Non-Parametric Neuro-Adaptive Control

Christos K. Verginis[1] and Zhe Xu[2] and Ufuk Topcu[3]

***Abstract*— We develop a learning-based algorithm for the control of autonomous systems governed by unknown, nonlinear dynamics to satisfy user-specified tasks expressed via time-varying reference trajectories. Most existing algorithms either assume certain parametric forms for the unknown dynamic terms or resort to unnecessarily large control inputs in order to provide theoretical guarantees. The proposed algorithm addresses these drawbacks by integrating neural-network-based learning with adaptive control. More specifically, the algorithm learns a controller, represented as a neural network, using training data that correspond to a collection of system parameters and tasks. These parameters and tasks are derived by varying the nominal parameters and the reference trajectories, respectively. It then incorporates this neural network into an online closed-form adaptive control law in such a way that the resulting behavior satisfies the user-defined task. The proposed algorithm does not use any a priori information on the unknown dynamic terms or any approximation schemes. We provide formal theoretical guarantees on the satisfaction of the task. Numerical experiments on a robotic manipulator and a unicycle robot demonstrate the effectiveness of the proposed algorithm with respect to algorithms that do not employ the neural-network controller.**

I. INTRODUCTION

Learning and control of autonomous systems with uncertain dynamics is a critical and challenging topic that has been widely studied during the last decades. One can identify plenty of motivating reasons, ranging from uncertain geometrical or dynamical parameters and unknown exogenous disturbances to abrupt faults that significantly modify the dynamics. There has been, therefore, an increasing need for developing control algorithms that do not rely on the underlying system dynamics. At the same time, such algorithms can be easily implemented on heterogeneous systems since one does not need to be occupied with the tedious computation of the dynamic terms.

A promising step towards the control of systems with uncertain dynamics is the use of data obtained a priori from system runs. However, engineering systems often undergo purposeful modifications (e.g., substitution of a motor or link in a robotic arm or exposure to new working environments) or suffer gradual faults (e.g., mechanical degradation), which might change the systems' dynamics or operating conditions. Therefore, one cannot rely on the aforementioned data to provably guarantee the successful control of the system. On the other hand, the exact incorporation of these changes in the dynamic model, and consequently, the design of new model-based algorithms, can be a challenging and often impossible procedure. Hence, the goal in such cases is to exploit the data obtained a priori and construct intelligent online policies that achieve a user-defined task while adapting to the aforementioned changes.

There has been a large variety of works that tackle the problem of control of autonomous systems with uncertain dynamics, exhibiting, however, certain limitations. Neuro-adaptive control works draw motivation from the neural network density property (see, e.g., [1])[1] and assume sufficiently small approximation errors and linear parametrizations of the unknown terms. Similarly, more traditional adaptive control methodologies assume either linear parametrizations of the unknown dynamic terms [8]–[17], use known upper-bound functions [16], or provide local stability results dictated by the dynamic bounds [17], [18]. Other learning-based related works include modeling with Gaussian processes [19], [20], using, however, partial information on the underlying system dynamics. Control of unknown nonlinear continuous-time systems has been also tackled in the literature by using funnel control, without necessarily using off-line data or dynamic approximations [21]–[23]. Nevertheless, funnel controllers usually depend on reciprocal time-varying barrier functions that drive the control input to infinity when the error approaches a pre-specified funnel, creating thus unnecessarily large control inputs that might damage the system actuators.

This paper addresses the control of systems with continuous, *unknown* nonlinear dynamics subject to tasks expressed via time-varying reference trajectories. Our main contribution lies in the development of a learning-based control algorithm that guarantees the accomplishment of a given task using only mild assumptions on the system dynamics. The algorithm draws a novel connection between adaptive control and learning with neural network representations, and consists of the following steps. Firstly, it trains a neural network that aims to learn a controller that accomplishes a given task from data obtained off-line. Secondly, we develop an online adaptive feedback control law that uses the trained network to guarantee convergence to the given reference trajectory and hence satisfaction of the task. Essentially, our approach builds on a combination of off-line trained controllers and on-line adaptations, which was recently shown to significantly enhance performance with respect to single use of the off-line part [24].

The rest of the paper is organized as follows. Section

[1]Christos K. Verginis is with the Division of Signals and Systems, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden christos.verginis@angstrom.uu.se

[2]Zhe Xu is with the Department of Electrical Engineering, Arizona State University, Tempe, Arizona, USA xzhe1@asu.edu

[3]Ufuk Topcu is with the Department of Mechanical Engineering, University of Texas at Austin, Austin, Texas, USA utopcu@utexas.edu

[1]A sufficiently large neural network can approximate a continuous function arbitrarily well in a compact set [2]–[7].

II formulates the considered problem. Section III provides the details of the proposed control algorithm. Section IV demonstrates the effectiveness of the proposed algorithm using numerical experiments and Section V concludes the paper.

II. Problem Formulation

Consider a dynamical system governed by the 2nd-order continuous-time 2nd-order dynamics

$$\dot{x}_1 = x_2 \tag{1a}$$
$$\dot{x}_2 = f(x,t) + g(x,t)u, \tag{1b}$$

where $x := [x_1^\top, x_2^\top]^\top \in \mathbb{R}^{2n}$, $n \in \mathbb{N}$, is the system state, assumed available for measurement, and u is the control input. The terms $f(\cdot)$ and $g(\cdot)$ are nonlinear vector fields that are locally Lipschitz in x over $\mathbb{R}^{2n}$ for each fixed $t \geq 0$, and continuous and uniformly bounded in t over $\mathbb{R}_{\geq 0}$ for each fixed $x \in \mathbb{R}^{2n}$. The dynamics (1) comprise a large class of nonlinear dynamical systems that capture contemporary engineering problems in mechanical, electromechanical and power electronics applications, such as rigid/flexible robots, induction motors and DC-to-DC converters, to name a few. The continuity in time and state provides a direct link to the actual underlying system, and we further do not require any time or state discretizations. We consider that $f(\cdot)$ and $g(\cdot)$ are completely unknown; we do not assume any knowledge of the structure, Lipschitz constants, or bounds, and we do not use any scheme to approximate them. Note also that we do not assume *global* Lipschitz continuity or global boundedness of $f(\cdot,t)$ and $g(\cdot,t)$ or the solution $x(t)$ of (1). Nevertheless, we do assume that $g(x,t)$ is positive definite:

Assumption 1: The matrix $g(x,t)$ is positive definite, for all $(x,t) \in \mathbb{R}^{2n} \times \mathbb{R}_{\geq 0}$.

Such assumption is a sufficiently controllability condition for (1); intuitively, it states that the multiplier of u (the input matrix $g(\cdot)$) does not change the direction imposed to the system by the underlying control algorithm. Systems not covered by (1) or Assumption 1 consist of underactuated or non-holonomic systems, such as unicycle robots or underactuated aerial vehicles. Nevertheless, we provide an extension of our results for a non-holonomic unicycle vehicle in Section III-B. Moreover, the 2nd-order model (1) can be easily extended to account for higher-order integrator systems [25].

Consider now a time-constrained task expressed as a time-varying reference trajectory $p_\text{d} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. The objective of this paper is to construct a time-varying feedback-control algorithm $u(x,t)$ such that the state of the closed-loop system (1) asymptotically tracks p_d, i.e., $\lim_{t\to\infty}(x_1(t) - p_\text{d}(t)) = 0$. Before proceeding, we formally define the errors

$$e_1 := x_1 - p_\text{d} \tag{2a}$$
$$e_2 := \dot{e}_1 + k_1 e_1 \tag{2b}$$

for a positive constant k_1, which we aim to drive to zero via the control design. We further define $e := [e_1^\top, e_2^\top]^\top \in \mathbb{R}^{2n}$ that will be used in the sequel.

