(\mathbf{a})$. Note that the function is named $v_e(\cdot)$ to be consistent with notation in Section IV. It does this by assuming that the function values $v_e(\mathbf{a})$, associated with different inputs $\mathbf{a}$, are random variables and that any finite number of these random variables have a joint Gaussian distribution. This nonparametric approach still requires us to define two priors: a prior mean function of $v_e(\mathbf{a})$, generally set to zero, and a covariance or kernel function $k(\cdot,\cdot)$ which encodes a form of similarity between two input points and their associated function values. For example, a common kernel function is the squared-exponential (SE) function:

$$k(\mathbf{a}_i,\mathbf{a}_j) = \sigma_f^2 \text{exp}\left(-\frac{1}{2}(\mathbf{a}_i-\mathbf{a}_j)^T\mathbf{L}^{-2}(\mathbf{a}_i-\mathbf{a}_j)\right) + \delta_{ij}\sigma_\eta^2,$$

which is characterized by three types of hyperparameters: the prior variance σ_f^2, measurement noise σ_η^2, where $\delta_{ij}=1$ if $i=j$ and 0 otherwise, and the length scales, or the diagonal elements of the diagonal matrix $\mathbf{L}$, which encode a measure of how quickly the function $v_e(\mathbf{a})$ changes with respect to $\mathbf{a}$. These hyperparameters can be optimized by solving a maximum log-likelihood problem [12].

Prediction: This GP framework can be used to predict the function value at any query point $\mathbf{a}^*$ based on N noisy observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{v}_e(\mathbf{a}_i)\}_{i=1}^N$. It does this by using the key underlying principle that the observed data, or function values, and the function value at a query point $v_e(\mathbf{a}^*)$ are all jointly normal:

$$\begin{bmatrix}\hat{\mathbf{v}}_\mathbf{e}\\ v_e(\mathbf{a}^*)\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{k}^T(\mathbf{a}^*)\\ \mathbf{k}(\mathbf{a}^*) & k(\mathbf{a}^*,\mathbf{a}^*)\end{bmatrix}\right),$$

where $\hat{\mathbf{v}}_\mathbf{e} = [\hat{v}_e(\mathbf{a}_1),...,\hat{v}_e(\mathbf{a}_N)]^T$ is the vector of observed function values, the covariance matrix has entries $\mathbf{K}_{(i,j)} = k(\mathbf{a}_i,\mathbf{a}_j)$, $i,j \in 1,...,N$, and $\mathbf{k}(\mathbf{a}^*) = [k(\mathbf{a}^*,\mathbf{a}_1),...,k(\mathbf{a}^*,\mathbf{a}_N)]$ is the vector of the covariances between the query point $\mathbf{a}^*$ and the observed data points in $\mathcal{D}$. By the conditioning property of Gaussian distributions, the mean and variance at our query point $\mathbf{a}^*$ conditioned on the observed data $\mathcal{D}$ are given by:

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

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

Derivatives: By using the fact that the derivative is a linear operator, it can be shown that the derivative of a GP with respect to v, where v (variable name chosen to match future use case in Section IV) represents an element of input $\mathbf{a}$, is a GP as well [12]. Consequently, the observed data and the function derivative with respect to input v at the query point $\mathbf{a}^*$ are also jointly Gaussian:

$$\begin{bmatrix}\hat{\mathbf{v}}_\mathbf{e}\\ \frac{\partial v_e(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \frac{\partial \mathbf{k}^T(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\\ \frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*} & \frac{\partial^2 k(\mathbf{a},\mathbf{a})}{\partial v \partial v}\Big|_{\mathbf{a}^*}\end{bmatrix}\right).$$

Similarly, by the conditioning property of joint Gaussian distributions, the mean and variance of the derivative at our query point $\mathbf{a}^*$ with respect to input v conditioned on the data $\mathcal{D}$ is given by:

$$\mu'(\mathbf{a}^*) = \frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\mathbf{K}^{-1}\hat{\mathbf{v}}_\mathbf{e}, \tag{7}$$

$$\sigma'^2(\mathbf{a}^*) = \frac{\partial^2 k(\mathbf{a},\mathbf{a})}{\partial v \partial v}\Big|_{\mathbf{a}^*} - \frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\mathbf{K}^{-1}\left(\frac{\partial \mathbf{k}(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\right)^T. \tag{8}$$

