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

***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.**

***Index Terms*—Neuro–adaptive control, constrained optimization, deep neural network, Euler–Lagrange system, input constraint.**

I. Introduction

A. Background

MANY engineering systems, including those in aerospace, robotics, and automotive applications, can be modeled using Euler–Lagrange systems. These systems are governed by dynamic equations derived from energy principles and describe the motion of mechanical systems with constraints. In practice, however, such systems often exhibit uncertainties due to unmodeled dynamics, parameter variations, or external disturbances. These uncertainties can significantly degrade control performance and, in some cases, lead to instability. To address these challenges, adaptive control methods have been widely employed to ensure robust performance in the presence of system uncertainties [1], [2].

More recently, neuro–adaptive control approaches have been introduced to approximate unknown system dynamics or entire control laws using neural networks (NNs) [3]. 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 [4], [5] and radial basis function (RBF) neural networks [6], [7], as well as more complex models like deep neural networks (DNNs) [8] and their variations. SHL and RBF NNs are often employed to approximate uncertain system dynamics or controllers due to their simplicity [5], [9]–[11], while DNNs offer greater expressive power, making them more effective for complex system approximations [12]. Additionally, variations of DNNs, such as long short-term memory (LSTM) networks for time-varying dynamics [13] and physics-informed neural networks (PINNs) for leveraging physical system knowledge [14], 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 [15]. 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. The following section will provide a detailed review of the existing solutions to these challenges.

B. Literature Review

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 [8], [13], [16], 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 [7] and ϵ-modification [10], [11]. 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.

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 [9], [10], [17], 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 [18]–[20]. In [11], 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.

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 [21].

In existing literature, constrained optimization techniques, such as the Augmented Lagrangian Method (ALM) [22] and the Alternating Direction Method of Multipliers (ADMM) [23], [24], 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.

C. 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.

D. Organization

The remainder of this paper is organized as follows. Section II presents the target system and control objective. Section III introduces the proposed controller and the architecture of DNN in the controller. In Section IV, the weight adaptation laws are developed. Candidates of the weight and input constraints are presented in Section V. Section VI analyzes the stability of the proposed controller. A comparative study of the four selected controllers, including the proposed controller, is reported in Section VII. Finally, Section VIII concludes the paper and discusses potential future work.

II. Problem Formulation

A. Notation

In this study, the following notation is used:

- $\otimes$ denotes the Kronecker product [25].
- $x_{(i)}$ denotes the i^{th} element of the vector $x \in \mathbb{R}^n$.
- $\text{row}_i(A)$ denotes the i^{th} row of the matrix $A \in \mathbb{R}^{n\times m}$.
- $\text{vec}(A) \triangleq [\text{row}_1(A^T), \cdots, \text{row}_m(A^T)]^T$ for $A \in \mathbb{R}^{n\times m}$.
- $\lambda_{\text{min}}(A)$ denotes the minimum eigenvalue of the matrix $A \in \mathbb{R}^{n\times n}$.

B. Model Dynamics and Control Objective

Consider an uncertain Euler–Lagrange system modeled as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(q) = h(\tau) \tag{1}$$

where $q \in \mathbb{R}^n$ denotes the generalized coordinate, and $\tau \in \mathbb{R}^n$ denotes the control input. The terms $M(q) \in \mathbb{R}^{n\times n}$, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$, and $G(q) \in \mathbb{R}^n$ represent the unknown system function matrices, while $F(q) \in \mathbb{R}^n$ denotes the external force. The function $h(\cdot) \in \mathbb{R}^n$ is a control input saturation function, where each element represents the control input saturation for each element of τ. The gradient of $h(\cdot)$ with respect to τ is continous and bounded, i.e., $\|\partial h/\partial \tau\|_F \in L_\infty$.