III. Main Results

This section describes the proposed algorithm, which consists of two steps. Firstly, it learns a controller, represented as a neural network, using training data that correspond to a collection of different tasks and system parameters. Secondly, we design an adaptive, time-varying feedback controller that uses the neural-network approximation and guarantees tracking of the reference trajectory, consequently achieving satisfaction of the task.

A. Neural-network learning

As discussed in Section I, we are inspired by cases where systems undergo changes that modify their dynamics and hence the underlying controllers no longer guarantee the satisfaction of a specific task. In such cases, instead of carrying out the challenging and tedious procedure of identification of the new dynamic models and design of new model-based controllers, we aim to exploit data from off-line system trajectories and develop an online control law that is able to adapt to the aforementioned changes and achieve the task expressed via p_d. Consequently, we assume the existence of offline data from a finite set of T system trajectories that satisfy a collection of tasks, corresponding to bounded reference trajectories, including p_d, and possibly produced by systems with different dynamic parameters. The data from each trajectory $i \in \{1,\dots,T\}$ comprise a finite set of triplets $\{x_s(t), t, u_s(t)\}_{t\in\mathcal{T}_i}$, where $\mathcal{T}_i$ is a finite set of time instants, $x_s(t) \in \mathbb{R}^{2n}$ are system states, and $u_s(t) \in \mathbb{R}^n$ are the respective control inputs, compliant with the dynamics (1). Further, we assume that the aforementioned data triplets are bounded in a compact subset of $\mathbb{R}^{2n} \times \mathbb{R}_{\geq 0} \times \mathbb{R}^n$. We use the data to train a neural network in order to approximate the respective controller $u(x,t)$. More specifically, we use the pairs $(x_s(t), t)_{t\in\mathcal{T}_i}$ as input to a neural network, and $u_s(t)_{t\in\mathcal{T}_i}$ as the respective output targets, for all trajectories $i \in \{1,\dots,T\}$. For given $x \in \mathbb{R}^{2n}, t \in \mathbb{R}_{\geq 0}$, we denote by $u_\text{nn}(x,t)$ the output of the neural network. Note that the controller $u(x,t)$, which the neural network aims to approximate, is not associated to the specific task expressed via p_d and mentioned in Section II, but a collection of several tasks. Therefore, we do not expect the neural network to learn how to track p_d, but rather to be able to adapt to the entire collection of tasks. This is an important attribute of the proposed scheme, since it can generalize over tasks. The motivation for training the neural network with different tasks and dynamic parameters is the following. Since the tasks correspond to bounded trajectories, the respective stabilizing controllers compensate successfully the dynamics in (1). Therefore, the neural network aims to approximate an "average" controller the retains this property, i.e., the boundedness of the dynamics of (1). By using such approximation, the online feedback control law - illustrated in the next section - is able to guarantee tracking of p_d without using any explicit information on the dynamics. We explicitly model the aforementioned approximation via the following assumption on the closed-loop system trajectory that is driven by neural network's output, where we slightly abuse the notation and use (2) to express x as a function of e and t.

Assumption 2: There exists $r > 0$ such that, for all $\|x\| \leq r$, $t \geq 0$, the output $u_{\text{nn}}(x,t)$ of the neural network satisfies

$$e_2^\top \left(f(x(e,t),t) + g(x(e,t),t)u_{\text{nn}}(x(e,t),t) - \ddot{p}_{\text{d}}(t) \right) \leq \kappa \|e_2\|^2 \quad (3)$$

where κ is an unknown positive constant.

Assumption 2 is a sufficient condition for the prevention of finite-time escape of the the error trajectory $e(t)$ when the system applies only the neural-network controller, i.e., of the solution of the differential equation $\dot{e}_2 = f(x,t) + g(x,t)u_{nn}(x,t) - \dot{p}_{\text{d}} - k_1^2 e_1 + k_1 e_2$. Indeed, when the system is driven solely by the neural-network controller and satisfies (3), one can find a Lyapunov function $V(e) = e^\top G e$, for a suitable constant matrix $G \in \mathbb{R}^{2n \times 2n}$, satisfying[2] $\lambda_{\min}(G)\|e\|^2 \leq V(e) \leq \lambda_{\max}(G)\|e\|^2$ and $\dot{V} \leq \alpha \|e\|^2 \leq \frac{\alpha}{\lambda_{\min}\{G\}} V$, for all $\|x\| \leq r$ and a positive constant α. Therefore, we conclude that $\lambda_{\min}(G)\|e(t)\|^2 \leq V(e(t)) \leq V(e(0)) \exp\left(\frac{\alpha}{\lambda_{\min}\{G\}} t \right)$, which prevents any finite-time escape of $e(t)$. Further note that the constants r and κ in (3) are *unknown*.

Assumption 2 is motivated by (i) the property of neural networks to approximate a continuous function arbitrarily well in a compact domain for a large enough number of neurons and layers [1], and (ii) the fact that the neural network is trained with bounded trajectories. As mentioned before, the collection of tasks that the neural network is trained with correspond to bounded trajectories. Hence, in view of the similarity of the dynamic terms, the neural network is expected to approximate a control algorithm that maintains the boundedness of the state trajectory as per (3). Contrary to the related works (e.g., [2], [3], [5]–[7], [26]) however, we do not adopt approximation schemes for the system dynamics. In fact, a standard assumption in the related literature is the approximation of an unknown function by a single-layer neural network as $\Theta(x)\vartheta + \epsilon$, where $\Theta(x)$ is a *known* matrix of radial basis function, ϑ is a vector of unknown constants, and ϵ is a constant error assumed sufficiently small. Nevertheless, Assumption 2 is a less strict assumption; it does not require sufficiently good neural-network approximation through a sufficiently small error ϵ or knowledge of any radial-basis term $\Theta(x)$. Moreover, Assumption 2 does not imply that the neural-network output $u_{\text{nn}}(x,t)$ guarantee accomplishment of the formation task. It is merely a growth condition on the the solution of the system driven by $u_{\text{nn}}(x,t)$. In practice, (3) can be achieved by rich exploration of the state space in the tasks used for training. In the numerical experiments of Section IV, we show that (3) holds true along the executed system trajectories. Finally, we note that the neural-network controller u_{nn} can be replaced by other learning methodologies, as long as Assumption 2 holds. Nevertheless, the rich structure of neural networks makes them great candidates for approximating a control algorithm that satisfies (3).

[2] $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum eigenvalues, respectively.

B. Feedback control design

We are now ready to design the feedback control law. We first define an adaptation variable $\hat{\ell}$, with $\hat{\ell}(0) > 0$, and design the neural-network-based adaptive control law as

$$u(x, \hat{\ell}, t) = u_{\text{nn}}(x,t) - (k_2 + \hat{\ell})e_2 \quad (4a)$$

$$\dot{\hat{\ell}} = k_\ell \|e_2\|^2 \quad (4b)$$

where k_2 and k_ℓ are positive constants.

The control design is inspired by adaptive control methodologies [8], where the time-varying gain $\hat{\ell}(t)$ adapts, in coordination with the neural-network controller, to the unknown dynamics in order to ensure closed-loop stability. In particular, by inspecting the proof of Theorem 1, it can be concluded that $\hat{\ell}$ aims to counteract the term $\frac{\kappa + k_1}{\lambda_{\min}(g)}$. Intuitively, $\hat{\ell}$ increases according to (4b) until it dominates the aforementioned term, leading to convergence of e_2 to zero. Note further that the control algorithm (4) does not use any information on the system dynamics $f(\cdot)$, $g(\cdot)$ or the constants κ, r of Assumption 2. The tracking of p_{d} is guaranteed by the following theorem.

