Provably Robust Learning-Based Approach for High-Accuracy Tracking Control of Lagrangian Systems

Mohamed K. Helwa, Adam Heins, and Angela P. Schoellig

***Abstract*—Lagrangian systems represent a wide range of robotic systems, including manipulators, wheeled and legged robots, and quadrotors. Inverse dynamics control and feedforward linearization are typically used to convert the complex nonlinear dynamics of Lagrangian systems to a set of decoupled double integrators, and then a standard, outer-loop controller can be used to calculate the commanded acceleration for the linearized system. However, these methods typically depend on having a very accurate system model, which is often not available in practice. While this challenge has been addressed in the literature using different learning approaches, most of these approaches do not provide safety guarantees in terms of stability of the learning-based control system. In this letter, we provide a novel, learning-based control approach based on Gaussian processes (GPs) that ensures both stability of the closed-loop system and high-accuracy tracking. We use GPs to approximate the error between the commanded and the actual acceleration of the system, and then use the predicted mean and variance of the GP to calculate an upper bound on the uncertainty of the linearized model. This uncertainty bound is then used in a robust, outer-loop controller to ensure stability of the overall system. Moreover, we show that the tracking error converges to a ball with a radius that can be made arbitrarily small. Finally, we verify the effectiveness of our approach via simulations on a 2 degree-of-freedom (DOF) planar manipulator and experimentally on a 6 DOF industrial manipulator.**

***Index Terms*—Model learning for control, robot safety, robust/adaptive control of robotic systems, learning and adaptive systems.**

I. Introduction

HIGH-ACCURACY tracking is an essential requirement in advanced manufacturing, self-driving cars, medical robots, and autonomous flying vehicles, among others. To achieve high-accuracy tracking for these complex, typically high-dimensional, nonlinear robotic systems, a standard approach is to use inverse dynamics control [1] or feedforward linearization techniques [2] to convert the complex nonlinear dynamics into a set of decoupled double integrators. Then, a standard, linear, outer-loop controller, e.g., a proportional-derivative (PD) controller, can be used to make the decoupled linear system track the desired trajectory [1]. However, these linearization techniques depend on having accurate system models, which are difficult to obtain in practice.

To address this problem, robust control techniques have been used for many decades to design the outer-loop controllers to account for the uncertainties in the model [3]. However, the selection of the uncertainty bounds in the robust controller design is challenging. On the one hand, selecting high bounds typically results in a conservative behavior, and hence, a large tracking error. On the other hand, relatively small uncertainty bounds may not represent the true upper bounds of the uncertainties, and consequently, stability of the overall system is not ensured. Alternatively, several approaches have been proposed for learning the inverse system dynamics from collected data where the system models are not available or not sufficiently accurate; see [4]–[7]. Combining a-priori model knowledge with learning data has also been studied in [4], [8]. However, these learning approaches typically neglect the learning regression errors in the analysis, and they do not provide a proof of stability of the learning-based control system, which is crucial for safety-critical ap-

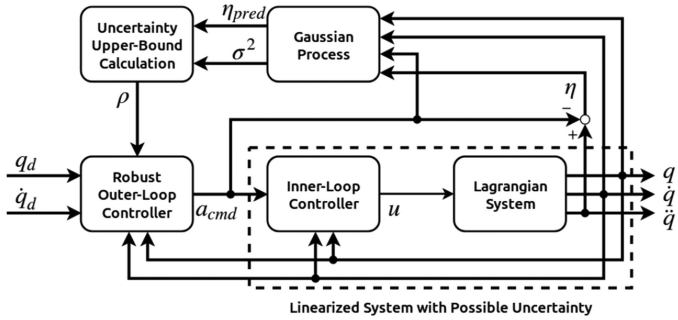

Fig. 1. Block diagram of our proposed strategy. GP regression models learn the uncertainty in the linearized model. Using the mean and variance of the GP, one can calculate an upper bound on the uncertainty to be used in a robust, outer-loop controller. The symbol q is the actual position vector, q_d is the desired position vector, a_{cmd} is the commanded acceleration vector, u is the force/torque input vector, and η is the uncertainty vector.

Manuscript received September 10, 2018; accepted January 12, 2019. Date of publication January 31, 2019; date of current version February 19, 2019. This work was supported in part by the NSERC Grant RGPIN-2014-04634, in part by the OCE/SOSCIP TalentEdge Project #27901, and in part by the Vector Institute. This letter was recommended for publication by Associate Editor M. C. Yip and Editor P. Rocco upon evaluation of the reviewers' comments. *(Corresponding author: Mohamed K. Helwa.)*

M. K. Helwa is with the Dynamic Systems Lab, Institute for Aerospace Studies, University of Toronto, Toronto, ON M3H 5T6, Canada, and also with the Department of Electrical Power and Machines, Cairo University, Giza 12613, Egypt (e-mail: mohamed.helwa@robotics.utias.utoronto.ca).

A. Heins and A. P. Schoellig are with the Dynamic Systems Lab, Institute for Aerospace Studies, University of Toronto, Toronto, ON M3H 5T6, Canada (e-mail: adam.heins@robotics.utias.utoronto.ca; schoellig@utias.utoronto.ca).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The Supplemental Materials contain a video demonstrating in experiment the six trajectories used to test the proposed robust learning approach on the UR10 6 degree-of-freedom manipulator. This material is 28.9 MB in size.

Digital Object Identifier 10.1109/LRA.2019.2896728

plications such as medical robots. The limitations of the robust control and the learning-based techniques show the urgent need for novel, robust, learning-based control approaches that ensure both stability of the control system and high-accuracy tracking. This sets the stage for the research carried out in this letter.

In this letter, we provide a novel, robust, online learning-based control technique that achieves both closed-loop stability and high-accuracy tracking. In particular, we use Gaussian processes (GPs) to approximate the error between the commanded acceleration to the linearized system and the actual acceleration of the robotic system, and then use the predicted mean and variance of the GP to calculate an upper bound on the uncertainty of the linearization. This uncertainty bound is then used in a robust, outer-loop controller to ensure stability of the overall system (see Figure 1). Moreover, we show that using our proposed strategy, the tracking error converges to a ball with a radius that can be made arbitrarily small through appropriate control design, and hence, our proposed approach also achieves high-accuracy tracking. Furthermore, we verify the effectiveness of the proposed approach via simulations on a 2 DOF planar manipulator using MATLAB Simulink and experimentally on a UR10 6 DOF industrial manipulator.

This letter is organized as follows. Section II provides a summary of recent related work. Section III describes the problem, and Section IV provides the proposed approach. Section V derives theoretical guarantees for the proposed approach. Section VI and VII provide the simulation and experimental results, and Section VIII concludes the letter.

