Graphical Abstract

Constrained Optimization-Based Neuro-Adaptive Control (CONAC) for Weight and Convex Input Constraints

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

Highlights

Constrained Optimization-Based Neuro-Adaptive Control (CONAC) for Weight and Convex Input Constraints

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

- Research highlight 1

- Research highlight 2

Constrained Optimization-Based Neuro-Adaptive Control (CONAC) for Weight and Convex Input Constraints

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

Department of Mechanical and Robotics Engineering, Gwangju Institute of Science and Technology (GIST) , 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, Korea

Abstract

This study presents a constrained optimization-based neuro-adaptive controller (CONAC) for weight norm and convex input constraints. A deep neural network (DNN) is employed to approximate the ideal stabilizing control law, while addressing both types of constraints. The weight adaptation laws are formulated through a constrained optimization problem, ensuring first-order optimality conditions at steady state. The controller's stability is rigorously analyzed using Lyapunov theory, guaranteeing bounded tracking errors and DNN weights. Real-time implementation was conducted on a 2-DOF robotic manipulator, demonstrating the controller's effectiveness in achieving desired trajectory tracking while satisfying constraints.

Keywords: Neuro-adaptive control, constrained optimization, deep neural network, input constraint

Notation

In this study, the following notation is used:

- $\otimes$ denotes the Kronecker product (Bernstein, 2009, Definition 7.1.2).
- $\boldsymbol{x} = [x_i]_{i\in\{1,\cdots,n\}} \in \mathbb{R}^n$ and $\boldsymbol{A} := [a_{ij}]_{i\in\{1,\cdots,n\},j\in\{1,\cdots,m\}} \in \mathbb{R}^{n\times m}$ denote a vector and a matrix.
- $\text{row}_i(\boldsymbol{A})$ denotes the i^{th} row of the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$.
- $\text{vec}(\boldsymbol{A}) := [\text{row}_1(\boldsymbol{A}^\top), \cdots, \text{row}_m(\boldsymbol{A}^\top)]^\top$ for $\boldsymbol{A} \in \mathbb{R}^{n\times m}$.
- $\lambda_{\min}(\boldsymbol{A})$ denotes the minimum eigenvalue of the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.
- $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix, and $\boldsymbol{0}_{n\times m}$ denotes the $n \times m$ zero matrix.

1. Introduction

1.1. 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. To address these challenges, adaptive control methods have been widely employed to ensure robust performance in the presence of system uncertainties Ioannou and Fidan (2006); Tao (2003).

More recently, neuro-adaptive control approaches have been introduced to approximate unknown system dynamics or entire control laws using neural networks (NNs) Farrell and Polycarpou (2006). 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 neuro-adaptive control, including simpler architectures like single-hidden layer (SHL) neural networks Ge et al. (2010); Yeşildirek and Lewis (1995) and radial basis function (RBF) neural networks Liu (2013); Ge and Wang (2002), as well as more complex models like deep neural networks (DNNs) Patil et al. (2022) and their variations. SHL and RBF NNs are often employed to approximate uncertain system dynamics or controllers due to their simplicity Esfandiari et al. (2014, 2015); Yeşildirek and Lewis (1995); Gao and Selmic (2006), while DNNs offer greater expressive power, making them more effective for complex system approximations Rolnick and Tegmark (2018). Additionally, variations of DNNs, such as long short-term memory (LSTM) networks for time-varying dynamics Liu (2013) and physics-informed neural networks (PINNs) for leveraging physical system knowledge Hart et al. (2024), have further extended the capabilities of neuro-adaptive control systems.

A critical aspect of neuro-adaptive control is the weight adaptation law, which governs how NN parameters 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 Esfandiari et al. (2021). 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 neuro-adaptive controllers. he following section will provide a detailed review of the existing solutions to these challenges.

1.2. Literature Review

1.2.1. Ensuring Weight Norm Boundedness

A common challenge in neuro-adaptive control 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 Zhou et al. (2023); Griffis et al. (2023); Patil et al. (2022), 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 σ-modification Ge and Wang (2002) and ϵ-modification Esfandiari et al. (2015); Gao and Selmic (2006). 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, leaving room for improvement in terms of performance optimization.

1.2.2. Satisfying Input Constraints

The second major issue is satisfying input constraints, particularly in systems where actuators are subject to physical limitations. The unpredictable outputs of NNs can sometimes lead to excessively large control inputs, violating these constraints. This problem is exacerbated in neuro-adaptive controllers that attempt to cancel out system dynamics using conventional methods like feedback linearization or backstepping. In such cases, controllers may produce overly aggressive control inputs, even when the system's natural dynamics are stabilizing, leading to unnecessary saturation of the control inputs.

To address input saturation, many studies introduced auxiliary systems. These systems mitigated the effects of control input saturation by modifying the control strategy when saturation occurred. For instance, in Esfandiari et al. (2014); Karason and Annaswamy (1994); Esfandiari et al. (2015), auxiliary states were generated whenever input saturation was detected, and these states were incorporated into the adaptation law to adjust the NN weights accordingly. This approach helped the controller reduce input saturation by indirectly regulating the auxiliary states.

Alternatively, auxiliary states can also be used as feedback terms in the control law to directly compensate for the effects of input saturation constraints, as demonstrated in Arefinia et al. (2020); He et al. (2016); Peng et al. (2020). In Gao and Selmic (2006), the NN was used to approximate the desired control input, compensating for input saturation. 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.

1.2.3. Limitations of Existing Approaches and Potential of Constrained Optimization

Although both the projection operator methods for weight norm boundedness and the auxiliary system approach for input constraints have shown effectiveness, they come with significant limitations. Projection operators and modification techniques do not guarantee the optimality of the adapted weights. Moreover, auxiliary systems typically handle only simple forms of input constraints, such as input bound constraints, limiting their ability to address more complex, nonlinear constraints like input norm constraints.

To overcome these limitations, constrained optimization offers a promising approach. By formulating the neuro-adaptive control problem as an 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 norm and input constraints. Constrained optimization provides a theoretical framework for defining optimality and presents numerical methods for finding solutions that satisfy the constraints Nocedal and Wright (2006).

In existing literature, constrained optimization techniques, such as the Augmented Lagrangian Method (ALM) Evens et al. (2021) and the Alternating Direction Method of Multipliers (ADMM) Wang et al. (2019); Taylor et al. (2016), have been used to train NNs offline. These methods impose constraints to address issues like gradient vanishing in backpropagation. However, to the best of the authors' knowledge, no prior work has applied constrained optimization to adaptive control systems with real-time weight adaptation. This gap suggests that constrained optimization could be key to addressing both weight norm boundedness and input constraints in a unified, theoretically grounded framework, particularly in real-time neuro-adaptive control.

1.3. Contributions

The main contributions of this study are listed as follows:

- A constrained optimization-based neuro-adaptive controller (CONAC) is developed using a DNN, where input constraints and the boundedness of NN weights are formulated as inequality constraints within the adaptation process.

- The weight adaptation laws in CONAC are derived using constrained optimization theory to minimize the objective function while satisfying the inequality constraints. The adaptation laws ensure convergence of the weights to the first-order optimality conditions, specifically the Karush-Kuhn-Tucker (KKT) conditions.

1.4. Organization

The remainder of this paper is organized as follows. Section 2 presents the target system and control objective. Section 3 introduces the proposed controller and the architecture of DNN in the controller. In Section 4, the weight adaptation laws are developed. Section 5 analyzes the stability of the proposed controller. A real-time implementation of the proposed controller is presented in Section 6, where the proposed controller is applied to a 2-DOF robotic manipulator. Finally, Section 7 concludes the paper and discusses potential future work. The candidates of input constraints are presented in Appendix A.

2. Problem Formulation

2.1. Model Dynamics and Control Objective

Consider an uncertain Euler-Lagrange system modeled as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{V}_m + \boldsymbol{F} + \boldsymbol{G} + \boldsymbol{\tau}_d = \text{sat}(\boldsymbol{\tau}) \tag{1}$$

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

The Euler-Lagrange system (1) satisfy some important physical properties (Lewis et al., 1998, see, Chap. 3 Tab. 3.2.1). We introduce the following properties:

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 $\dot{\boldsymbol{M}} - 2\boldsymbol{V}_m$ is skew-symmetric, i.e., $\boldsymbol{x}^\top(\dot{\boldsymbol{M}} - 2\boldsymbol{V}_m)\boldsymbol{x} = 0,\ \forall \boldsymbol{x} \in \mathbb{R}^n$.*

Property 3. *The disturbance $\boldsymbol{\tau}_d$ is bounded so that $\|\boldsymbol{\tau}_d\| \leq \overline{\tau}_d$.*

In conclusion, the control objective is to develop a neuro-adaptive controller that enables $\boldsymbol{q}$ to track a continuously differentiable desired trajectory $\boldsymbol{q}_d := \boldsymbol{q}_d(t) : \mathbb{R} \to \mathbb{R}^n$, compensating for the external disturbance while addressing the imposed constraints. Considering the control input saturation function, $\boldsymbol{q}_d(t)$ is supposed to satisfy the following assumption:

Assumption 1. *The desired trajectory $\boldsymbol{q}_d(t)$ is bounded so that $\|\boldsymbol{q}_d(t)\| \leq \overline{q}_d$ and available to design a feasible control input in the presence of the control input saturation.*

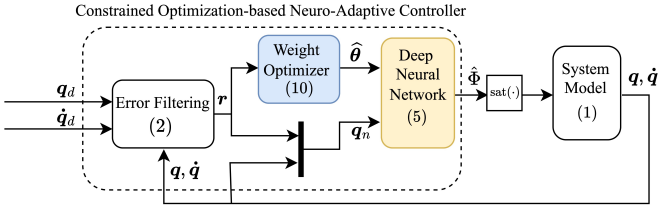


Figure 1: Architecture of the constrained optimization-based neuro-adaptive controller (CONAC).

