Graphical Abstract

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

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

Highlights

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

Myeongseok Ryu, Donghwa Hong, and Kyunghwan Choi

- Research highlight 1
- Research highlight 2

Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Uncertain Euler-Lagrange Systems Under Weight and 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 uncertain Euler-Lagrange systems subject to weight norm and input constraints. A deep neural network (DNN) is employed to approximate the ideal stabilizing control law, compensating for lumped system uncertainties 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. Numerical simulations comparing CoNAC with three benchmark controllers demonstrate its effectiveness in tracking error regulation and satisfaction of constraints.

Keywords: Neuro-adaptive control, constrained optimization, deep neural network, Euler-Lagrange system, 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.
- $\mathrm{row}_i(\boldsymbol{A})$ denotes the i^{th} row of the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$.
- $\mathrm{vec}(\boldsymbol{A}) := [\mathrm{row}_1(\boldsymbol{A}^\top), \cdots, \mathrm{row}_m(\boldsymbol{A}^\top)]^\top$ 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.

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.
- The forward sensitivity method is applied in CoNAC to accurately calculate the gradient of the objective function. [MS: WILL BE DELETED]

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. Candidates of input constraints are presented in Appendix A. Section 5 analyzes the stability of the proposed controller. A comparative study of the four selected controllers, including the proposed controller, is reported in Section 6. Finally, Section 7 concludes the paper and discusses potential future work.

2. Problem Formulation

2.1. Model Dynamics and Control Objective

Consider an uncertain Euler-Lagrange system modeled as

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

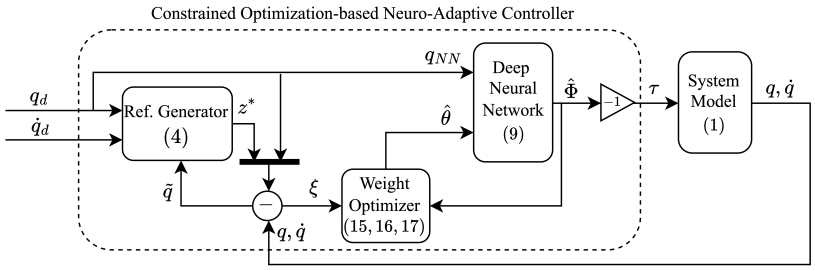


Figure 1: Architecture of the constrained optimization-based neuro-adaptive controller (CoNAC).

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 and is defined as

$$\text{sat}(\boldsymbol{\tau}) := \begin{cases} \boldsymbol{\tau}, & \text{if } \|\boldsymbol{\tau}\| \leq \bar{\tau}, \\ \bar{\tau} \cdot \frac{\boldsymbol{\tau}}{\|\boldsymbol{\tau}\|}, & \text{otherwise,} \end{cases} \tag{2}$$

where $\bar{\tau}$ 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 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.*

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

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 $\|\boldsymbol{q}_d(t)\| \leq \bar{q}_d$ and available to design a feasible control input in the presence of the control input saturation.*

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

$$\boldsymbol{r} := \dot{\boldsymbol{e}} - \boldsymbol{\Lambda}\boldsymbol{e}, \tag{3}$$

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 $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ is a user-designed positive definite matrix. Since (3) 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

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

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

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

where $\boldsymbol{\tau}^* := -\boldsymbol{f}$ is the ideal control input whose maximum norm is $\bar{\tau}$ according to Assumption 1 and (2). Therefore, one can conclude that $\boldsymbol{r}$ is exponentially stable so that $\lim_{t\to\infty}\|\boldsymbol{r}\| = \frac{\bar{\tau}_d}{\lambda_{\min}(\boldsymbol{K})}$, if $\boldsymbol{\tau} = -\boldsymbol{\tau}^*$ can be realized. However, $\boldsymbol{\tau}^*$ is not available in practice, since $\boldsymbol{f}$ is unknown.