Theorem 1: Let a system evolve according to (1) and let a reference trajectory $p_{\text{d}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ encoding a user-defined task. Under Assumptions 1 and 2, there exists a set $\mathbb{M}_x$ such that, if $[x(0)^\top, \hat{\ell}(0)]^\top \in \mathbb{M}_x$, the control algorithm (4) guarantees $\lim_{t\to\infty} e(t) = 0$, as well as the boundedness of all closed-loop signals.

Proof: Let the constants $\beta := \frac{1}{\lambda_{\min}(g(x,t))}$, $\ell := \beta(k_1 + \kappa)$, where $\lambda_{\min}(\cdot)$ is the minimum eigenvalue operator, k_1 is the constant defined in (2b), and κ is the constant in the right-hand-side of (3). Note that $\lambda_{\min}(g(x,t)) > 0$ due to Assumption 1. Consider now the candidate Lyapunov function

$$V = \frac{\alpha}{2}\|e_1\|^2 + \frac{\beta}{2}\|e_2\|^2 + \frac{1}{2k_\ell}\widetilde{\ell}^2$$

where $\alpha := k_1^2 \beta$ and $\widetilde{\ell} := \hat{\ell} - \ell$. Differentiation of V and use of $\dot{e}_1 = e_2 - k_1 e_1$ from (2b) yields

$$\dot{V} = -\alpha k_1 \|e_1\|^2 + \beta e_2^\top (f(x,t) + g(x,t)u - \ddot{p}_{\text{d}} + k_1 e_2) + \frac{1}{k_\ell}\widetilde{\ell}\dot{\hat{\ell}}$$

Substitution of (4) yields

$$\dot{V} = -\alpha k_1 \|e_1\|^2 - \beta(k_2 + \hat{\ell})e_2^\top g(x,t) e_2 + \beta k_1 \|e_2\|^2 + \beta e_2^\top \left(g(x,t)u_{\text{nn}}(x,t) + f(x,t) - \ddot{p}_{\text{d}} \right) + \frac{1}{k_\ell}\widetilde{\ell}\|e_2\|^2$$

Next, note that $\hat{\ell}(t) > 0$ due to (4b) and the fact that $\hat{\ell}(0) > 0$. Therefore, by further using (3) and $\beta = \frac{1}{\lambda_{\min}(g(x,t))}$, $\dot{V}$ becomes

$$\dot{V} \leq -\alpha k_1 \|e_1\|^2 - (k_2 + \hat{\ell})\|e_2\|^2 + \beta(\kappa + k_1)\|e_2\|^2 + \frac{1}{k_\ell}\widetilde{\ell}\|e_2\|^2$$

Finally, by using $\ell = \beta(k_1 + \kappa)$ and $\widetilde{\ell} = \hat{\ell} - \ell$, we obtain $\dot{V} \leq -\alpha k_1 \|e_1\|^2 - k_2\|e_2\|^2$, which implies that $V(t) \leq V(0)$, for all $t \geq 0$ and hence the boundedness of $e_1(t)$, $e_2(t)$, and $\widetilde{\ell}(t)$. Consequently and since $p_\text{d}(t)$ and $\dot{p}_\text{d}(t)$ are bounded, we conclude the boundedness of $\hat{\ell}(t)$ and $x(t)$ for all $t \geq 0$. By differentiating $\dot{V}$ and using the boundedness of $x(t)$, $\dot{p}_\text{d}(t)$, $\ddot{p}_\text{d}(t)$, and the continuity of $f(\cdot)$ and $g(\cdot)$, we conclude the boundedness of $\ddot{V}(t)$. Therefore, $\dot{V}$ is uniformly continuous and hence, application of Barbalat's Lemma (Theorem 8.4 of [27]) yields $\lim_{t\to\infty} \dot{V}(t) = 0$, which implies that $\lim_{t\to\infty} e_1(t) = \lim_{t\to\infty} e_2(t) = 0$.

Under Assumption 2, the aforementioned results hold under the condition that $\|x\| \leq r$. Therefore, we need to establish that the proposed control algorithm and initial conditions do not force $\|x(t)\|$ to grow larger than r at any point in time $t \geq 0$. To that end, let first compact sets $\Omega_\ell \subset \mathbb{R}_{\geq 0}$, $\Omega_{1,\text{d}} \subset \mathbb{R}^n$, $\Omega_{2,\text{d}} \subset \mathbb{R}^n$ satisfying $\widetilde{\ell}(t) \in \Omega_\ell$, $p_\text{d}(t) \in \Omega_{1,\text{d}}$, and $\dot{p}_\text{d}(t) \in \Omega_{2,\text{d}}$, for all $t \geq 0$. Additionally, let $\Omega_e := \{e = [e_1^\top, e_2^\top]^\top \in \mathbb{R}^{2n} : \|x\| \leq r, p_\text{d}(t) \in \Omega_{1,\text{d}}, \dot{p}_\text{d}(t) \in \Omega_{2,\text{d}}\}$ as well as $\mathbb{M} := \{[e^\top, \widetilde{\ell}]^\top \in \mathbb{R}^{2n+1} : V \leq V_0\}$, where V_0 is the largest positive constant for which $\mathbb{M} \subseteq \Omega_e \times \Omega_\ell$. Then, for all $[e(0)^\top, \widetilde{\ell}(0)]^\top \in \mathbb{M} \Leftrightarrow [x(0)^\top, \hat{\ell}(0)]^\top \in \mathbb{M}_x$, for some $\mathbb{M}_x$, it follows that $V(t)$ is bounded from above by V_0 for all $t \geq 0$, which implies that $e(t) \in \Omega_e$, for all $t \geq 0$. Hence, it holds that $\|x(t)\| \leq r$ for all $t \geq 0$, leading to the conclusion of the proof. ∎

Remark 1: Note that, contrary to works in the related literature (e.g., [22]), we do not impose reciprocal terms in the control input that grow unbounded in order to guarantee closed-loop stability. The resulting controller is essentially a simple linear feedback on $e(t)$ with time-varying adaptive control gains, accompanied by the neural network output that ensures the growth condition (3). The positive gains k_1, k_2, and k_ℓ do not affect the stability results of Theorem 1, but might affect the evolution of the closed-loop system; e.g., larger gains lead to faster convergence but possibly larger control inputs. Further, the proposed control algorithm does not require any of the long-standing assumptions on the system dynamics (1), such as linear parameterization, growth conditions, or boundedness by known functions (e.g., [8]–[14], [16], [17], [17], [18]). Additionally, we do not assume the boundedness of the solution of (1) or of the dynamic terms $f(\cdot, t)$, $g(\cdot, t)$; instead, the control algorithm guarantees via Theorem 1 the boundedness of the system state as well as the asymptotic tracking of $p_\text{d}(t)$. The only boundedness condition that we require is (3) in Assumption 2, which can be accomplished by neural-network component in view of the universal approximation property [1].