3. Control Law Development

The architecture of the proposed CONAC is illustrated in Fig. 1, consisting of: a DNN that functions as a neuro-adaptive controller, and a weight optimizer for the DNN. Section 3.1 presents the neuro-adaptive controller along with the reference generator, and Section 3.2 defines the DNN model. The weight optimizer will be detailed in Section 4.1.

3.1. Neuro-Adaptive Controller Design

First, let us define the filtered tracking error $\boldsymbol{r} \in \mathbb{R}^n$ as

$$\boldsymbol{r} := \dot{\boldsymbol{e}} - \boldsymbol{\Lambda}\boldsymbol{e}, \tag{2}$$

where $\boldsymbol{e} := \boldsymbol{q} - \boldsymbol{q}_d$ denotes the tracking error, $\dot{\boldsymbol{e}} := \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_d$ represents the derivative of the tracking error, and $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}_{>0}$ is a user-designed filtering matrix. Since (2) 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

$$\boldsymbol{M}\dot{\boldsymbol{r}} = -\boldsymbol{V}_m\boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \text{sat}(\boldsymbol{\tau}), \tag{3}$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ denotes arbitrary symmetric positive definite matrix and $\boldsymbol{f} := \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d) = \boldsymbol{K}\boldsymbol{r} + \boldsymbol{M}(-\ddot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}) + \boldsymbol{V}_m(-\dot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\boldsymbol{e}) - \boldsymbol{F} - \boldsymbol{G} \in \mathbb{R}^n$ denotes the lumped system uncertainty.

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

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_1 =& \boldsymbol{r}^\top\dot{\boldsymbol{M}}\boldsymbol{r} + \tfrac{1}{2}\boldsymbol{r}^\top\tfrac{\mathrm{d}\boldsymbol{M}}{\mathrm{d}\boldsymbol{r}}\boldsymbol{r} \\
=& \boldsymbol{r}^\top(-\boldsymbol{V}_m\boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \text{sat}(\boldsymbol{\tau})) \\
&+ \tfrac{1}{2}\boldsymbol{r}^\top\dot{\boldsymbol{M}}\boldsymbol{r} \\
=& -\boldsymbol{r}^\top\boldsymbol{K}\boldsymbol{r} + \boldsymbol{r}^\top(\boldsymbol{f} + \text{sat}(\boldsymbol{\tau}) - \boldsymbol{\tau}_d) \\
&+ \tfrac{1}{2}\boldsymbol{r}^\top(\dot{\boldsymbol{M}} - 2\boldsymbol{V}_m)\boldsymbol{r} \\
\leq& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\boldsymbol{f} + \text{sat}(\boldsymbol{\tau})) \\
\leq& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\text{sat}(\boldsymbol{\tau}) - \boldsymbol{\tau}^*),
\end{aligned} \tag{4}$$

where $\boldsymbol{\tau}^* := -\boldsymbol{f}$ is the ideal control input whose maximum norm is $\overline{\tau}$ according to Assumption 1 and $\text{sat}(\cdot)$. Therefore, one can conclude that $\boldsymbol{r}$ is exponentially stable so that $\lim_{t\to\infty}\|\boldsymbol{r}\| = \frac{\overline{\tau}_d}{\lambda_{\min}(\boldsymbol{K})}$, if $\boldsymbol{\tau} = -\boldsymbol{\tau}^*$ can be realized. However, $\boldsymbol{\tau}^*$ is not available in practice, since $\boldsymbol{f}$ is unknown. [MS: 여기서는 MVT, IFT을 사용하지 않았기 때문에 아이디얼 제어가 포화되지 않을 필요성이 없다. 실험에서도 포화를 핸들링하는 중에 추종이 완벽히 되지 못하는 것을 보인다.]

To overcome this issue, a DNN is employed to approximate $\boldsymbol{\tau}^*$. Let $\boldsymbol{\Phi} := \boldsymbol{\Phi}(\boldsymbol{q}_n; \boldsymbol{\theta}) : \mathbb{R}^l \times \mathbb{R}^\Xi \to \mathbb{R}^n$ represent the DNN,

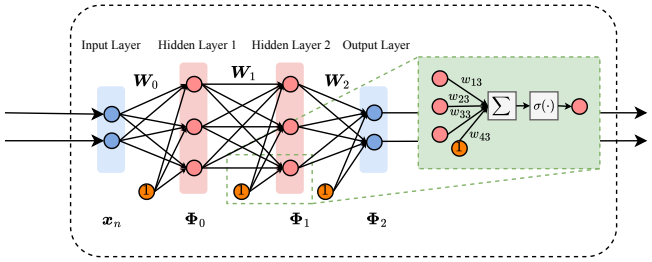


Figure 2: Architecture of the DNN $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ with 2 hidden layers.

where $\boldsymbol{q}_n \in \mathbb{R}^l$ is the DNN input vector, and $\boldsymbol{\theta} \in \mathbb{R}^\Xi$ is the vector of trainable weights. The architecture of $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ will be defined in Section 8. According to the universal approximation theorem for DNNs Kidger and Lyons (2020), $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ can approximate a nonlinear function $\boldsymbol{g}(\cdot)$ with an ideal weight vector $\boldsymbol{\theta}^*$ on a compact subset $\Omega_{NN} \in \mathbb{R}^l$ to $\boldsymbol{\epsilon}$-accuracy, such that $\sup_{\boldsymbol{q}_n\in\Omega_{NN}}\|\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}^*)-\boldsymbol{g}(\cdot)\| = \boldsymbol{\epsilon} < \infty$. Furthermore, the theorem states that the norm of $\boldsymbol{\theta}^*$ is bounded, i.e., $\|\boldsymbol{\theta}^*\| \le \overline{\boldsymbol{\theta}} < \infty$. In this study, $\boldsymbol{\theta}^*$ is defined as a local optimal point, rather than a global optimal point.

In conclusion, the ideal control law $\boldsymbol{\tau}^*$ is expressed by the DNN approximation with the ideal weight vector $\boldsymbol{\Phi}^* := \boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta}^*)$ as follows:

$$\boldsymbol{\tau}^* = \boldsymbol{\Phi}^* + \boldsymbol{\epsilon}, \tag{5}$$

which is estimated online by

$$\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}}, \tag{6}$$

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

Substituting (5) and (6) into (4), the time derivative of V_1 can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}V_1 \le -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\mathrm{sat}(\hat{\boldsymbol{\Phi}}) - \boldsymbol{\Phi}^* - \boldsymbol{\epsilon}). \tag{7}$$

Finally, one can conclude that the result of (4) can be obtained by adapting $\widehat{\boldsymbol{\theta}}$ to $\boldsymbol{\theta}^*$ (i.e., $\hat{\boldsymbol{\Phi}} \to \boldsymbol{\Phi}^*$).

3.2. Deep Neural Network (DNN) Model

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

$$\boldsymbol{\Phi}_i := \begin{cases} \boldsymbol{W}_i^\top \boldsymbol{\phi}_i(\boldsymbol{\Phi}_{i-1}), & i \in [1,\dots,k], \\ \boldsymbol{W}_0^\top \boldsymbol{q}_n, & i = 0, \end{cases} \tag{8}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$ is the weight matrix of the i^{th} layer, and $\boldsymbol{\phi}_i : \mathbb{R}^{l_i} \to \mathbb{R}^{l_i+1}$ represents the activation function of the i^{th} layer. 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. Notice that the output size of $\boldsymbol{\Phi}(\cdot)$ is the same as that of the control input $\boldsymbol{\tau}$ (i.e., $l_{k+1} = n$).

One of the widely used activation functions for large DNNs is from the ReLU family Maas et al. (2013), which effectively avoids 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 activation function. These functions simplify stability analysis due to their continuous differentiability, and their outputs and gradients satisfy $\|\boldsymbol{\phi}_i(\boldsymbol{x})\| < \infty$ and $\|\frac{\mathrm{d}\boldsymbol{\phi}_i(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}}\|_F < \infty$, $\forall \boldsymbol{x} \in \mathbb{R}^{l_i}$. In this study, the hyperbolic tangent function $\tanh(\cdot)$ was selected as the activation function (i.e., $\sigma(x) = \tanh(x)$, $\forall x \in \mathbb{R}$), which provides desirable boundedness with $\|\sigma(x)\| < 1$ and $\|\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}\| < 1$.