Notation and Basic Definitions: For a set S, $\bar{S}$ denotes its closure and S° its interior. The notation $B_\delta(y)$ denotes a ball of radius δ centered at a point y. A matrix P is *positive definite* if it is symmetric and all its eigenvalues are positive. For a vector x, $\|x\|$ denotes its Euclidean norm. A function $f(x)$ is *smooth* if its partial derivatives of all orders exist and are continuous. The solutions of $\dot{x} = f(t, x)$ are *uniformly ultimately bounded with ultimate bound* b if there exist positive constants b, c, and for every $0 < a < c$, there exists $T(a, b) \geq 0$ such that $\|x(t_0)\| \leq a$ implies $\|x(t)\| \leq b$, for all $t \geq (T + t_0)$, where t_0 is the initial time instant. A *kernel* is a symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A *reproducing kernel Hilbert space (RKHS)* corresponding to a kernel $\kappa(.,.)$ includes functions of the form $f(x) = \sum_{j=1}^{m} \alpha_j \kappa(x, x_j)$ with $m \in \mathbb{N}$, $\alpha_j \in \mathbb{R}$ and representing points $x_j \in \mathcal{X}$.

II. Related Work

The study of safe learning dates back to the beginning of this century [9]. In [10] and [11], Lyapunov-based reinforcement learning is used to allow a learning agent to safely switch between pre-computed baseline controllers. Then, in [12], risk-sensitive reinforcement learning is proposed, in which the expected return is heuristically weighted with the probability of reaching an error state. In several other papers, including [13]–[15], safe exploration methods are used to allow the learning modules to achieve a desired balance between ensuring safe operation and exploring new states for improved performance. In [9], a general framework is proposed for ensuring safety of learning-based control strategies for uncertain robotic systems. In this framework, robust reachability guarantees from control theory are combined with Bayesian analysis based on empirical observations. The result is a safety-preserving, supervisory controller of the learning module that allows the system to freely execute its learning policy almost everywhere, but imposes control actions to ensure safety at critical states. Despite its effectiveness for ensuring safety, the supervisory controller in this approach has no role in reducing tracking errors.

Focusing our attention on safe, learning-based inverse dynamics control, we refer to [16]–[18]. In [16], a model reference adaptive control (MRAC) architecture based on GPs is proposed, and stability of the overall control system is proved. In contrast to [16], which uses only the mean estimate of the GPs in the control law, we use both the mean and variance of the GPs in our control law. As a result, our control law adjusts its aggressiveness based on the certainty of the GPs' predictions. While the results of [16] are shown in simulations on a two-dimensional system, we validate our algorithm experimentally on a 6 DOF manipulator.

In [17], [18], GPs are used to predict the errors in the output torques of the inverse dynamics model online. In [17], the GP learning is combined with a state-of-the-art gradient descent method for learning feedback terms online. The main idea behind this approach is that the gradient descent method would correct for fast perturbations, while the GP is responsible for correcting slow perturbations. This allows for exponential smoothing of the GP hyperparameters, which increases the robustness of the GP at the cost of having slower reactiveness. Nevertheless, [17] does not provide a proof of the robust stability of the closed-loop system. In [18], the variance of the GP prediction is utilized to adapt the parameters of an outer-loop PD controller online, and the uniform ultimate boundedness of the tracking error is proved under some assumptions on the structure of the PD controller (e.g., the gain matrix was assumed to be diagonal, which imposes a decentralized gain control scheme). The results of [18] are verified via simulations on a 2 DOF manipulator. Another interesting, similar work by the same authors is [19], which is tested experimentally on a 3 DOF manipulator.

Our approach differs from [18], [19] in several respects. First, unlike [18], we do not use an adaptive PD controller in the outer loop, but add a robustness term to the output of the outer-loop controller. Second, while [18], [19] use the GP to learn the error in the estimated torque from the nominal inverse dynamics, in our approach, we learn the error between the commanded and actual accelerations. This can be beneficial in two ways: *(i)* This makes our approach applicable to industrial manipulators that have onboard controllers for calculating the torque and only allow the user to send commanded acceleration/velocity; *(ii)* this makes it applicable beyond inverse dynamics control of manipulators; indeed, our proposed approach can be applied to any Lagrangian system for which feedforward/feedback linearization can be used to convert the nonlinear dynamics of the system to a set of decoupled double integrators, such as a quadrotor under a feedforward linearization (see Section 6 of [20]). Third, while [18] shows uniform ultimate boundedness of the tracking error, it does not provide discussions on the size of the ultimate ball. We show that using our proposed approach, the size of the ball can be made arbitrarily small through the control design. Fourth, in our approach, we do not impose any assumption on the structure of the outer-loop PD controller and decentralized outer-loop control is not needed for our proof. Finally, we verify our approach experimentally on a 6 DOF manipulator.

III. Problem Statement

We consider Lagrangian systems, which represent a wide class of mechanical systems [21]. In what follows, we focus on a class of Lagrangian systems represented by:

$$M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + g(q(t)) = u(t), \quad (1)$$

where $q = (q_1, \ldots, q_N)$ is the vector of generalized coordinates (displacements or angles), $\dot{q} = (\dot{q}_1, \ldots, \dot{q}_N)$ is the vector of generalized velocities, $u = (u_1, \ldots, u_N)$ is the vector of generalized forces (forces or torques), N is the system's degree of freedom, M, C, and g are matrices of proper dimensions and smooth functions, and $M(q)$ is a positive definite matrix. Fully-actuated robotic manipulators are an example of Lagrangian systems that can be expressed by (1). Despite focusing our discussion on systems represented by (1), we emphasize that our results in this letter can be easily generalized to a wider class of nonlinear Lagrangian systems for which feedback/feedforward linearization can be used to convert the dynamics of the system into a set of decoupled double integrators plus an uncertainty vector.

For the nonlinear system (1), we aim to make the system positions and velocities $(q(t), \dot{q}(t))$ track a desired smooth trajectory $(q_d(t), \dot{q}_d(t))$. For simplicity of notation, in our discussion, we drop the dependency on time t from q, q_d, their derivatives, and u. In this letter, we consider the scenario where the system matrices M, C, and g are not accurately known. Our goal is to design a novel, learning-based control strategy that is easy to interpret and implement, and that satisfies the following desired objectives:

- (O1) *Robustness:* The overall, closed-loop control system satisfies robust stability in the sense that the tracking error has an upper bound under the system uncertainties.
- (O2) *High-Accuracy Tracking:* For feasible desired trajectories, the tracking error converges to a ball around the origin that can be made arbitrarily small through the control design.
- (O3) *Adaptability:* The proposed strategy should incorporate online learning to continuously adapt to online changes of the system parameters and disturbances.
- (O4) *Generalizability of the Approach:* The proposed approach should be general enough to be also applicable to industrial robots that have onboard controllers for calculating the forces/torques and only allow the user to send commanded acceleration/velocity.