To overcome this issue, a DNN is employed to approximate $\boldsymbol{\tau}^*$. Let $\boldsymbol{\Phi} := \boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta}) : \mathbb{R}^l \times \mathbb{R}^\Xi \to \mathbb{R}^n$ represent the DNN, where $\boldsymbol{q}_{NN} \in \mathbb{R}^l$ is the DNN input vector, and $\boldsymbol{\theta} \in \mathbb{R}^\Xi$ is the vector of trainable weights. The architecture of $\boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta})$ will be defined in Section 8. According to the universal approximation theorem for DNNs Kidger and Lyons (2020), $\boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta})$ 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 $\sup_{\boldsymbol{q}_{NN}\in\Omega_{NN}}\|\boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta}^*) - \boldsymbol{g}(\cdot)\| = \boldsymbol{\epsilon} < \infty$. Furthermore, the theorem states that the norm of $\boldsymbol{\theta}^*$ is bounded, i.e., $\|\boldsymbol{\theta}^*\| \leq \bar{\boldsymbol{\theta}} < \infty$. 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 $\boldsymbol{\Phi}^* := \boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta}^*)$ as follows:

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

which is estimated online by

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

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

Substituting (5) and (6) into (4), the time derivative of $\mathcal{V}_1$ can be rewritten as

$$\dot{\mathcal{V}}_1 \leq -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{r}\|^2 + \bar{\tau}_d\|\boldsymbol{r}\| + \boldsymbol{r}^\top(\text{sat}(\hat{\boldsymbol{\Phi}}) - \boldsymbol{\Phi}^* - \boldsymbol{\epsilon}). \tag{7}$$

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 $\boldsymbol{\Phi}(\boldsymbol{q}_{NN}; \boldsymbol{\theta}) := \boldsymbol{\Phi}_k$ can be recursively represented as follows:

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

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 $\|\frac{\mathrm{d}\boldsymbol{\phi}_i(x)}{\mathrm{d}\boldsymbol{x}}\|_F < \infty,\ \forall \boldsymbol{x} \in \mathbb{R}^{l_i}$. 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

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

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

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

where

$$\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\theta}_i} = \begin{cases} (\boldsymbol{I}_{l_{k+1}} \otimes \boldsymbol{\phi}_k^\top), & i = k \\ \boldsymbol{W}_k^\top \boldsymbol{\phi}_k'(\boldsymbol{I}_{l_k} \otimes \boldsymbol{\phi}_{k-1}^\top), & i = k-1 \\ \qquad\qquad\vdots & \\ \boldsymbol{W}_k^\top \boldsymbol{\phi}_k' \cdots \boldsymbol{W}_1^\top \boldsymbol{\phi}_1'(\boldsymbol{I}_{l_1} \otimes \boldsymbol{q}_{NN}^\top), & i = 0 \end{cases}, \tag{9}$$

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

$$\begin{aligned} &\min_{\widehat{\boldsymbol{\theta}}}\ J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}}) := \tfrac{1}{2}\boldsymbol{r}^\top \boldsymbol{W} \boldsymbol{r} \\ &\text{s.t. } c_j(\widehat{\boldsymbol{\theta}}) \le 0, \quad j \in \mathcal{I}, \end{aligned} \tag{10}$$

where $J(\boldsymbol{r}; \widehat{\boldsymbol{\theta}}) := \frac{1}{2}\boldsymbol{r}^\top \boldsymbol{W} \boldsymbol{r}$ is the objective function, and $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is a symmetric positive weighting matrix. Inequality constraints $c_j,\ j \in \mathcal{I}$, 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

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

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 (10) (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:

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} &= -\alpha \frac{\partial L}{\partial \widehat{\boldsymbol{\theta}}} = -\alpha \left(\frac{\partial J}{\partial \widehat{\boldsymbol{\theta}}} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right), \\ \tfrac{\mathrm{d}}{\mathrm{d}t}\lambda_j &= \beta_j \frac{\partial L}{\partial \lambda_j} = \beta_j c_j, \quad \forall j \in \mathcal{I}, \\ \lambda_j &= \max(\lambda_j, 0), \end{aligned} \tag{11a}$$

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 $\lambda_j \leftarrow \max(\lambda_j - \frac{c_j}{\mu}, 0)$, 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., $\partial L/\partial\widehat{\boldsymbol{\theta}} = 0$, $c_j \leq 0$, $\lambda_j \geq 0$, 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 (11a) 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:

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

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 (11a) as follows:

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\widehat{\boldsymbol{\theta}} \approx -\alpha \left(\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}}^\top \boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}} \right). \tag{12}$$

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:

$$\begin{aligned}\boldsymbol{\tau} =& \widehat{\boldsymbol{\Phi}} = \mathrm{vec}(\widehat{\boldsymbol{W}}_k^\top \widehat{\boldsymbol{\phi}}_k) = (\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top \mathrm{vec}(\widehat{\boldsymbol{W}}_k) \\ =& \underbrace{(\boldsymbol{I}_{l_{k+1}} \otimes \widehat{\boldsymbol{\phi}}_k^\top)^\top}_{:=\boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)} \widehat{\boldsymbol{\theta}}_k = \boldsymbol{T}(\widehat{\boldsymbol{\phi}}_k)\widehat{\boldsymbol{\theta}}_k.\end{aligned} \tag{13}$$