For simplicity, each layer's weights are vectorized as $\boldsymbol{\theta}_i := \mathrm{vec}(\boldsymbol{W}_i) \in \mathbb{R}^{\Xi_i}$, 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 $\boldsymbol{\theta}_i$ for all $i \in \{0,\cdots,k\}$ as

$$\boldsymbol{\theta} := \begin{pmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{\theta}_{k-1} \\ \vdots \\ \boldsymbol{\theta}_0 \end{pmatrix} = \begin{pmatrix} \mathrm{vec}(\boldsymbol{W}_k) \\ \mathrm{vec}(\boldsymbol{W}_{k-1}) \\ \vdots \\ \mathrm{vec}(\boldsymbol{W}_0) \end{pmatrix}, \tag{9}$$

where $\Xi = \sum_{i=0}^{k}\Xi_i$ represents the total number of weights. The gradient of $\boldsymbol{\Phi}(\boldsymbol{q}_n;\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is defined as

$$\frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{\theta}_k} & \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{\theta}_{k-1}} & \cdots & \frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{\theta}_0} \end{bmatrix} \in \mathbb{R}^{n\times\Xi} \tag{10}$$

where

$$\frac{\partial\boldsymbol{\Phi}}{\partial\boldsymbol{\theta}_i} = \begin{cases} (\boldsymbol{I}_{l_{k+1}} \otimes \boldsymbol{\phi}_k^\top), & i = k, \\ \boldsymbol{W}_k^\top\boldsymbol{\phi}_k'(\boldsymbol{I}_{l_k} \otimes \boldsymbol{\phi}_{k-1}^\top), & i = k-1, \\ \quad\vdots & \\ \boldsymbol{W}_k^\top\boldsymbol{\phi}_k'\cdots\boldsymbol{W}_1^\top\boldsymbol{\phi}_1'(\boldsymbol{I}_{l_1} \otimes \boldsymbol{q}_n^\top), & i = 0, \end{cases} \tag{11}$$

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

In the following sections, let $\boldsymbol{\Phi}_i^*$ represent the output of the i^{th} layer with the ideal weight vector $\boldsymbol{\theta}^*$. Additionally, define $\boldsymbol{\phi}_i^* := \boldsymbol{\phi}_i(\boldsymbol{\Phi}_{i-1}^*)$ and $\boldsymbol{\phi}_i^{*\prime} := \frac{\partial\boldsymbol{\phi}_i^*}{\partial\boldsymbol{\Phi}_{i-1}^*}$. Similarly, $\hat{\boldsymbol{\Phi}}_i$ denotes the output of the i^{th} layer with the estimated weight vector $\hat{\boldsymbol{\theta}}$, and define $\hat{\boldsymbol{\phi}}_i := \boldsymbol{\phi}_i(\hat{\boldsymbol{\Phi}}_{i-1})$ and $\hat{\boldsymbol{\phi}}_i' := \frac{\partial\hat{\boldsymbol{\phi}}_i}{\partial\hat{\boldsymbol{\Phi}}_{i-1}}$, respectively.

4. Weight Adaptation Laws

4.1. Weight Optimizer Design

The control objective can be represented as follows:

$$\begin{aligned} &\min_{\widehat{\boldsymbol{\theta}}}\ J(\boldsymbol{r};\widehat{\boldsymbol{\theta}}) := \tfrac{1}{2}\boldsymbol{r}^\top\boldsymbol{r} \\ &\text{subject to } c_j(\widehat{\boldsymbol{\theta}}) \le 0, \quad \forall j \in \mathcal{I}, \end{aligned} \tag{12}$$

where $J(\boldsymbol{r};\widehat{\boldsymbol{\theta}}) := \frac{1}{2}\boldsymbol{r}^\top\boldsymbol{r}$ is the objective function. Inequality constraints c_j, $\forall j \in \mathcal{I}$, are imposed during the weight adaptation process to minimize the objective function, where $\mathcal{I}$ denotes the set of the imposed inequality constraints. Here, $\boldsymbol{r}$ is considered

a pre-defined data or parameter for this optimization problem. The Lagrangian function is defined as

$$L(\boldsymbol{r}, \widehat{\boldsymbol{\theta}}, [\lambda_j]_{j\in\mathcal{I}}) := J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}}) + \sum_{j\in\mathcal{I}} \lambda_j c_j(\widehat{\boldsymbol{\theta}}), \tag{13}$$

where λ_j denotes the Lagrange multiplier for each constraint.

The adaptation laws for $\widehat{\boldsymbol{\theta}}$ and $[\lambda]_{j\in\mathcal{I}}$ are derived to solve the dual problem of (12) (i.e., $\min_{\widehat{\boldsymbol{\theta}}} \max_{[\lambda]_{j\in\mathcal{I}}} L(\boldsymbol{r}, \widehat{\boldsymbol{\theta}}, [\lambda]_{j\in\mathcal{I}})$), as follows:

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} = -\alpha \tfrac{\partial L}{\partial \widehat{\boldsymbol{\theta}}} = -\alpha \left(\tfrac{\partial J}{\partial \widehat{\boldsymbol{\theta}}} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \tfrac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right), \tag{14a}$$

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\lambda_j = \beta_j \tfrac{\partial L}{\partial \lambda_j} = \beta_j c_j, \quad \forall j \in \mathcal{I}, \tag{14b}$$

$$\lambda_j = \max(\lambda_j, 0), \tag{14c}$$

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 and J are suppressed for brevity. The Lagrange multipliers associated with inequality constraints are non-negative, i.e., $\lambda_j \geq 0$, and they become zero when their corresponding constraints are inactive. When a constraint c_j becomes active (i.e., violated), the corresponding Lagrange multiplier λ_j increases to address the violation. Once the violation is resolved and the constraint is no longer active (i.e., $c_j < 0$), the multiplier decreases gradually until it returns to zero. Note that this adaption law is similar to the ALM in Nocedal and Wright (2006), where the adaptation law for Lagrange multipliers is given by $\lambda_j \leftarrow \max\left(\lambda_j - \frac{c_j}{\mu}, 0\right)$, with $\mu \in \mathbb{R}_{>0}$ being the penalty parameter.

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 $\lambda_j c_j = 0$ (Nocedal and Wright, 2006, Chap. 12 T. 12.1). In other words, the proposed optimizer updates $\widehat{\boldsymbol{\theta}}$ and λ_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.

4.2. Approximation of the Gradient of Objective Function

The adaptation law for $\widehat{\boldsymbol{\theta}}$ defined in (14a) involves the partial derivative of the state vector with respect to control input $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ (i.e., $\frac{\partial J}{\partial \boldsymbol{\tau}} = ((\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}})(\frac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\theta}}}))^\top \boldsymbol{r} = ((\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}})(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}))^\top \boldsymbol{r}$). Since the objective function depends on $\boldsymbol{r}$ of a dynamic system, obtaining the gradient is not straightforward. The recommended method to calculate the exact value of $\frac{\partial J}{\partial \widehat{\boldsymbol{\theta}}}$ is to use the forward sensitivity method Sengupta et al. (2014) by simulating the sensitivity equation as follows:

$$\tfrac{\mathrm{d}}{\mathrm{d}t}(\tfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}) = \tfrac{\partial}{\partial \boldsymbol{\tau}} \left[\boldsymbol{M}^{-1}(-\boldsymbol{V}_m \boldsymbol{r} - \boldsymbol{K}\boldsymbol{r} + \boldsymbol{f} - \boldsymbol{\tau}_d + \mathrm{sat}(\boldsymbol{\tau})) \right]. \tag{15}$$

However, this method cannot be realized, since we do not have 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 NN's weights are generally large.

In Douratsos and Gomm (2007); Saerens and Soquet (1991), the authors approximate $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ as $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}} \approx [\mathrm{sign}(\frac{\partial r_i}{\partial \tau_j})]_{i,j\in\{1,\cdots,m\}}$ using the sign of each entry (i.e., control direction). However, this method is not suitable for (1), since the control directions are unknown, but the sign of the control input is known (i.e., $\boldsymbol{M}^{-1}$ is positive definite matrix in the view of Property 1). Therefore, we propose to approximate $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}}$ as $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}} \approx \boldsymbol{I}_n$ and one can conclude (14a) as follows:

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} \approx -\alpha \left(\tfrac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}^\top \boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \tfrac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right). \tag{16}$$

4.3. Constraint Candidates

This section introduces conditions which imposed constraints must satisfy and the weight constraints. The potential control input constraints are presented in Appendix Appendix A. First, we introduce the following assumptions for the constraints.

Assumption 2. *The constraint functions $c_j(\widehat{\boldsymbol{\theta}})$, $\forall j \in \mathcal{I}$ are convex in the τ-space and satisfy $c_j(0) \leq 0$ and $c_j(\boldsymbol{\theta}^*) \leq 0$.*

Assumption 3. *The selected constraints satisfy the Linear Independence Constraint Qualification (LICQ) (Nocedal and Wright, 2006, Chap. 12 Def. 12.1).*

Remark 1. *Assumption 2 is not restrictive, since the saturation functions are convex in practice including the origin. Moreover, Assumption 3 is a standard assumption in optimization problems, which ensures that the gradients of the active constraints are linearly independent. This assumption will be used in the stability analysis (see, Lemma 1).*

The following Lemma is introduced for the stability analysis.

Lemma 1. *If Assumptions 2 and 3 are satisfied, the angle between $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}$ and $\widehat{\boldsymbol{\theta}}_k$ is positive when c_j is active, i.e., $\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}^\top \widehat{\boldsymbol{\theta}}_k > 0$.*

Proof. Since $\boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}}$, using (Bernstein, 2009, Proposition 7.1.9), a linear map $\boldsymbol{T}(\cdot) : \widehat{\boldsymbol{\theta}}_k \to \boldsymbol{\tau}$ can be derived:

$$\begin{aligned} \boldsymbol{\tau} = \widehat{\boldsymbol{\Phi}} &= \mathrm{vec}(\widehat{\boldsymbol{W}}_k^\top \widehat{\boldsymbol{\phi}}_k) = (\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top \mathrm{vec}(\widehat{\boldsymbol{W}}_k) \\ &= \underbrace{(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top}_{:=\boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)} \widehat{\boldsymbol{\theta}}_k = \boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)\widehat{\boldsymbol{\theta}}_k. \end{aligned} \tag{17}$$

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

□

The weight constraints are essential to prevent the weights from diverging during the adaptation process. The weight constraints $\boldsymbol{c}_\theta := [c_{\theta_i}]_{i\in\{0,\cdots,k\}} \in \mathbb{R}^{k+1}$ are defined for each layer's weight as follows:

$$c_{\theta_i} = \tfrac{1}{2}(\|\widehat{\boldsymbol{\theta}}_i\|^2 - \overline{\theta}_i^2) \leq 0 \tag{18}$$

with $\overline{\theta}_i < \infty$ denoting the maximum allowable norm for $\widehat{\boldsymbol{\theta}}_i$. The gradient of $\boldsymbol{c}_\theta$ with respect to $\widehat{\boldsymbol{\theta}}$ is given by

$$\tfrac{\partial c_{\theta_i}}{\partial \widehat{\boldsymbol{\theta}}_j} = \begin{cases} \widehat{\boldsymbol{\theta}}_i, & \text{if } i = j, \\ \boldsymbol{0}, & \text{if } i \neq j. \end{cases} \tag{19}$$

Note that $\boldsymbol{c}_\theta$ satisfies Assumption 2 and 3 invoking Lemma 1 and (19), respectively.