IV. Methodology

We present our proposed methodology, and then in the next sections, we show that it satisfies objectives (O1)–(O4).

A standard approach for solving the tracking control problem for (1) is inverse dynamics control. Since $M(q)$ is positive definite by assumption, it is invertible. Hence, it is evident that if the matrices M, C, and g are all known, then the following inverse dynamics control law

$$u = C(q, \dot{q})\dot{q} + g(q) + M(q)a_{cmd} \quad (2)$$

converts the complex nonlinear dynamic system (1) into

$$\ddot{q} = a_{cmd}, \quad (3)$$

where a_{cmd} is the commanded acceleration, a new input to the linearized system (3) to be calculated by an outer-loop control law, e.g., a PD controller (see Figure 1). However, the standard inverse dynamics control (2) heavily depends on accurate knowledge of the system parameters. In practice, the matrices M, C, and g are not perfectly known, and consequently, one has to use estimated values of these matrices $\hat{M}$, $\hat{C}$, and $\hat{g}$, respectively, where $\hat{M}$, $\hat{C}$, and $\hat{g}$ are composed of smooth functions. Hence, in practice, (2) should be replaced with

$$u = \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q) + \hat{M}(q)a_{cmd}. \quad (4)$$

Now by substituting (4) into the system model (1), we get

$$\ddot{q} = a_{cmd} + \eta(q, \dot{q}, a_{cmd}), \quad (5)$$

where $\eta(q, \dot{q}, a_{cmd}) = M^{-1}(q)(\tilde{M}(q)a_{cmd} + \tilde{C}(q, \dot{q})\dot{q} + \tilde{g}(q))$, with $\tilde{M} = \hat{M} - M$, $\tilde{C} = \hat{C} - C$, and $\tilde{g} = \hat{g} - g$. It can be shown that even if the left hand side (LHS) of (1) has a smooth, unstructured, added uncertainty $E(q, \dot{q})$, e.g., unmodeled friction, (5) is still valid with modified η. Because of η, the dynamics (5) resulting from the inverse dynamics control are still nonlinear and coupled. To control the uncertain system (5), on the one hand, robust control methods are typically very conservative, while on the other hand, learning methods do not provide stability guarantees.

Hence, in this letter, we combine ideas from robust control theory with ideas from machine learning, particularly Gaussian processes (GPs) for regression, to provide a robust, learning-based control strategy that satisfies objectives (O1)–(O4). The main idea behind our proposed approach is to use GPs to learn the uncertainty vector $\eta(q, \dot{q}, a_{cmd})$ in (5) online. Following [18], we use a set of N independent GPs, one for learning each element of η, to reduce the complexity of the regression. It is evident that conditioned on knowing q, $\dot{q}$, and a_{cmd}, one can learn each element of η independently from the rest of the elements of η. A main advantage of GP regression is that it does not only provide an estimated value of the mean μ, but also an expected variance σ^2, which represents the accuracy of the regression model based on the distance to the training data. The punchline here is that one can use both the mean and variance of the GP to calculate an upper bound ρ on $\|\eta\|$ that is guaranteed to be correct with high probability, as we will show later in this section. One can then use this upper bound to design a robust, outer-loop controller that ensures robust stability of the overall system. Hence, our proposed strategy consists of three parts:

(i) Inner-Loop Controller: We use the inverse dynamics control law (4), where $\hat{M}$, $\hat{C}$, and $\hat{g}$ are estimated values of the system matrices from an a-priori model.

(ii) GPs for Learning the Uncertainty: We use a set of N GPs to learn the uncertainty vector η in (5). We start by reviewing GP regression [15], [22]. A GP is a nonparametric regression model that is used to approximate a nonlinear function $J(x) : \mathcal{X} \to \mathbb{R}$, where $x \in \mathcal{X}$ is the input vector. The ability of the GP to approximate the function is based on the assumptions that function values $J(x)$ associated with different values of x are random variables, and that any finite number of these variables have a joint Gaussian distribution. The GP predicts the value of the function, $J(x^*)$, at an arbitrary input $x^* \in \mathcal{X}$ based on a set of n observations $\boldsymbol{D}_n := \{x_j, \hat{J}(x_j)\}_{j=1}^n$, where $\hat{J}(x_j)$, $j \in \{1, \ldots, n\}$, are assumed to be noisy measurements of the function's true values. That is, $\hat{J}(x_j) = J(x_j) + \omega'$, where ω' is a zero mean Gaussian noise with variance σ_ω^2. Assuming, without loss of generality (w.l.o.g.), a zero prior mean of the

GP and conditioned on the previous observations, the mean and variance of the GP prediction are given by:

$$\mu(x^*) = \mathbf{k}_n(x^*)(\mathbf{K}_n + \mathbf{I}_n\sigma_\omega^2)^{-1}\hat{\mathbf{J}}_n, \quad (6)$$

$$\sigma^2(x^*) = \kappa(x^*, x^*) - \mathbf{k}_n(x^*)(\mathbf{K}_n + \mathbf{I}_n\sigma_\omega^2)^{-1}\mathbf{k}_n^T(x^*), \quad (7)$$

respectively, where $\hat{\mathbf{J}}_n = [\hat{J}(x_1), \ldots, \hat{J}(x_n)]^T$ is the vector of observed, noisy function values. The matrix $\mathbf{K}_n \in \mathbb{R}^{n\times n}$ is the covariance matrix with entries $[\mathbf{K}_n]_{(i,j)} = \kappa(x_i, x_j)$, $i, j \in \{1, \ldots, n\}$, where $\kappa(x_i, x_j)$ is the covariance function defining the covariance between two function values $J(x_i)$, $J(x_j)$ (also called the kernel). The vector $\mathbf{k}_n(x^*) = [\kappa(x^*, x_1), \ldots, \kappa(x^*, x_n)]$ contains the covariances between the new input and the observed data points, and $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix. The tuning of the GP is typically done through the selection of the kernel function and the tuning of its hyperparameters. For information about different standard kernel functions, please refer to [22].