We can also use this property of a derivative of a GP to infer the coefficient of correlation $\rho(\mathbf{a}^*)$ between the function value at a query point, $v_e(\mathbf{a}^*)$, and its derivative with respect to input v at a query point, $\frac{\partial v_e(\mathbf{a})}{\partial v}\big|_{\mathbf{a}^*}$. Using the kernel function, we first find the covariance $\text{COV}(\cdot,\cdot)$ between the function value $v_e(\mathbf{a}^*)$ and its derivative $\frac{\partial v_e(\mathbf{a})}{\partial v}\big|_{\mathbf{a}^*}$ using $\text{COV}\left(v_e(\mathbf{a}^*), \frac{\partial v_e(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\right) = \frac{\partial k(\mathbf{a}^*,\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}$ [12]. Then by the definition of the coefficient of correlation:

$$\rho(\mathbf{a}^*) = \frac{\text{COV}\left(v_e(\mathbf{a}^*), \frac{\partial v_e(\mathbf{a})}{\partial v}\Big|_{\mathbf{a}^*}\right)}{\sigma(\mathbf{a}^*)\sigma'(\mathbf{a}^*)}, \tag{9}$$

where $\sigma(\mathbf{a}^*)$ is found from (6) and $\sigma'(\mathbf{a}^*)$ is found from (8).

IV. METHODOLOGY

We exploit the differential flatness of both our system (1) and our nominal system model (2), and apply Lemma 1 from Section III-B. Following (3)-(4), our nominal system model is equivalent to (see Fig. 1):

$$v_{cmd} = \frac{u-\hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})}, \tag{10}$$

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}v_{cmd}. \tag{11}$$

We exploit this structure of our nominal system model to design a nominal FB linearization controller:

$$u = \hat{\alpha}(\mathbf{z}) + \hat{\beta}(\mathbf{z})v_{cmd}, \tag{12}$$

where the commanded input v_{cmd} can be designed based on the linear model (11).

As seen in Fig. 1, we term the mapping from commanded input v_{cmd} to the actual input v (seen by the linear system) the *Nonlinear Mismatch*. If our nominal model was identical to our system, i.e. $\alpha(\cdot) = \hat{\alpha}(\cdot)$ and $\beta(\cdot) = \hat{\beta}(\cdot)$, this would be a unity mapping. However, given the mismatch between the nominal model and the actual system, this will not be the case. In our proposed scheme we attempt to correct for this *Nonlinear Mismatch* in two phases:

Learning Phase: We propose learning the *Inverse Nonlinear Mismatch*, that is, the mapping from actual input v to commanded input v_{cmd}.

Running Phase: We use the learned *Inverse Nonlinear Mismatch* to correct the nominal FB linearization controller as shown in Fig. 1. The learned *Inverse Nonlinear Mismatch*

takes in a desired input v_d and outputs a commanded input v_{cmd}. In a similar vein to FB linearization, if our *Inverse Nonlinear Mismatch* exactly cancels our *Nonlinear Mismatch*, then $v = v_d$. Of course, in practice our *Inverse Nonlinear Mismatch* will not exactly correct for our *Nonlinear Mismatch*, however, in Section IV-B we use the GP uncertainty to estimate a probabilistic bound on the quality of this correction. In Section IV-C, we show how this bound can be used in a robust LQR controller to guarantee (probabilistically) an ultimate bound on the trajectory tracking error (see Section V).