5. 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 $\tilde{\boldsymbol{\theta}}_i := \widehat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i^*$. The following lemma is introduced for the stability analysis.

Lemma 2. *If $c_j(\widehat{\boldsymbol{\theta}})$, $\forall j \in \mathcal{I} \setminus \{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}$ 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.*

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

$$\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i} = \frac{\partial c_j}{\partial \boldsymbol{\tau}} \frac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\Phi}}} \frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i} \tag{20}$$

where $\frac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\Phi}}} = \boldsymbol{I}_n$, which is bounded. From the linear mapping in (17), $\boldsymbol{\tau}$ is bounded as long as $\widehat{\boldsymbol{\theta}}_k$ is bounded (by the condition of the lemma), and $\|\boldsymbol{\phi}_k(\cdot)\|$ is bounded due to the properties of the activation functions. By Assumption 2, the function c_j is convex. The convex function has a bounded derivative with respect to $\boldsymbol{\tau}$, since $\boldsymbol{\tau}$ is a bounded variable (i.e., $\frac{\partial c_j}{\partial \boldsymbol{\tau}}$ is bounded). Furthermore, $\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}_i}$ 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 (11). Consequently, $\|\frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_i}\|$, $\forall j \in \mathcal{I} \setminus \{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}$, is bounded, when $\widehat{\boldsymbol{\theta}}_i$, $\forall i \in [k,\cdots,i+1]$ are bounded.

□

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

Theorem 1. *For the dynamical system in (1), the neuro-adaptive controller (6) and weight adaptation laws (14) ensure the boundedness of the filtered error $\boldsymbol{r}$ and the weight estimate $\widehat{\boldsymbol{\theta}}$. This holds with the weight norm constraint (18) and input constraints satisfying Assumption 2 and 3.*

Proof. The boundednesses of $\widehat{\boldsymbol{\theta}}$ and $\boldsymbol{r}$ are analyzed recursively from the last k^{th} layer to the first layer of $\widehat{\boldsymbol{\Phi}}$. Without loss of generality, weight constraint c_{θ_i}, $\forall i \in [0,\cdots,k]$ is supposed to be active. This assumption is reasonable, since even if the constraint is inactive [MS: Add the reason why it is reasonable.] We consider two cases: (C_1) control input constraints are active, and (C_2) control input constraints are inactive. Using the fact that amplitude of the control input is only depends on $\widehat{\boldsymbol{\theta}}_k$ (i.e., see (6), (8) and $\|\boldsymbol{\phi}_k\| < \infty$), the boundedness of inner layers' weights will be verified after the boundedness of $\widehat{\boldsymbol{\theta}}_k$ and $\boldsymbol{r}$ is confirmed.

Case (C_1): Control input constraints are active.

As the result of the time derivative of V_1, the invariant set $\Theta_{\boldsymbol{r}}^{c_1}$ of $\boldsymbol{r}$ (i.e., if $\boldsymbol{r}$ leaves $\Theta_{\boldsymbol{r}}^{c_1}$, $\frac{\mathrm{d}}{\mathrm{d}t}V_1$ is negative) can be obtained as w

$$\Theta_{\boldsymbol{r}}^{c_1} = \left\{\boldsymbol{r} \in \mathbb{R}^n \mid \|\boldsymbol{r}\| \le \tfrac{2\overline{\tau}+\overline{\tau}_d}{\lambda_{\min}(\boldsymbol{K})}\right\}. \tag{21}$$

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:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_2 =& -\widehat{\boldsymbol{\theta}}_k^\top((\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)^\top \boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}) \\
=& -\widehat{\boldsymbol{\theta}}_k^\top((\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)^\top \boldsymbol{r} + \lambda_{\theta_k}\widehat{\boldsymbol{\theta}}_k \\
&+ \textstyle\sum_{j\in\mathcal{I}\setminus\{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}) \\
\le& -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + \underbrace{\|(\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)\|}_{:=c_1\in\mathbb{R}_{>0}}\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\| \\
&- \underbrace{\textstyle\sum_{j\in\mathcal{I}\setminus\{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}} \lambda_j \widehat{\boldsymbol{\theta}}_k^\top \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}}_{:=c_2>0,\text{ by Lemma 1 and Assumption 3}} \\
\le& -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + c_1\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\|.
\end{aligned} \tag{22}$$

According to (22), the invariant set of $\widehat{\boldsymbol{\theta}}_k$ is defined as

$$\Theta_{\theta_k}^{c_1} = \left\{\widehat{\boldsymbol{\theta}}_k \in \mathbb{R}^{\Xi_k} \mid \|\widehat{\boldsymbol{\theta}}_k\| \le \tfrac{c_1(2\overline{\tau}+\overline{\tau}_d)}{\lambda_{\theta_k}\lambda_{\min}(\boldsymbol{K})}\right\}. \tag{23}$$

The satisfaction of the constraints can be verified by the boundedness of $\widehat{\boldsymbol{\theta}}_k$. For c_{θ_k}, if the corresponding Lagrange multiplier λ_{θ_k} increases infinitely by the constraint violation, $\widehat{\boldsymbol{\theta}}_k$ approaches to the origin as $\Theta_{\widehat{\boldsymbol{\theta}}_k}^{c_1}$ is squeezed to a point. The satisfactions of the remaining control input constraints can be verified implicitly using c_2 in (22). Similarly to c_{θ_k}, as the control input constraint c_j, $\forall j \in \mathcal{I} \setminus \{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}$ is violated, the corresponding Lagrange multiplier λ_j increases infinitely, so does c_2. It makes c_2 dominates the right-hand side of (22), which leads $\frac{\mathrm{d}}{\mathrm{d}t}V_2$ negative definite. This drives $\widehat{\boldsymbol{\theta}}_k$ to the origin until the constraint is satisfied (i.e., by Assumption 2 and Lemma 1, the control input constraints can be considered as convex constraint in $\widehat{\boldsymbol{\theta}}_k$-space).

Therefore, the boundedness of $\widehat{\boldsymbol{\theta}}$ and $\boldsymbol{r}$ are guaranteed by the invariant sets $\Theta_{\boldsymbol{r}}^{c_1}$ and $\Theta_{\boldsymbol{\theta}_k}^{c_1}$ and satisfactions of c_j, $\forall j \in \mathcal{I}$ are confirmed.

Case (C_2): Control input constraints are inactive.

Since, here, the control input constraints are inactive, sat($\cdot$) in (7) and c_j, $\forall j \in \mathcal{I}$ in (22) can be removed. Consider the Lyapunov function candidate $V_3 := V_1 + V_2$ and its time derivative:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_3 =& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \overline{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\widehat{\boldsymbol{\Phi}} - \boldsymbol{\Phi}^* - \boldsymbol{\epsilon}) \\
&- \widehat{\boldsymbol{\theta}}_k^\top((\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)^\top \boldsymbol{r} + \lambda_{\theta_k}\widehat{\boldsymbol{\theta}}_k) \\
\le& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \boldsymbol{r}^\top\widehat{\boldsymbol{\Phi}} + (\overline{\tau}_d + \underbrace{\|\boldsymbol{\Phi}^* + \boldsymbol{\epsilon}\|}_{=\boldsymbol{\tau}^*})\|\boldsymbol{r}\| \\
&- \widehat{\boldsymbol{\Phi}}^\top\boldsymbol{r} - \lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 \\
\le& -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + (\overline{\tau}_d + \overline{\tau})\|\boldsymbol{r}\| - \lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2.
\end{aligned} \tag{24}$$

Like in Case (C_1), the boundedness of $\boldsymbol{r}$ can be verified by the invariant sets $\Theta_{\boldsymbol{r}}^{c_2}$ defined as

$$\Theta_{\boldsymbol{r}}^{c_2} = \left\{\boldsymbol{r} \in \mathbb{R}^n \mid \|\boldsymbol{r}\| \le \tfrac{\overline{\tau}_d+\overline{\tau}}{\lambda_{\min}(\boldsymbol{K})}\right\}. \tag{25}$$

The boundedness of $\widehat{\boldsymbol{\theta}}_k$ also can be ensured by the invariant set $\Theta_{\widehat{\boldsymbol{\theta}}_k}^{c_2} = \{\widehat{\boldsymbol{\theta}}_k \in \mathbb{R}^{\Xi_k} \mid \|\widehat{\boldsymbol{\theta}}_k\| \le \overline{\theta}_k\}$, as λ_{θ_k} exists positive until c_{θ_k} is satisfied.

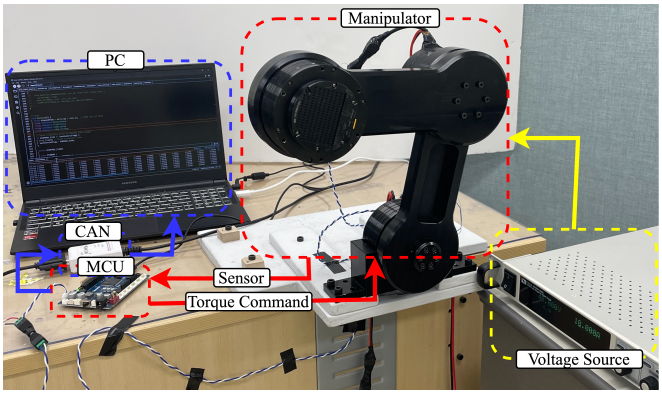


Figure 3: Robot setup for the real-time experiment.