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:

$$c_{\theta_i} = \tfrac{1}{2}(\|\widehat{\boldsymbol{\theta}}_i\|^2 - \bar{\theta}_i^2) \leq 0 \tag{14}$$

with $\bar{\theta}_i < \infty$ 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

$$\frac{\partial c_{\theta_i}}{\partial \widehat{\boldsymbol{\theta}}_j} = \begin{cases} \widehat{\boldsymbol{\theta}}_i, & \text{if } i = j \\ \boldsymbol{0}, & \text{if } i \neq j. \end{cases} \tag{15}$$

Note that $\boldsymbol{c}_\theta$ satisfies Assumption 2 and 3 invoking Lemma 1 and (15), 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

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

where $\frac{\partial \boldsymbol{\tau}}{\partial \widehat{\boldsymbol{\Phi}}} = \boldsymbol{I}_n$, which is bounded. From the linear mapping in (13), $\boldsymbol{\tau}$ 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 (9). 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 (11) ensure the boundedness of the filtered error $\boldsymbol{r}$ and the weight estimate $\widehat{\boldsymbol{\theta}}$. This holds with the weight norm constraint (14) 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 $c_{\theta_i},\ \forall i \in [0, \cdots, k]$ 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

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

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:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}V_2 &= -\widehat{\boldsymbol{\theta}}_k^\top ((\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)^\top \boldsymbol{r} + \textstyle\sum_{j\in\mathcal{I}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}) \\
&= -\widehat{\boldsymbol{\theta}}_k^\top ((\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)^\top \boldsymbol{r} + \lambda_{\theta_k}\widehat{\boldsymbol{\theta}}_k \\
&\quad + \textstyle\sum_{j\in\mathcal{A}\setminus\{\boldsymbol{\theta}_i\}_{i\in[0,\cdots,k]}} \lambda_j \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}) \\
&\leq -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + \underbrace{\|(\boldsymbol{I}_{l_{k+1}} \otimes \hat{\boldsymbol{\phi}}_k^\top)\|}_{:=c_1\in\mathbb{R}_{>0}}\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\| \\
&\quad - \underbrace{\textstyle\sum_{j\in\mathcal{A}\setminus\{\boldsymbol{\theta}_i\}_{i\in[0,\cdots,k]}} \lambda_j \widehat{\boldsymbol{\theta}}_k^\top \frac{\partial c_j}{\partial \widehat{\boldsymbol{\theta}}_k}}_{:=c_2>0,\text{ by Lemma 1 and Assumption 3}} \\
&\leq -\lambda_{\theta_k}\|\widehat{\boldsymbol{\theta}}_k\|^2 + c_1\|\boldsymbol{r}\|\|\widehat{\boldsymbol{\theta}}_k\|.
\end{aligned} \tag{16}$$

According to (16), the invariant set of $\widehat{\boldsymbol{\theta}}_k$ is defined as

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

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 (16). Similarly to c_{θ_k}, as the control input constraint $c_j,\ \forall j \in \mathcal{A}\setminus\{\boldsymbol{\theta}_i\}_{i\in[0,\cdots,k]}$ is violated, the corresponding Lagrange multiplier λ_j increases infinitely, so does c_2. It makes c_2 dominates the right-hand side of (16), 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 $\Theta_{\theta_k}^{c_1}$ 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 (16) can be removed. Consider the Lyapunov function candidate $V_3 := V_1 + V_2$ and its time derivative:

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

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

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

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

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

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

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. Implementation and Validation

The proposed CoNAC was validated via numerical simulations and real-time experiments on a two-link manipulator.

6.1. Numerical Simulation Validation

The proposed CoNAC was validated using a two-link manipulator model, as depicted in Fig. 2, adapted from Markus et al. (2013). The system dynamics of the two-link manipulator is described by the following equations: [MS: ADD MORE DETAIL; Following Eqn is just example]

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q})\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q} + \boldsymbol{V}_m(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) &= \boldsymbol{\tau}, \\ \boldsymbol{\tau} &= \widehat{\boldsymbol{\Phi}}(\boldsymbol{q},\dot{\boldsymbol{q}}),\end{aligned}$$

The parameters $q_p, q_{dp}, \tau_p, m_p, l_p, l_{cp}, b_p$ and f_{cp} [MS: more parameters] denote the joint angle, desired joint angle, torque, mass, length, center of mass, viscous coefficient, and friction coefficient, respectively, for link $p \in [1,2]$. The

