# Constrained Optimization-Based Neuro-Adaptive Control (CONAC) for Euler-Lagrange Systems Under Weight and Input Constraints

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

***Abstract*—This study presents a constrained optimization-based neuro-adaptive control (CONAC) for unknown Euler-Lagrange systems subject to weight and convex input constraints. A deep neural network (DNN) is employed to approximate the ideal stabilizing control law while simultaneously learning the unknown system dynamics and addressing both types of constraints within a unified constrained optimization framework. The adaptation law of DNN weights is formulated to solve the defined constrained optimization problem, ensuring satisfaction of first-order optimality conditions at steady state. The controller's stability is rigorously analyzed using Lyapunov theory, guaranteeing bounded tracking errors and bounded DNN weights. The proposed controller is validated through a real-time implementation on a 2-DOF robotic manipulator, demonstrating its effectiveness in achieving accurate trajectory tracking while satisfying all imposed constraints.**

***Index Terms*—Neuro-adaptive control, constrained optimization, deep neural network, input constraint, weight constraint.**

## NOTATION

The Kronecker product is denoted by $\otimes$ [1, Chap. 7, Def. 7.1.2]. A vector and a matrix are denoted by $\boldsymbol{x} = [x_i]_{i\in\{1,\cdots,n\}} \in \mathbb{R}^n$ and $\boldsymbol{A} := [a_{ij}]_{i\in\{1,\cdots,n\},j\in\{1,\cdots,m\}} \in \mathbb{R}^{n\times m}$, respectively. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, $\mathrm{row}_i(\boldsymbol{A})$ denotes the $i$th row of $\boldsymbol{A}$. For the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, $\mathrm{vec}(\boldsymbol{A}) := (\mathrm{row}_1(\boldsymbol{A}^\top), \cdots, \mathrm{row}_m(\boldsymbol{A}^\top))^\top \in \mathbb{R}^{nm}$ denotes the vectorization of $\boldsymbol{A}$. $\lambda_{\min}(\boldsymbol{A})$ denotes the minimum eigenvalue of the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. The $n \times n$ identity and $n \times m$ zero matrices are denoted by $\boldsymbol{I}_n$ and $\boldsymbol{0}_{n\times m}$, respectively.

## I. INTRODUCTION

### A. Background

MANY engineering systems, including those in aerospace, robotics, and automotive applications, can be modeled using Euler-Lagrange systems. These systems are governed by dynamic equations derived from energy principles and describe the motion of mechanical systems with constraints. In practice, however, such systems often exhibit uncertainties due to unmodeled dynamics, parameter variations, or external disturbances. These uncertainties can significantly degrade control performance and, in some cases, lead to instability. In the worst cases, any prior knowledge of the system dynamics may not be available, making it challenging to design effective controllers. To address these challenges, adaptive control methods have been widely employed to ensure robust performance for uncertain or unknown systems [2], [3].

More recently, neuro-adaptive control (NAC) approaches have been introduced to approximate uncertain or unknown system dynamics or entire control laws using neural networks (NNs) [4], [5]. NNs are well-known for their universal approximation property, which allows them to approximate any smooth function over a compact set with minimal error. Various types of NNs have been utilized in NAC, including simpler architectures like single-hidden layer (SHL) NNs [6]–[10] and radial basis function (RBF) NNs [11], [12], as well as more complex models like deep NNs (DNNs) [13] and their variations. Conventionally, SHL and RBF NNs are often employed to approximate uncertain or unknown system dynamics or controllers due to their simplicity [8], [14]–[16]. However, since DNNs offer greater expressive power, making them more effective for complex system approximations [17], variations of DNNs, such as long short-term memory (LSTM) networks for time-varying dynamics [18], physics-informed NNs (PINNs) for leveraging physical system knowledge [19], and convolutional NNs (CNNs) for learning from historical sensor data [20], have further extended the capabilities of NAC systems.

A critical aspect of NAC is the NN weight adaptation law, which governs how NN weights are updated. Most studies derived these laws using Lyapunov-based methods, ensuring the boundedness of the tracking error and weight estimation error, thus maintaining system stability under uncertainty.

However, two significant challenges persist in using NNs for adaptive control. First, the boundedness of NN weights is not inherently guaranteed, which can result in unbounded outputs. When NN outputs are used directly in the control law, this may lead to excessive control inputs, violating input constraints. Such constraints are commonly encountered in industrial systems, where actuators are limited by physical and safety requirements in terms of amplitude, rate, or energy [21]. Failing to address these constraints can degrade control performance or even destabilize the system.

Therefore, addressing these two key issues—ensuring weight boundedness and satisfying input constraints—is essential for the reliable design of NAC. The following section provides a detailed review of existing solutions to these

Myeongseok Ryu and Kyunghwan Choi are with the Cho Chun Shik Graduate School of Mobility, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34051, Korea (e-mail: dding_98@kaist.ac.kr and kh.choi@kaist.ac.kr).

Donghwa Hong is with the Department of Mechanical and Robotics Engineering, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Korea (e-mail: fairytalef00@gm.gist.ac.kr).

This work was supported in part by a Korea Research Institute for Defense Technology planning and advancement (KRIT) grant funded by Korea government DAPA (Defense Acquisition Program Administration) (No. KRIT-CT-22-087, Synthetic Battlefield Environment based Integrated Combat Training Platform) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025- 00554087). (Corresponding author: Kyunghwan Choi)

challenges.

### B. Literature Review

*1) Ensuring Weight Norm Boundedness:* A common challenge in NAC is maintaining the boundedness of the NN weights to ensure stability. In many studies, projection operators were employed to enforce upper bounds on the weight norms, ensuring that the weights do not grow unboundedly. For example, in [13], [18], [22], projection operators were used to constrain the weight norms to remain below predetermined constants. However, these constants were often selected as large as possible due to the lack of theoretical guarantees regarding the global optimality of the weight values. While this approach ensured that the NN remained stable, it did not necessarily result in optimal performance.

In addition to projection operators, some studies utilized modification techniques like $\sigma$-modification [12] and $\epsilon$-modification [15], [16]. These methods ensured that the NN weights remained within an invariant set by incorporating stabilizing functions into the adaptation law. Although these techniques were effective in ensuring boundedness and avoiding weight divergence, they similarly lacked a formal analysis of the optimality of the adapted weights by biasing the weights toward the origin [9]. This leaves room for improvement in terms of performance optimization.

*2) Satisfying Input Constraints:* The second major issue is satisfying input constraints, particularly in systems where actuators are subject to physical limitations. This issue becomes more pronounced in NAC architectures where NNs are used to augment conventional model-based controllers—such as feedback linearization or backstepping—rather than acting as the sole control input. In such cases, if the system model is inaccurate or the dynamics targeted by the NN are already contributing to stability, the combined control effort may result in overly aggressive inputs, increasing the risk of input saturation [23].

To address the aforementioned issue, several approaches have been proposed in [14]–[16], [24]–[27] to handle input constraints in NAC systems. Particularly, the auxiliary systems have been widely introduced. In [24]–[26], the auxiliary states were generated whenever input saturation was detected, and these states were used as feedback terms in the control law to directly compensate for the effects of input saturation constraints. On the other hand, in [14], [15], [27], the auxiliary systems were incorporated into the adaptation law by adding the auxiliary states to the feedback error for learning. This approach helped the NN reduce input saturation by indirectly regulating the auxiliary states, during the adaptation process.

However, these approaches typically handle input bound constraints on a per-input basis, i.e., applying bounds to each scalar control variable individually, and may not account for more complex, nonlinear constraints, like input norm constraints, which are commonly found in physical systems such as robotic actuators or motor systems.

Furthermore, from a learning perspective, augmenting conventional control methods with NNs by simply adding NN outputs to existing control inputs can interfere with the adaptation process. This occurs because the adaptation law typically depends on the feedback error, which becomes insensitive to the NN output when the conventional controller dominates the system behavior. As a result, the NN ends up counteracting the conventional control effort rather than directly learning the system dynamics, thereby weakening its ability to adapt. To address this, input constraints should be handled within a unified adaptive framework—such as those proposed in [14], [15], [27]—where both adaptation and constraint satisfaction are treated jointly without relying on conventional control inputs.

*3) Potential of Constrained Optimization:* A promising approach to overcoming the limitations of the existing methods lies in constrained optimization. By formulating the NAC problem as a unified optimization problem with constraints, it is possible to adapt the NN weights while minimizing an objective function, e.g., tracking error, subject to both weight and input constraints, simultaneously. Constrained optimization provides a theoretical framework for defining optimality and presents numerical methods for finding solutions that satisfy the constraints [28].

In existing literature, constrained optimization techniques, such as the Augmented Lagrangian Method (ALM) [29] and the Alternating Direction Method of Multipliers (ADMM) [30], [31], have been used to train NNs offline. However, there are few works that have applied constrained optimization to NAC systems with real-time weight adaptation [9], [10]. This gap suggests that constrained optimization could be key to addressing both weight boundedness and input constraints in a unified, theoretically grounded framework, particularly in real-time NAC.

### C. Contributions

The main contributions of this study are listed as follows:

- A constrained optimization-based neuro-adaptive control (CONAC) is proposed, where the control problem is formulated as a constrained optimization problem. In this formulation, weight and convex input constraints are incorporated as inequality constraints, while minimizing a given objective function.
- Convex input constraints can be applied to the proposed CONAC, including input bound constraints and norm constraints. Moreover, the proposed CONAC can handle any combination of these constraints, which can be useful in practical applications where multiple constraints need to be satisfied simultaneously.
- Since the proposed CONAC approximates the entire ideal control law without any conventional controller, prior knowledge of the system dynamics or the system identification process for designing a nominal conventional controller can be avoided, which is often difficult and time-consuming to implement in practice.
- The adaptation laws for DNN weights and Lagrange multipliers are systematically derived to solve the defined problem, using constrained optimization theory.

These laws guarantee convergence to the first-order optimality conditions at steady state, specifically the Karush–Kuhn–Tucker (KKT) conditions, as described in [28, Chap. 12, Thm. 12.1].
- The proposed CONAC's stability is rigorously analyzed using Lyapunov theory, ensuring that the tracking error and DNN weights remain bounded. This analysis guarantees that the proposed CONAC can achieve stable online adaptation.
- The proposed CONAC is implemented in real-time on a 2-DOF robotic manipulator, demonstrating its effectiveness in achieving accurate trajectory tracking while satisfying all imposed constraints. The experimental results validate the theoretical findings and confirm the practical applicability of the proposed CONAC.

This study extends our preliminary work in [10], which introduced inequality constraints on the weight and input norms within a constrained optimization framework for neuro-adaptive control (NAC) systems using a SHLNN. In the present work, the same framework is generalized to accommodate arbitrary convex input constraints and extended to DNNs, while still retaining the capability to regulate the weight norm. Furthermore, unlike the previous study, which was limited to numerical simulations, the proposed CONAC is experimentally validated through real-time implementation on a 2-DOF robotic manipulator.

### D. Organization

The remainder of this paper is organized as follows. Section II presents the target system and control objective. Section III introduces the proposed CONAC and the architecture of the DNN used in the controller. In Section IV, the adaptation law is developed. Section V analyzes the stability of CONAC. Section VI presents a real-time implementation of CONAC on a 2-DOF robotic manipulator. Finally, Section VII concludes the paper and discusses potential future work. Candidates for convex input constraints are provided in Appendix A.

## II. Problem Formulation

### A. Model Dynamics and Control Objective

We consider an unknown Euler-Lagrange system modeled as

$$\boldsymbol{M}\tfrac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{q}+\boldsymbol{V}_m\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}+\boldsymbol{F}+\boldsymbol{G}+\boldsymbol{\tau}_d=\mathrm{sat}(\boldsymbol{\tau}), \tag{1}$$

where $\boldsymbol{q}\in\mathbb{R}^n$ denotes the generalized coordinate, $\boldsymbol{\tau}_d$ represents the disturbance and $\boldsymbol{\tau}\in\mathbb{R}^n$ denotes the generalized control input. The terms $\boldsymbol{M}:=\boldsymbol{M}(\boldsymbol{q})\in\mathbb{R}^{n\times n}$, $\boldsymbol{V}_m:=\boldsymbol{V}_m(\boldsymbol{q},\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q})\in\mathbb{R}^{n\times n}$, $\boldsymbol{F}:=\boldsymbol{F}(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q})\in\mathbb{R}^n$, and $\boldsymbol{G}:=\boldsymbol{G}(\boldsymbol{q})\in\mathbb{R}^n$ represent the unknown inertia matrix, Coriolis/centripetal matrix, friction vector, and gravity vector, respectively. The function $\mathrm{sat}(\cdot):\mathbb{R}^n\to\mathbb{R}^n$ represents the inherent physical limitations of the actuators such that $\|\mathrm{sat}(\boldsymbol{\tau})\|\le\overline{\tau}$, where $\overline{\tau}\in\mathbb{R}_{>0}$ denotes the maximum norm of control input.