The boundedness of inner layers' weights and satisfactions of their constraints c_j, $\forall j \in \{\theta_i\}_{i\in\{k-1,\cdots,0\}}$ can be shown recursively, using (Desoer and Vidyasagar, 2009, Chap. 4 T. 1.9) The dynamics of $\widehat{\boldsymbol{\theta}}_i$, $\forall i \in \{k-1,\cdots,0\}$ are represented as

$$\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}}_i = -\alpha\left(\frac{\partial\widehat{\boldsymbol{\Phi}}}{\partial\widehat{\boldsymbol{\theta}}_i}^{\top}\boldsymbol{r} + \lambda_{\theta_i}\widehat{\boldsymbol{\theta}}_i + \sum_{j\in\mathcal{I}\setminus\{\boldsymbol{\theta}_i\}_{i\in\{0,\cdots,k\}}}\lambda_j\frac{\partial c_j}{\partial\widehat{\boldsymbol{\theta}}_i}\right). \tag{26}$$

Invoking Lemma 2, $\widehat{\boldsymbol{\theta}}_i$ is bounded, provided that $\widehat{\boldsymbol{\theta}}_i$, $\forall i \in \{k,\cdots,i+1\}$ are bounded, since the system matrix $-\lambda_{\theta_i}\boldsymbol{I}_{\Xi_i}$ is stable and the residual terms are bounded. Therefore, starting from $(k-1)^{\text{th}}$ layer, the boundedness of $\widehat{\boldsymbol{\theta}}_i$ can be established recursively down to the input layer ($i = 0$).

Finally, the filtered error $\boldsymbol{r}$ and the estimated weights $\widehat{\boldsymbol{\theta}}$ are bounded and the imposed constraints c_j, $\forall j \in \mathcal{I}$ are satisfied. □

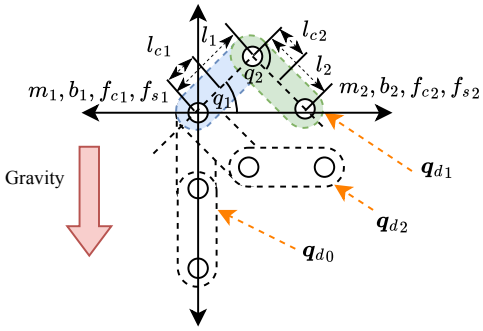
6. Real-Time Implementation and Validation

The proposed CONAC was validated via real-time experiment on a two-link manipulator.

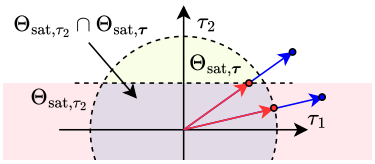
6.1. Validation Setup

To validate the effectiveness of the proposed CONAC, a real-time experiment was conducted on a two-link manipulator, as shown in Fig. 3. The OpenCR1.0 ROBOTIS (2016) with a 32-bit ARM Cortex and a 216 MHz clock frequency was used for the real-time implementation. The experimental setup consisted of a two-link manipulator with a 2-DOF arm, equipped with two @@@@@ servos. The servos were controlled using the OpenCR1.0 board, which was connected to a computer via USB. The reference trajectory was generated using the same method as in the numerical simulation.

The two-link manipulator model is depicted in Fig. 4a, adapted from Markus et al. (2013). [MS: CORRECT HERE] The system dynamics of the two-link manipulator is described in (1), and for the numerical validation, the parameters are assumed to be constant such that $\boldsymbol{M} = \mathrm{diag}([m_1, m_2])$, $\boldsymbol{V}_m = \mathrm{diag}([b_1, b_2])$, and $\boldsymbol{F} = \mathrm{diag}([f_{c_1}, f_{s_1}, f_{c_2}, f_{s_2}])$, where the parameters $q_p, q_{d_p}, \tau_p, m_p, l_p, l_{cp}, b_p, f_{cp}$ and f_{sp} denote the joint angle, desired joint angle, torque, mass, length, center of mass, viscous friction coefficient, coulomb friction coefficient, and static friction coefficient, respectively, for link $p \in [1, 2]$. The values of the system model parameters are provided in Table 1.



(a) Two-link manipulator model.



(b) Control input saturation function.

Figure 4: Two-link manipulator model and control saturation function.

Table 1: Two-link manipulator parameters of p^{th} link.

Symbol	Description	Link 1	Link 2
m_p	Mass	2.465 (kg)	2.465 (kg)
l_p	Length	0.2 (m)	0.18 (m)
l_{cp}	Center of mass	0.139 (m)	0.087 (m)
b_p	Viscous friction coef.	0.1 (Nms)	0.1 (Nms)
f_{cp}	Coulomb friction coef.	0.24 (Nm)	0.24 (Nm)
f_{sp}	Static friction coef.	0.3 (Nm)	0.2 (Nm)
I_p	Inertia	0.069 (kgm^2)	0.015 (kgm^2)

The desired trajectory for $\boldsymbol{q} = (q_1, q_2)^{\top}$ was generated using fifth-order polynomial functions $\mathrm{Poly}^5(\boldsymbol{x}_1, \boldsymbol{x}_2, t_f, t)$, to satisfy the initial $\boldsymbol{x}_1$ and final $\boldsymbol{x}_2$ angle conditions and zero velocities of the joints, where t_f is the time duration. The desired trajectory is defined as follows:

$$\boldsymbol{q}_d(t) = \begin{pmatrix} q_{d1} \\ q_{d2} \end{pmatrix} = \begin{cases} \mathrm{Poly}^5(\boldsymbol{q}_{d0}, \boldsymbol{q}_{d1}, t_f, t), & \text{if } 0 \le t < t_f, \\ \mathrm{Poly}^5(\boldsymbol{q}_{d1}, \boldsymbol{q}_{d2}, t_f, t - t_f), & \text{if } t_f \le t < 2t_f, \\ \mathrm{Poly}^5(\boldsymbol{q}_{d2}, \boldsymbol{q}_{d1}, t_f, t - 2t_f), & \text{if } 2t_f \le t < 3t_f. \\ \mathrm{Poly}^5(\boldsymbol{q}_{d1}, \boldsymbol{q}_{d0}, t_f, t - 3t_f), & \text{if } 3t_f \le t < 4t_f, \end{cases} \tag{27}$$

where $\boldsymbol{q}_{d0} := \left(-\frac{1}{2}\pi, 0\right)^{\top}$, $\boldsymbol{q}_{d1} := \left(\frac{1}{4}\pi, -\frac{1}{2}\pi\right)^{\top}$, $\boldsymbol{q}_{d2} := \left(-\frac{1}{4}\pi, \frac{1}{4}\pi\right)^{\top}$, and $t_f = 3$ s. To demonstrate the learning capa-

bility of NN, the desired trajectory was applied twice; let us call the first and second iterations of the reference trajectory as Episode 1 and Episode 2, respectively.

To demonstrate the effectiveness of the proposed CONAC to handle the combination of the convex input constraints, the saturation function is defined to project the control input onto following convex domain $\Theta_{\text{sat}} := \{\Theta_{\text{sat},\boldsymbol{\tau}} \cap \Theta_{\text{sat},\tau_2}\}$ as depicted in Fig. 4b, where

$$\Theta_{\text{sat},\boldsymbol{\tau}} := \{\boldsymbol{\tau} \mid \|\boldsymbol{\tau}\| \leq 10\}, \tag{28a}$$

$$\Theta_{\text{sat},\tau_2} := \{\tau_2 \mid |\tau_2| \leq 2.5\}. \tag{28b}$$

These domains (28a) and (28b) can be physically interpreted as a combination of the voltage source limitation of the actuators and the torque limitation of the second link actuator, respectively.

For a comparative study, we implemented 4 controllers. The properties of the controllers are summarized in Table 2.

Controller 1: (C_1) CONAC; proposed

To handle the input saturation, the proposed CONAC was implemented with the input bound constraint for second link $c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$ (A.1) and the nonlinear input norm constraint $c_{\boldsymbol{\tau}}$ (A.3), as follows:

$$c_{\overline{\tau}_2} = \tau_2 - \overline{\tau}_2 \leq 0,\ c_{\underline{\tau}_2} = \underline{\tau}_2 - \tau_2 \leq 0,\ c_{\boldsymbol{\tau}} = \tfrac{1}{2}(\|\boldsymbol{\tau}\| - \overline{\tau})^2 < 0, \tag{29}$$

where $\overline{\tau}_2 = -\underline{\tau}_2 = 2.5$ are the absolute bound of the second link actuator, and $\overline{\tau} = 10$ is the maximum feasible norm of the control input. Hence, (C_1) is possible to use full capability of the actuator. The weight constraints $c_{\theta_i},\ \forall i \in [0, \cdots, k]$ were also considered in the CONAC. As described in (6), the control input is equal to the output of the NN, and the adaptation law is defined in (14).

The update rate of the Lagrange multipliers was set to $\beta_{\theta_i} = 10^{-3},\ \forall i \in [0, \cdots, k]$, and $\beta_{\boldsymbol{\tau}} = 10$ and $\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 10^3$.