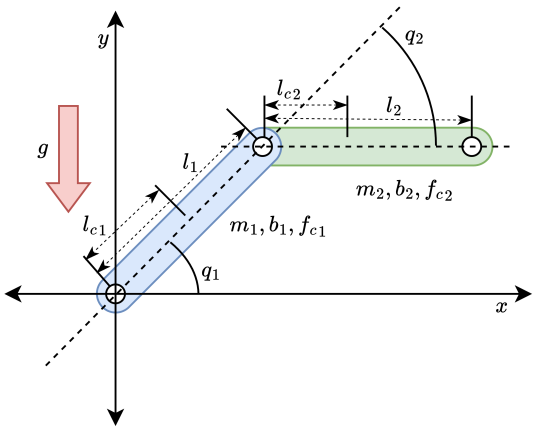


Figure 2: Two-link manipulator model.

Table 1: System model parameters.

Symbol	Description	Link 1	Link 2
m_p	Mass of p^{th} link	23.902 (kg)	3.88 (kg)
l_p	Length of p^{th} link	0.45 (m)	0.45 (m)
l_{cp}	COM of p^{th} link	0.091 (m)	0.048 (m)
b_p	Viscous coef. of p^{th} link	2.288 (Nms)	0.172 (Nms)
f_{cp}	Friction coef. of p^{th} link	7.17 (Nm)	1.734 (Nm)
f_{sp}	?? p^{th} link	(Nm)	(Nm)

values of the system model parameters are provided in Table 1. The desired trajectory for $\boldsymbol{q} = [q_1, q_2]^\top$ was defined with [MS: 5th Poly functions] as

$$q_d(t) = \begin{bmatrix} q_{d1} \\ q_{d2} \end{bmatrix} = \begin{cases} \text{Poly}(\boldsymbol{q}_0, \boldsymbol{q}_1, t_f, t) & \text{if } 0 \le t < t_f \\ \text{Poly}(\boldsymbol{q}_1, \boldsymbol{q}_2, t_f, t - t_f) & \text{if } t_f \le t < 2t_f \\ \text{Poly}(\boldsymbol{q}_2, \boldsymbol{q}_3, t_f, t - 2t_f) & \text{if } 2t_f \le t < 3t_f \\ \text{Poly}(\boldsymbol{q}_3, \boldsymbol{q}_0, t_f, t - 3t_f) & \text{if } 3t_f \le t < 4t_f \end{cases},$$

where [MS: CAN BE CHANGED]$\boldsymbol{q}_0 = [0, 0]^\top$, $\boldsymbol{q}_1 = [\pi/2, \pi/2]^\top$, $\boldsymbol{q}_2 = [\pi/2, 0]^\top$, $\boldsymbol{q}_3 = [0, \pi/2]^\top$, and $t_f = 5$ s. [MS: VIDEO IS PROVIDED; IF WE CAN PROVIDE]

To demonstrate the effectiveness of the proposed CoNAC to handle the combination of the nonlinear input constraints, the following control input saturations were imposed:

$$\text{sat}(\boldsymbol{\tau}) =,$$

where $\bar{\tau} = 50, \bar{\tau}_1$ and $\bar{\tau}_2$ are the saturation limit values for the norm of the control input and the first and second links, respectively.

For a comparative study, NAC with the auxiliary system presented in Esfandiari et al. (2014); Karason and Annaswamy (1994); Esfandiari et al. (2015) (NAC-AUX) was implemented, which only considered the input bound constraint introcuded in Appendix A.2 and not the nonlinear input norm constraint (A.2). As a result, an approximation of the input norm constraint was used as an input bound constraint. Besides, the proposed controller, CoNAC, rigorously considered the weight norm constraint, and input bound and norm constraints within a constrained optimization framework. The properties of these four controllers are summarized in Table 2.