A. Update to Inverse Nonlinear Mismatch

Learning Phase: To find the *Inverse Nonlinear Mismatch* we write the commanded input v_{cmd} in terms of state $\mathbf{z}$ and input v by plugging (12) into (3), $v_{cmd} = \frac{\alpha(\mathbf{z})-\hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})} + \frac{\beta(\mathbf{z})}{\hat{\beta}(\mathbf{z})}v$, or equivalently $v_{cmd} = v + v_e(\mathbf{z}, v)$ where the difference $v_{cmd} - v$ is given by the function:

$$v_e(\mathbf{z}, v) = \frac{\alpha(\mathbf{z}) - \hat{\alpha}(\mathbf{z})}{\hat{\beta}(\mathbf{z})} + \frac{\beta(\mathbf{z}) - \hat{\beta}(\mathbf{z})}{\hat{\beta}(\mathbf{z})}v. \tag{13}$$

If our model is identical to the system, then the *Inverse Nonlinear Mismatch* is unity, i.e. $v_{cmd} = v$. We propose using a GP framework to approximate (13) by considering a set of N past observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{v}_e(\mathbf{a}_i)\}_{i=1}^N$ where inputs to the model are given by $\mathbf{a}_i = \{\mathbf{z}_i, v_i\}$, and we assume we have noisy measurements of the true function $\hat{v}_e(\mathbf{a}_i) = (v_{cmd} - v) + \eta$ with $\eta = \mathcal{N}(0, \sigma_\eta^2)$.

Running Phase: During the run phase, we consider a query input $\mathbf{a}^* = \{\mathbf{z}, v_d\}$ where v_d is some desired input computed by an outer-loop linear controller. By using the properties of joint Gaussian distributions we predict $v_e(\mathbf{a}^*)$ conditioned on the data set $\mathcal{D}$ by: $v_e(\mathbf{a}^*)|\mathcal{D} = \mathcal{N}(\mu(\mathbf{a}^*), \sigma^2(\mathbf{a}^*))$ where $\mu(\mathbf{a}^*)$ and $\sigma^2(\mathbf{a}^*)$ are obtained using (5) and (6), respectively.

We update the *Inverse Nonlinear Mismatch* using the mean from our prediction, i.e.,

$$v_{cmd} = v_d + \mu(\mathbf{a}^*) + v_{rob}, \tag{14}$$

where v_{rob} is an additional robustness input as described in Section IV-C. To find the input u, we feed (14) through the nominal FB linearization controller (12). As seen in Fig. 1, ideally we learn an updated v_{cmd} (14) to achieve a unity mapping, i.e., $v = v_d$. However, in practice there is a model uncertainty in this *ideally unity mapping* that we need to quantify in order to provide tracking guarantees. To do this, we can find the actual input v (sent to the system), from (3), (12), (14): $v = v_d + \left(\frac{\hat{\alpha}(\mathbf{z})-\alpha(\mathbf{z})}{\beta(\mathbf{z})} + \frac{\hat{\beta}(\mathbf{z})-\beta(\mathbf{z})}{\beta(\mathbf{z})}v_d\right) + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) + v_{rob})$. By noticing that $\frac{\hat{\alpha}(\mathbf{z})-\alpha(\mathbf{z})}{\beta(\mathbf{z})} + \frac{\hat{\beta}(\mathbf{z})-\beta(\mathbf{z})}{\beta(\mathbf{z})}v_d = -\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_e(\mathbf{a}^*)$, this reduces to:

$$v = v_d + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right) + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob}. \tag{15}$$

It is clear that if our mean $\mu(\mathbf{a}^*)$ perfectly characterizes $v_e(\mathbf{a}^*)$, we have exactly learned the *Inverse Nonlinear Mismatch* and $v = v_d$ without a robustness term v_{rob}. However, in practice this is not the case. We, therefore, require an estimate of a bound on $\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))$ such that we can use it in a robust control framework to design v_{rob} such that we can probabilistically guarantee a robust stability by establishing an ultimate bound on the tracking error.