We next discuss our implementation of the GPs. The GPs run online in discrete time with sampling interval T_s. At a sampling time instant k, the inputs to each GP are the same $(q(k), \dot{q}(k), a_{cmd}(k))$, and the output is an estimated value of an element of the η vector at k. For the training data for each GP, n observations of $(q, \dot{q}, a_{cmd})$ are used as the labeled input together with n observations of an element of the vector $\ddot{q} - a_{cmd} + \omega_v$ as the labeled output, where $\omega_v \in \mathbb{R}^N$ is Gaussian noise with zero mean and variance $\text{diag}(\sigma_{\omega_1}^2, \ldots, \sigma_{\omega_N}^2)$; see (5). For selecting the n observations, we use the oldest point (OP) scheme for simplicity; this scheme depends on removing the oldest observation to accommodate for a new one [16]. We use the squared exponential kernel

$$\kappa(x_i, x_j) = \sigma_\eta^2 \exp\left(-\frac{1}{2}(x_i - x_j)^T L^{-2}(x_i - x_j)\right), \quad (8)$$

which is parameterized by the hyperparameters: σ_η^2, the prior variance, and the positive length scales $l_1, \ldots, l_{3N}$ which are the diagonal elements of the diagonal matrix L. Hence, the expected mean and variance of each GP can be obtained through equations (6)–(8). For the experiments, we tune the GP hyperparameters $\sigma_\omega^2, \sigma_\eta^2, l_1, \ldots, l_{3N}$ offline using the Python library GPy. Guidelines for tuning the GP hyperparameters can be found in [15]. We note that by fixing the GP hyperparameters offline, we impose a-priori assumptions on the function to be learned. We then learn the function online based on the last n observations [15].

As stated before, a main advantage of GP regression is that the GP provides a variance, which represents the uncertainty of the regression model based on the distance between the new input and the training data. One can then use the predicted mean and variance of the GP to provide a confidence interval around the mean that is guaranteed to be correct with high probability. There are several comprehensive studies in the machine learning literature on calculating these confidence intervals [22], [23]. We review one of these results, namely Theorem 6 of [22]. Let $x = (q, \dot{q}, a_{cmd})$, and $\eta_i(x)$ denote the i-th element of the unknown vector η.

Assumption 4.1: The function $\eta_i(x)$, $i \in \{1, \ldots, N\}$, has a bounded RKHS norm $\|\eta_i\|_\kappa$ with respect to the kernel $\kappa(x, x')$ of the GP, and the noise ω_i added to the output observations, $i \in \{1, \ldots, N\}$, is uniformly bounded by $\bar{\sigma}$.

The RKHS norm is a measure of the function smoothness, and its boundedness implies that the function is well-behaved in the sense that it is regular with respect to the kernel [22].

Lemma 4.1 (Theorem 6 of [22]): Suppose that Assumption 4.1 holds. Let $\delta_p \in (0, 1)$. Then,

$$\Pr\{\forall x \in \mathcal{X}, \|\mu(x) - \eta_i(x)\| \le \beta^{1/2}\sigma(x)\} \ge 1 - \delta_p,$$

where Pr stands for the probability, $\mathcal{X} \subset \mathbb{R}^{3N}$ is compact, $\mu(x)$, $\sigma^2(x)$ are the GP mean and variance evaluated at x conditioned on n past observations, and

$$\beta = 2\|\eta_i\|_\kappa^2 + 300\gamma \ln^3((n+1)/\delta_p).$$

The variable $\gamma \in \mathbb{R}$ is the maximum information gain and is given by $\gamma = \max_{\{x_1,\ldots,x_{n+1}\}\in\mathcal{X}} 0.5\,\log(\det(I + \bar{\sigma}^{-2}\mathbf{K}_{n+1}))$, where det is the matrix determinant, $I \in \mathbb{R}^{(n+1)\times(n+1)}$ is the identity matrix, and $\mathbf{K}_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$ is the covariance matrix given by $[\mathbf{K}_{n+1}]_{(i,j)} = \kappa(x_i, x_j), i, j \in \{1, \ldots, n+1\}$.

Finding the information gain maximizer can be approximated by an efficient greedy algorithm [22].

The punchline here is that we know from Lemma 4.1 that one can define for each GP a confidence interval around the mean that is guaranteed to be correct for all points $x \in \mathcal{X}$, a compact set, with probability higher than $(1 - \delta_p)$, where δ_p is typically picked very small. Let $\mu_{k,i}$ and $\sigma_{k,i}^2$ represent the expected mean and variance of the i-th GP at the sampling time instant k, respectively, and let β_i denote the β parameter in Lemma 4.1 of the i-th GP, where $i \in \{1, \ldots, N\}$. We select the upper bound on the absolute value of η_i at k to be

$$\rho_{k,i}(\mu_{k,i}, \sigma_{k,i}) = \max(|\mu_{k,i} - \beta_i^{1/2}\sigma_{k,i}|, |\mu_{k,i} + \beta_i^{1/2}\sigma_{k,i}|). \quad (9)$$

Then, it is evident that an upper bound on $\|\eta\|$ at k is

$$\rho_k = \sqrt{\rho_{k,1}(\mu_{k,1}, \sigma_{k,1})^2 + \cdots + \rho_{k,N}(\mu_{k,N}, \sigma_{k,N})^2}. \quad (10)$$

(iii) Robust, Outer-Loop Controller: We use the estimated upper bound ρ_k to design a robust, outer-loop controller. In particular, for a smooth, bounded desired trajectory $(q_d(t), \dot{q}_d(t))$, we use the outer-loop control law

$$a_{cmd}(t) = \ddot{q}_d(t) + K_P(q_d(t) - q(t)) + K_D(\dot{q}_d(t) - \dot{q}(t)) + r(t), \quad (11)$$

where $K_P \in \mathbb{R}^{N\times N}$ and $K_D \in \mathbb{R}^{N\times N}$ are the proportional and derivative matrices of the PD control law, respectively, and $r \in \mathbb{R}^N$ is an added vector to the PD control law that will be designed to achieve robustness. Let $e(t) := (q(t) - q_d(t), \dot{q}(t) - \dot{q}_d(t))$ denote the tracking error vector. From (11) and (5), it can be shown that the tracking error dynamics are

$$\dot{e}(t) = Ae(t) + B(r(t) + \eta(q(t), \dot{q}(t), a_{cmd}(t))), \quad (12)$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix} \in \mathbb{R}^{2N\times 2N},\ B = \begin{bmatrix} 0 \\ I \end{bmatrix} \in \mathbb{R}^{2N\times N}, \quad (13)$$

and $I \in \mathbb{R}^{N\times N}$ is the identity matrix. From (12) and (13), it is clear that the controller matrices K_P and K_D should be designed to make A a Hurwitz matrix.