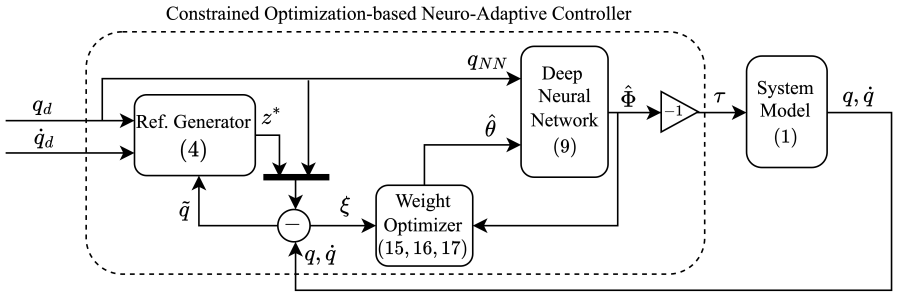


Fig. 1: Architecture of the constrained optimization-based neuro–adaptive controller (CoNAC).

The control input saturation function represents the inherent physical limitations of the actuators. To account for these limitations, it is essential to incorporate physically motivated constraints into the controller design. Section V introduces candidate constraints that can be applied to ensure compliance with these physical limitations.

Using user-designed nominal system matrices $M_0 > 0$, C_0, and G_0, (1) can be reformulated as

$$M_0\ddot{q} + C_0\dot{q} + G_0 = h(\tau) + f(q, \dot{q}, \ddot{q}) \tag{2}$$

where $f(q, \dot{q}, \ddot{q}) \triangleq -\Delta M(q)\ddot{q} - \Delta C(q, \dot{q})\dot{q} - \Delta G(q) - F(q) \in \mathbb{R}^n$ is the lumped system uncertainty function. Here, $\Delta M(q) \triangleq M(q) - M_0$, $\Delta C(q, \dot{q}) \triangleq C(q, \dot{q}) - C_0$, and $\Delta G(q) \triangleq G(q) - G_0$.

The function f acts like an external disturbance, leading to a poor performance index and potential instability. The control objective is to develop a neuro–adaptive controller that enables q to track a continuously differentiable desired trajectory $q_d(t) : \mathbb{R} \to \mathbb{R}^n$, compensating for the external disturbance while addressing the imposed constraints.

III. Control Law Development

The architecture of the proposed CoNAC is illustrated in Fig. 1, consisting of: a reference generator, a DNN that functions as a neuro–adaptive controller, and a weight optimizer for the DNN. The reference generator is designed based on backstepping control (BSC), similar to the approach in [26], to generate a tracking reference for both q and $\dot{q}$. Section III-A presents the neuro–adaptive controller along with the reference generator, and Section III-B defines the DNN model. The weight optimizer will be detailed in Section IV.

A. Neuro–Adaptive Controller Design

The system dynamics (2) can be represented as

$$\begin{aligned} \dot{q} &= z, \\ \dot{z} &= -M_0^{-1}C_0 z - M_0^{-1}G_0 + M_0^{-1}h(\tau) + M_0^{-1}f, \end{aligned} \tag{3}$$

where $z \triangleq \dot{q}$.

Consider the Lyapunov function $\mathcal{V}_{c1} \triangleq (1/2)\tilde{q}^T\tilde{q}$, where $\tilde{q} \triangleq q - q_d$ represents the tracking error between the actual trajectory q and the desired trajectory q_d. The desired trajectory of z, ensuring $\dot{\mathcal{V}}_{c1} = \tilde{q}^T(z - \dot{q}_d) < 0$ is

$$z^* \triangleq -k_q\tilde{q} + \dot{q}_d, \tag{4}$$

which functions as the reference generator with control gain $k_q \in \mathbb{R}_{>0}$. The tracking error of z relative to the desired trajectory z^* is defined as

$$\tilde{z} \triangleq z - z^* = z - (-k_q\tilde{q} + \dot{q}_d). \tag{5}$$

Next, consider the Lyapunov function $\mathcal{V}_{c2} \triangleq \mathcal{V}_{c1} + (1/2)\tilde{z}^T\tilde{z}$. Its time derivative is