In this study, the saturation function $\mathrm{sat}(\cdot)$ is assumed to be convex, which is practically common in many engineering systems. This is because actuator limits are typically convex—for example, minimum and maximum torque constraints—which define box- or ball-shaped bounds in the input space. To account for these limitations, it is essential to incorporate physically motivated constraints into the controller design. Appendix A introduces candidate constraints that can be applied to ensure compliance with these physical limitations.

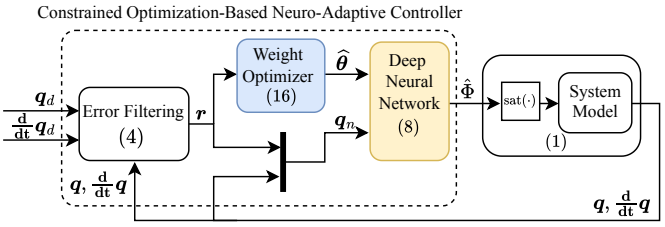



Fig. 1: Architecture of the proposed constrained optimization-based neuro-adaptive controller (CONAC).

The Euler-Lagrange system (1) satisfies several important physical properties, as presented in [4, Chap. 3, Tab. 3.2.1]. The key properties are introduced below:

**Property 1.** *The inertia matrix $\boldsymbol{M}$ is symmetric, positive definite, and bounded.*

**Property 2.** *The Coriolis/centripetal matrix $\boldsymbol{V}_m$ can always be selected, so that the matrix $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M}-2\boldsymbol{V}_m$ is skew-symmetric, i.e., $\boldsymbol{x}^\top(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M}-2\boldsymbol{V}_m)\boldsymbol{x}=0,\ \forall\boldsymbol{x}\in\mathbb{R}^n$.*

**Property 3.** *The disturbance $\boldsymbol{\tau}_d$ is bounded, i.e., $\|\boldsymbol{\tau}_d\|\le\overline{\tau}_d$, where $\overline{\tau}_d\in\mathbb{R}_{\ge0}$.*

The control objective is to develop a NAC that enables $\boldsymbol{q}$ to track a continuously differentiable desired trajectory $\boldsymbol{q}_d:=\boldsymbol{q}_d(t):\mathbb{R}_{\ge0}\to\mathbb{R}^n$, compensating for the unknown system dynamics while addressing the imposed constraints. The desired trajectory $\boldsymbol{q}_d(t)$ is supposed to satisfy the following assumption:

**Assumption 1.** *The desired trajectory $\boldsymbol{q}_d(t)$ is assumed to be bounded, i.e., $\|\boldsymbol{q}_d(t)\|\le\overline{q}_d\in\mathbb{R}_{>0}$, and available for designing a bounded control input.*

## III. Control Law Development

The architecture of the proposed CONAC consists of a DNN that functions as a NAC and a weight optimizer for the DNN, as illustrated in Fig. 1. Before proceeding to the controller design, some mathematical preliminaries are revisited in Section III-A. Section III-B introduces the NAC, followed by the DNN model in Section III-C. The constrained optimization-based weight adaptation law is presented in Section IV.

### A. Mathematical Preliminaries

The following proposition and lemma will be used in subsequent sections:

**Proposition 1** (see [1, Chap. 7, Prop. 7.1.9])**.** *For a matrix $\boldsymbol{A}\in\mathbb{R}^{n\times m}$ and a vector $\boldsymbol{b}\in\mathbb{R}^n$, the following property holds:*

$$\boldsymbol{A}^\top\boldsymbol{b}=\mathrm{vec}(\boldsymbol{A}^\top\boldsymbol{b})=\mathrm{vec}(\boldsymbol{b}^\top\boldsymbol{A})=(\boldsymbol{I}_m\otimes\boldsymbol{b}^\top)\,\mathrm{vec}(\boldsymbol{A}). \tag{2}$$

**Lemma 1.** *For a Lyapunov function $V := V(\boldsymbol{x}) \geq 0$ with $\boldsymbol{x} \in \mathbb{R}^n$, if the time derivative of $V$ is given by $\frac{\mathrm{d}}{\mathrm{d}t}V \leq -a_1\boldsymbol{x}^\top\boldsymbol{x} + a_2\boldsymbol{x}$ for some positive constants $a_1 \in \mathbb{R}_{>0}$ and $a_2 \in \mathbb{R}_{>0}$, then $\boldsymbol{x}$ is bounded as $\boldsymbol{x} \in \{\boldsymbol{x} \mid \|\boldsymbol{x}\| \leq \frac{a_2}{a_1}\}$.*

*Proof.* The time derivative of $V$ can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}V \leq -a_1\|\boldsymbol{x}\|^2 + a_2\|\boldsymbol{x}\| = -a_1\left(\|\boldsymbol{x}\| - \frac{a_2}{2a_1}\right)^2 + \frac{a_2^2}{4a_1}. \quad (3)$$

According to (3), it can be concluded that $\frac{\mathrm{d}}{\mathrm{d}t}V$ is negative definite if $\|\boldsymbol{x}\| > \frac{a_2}{a_1}$ holds. In other words, $\boldsymbol{x}$ remains within the bounded set $\{\boldsymbol{x} \mid \|\boldsymbol{x}\| \leq \frac{a_2}{a_1}\}$, since $\frac{\mathrm{d}}{\mathrm{d}t}V$ is negative definite when $\boldsymbol{x}$ is outside the set. □

### B. Neuro-Adaptive Controller Design

Given that the Euler-Lagrange system (1) exhibits second-order dynamics, a filtered error $\boldsymbol{r} \in \mathbb{R}^n$ is introduced to convert the system into a first-order system, as follows:

$$\boldsymbol{r} := \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e} + \boldsymbol{\Lambda}\boldsymbol{e}, \quad (4)$$

where $\boldsymbol{e} := \boldsymbol{q} - \boldsymbol{q}_d$ denotes the tracking error, $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e} := \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q} - \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_d$ represents the time derivative of the tracking error, and $\boldsymbol{\Lambda} \in \mathbb{R}_{>0}^{n\times n}$ is a user-designed filtering matrix. Since (4) is stable system, it implies that $\boldsymbol{e}$ is bounded if $\boldsymbol{r}$ is bounded.

Using $\boldsymbol{r}$, the system dynamics (1) can be rewritten as

$$\boldsymbol{M}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r} = -\boldsymbol{V}_m\boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \mathrm{sat}(\boldsymbol{\tau}), \quad (5)$$

where $\boldsymbol{K} = \boldsymbol{K}^\top \in \mathbb{R}_{>0}^{n\times n}$ denotes an arbitrary unknown matrix and $\boldsymbol{f} := \boldsymbol{f}\left(\boldsymbol{q}, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}, \boldsymbol{q}_d, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_d, \frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{q}_d\right) = \boldsymbol{K}\boldsymbol{r} + \boldsymbol{M}\left(-\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{q}_d + \boldsymbol{\Lambda}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e}\right) + \boldsymbol{V}_m\left(-\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_d + \boldsymbol{\Lambda}\boldsymbol{e}\right) - \boldsymbol{F} - \boldsymbol{G} \in \mathbb{R}^n$ denotes the lumped unknown system.

Consider the Lyapunov function $V_1 := \frac{1}{2}\boldsymbol{r}^\top\boldsymbol{M}\boldsymbol{r}$. Invoking Property 2, the time derivative of $V_1$ is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V_1 =& \boldsymbol{r}^\top\boldsymbol{M}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r} + \frac{1}{2}\boldsymbol{r}^\top\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M}\boldsymbol{r}\\ =& \boldsymbol{r}^\top\left(-\boldsymbol{V}_m\boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \mathrm{sat}(\boldsymbol{\tau})\right)\\ &+ \frac{1}{2}\boldsymbol{r}^\top\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M}\boldsymbol{r}\\ =& -\boldsymbol{r}^\top\boldsymbol{K}\boldsymbol{r} + \boldsymbol{r}^\top\left(\boldsymbol{f} - \boldsymbol{\tau}_d + \mathrm{sat}(\boldsymbol{\tau})\right)\\ &+ \frac{1}{2}\boldsymbol{r}^\top\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M} - 2\boldsymbol{V}_m\right)\boldsymbol{r}\\ \leq& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\boldsymbol{f} + \mathrm{sat}(\boldsymbol{\tau})).\end{aligned} \quad (6)$$

Therefore, invoking Lemma 1, it can be concluded that the filtered error $\boldsymbol{r}$ is uniformly ultimately bounded with $\lim_{t\to\infty}\|\boldsymbol{r}\| \leq \frac{\overline{\tau}_d}{\lambda_{\min}(\boldsymbol{K})}$, provided that the lumped unknown system $\boldsymbol{f}$ can be perfectly cancelled by the control input $\boldsymbol{\tau}$, neglecting the effect of input saturation. Hence, the ideal control input $\boldsymbol{\tau}^*$ can be designed as $-\boldsymbol{f}$. However, $\boldsymbol{\tau}^*$ is not available in practice, since the system dynamics encapsulated in $\boldsymbol{f}$ are unknown.

To overcome this issue, a DNN is employed to approximate $\boldsymbol{\tau}^*$. Let $\boldsymbol{\Phi} := \boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}) : \mathbb{R}^{l_0+1} \times \mathbb{R}^{\Xi} \to \mathbb{R}^n$ represent the DNN, where $\boldsymbol{q}_n \in \mathbb{R}^{l_0+1}$ is the DNN input vector, and $\boldsymbol{\theta} \in \mathbb{R}^{\Xi}$ is the vector of trainable weights. The architecture of DNN $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ will be presented in Section III-C, later. According to the universal approximation theorem for DNNs [32], $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ can approximate a nonlinear function, denoted as $\boldsymbol{g}(\cdot)$, with an ideal weight vector $\boldsymbol{\theta}^* \in \mathbb{R}^{\Xi}$ over a compact subset $\Omega_n \in \mathbb{R}^{l_0+1}$ to within $\epsilon$-accuracy, such that $\sup_{\boldsymbol{q}_n\in\Omega_n}\|\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}^*) - \boldsymbol{g}(\cdot)\| = \epsilon < \infty$. The compact set $\Omega_n$ is defined as the set of all feasible states $\boldsymbol{q}, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}$ and filtered errors $\boldsymbol{r}$. In this study, the ideal weight vector $\boldsymbol{\theta}^*$ is assumed to be bounded by Assumption 1 and [6, Assum. 1], and considered a locally optimal solution, rather than a global one.

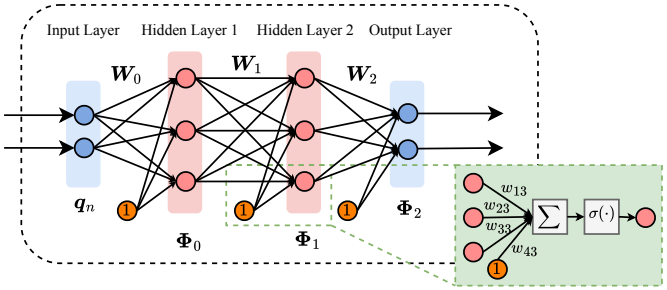



Fig. 2: Architecture of the DNN $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ with $k = 2, l_0 = 2, l_1 = l_2 = 3$, and $l_3 = 2$.