We now discuss how to design the robustness vector $r(t)$. To that end, let $P \in \mathbb{R}^{2N\times 2N}$ be the unique positive definite

matrix satisfying $A^T P + PA = -Q$, where $Q \in \mathbb{R}^{2N\times 2N}$ is a positive definite matrix. We define $r(t)$ as follows

$$r(t) = \begin{cases} -\rho(t)\frac{B^T Pe(t)}{\|B^T Pe(t)\|} & \|B^T Pe(t)\| > \epsilon, \\ -\rho(t)\frac{B^T Pe(t)}{\epsilon} & \|B^T Pe(t)\| \le \epsilon, \end{cases} \tag{14}$$

where $\rho(t) \in \mathbb{R}$ is the last received upper bound on $\|\eta\|$ from the GPs, i.e., we use

$$\rho(t) = \rho_k, \forall t \in [kT_s, (k+1)T_s), \tag{15}$$

and ϵ is a small positive number. It should be noted that ϵ is a design parameter that can be selected to ensure high-accuracy tracking, as we will discuss in the next section.

V. Theoretical Guarantees

Having discussed the proposed strategy, we now justify that it achieves both robust stability and high-accuracy tracking. To that end, we make some reasonable assumptions. First, since the GPs provide an estimate of the upper bound of the uncertainty in discrete time, we must rely on the last calculated upper bound until a new value is calculated, see (15). Hence, we impose the following assumption on the sampling rate of the GPs:

Assumption 5.1: The GPs run at a sufficiently fast sampling rate such that the calculated upper bound on $\|\eta\|$, ρ_k, remains correct between two consecutive sampling time instants, i.e., $\|\eta(t)\| \le \rho_k$ for $t \in [kT_s, (k+1)T_s)$.

In practice, one should select the GP sampling interval as small as the computational resources permit. In Section VII, we successfully implement the proposed approach in real time on a 6 DOF robot. Thus, this assumption should not limit the applicability of the approach.

We impose another assumption to ensure that the added robustness vector $r(t)$ will not cause the uncertainty vector norm $\|\eta(q(t), \dot{q}(t), a_{cmd}(t))\|$ to blow up. It is easy to show that the function $\eta(q, \dot{q}, a_{cmd})$ is smooth, and so $\|\eta\|$ attains a maximum value on any compact set in its input space $(q, \dot{q}, a_{cmd})$. However, since from (11) and (14), a_{cmd} is a function of $\rho(t)$, an upper bound on $\|\eta\|$, one still needs to ensure the boundedness of $\|\eta\|$ for bounded q, $\dot{q}$ or bounded tracking error e. Hence, we present the following assumption.

Assumption 5.2: For a smooth, bounded desired trajectory $(q_d(t), \dot{q}_d(t))$, there exists $\bar{\rho} > 0$ such that $\|\eta\| \le \bar{\rho}$ for each $e \in D$, where D is a compact set containing $\{e \in \mathbb{R}^{2N} : e^T Pe \le e(0)^T Pe(0)\}$, and $e(0)$ is the initial tracking error.

Remark 5.1: Following [1], it can be shown that Assumption 5.2 is satisfied for small uncertainties in the inertia matrix $M(q)$. In particular, it can be shown that if $\rho(t)$ in (14) satisfies $\rho(t) \le \|\eta(t)\| + c$, where c is a positive scalar, and the uncertainty in the matrix $M(q)$, $\tilde{M}(q)$, is sufficiently small such that $\|M^{-1}(q)\tilde{M}(q)\| < 1$ is satisfied, then Assumption 5.2 holds (see the supplementary material for details). This argument is true even if we have large uncertainties in the other system matrices, $C(q, \dot{q})$ and $g(q)$. As indicated in Chapter 8 of [1], if the bounds on $\|M\|$ are known ($\underline{m} \le \|M\| \le \overline{m}$), then one can always select $\hat{M}$ in (4) such that $\|M^{-1}(q)\tilde{M}(q)\| < 1$ is satisfied. In particular, by selecting $\hat{M} = \frac{\overline{m}+\underline{m}}{2} I$, where I is the identity matrix, it can be shown that $\|M^{-1}(q)\tilde{M}(q)\| \le \frac{\overline{m}-\underline{m}}{\overline{m}+\underline{m}} < 1$. Consequently, it is not difficult to satisfy the condition $\|M^{-1}(q)\tilde{M}(q)\| < 1$ in practice, and Assumption 5.2 is not restrictive.

From Assumption 5.2, we know that $\|\eta(t)\| \le \bar{\rho}$ if $e(t) \in D$, and consequently, it is reasonable to saturate any estimate of $\rho(t)$ beyond $\bar{\rho}$. Hence, we suppose that the estimation of ρ is slightly modified to be

$$\rho(t) = \min(\rho_{GP}(t), \bar{\rho}), \tag{16}$$

where $\rho_{GP}(t)$ is the upper bound on the uncertainty norm $\|\eta\|$ calculated by the GPs in equations (9), (10), and (15). It is straightforward to show that with the choice of $\rho(t)$ in (16) and for bounded smooth trajectories, the condition $e(t) \in D$ for all $t \ge 0$ implies that $a_{cmd}(t)$ in (11) is always bounded, and so $x = (q, \dot{q}, a_{cmd})$ always lies in a compact set. To be able to provide theoretical guarantees, we also assume w.l.o.g. that the small positive number ϵ in (14) is selected sufficiently small such that

$$\sqrt{\frac{\epsilon\bar{\rho}}{2\lambda_{min}(Q)}} \ll \delta_1, \tag{17}$$

where $\delta_1 > 0$ is such that $B_{\delta_1}(0) \subset \{e \in \mathbb{R}^{2N} : e^T Pe < e(0)^T Pe(0)\}$, and $\lambda_{min}(Q) > 0$ is the smallest eigenvalue of the positive definite matrix Q.

Based on Assumptions 4.1, 5.1 and 5.2, we provide the following main result.

Theorem 5.1: Consider the Lagrangian system (1) and a smooth, bounded desired trajectory $(q_d(t), \dot{q}_d(t))$. Suppose that Assumptions 4.1, 5.1, and 5.2 hold, and that $\bar{\rho}$ in (16) is a correct (possibly conservative) upper bound on the uncertainty norm $\|\eta\|$. Then, the proposed, robust, learning-based control strategy in (4), (11), and (14), with the uncertainty upper bound ρ calculated by (16) and the design parameter ϵ satisfying (17), ensures with high probability of at least $(1-\delta_p)^N$ that the tracking error $e(t)$ is uniformly ultimately bounded with an ultimate bound that can be made arbitrarily small through the selection of the design parameter ϵ.

Proof: From Assumption 5.2, we know that $\|\eta(t)\| \le \bar{\rho}$ when $e(t) \in D$, where D is a compact set containing $\{e \in \mathbb{R}^{2N} : e^T Pe \le e(0)^T Pe(0)\}$. In the first part of the proof, we assume that the upper bound $\rho_{GP}(t)$ calculated by (9), (10) and (15) is a correct upper bound on $\|\eta(t)\|$ when $e(t) \in D$. Thus, in the first part of the proof, we know that $\rho(t)$ calculated by (16) is a correct upper bound on $\|\eta(t)\|$ when $e(t) \in D$, and we use Lyapunov stability analysis to prove that $e(t)$ is uniformly ultimately bounded. Then, in the second part of the proof, we use Lemma 4.1 to evaluate the probability of satisfying the assumption that $\rho_{GP}(t)$ is a correct upper bound on $\|\eta(t)\|$ when $e(t) \in D$, and hence, the probability that the provided guarantees hold.