B. Bound Computation

We propose finding a probabilistic bound c such that:

$$\Pr\left\{\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right)\right| < \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\right\} \geq 1 - \delta, \tag{16}$$

where $\delta \in (0, 1)$ is some user-selected small value. We can rewrite (16), arbitrarily dividing the probability $1 - \delta$, by introducing an intermediate bound $\hat{c}$ such that:

$$\Pr\left\{\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}\left(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)\right)\right| < \hat{c}\right\} \geq \sqrt{1-\delta}, \tag{17}$$

and

$$\Pr\left\{\hat{c} < \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\right\} \geq \sqrt{1-\delta}. \tag{18}$$

Consequently, to compute bound c we first find the intermediate bound $\hat{c}$ in (17). To compute such a bound, we exploit the derivative properties of GPs. From (10), we see that $\hat{\beta}(\mathbf{z}) = \frac{\partial u}{\partial v_{cmd}}$, and from (3), $\beta(\mathbf{z}) = \frac{\partial u}{\partial v}$. Consequently, the ratio is given by the partial derivative relationship $\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})} = \frac{\partial v}{\partial v_{cmd}}$. Using $v_{cmd} = v + v_e(\mathbf{z}, v)$, this ratio becomes:

$$\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})} = \frac{1}{1 + \frac{\partial v_e(\mathbf{z},v)}{\partial v}}.$$

We utilize a key characteristic of the GP framework, from Section III-C, which is that the derivative of a GP is a GP as well. In other words, given that we have learned the function $v_e(\mathbf{z}, v)$ as a GP, the partial derivative $\frac{\partial v_e(\mathbf{z},v)}{\partial v}$ is also a GP and we can predict its value at a query point $\mathbf{a}^* = \{\mathbf{z}, v_d\}$ conditioned on the data $\mathcal{D}$ as $\frac{\partial v_e(\mathbf{a}^*)}{\partial v}|\mathcal{D} = \mathcal{N}(\mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$ where $\mu'(\mathbf{a}^*)$ and $\sigma'^2(\mathbf{a}^*)$ can be found from (7) and (8), respectively.

Our analysis requires sufficient training data $\mathcal{D}$ and a GP kernel that can model our model mismatch to make the following two simplifying assumptions (cf. [4]):

Assumption 3: The actual error $\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)$ is normally distributed and described by the random variable: $Y := \mu(\mathbf{a}^*) - v_e(\mathbf{a}^*) \sim \mathcal{N}(0, \sigma^2(\mathbf{a}^*))$.

Assumption 4: The partial derivative $\frac{\partial v_e(\mathbf{a}^*)}{\partial v}$ is also normally distributed such that we can define the random variable: $X := 1 + \frac{\partial v_e(\mathbf{a}^*)}{\partial v} \sim \mathcal{N}(1 + \mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$.

Furthermore, X and Y are jointly correlated with some coefficient of correlation $\rho(\mathbf{a}^*)$ given by (9). We write this as a bivariate correlated normal random variable: $(Y, X) \sim \mathcal{N}(0, 1 + \mu'(\mathbf{a}^*), \sigma^2(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*), \rho(\mathbf{a}^*)) = \mathcal{N}(\mu_Y, \mu_X, \sigma_Y^2, \sigma_X^2, \rho)$.

We rewrite our probability bound (17) as: $\Pr\{|\frac{Y}{X}| < \hat{c}\} \geq 1-\delta$, where the left-hand side or cumulative density function is found from $\Pr\{|\frac{Y}{X}| < \hat{c}\} = F(\hat{c}) - F(-\hat{c})$ where $F(\hat{c}) = \Pr\{\frac{Y}{X} < \hat{c}\}$ has an analytical form [13]:

$$F(\hat{c}) = L\left(\frac{t_a - t_b t_{\hat{c}}}{\sqrt{1 + t_{\hat{c}}^2}}, -t_b, \frac{t_{\hat{c}}}{\sqrt{1 + t_{\hat{c}}^2}}\right) + L\left(\frac{t_b t_{\hat{c}} - t_a}{\sqrt{1 + t_{\hat{c}}^2}}, t_b, \frac{t_{\hat{c}}}{\sqrt{1 + t_{\hat{c}}^2}}\right),$$

where $L(\cdot,\cdot,\cdot)$ is the bivariate normal integral and $t_a = \sqrt{\frac{1}{1-\rho^2}}(\frac{\mu_Y}{\sigma_Y} - \rho\frac{\mu_X}{\sigma_X})$, $t_b = \frac{\mu_X}{\sigma_X}$, $t_{\hat{c}} = \sqrt{\frac{1}{1-\rho^2}}(\frac{\sigma_X}{\sigma_Y}\hat{c} - \rho)$.