Accordingly, the ideal control input $\boldsymbol{\tau}^*$ can be approximated by the DNN with the ideal weight vector $\boldsymbol{\Phi}^* := \boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}^*)$ as follows:

$$\boldsymbol{\tau}^* = \boldsymbol{\Phi}^* + \boldsymbol{\epsilon}, \quad (7)$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is the approximation error vector, bounded by $\|\boldsymbol{\epsilon}\| \leq \overline{\epsilon}$ for some $\overline{\epsilon} \in \mathbb{R}_{>0}$. The ideal control input $\boldsymbol{\tau}^*$ is estimated online by

$$\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}}, \quad (8)$$

where $\hat{\boldsymbol{\Phi}} := \boldsymbol{\Phi}(\boldsymbol{q}_n;\widehat{\boldsymbol{\theta}})$, and $\widehat{\boldsymbol{\theta}} \in \mathbb{R}^{\Xi}$ is the estimated weight vector for the ideal weight vector $\boldsymbol{\theta}^*$.

Substituting (7) and (8) into (6), the time derivative of $V_1$ can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}V_1 \leq -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top\left(-\boldsymbol{\Phi}^* - \boldsymbol{\epsilon} + \mathrm{sat}(\hat{\boldsymbol{\Phi}})\right). \quad (9)$$

Therefore, it is concluded that the filtered error $\boldsymbol{r}$ can be stabilized by adapting $\widehat{\boldsymbol{\theta}}$ to $\boldsymbol{\theta}^*$, i.e., $\hat{\boldsymbol{\Phi}} \to \boldsymbol{\Phi}^*$, neglecting the effect of input saturation. Note that the control input saturation has not yet been considered in the derivation of the ideal control input; this will be addressed in the subsequent section.

### C. Deep Neural Network (DNN) Model

The DNN architecture $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}) = \boldsymbol{\Phi}_k$ can be recursively represented, as follows:

$$\boldsymbol{\Phi}_i := \begin{cases}\boldsymbol{W}_i^\top\boldsymbol{\phi}_i(\boldsymbol{\Phi}_{i-1}), & i \in \{1,\ldots,k\},\\ \boldsymbol{W}_0^\top\boldsymbol{q}_n, & i = 0,\end{cases} \quad (10)$$

where $\boldsymbol{W}_i = [w_{ij}]_{i\in\{1,\cdots,l_i+1\},j\in\{1,\cdots,l_{i+1}\}} \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$ and $\boldsymbol{\phi}_i : \mathbb{R}^{l_i} \to \mathbb{R}^{l_i+1}$ represent the weight matrix and the activation function of the $i$th layer, respectively. The DNN $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ has $k+1$ layers, where the input layer is indexed by $i = 0$, and the output layer is indexed by $i = k$. For instance, in Fig. 2, the DNN $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ with $k = 2, l_0 = 2, l_1 = l_2 = 3$,

and $l_3 = 2$ is illustrated. Notice that the output size of $\boldsymbol{\Phi}(\cdot)$ is the same as that of the control input $\boldsymbol{\tau}$, i.e., $l_{k+1} = n$.

The DNN input vector $\boldsymbol{q}_n$ is defined as $\boldsymbol{q}_n = (\boldsymbol{q}^\top, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}^\top, \boldsymbol{r}^\top, 1)^\top \in \mathbb{R}^{3n+1}$ to include the current state of the system, the filtered error, and the augmentation term 1 to account for the bias term in the input weight matrix $\boldsymbol{W}_0$. The activation function is defined as $\boldsymbol{\phi}_i(\boldsymbol{x}) = (\sigma(x_1), \sigma(x_2), \cdots, \sigma(x_{l_i}), 1)^\top,\ \forall \boldsymbol{x} \in \mathbb{R}^{l_i}$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear function, and the augmentation of 1 is used to account for bias terms in the weight matrices, similar to the input vector $\boldsymbol{q}_n$. The weights $\boldsymbol{W}_i,\ \forall i \in \{0, \cdots, k\}$, are initialized randomly using a uniform distribution at the beginning of the control process.

In selecting the nonlinear function $\sigma(\cdot)$, the ReLU function [33] is one of the most widely used choices for large DNNs, since it effectively mitigates the gradient vanishing problem during error backpropagation. However, for control applications, where relatively shallow DNNs are typically sufficient and the gradient vanishing issue is less severe, the sigmoid function or the hyperbolic tangent function is commonly used as the nonlinear function $\sigma(\cdot)$. These functions simplify stability analysis, since they are continuously differentiable and their outputs and gradients are bounded. In this study, the hyperbolic tangent function $\tanh(\cdot)$ was selected as the nonlinear function, i.e., $\sigma(\cdot) = \tanh(\cdot)$, which provides desirable boundedness with $\|\sigma(x)\| < 1$ and $0 < \|\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}\| \leq 1$.

For simplicity, each layer's weights are vectorized as $\boldsymbol{\theta}_i := \mathrm{vec}(\boldsymbol{W}_i) \in \mathbb{R}^{\Xi_i},\ \forall i \in \{0, \cdots, k\}$, where $\Xi_i := (l_i+1)l_{i+1}$ is the number of weights in the $i$th layer. The total weight vector $\boldsymbol{\theta} \in \mathbb{R}^{\Xi}$ is defined by augmenting $\boldsymbol{\theta}_i, \forall i \in \{0, \cdots, k\}$, as

$$\boldsymbol{\theta} := \begin{pmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{\theta}_{k-1} \\ \vdots \\ \boldsymbol{\theta}_0 \end{pmatrix} = \begin{pmatrix} \mathrm{vec}(\boldsymbol{W}_k) \\ \mathrm{vec}(\boldsymbol{W}_{k-1}) \\ \vdots \\ \mathrm{vec}(\boldsymbol{W}_0) \end{pmatrix}, \tag{11}$$

where $\Xi := \sum_{i=0}^{k} \Xi_i$ represents the total number of weights in the DNN.

The gradient of $\boldsymbol{\Phi}(\boldsymbol{q}_n; \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ can be obtained using Proposition 1 and the chain rule as follows:

$$\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}_k} & \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}_{k-1}} & \cdots & \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}_0} \end{bmatrix} \in \mathbb{R}^{n \times \Xi}, \tag{12}$$

where

$$\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}_i} = \begin{cases} (\boldsymbol{I}_{l_{k+1}} \otimes \boldsymbol{\phi}_k^\top), & i = k, \\ \boldsymbol{W}_k^\top \boldsymbol{\phi}_k'(\boldsymbol{I}_{l_k} \otimes \boldsymbol{\phi}_{k-1}^\top), & i = k-1, \\ \vdots & \vdots \\ \boldsymbol{W}_k^\top \boldsymbol{\phi}_k' \cdots \boldsymbol{W}_1^\top \boldsymbol{\phi}_1'(\boldsymbol{I}_{l_1} \otimes \boldsymbol{q}_n^\top), & i = 0, \end{cases} \tag{13}$$

and $\boldsymbol{\phi}_i := \boldsymbol{\phi}_i(\boldsymbol{\Phi}_{i-1})$ and $\boldsymbol{\phi}_i' := \frac{\mathrm{d}\boldsymbol{\phi}_i}{\mathrm{d}\boldsymbol{\Phi}_{i-1}}$.

In the following sections, let $\boldsymbol{\Phi}_i^*$ represent the output of the $i$th layer with the ideal weight vector $\boldsymbol{\theta}^*$. Define $\boldsymbol{\phi}_i^* = \boldsymbol{\phi}_i(\boldsymbol{\Phi}_{i-1}^*)$ and $\boldsymbol{\phi}^{*\prime}{}_i = \frac{\mathrm{d}\boldsymbol{\phi}_i^*}{\mathrm{d}\boldsymbol{\Phi}_{i-1}^*}$. Similarly, let $\widehat{\boldsymbol{\Phi}}_i$ denote the output of the $i$th layer with the estimated weight vector $\widehat{\boldsymbol{\theta}}$, and define $\widehat{\boldsymbol{\phi}}_i = \boldsymbol{\phi}_i(\widehat{\boldsymbol{\Phi}}_{i-1})$ and $\widehat{\boldsymbol{\phi}}_i' = \frac{\mathrm{d}\widehat{\boldsymbol{\phi}}_i}{\mathrm{d}\widehat{\boldsymbol{\Phi}}_{i-1}}$.

## IV. Weight Adaptation Laws

### A. Weight Optimizer Design

The control objective can be represented as follows:

$$\begin{aligned} &\min_{\widehat{\boldsymbol{\theta}}}\ J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}}) := \tfrac{1}{2}\boldsymbol{r}^\top \boldsymbol{r}, \\ &\text{subject to } c_j(\widehat{\boldsymbol{\theta}}) \leq 0, \quad \forall j \in \mathcal{I}, \end{aligned} \tag{14}$$

where $J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}})$ is the objective function. Inequality constraints $c_j := c_j(\widehat{\boldsymbol{\theta}}),\ \forall j \in \mathcal{I}$, are imposed during the weight adaptation process, where $\mathcal{I}$ denotes the set of the imposed inequality constraints. The Lagrangian function $L(\cdot)$ is defined as

$$L(\boldsymbol{r}, \widehat{\boldsymbol{\theta}}, [\lambda_j]_{j\in\mathcal{I}}) = J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}}) + \textstyle\sum_{j\in\mathcal{I}} \lambda_j c_j(\widehat{\boldsymbol{\theta}}), \tag{15}$$

where $\lambda_j,\ \forall j \in \mathcal{I}$, denotes the Lagrange multiplier for each constraint.

The adaptation laws for $\widehat{\boldsymbol{\theta}}$ and $[\lambda_j]_{j\in\mathcal{I}}$ are derived to solve the dual problem of (14), i.e., $\min_{\widehat{\boldsymbol{\theta}}} \max_{[\lambda_j]_{j\in\mathcal{I}}} L(\boldsymbol{r}, \widehat{\boldsymbol{\theta}}, [\lambda_j]_{j\in\mathcal{I}})$, as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} = -\alpha \frac{\partial L}{\partial \widehat{\boldsymbol{\theta}}} = -\alpha \left( \frac{\partial J}{\partial \widehat{\boldsymbol{\theta}}} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right), \tag{16a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\lambda_j = \beta_j \frac{\partial L}{\partial \lambda_j} = \beta_j c_j, \quad \forall j \in \mathcal{I}, \tag{16b}$$

$$\lambda_j \leftarrow \max(\lambda_j, 0), \tag{16c}$$

where $\alpha \in \mathbb{R}_{>0}$ denotes the adaptation gain (learning rate) and $\beta_j \in \mathbb{R}_{>0}$ denotes the update rate of the Lagrange multipliers in $\mathcal{I}$, and the arguments of $L(\cdot)$ and $J(\cdot)$ are suppressed for brevity. The Lagrange multipliers associated with inequality constraints are non-negative, i.e., $\lambda_j \geq 0$, due to (16c). When a constraint $c_j$ becomes active, i.e., $c_j > 0$, the corresponding Lagrange multiplier $\lambda_j$ increases, see (16b), to address the violation, see (16a). Once the violation is resolved and the constraint is no longer active, i.e., $c_j \leq 0$, the Lagrange multiplier $\lambda_j$ decreases gradually until it returns to zero, see (16c).

At steady state, where $\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} = 0$ and $\frac{\mathrm{d}}{\mathrm{d}t}\lambda_j = 0$, the KKT conditions are satisfied, i.e., $\frac{\partial L}{\partial \widehat{\boldsymbol{\theta}}} = 0$, $c_j \leq 0$, $\lambda_j \geq 0$, and $\lambda_j c_j = 0$. In other words, the proposed weight optimizer updates $\widehat{\boldsymbol{\theta}}$ and $\lambda_j$ in a way that satisfies the KKT conditions. These conditions represent the first-order necessary conditions for optimality, guiding the updates toward candidates for a locally optimal point.

### B. Approximation of the Gradient of Objective Function

The adaptation law for $\widehat{\boldsymbol{\theta}}$ defined in (16a) involves the partial derivative of the filtered error with respect to control input $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$, i.e., $\frac{\partial J}{\partial \widehat{\boldsymbol{\theta}}} = \left( (\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}})(\frac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\theta}}}) \right)^\top \boldsymbol{r} = \left( (\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}})(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}) \right)^\top \boldsymbol{r}$. Since the objective function depends on $\boldsymbol{r}$ of a dynamic system, obtaining $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ is not straightforward. The recommended method to calculate the exact value of $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ is to use the forward sensitivity method [9], [34] by simulating the sensitivity equation as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}\right) = \frac{\partial}{\partial \boldsymbol{\tau}}\left( \boldsymbol{M}^{-1}(-\boldsymbol{V}_m \boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \mathrm{sat}(\boldsymbol{\tau})) \right). \tag{17}$$

However, this cannot be realized, since we do not know the exact system dynamics, i.e., $\boldsymbol{M}, \boldsymbol{V}_m, \boldsymbol{F}$, and $\boldsymbol{G}$. Additionally,

the computational cost of the forward sensitivity method is high, as the number of DNN weights is generally large.