The first part of the proof closely follows the proof of the effectiveness of the robust controller in Theorem 3 of Chapter 8 of [1], and we include the main steps here for convenience. Consider a candidate Lyapunov function $V(e) = e^T Pe$. From (12), it can be shown that $\dot{V} = -e^T Qe + 2w^T(\eta + r)$, where $w = B^T Pe$. Then, from (14), we must study two cases.

For the case where $\|w\| > \epsilon$, we have

$$w^T(\eta + r) = w^T\left(\eta - \rho\frac{w}{\|w\|}\right) = w^T\eta - \rho\|w\|$$

$$\le \|\eta\|\|w\| - \rho\|w\|$$

from the Cauchy-Schwartz inequality. Since $\{e \in \mathbb{R}^{2N} : e^T Pe \leq e(0)^T Pe(0)\} \subset D$ by definition and from Assumption 5.2, we know that $\|\eta\| \leq \bar{\rho}$. Also, by our assumption in this part of the proof, $\|\eta\| \leq \rho_{GP}$. Then, from (16), $\|\eta\| \leq \rho$, and $w^T(\eta + r) \leq 0$. Thus, for this case, $\dot{V} \leq -e^T Qe$, which ensures exponential decrease of the Lyapunov function.

Next, consider the case where $\|w\| \leq \epsilon$. If $w = 0$, then $\dot{V} = -e^T Qe < 0$. Then, for $\|w\| \leq \epsilon$ and $w \neq 0$, it is easy to show

$$\dot{V} = -e^T Qe + 2w^T(\eta + r) \leq -e^T Qe + 2w^T \left(\rho \frac{w}{\|w\|} + r\right).$$

From (14), we have

$$\dot{V} \leq -e^T Qe + 2w^T \left(\rho \frac{w}{\|w\|} - \rho \frac{w}{\epsilon}\right).$$

It can be shown that the term $2w^T(\rho \frac{w}{\|w\|} - \rho \frac{w}{\epsilon})$ has a maximum value of $(\epsilon\rho)/2$ when $\|w\| = \epsilon/2$. Thus, $\dot{V} \leq -e^T Qe + (\epsilon\rho)/2$. From (16), $\rho \leq \bar{\rho}$, and consequently $\dot{V} \leq -e^T Qe + (\epsilon\bar{\rho})/2$. If the condition $e^T Qe > (\epsilon\bar{\rho})/2$ is satisfied, then $\dot{V} < 0$. Since Q is positive definite by definition, then $e^T Qe \geq \lambda_{min}(Q)\|e\|^2$, where $\lambda_{min}(Q) > 0$ is the smallest eigenvalue of Q. Hence, if $\lambda_{min}(Q)\|e\|^2 > (\epsilon\bar{\rho})/2$, then $\dot{V} < 0$. Thus, the Lyapunov function is strictly decreasing if $\|e\| > \sqrt{\frac{\epsilon\bar{\rho}}{2\lambda_{min}(Q)}}$. Let B_δ be the ball around the origin of radius $\delta := \sqrt{\frac{\epsilon\bar{\rho}}{2\lambda_{min}(Q)}}$, S_δ be a sufficiently small sublevel set of the Lyapunov function V satisfying $\bar{B}_\delta \subset S_\delta^\circ$, and B_c be the smallest ball around the origin satisfying $S_\delta \subset \bar{B}_c$. Since the Lyapunov function V is strictly decreasing outside $\bar{B}_\delta$, the tracking error $e(t)$ eventually reaches and remains in $S_\delta \subset \bar{B}_c$, and so the tracking error $e(t)$ is uniformly ultimately bounded, and its ultimate bound is the radius of B_c. Note that from (17), $B_\delta \subset \{e \in \mathbb{R}^{2N} : e^T Pe < e(0)^T Pe(0)\} \subset D$, and ρ is a correct upper bound on $\|\eta\|$. One can see that δ and hence the radius of B_c depend on the choice of the design parameter ϵ. Indeed, ϵ can be selected sufficiently small to make B_δ and B_c arbitrarily small.

In the second part of the proof, we calculate the probability of our assumption in the first part that $\rho_{GP}(t)$ is a correct upper bound on $\|\eta(t)\|$ when $e(t) \in D$. Recall that $e(t) \in D$ implies that $x(t)$ is in a compact set, as discussed after (16). From Assumption 5.1, our problem reduces to calculating the probability that ρ_{GP} is a correct upper bound on $\|\eta\|$ for all the sampling time instants. Using the confidence region proposed in Lemma 4.1 for calculating the upper bound on the absolute value of each element of η, and under Assumption 4.1, the probability that this upper bound is correct for all samples is higher than $(1 - \delta_p)$ from Lemma 4.1. Since the N GPs are independent and the added noise to the output observations ω_v is uncorrelated, the probability that the upper bounds on the absolute values of all the elments of η, and hence the upper bound on $\|\eta(t)\|$, are correct is higher than $(1 - \delta_p)^N$. ■

Remark 5.2: Although in practice it is difficult to estimate the upper bound $\bar{\rho}$ in (16), one can be conservative in this choice. Unlike robust control techniques that keep this conservative bound unchanged, (16) relaxes the bound $\bar{\rho}$ when the GPs learn a lower upper bound from collected data. It can be shown that if $\rho(t) \leq \rho' < \bar{\rho}$ for all t, then the tracking error will converge to an ultimate ball $B_{c'}$ smaller than B_c.

Remark 5.3: In theory, ϵ can be selected sufficiently small to ensure arbitrarily accurate tracking as shown in the proof of Theorem 5.1. Achieving that for cases with large uncertainties may be limited by the actuation limits of the robots. Incorporating the actuation limits in the theoretical analysis is an interesting direction for future research.

VI. Simulation Results

The proposed approach is first verified via simulations on a 2 DOF planar manipulator using MATLAB Simulink.

We use the robot dynamics (1) for the system, where M, C, and g are as defined in Chapter 7 of [1]. For the system parameters, a value of 1 kg is used for each link mass, 2 m for each link length, and 1 kg·m^2 for each link inertia. The joints are assumed to have no mass and are not affected by friction. Then, it is assumed that these parameters are not perfectly known. Thus, in the inverse dynamics controller (4), we use parameters with different levels of uncertainty. For the PD controller parameters, we use $K_p = 2I$ and $K_d = I$ for all simulations, where I is the identity matrix. The desired trajectories are sinusoidal with different amplitudes and frequencies. All simulation runs start at zero initial conditions.