Extension to unicycle dynamics: As mentioned in Section I, the dynamics 1 do not represent all kinds of systems, with one particular example being when non-holonomic constraints are present. In such cases, the control law design (4) can no longer guarantee the stability results of Theorem 1. In this section, we extend the control algorithm to account for unicycle vehicles subject to first-order non-holonomic

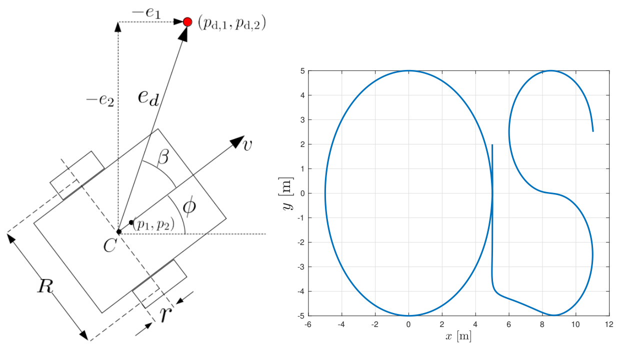

Fig. 1: Left: A unicycle vehicle. Right: Unicycle's reference trajectory for the experimental results.

constraints. More specifically, we consider the dynamics

$$\dot{p}_1 = v \cos\phi, \tag{5a}$$
$$\dot{p}_2 = v \sin\phi, \tag{5b}$$
$$\dot{\phi} = \omega \tag{5c}$$
$$M\ddot{\theta} = u + f_\theta(x, t) \tag{5d}$$

where $x_1 = [p_1, p_2, \phi]^\top \in \mathbb{R}^3$ are the unicycle's position and orientation, $x_2 = [v, \omega]^\top$ are its linear and angular velocity (see left of Fig. 1), $\theta := [\theta_R, \theta_L]^\top \in \mathbb{R}^2$ are its wheel's angular positions, and $u = [u_R, u_L]^\top \in \mathbb{R}^2$ are the wheel's torques, representing the control input. The unicycle vehicle is subject to the non-holonomic constraint $\dot{p}_1 \sin\phi - \dot{p}_2 \cos\phi = 0$, which implies that the vehicle cannot move laterally. Additionally, M is the vehicle's inertia matrix and $f_\theta(x,t) = [f_{\theta,R}(x,t), f_{\theta,L}(x,t)]^\top$ is a function representing friction and external disturbances. The velocities satisfy the relations $v = \frac{r_w}{2}(\dot{\theta}_R + \dot{\theta}_L)$, $\omega = \frac{r_w}{2R}(\dot{\theta}_R - \dot{\theta}_L)$, where r_w and R are the wheels' radius and axle length, respectively. The terms r_w, R, M, and $f_\theta(\cdot)$ are considered to be *completely unknown*. As before, the goal is for the vehicle's position $p := [p_1, p_2]^\top$ to track a desired trajectory $p_\text{d} = [p_{\text{d},1}, p_{\text{d},2}]^\top \in \mathbb{R}^2$. Towards that end, we define the error variables $e_1 := p_1 - p_{\text{d},1}$, $e_2 := p_2 - p_{\text{d},2}$, $e_d := \|p - p_\text{d}\|$, as well as the angle β measured from the the longitudinal axis of the vehicle, i.e., the unicycle's direction vector $[\cos\phi, \sin\phi]$, to the error vector $-[e_1, e_2]$ (see left of Fig. 1). The angle β can be derived by using the cross product between the aforementioned vectors, i.e., $e_d \sin(\beta) = [\cos\phi, \sin\phi] \times [-e_1, -e_2] = e_1 \sin\phi - e_2 \cos\phi$. The purpose of the control design, illustrated next, is to drive e_d and β to zero. Towards that purpose, we set reference signals for the vehicle's velocity as

$$v_\text{d} := \frac{1}{\cos(\beta)}(\dot{p}_{\text{d},1} \cos(\beta + \phi) + \dot{p}_{\text{d},2} \sin(\beta + \phi) + k_d e_d) \tag{6a}$$

$$\omega_\text{d} := -\frac{\sin(\phi)\dot{p}_{\text{d},1}}{\cos(\beta) e_d} + \frac{\cos(\phi)\dot{p}_{\text{d},2}}{\cos(\beta) e_d} + k_d \tan\beta + k_\beta \beta \tag{6b}$$

where k_d, k_β are positive gains, aiming to create exponentially stable subsystems for $\dot{e}_d$ and $\dot{\beta}$. We define the respective velocity errors $e_v := v - v_\text{d}$, $e_\omega := \omega - \omega_\text{d}$ and design the adaptive and neural-network-based control input as $u(x,\hat{\ell}_v,\hat{\ell}_\omega,\hat{\ell}_a,t) := [\frac{u_S+u_D}{2}, \frac{u_S-u_D}{2}]^\top + u_{\text{nn}}(x,t)$, with

$$u_S := \hat{\ell}_v\dot{v}_{\text{d}} - (k_v+\hat{\ell}_a)e_v + e_d\cos\beta - \beta\frac{\sin\beta}{e_d} \tag{7a}$$

$$u_D := \hat{\ell}_\omega\dot{\omega}_{\text{d}} - (k_\omega+\hat{\ell}_a)e_\omega + \beta \tag{7b}$$

$$\dot{\hat{\ell}}_v := -k_v e_v\dot{v}_{\text{d}},\ \dot{\hat{\ell}}_\omega := -k_\omega e_\omega\dot{\omega}_{\text{d}},\ \dot{\hat{\ell}}_a := k_a(e_v^2+e_\omega^2) \tag{7c}$$

where $\hat{\ell}_v$, $\hat{\ell}_\omega$, and $\hat{\ell}_a$ are adaptation variables (similar to (4)), with $\hat{\ell}_a(0)>0$, and k_v, k_ω, k_a, are positive gains, We now restate Assumption 3 to apply for the unicycle analysis, where we use $\widetilde{u}_{\text{nn}}(x,t) := [u_{\text{nn},R}(x,t)+u_{\text{nn},L}(x,t), u_{\text{nn},R}(x,t)-u_{\text{nn},L}(x,t)]^\top$, $\widetilde{f}_\theta(x,t) := [f_{\theta,R}(x,t)+u_{\theta,L}(x,t), u_{\theta,R}(x,t)-u_{\theta,L}(x,t)]^\top$, and $e_g := [e_v,e_\omega]^\top$.

Assumption 3: There exists $r>0$ such that the output $u_{\text{nn}}(x,t)$ of the trained neural network satisfies

$$e_g^\top\left(\widetilde{u}_{\text{nn}}(x(e,t),t) + \widetilde{f}_\theta(x(e,t),t)\right) \leq \kappa e_g \tag{8}$$

for all $\|x\|\leq r$, $t\geq 0$, where κ is an unknown positive constant.

The next corollary establishes the stability of the proposed scheme for the unicycle dynamics.

Corollary 1: Let the unicycle system (5) and let a reference trajectory $p_{\text{d}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$. Assume that $\beta(t)\in(-\bar{\beta},\bar{\beta})$, $|\dot{p}_{\text{d},1}\sin\phi - \dot{p}_{\text{d},2}\cos\phi| < e_d\alpha_1$, for positive constants $\bar{\beta}<\frac{\pi}{2}$, α_1, and all $t\geq 0$. Under Assumption 3, there exists a set $\mathbb{M}_x$ such that, if $[x(0)^\top,\hat{\ell}_v(0),\hat{\ell}_\omega(0),\hat{\ell}_a(0)]^\top \in \mathbb{M}_x$, the control algorithm (7) guarantees $\lim_{t\to\infty}e_d(t) = \lim_{t\to\infty}\beta(t) = \lim_{t\to\infty}e_v(t) = \lim_{t\to\infty}e_\omega(t) = 0$, and the boundedness of all closed-loop signals.

The assumptions $|\dot{p}_{\text{d},1}\sin\phi - \dot{p}_{\text{d},2}\cos\phi| < e_d\alpha_1$, $\sin\beta < e_d\alpha_2$ are imposed to avoid the singularity of $e_d=0$; note that β and ω_{d} are not defined in that case. Intuitively, they imply that e_d will not be driven to zero faster than β or $\dot{p}_{\text{d},1}\sin\phi - \dot{p}_{\text{d},2}\cos\phi$; the latter becomes zero when the vehicle's velocity vector v aligns with the desired one $\dot{p}_{\text{d}}$.