In [35], [36], the authors approximate $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ by taking the sign of each entry, i.e., $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}} \approx \left[\text{sign}\left(\frac{\partial r_i}{\partial \tau_j}\right)\right]_{i,j\in\{1,\cdots,n\}}$, assuming known control directions. However, this approach is not applicable to (5), where control directions are unknown. Instead, since the sign of the control input matrix is known—owing to the positive definiteness of $\boldsymbol{M}^{-1}$, see Property 1—we approximate $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}} \approx \boldsymbol{I}_n$. Consequently, the adaptation law in (16a) becomes:

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} \approx -\alpha \left( \tfrac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}^\top \boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \tfrac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right). \tag{18}$$

### C. Essential and Potential Constraint Candidates

This section introduces essential constraints required to ensure the stability of the proposed CONAC, as well as conditions for potential input constraints.

First, weight norm constraints $c_{\theta_i}(\widehat{\boldsymbol{\theta}}),\ \forall i \in \{0,\cdots,k\}$, defined in (19), are essential to prevent the weights from diverging during the adaptation process:

$$c_{\theta_i} = \tfrac{1}{2}\left(\|\widehat{\boldsymbol{\theta}}_i\|^2 - \overline{\theta}_i^2\right), \tag{19}$$

with $\overline{\theta}_i \in \mathbb{R}_{>0}$ denoting the maximum allowable norm for $\widehat{\boldsymbol{\theta}}_i$, and the arguments of $c_{\theta_i}(\cdot)$ are omitted for brevity. The gradient of $c_{\theta_i}$ with respect to $\widehat{\boldsymbol{\theta}}$ can be obtained by accumulating the gradients of each layer as follows:

$$\tfrac{\partial c_{\theta_i}}{\partial \widehat{\boldsymbol{\theta}}_j} = \begin{cases} \widehat{\boldsymbol{\theta}}_i, & \text{if } i = j, \\ \boldsymbol{0}, & \text{if } i \neq j. \end{cases} \tag{20}$$

Next, we present the following assumptions, which specify the conditions for the potential input constraints introduced in Appendix A:

**Assumption 2.** *The constraint function $c_j(\widehat{\boldsymbol{\theta}})$ is convex in the (input-) $\tau$-space and satisfies $c_j(0) \leq 0$ and $c_j(\boldsymbol{\theta}^*) \leq 0$.*

**Assumption 3.** *The selected constraints satisfy the Linear Independence Constraint Qualification (LICQ) [28, Chap. 12, Def. 12.1].*

**Remark 1.** *Assumption 2 is not restrictive, since, as aforementioned, the control input saturation is practically convex. Moreover, Assumption 3 is a standard assumption in optimization problems, ensuring that the gradients of the active constraints are linearly independent. This assumption will be used in the stability analysis (see Lemma 2).*

The algorithm of the proposed CONAC is presented by Algorithm 1, implemented in a discrete-time controller with a sampling time of $T_s$, where $(\cdot^{(k)})$ denotes the value at the $k$th time step.

## V. Stability Analysis

Before conducting the stability analysis, let us define the weight estimation error as $\tilde{\boldsymbol{\theta}} := [\tilde{\boldsymbol{\theta}}_i]_{i\in\{0,\cdots,k\}}$, where $\tilde{\boldsymbol{\theta}}_i := \widehat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i^*, \forall i \in \{0,\cdots,k\}$. The following lemmas are introduced for the stability analysis.

**Algorithm 1** Proposed CONAC

**Initialize:** DNN weights $\widehat{\boldsymbol{\theta}}^{(0)}$, Lagrange multipliers $\lambda_j^{(0)}$
**for** $k = 0$ to $\infty$ **do**
  Measure $\boldsymbol{q}^{(k)}$, $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}^{(k)}$
  Obtain $\boldsymbol{q}_d^{(k)}$, $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_d{}^{(k)}$ from given $\boldsymbol{q}_d(t)$
  $\boldsymbol{e}^{(k)} \leftarrow \boldsymbol{q}^{(k)} - \boldsymbol{q}_d^{(k)}$ and $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e}^{(k)} \leftarrow \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}^{(k)} - \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_d{}^{(k)}$
  $\boldsymbol{r}^{(k)} \leftarrow \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{e}^{(k)} + \boldsymbol{\Lambda}\boldsymbol{e}^{(k)}$ based on (4)
  *// Neuro-adaptive control law (Section III)*
  $\boldsymbol{q}_n^{(k)} \leftarrow \left(\boldsymbol{q}^{(k)\top}, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}^{(k)\top}, \boldsymbol{r}^{(k)\top}, 1\right)^\top$
  $\boldsymbol{\tau}^{(k)} \leftarrow \boldsymbol{\Phi}(\boldsymbol{q}_n^{(k)}; \widehat{\boldsymbol{\theta}}^{(k)})$ based on (8)
  Apply $\boldsymbol{\tau}^{(k)}$ to the system in (1)
  *// Train DNN (Section IV)*
  Compute $\frac{\partial \boldsymbol{\Phi}}{\partial \widehat{\boldsymbol{\theta}}^{(k)}}$ as defined in (12), $c_j$ and $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}^{(k)}}$ for all $j \in \mathcal{I}$
  $\widehat{\boldsymbol{\theta}}^{(k+1)} \leftarrow \widehat{\boldsymbol{\theta}}^{(k)} - \alpha\left(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}^{(k)}}^\top \boldsymbol{r}^{(k)} + \sum_{j\in\mathcal{I}} \lambda_j^{(k)} \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}^{(k)}}\right) \cdot T_s$ based on (16a)
  $\lambda_j^{(k+1)} \leftarrow \lambda_j^{(k)} + \beta_j c_j \cdot T_s$ for all $j \in \mathcal{I}$ based on (16b)
  $\lambda_j^{(k+1)} \leftarrow \max(\lambda_j^{(k+1)}, 0)$ for all $j \in \mathcal{I}$ from (16c)
**end for**

**Lemma 2.** *If Assumptions 2 and 3 hold, then for each active constraint $c_j$, the angle between the output layer's weight vector $\widehat{\boldsymbol{\theta}}_k$ and $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}$ is positive; that is $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}^\top \widehat{\boldsymbol{\theta}}_k > 0$.*

*Proof.* Since $\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}}$, a linear mapping $\boldsymbol{T}(\cdot) : \widehat{\boldsymbol{\theta}}_k \to \boldsymbol{\tau}$, which is independent of $\widehat{\boldsymbol{\theta}}_k$, can be derived using Proposition 1:

$$\begin{aligned} \boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}} &= \text{vec}(\widehat{\boldsymbol{W}}_k^\top \widehat{\boldsymbol{\phi}}_k) = (\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top \text{vec}(\widehat{\boldsymbol{W}}_k) \\ &= \underbrace{(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top}_{=:\boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)} \widehat{\boldsymbol{\theta}}_k = \boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)\widehat{\boldsymbol{\theta}}_k. \end{aligned} \tag{21}$$

Therefore, the convexity of the input constraints in the $\boldsymbol{\tau}$-space (see Assumption 2) is preserved in $\widehat{\boldsymbol{\theta}}_k$-space, implying that $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}^\top \widehat{\boldsymbol{\theta}}_k > 0$.

□

**Lemma 3.** *If input constraint $c_j(\widehat{\boldsymbol{\theta}}),\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$, satisfies Assumption 2, then $\|\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i}\|,\ \forall i \in \{k-1,\cdots,0\}$, is bounded, provided the norms of $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{k,\cdots,i+1\}$, remain bounded.*

For instance, if $k = 3$ and $i = 1$, $\|\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_1}\|$ is bounded, provided that $\|\widehat{\boldsymbol{\theta}}_3\|$ and $\|\widehat{\boldsymbol{\theta}}_2\|$ are bounded.

*Proof.* The gradient of $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$, with respect to $\widehat{\boldsymbol{\theta}}_i$ is represented as

$$\tfrac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i} = \tfrac{\partial c_j}{\partial \boldsymbol{\tau}} \tfrac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\Phi}}} \tfrac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i} = \tfrac{\partial c_j}{\partial \boldsymbol{\tau}} \boldsymbol{I}_n \tfrac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}. \tag{22}$$

The boundedness of the first term $\frac{\partial c_j}{\partial \boldsymbol{\tau}}$ is guaranteed, if the input argument $\boldsymbol{\tau}$ is bounded owing to the convexity of $c_j$ by Assumption 2. In addition, the boundedness of $\boldsymbol{\tau}$ can be ensured by the boundedness of $\widehat{\boldsymbol{\theta}}_k$, since $\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}} = (\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)\widehat{\boldsymbol{\theta}}_k$ and $\|\widehat{\boldsymbol{\phi}}_k\|$ is bounded due to the properties of the activation

functions. The third term, $\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}$ is bounded, provided that the norms of $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{k, \cdots, i+1\}$, are bounded. This can be verified by using the definition of $\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}$ given in (13). Consequently, the boundedness of $\|\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i}\|,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$, depends on the boundedness of $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{k, \cdots, i+1\}$. □

The following theorem states that $\boldsymbol{r}$ and $\widehat{\boldsymbol{\theta}}$ are bounded.

**Theorem 1.** *For the dynamical system described in* (1)*, the NAC in* (8) *with the weight adaptation laws in* (16) *ensure the boundedness of the filtered error $\boldsymbol{r}$ and the weight estimate $\widehat{\boldsymbol{\theta}}$, under the input constraints satisfying Assumption 2 and 3, provided that the weight norm constraints* (19) *are imposed.*

*Proof.* Invoking the fact that the amplitude of the control input depends only on $\widehat{\boldsymbol{\theta}}_k$, i.e., $\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}} = (\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)\widehat{\boldsymbol{\theta}}_k$ and $\|\widehat{\boldsymbol{\phi}}_k\|$ is bounded, the boundedness of inner layers' weights will be established after proving the boundedness of $\boldsymbol{r}$ and $\widehat{\boldsymbol{\theta}}_k$. In addition, without loss of generality, weight norm constraints $c_{\theta_i},\ \forall i \in \{0, \cdots, k\}$, are supposed to be active. This assumption is justified because, even if $c_{\theta_i}$ is inactive, $\widehat{\boldsymbol{\theta}}$ is adapted to minimize the objective function $J$ as long as the constraint is inactive. We consider two cases: (1) the control input is saturated, and (2) the control input is not saturated.

*Case 1: Control input is saturated.*

As a result of the time derivative of $V_1$ in (9), the invariant set $\Theta^1_{\boldsymbol{r}}$ of $\boldsymbol{r}$, i.e., if $\boldsymbol{r}$ leaves $\Theta^1_{\boldsymbol{r}}$, $\frac{\mathrm{d}}{\mathrm{d}t}V_1$ is negative, can be obtained using Lemma 1 as follows:

$$\Theta^1_{\boldsymbol{r}} = \left\{ \boldsymbol{r} \in \mathbb{R}^n \mid \|\boldsymbol{r}\| \leq \tfrac{\overline{\tau}_d + \overline{\tau} + \overline{\theta}^*_k\sqrt{l_k+1}+\overline{\epsilon}}{\lambda_{\min}(\boldsymbol{K})} \right\}. \tag{23}$$

It is notable that the ideal DNN $\boldsymbol{\Phi}^*_k = {\boldsymbol{W}^*_k}^\top \boldsymbol{\phi}^*_k$ is bounded by $\overline{\theta}^*_k\sqrt{l_k+1}$. This is because, the ideal weight $\boldsymbol{\theta}^*_k$ is bounded by Assumption 1 such that $\|\boldsymbol{\theta}^*_k\| \leq \overline{\theta}^*_k$ for some positive constant $\overline{\theta}^*_k$. Hence, invoking $\|\boldsymbol{W}^*_k\|_F = \|\boldsymbol{\theta}^*_k\| \leq \overline{\theta}^*_k$ and $\|\boldsymbol{\phi}^*_k\| \leq \sqrt{l_k+1}$, we have $\|\boldsymbol{\Phi}^*\| \leq \overline{\theta}^*_k\sqrt{l_k+1}$.