Therefore, to find the intermediate probabilistic bound $\hat{c}$, at each query point $\mathbf{a}^*$, we solve the nonlinear optimization problem:

$$\begin{aligned} \min \quad & \hat{c} \\ \text{s.t.} \quad & \hat{c} \geq 0 \\ & F(\hat{c}) - F(-\hat{c}) \geq \sqrt{1-\delta}. \end{aligned} \tag{19}$$

To find bound c in (16) we use the intermediate bound $\hat{c}$ found from (19). To do this, we rewrite (18) as $\Pr\left\{\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c < \hat{c}\right\} \leq \sqrt{1-\delta}$. By introducing random variables $W \sim \mathcal{N}(c, 0)$ and $X \sim \mathcal{N}(1+\mu'(\mathbf{a}^*), \sigma'^2(\mathbf{a}^*))$, by Assumption 4 above, we can rewrite the inequality as a special case of the ratio of uncorrelated normal random variables $\Pr\{\frac{W}{X} < \hat{c}\} = F(\hat{c})$ where $(W, X) \sim \mathcal{N}(c, 1+\mu'(\mathbf{a}^*), 0, \sigma'^2(\mathbf{a}^*), 0)$. In this case, we can then find bound c (mean of the numerator) such that $F(\hat{c}) = \sqrt{1-\delta}$.

C. Robust Linear Quadratic Regulator

Our aim is to track a reference trajectory with reference state $\mathbf{z_{ref}} = [y_{ref}, \dot{y}_{ref}, ..., y_{ref}^{(n-1)}]^T$ and reference input $v_{ref} = y_{ref}^{(n)}$ where y_{ref} is the reference output. We design the desired input v_d in (14) using a nominal LQR:

$$v_d = -\tilde{\mathbf{K}}(\mathbf{z} - \mathbf{z_{ref}}) + v_{ref}. \tag{20}$$

The gain $\tilde{\mathbf{K}} = \mathbf{R^{-1}B^TP}$ is found by solving the algebraic Riccati equation $\mathbf{A}^T\mathbf{P} + \mathbf{PA} - \mathbf{PBR^{-1}B^TP} + \mathbf{Q} = \mathbf{0}$, $\mathbf{Q}, \mathbf{R} > 0$ where matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained from the linear dynamics (11). The robustness term in (14) is designed as:

$$v_{rob} = \begin{cases} -c\frac{\mathbf{B^TPe}}{||\mathbf{B^TPe}||}, & \text{if } ||\mathbf{B^TPe}|| > \epsilon \\ -c\frac{\mathbf{B^TPe}}{\epsilon}, & \text{otherwise} \end{cases} \tag{21}$$

where $\mathbf{e} = \mathbf{z} - \mathbf{z_{ref}}$ is the tracking error, c is the probabilistic bound found from (16) and $\epsilon > 0$ is some small user-selected parameter and $||\cdot||$ denotes the Euclidean norm (cf. [10]).

V. THEORETICAL GUARANTEES

In this section, we show that under our proposed learning-based controller, the trajectory tracking error is uniformly ultimately bounded.

Theorem 2: Consider the differentially flat system (3)-(4) and a smooth bounded reference state $\mathbf{z_{ref}}(t)$ and input $v_{ref}(t)$ trajectory. Suppose that Assumptions 1-4 hold and that bound c satisfies (16). Then the tracking error $\mathbf{e}(t) = \mathbf{z}(t) - \mathbf{z_{ref}}(t)$ is uniformly ultimately bounded (u.u.b.) with probability greater than $1-\delta$ using the proposed robust learning-based control governed by (12), (14), (20) and (21).