Controller 2: (C_2) CONAC with auxiliary system

On the other hand, the existing auxiliary system Esfandiari et al. (2014); Karason and Annaswamy (1994); Esfandiari et al. (2015) was used to compare with the proposed (C_1). The auxiliary system is defined as $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\zeta} = \boldsymbol{A}_{\zeta}\boldsymbol{\zeta} + \boldsymbol{B}_{\zeta}\Delta\boldsymbol{\tau},\ \boldsymbol{\zeta}|_{t=0} = \boldsymbol{0}_{2\times 1}$, where $\boldsymbol{\zeta} \in \mathbb{R}^n$ denotes the auxiliary state, $\boldsymbol{A}_{\zeta} \in \mathbb{R}^{n\times n}_{<0}$, and $\boldsymbol{B}_{\zeta} \in \mathbb{R}^{n\times n}$. The auxiliary state $\boldsymbol{\zeta}$ is affected by the saturated control input $\Delta\boldsymbol{\tau}$, whose entry is defined as $\Delta\tau_i = \tau_i - \max(\underline{\tau}_i, \min(\overline{\tau}_i, \tau_i)),\ \forall i \in \{1, 2\}$. Since, the auxiliary system only considers the input bound constraint, $\overline{\tau}_1$ is set to 9.682 to approximately match the maximum allowable norm of the control input $\overline{\tau}$, (i.e., $\overline{\tau}_1^2 + \overline{\tau}_2^2 = \overline{\tau}^2$, and $\overline{\tau}_2 = 2.5$ and $\overline{\tau} = 10$).

The auxiliary state are used in the adaptation law (16) by substituting $\boldsymbol{r}$ with $\boldsymbol{r} + \boldsymbol{\zeta}$. Hence, the weights are adapted to reduce not only the tracking error $\boldsymbol{r}$ but also the auxiliary state $\boldsymbol{\zeta}$.

The matrices of the auxiliary system are set to $\boldsymbol{A}_{\zeta} = \text{diag}([-10, -10])$ and $\boldsymbol{B}_{\zeta} = \text{diag}([10^3, 10^3])$, considering the tradeoff between the sensitivity of the control saturation, and the convergence speed. [MS: sensitivity와 convergence에 대해서 더 설명이 필요한가?]

Table 2: Properties of the controllers used in simulation

	Description	Input Constraint
(C_1)	proposed	$\overline{\tau} = 10, \overline{\tau}_2 = -\underline{\tau}_2 = 2.5$
(C_2)	aux. sys.	$\overline{\tau}_1 = -\underline{\tau}_1 = 9.682, \overline{\tau_2} = -\underline{\tau_2} = 2.5$
(C_3)	small β_i	$\overline{\tau} = 10, \overline{\tau}_2 = -\underline{\tau}_2 = 2.5$
(C_4)	no $\boldsymbol{\tau}$ constraints	$[\beta_j]_{j\in\{\boldsymbol{\tau},\overline{\tau}_2,\underline{\tau}_2\}} = \boldsymbol{0}$

Table 3: Quantitative comparison of performances' L_2 norm.

Episode 1				
	(C_1)	(C_2)	(C_3)	(C_4)
r_1	0.389	0.403	0.370	0.370
r_2	0.426	0.456	0.427	0.370
Episode 2				
	(C_1)	(C_2)	(C_3)	(C_4)
r_1	0.389	0.403	0.370	0.370
r_2	0.426	0.456	0.427	0.370

Controller 3: (C_3) CONAC with small β_i

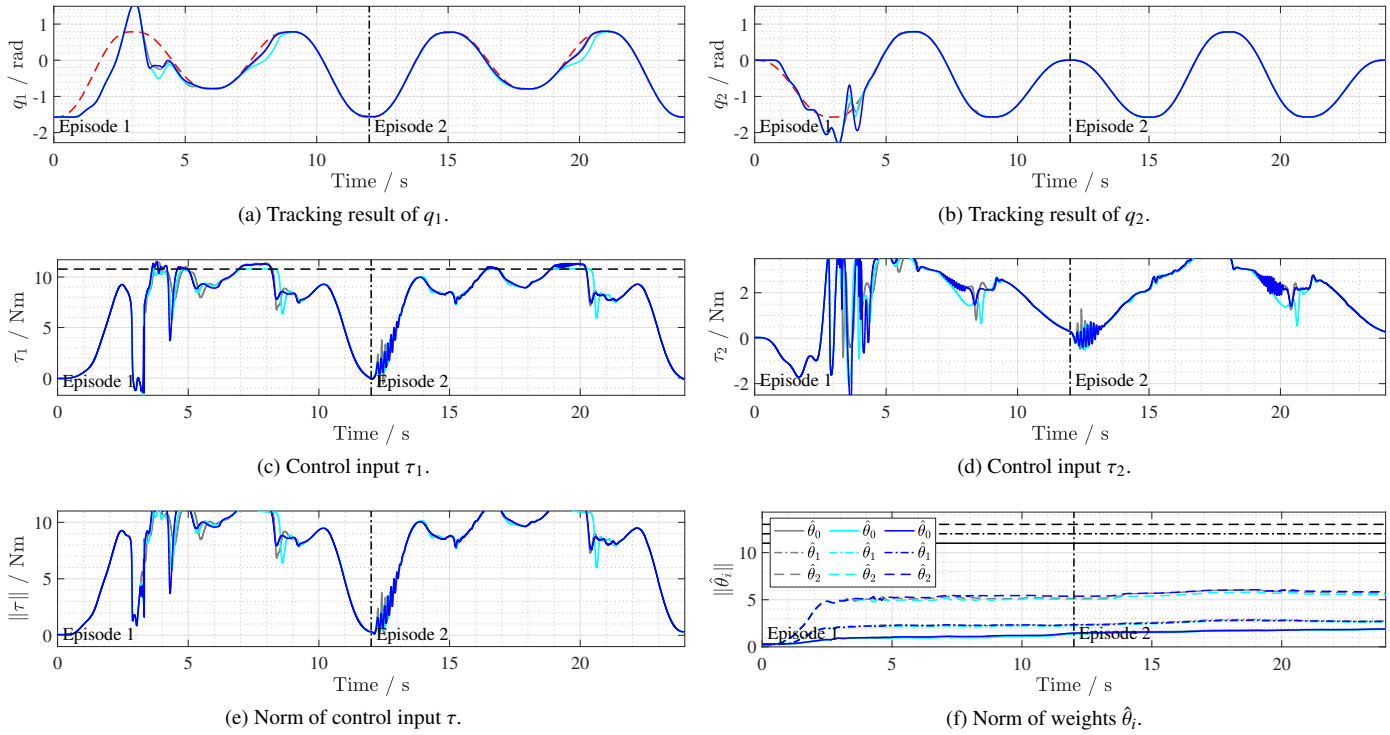
To investigate the effect of the update rate of Lagrange multipliers, (C_3) was implemented with small β_i values, which are set to $\beta_{\boldsymbol{\tau}} = 1$, and $\beta_{\overline{\tau}_2} = \beta_{\underline{\tau}_2} = 10^2$. The remaining parameters are the same as (C_1).

Controller 4: (C_4) CONAC without control input constraints

The controller 4 is the same as (C_1) but without the control input constraints, to show the effect of the control input constraints. In this case, the update rate of the Lagrange multipliers regarding the control input constraints i.e., $\beta_{\boldsymbol{\tau}}, \beta_{\overline{\tau}_2}, \beta_{\underline{\tau}_2}$ was set to 0, so that $[\lambda_j]_{j\in\{\boldsymbol{\tau},\overline{\tau}_2,\underline{\tau}_2\}}$ are not updated.

For all controllers ((C_1), (C_2), and (C_3)), the NN input vector $\boldsymbol{q}_n$ was set as $\boldsymbol{q}_n = (\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top, \boldsymbol{r}^\top, 1)^\top \in \mathbb{R}^{3n+1}$ with the augmented scalar 1 included to account for the bias term in the weight matrix. The NNs' weights were initialized randomly with uniform distribution in the range of $[-0.1, 0.1]$, and the same random seed. Each NN architecture had two hidden layers with four nodes (i.e., $k = 2, l_0 = 6, l_1 = 4, l_2 = 4, l_3 = 2$), and the adaptation gain was set to $\alpha = 0.5$. The filtering matrix $\boldsymbol{\Lambda}$ was set to $\boldsymbol{\Lambda} = \text{diag}([5, 15])$. The sampling rate of the controller was 250 Hz considering the real-time implementation, and 1000 Hz for the simulation. [MS: CAN Hz?] To emphasize the control input constraint handling capability, the weight constraint parameters were selected sufficiently large (i.e., $\overline{\boldsymbol{\theta}}_0 = 11, \overline{\boldsymbol{\theta}}_1 = 12$, and $\overline{\boldsymbol{\theta}}_2 = 13$), avoiding the violation of the weight constraints.

The quantitative comparison of the tracking performance was evaluated using the L_2-norm of them, (i.e., $\sqrt{\int_0^T \|h\|\,\mathrm{d}t}$, where $h \in \{r_1, r_2\}$ and $T \in \mathbb{R}_{>0}$ denotes the each episode duration time.



(a) Tracking result of q_1.

(b) Tracking result of q_2.

(c) Control input τ_1.

(d) Control input τ_2.

(e) Norm of control input τ.

(f) Norm of weights $\hat{\theta}_i$.

Figure 5: Simulation results of (C_1) (blue), (C_2) (cyan), (C_3) (gray), and reference signal of $\boldsymbol{q}_d$ (red dashed line).

6.1.1. Tracking Performance

The simulation results are shown in Fig. 5 and the quantitative comparison of the tracking performance is given in Table 3. Across the two episodes, all controllers enhanced their tracking performances about 60.2% for (C_1), 52.7% for (C_2), and 61.9% for (C_3) in L_2 norm of r_1, and 84.5% for (C_1), 79.3% for (C_2), and 80.5% for (C_3) in L_2 norm of r_2, respectively, as shown in Table 3. Qualitatively, as described in Fig. 5a and Fig. 5b, the oscillations were suppressed in the second episode, as well. This results lead to the conclusion that the NNs in all controllers were able to learn the ideal control input $\boldsymbol{\tau}^*$ properly, by the filtered error $\boldsymbol{r}$, despite the fact that $\boldsymbol{\tau}^*$ was not known in advance. Moreover, since the learning began from the random initialization of the weights and no prior knowledge of the system, the online learning capability of the proposed CONAC was demonstrated.