To investigate the boundedness of $\widehat{\boldsymbol{\theta}}_k$, consider the Lyapunov function candidate $V_2 := \frac{1}{2\alpha}\widehat{\boldsymbol{\theta}}_k^\top\widehat{\boldsymbol{\theta}}_k$. Taking the time derivative of $V_2$ yields:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_2 =& -\widehat{\boldsymbol{\theta}}_k^\top\left((\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top\boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}}\lambda_j\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}\right)\\
=& -\widehat{\boldsymbol{\theta}}_k^\top\Big((\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top\boldsymbol{r} + \lambda_{\theta_k}\widehat{\boldsymbol{\theta}}_k\\
&+ \textstyle\sum_{j\in\mathcal{I}\setminus\{\theta_i\}_{i\in\{0,\cdots,k\}}}\lambda_j\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}\Big)\\
\leq& -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + \underbrace{\|(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)\|}_{=:\gamma_1\in\mathbb{R}_{\geq 0}}\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\| \qquad (24)\\
&- \underbrace{\textstyle\sum_{j\in\mathcal{I}\setminus\{\theta_i\}_{i\in\{0,\cdots,k\}}}\lambda_j\widehat{\boldsymbol{\theta}}_k^\top\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}}_{=:\gamma_2\in\mathbb{R}_{\geq 0},\ \text{by Lemma 2 and Assumption 3}}\\
\leq& -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + \gamma_1\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\|.
\end{aligned}$$

According to (24) and Lemma 1, the invariant set of $\widehat{\boldsymbol{\theta}}_k$ is defined as

$$\Theta^1_{\theta_k} = \left\{ \widehat{\boldsymbol{\theta}}_k \in \mathbb{R}^{\Xi_k} \mid \|\widehat{\boldsymbol{\theta}}_k\| \leq \tfrac{\gamma_1(\overline{\tau}_d + \overline{\tau} + \overline{\theta}^*_k\sqrt{l_k+1}+\overline{\epsilon})}{\lambda_{\theta_k}\lambda_{\min}(\boldsymbol{K})} \right\}. \tag{25}$$

The satisfaction of the constraints, i.e., $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k-1\}}$, can be verified through the boundedness of $\widehat{\boldsymbol{\theta}}_k$. For the output layer's weight norm constraint $c_{\theta_k}$, if the corresponding Lagrange multiplier $\lambda_{\theta_k}$ increases sufficiently large due to the constraint violation, $\widehat{\boldsymbol{\theta}}_k$ approaches toward the origin as the invariant set $\Theta^1_{\theta_k}$ shrinks to a point. The remaining input constraints can be validated implicitly through the term $\gamma_2$ in (24). Similarly to $c_{\theta_k}$, when an input constraint $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$, is violated, the corresponding Lagrange multiplier $\lambda_j$ increases sufficiently to a large value, leading to an increase in $\gamma_2$. It makes $\gamma_2$ dominate the right-hand side of (24), which makes $\frac{\mathrm{d}}{\mathrm{d}t}V_2$ negative definite. This drives $\widehat{\boldsymbol{\theta}}_k$ toward the origin until the constraint is satisfied.

Therefore, the boundedness of $\boldsymbol{r}$ and $\widehat{\boldsymbol{\theta}}_k$ is guaranteed by the invariant sets $\Theta^1_{\boldsymbol{r}}$ and $\Theta^1_{\theta_k}$, and the satisfaction of $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k-1\}}$, is ensured.

*Case 2: Control input is not saturated.*

Since the input constraints are inactive, the saturation function $\mathrm{sat}(\cdot)$ in (9) and the input constraint functions $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$, in (24) can be neglected. Consider the Lyapunov function candidate $V_3 := V_1 + V_2$, whose time derivative is given by:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_3 =& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\widehat{\boldsymbol{\Phi}} - \boldsymbol{\Phi}^* - \boldsymbol{\epsilon})\\
&- \widehat{\boldsymbol{\theta}}_k^\top\left((\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top\boldsymbol{r} + \lambda_{\theta_k}\widehat{\boldsymbol{\theta}}_k\right)\\
\leq& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \boldsymbol{r}^\top\widehat{\boldsymbol{\Phi}} + (\overline{\tau}_d + \|\boldsymbol{\Phi}^* + \boldsymbol{\epsilon}\|)\|\boldsymbol{r}\|\\
&- \widehat{\boldsymbol{\Phi}}^\top\boldsymbol{r} - \lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2\\
\leq& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \left(\overline{\tau}_d + \overline{\theta}^*_k\sqrt{l_k+1} + \overline{\epsilon}\right)\|\boldsymbol{r}\|\\
&- \lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2.
\end{aligned} \tag{26}$$

Based on the same reasoning as in Case 1, the boundedness of $\boldsymbol{r}$ is ensured by the invariant set:

$$\Theta^2_{\boldsymbol{r}} = \left\{ \boldsymbol{r} \in \mathbb{R}^n \mid \|\boldsymbol{r}\| \leq \tfrac{\overline{\tau}_d + \overline{\theta}^*_k\sqrt{l_k+1}+\overline{\epsilon}}{\lambda_{\min}(\boldsymbol{K})} \right\}. \tag{27}$$

The boundedness of $\widehat{\boldsymbol{\theta}}_k$ can also be ensured by the invariant set $\Theta^2_{\theta_k} = \{\widehat{\boldsymbol{\theta}}_k \in \mathbb{R}^{\Xi_k} \mid \|\widehat{\boldsymbol{\theta}}_k\| \leq \overline{\theta}_k\}$, as $\lambda_{\theta_k}$ remains positive unless the weight norm constraint $c_{\theta_k}$ is satisfied.

The boundedness of the inner layer weights and the satisfaction of their corresponding constraints $c_j,\ \forall j \in \{\theta_i\}_{i\in\{k-1,\cdots,0\}}$, can be established recursively using [37, Chap. 4, Thm. 1.9]. The dynamics of $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{k-1, \cdots, 0\}$, are represented as

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}}_i = -\alpha\left(\tfrac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}^\top\boldsymbol{r} + \lambda_{\theta_i}\widehat{\boldsymbol{\theta}}_i + \textstyle\sum_{j\in\mathcal{I}\setminus\{\theta_i\}_{i\in\{0,\cdots,k\}}}\lambda_j\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i}\right). \tag{28}$$

By Lemma 3, $\widehat{\boldsymbol{\theta}}_i$ is bounded, provided that $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{k, \cdots, i+1\}$, are bounded. This holds because the system matrix $-\lambda_{\theta_i}\boldsymbol{I}_{\Xi_i}$ is stable, and the residual terms—such as $\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}, \boldsymbol{r}$, and $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i}$ —are bounded. Moreover, the Lagrange multiplier $\lambda_j$ does not diverge since the input constraints $c_j,\ \forall j \in \mathcal{I} \setminus \{\theta_i\}_{i\in\{0,\cdots,k\}}$ are satisfied before divergence can occur. Therefore, starting from the $(k-1)$th layer, the

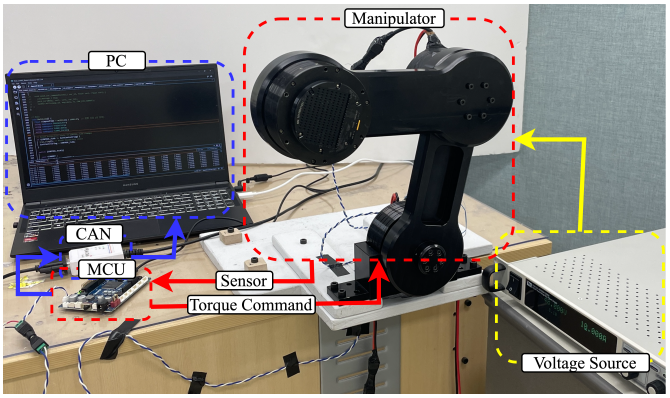



Fig. 3: Experimental setup of the two-link robotic manipulator used for real-time validation.

TABLE I: Two-link robotic manipulator's parameters.

| Symbol | Description | Value |
|---|---|---|
| $m_1, m_2$ | Mass | 2.465 kg |
| $l_1, l_2$ | Length | 0.2 m |
| $l_{c1}, l_{c2}$ | Center of mass | 0.139 m |
| $I_1, I_2$ | Inertia | 0.069 kg $\cdot$ m$^2$ |

boundedness of $\widehat{\boldsymbol{\theta}}_i$ can be established recursively down to the input layer ($i = 0$). Moreover the satisfaction of the weight norm constraints for the inner layers can be verified, as the Lagrange multipliers $\lambda_{\theta_i}, \forall i \in \{k-1, \cdots, 0\}$, increase sufficiently large to stabilize the $\widehat{\boldsymbol{\theta}}_i$ dynamics in (28), leading to its convergence toward the origin.

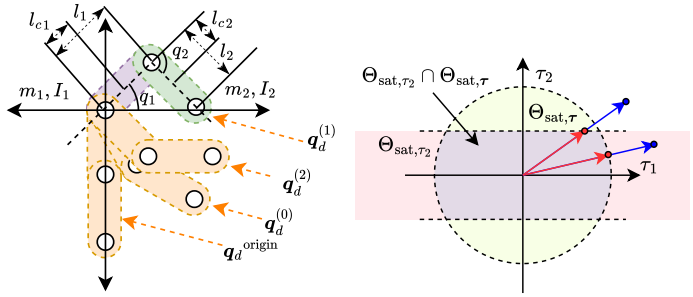
In conclusion, the filtered tracking error $\boldsymbol{r}$ and the weight estimate $\widehat{\boldsymbol{\theta}}$ are bounded, and all imposed constraints $c_j, \ \forall j \in \mathcal{I}$, are satisfied.

□

## VI. Real-Time Implementation and Validation

### A. Validation Setup

To validate the effectiveness of the proposed CONAC, a real-time experiment was conducted on a two-link robotic manipulator, as shown in Fig. 3. The OpenCR1.0 board [38] with a 32-bit ARM Cortex and a 216 MHz clock frequency was used for the real-time implementation. The two-link robotic manipulator was actuated by two Cubemars AK10-90 servo motors. The servos were controlled using the OpenCR1.0 board, which transmits the torque reference to the corresponding servo via controller area network (CAN) communication. The system dynamics of the two-link robotic manipulator can follow the Euler-Lagrange equation (1), where $\boldsymbol{q}$ denotes the joint angles and $\boldsymbol{\tau}$ denotes the control input (joint torques), and is illustrated in Fig. 4a. The measured and/or estimated system parameters are provided in Table I.




(a) Two-link robotic manipulator. (b) Control input saturation function.

Fig. 4: Two-link robotic manipulator and control saturation function.

The desired trajectory is designed with 4 segments as follows:

$$\boldsymbol{q}_d(t) = \begin{pmatrix} q_{d1} \\ q_{d2} \end{pmatrix} = \begin{cases} \mathcal{P}^5(t; \boldsymbol{q}_d^{(0)}, \boldsymbol{q}_d^{(1)}, t_f), & \text{if } 0 \le t < t_f, \\ \mathcal{P}^5(t; \boldsymbol{q}_d^{(1)}, \boldsymbol{q}_d^{(2)}, t_f), & \text{if } t_f \le t < 2t_f, \\ \mathcal{P}^5(t; \boldsymbol{q}_d^{(2)}, \boldsymbol{q}_d^{(1)}, t_f), & \text{if } 2t_f \le t < 3t_f, \\ \mathcal{P}^5(t; \boldsymbol{q}_d^{(1)}, \boldsymbol{q}_d^{(0)}, t_f), & \text{if } 3t_f \le t < 4t_f, \end{cases} \tag{29}$$

where $\mathcal{P}^5(t; \boldsymbol{x}_1, \boldsymbol{x}_2, t_f)$ denotes a quintic (5th-order) polynomial trajectory that moves the system from $\boldsymbol{x}_1 \in \mathbb{R}^2$ to $\boldsymbol{x}_2 \in \mathbb{R}^2$ in $t_f$ seconds, with zero initial and final velocities. The initial and final joint angles of the segments are defined as follows: $\boldsymbol{q}_d^{(0)} = \left(-\frac{1}{3}\pi, \frac{1}{3}\pi\right)^\top$, $\boldsymbol{q}_d^{(1)} = \left(\frac{1}{4}\pi, -\frac{1}{2}\pi\right)^\top$, $\boldsymbol{q}_d^{(2)} = \left(-\frac{1}{4}\pi, \frac{1}{4}\pi\right)^\top$, as illustrated in Fig. 4a. The time duration of the segments is defined as $t_f = 3$ s. To demonstrate the learning capability of the DNN, the desired trajectory was applied twice. The first and second repetitions of the desired trajectory are referred to as Episode 1 and Episode 2, respectively.