Table 2: Properties of the controllers used in simulation

	Handling Capability	
	Weight Norm Constraint	Input Norm Constraint
CoNAC	O (by projection)	△ (by aux. system)
NAC-AUX	O (by optimization)	O (by optimization)

Table 3: Parameters of the controllers used in simulation

	l_0	l_1	l_2	l_3	α	β	$\bar{\theta}_0$	$\bar{\theta}_1$	$\bar{\theta}_2$	$\bar{\tau}$	τ_1	τ_2	$\boldsymbol{A}_\zeta$	$\boldsymbol{B}_\zeta$
BSC	6	4	4	2	10^3	10^{-4}	11	12	13	10	11	0.5		
NAC-AUX												0.5	$[20, 0; 0, 20]$	$[10, 0; 0, 10]$

The control law for NAC-AUX was the same CoNAC, but with an auxiliary system to compensate for control input violations. Since the auxiliary system handled only the input bound constraint, not the more complex input norm constraint (A.2), an approximated version of the input norm constraint was used as an input bound constraint (A.1) with $\bar{\tau}_i = -\underline{\tau}_i = (\bar{\tau}/\sqrt{2}+\bar{\tau})/2$. The comparison between the original input norm constraint and its approximation is shown in Fig. [MS: ADD FIGURE]. The auxiliary system is defined as $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\zeta} = \boldsymbol{A}_\zeta\boldsymbol{\zeta} + \boldsymbol{B}_\zeta\Delta\boldsymbol{\tau}$, $\boldsymbol{\zeta}|_{t=0} = 0$, where $\boldsymbol{\zeta} \in \mathbb{R}^n$ denotes the auxiliary state, $\boldsymbol{A}_\zeta = -[20, 0; 0; 20]$, $\boldsymbol{B}_\zeta = [10, 0; 0, 10]$[MS: SEE HERE], and $\Delta\tau_{(i)} = \tau_{(i)} - \mathrm{sat}(\tau_{(i)}, \bar{\tau}_i, \underline{\tau}_i)$ The auxiliary state variables were used in the adaptation law (12) by substituting $\boldsymbol{r}$ with $\boldsymbol{r} + \boldsymbol{\zeta}$.

The proposed CoNAC directly approximated the control law using the DNN as defined in (6). The update rates for the multipliers were set as [MS: SEE HERE]$\beta_j = 0.1$. The weight matrix W was selected as $W = \mathrm{diag}([5, 1, 15, 15])$].

For all DNN-based controllers (DNN-BSC, DNN-BSC-A, and CoNAC), the DNN input vector q_{NN} was set as the desired trajectory for q, i.e., $q_{NN} = [{q_d}^\top, 1]^\top$ with the augmented scalar 1 included to account for the bias term in the weight matrix. Each DNN architecture had two hidden layers with eight nodes (i.e., $k = 2, l_0 = 2, l_1 = 8, l_2 = 8, l_3 = 2$), and the adaptation gain was set to $\alpha = 10^3$. The constraint parameters were $\bar{\boldsymbol{\theta}}_0 = 20, \bar{\boldsymbol{\theta}}_1 = 30, \bar{\boldsymbol{\theta}}_2 = 40$, and $\bar{\tau} = 50$ [MS: SEE HERE]. The control parameters for all the controllers were set as $k_q = 1.1, k_z = 10, M_0 = I_2, C_0 = I_2, G_0 = [0, 0]^\top$. The sampling time of the simulations was selected as $T_s = 10^{-4}$. The parameters for CoNAC to NAC-AUX were listed in Table. 3.

6.1.1. System Uncertainty Handling

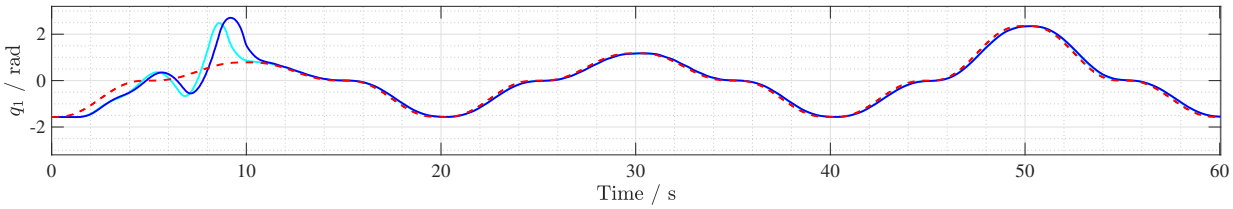
The tracking results of the selected controllers are shown in Fig. **??**. To demonstrate the effectiveness of using DNNs for compensating the lumped system uncertainty function f, the gains k_q and k_z for BSC were intentionally selected as small values, resulting in a weak ability to handle these uncertainties. As a result, BSC failed to track the reference trajectory, as shown in Fig. **??**.

By leveraging the DNN to compensate for the lumped system uncertainty within the BSC, DNN-BSC achieved improved tracking performance compared to BSC, as seen in Fig. **??**. Fig. **??** shows that DNN-BSC-A enhanced tracking performance for q_2, but tracking for q_1 remained unsatisfactory due to incomplete constraint handling, which will be discussed in detail in Section 6.2.