$$\begin{aligned} \dot{\mathcal{V}}_{c2} &= \tilde{q}^T(-k_q\tilde{q} + \tilde{z}) + \tilde{z}^T(-M_0^{-1}C_0 z - M_0^{-1}G_0 \\ &\quad + M_0^{-1}h(\tau) + M_0^{-1}f - \dot{z}^*) \\ &= -k_q\tilde{q}^T\tilde{q} - k_z\tilde{z}^T\tilde{z} + \tilde{z}^T(k_z\tilde{z} + \tilde{q} \\ &\quad - M_0^{-1}C_0 z - M_0^{-1}G_0 + M_0^{-1}h(\tau) + M_0^{-1}f - \dot{z}^*) \end{aligned} \tag{6}$$

with control gain $k_z \in \mathbb{R}_{>0}$. The stabilizing control law, which does not account for weight norm and input constraints, is defined as follows:

$$\tau^* \triangleq -M_0 \cdot (k_z\tilde{z}) + (-M_0\tilde{q} + C_0 z + G_0 - f + M_0\dot{z}^*). \tag{7}$$

This control law ensures that the time derivative of the Lyapunov function is negative definite, as $\dot{\mathcal{V}}_{c2} = -k_q\tilde{q}^T\tilde{q} - k_z\tilde{z}^T\tilde{z} < 0$, in the absence of any constraints. However, the control law τ^* cannot be realized in practice because the lumped system uncertainty function f, which accounts for unmodeled dynamics and disturbances, is not available.

Let $\Phi \triangleq \Phi(q_{NN}; \theta) : \mathbb{R}^l \times \mathbb{R}^\Xi \to \mathbb{R}^n$ represent the DNN, where $q_{NN} \in \mathbb{R}^l$ is the DNN input vector, and $\theta \in \mathbb{R}^\Xi$ is the vector of trainable weights. The architecture of $\Phi(q_{NN}; \theta)$ will be defined in Section III-B. According to the universal approximation theorem for DNNs [27], $\Phi(q_{NN}; \theta)$ can approximate a nonlinear function $g(\cdot)$ with an ideal weight vector θ^* on a compact subset $\Omega_{NN} \in \mathbb{R}^l$ to ϵ-accuracy, such that $\sup_{q_{NN} \in \Omega_{NN}} \|\Phi(q_{NN}; \theta^*) - g(\cdot)\| = \epsilon < \infty$. Furthermore, the theorem states that the norm of θ^* is bounded, i.e., $\|\theta^*\| \leq \bar{\theta} < \infty$. In this study, θ^* is defined as a local optimal point, rather than a global optimal point.

The stabilizing control law is expressed by the DNN approximation with the ideal weight vector $\Phi^* \triangleq \Phi(q_{NN}; \theta^*)$ as follows:

$$\tau^* = -(\Phi^* + \epsilon), \tag{8}$$

which is estimated online by

$$\tau = -\hat{\Phi}, \tag{9}$$

where $\hat{\Phi} \triangleq \Phi(q_{NN};\hat{\theta})$, and $\hat{\theta}$ is the estimated weight vector for θ^*.

Using (3), (5), (7), and (9), the error dynamics can be derived as

$$\begin{aligned}\dot{\tilde{q}} &= -k_q\tilde{q} + \tilde{z}\\ \dot{\tilde{z}} &= -\tilde{q} - k_z\tilde{z} + M_0^{-1}(\Phi^* - h(\hat{\Phi}) + \epsilon).\end{aligned} \tag{10}$$

The error dynamics (10) can be represented as a first-order system:

$$\dot{\xi} = A_\xi\xi + B_\xi(\Phi^* - h(\hat{\Phi}) + \epsilon) \tag{11}$$

where $\xi \triangleq [\tilde{q}^T, \tilde{z}^T]^T \in \mathbb{R}^{2n}$ denotes the augmented error, and

$$A_\xi \triangleq \begin{bmatrix} -k_qI_n & I_n \\ -I_n & -k_zI_n \end{bmatrix},\ B_\xi \triangleq \begin{bmatrix} 0_{n\times n} \\ M_0^{-1} \end{bmatrix}.$$

Note that A_ξ is a stable matrix, and $\|B_\xi\|_F < \infty$.