Proof: By differentiating e_d and β, and using (5) as well as the relations $e_1 = -e_d\cos(\phi+\beta)$, $e_2 = -e_d\sin(\phi+\beta)$ (see left of Fig. 1), we derive

$$\dot{e}_d = -v\cos\beta + \dot{p}_{\text{d},1}\cos(\phi+\beta) + \dot{p}_{\text{d},2}\sin(\phi+\beta) \tag{9a}$$

$$\dot{\beta} = -\omega + \frac{\sin\beta}{e_d}v - \frac{\dot{p}_{\text{d},1}}{e_d}\sin(\phi+\beta) + \frac{\dot{p}_{\text{d},2}}{e_d}\cos(\phi+\beta) \tag{9b}$$

which can be written as

$$\begin{bmatrix}\dot{e}_d\\ \dot{\beta}\end{bmatrix} = G\begin{bmatrix}v\\ \omega\end{bmatrix} + \begin{bmatrix}\dot{p}_{\text{d},1}\cos(\phi+\beta)+\dot{p}_{\text{d},2}\sin(\phi+\beta)\\ \frac{\dot{p}_{\text{d},2}}{e_d}\cos(\phi+\beta) - \frac{\dot{p}_{\text{d},1}}{e_d}\sin(\phi+\beta)\end{bmatrix}$$

where $G := \begin{bmatrix}-\cos\beta & 0\\ \frac{\sin\beta}{e_d} & -1\end{bmatrix}$. It can be verified that the reference velocity signals, designed in (6), can be written as

$$\begin{bmatrix}v_{\text{d}}\\ \omega_{\text{d}}\end{bmatrix} = G^{-1}\begin{bmatrix}-\dot{p}_{\text{d},1}\cos(\phi+\beta) - \dot{p}_{\text{d},2}\sin(\phi+\beta) - k_d e_d\\ -\frac{\dot{p}_{\text{d},2}}{e_d}\cos(\phi+\beta) + \frac{\dot{p}_{\text{d},1}}{e_d}\sin(\phi+\beta) - k_\beta\beta\end{bmatrix}$$

and therefore, by using the relations $v = e_v + v_{\text{d}}$ and $\omega = e_\omega + \omega_{\text{d}}$, $\dot{e}_d$ and $\dot{\beta}$ can be written as

$$\begin{bmatrix}\dot{e}_d\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-k_d e_d\\ -k_\beta\beta\end{bmatrix} + G\begin{bmatrix}e_v\\ e_\omega\end{bmatrix} = \begin{bmatrix}-k_d e_d\\ -k_\beta\beta\end{bmatrix} + \begin{bmatrix}-e_v\cos\beta\\ e_v\frac{\sin\beta}{e_d} - e_\omega\end{bmatrix} \tag{10}$$

The control design follows the back-stepping methodology [8]. Let, therefore, the function $V_1 := \frac{1}{2}(e_d^2+\beta^2)$, which, after time differentiation and use of (10), yields

$$\dot{V}_1 = -k_d e_d^2 - k_\beta\beta^2 - e_d e_v\cos\beta + \beta e_v\frac{\sin\beta}{e_d} - \beta e_\omega \tag{11}$$

We will cancel the last terms of $\dot{V}_1$ using the control input (7). First, we note that the inertia matrix of the system M, appearing in the unicycle dynamics (5), has the form [28] $M = \begin{bmatrix}M_1 & M_2\\ M_2 & M_1\end{bmatrix}$ with $M_1 := \frac{mr_w^2}{4} + \frac{(I_C+md^2)r_w^2}{4R^2} + I_0$, $M_2 := \frac{mr_w^2}{4} - \frac{(I_C+md^2)r_w^2}{4R^2}$, and where I_C is the moment of inertia of the vehicle with respect to point C (see left of Fig. 1), I_0 is the moment of inertia of the the wheels, and d is the distance of the between point C and the vehicle's center of mass (p_1,p_2). Therefore, the second part of the unicycle dynamics (5) can be written as

$$M_1\ddot{\theta}_R + M_2\ddot{\theta}_L = u_R + f_{\theta,R}$$
$$M_2\ddot{\theta}_R + M_1\ddot{\theta}_L = u_L + f_{\theta,L}$$

with $f_{\theta,R}$ and $f_{\theta,L}$ denoting the elements of f_θ. By summing and subtracting the aforementioned equations, we obtain

$$(M_1+M_2)(\ddot{\theta}_R+\ddot{\theta}_L) = u_R + u_L + f_{\theta,R} + f_{\theta,L} \tag{12a}$$

$$(M_1-M_2)(\ddot{\theta}_R-\ddot{\theta}_L) = u_R - u_L + f_{\theta,R} - f_{\theta,L} \tag{12b}$$

From the definition of e_v and e_ω, it holds that $\dot{e}_v = \dot{v} - \dot{v}_{\text{d}}$, $\dot{e}_\omega = \dot{\omega} - \dot{\omega}_{\text{d}}$, which, by using the relations $v = \frac{r_w}{2}(\dot{\theta}_R + \dot{\theta}_L)$, $\omega = \frac{r_w}{2R}(\dot{\theta}_R - \dot{\theta}_L)$ and (12), becomes

$$\dot{e}_v = \frac{r_w}{2(M_1+M_2)}\left(u_R + u_L + f_{\theta,R}(x,t) + f_{\theta,L}(x,t)\right) - \dot{v}_{\text{d}} \tag{13a}$$

$$\dot{e}_\omega = \frac{r_w}{2R(M_1-M_2)}\left(u_R - u_L + f_{\theta,R}(x,t) - f_{\theta,L}(x,t)\right) - \dot{\omega}_{\text{d}} \tag{13b}$$

Define now $\ell_v := 2\frac{M_1+M_2}{r_w}$, $\ell_\omega := 2R\frac{M_1-M_2}{r_w}$. The adaptation terms $\hat{\ell}_v$, and $\hat{\ell}_\omega$, used in the control mechanism (7), aim to approximate ℓ_v and ℓ_ω, respectively. Note that $\hat{\ell}_v$ and $\hat{\ell}_\omega$ are positive, and we define the errors $\widetilde{\ell}_v := \hat{\ell}_v - \ell_v$, $\widetilde{\ell}_\omega := \hat{\ell}_\omega - \ell_\omega$.

Furthermore, we define the error $\widetilde{\ell}_a := \hat{\ell}_a - \kappa$, where κ as in the right-hand-side of (8), and we consider the continuously differentiable and positive definite function

$$V := V_1 + \frac{\ell_v}{2}e_v^2 + \frac{\ell_\omega}{2}e_\omega^2 + \frac{1}{2k_v}\widetilde{\ell}_v^2 + \frac{1}{2k_\omega}\widetilde{\ell}_\omega^2 + \frac{1}{2k_a}\widetilde{\ell}_a^2$$

which, by differentiating V and employing (11) and (12), becomes

$$\begin{aligned}\dot{V} = &-k_d e_d^2 - k_\beta\beta^2 - e_d e_v\cos\beta + \beta e_v\frac{\sin\beta}{e_d} - \beta e_\omega\\ &+ e_v\big(u_R + u_L + f_{\theta,R} + f_{\theta,L} - \ell_v\dot{v}_{\text{d}}\big)\\ &+ e_\omega\big(u_R - u_L + f_{\theta,R} - f_{\theta,L} - \ell_\omega\dot{\omega}_{\text{d}}\big)\\ &+ \frac{1}{k_v}\widetilde{\ell}_v\dot{\hat{\ell}}_v + \frac{1}{k_\omega}\widetilde{\ell}_\omega\dot{\hat{\ell}}_\omega + \frac{1}{k_i}\widetilde{\ell}_a\dot{\hat{\ell}}_a\end{aligned}$$