In addition, it is notable that in both episodes, all controllers failed to track the desired trajectory of q_1 in time interval from 7.5 s to 8.5 for Episode 1, and from 19.5 s to 20.5 s for Episode 2. This is because the control inputs were saturated in the time intervals, as shown in Fig. 5c and Fig. 5e. However,

The tracking performance of CONAC, (i.e., (C_1), and (C_3)), outperformed (C_2) in both episodes, as shown in Table 3. This is because, (as wil be discussed in Section 6.1.2) (C_1) was able to use the full capability of the actuator, while (C_2) was limited its first link actuator due to the feature of the auxiliary system.

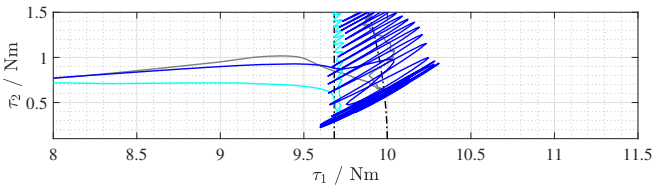
6.1.2. Constraint Handling

In this section, the constraint handling capability of all controllers are investigated. First, the weight norm constraints c_{θ_i}, $\forall i \in \{0, 1, 2\}$ were not violated, as shown in Fig. 5f, since the limits $\overline{\theta}_i$, $\forall i \in \{0, 1, 2\}$ were set sufficiently large.

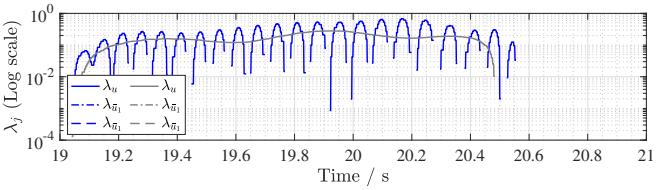
As shown in Fig. 5c, Fig. 5d, and Fig. 5e, all controllers satisfied the control input saturation illustrated in Fig. 4b, with $c_{\boldsymbol{\tau}}, c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$ for (C_1) and (C_3), and $c_{\overline{\tau}_1}, c_{\underline{\tau}_1}, c_{\overline{\tau}_2}$, and $c_{\underline{\tau}_2}$ for (C_2), as the weights were adapted to reduce the constraint violations, using Lagrange multipliers λ_j, $\forall j \in \mathcal{I}$, and auxiliary state $\boldsymbol{\zeta}$, respectively. It is worth to note that the control input constraints were violated repeatedly, since the controllers were implemented in discrete time with a low sampling rate of 250 Hz.

Even though all controllers satisfied the input saturation, (C_2) was not able to use the full capability of the actuator, as shown in Fig. 6a which shows the bird-eye view of the control input $\boldsymbol{\tau}$ in time interval from 19 s to 21 s. This is because, as aforementioned, the auxiliary system only considers the input bound constraint, and the control input bound constraint for first link was imposed to approximately match the maximum allowable norm of the control input $\overline{\tau}$, instead of $c_{\boldsymbol{\tau}}$. In contrast, (C_1) and (C_3) were able to use the full capability of the actuator with the combination of the input bound constraints $c_{\overline{\tau}_2}$ and $c_{\underline{\tau}_2}$, and the nonlinear input norm constraint $c_{\boldsymbol{\tau}}$ which is completely match the feasible domain of the control input Θ_{sat}.

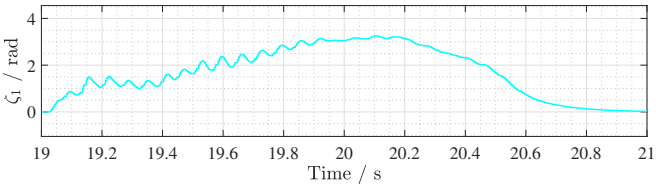
The detail of the constraint handling in the time interval from 19 s to 21 s, is now investigated. First, for (C_2), the auxiliary state ζ_1 generated by constraint violation of $c_{\overline{\tau}_1}$. As the weights were adapted to reduce not only $\boldsymbol{r}$ but also $\boldsymbol{\zeta}$, the constraint violation of $c_{\overline{\tau}_1}$ was reduced, as shown in Fig. 6a. After



(a) Control input τ in time interval from 19 s to 21 s (bird's eye view).



(b) Lagrange multipliers λ_j in time interval from 19 s to 21 s.



(c) Auxiliary states in time interval from 19 s to 21 s.

Figure 6: Constraint handling process of (C_1) (blue), (C_2) (cyan), and (C_3) (gray) in time interval from 19 s to 21 s.

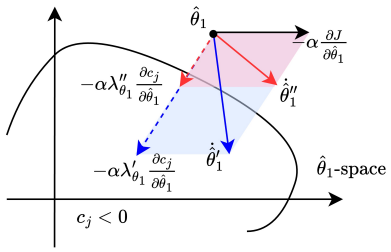


Figure 7: The effect of Lagrange multiplier λ_j on the adaptation direction of $\widehat{\boldsymbol{\theta}}_1$. The notation $(\cdot')$ and $(\cdot'')$ represent two different cases of bigger and smaller Lagrange multiplier, respectively (i.e., $\lambda'_j > \lambda''_j$).

the constraint was satisfied, the auxiliary state ζ_1 converged to zero, by the stable state matrix $\boldsymbol{A}_\zeta$ in the auxiliary system, i.e., $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\zeta} = \boldsymbol{A}_\zeta\boldsymbol{\zeta} + \boldsymbol{B}_\zeta\Delta\boldsymbol{\tau}$ with $\Delta\boldsymbol{\tau} = \boldsymbol{0}_{2\times1}$.

For the cases of (C_1) and (C_3), the Lagrange multipliers $\lambda_{\boldsymbol{\tau}}$ adjusted the adaptation direction of the weights $\widehat{\boldsymbol{\theta}}_1$ to reduce the constraint violation $c_{\boldsymbol{\tau}}$, as $\lambda_{\boldsymbol{\tau}}$ increased by the constraint violations. For more details, the effect of the update rate β_j, $\forall j \in \mathcal{I}$ of Lagrange multipliers is shown in Fig. 7 for convex constraint c_j, $\forall j \in \mathcal{I}$ and two different cases of λ_j. With the bigger λ_j, the adaptation of $\widehat{\boldsymbol{\theta}}_1$ is more assertive than that with smaller λ_j. This leads to a faster satisfaction of the constraint c_j, $\forall j \in \mathcal{I}$, but this also causes oscillation in the adaptation process. In Fig. 6a and Fig. 6b, λ_j of (C_1) repeatedly increased and decreased, which indicates that the constraint c_j, $\forall j \in \mathcal{I}$ was satisfied and violated, respectively, in the discrete time implementation. Besides, with small λ_j, the adaptation direction of $\widehat{\boldsymbol{\theta}}_1$ is adjusted more conservatively, which leads to a slower satisfaction of the constraint c_j, $\forall j \in \mathcal{I}$, as shown in Fig. 6a and the

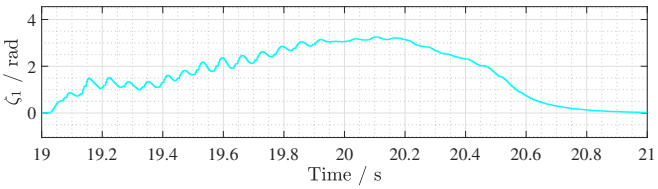


Figure 8: Computational time of CONAC and CONAC-AUX.

positively existence of λ_j in Fig. 6b. This allows the controller to show a smoother control input. However, since the weights were adapted to reduce the filtered error $\boldsymbol{r}$ while the control input was saturated, the weights were not able to effectively adapt the weights with respect to the filtered error $\boldsymbol{r}$.

6.1.3. *Satisfaction of Karsh-Kuhn-Tucker (KKT) conditions*

The KKT conditions defined in (Nocedal and Wright, 2006, Def. @@) were satisfied in the simulation, as shown in Fig. 6b, where the Lagrange multipliers λ_j, $\forall j \in \mathcal{I}$ were positive and the constraint violations c_j, $\forall j \in \mathcal{I}$ were negative.

6.1.4. *Computational Time*

is shown in Fig. 8. The computational time of CONAC was 0.5 ms, which is less than the sampling time of 4 ms. This indicates that the proposed CONAC can be implemented in real-time applications.

7. Conclusion

This paper presented a constrained optimization-based neuro-adaptive controller (CONAC) for the uncertain Euler-Lagrange system, addressing both weight norm and input constraints through a rigorous optimization framework. The stability of the proposed controller was analyzed using Lyapunov theory, ensuring that the system maintained bounded tracking and estimation errors under real-time adaptation.

The controller effectively incorporated both the input (bound or norm) constraint and the weight norm constraint, ensuring that both actuator limitations and neural network weights were kept within predefined bounds. By formulating these constraints as part of the optimization process, CONAC ensured that the weights converged in a way that satisfied the Karush-Kuhn-Tucker (KKT) conditions, guaranteeing optimality and stability.

Simulation results validated the superior performance of CONAC compared to conventional methods, such as DNN-BSC and DNN-BSC-A. CONAC not only handled complex input constraints but also managed the weight norm constraints rigorously, leading to improved tracking accuracy and stability without notable oscillations.

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

Appendix A. Input Constraint Candidates

This section introduces potential weight and input constraints that can be used in the proposed neuro-adaptive controller.