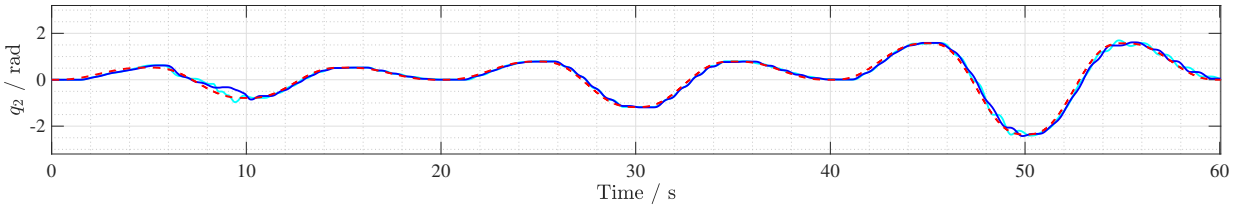
Finally, CoNAC, which directly approximates the stabilizing control law along with the compensation term, demonstrated satisfactory tracking performance across both states, as illustrated in Fig. **??**.

6.2. Input Norm Constraint Handling

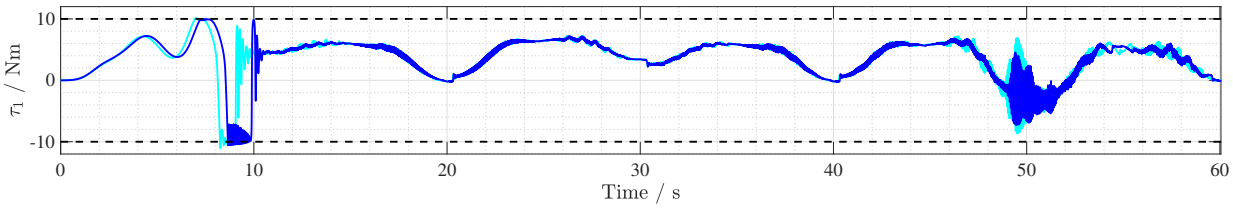
The resulting control input τ and physically saturation control input $h(\tau)$ for the selected controllers are shown in Fig. **??**. As illustrated in Fig. **??**, BSC did not violate the input norm constraint (i.e., $\tau = h(\tau)$). However, in DNN-BSC, the added compensation term from the DNN caused violations of the input norm constraint (i.e., $\tau > h(\tau)$)



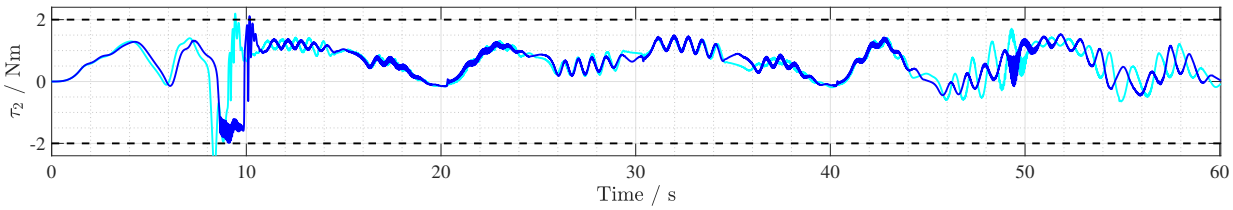
(a) Tracking result of d-axis stator current .



(b) Tracking result of d-axis stator current .



(c) Control input.



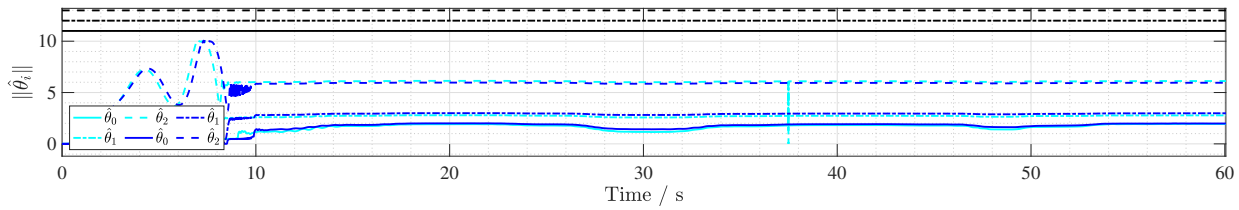
(d) Control input.