Proof: Under the proposed robust learning-based control, the closed-loop dynamics are given by (4) and (15), where $v_d = -\tilde{\mathbf{K}}(\mathbf{z}-\mathbf{z_{ref}}) + v_{ref}$. By exploiting the integrator chain structure of matrices $\mathbf{A}$ and $\mathbf{B}$, we write the closed-loop error dynamics as:

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})\mathbf{e} + \mathbf{B}(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))).$$

We propose the following Lyapunov function $V = \mathbf{e^Te}$ where $\mathbf{P}$ is the positive definite matrix that solves the algebraic Riccati equation, i.e., $\mathbf{A}^T\mathbf{P} + \mathbf{PA} - \mathbf{PBR^{-1}B^TP} + \mathbf{Q} = \mathbf{0}$, $\mathbf{Q}, \mathbf{R} > 0$, or equivalently, $(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})^T\mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}}) + \tilde{\mathbf{S}} = 0$, where $\tilde{\mathbf{S}} = \mathbf{Q} + \tilde{\mathbf{K}}^T\mathbf{R}\tilde{\mathbf{K}} > 0$ since $\mathbf{Q}, \mathbf{R} > 0$. The time derivative of our Lyapunov function is $\dot{V} = 2\mathbf{e^TP}(\mathbf{A} - \mathbf{B}\tilde{\mathbf{K}})\mathbf{e} + 2\mathbf{e^TPB}(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)))$, or equivalently, by using the algebraic Ricatti relationship and defining $\mathbf{w} := \mathbf{B^TPe}$,

$$\dot{V} = -\mathbf{e^T}\tilde{\mathbf{S}}\mathbf{e} + 2\mathbf{w}^T(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))).$$

Since the first term $V_1 := -\mathbf{e^T}\tilde{\mathbf{S}}\mathbf{e} < 0$ is strictly negative, we consider only the second term $V_2 := 2\mathbf{w}^T(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob} + \frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*)))$. There are two cases. *Case 1:* In this case, $||\mathbf{w}|| > \epsilon$ and $v_{rob} = -c\frac{\mathbf{w}}{||\mathbf{w}||}$ in (21). The second term of $\dot{V}$ becomes $V_2 = 2\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c||\mathbf{w}|| + \mathbf{w}^T\left(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))\right)\right)$. We can use the Cauchy-Schwartz inequality to show $V_2 \leq 2\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c||\mathbf{w}|| + ||\mathbf{w}||\left|\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}(\mu(\mathbf{a}^*) - v_e(\mathbf{a}^*))\right|\right)$. Since c is an upper bound that satisfies (16), V_2 is less than zero with probability greater than $1-\delta$. *Case 2:* In this case $||\mathbf{w}|| \leq \epsilon$ and $v_{rob} = -c\frac{\mathbf{w}}{\epsilon}$ in (21). The second term of $\dot{V}$ becomes $V_2 \leq 2\mathbf{w}^T\left(\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}v_{rob} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right) = 2\mathbf{w}^T\left(-\frac{\hat{\beta}(\mathbf{z})}{\beta(\mathbf{z})}c\frac{\mathbf{w}}{\epsilon} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right)$. Since $\max\left(2\mathbf{w}^T\left(-\hat{c}\frac{\mathbf{w}}{\epsilon} + \hat{c}\frac{\mathbf{w}}{||\mathbf{w}||}\right)\right) = \frac{\hat{c}\epsilon}{2}$, and using (18), the second term $V_2 \leq \frac{\hat{c}\epsilon}{2}$. Then $\dot{V} = V_1 + V_2 \leq -\mathbf{e^T}\tilde{\mathbf{S}}\mathbf{e} + \frac{\hat{c}\epsilon}{2} < 0$ provided that:

$$||\mathbf{e}|| > \sqrt{\frac{\hat{c}\epsilon}{2\lambda_{min}(\tilde{\mathbf{S}})}} := B_{\tilde{\mathbf{S}}}.$$