By substituting the control and adaptation algorithm (7), $\dot{V}$ becomes

$$\begin{aligned}\dot{V} =& -k_d e_d^2 - k_\beta \beta^2 - k_v e_v^2 - k_\beta e_\omega^2 + e_v \dot{v}_\mathrm{d} \widetilde{\ell}_v + e_\omega \dot{\omega}_\mathrm{d} \widetilde{\ell}_\omega \\ & - \hat{\ell}_a (e_v^2 + e_\omega^2) + e_v (u_{\mathrm{nn},R} + u_{\mathrm{nn},L} + f_{\theta,R} + f_{\theta,L}) \\ & + e_\omega (u_{\mathrm{nn},R} - u_{\mathrm{nn},L} + f_{\theta,R} - f_{\theta,L}) - \widetilde{\ell}_v e_v \dot{v}_\mathrm{d} - \widetilde{\ell}_\omega e_\omega \dot{\omega}_\mathrm{d} \\ & + \widetilde{\ell}_a (e_v^2 + e_\omega^2)\end{aligned}$$

Finally, by using (8) and $\widetilde{\ell}_a = \hat{\ell}_a - \kappa$, $\dot{V}$ becomes

$$\dot{V} = -k_d e_d^2 - k_\beta \beta^2 - k_v e_v^2 - k_\beta e_\omega^2$$

which implies that $V(t) \leq V(0)$ and the boundedness of $e_d(t)$, $\beta(t)$, $e_v(t)$, $e_\omega(t)$, $\widetilde{\ell}_v(t)$, $\widetilde{\ell}_\omega(t)$, and $\widetilde{\ell}_a(t)$, for all $t \geq 0$. By differentiating $\dot{V}$ and using (10), (13), and (7), one further concludes the boundedness of $\ddot{V}$ and hence the uniform continuity of $\dot{V}$. Therefore, by applying Barbalat's Lemma (Theorem 8.4 of [27]), we conclude that $\lim_{t\to\infty} e_d(t) = \lim_{t\to\infty} \beta(t) = \lim_{t\to\infty} e_v(t) = \lim_{t\to\infty} e_\omega(t) = 0$.

Under Assumption 3, the aforementioned results hold under the condition that $\|x\| \leq r$. Therefore, we need to establish that the proposed control algorithm and initial conditions do not force $\|x(t)\|$ to grow larger than r at any point in time $t \geq 0$. To that end, let first compact sets $\Omega_\ell \subset \mathbb{R}^3_{\geq 0}$, $\Omega_{1,\mathrm{d}} \subset \mathbb{R}^n$, $\Omega_{2,\mathrm{d}} \subset \mathbb{R}^n$ satisfying $\widetilde{\ell}(t) := [\widetilde{\ell}_v, \widetilde{\ell}_\omega, \widetilde{\ell}_a]^\top \in \Omega_\ell$, $p_\mathrm{d}(t) \in \Omega_{1,\mathrm{d}}$, and $\dot{p}_\mathrm{d}(t) \in \Omega_{2,\mathrm{d}}$, for all $t \geq 0$. Additionally, let $\Omega_e := \{e := [e_d, \beta, e_v, e_\omega]^\top \in \mathbb{R}^4 : \beta \in (-\bar{\beta}, \bar{\beta}), |\dot{p}_{\mathrm{d},1} \sin\phi - \dot{p}_{\mathrm{d},2} \cos\phi| < e_d \alpha_1, \|x\| \leq r, p_\mathrm{d}(t) \in \Omega_{1,\mathrm{d}}, \dot{p}_\mathrm{d}(t) \in \Omega_{2,\mathrm{d}}\}$ as well as $\mathbb{M} := \{[e_d, \beta, e_v, e_\omega, \widetilde{\ell}^\top]^\top \in \mathbb{R}^7 : V \leq V_0\}$, where V_0 is the largest positive constant for which $\mathbb{M} \subseteq \Omega_e \times \Omega_\ell$. Then, for all $[e(0)^\top, \widetilde{\ell}(0)^\top]^\top \in \mathbb{M} \Leftrightarrow [x(0)^\top, \hat{\ell}_v(0), \hat{\ell}_\omega(0), \hat{\ell}_a(0)]^\top \in \mathbb{M}_x$, for some $\mathbb{M}_x$, it follows that $V(t)$ is bounded from above by V_0 for all $t \geq 0$, which implies that $e(t) \in \Omega_e$, for all $t \geq 0$. Hence, it holds that $\|x(t)\| \leq r$ for all $t \geq 0$, leading to the conclusion of the proof.

■

IV. Numerical Experiments

This section is devoted to a series of numerical experiments. We first test the proposed algorithm on a 6-dof UR5 robotic manipulator with dynamics

$$\dot{x}_1 = x_2 \tag{14a}$$

$$\dot{x}_2 = B(x_1)^{-1} \left(u - C(x) x_2 - g(x_1) + d(x,t)\right) \tag{14b}$$

where $x_1, x_2 \in \mathbb{R}^6$ are the vectors of robot joint angles and angular velocities, respectively; $B(x_1) \in \mathbb{R}^{6\times 6}$ is the manipulator's positive definite inertia matrix, $C(x) \in \mathbb{R}^{6\times 6}$ is the Coriolis matrix, $g(x_1) \in \mathbb{R}^6$ is the gravity vector, and $d(x,t) \in \mathbb{R}^6$ is a vector of friction terms and exogenous time-varying disturbances. The workspace consists of obstacles and four points of interest $T_1 = [-0.15, -0.475, 0.675, \frac{\pi}{2}, 0, 0]^\top$, $T_2 = [-0.6, 0, 2.5, 0, -\frac{\pi}{2}, -\frac{\pi}{2}]^\top$, $T_3 = [-0.025, 0.595, 0.6, -\frac{\pi}{2}, 0, \pi]^\top$, and $T_4 = [-0.525, -0.55, 0.28, \pi, 0, -\frac{\pi}{2}]^\top$ (end-effector position and Euler-angle orientation), and the corresponding joint-angle vectors as $c_1 = [-0.07, -1.05, 0.45, 2.3, 1.37, -1.33]^\top$,

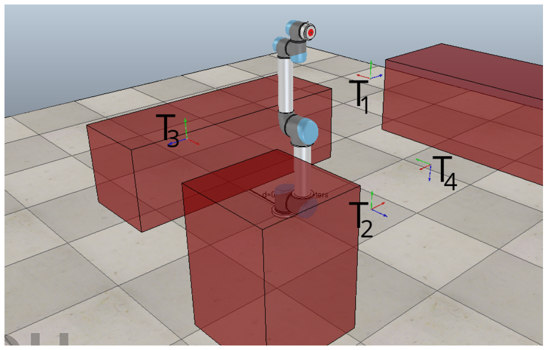

Fig. 2: A UR5 robot in a workspace with four points of interest T_i, $i \in \{1, \dots, 4\}$.

$c_2 = [1.28, 0.35, 1.75, 0.03, 0.1, -1.22]^\top$, $c_3 = [-0.08, 0.85, -0.23, 2.58, 2.09, -2.36]^\top$, $c_4 = [-0.7, -0.76, -1.05, -0.05, -3.08, 2.37]^\top$ (radians).

We consider a nominal task expressed via the spatio-temporal constraint $\bigwedge_{i\in\{1,\dots,4\}} G_{[0,\infty)} F_{I_i} (\|x_1 - c_i\| \leq 0.1)$, where G and F are the always and eventually operators respectively. The task consists of visits of x_1 to $c_i \in \mathbb{R}^6$ (within the radius 0.1) infinitely often within the time intervals dictated by I_i, for $i \in \{1, \dots, 4\}$, while avoiding obstacles. The reference trajectories are generated by an RRT algorithm.