We use 2 GPs to learn η in (5). Each GP uses the squared exponential kernel parameterized with $\sigma_{\eta,i} = 1$, $\sigma_{\omega,i} = 0.001$, and $l_{j,i} = 0.5$, for all $j \in \{1, \ldots, 6\}$ and $i \in \{1, 2\}$. The GPs run at 10 Hz and use the past $n = 20$ observations for prediction. Note that for fixed n, the parameter β in Lemma 4.1 is a constant scalar, the value of which depends on the preselected, desired probability $(1 - \delta_p)$. Hence, to generate confidence intervals, we use $[\mu_k - 3\sigma_k, \mu_k + 3\sigma_k]$, which is simple to implement and found to be effective in practice [15]. For the robust controller, we use $\epsilon = 0.001$.

A sequence of *12 trajectories* is run for *3 different cases of model uncertainty*. Each case makes the $\hat{M}$ matrix differ from the M matrix by using values for the estimated link masses that differ from the true values. In particular, in the three uncertainty cases, the estimated mass differs from the actual mass by 10%, 20%, and 30% for each link.

The tracking performance is compared between four controllers: a nominal controller with no robust control, a robust controller with a fixed upper bound on the uncertainty norm $\rho = 1000$, a learning-based inverse dynamics controller in which GPs are used to learn the error of the nominal inverse model at the torque level Δu with a non-robust outer-loop controller, and our proposed robust learning controller. For the robust controller, we intentionally selected ρ quite large to emulate a possible practical scenario when it is not obvious beforehand how much uncertainty is present in the system, so a large value is a safe but conservative choice. For fairness of comparison, in our proposed approach, we also use 1000 as the initial guess of the upper bound on the uncertainty, $\bar{\rho}$ in equation (16). Unlike the fixed robust controller that keeps this conservative value unchanged, our proposed algorithm updates it when a less-conservative upper bound is learned using the GPs. The root-mean-square (RMS) error of the joint angles was averaged over the 12 trajectories. The results are in Table I. On average, our proposed controller reduces the tracking errors by 95.8% compared to the nominal controller, by 78.2% compared to the fixed, robust controller, and by 66% compared to the non-robust learning controller that learns Δu.

TABLE I
AVERAGE RMS TRACKING ERROR (IN RAD) OVER 12 TRAJECTORIES FOR DIFFERENT CONTROLLERS ON A 2 DOF MANIPULATOR

Uncertainty	Nominal	Fixed Robust	Learning Δu	**Robust Learning**
10%	0.1554	0.0476	0.0190	0.0082
20%	0.2793	0.0498	0.0319	0.0103
30%	0.3768	0.0519	0.0539	0.0141

Fig. 2. The UR10 industrial manipulator used in the experiments.

VII. EXPERIMENTAL RESULTS

The proposed approach is tested on a UR10 6 DOF manipulator (Figure 2) using the Robot Operating System (ROS).

A. Experimental Setup

The interface to the UR10 does not permit direct torque control. Instead, only position and velocity control of the joints are available, with joint velocity saturation limits of $(2.16, 2.16, 3.15, 3.2, 3.2, 3.2)$ rad/s. Thus, for our proposed approach, we implement only the GP regression models and the robust, outer-loop controller. The commanded acceleration a_{cmd} calculated by the outer-loop controller in (11) is integrated to obtain a velocity command that can be sent to the UR10. To test our approach for various uncertainties, we introduce artificial model uncertainty by adding a function $\eta(q, \dot{q}, a_{cmd})$ to our calculated acceleration command a_{cmd}.

The PD gains of the outer-loop controller are tuned to achieve good tracking performance on a baseline desired trajectory in a nominal scenario with no added uncertainty. A desired trajectory of $q_d = 0.25(1 - \cos(2.0t))$ for each joint is used for this purpose, with gains selected to produce a total joint angle RMS error less than 0.01 rad. This resulted in $K_p = 7I$ and $K_d = I$, where I is the identity matrix.

We use 6 GPs to learn the uncertainty vector η, each of which uses the squared exponential kernel. The prior variance and length scale hyperparameters are optimized by maximizing the marginal likelihood function, while each noise variance is set to 0.001. Hyperparameter optimization is performed offline on data collected from sinusoidal trajectories. We test the proposed method using two sets of hyperparameters, trained in two different ways: *(i) basic training*, where the optimization is done using approximately 1,000 data points under linear uncertainty $\eta = 0.5\dot{q}$, and *(ii) improved training*, where the optimization is done using 5,000 data points under nonlinear uncertainty $\eta = 0.25(q - \dot{q} + \sin(q) - \sin(\dot{q}))$, with sin applied element-wise. The GPs are implemented and tuned using the Python library GPy. For prediction, each GP uses the past $n = 50$ observations collected at 10 Hz. For the confidence intervals, we use $[\mu_k - 3\sigma_k, \mu_k + 3\sigma_k]$ for simplicity [15]. For the robust learning controller, we use $\epsilon = 0.1$ and $\bar{\rho} = 8$.

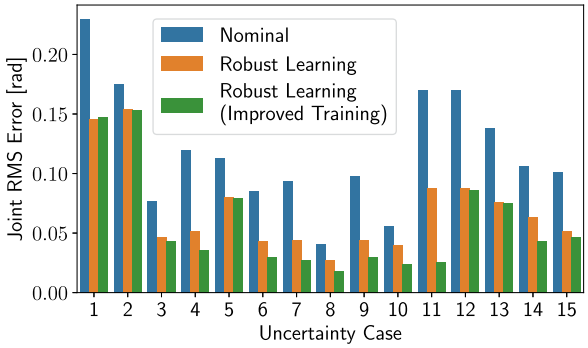

Fig. 3. Joint RMS error of the nominal controller and the proposed robust learning controller for 15 different uncertainties and a desired trajectory $q_d = 0.25(1 - \cos(2.0t))$ for each joint. Compared to the nominal controller, the proposed method reduces the average error by 41.7% (basic training) and 53.7% (improved training).