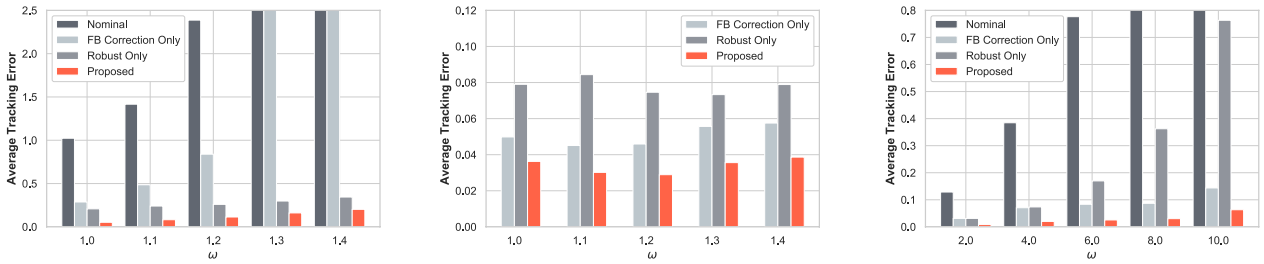
Let S be a level set of V such that it contains $B_{\tilde{\mathbf{S}}}$. Since $\dot{V} < 0$ for $\mathbf{e}(t)$ outside of S, by Theorem 1, $\mathbf{e}(t)$ is u.u.b. with respect to S. Note that ϵ can be chosen to be very small and therefore the ball $B_{\tilde{\mathbf{S}}}$ can be made arbitrarily small. ■

VI. SIMULATION RESULTS

The proposed approach is verified via simulations on A) a SISO 1-D quadrotor moving in the horizontal direction and B) the pole dynamics of an inverted pendulum on a cart (cf. [14]). For both simulations, $\delta = 0.01$ and $\epsilon = 0.1$.

A. 1-D Quadrotor

Our simulation uses the following dynamics $\ddot{x} = T\sin(\theta) - \gamma\dot{x}$, $\dot{\theta} = \frac{1}{\tau}(u - \theta)$, where x is the horizontal position, θ is the pitch angle and input u is the commanded pitch angle. The system dynamics (1) have a time constant $\tau = 0.2$, thrust $T = 10$ and drag $\gamma = 3$. Our nominal model (2) has a time constant $\tau = 0.15$, thrust $T = 10$ and $\gamma = 0$. Both our system and model are differentially flat in the output $y = x$. We use a GP to learn $v_e(\cdot)$ in (13). The GP uses a SE kernel parametrized with $\sigma_f^2 = 225, \sigma_\eta^2 = 0.1, \mathbf{L} = \text{diag}\{57, 2, 2, 66\}$, where these hyperparameters maximize the log-likehood of the data collected during the *Offline Case*. We consider two cases. *Offline Learned Model*: We consider 500 data points collected from $10s$ of tracking $y_{ref} = 2\sin(t)$ under nominal control (i.e., no learning).



(a) 1-D Quadrotor: Offline Learned Model (b) 1-D Quadrotor: Online Learned Model (c) Pendulum: Online Learned Model

Fig. 2. Average output tracking error under *Nominal* control (no learning), *FB Correction Only* (a learning-based control that only improves the FB linearization), *Robust Only* (a learning-based control that only uses a bound to design a robust LQR) and our *Proposed* learning-based control (as in Fig. 1). This is shown on a A) 1-D quadrotor simulation when using an (a) *Offline Learned Model* and an (b) *Online Learned Model*, and B) an inverted pendulum using an (c) *Online Learned Model*.