We set a nominal value for the time intervals as $I_i = [0, 20]$ (seconds), and we create 100 problem instances by varying the following attributes: firstly, we add uniformly random offsets in $[-0.3, 0.3]$ (radians) to the elements of all c_i, and in $[-2, 2]$ (seconds) to the right end-points of the intervals I_i; secondly, we add random offsets to the dynamic parameters of the robot (masses and moments of inertia of the robot's links and actuators) and we set a different friction and disturbance term $d(\cdot)$, leading to a different dynamic model in (14); thirdly, we set different sequences of visits to the points c_i, $i \in \{1, \dots, 4\}$, as dictated by ϕ, i.e., one trajectory might correspond to the visit sequence $((x(0), 0) \to (c_1, t_{1_1}) \to (c_2, t_{1_2}) \to (c_3, t_{1_3}) \to (c_4, t_{1_4})$, and another to $((x(0), 0) \to (c_3, t_{1_3}) \to (c_1, t_{1_1}) \to (c_4, t_{1_4}) \to (c_2, t_{1_2})$. Finally, we add uniformly random offsets in $[-0.5, 0.5]$ to the initial position of the robot (from the first point of the sequence), and we set its initial velocity randomly in the interval $[0, 1]^6$.

Regarding the dynamics (14), we use the methodology described in [29] to derive the B, C, and g terms. We set nominal link masses and moments of inertia as $m = [1, 2.5, 5.7, 3.9, 2.5, 2.5, 0.7]$ (kg) and $I = [0.02, 0.04, 0.06, 0.05, 0.04, 0.04, 0.01]$ (kgm^2), respectively, and we add random offsets in $(-\frac{m}{2}, \frac{m}{2})$, $(-\frac{I}{2}, \frac{I}{2})$ in the created instances. Regarding the function $d()$ used in (14); we set $d(x,t) = d_t(t) + d_f(x)$, where

$$d_t = A_t \begin{bmatrix} \sin(\eta_1 t + \varphi_1) \\ \vdots \\ \sin(\eta_6 t + \varphi_6) \end{bmatrix}$$

$$d_f = B_t \dot{x} \otimes \dot{x}$$

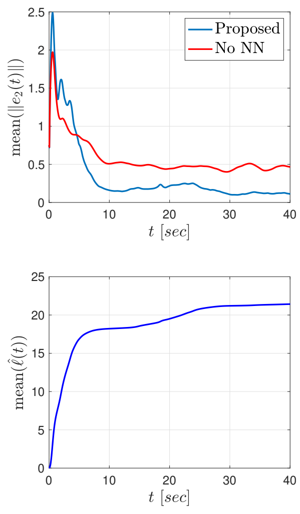
Fig. 3: Top: Mean (left) and standard deviation (right) of $\|e_2(t)\|$ for the proposed and no-neural-network control algorithms. Bottom: Mean (left) and standard deviation (right) of the adaptation variable $\hat{\ell}(t)$ for the proposed algorithm.

$A_t = \text{diag}\{ A_{t_i}\}_{i\in\{1,\dots,6\}} \in \mathbb{R}^{6\times 6}$, A_{t_i} is a random term in $(0, 2m_i)$, η_i is a random term in $(0,1)$, φ_i is a random term in $(0,2)$, $B_t \in \mathbb{R}^{6\times 36}$ is a random matrix taking values in $(0, 2m_i)$, and $\otimes$ denotes the Kronecker product.

We separate the aforementioned 100 problem instances into 50 training instances and 50 test instances. We generate trajectories using the 50 training instances from system runs that satisfy different variations of one cycle of the task (i.e., one visit to each point). Each trajectory consists of 500 points and is generated using a nominal model-based controller. We use these trajectories to train a neural network and we test the control algorithm (4) in the 50 test instances. The neural networks consist of 4 fully connected layers of 512 neurons; each layer is followed by a batch normalization module and a ReLU activation function. For the training we use the adam optimizer and the mean-square-error loss function. In all cases we choose a batch size of 256, and we train until a desirable average (per batch) loss of the order of 10^{-4} is achieved. All the values of the data used for the training were normalized in $[0,1]$. We chose the control gains of the control law (4) as $k_1 = 1, k_2 = 10$, and $k_\ell = 1$. We also compare our algorithm with the the adaptive controller $u_c(x,t) = -(k_2 + \hat{\ell})e_2$, $\dot{\hat{\ell}} = k_\ell \|e_2\|^2$ that does not employ the neural network (i.e., the term $u_{\text{nn}}(x,t)$), using the same control gains.

The comparison results are depicted in the top of Fig. 3, which depicts the mean and standard deviation of the signal $\|e_2(t)\|$ for the 50 instances and 40 seconds. One concludes that the proposed algorithm achieves better performance in terms of error magnitude than the no-neural-network algorithm. Additionally, the bottom of Fig. 3 depicts the mean and standard deviation of the adaptation variable $\hat{\ell}(t)$ for the 50 test instances. Finally, Fig. 4 illustrates the control inputs $u(x,t)$ as well as the neural-network outputs $u_{\text{nn}}(x,t)$. Note that the control input converges to the neural-network

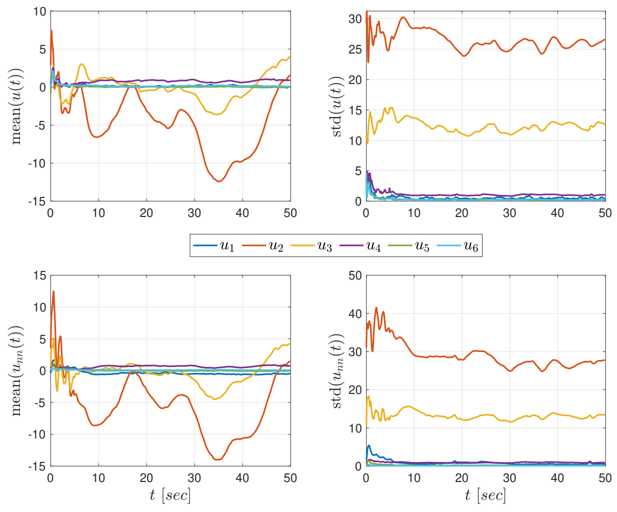
Fig. 4: Mean (left) and standard deviation (right) of the control inputs $u(x,t)$ (top) and the neural-network outputs $u_{\text{nn}}(x,t)$ (bottom) for the proposed algorithm.

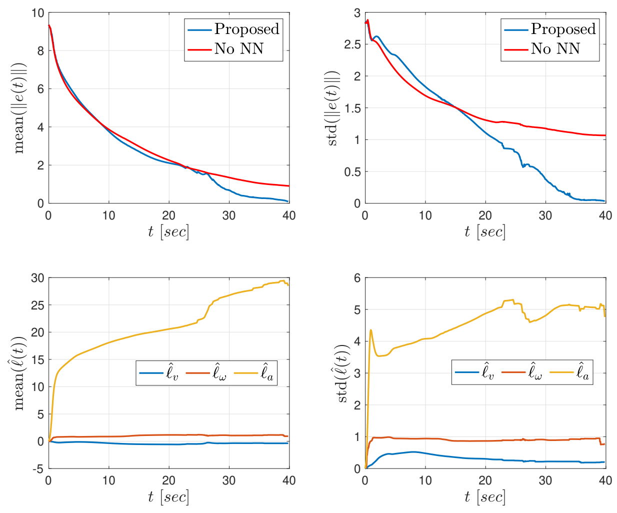
Fig. 5: Top: Mean (left) and standard deviation (right) of $\|e(t)\| = \|e_d(t), \beta(t), e_v(t), e_\omega(t)\|$ for the proposed and no-neural-network control algorithms. Bottom: Mean (left) and standard deviation (right) of the adaptation variables $\hat{\ell}_v(t)$, $\hat{\ell}_\omega(t)$, and $\hat{\ell}_a(t)$ for the proposed algorithm.

output, i.e., $\lim_{t\to\infty}(u(t) - u_{\text{nn}}(t)) = 0$, which can be also verified by (4) and the fact that $\lim_{t\to\infty} e_2(t) = 0$.