B. Results

The performance of the proposed robust learning controller is initially compared to that of the nominal, outer-loop PD controller using a single trajectory and various cases of model uncertainty. Fifteen different cases of uncertainty of the form $\eta(q, \dot{q}, a_{cmd})$ are tested over the desired trajectory $q_d = 0.25(1 - \cos(2.0t))$ for each joint, with the results shown in Figure 3. The average RMS errors for the nominal controller, the proposed controller with basic training, and the proposed controller with improved training are 0.118 rad, 0.070 rad, and 0.058 rad, respectively. For a standard robust controller with fixed ρ, it is necessary to use trial-and-error to find a suitable value for ρ. We test its performance for Uncertainty #13 in Figure 3, $\eta_i = 0.5\sin(q_2 + q_3)$ for all $i \in \{1, \ldots, 6\}$, and the same desired trajectory q_d. If we select $\rho \geq 5$, the robot enters a protective stop mode. Instead, if we select $\rho = 0.5$, the robot does not enter a protective stop mode but it has a RMS error of 0.093 rad, which is worse than the proposed approach which has a RMS error of 0.075 rad for this case. One can verify that for this combination of uncertainty and desired trajectory $\|\eta\| \leq 1.23$ for the entire run, and $\rho = 0.5$ is an incorrect upper bound of the uncertainty. One can find fixed values of ρ for which the performance is similar to the proposed approach for this particular case, such as $\rho = 1.5$, which yields a RMS error of 0.075 rad. However, this particular value of ρ does not necessarily perform well for other combinations of uncertainty and trajectory. For example, if a fixed value of $\rho = 1.5$ is used for Uncertainty #1 in Figure 3, $\eta_i = 0.5\dot{q}_i + 0.5$ for all $i \in \{1, \ldots, 6\}$, the RMS error is 0.156 rad. This is worse than the proposed approach, which has a RMS error of 0.147 rad for this case. Thus, one can use trial-and-error to find a value for ρ that is safe and gives similar performance to the proposed method for a particular run. However, this trial-and-error process must be repeated for different desired trajectories, initial conditions, and/or uncertainties, which is neither practical nor safe.

Further experiments were performed to verify the generalizability of the proposed approach for different desired trajectories. A single uncertainty case, $\eta = 0.3\dot{q} + 0.01q \odot \dot{q}$,

TABLE II
RMS TRACKING ERROR (IN RAD) FOR SIX TRAJECTORIES WITH UNCERTAINTY $\eta = 0.3\dot{q} + 0.01q \odot \dot{q}$, WHERE $\odot$ IS THE ELEMENTWISE PRODUCT

Trajectory	Nominal	Robust Learning	Robust Learning (Improved Training)
1	0.085	0.043	0.030
2	0.058	0.037	0.033
3	0.070	0.037	0.027
4	0.095	0.058	0.043
5	0.029	0.021	0.015
6	0.050	0.029	0.022
Average	0.065	0.038	0.028

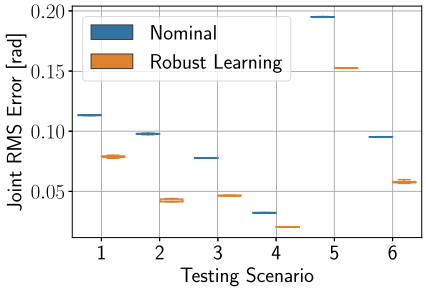

Fig. 4. Box plots of six testing scenarios swith different combinations of uncertainty and trajectory, each repeated five times. The bottom and top edges of each box show the 25th and 75th percentiles, respectively. The whiskers show the extreme values. It is clear that the performance is highly repeatable.

where $\odot$ is the elementwise product, is selected and the performance of the proposed and nominal controllers under this uncertainty is compared on five additional trajectories. The results are in Table II. The six trajectories are shown in a video at http://tiny.cc/man-traj. The proposed method provides an average overall improvement of 41.7% with basic training and 53.7% with improved training compared to the nominal controller. Figure 4 shows the high repeatability of this improvement.

VIII. CONCLUSIONS

We have provided a novel, learning-based control strategy based on GPs that ensures stability of the closed-loop system and high-accuracy tracking of smooth trajectories for an important class of Lagrangian systems. The main idea is to use GPs to estimate an upper bound on the uncertainty of the linearized model, and then use the uncertainty bound in a robust, outer-loop controller. Unlike most of the existing learning-based inverse dynamics control techniques, we have provided a proof of the closed-loop stability of the system that takes into consideration the regression errors of the learning module. Moreover, we have proved that the tracking error converges to a ball with a radius that can be made arbitrarily small. Furthermore, we have verified the effectiveness of our approach via simulations on a planar manipulator and experimentally on a 6 DOF industrial manipulator.

REFERENCES

[1] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006.
[2] V. Hagenmeyer, and E. Delaleau, "Exact feedforward linearization based on differential flatness," *Int. J. Control*, vol. 76, no. 6, pp. 537–556, 2003.
[3] C. Abdallah, D. M. Dawson, P. Dorato, and M. Jamshidi, "Survey of robust control for rigid robots," *IEEE Control Syst. Mag.*, vol. 11, no. 2, pp. 24–30, Feb. 1991.
[4] J. Sun de la Cruz, "Learning inverse dynamics for robot manipulator control," M.A.Sc. thesis, Dept. Elect. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2011.
[5] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters, "Learning inverse dynamics models with contacts," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 3186–3191.
[6] A. S. Polydoros, L. Nalpantidis, and V. Kruger, "Real-time deep learning of robotic manipulator inverse dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3442–3448.
[7] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 5201–5207.
[8] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2677–2682.
[9] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," 2017, arXiv:1705.01292.
[10] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 803–832, 2003.
[11] J. W. Roberts, I. R. Manchester, and R. Tedrake, "Feedback controller parameterizations for reinforcement learning," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn.*, 2011, pp. 310–317.
[12] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, 2005.
[13] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite Markov decision processes with Gaussian processes," *Adv. Neural Inf. Process. Syst.*, 2016, pp. 4312–4320.
[14] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes," in *Proc. IEEE Conf. Decis. Control*, 2016, pp. 4661–4666.
[15] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 491–496.
[16] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using Gaussian processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 537–550, Mar. 2015.
[17] F. Meier, D. Kappler, N. Ratliff, and S. Schaal, "Towards robust online inverse dynamics learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4034–4039.
[18] T. Beckers, J. Umlauft, D. Kulić, and S. Hirche, "Stable Gaussian process based tracking control of Lagrangian systems," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 5580–5585.
[19] T. Beckers, J. Umlauft, and S. Hirche, "Stable model-based control with Gaussian process regression for robot manipulators," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3877–3884, 2017.
[20] M. K. Helwa and A. P. Schoellig, "On the construction of safe controllable regions for affine systems with applications to robotics," *Automatica*, vol. 98, pp. 323–330, 2018.
[21] R. M. Murray, "Nonlinear control of mechanical systems: A Lagrangian perspective," *Annu. Rev. Control*, vol. 21, pp. 31–42, 1997.
[22] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 3250–3265, May 2012.
[23] S. R. Chowdhury, and A. Gopalan, "On kernelized multi-armed bandits," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 844–853.