This GP model is fixed as we use it to follow different trajectories. *Online Learned Model*: We update our GP model online based on the last 100 data points collected during the current trajectory tracking. The previously-tuned hyperparameters stay fixed. In both cases, we compare our proposed approach with three other approaches. All controllers use LQR parameters $\mathbf{Q} = \text{diag}(100.0, 0.1, 0.1)$, $\mathbf{R} = 0.1$. We use a simulation time of $10s$ for each trajectory. We consider the following reference trajectory to track $y_{ref} = At\sin(\omega t)$. We fix $A = 0.4$ and vary ω between 1 and 1.4 to obtain progressively more aggressive trajectories. We compare average output tracking error in Fig. 2(a)-(b). In the *Offline Learned Model* simulation, we demonstrate that for some trajectories ($\omega \geq 1.3$) the *FB Linearization Only* case can cause instability. In this case, our *Proposed* approach still outperforms the *Robust Only* approach with an average tracking error reduction of 40-75%. Relying on an *Online Learned Model* improves the performance of all learning-based controllers, however, our *Proposed* approach achieves an average tracking error reduction of 50-65% over *Robust Only* and 27-37% over *FB Linearization Only*.

B. Inverted Pendulum

Our simulation uses the dynamics given in Chapter 3 (p. 112) in [14]. Our nominal model considers the mass of cart M, the mass of pendulum m, and the length of pendulum pole l to be 1.5 kg, 0.05 kg and 0.8 m, respectively, while the actual systems parameters are 1.0 kg, 0.1 kg, and 0.5 m. All controllers use LQR parameters $\mathbf{Q} = \text{diag}(10.0, 10.0)$, $\mathbf{R} = 0.1$. We use a simulation time of $4.5s$ for each trajectory. We consider the following reference trajectory to track $y_{ref} = At\sin(\omega t)$. We fix $A = 0.3$ and vary ω between 2 and 10. We compare average output tracking error in Fig. 2(c). In Fig. 3, we highlight the computed tracking error bound (based on Theorem 2).

VII. CONCLUSION

We exploit both a structural property of many dynamical systems, differential flatness, and the ability of GPs to predict derivatives and quantify uncertainty. We develop a learning-based controller that achieves high-accuracy tracking by improving FB linearization. Furthermore, we probabilistically guarantee safety by using a bound on how well we linearize the system to design a robust linear controller.

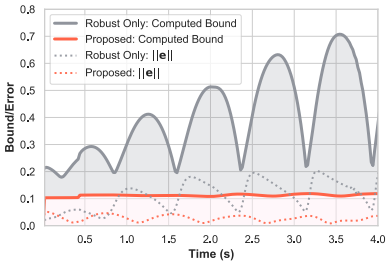


Fig. 3. Comparison of computed tracking error bound (Theorem 2) and actual tracking error for the inverted pendulum when tracking $y_{ref} = 0.3t\sin(\omega t)$, $\omega = 4.0$, for *Proposed* and *Robust Only* approaches.

REFERENCES

[1] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[2] J. Umlauft, L. Pohler, and S. Hirche, "An uncertainty-based control lyapunov approach for control-affine systems modelled by Gaussian processes," *IEEE Control Systems Letters*, 2018.

[3] C. J. Ostafew, A. P. Schoellig and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile path-tracking," *International Journal of Robotics Research*, 2016.

[4] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *Proc. European Control Conference (ECC)*, 2015.

[5] M. Fliess, J. Lévine, P. Martin and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, 1995.

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[7] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[8] T. Westenbroek, D. F. Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, C. J. Tomlin, "Feedback linearization for unknown systems via reinforcement learning," *arXiv preprint arXiv: 1910.13272 (2019)*.

[9] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche, "Feedback linearization using Gaussian processes," in *Proc. IEEE Conference on Decision and Control (CDC)*, 2017.

[10] M. K. Helwa, A. Heins, and A. P. Schoellig, "Provably robust learning-based approach for high-accuracy tracking control of Lagragian systems," *IEEE Robotics and Automation Letters*, 2019.

[11] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modelling and control*. Hoboken, NJ, USA: Wiley, 2006.

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

[13] A. Pollastri and V. Tulli, "The distribution of the absolute value of the ratio of two correlated normal random variables," *Statistica Applicazioni*, 2015.

[14] G. G. Rigatos, *Nonlinear Control and Filtering Using Differential Flatness Approaches*. Switzerland: Springer, 2015.