We next test the proposed algorithm, following a similar procedure, on a unicycle robot. The dynamic terms in (5) have the form

$$M = \begin{bmatrix} M_1 & M_2 \\ M_2 & M_1 \end{bmatrix}$$
$$f_\theta(x,t) = d(x,t)$$

with $M_1 := \frac{mr^2}{4} + \frac{(I_C + md^2)r^2}{4R^2} + I_0$, $M_2 := \frac{mr^2}{4} - \frac{(I_C+md^2)r^2}{4R^2}$, and where I_C is the moment of inertia of the vehicle with respect to point C (see left of Fig. 1), I_0 is the moment of inertia of the the wheels, and d is the distance of the between point C and the vehicle's center of mass (p_1, p_2). The term $d(x,t)$ is chosen as in the robotic-manipulator case. The task here is to track a reference trajectory $p_{\text{d}}(t) \in \mathbb{R}^2$, shown in the right of Fig. 1. We derived 100 problem instances by varying the dynamic and geometric parameters (elaborated subsequently) and the function $d(x,t)$, the unicycle's initial position and orientation with random offsets in $[-0.25, 0.25]$ (rad) from $\theta(0) = \arctan(e_2(0)/e_1(0))$, and we further set random values in $[-0.25, 0.25]$ (rad/s) to the initial wheel velocities. Regarding the dynamic and geometric parameters, which appear in the inertia matrix, we set $m = 28$ (kg), $I_C = 0.1$ (kgm^2), $I_0 = 0.01$ (kgm^2), $r = 0.01$ (m), $d = 0.01$ (m), and we added random offsets in $(-\frac{m}{2}, \frac{m}{2})$, $(-\frac{I_A}{2}, \frac{I_A}{2})$, $(-\frac{I_0}{2}, \frac{I_0}{2})$, $(-\frac{r}{2}, \frac{r}{2})$, $(-\frac{d}{2}, \frac{d}{2})$, respectively, for the problem instances. We chose the control gains of (7) as $k_d = 0.01$, $k_\beta = 2$, $k_v = 0.01$, $k_\omega = 0.01$, , $k_v = 3$, $k_\omega = 1$, and $k_a = 1$. For the training, we chose the parameters as in the robotic manipulator case. We further compared with a version of (7) that does not use the neural-network outputs.

The comparison results are depicted in the top pf Fig. 5, which depicts the mean and standard deviation of the error $e = [e_d, \beta, e_v, e_\omega]^\top$, for the 50 test instances. One concludes that the proposed algorithm achieves better performance in terms of error magnitude than the no-neural-network algorithm. Additionally, the bottom of Fig. 5 depicts the mean and standard deviation of the adaptation variables $\hat{\ell} = [\hat{\ell}_v, \hat{\ell}_\omega, \hat{\ell}_a]^\top$ for the 50 test instances.

V. Conclusion and Future Work

We develop an algorithm for the control of systems with unknown nonlinear dynamics. The algorithm integrates neural network-based learning and adaptive control. Future directions will focus on relaxing the considered assumptions and create connections among the off-line data, the training, and the control performance.

References

[1] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[2] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[3] Y. Yang, K. G. Vamvoudakis, and H. Modares, "Safe reinforcement learning for dynamical games," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 9, pp. 3706–3726, 2020.

[4] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "A neural network solution for fixed-final time optimal control of nonlinear systems," *Automatica*, vol. 43, no. 3, pp. 482–490, 2007.

[5] J. Zhao and M. Gan, "Finite-horizon optimal control for continuous-time uncertain nonlinear systems using reinforcement learning," *International Journal of Systems Science*, vol. 51, no. 13, pp. 2429–2440, 2020.

[6] R. Kamalapurkar, H. Dinh, S. Bhasin, and W. E. Dixon, "Approximate optimal trajectory tracking for continuous-time nonlinear systems," *Automatica*, vol. 51, pp. 40–48, 2015.

[7] X. Huang, Y. Song, and J. Lai, "Neuro-adaptive control with given performance specifications for strict feedback systems under full-state constraints," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 1, pp. 25–34, 2018.

[8] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, "Nonlinear and Adaptive Control Design," *Publisher: Wiley New York*, 1995.

[9] F. Hong, S. Ge, B. Ren, and T. Lee, "Robust adaptive control for a class of uncertain strict-feedback nonlinear systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 19, no. 7, pp. 746–767, 2009.

[10] Z. Chen, "Nussbaum functions in adaptive control with time-varying unknown control coefficients," *Automatica*, vol. 102, pp. 72–79, 2019.

[11] J. Huang, W. Wang, C. Wen, and J. Zhou, "Adaptive control of a class of strict-feedback time-varying nonlinear systems with unknown control coefficients," *Automatica*, vol. 93, pp. 98–105, 2018.

[12] H. Xu and P. A. Ioannou, "Robust adaptive control for a class of mimo nonlinear systems with guaranteed error bounds," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 728–742, 2003.

[13] S. S. Ge and C. Wang, "Adaptive neural control of uncertain mimo nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 674–692, 2004.

[14] C. Wen, J. Zhou, Z. Liu, and H. Su, "Robust adaptive control of uncertain nonlinear systems in the presence of input saturation and external disturbance," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1672–1678, 2011.

[15] J.-J. E. Slotine and W. Li, "On the adaptive control of robot manipulators," *The international journal of robotics research*, vol. 6, no. 3, pp. 49–59, 1987.

[16] J.-J. Slotine and J. Coetsee, "Adaptive sliding controller synthesis for non-linear systems," *International Journal of Control*, vol. 43, no. 6, pp. 1631–1651, 1986.

[17] J. Chen, A. Behal, and D. M. Dawson, "Robust feedback control for a class of uncertain mimo nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 591–596, 2008.

[18] B. Xian, D. M. Dawson, M. S. de Queiroz, and J. Chen, "A continuous asymptotic tracking control strategy for uncertain nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 49, no. 7, pp. 1206–1211, 2004.

[19] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, "Learning and control using gaussian processes," *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pp. 140–149, 2018.

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

[21] T. Berger, H. H. Lê, and T. Reis, "Funnel control for nonlinear systems with known strict relative degree," *Automatica*, vol. 87, pp. 345–357, 2018.

[22] C. P. Bechlioulis and G. A. Rovithakis, "A low-complexity global approximation-free control scheme with prescribed performance for unknown pure feedback systems," *Automatica*, vol. 50, no. 4, pp. 1217–1226, 2014.

[23] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki, "Kdf: Kinodynamic motion planning via geometric sampling-based algorithms and funnel control," *IEEE Transactions on Robotics*, 2022.

[24] D. Bertsekas, "Lessons from alphazero for optimal, model predictive, and adaptive control," *arXiv preprint arXiv:2108.10315*, 2021.

[25] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control.* Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[26] G. Joshi, J. Virdi, and G. Chowdhary, "Asynchronous deep model reference adaptive control," *Conference on Robot Learning*, 2020.

[27] H. K. Khalil, "Nonlinear Systems," *Prentice Hall*, 2002.

[28] E. Ivanjko, T. Petrinic, and I. Petrovic, "Modelling of mobile robot dynamics," *7th EUROSIM Congress on Modelling and Simulation*, vol. 2, 2010.

[29] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Modelling, planning and control," *Advanced Textbooks in Control and Signal Processing. Springer,*, 2009.