Figure 3: Simulation results of the CoNAC (blue) and NAC-AUX (cyan), and reference signal of $\boldsymbol{q}_d$ (red dashed line).

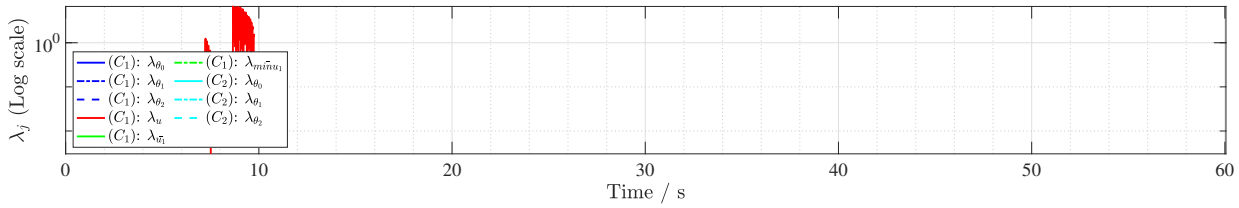
at several points; see Fig. **??**. This failure to account for the input norm constraint led to oscillations in the control input τ. The DNN adaptation process attempted to increase the weights to reduce the residual errors that were not constrained by saturation, but after saturation ceased, the control input had to rapidly adjust back to realistic levels, leading to oscillatory behavior. Such high-frequency oscillations may induce instability in the control system or cause fatigue in the actuators.

On the other hand, both DNN-BSC-A and CoNAC successfully handled their imposed input constraints, as shown in Fig. **??** and Fig. **??**, respectively, without causing notable oscillations in the control input τ even after the input constraint was activated. However, the tracking performance of DNN-BSC-A was lower than that of DNN-BSC and CoNAC, as the auxiliary system used in DNN-BSC-A approximated the input norm constraint with an input bound constraint, creating a rectangular constraint in the τ-space (see Fig. **??**). In contrast, CoNAC satisfied the nonlinear input norm constraint and produced the physically maximum control input, resulting in improved tracking performance.

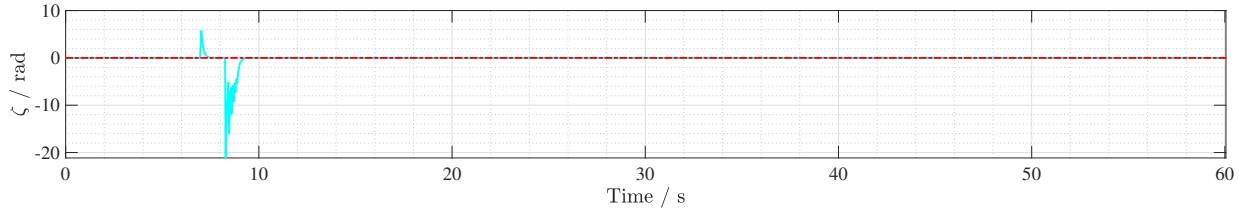
It is also important to note that the control input trajectory in DNN-BSC-A depends on the dynamics of the auxiliary system. The auxiliary system regulates the violated control input after sufficient auxiliary state ζ is generated to compensate for the violation. This can be observed in Fig. **??**, where DNN-BSC-A exhibited minor violations of the input bound constraint. In contrast, CoNAC satisfied the constraint without being affected by such dynamics, as



(a) Tracking result of d-axis stator current .



(b) Tracking result of d-axis stator current .



(c) Control input.

Figure 4: Simulation results of the CoNAC (blue) and NAC-AUX (cyan), and reference signal of $\boldsymbol{q}_d$ (red dashed line).

its Lagrange multiplier adjusted as soon as the constraint was violated.

6.3. Weight Norm Constraint Handling

The resulting weight norms of DNN-BSC, DNN-BSC-A, and CoNAC, along with the Lagrange multipliers of CoNAC, are shown in Fig. **??**. All three controllers—DNN-BSC, DNN-BSC-A, and CoNAC—maintained weight norms within the imposed weight norm constraints.

IN DNN-BSC, as shown in Fig. **??**, the weight norm of the last layer (i.e., $\|\hat{\boldsymbol{\theta}}_2\|$) fluctuated significantly over time, proportional to the control input norm. This is because the last layer's weights directly determine the control input. When the control input violated the input norm constraint, the last layer's weight norm hit the boundary and stayed there due to the projection operator. However, the projection operator only responded to violations without considering optimality or behavior.