Appendix A.1. Input Bound Constraint

Most physical systems have control input limits due to electrical and mechanical limitations. 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

$$c_{\overline{\tau}_i} = \tau_{(i)} - \overline{\tau}_i \leq 0, \quad c_{\underline{\tau}_i} = \underline{\tau}_i - \tau_{(i)} \leq 0, \tag{A.1}$$

with $\overline{\tau}_i$ and $\underline{\tau}_i$ 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

$$\begin{aligned}
\frac{\partial \boldsymbol{c}_{\overline{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} &= \begin{bmatrix} \frac{\partial c_{\overline{\tau}_1}}{\partial \widehat{\boldsymbol{\theta}}}^\top \\ \vdots \\ \frac{\partial c_{\overline{\tau}_n}}{\partial \widehat{\boldsymbol{\theta}}}^\top \end{bmatrix} = +\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = +\begin{bmatrix} (I_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top) & \cdots & (\cdot) \end{bmatrix} \in \mathbb{R}^{n\times\Xi}, \\
\frac{\partial \boldsymbol{c}_{\underline{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} &= \begin{bmatrix} \frac{\partial c_{\underline{\tau}_1}}{\partial \widehat{\boldsymbol{\theta}}}^\top \\ \vdots \\ \frac{\partial c_{\underline{\tau}_n}}{\partial \widehat{\boldsymbol{\theta}}}^\top \end{bmatrix} = -\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = -\begin{bmatrix} (I_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top) & \cdots & (\cdot) \end{bmatrix} \in \mathbb{R}^{n\times\Xi}.
\end{aligned} \tag{A.2}$$

Appendix A.2. Input Norm Constraint

Consider the control input τ as the torque of each actuator 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:

$$c_{\boldsymbol{\tau}} = \|\boldsymbol{\tau}\|^2 - \overline{\tau}^2 \leq 0, \tag{A.3}$$

with $\overline{\tau} \in \mathbb{R}_{>0}$ denoting the maximum allowable control input magnitude. This input norm constraint is also commonly applied in current and torque control problems for electric motors Choi et al. (2024). The gradients of $c_{\boldsymbol{\tau}}$ with respect to $\widehat{\boldsymbol{\theta}}$ are given by

$$\frac{\partial \boldsymbol{c}_{\boldsymbol{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} = \sum_{i=1}^{n} 2\tau_{(i)} \left(\mathrm{row}_i(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}})\right)^\top = \frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}^\top \boldsymbol{\tau} \in \mathbb{R}^{\Xi}. \tag{A.4}$$

It should be noted that constraints (A.1) and (A.3) can be imposed simultaneously, as their gradients (A.2) and (A.4) are linearly independent, satisfying the LICQ condition.

References

Arefinia, E., Talebi, H.A., Doustmohammadi, A., 2020. A robust adaptive model reference impedance control of a robotic manipulator with actuator saturation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 50, 409–420. doi:10.1109/TSMC.2017.2759148.

Bernstein, D.S., 2009. Matrix Mathematics: Theory, Facts, and Formulas (Second Edition). Princeton University Press. URL: http://www.jstor.org/stable/j.ctt7t833.

Choi, K., Kim, J., Park, K.B., 2024. Generalized model predictive torque control of synchronous machines. IEEE/ASME Transactions on Mechatronics , 1–11doi:10.1109/TMECH.2024.3461209.

Desoer, C.A., Vidyasagar, M., 2009. Feedback Systems. Society for Industrial and Applied Mathematics. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719055, doi:10.1137/1.9780898719055, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.978089

Douratsos, I., Gomm, J.B., 2007. Neural network based model reference adaptive control for processes with time delay. URL: https://api.semanticscholar.org/CorpusID:17355706.

Esfandiari, K., Abdollahi, F., Talebi, H., 2014. A stable nonlinear in parameter neural network controller for a class of saturated nonlinear systems. IFAC Proceedings Volumes 47, 2533–2538. URL: https://www.sciencedirect.com/science/article/pii/S1474667016419907, doi:https://doi.org/10.3182/20140824-6-ZA-1003.00853. 19th IFAC World Congress.

Esfandiari, K., Abdollahi, F., Talebi, H.A., 2015. Adaptive control of uncertain nonaffine nonlinear systems with input saturation using neural networks. IEEE Transactions on Neural Networks and Learning Systems 26, 2311–2322. doi:10.1109/TNNLS.2014.2378991.

Esfandiari, K., Abdollahi, F., Talebi, H.A., 2021. Neural network-based adaptive control of uncertain nonlinear systems. 2022 ed., Springer Nature, Cham, Switzerland.

Evens, B., Latafat, P., Themelis, A., Suykens, J., Patrinos, P., 2021. Neural network training as an optimal control problem : — an augmented lagrangian approach —, in: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 5136–5143. doi:10.1109/CDC45484.2021.9682842.

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

Gao, W., Selmic, R., 2006. Neural network control of a class of nonlinear systems with actuator saturation. IEEE Transactions on Neural Networks 17, 147–156. doi:10.1109/TNN.2005.863416.

Ge, S., Wang, C., 2002. Direct adaptive nn control of a class of nonlinear systems. IEEE Transactions on Neural Networks 13, 214–221. doi:10.1109/72.977306.

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

Griffis, E.J., Patil, O.S., Bell, Z.I., Dixon, W.E., 2023. Lyapunov-based long short-term memory (lb-lstm) neural network-based control. IEEE Control Systems Letters 7, 2976–2981. doi:10.1109/LCSYS.2023.3291328.

Hart, R.G., Griffis, E.J., Patil, O.S., Dixon, W.E., 2024. Lyapunov-based physics-informed long short-term memory (lstm) neural network-based adaptive control. IEEE Control Systems Letters 8, 13–18. doi:10.1109/LCSYS.2023.3347485.

He, W., Dong, Y., Sun, C., 2016. Adaptive neural impedance control of a robotic manipulator with input saturation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 46, 334–344. doi:10.1109/TSMC.2015.2429555.

Ioannou, P., Fidan, B., 2006. Adaptive Control Tutorial. Society for Industrial and Applied Mathematics, Philadelphia, PA. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718652, doi:10.1137/1.9780898718652, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898718652.

Karason, S., Annaswamy, A., 1994. Adaptive control in the presence of input constraints. IEEE Transactions on Automatic Control 39, 2325–2330. doi:10.1109/9.333787.

Kidger, P., Lyons, T., 2020. Universal Approximation with Deep Narrow Networks, in: Abernethy, J., Agarwal, S. (Eds.), Proceedings of Thirty Third Conference on Learning Theory, PMLR. pp. 2306–2327. URL: https://proceedings.mlr.press/v125/kidger20a.html.

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

Liu, J., 2013. Radial basis function (RBF) neural network control for mechanical systems. 2013 ed., Springer, Berlin, Germany.

Maas, A.L., Hannun, A.Y., Ng, A.Y., et al., 2013. Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, Atlanta, GA. p. 3.

Markus, E.D., Agee, J.T., Jimoh, A.A., 2013. Trajectory control of a two-link robot manipulator in the presence of gravity and friction, in: 2013 Africon, pp. 1–5. doi:10.1109/AFRCON.2013.6757809.

Nocedal, J., Wright, S., 2006. Numerical optimization. Springer series in operations research and financial engineering. 2. ed. ed., Springer, New York, NY. URL: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+502988711&sourceid=fbw_bibsonomy.

Patil, O.S., Le, D.M., Greene, M.L., Dixon, W.E., 2022. Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network. IEEE Control Systems Letters 6, 1855–1860. doi:10.1109/LCSYS.2021.3134914.

Peng, G., Yang, C., He, W., Chen, C.L.P., 2020. Force sensorless admittance control with neural learning for robots with actuator saturation. IEEE Transactions on Industrial Electronics 67, 3138–3148. doi:10.1109/TIE.2019.2912781.

ROBOTIS, 2016. OpenCR1.0. https://www.robotis.us/opencr1-0/. Accessed: 2025-04-08.

Rolnick, D., Tegmark, M., 2018. The power of deeper networks for expressing natural functions. arXiv preprint arXiv:1705.05502 URL: https://arxiv.org/abs/1705.05502, arXiv:1705.05502.

Saerens, M., Soquet, A., 1991. Neural controller based on back-propagation algorithm. IEE Proceedings F (Radar and Signal Processing) 138, 55–62. URL: https://digital-library.theiet.org/doi/abs/10.1049/ip-f-2.1991.0009, doi:10.1049/ip-f-2.1991.0009, arXiv:https://digital-library.theiet.org/doi/pdf/10.1

Sengupta, B., Friston, K., Penny, W., 2014. Efficient gradient computation for dynamical models. NeuroImage 98, 521–527. URL: https://www.sciencedirect.com/science/article/pii/S1053811914003097, doi:https://doi.org/10.1016/j.neuroimage.2014.04.040.

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

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., Goldstein, T., 2016. Training neural networks without gradients: A scalable admm approach, in: Balcan, M.F., Weinberger, K.Q. (Eds.), Proceedings of The 33rd International Conference on Machine Learning, PMLR, New York, New York, USA. pp. 2722–2731. URL: https://proceedings.mlr.press/v48/taylor16.html.

Wang, J., Yu, F., Chen, X., Zhao, L., 2019. Admm for efficient deep learning with global convergence, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA. pp. 111–119. URL: https://doi.org/10.1145/3292500.3330936, doi:10.1145/3292500.3330936.

Yeşildirek, A., Lewis, F., 1995. Feedback linearization using neural networks. Automatica 31, 1659–1664. URL: https://www.sciencedirect.com/science/article/pii/000510989500078B, doi:https://doi.org/10.1016/0005-1098(95)00078-B.

Zhou, X., Shen, H., Wang, Z., Ahn, H., Wang, J., 2023. Driver-centric lane-keeping assistance system design: A noncertainty-equivalent neuro-adaptive control approach. IEEE/ASME Transactions on Mechatronics 28, 3017–3028. doi:10.1109/TMECH.2023.3236245.