However, due to the random initialization of the DNN weights, CONAC's initial performance is expected to be poor, leading to unstable behavior. Therefore, to avoid the instability caused by the initial poor performance of CONAC, a warm-up phase was introduced prior to applying the desired trajectory. In the warm-up phase, the manipulator was initially positioned at a stable equilibrium point, denoted as $\boldsymbol{q}_d{}^{\text{origin}} = \left(-\frac{1}{2}\pi, 0\right)^\top$. Subsequently, a trajectory $\mathcal{P}^5(t; \boldsymbol{q}_d{}^{\text{origin}}, \boldsymbol{q}_d^{(0)}, 3)$ was used to move the manipulator from $\boldsymbol{q}_d{}^{\text{origin}}$ to $\boldsymbol{q}_d^{(0)}$ over a duration of 3 seconds. Once the manipulator reached $\boldsymbol{q}_d^{(0)}$, a constant desired trajectory $\boldsymbol{q}_d = \boldsymbol{q}_d^{(0)}$ was maintained for 8 seconds to allow the system to settle and eliminate the influence of transient dynamics.

In the validation, we assume that the actuators are subject to a convex saturation function that projects the control input onto the convex set $\Theta_{\text{sat}} := \{\Theta_{\text{sat},\tau_2} \cap \Theta_{\text{sat},\boldsymbol{\tau}}\}$, as illustrated in Fig. 4b, where

$$\Theta_{\text{sat},\tau_2} := \{\tau_2 \mid |\tau_2| \le 3.5\}, \tag{30a}$$

$$\Theta_{\text{sat},\boldsymbol{\tau}} := \{\boldsymbol{\tau} \mid \|\boldsymbol{\tau}\| \le 11\}. \tag{30b}$$

The sets defined in (30a) and (30b) can be physically interpreted as the current limit imposed by the torque limit of the second joint actuator and the power source, respectively. The corresponding input constraints are mathematically described

TABLE II: Properties of the controllers

| | Description | Input Constraint Handling |
|---|---|---|
| ($C_1$) | CONAC with large update rate $\beta_j$. ($\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 100$ and $\beta_{\boldsymbol{\tau}} = 10$) | (32) $c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$ with $\overline{\tau}_2 = -\underline{\tau}_2 = 3.5$, and (34) $c_{\boldsymbol{\tau}}$ with $\overline{\tau} = 11$. |
| ($C_2$) | CONAC with small update rate $\beta_j$. ($\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 1$ and $\beta_{\boldsymbol{\tau}} = 0.1$) | (32) $c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$ with $\overline{\tau}_2 = -\underline{\tau}_2 = 3.5$, and (34) $c_{\boldsymbol{\tau}}$ with $\overline{\tau} = 11$. |
| ($C_3$) | CONAC without input constraints. ($\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = \beta_{\boldsymbol{\tau}} = 0$) | No input constraints are imposed. |
| ($C_4$) | NAC with auxiliary system. (conventional method) | $\overline{\tau}_1 = -\underline{\tau}_1 = 10.428$ $\overline{\tau_2} = -\underline{\tau_2} = 3.5$. |

TABLE III: Quantitative comparison of performances' $L_2$ norm.

| | Episode 1 | | | |
|---|---|---|---|---|
| | ($C_1$) | ($C_2$) | ($C_3$) | ($C_4$) |
| $e_1 \times 10^3$ / rad | 25.130 | 21.719 | 23.168 | 24.195 |
| $e_2 \times 10^3$ / rad | 25.016 | 10.530 | 13.061 | 12.927 |
| | **Episode 2** | | | |
| | ($C_1$) | ($C_2$) | ($C_3$) | ($C_4$) |
| $e_1 \times 10^3$ / rad | 7.066 ($-71.6\%$) | 8.021 ($-63.1\%$) | 8.408 ($-63.7\%$) | 19.721 ($-18.5\%$) |
| $e_2 \times 10^3$ / rad | 2.198 ($-88.4\%$) | 2.435 ($-76.9\%$) | 3.554 ($-72.8\%$) | 4.256 ($-67.1\%$) |

in Appendix A as the input bound constraint (32) and the input norm constraint (34), respectively.

For comparative analysis, four controllers were implemented, with their properties summarized in Table II.

*Controller 1* ($C_1$)*—CONAC with a large update rate* $\beta_j,\ \forall j \in \mathcal{I}$*, for the Lagrange Multipliers*: To handle input saturation, the proposed CONAC was implemented with the input bound constraints $c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$ (32) for the second link, and the input norm constraint $c_{\boldsymbol{\tau}}$ (34), as follows:

$$c_{\overline{\tau}_2} = \tau_2 - \overline{\tau}_2,\ c_{\underline{\tau}_2} = \underline{\tau}_2 - \tau_2,\ c_{\boldsymbol{\tau}} = \tfrac{1}{2}\left(\|\boldsymbol{\tau}\| - \overline{\tau}\right)^2, \quad (31)$$

where $\overline{\tau}_2 = -\underline{\tau}_2 = 3.5$, and $\overline{\tau} = 11$. Therefore, controller ($C_1$) utilizes the full capability of the actuator, see (30a) and (30b). The weight norm constraints $c_{\theta_i}(\cdot),\ \forall i \in \{0, \cdots, k\}$, defined in (19), were also incorporated into the CONAC framework with $\overline{\theta}_0 = \overline{\theta}_1 = \overline{\theta}_2 = 6$. As described in (8), the control input (torque) corresponds to the output of the DNN, and the adaptation law is defined in (16).

The update rates of the Lagrange multipliers for the weight norm constraints were set to $\beta_{\theta_i} = 1,\ \forall i \in \{0, \cdots, k\}$. For the input constraints, large update rates were used: $\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 100$ and $\beta_{\boldsymbol{\tau}} = 10$, to investigate the effect of large update rates.

*Controller 2* ($C_2$)*—CONAC with a small update rate* $\beta_j,\ \forall j \in \mathcal{I}$*, for the Lagrange Multipliers*: The proposed CONAC was implemented with the same input constraints as in ($C_1$), but with update rates for the input constraints reduced by a factor of 100: $\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 1$ and $\beta_{\boldsymbol{\tau}} = 0.1$. All other properties and parameters were kept identical to those of ($C_1$).

*Controller 3* ($C_3$)*—CONAC without input constraints*: The third controller was identical to ($C_1$) and ($C_2$), except that the input constraints were removed to highlight the constraint-handling capability of the proposed CONAC. In this case, the update rates of the Lagrange multipliers associated with the input constraints, i.e., $\beta_{\overline{\tau}_2}, \beta_{\underline{\tau}_2}$ and $\beta_{\boldsymbol{\tau}}$, were set to zero, such that the corresponding Lagrange multipliers $[\lambda_j]_{j \in \{\overline{\tau}_2, \underline{\tau}_2, \boldsymbol{\tau}\}}$ were not updated, see (16b). All other properties and parameters were kept the same as in ($C_1$) and ($C_2$).

*Controller 4* ($C_4$)*—NAC with auxiliary system (conventional method)*: As a baseline for comparison, the auxiliary system-based approach from [14], [15], [27] was implemented. The auxiliary system is defined by $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\zeta} = \boldsymbol{A}_\zeta \boldsymbol{\zeta} + \boldsymbol{B}_\zeta \Delta\boldsymbol{\tau}$, with initial condition $\boldsymbol{\zeta}|_{t=0} = \mathbf{0}_{2\times 1}$, where $\boldsymbol{\zeta} \in \mathbb{R}^2$ represents the auxiliary state variables, and $\boldsymbol{A}_\zeta \in \mathbb{R}^{2\times 2}_{<0}$ and $\boldsymbol{B}_\zeta \in \mathbb{R}^{2\times 2}$ are user-designed matrices. The auxiliary state $\boldsymbol{\zeta}$ is driven by the overflow amount of the control input $\Delta\boldsymbol{\tau}$, where each element is defined as $\Delta\tau_i = \tau_i - \max(\underline{\tau}_i, \min(\overline{\tau}_i, \tau_i)),\ \forall i \in \{1, 2\}$.

This auxiliary state was incorporated into the adaptation law by replacing the filtered tracking error $\boldsymbol{r}$ with $\boldsymbol{r}+\boldsymbol{\zeta}$. As a result, the weights were adapted to minimize both the filtered tracking error and the auxiliary state, thereby mitigating saturation in each control input.

The auxiliary system matrices were chosen as $\boldsymbol{A}_\zeta = \mathrm{diag}(-10, -10)$ and $\boldsymbol{B}_\zeta = \mathrm{diag}(10^3, 10^3)$, to balance responsiveness to input overflow with the convergence speed of the auxiliary dynamics. Since the auxiliary system only accounts for element-wise input saturation—corresponding to the input bound constraint (32)—the upper bound of the first actuator torque $\overline{\tau}_1 = -\underline{\tau}_1$ was set to $10.428$ to approximate the input norm constraint $\overline{\tau} = 11$, given $\overline{\tau}_1^2 + \overline{\tau}_2^2 = \overline{\tau}^2$.

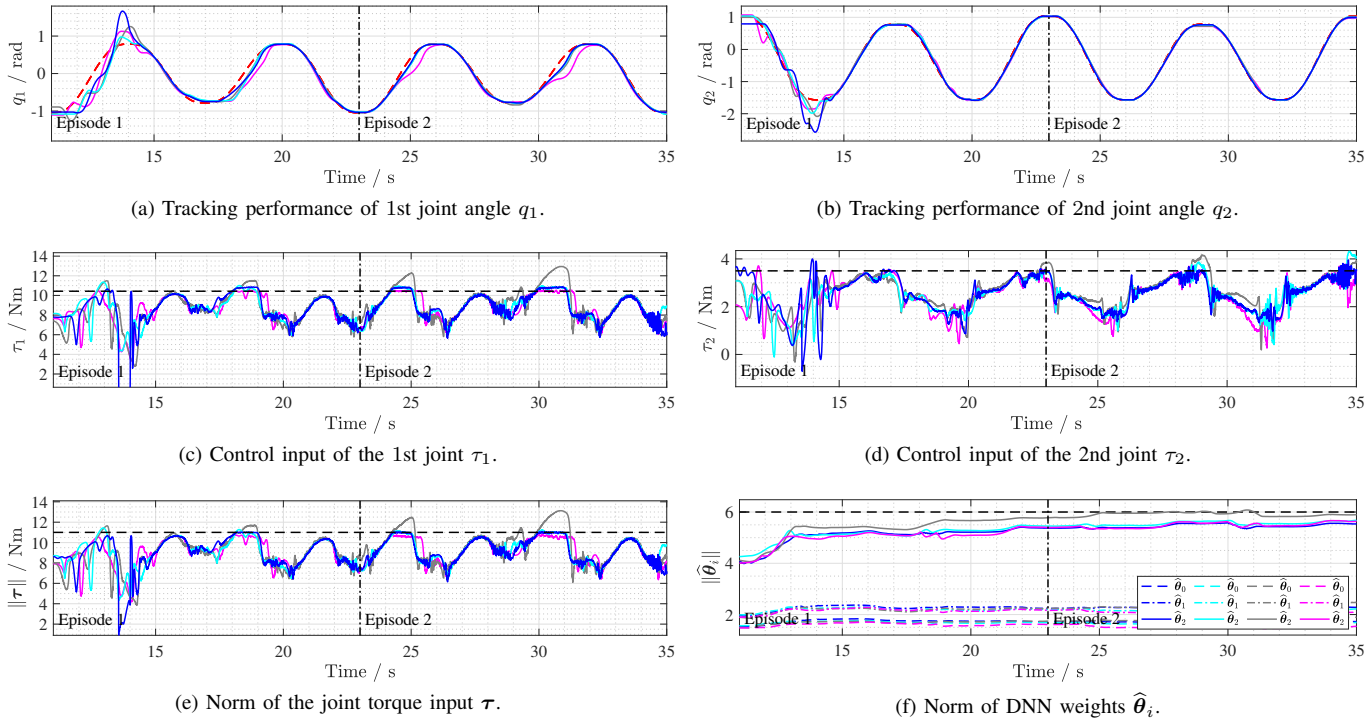
All other properties and parameters were kept the same as in ($C_1$), ($C_2$), and ($C_3$), except that the weight boundedness was handled using the projection operator used in [13].