In DNN-BSC-A, none of the weight norms reached their boundaries, as shown in Fig. **??**. This was due to the auxiliary system, where the auxiliary state ζ reduced the control input, ensuring it stayed within the input constraint.

In CoNAC, all weight norms complied with the imposed constraints through the constrained optimization approach, as illustrated in Fig.**??**. When any weight norm approached its upper limit, the Lagrange multiplier was promptly activated to steer the weight adaptation direction towards a constraint-satisfactory point (see Fig.**??**). Notably, the weight norms of the first and second layers ($\|\hat{\boldsymbol{\theta}}_0\|$ and $\|\hat{\boldsymbol{\theta}}_1\|$) remained nearly constant throughout the control period. The weight norm of the last layer $\|\hat{\boldsymbol{\theta}}_2\|$ stabilized within the upper bound by around 6.5 seconds (see Fig. **??** (B)), coinciding with the activation of the input norm constraint. This quasi-static behavior of the weight norm (i.e., $d\widehat{\boldsymbol{\theta}}/dt = -\alpha\partial L/\partial\widehat{\boldsymbol{\theta}} \approx 0$) along with the quasi-static behavior of the Lagrange multipliers (i.e., $\dot{\lambda}_j = \beta_j c_j \approx 0$) implies that the weights were updated near the KKT conditions, signifying local optimality in CoNAC. However, at around 2.5 seconds (see Fig. **??** (A)), the weight norm of the last layer reached the upper limit earlier, despite similar control

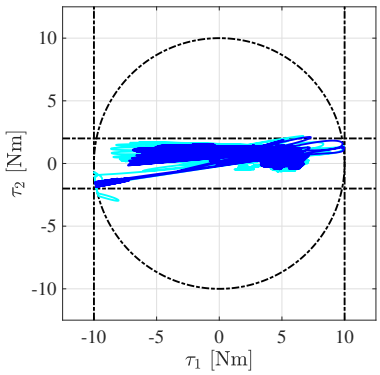


Figure 5: Simulation results of the CoNAC (blue) and NAC-AUX (cyan), and reference signal of $\boldsymbol{q}_d$ (red dashed line.

conditions as the case at 6.5 seconds. This earlier saturation likely occurred because the optimization process had not yet fully converged to the optimal weight values.

The overall weight norms of CoNAC were larger than those of DNN-BSC and DNN-BSC-A, since CoNAC approximated the entire stabilizing control law, whereas DNN-BSC and DNN-BSC-A only approximated the system uncertainty term within the BSC framework.

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. 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

$$\begin{aligned} c_{\overline{\tau}_i} &= \tau_{(i)} - \tau_{\overline{\tau}_i} \leq 0 \\ c_{\underline{\tau}_i} &= \tau_{\underline{\tau}_i} - \tau_{(i)} \leq 0 \end{aligned} \tag{A.1}$$

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

$$\frac{\partial \boldsymbol{c}_{\bar{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} = = -\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = -\left[(I_{l_{k+1}} \otimes \widehat{\boldsymbol{\Phi}}_k^\top) \quad \cdots \quad (\cdot)\right] \in \mathbb{R}^{n \times \Xi},$$
$$\frac{\partial \boldsymbol{c}_{\underline{\tau}}}{\partial \widehat{\boldsymbol{\theta}}} := = +\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}} = +\left[(I_{l_{k+1}} \otimes \widehat{\boldsymbol{\Phi}}_k^\top) \quad \cdots \quad (\cdot)\right] \in \mathbb{R}^{n \times \Xi}.$$

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:

$$c_u = \|\boldsymbol{\tau}\|^2 - \bar{\tau}^2 \leq 0 \tag{A.2}$$

with $\bar{\tau} \in \mathbb{R}_{>0}$ 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_u with respect to $\widehat{\boldsymbol{\theta}}$ are given by

$$\frac{\partial \boldsymbol{c}_u}{\partial \widehat{\boldsymbol{\theta}}} = -\sum_{i=1}^{n} 2\tau_{(i)} \left(\mathrm{row}_i(-\frac{\partial \widehat{\boldsymbol{\Phi}}}{\partial \widehat{\boldsymbol{\theta}}})\right)^\top = \tau^\top (I_{l_{k+1}} \otimes \widehat{\boldsymbol{\Phi}}_k^\top) \in \mathbb{R}^{\Xi}.$$

It should be noted that constraints (**??**) and (**??**) cannot be imposed simultaneously, as their gradients (**??**) and (**??**) are linearly dependent, violating 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/10.1137/1.9780898719055.

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.

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/10.1049/ip-f-2.1991.0009.

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.