For all controllers, i.e., ($C_1$), ($C_2$), ($C_3$), and ($C_4$), the DNNs shared the same architecture, consisting of two hidden layers with four nodes each, i.e., $k = 2$, and $l_0 = 6, l_1 = 4, l_2 = 4$, and $l_3 = 2$ in (10). The input vector to the DNNs was defined as $\boldsymbol{q}_n = \left(\boldsymbol{q}^\top, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}^\top, \boldsymbol{r}^\top, 1\right)^\top \in \mathbb{R}^7$, where the scalar 1 was included to account for the bias term in the weight matrix. The adaptation gain was set to $\alpha = 0.2$, and the filtering matrix in (4) was chosen as $\boldsymbol{\Lambda} = \mathrm{diag}(5, 15)$. To support real-time implementation, the controller operated at a sampling rate of 250 Hz, with control results collected via CAN communication using a PCAN-USB interface and recorded on a computer at the same rate.

The tracking performance was quantitatively evaluated using the $L_2$-norm of the each joint angle's tracking error, defined as $\sqrt{\int_0^T \|e_i(t)\|\,\mathrm{d}t},\ \forall i \in \{1, 2\}$, where $T \in \mathbb{R}_{>0}$ denotes the duration of each episode.

### B. Validation Results

*1) Tracking Performance:* The real-time implementation results are presented in Fig. 5, and the quantitative comparison of tracking performance is summarized in Table III. Across the two episodes, all controllers improved their tracking performance. Specifically, the $L_2$-norm of $e_1$ was reduced by




(a) Tracking performance of 1st joint angle $q_1$.

(b) Tracking performance of 2nd joint angle $q_2$.

(c) Control input of the 1st joint $\tau_1$.

(d) Control input of the 2nd joint $\tau_2$.

(e) Norm of the joint torque input $\boldsymbol{\tau}$.

(f) Norm of DNN weights $\widehat{\boldsymbol{\theta}}_i$.

Fig. 5: Real-time implementation results of ($C_1$) [—], ($C_2$) [—], ($C_3$) [—], ($C_4$) [—], and desired trajectory $\boldsymbol{q}_d$ [- - -].

71.6% for ($C_1$), 63.1% for ($C_2$), 63.7% for ($C_3$), and 18.5% for ($C_4$); while the $L_2$-norm of $e_2$ was reduced by 88.4%, 76.9%, 72.8%, and 67.1%, respectively. Qualitatively, as illustrated in Fig. 5a and Fig. 5b, oscillations were suppressed during the second episode. These results indicate that the DNNs in all controllers successfully learned the ideal control input $\boldsymbol{\tau}^*$ using the filtered tracking error $\boldsymbol{r}$, despite $\boldsymbol{\tau}^*$ being completely unknown a priori. Furthermore, the stable learning observed under random weight initialization and without prior knowledge of the system demonstrated the effectiveness of the proposed CONAC in enabling online learning capability.

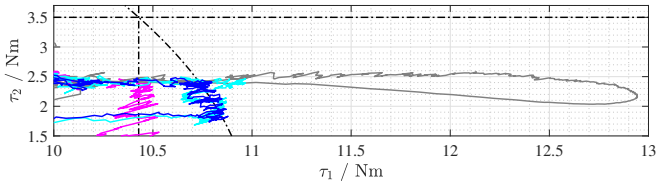
Notably, in both episodes, all controllers failed to track the desired trajectory of $q_1$ during certain time intervals: from 17.5 s to 19.9 s in Episode 1, and from 23.5 s to 25.9 s and from 29.5 s to 31.9 s in Episode 2, as shown in Fig. 5a. This degradation in tracking performance was attributed to actuator saturation during those intervals, as observed in Fig. 5d and Fig. 5e.

The tracking performance of CONAC with input constraints—i.e., ($C_1$) and ($C_2$)—outperformed the auxiliary system-based NAC ($C_4$) in the second episode, as shown in Table III. This improvement was attributed to the ability of ($C_1$) and ($C_2$) to fully utilize the actuator's capacity, whereas ($C_4$) was limited by the design of the auxiliary system, which constrained the first joint actuator. While ($C_3$) also showed better performance than ($C_4$), this was primarily because it did not consider input saturation and thus used more of the actuator's capacity. However, since ($C_3$) did not handle the control input saturation, it generated excessively large control efforts to minimize the tracking error, even though these efforts could not be realized due to actuator limits, as shown in Fig. 5c–5e. In addition, the severity of saturation violations in ($C_3$) increased during periods when the input constraints were active. These large violations pose a significant risk to the physical system, potentially causing actuator damage or failure. Furthermore, once the control saturation was lifted—typically due to changes in the desired trajectory—this resulted in oscillatory behavior, as CONAC's weights had been adapted to generate excessively high control inputs, see Fig. 5e, where the input constraints were lifted at 25.9 s and 31.9 s.

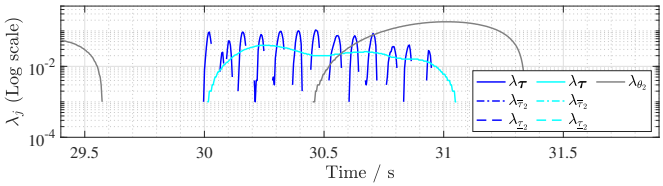
*2) Constraint Handling:* In this section, the constraint-handling capability of all controllers is investigated. As shown in Fig. 5c–5e, all controllers except ($C_3$), which did not consider the control input saturation, satisfied the control input saturation illustrated in Fig. 4b. In ($C_1$) and ($C_2$), control input saturation was addressed through corresponding input constraints within the constrained optimization framework, whereas ($C_4$) employed an auxiliary system, as summarized in Table II.

To investigate the constraint handling process, the time interval from 29.4 s to 31.9 s was selected, during which the control input saturation was active for all controllers. In Fig. 6, the control input locus, the Lagrange multipliers, and the auxiliary states are illustrated over this time interval. The properties of the controllers are illustrated in Fig. 6a, where the control input loci of ($C_1$) and ($C_2$) fully explore the feasible input domain shown in Fig. 4b, whereas ($C_3$) exceeds the input saturation limits, and ($C_4$) is restricted in its first actuator input limit due to the auxiliary system design.

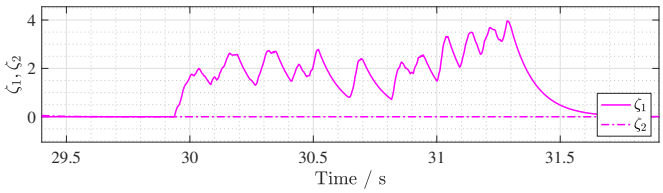
The input constraint handling from 29.4 s to 31.9 s is




(a) Control input $\boldsymbol{\tau}$ locus during the time interval from 29.4 s to 31.9 s.




(b) Lagrange multipliers $\lambda_j$ during the time interval from 29.4 s to 31.9 s.




(c) Auxiliary states $\boldsymbol{\zeta}$ during the time interval from 29.4 s to 31.9 s.

Fig. 6: Constraint handling behavior of ($C_1$) [—], ($C_2$) [—], ($C_3$) [—], and ($C_4$) [—] during the time interval from 29.4 s to 31.9 s.

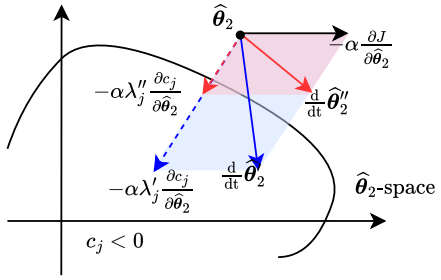



Fig. 7: The effect of the Lagrange multiplier $\lambda_j$ on the adaptation direction of $\widehat{\boldsymbol{\theta}}_2$ is illustrated. The notations $(\cdot')$ and $(\cdot'')$ denote two distinct cases corresponding to large and small values of the Lagrange multiplier, respectively, i.e., $\lambda_j' > \lambda_j''$.

examined in detail. For ($C_1$) and ($C_2$), activation of input constraints $c_j$ dynamically generated the corresponding Lagrange multipliers $\lambda_j$, as shown in Fig. 6b. These multipliers guided the adaptation of the weights $\widehat{\boldsymbol{\theta}}_i,\ \forall i \in \{0,1,2\}$, to reduce constraint violations in accordance with (16a).

To further illustrate this effect, Fig. 7 compares the adaptation direction of the output layer's weight $\widehat{\boldsymbol{\theta}}_2$ under two different magnitudes of $\lambda_j$. Under Assumptions 2 and 3, the convexity of $c_j$ in the $\widehat{\boldsymbol{\theta}}_2$-space (see Lemma 2) is illustrated in Fig. 7. The figure highlights the influence of the update rate $\beta_j$: a larger $\beta_j$ (as in ($C_1$)) yields a larger $\lambda_j$, resulting in a more aggressive adjustment of $\widehat{\boldsymbol{\theta}}_2$ via the term $-\lambda_j\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_2}$ in (16a), thereby accelerating constraint satisfaction—though at the cost of oscillatory learning behavior. In contrast, the smaller update rate in ($C_2$) leads to more conservative weight updates and smoother, but slower, convergence. These behaviors are clearly

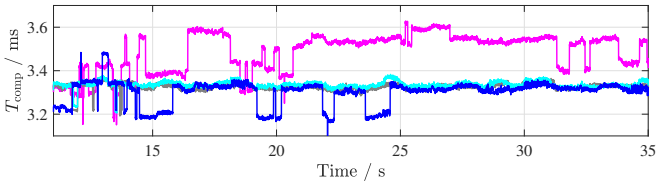



Fig. 8: Computational time $T_{\text{comp}}$ of ($C_1$) [—], ($C_2$) [—], ($C_3$) [—], and ($C_4$) [—].

observed in Fig. 6a and Fig. 6b, where ($C_1$) satisfies the input constraint $c_{\boldsymbol{\tau}}$ more quickly than ($C_2$), which shows repeated growth and reset of $\lambda_{\boldsymbol{\tau}}$.

For ($C_4$), the auxiliary state $\zeta_1$ was generated in response to the control input saturation of the first link actuator $\tau_1$, as shown in Fig. 6c. Since the weights were adapted to reduce both the filtered error $\boldsymbol{r}$ and the auxiliary state $\boldsymbol{\zeta}$, the control input saturation was subsequently reduced, as evident in Fig. 6a. Once the saturation was resolved at around 31.2 s (see Fig. 5c), the auxiliary state $\zeta_1$ converged to zero due to the stable state matrix $\boldsymbol{A}_\zeta$ in the auxiliary system dynamics, given by $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\zeta} = \boldsymbol{A}_\zeta\boldsymbol{\zeta} + \boldsymbol{B}_\zeta\Delta\boldsymbol{\tau}$ with $\Delta\boldsymbol{\tau} = \boldsymbol{0}_{2\times1}$, as shown again in Fig. 6c.

Finally, the handling of the weight norm constraints $c_{\theta_i},\ \forall i \in \{0,1,2\}$, is investigated. Among all controllers, the weight norm constraint was only activated for the output layer weights $\widehat{\boldsymbol{\theta}}_2$ in ($C_3$), as shown in Fig. 5f. This is because the weights in ($C_1$), ($C_2$), and ($C_4$) were implicitly suppressed by the input constraints, which were not applied in (C3). Furthermore, the constraints on the inner and hidden layer weights were never violated; that is, both $\|\widehat{\boldsymbol{\theta}}_0\|$ and $\|\widehat{\boldsymbol{\theta}}_1\|$ consistently remained below their limits. This was attributed to the fact that the backpropagated error was scaled by the gradients of the activation function, which are bounded in $(0,1]$, i.e., $\sigma(\cdot) = \tanh(\cdot)$.

Additional details on the constraint handling process are provided in Fig. 6b. Similar to the case of the input constraints, the Lagrange multipliers $\lambda_{\theta_2}$ increased at around 30.4 s, as shown in Fig. 6b, when $\widehat{\boldsymbol{\theta}}_2$ exceeded the prescribed norm constraint $\bar{\theta}_2$ (see Fig. 5f), thereby adjusting the adaptation direction of $\widehat{\boldsymbol{\theta}}_2$ to mitigate the violation of the constraint $c_{\theta_2}$.

*3) Effectiveness of CONAC in Real-time Applications:* The effectiveness of the proposed CONAC in real-time applications was evaluated by measuring the computational time using CPU clock ticks on the OpenCR1.0 board. As shown in Fig. 8, the computational times of all controllers remained below 4 ms, demonstrating that the proposed CONAC is suitable for real-time implementation at a sampling rate of 250 Hz.

## VII. Conclusion

This paper proposed a constrained optimization-based neuro-adaptive controller (CONAC) for unknown Euler-Lagrange systems, addressing both weight norm and convex input constraints within a unified optimization framework. The stability of CONAC was analyzed using Lyapunov theory,

demonstrating bounded tracking errors and weight estimation under real-time adaptation.

CONAC effectively incorporated both convex input constraints and weight norm constraints, ensuring that actuator limitations and neural network weights remained within predefined bounds. By embedding these constraints into the optimization process, CONAC ensured weight convergence in accordance with the Karush-Kuhn-Tucker (KKT) conditions, thereby ensuring both stability and optimality.

Real-time implementation validated the superior performance of CONAC compared to a conventional method. CONAC successfully handled complex input constraints while rigorously managing weight norms, resulting in improved tracking accuracy and stable behavior without significant oscillations.

Future work may extend this approach to include constraints on both control inputs and system states, further enhancing the flexibility and robustness of neuro-adaptive control systems within constrained optimization frameworks.

## APPENDIX

### A. Input Constraint Candidates

This section introduces potential input constraints that can be incorporated into the proposed CONAC framework.

*1) Input Bound Constraint:* Most physical systems are subject to control input limitations due to inherent electrical and mechanical constraints. These are expressed as $\boldsymbol{c}_{\overline{\tau}} := [c_{\overline{\tau}_i}]_{i\in\{1,\cdots,n\}}$ and $\boldsymbol{c}_{\underline{\tau}} := [c_{\underline{\tau}_i}]_{i\in\{1,\cdots,n\}}$, where

$$c_{\overline{\tau}_i} = \tau_{(i)} - \tau_{\overline{\tau}_i}, \quad c_{\underline{\tau}_i} = \tau_{\underline{\tau}_i} - \tau_{(i)}, \tag{32}$$

with $\tau_{\overline{\tau}_i} \in \mathbb{R}$ and $\tau_{\underline{\tau}_i} \in \mathbb{R}$ representing the maximum and minimum control input bounds, respectively. The gradients of $\boldsymbol{c}_{\overline{\tau}}$ and $\boldsymbol{c}_{\underline{\tau}}$ with respect to $\widehat{\boldsymbol{\theta}}$ are given by

$$\begin{aligned}\frac{\partial \boldsymbol{c}_{\overline{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} = = +\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = +\begin{bmatrix}(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top) & \cdots & (\cdot)\end{bmatrix} \in \mathbb{R}^{n\times\Xi},\\ \frac{\partial \boldsymbol{c}_{\underline{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} = = -\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = -\begin{bmatrix}(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top) & \cdots & (\cdot)\end{bmatrix} \in \mathbb{R}^{n\times\Xi}.\end{aligned} \tag{33}$$

*2) Input Norm Constraint:* Consider the control input $\boldsymbol{\tau}$ as the torque corresponding to its generalized coordinate. Since torque is typically linearly proportional to current, actuators that share a common power source are often subject to total current limitations. This can be captured by the following inequality constraint:

$$c_{\boldsymbol{\tau}} = \tfrac{1}{2}\ \left(\|\boldsymbol{\tau}\|^2 - \overline{\tau}^2\right), \tag{34}$$

with the maximum allowable control input magnitude $\overline{\tau} \in \mathbb{R}_{>0}$. This input norm constraint is also commonly applied in current and torque control problems for electric motors [39]. The gradient of $c_{\boldsymbol{\tau}}$ with respect to $\widehat{\boldsymbol{\theta}}$ is given by

$$\frac{\partial \boldsymbol{c}_{\boldsymbol{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} = \sum_{i=1}^{n} \tau_i \left(\mathrm{row}_i\left(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}\right)\right)^\top = \frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}^\top \boldsymbol{\tau} \in \mathbb{R}^{\Xi}. \tag{35}$$

It should be noted that constraints (32) and (34) can be imposed simultaneously, as their gradients (33) and (35) are linearly independent, satisfying Assumption 3, i.e., the LICQ condition.

## REFERENCES


[1] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas (Second Edition)*. Princeton University Press, 2009.

[2] P. Ioannou and B. Fidan, *Adaptive Control Tutorial*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006.

[3] G. Tao, *Adaptive Control Design and Analysis (Adaptive and Learning Systems for Signal Processing, Communications and Control Series)*. USA: John Wiley & Sons, Inc., 2003.

[4] F. L. Lewis, A. Yesildirak, and S. Jagannathan, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. USA: Taylor & Francis, Inc., 1998.

[5] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches (Adaptive and Learning Systems for Signal Processing, Communications and Control Series)*. USA: Wiley-Interscience, 2006.

[6] F. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 388–399, 1996.

[7] S. S. Ge, C. C. Hang, T. H. Lee, and Z. Tao, *Stable adaptive neural network control*, ser. The International Series on Asian Studies in Computer and Information Science. New York, NY: Springer, Dec. 2010.

[8] A. Yeşildirek and F. Lewis, "Feedback linearization using neural networks," *Automatica*, vol. 31, no. 11, pp. 1659–1664, 1995.

[9] M. Ryu, J. Kim, and K. Choi, "Imposing a weight norm constraint for neuro-adaptive control," TechRxiv, Preprint, Oct. 2024, accepted for publication in 2025 23rd European Control Conference (ECC).

[10] M. Ryu, N. Monzen, P. Seitter, K. Choi, and C. M. Hackl, "Constrained optimization-based neuro-adaptive control (CONAC) for synchronous machine drives under voltage constraints," TechRxiv, Preprint, Apr. 2025, accepted for publication in the 51st Annual Conference of the IEEE Industrial Electronics Society (IECON 2025).

[11] J. Liu, *Radial basis function (RBF) neural network control for mechanical systems*, 2013th ed. Berlin, Germany: Springer, Jan. 2013.

[12] S. Ge and C. Wang, "Direct adaptive NN control of a class of nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 214–221, 2002.

[13] O. S. Patil, D. M. Le, M. L. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Systems Letters*, vol. 6, pp. 1855–1860, 2022.

[14] K. Esfandiari, F. Abdollahi, and H. Talebi, "A stable nonlinear in parameter neural network controller for a class of saturated nonlinear systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2533–2538, 2014, 19th IFAC World Congress.

[15] K. Esfandiari, F. Abdollahi, and H. A. Talebi, "Adaptive control of uncertain nonaffine nonlinear systems with input saturation using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2311–2322, 2015.

[16] W. Gao and R. Selmic, "Neural network control of a class of nonlinear systems with actuator saturation," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 147–156, 2006.

[17] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," *arXiv preprint arXiv:1705.05502*, 2018.

[18] E. J. Griffis, O. S. Patil, Z. I. Bell, and W. E. Dixon, "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based control," *IEEE Control Systems Letters*, vol. 7, pp. 2976–2981, 2023.

[19] R. G. Hart, E. J. Griffis, O. S. Patil, and W. E. Dixon, "Lyapunov-based physics-informed long short-term memory (LSTM) neural network-based adaptive control," *IEEE Control Systems Letters*, vol. 8, pp. 13–18, 2024.

[20] M. Ryu and K. Choi, "CNN-based end-to-end adaptive controller with stability guarantees," arXiv, Preprint, 2024.

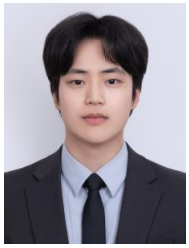
[21] K. Esfandiari, F. Abdollahi, and H. A. Talebi, *Neural network-based adaptive control of uncertain nonlinear systems*, 2022nd ed. Cham, Switzerland: Springer Nature, Jun. 2021.

[22] X. Zhou, H. Shen, Z. Wang, H. Ahn, and J. Wang, "Driver-centric lane-keeping assistance system design: A noncertainty-equivalent neuro-adaptive control approach," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 6, pp. 3017–3028, 2023.

[23] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, the book can be consulted by contacting: PH-AID: Wallet, Lionel.

[24] E. Arefinia, H. A. Talebi, and A. Doustmohammadi, "A robust adaptive model reference impedance control of a robotic manipulator with actuator saturation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 2, pp. 409–420, 2020.
[25] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 334–344, 2016.
[26] G. Peng, C. Yang, W. He, and C. L. P. Chen, "Force sensorless admittance control with neural learning for robots with actuator saturation," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 4, pp. 3138–3148, 2020.
[27] S. Karason and A. Annaswamy, "Adaptive control in the presence of input constraints," *IEEE Transactions on Automatic Control*, vol. 39, no. 11, pp. 2325–2330, 1994.
[28] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research and financial engineering. New York, NY: Springer, 2006.
[29] B. Evens, P. Latafat, A. Themelis, J. Suykens, and P. Patrinos, "Neural network training as an optimal control problem : — an augmented Lagrangian approach —," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 5136–5143.
[30] J. Wang, F. Yu, X. Chen, and L. Zhao, "ADMM for efficient deep learning with global convergence," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 111–119.
[31] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 2016, pp. 2722–2731.
[32] P. Kidger and T. Lyons, "Universal Approximation with Deep Narrow Networks," in *Proceedings of Thirty Third Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, J. Abernethy and S. Agarwal, Eds., vol. 125. PMLR, 09–12 Jul 2020, pp. 2306–2327.
[33] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.
[34] B. Sengupta, K. Friston, and W. Penny, "Efficient gradient computation for dynamical models," *NeuroImage*, vol. 98, pp. 521–527, 2014.
[35] I. Douratsos and J. B. Gomm, "Neural network based model reference adaptive control for processes with time delay," *International Journal of Information and Systems Sciences*, vol. 3, no. 1, pp. 161–179, 2007.
[36] M. Saerens and A. Soquet, "Neural controller based on back-propagation algorithm," *IEE Proceedings F (Radar and Signal Processing)*, vol. 138, pp. 55–62, 1991.
[37] C. A. Desoer and M. Vidyasagar, *Feedback Systems*. Society for Industrial and Applied Mathematics, 2009.
[38] ROBOTIS. (2016) OpenCR1.0 [Online]. Available: https://www.robotis.us/opencr1-0/. Accessed: Apr. 8, 2025.
[39] K. Choi, J. Kim, and K.-B. Park, "Generalized model predictive torque control of synchronous machines," *IEEE/ASME Transactions on Mechatronics*, pp. 1–11, 2024.

**Myeongseok Ryu** received the B.S. degree in Mechanical Engineering from Incheon National University, South Korea, in 2023, and the M.S. degree in Mechanical Engineering from GIST, South Korea, in 2025. He is currently pursuing the Ph.D. degree in the Cho Chun Shik Graduate School of Mobility at Korea Advanced Institute of Science and Technology (KAIST), South Korea. His research interests include neuro-adaptive control, online optimization, and contraction theory.



**Donghwa Hong** received his B.S. degree in Physics and Engineering Physics from Yonsei University, Wonju, South Korea, in 2023. He is currently pursuing the M.S. degree in Mechanical Engineering from GIST, Gwangju, South Korea. His research interests include robot control, neural-networks, and model identification.

**Kyunghwan Choi** received his B.S., M.S., and Ph.D. degrees in Mechanical Engineering from KAIST, Daejeon, South Korea, in 2014, 2016, and 2020, respectively. Following his Ph.D., he worked at the Center for Eco-Friendly & Smart Vehicles at KAIST as a Postdoctoral Fellow and later as a Research Assistant Professor. In 2022, he joined GIST as an Assistant Professor in the Department of Mechanical and Robotics Engineering. Currently, he serves as an Assistant Professor at the Cho Chun Shik Graduate School of Mobility at KAIST, where he is also the Director of the Mobility Intelligence and Control Laboratory. His research focuses on optimal and learning-based control for connected, automated, and electrified vehicles (CAEVs).