B. Deep Neural Network (DNN) Model

The DNN architecture presented in [8] is utilized in this study and is defined as follows:

$$\Phi(q_{NN};\theta) \triangleq V_k^T\phi_k(V_{k-1}^T\cdots\phi_2(V_1^T\phi_1(V_0^Tq_{NN}))\cdots)) \tag{12}$$

where $V_i \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$ is the weight matrix of the i^{th} layer, and $\phi_i : \mathbb{R}^{l_i} \to \mathbb{R}^{l_i+1}$ represents the activation function of the i^{th} layer. The activation function is defined as $\phi_i(x) = [\sigma(x_{(1)}), \sigma(x_{(2)}), \cdots, \sigma(x_{(l_i)}), 1]^T$, 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 $\Phi(\cdot)$ is the same as that of the control input τ (i.e., $l_{k+1} = n$). For a better understanding, (12) also can be represented recursively as

$$\Phi_i \triangleq \begin{cases} V_i^T\phi_i(\Phi_{i-1}), & i \in [1,\ldots,k], \\ V_0^Tq_{NN}, & i = 0, \end{cases}$$

where $\Phi_k = \Phi(q_{NN};\theta)$.

One of the widely used activation functions for large DNNs is from the ReLU family [28], 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 $\|\phi_i(\cdot)\| < \infty$ and $\|\nabla\phi_i(\cdot)\|_F < \infty$. In this study, the hyperbolic tangent function $\tanh(\cdot)$ was selected as the activation function (i.e., $\sigma(\cdot) = \tanh(\cdot)$), which provides desirable boundedness with $\|\sigma(\cdot)\| < 1$ and $\|\nabla\sigma(\cdot)\| < 1$.

For simplicity, each layer's weights are vectorized as $\theta_i \triangleq \text{vec}(V_i) \in \mathbb{R}^{\Xi_i}$, where $\Xi_i \triangleq (l_i+1)l_{i+1}$ is the number of weights in the i^{th} layer. The total weight vector $\theta \in \mathbb{R}^{\Xi}$ is defined by augmentating θ_i for all $i \in [0,\cdots,k]$ as

$$\theta \triangleq \begin{bmatrix} \theta_k \\ \theta_{k-1} \\ \vdots \\ \theta_0 \end{bmatrix} = \begin{bmatrix} \text{vec}(V_k) \\ \text{vec}(V_{k-1}) \\ \vdots \\ \text{vec}(V_0) \end{bmatrix},$$

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

$$\frac{\partial\Phi}{\partial\theta} = \begin{bmatrix} \dfrac{\partial\Phi}{\partial\theta_k} & \dfrac{\partial\Phi}{\partial\theta_{k-1}} & \cdots & \dfrac{\partial\Phi}{\partial\theta_0} \end{bmatrix} \in \mathbb{R}^{n\times\Xi} \tag{13}$$

where

$$\frac{\partial\Phi}{\partial\theta_i} = \begin{cases} (I_{l_{k+1}} \otimes \phi_k^T), & i = k \\ V_k^T\phi_k'(I_{l_k} \otimes \phi_{k-1}^T), & i = k-1 \\ & \vdots \\ V_k^T\phi_k'\cdots V_1^T\phi_1'(I_{l_1} \otimes q_{NN}^T), & i = 0 \end{cases},$$

where $\phi_i \triangleq \phi_i(\Phi_{i-1})$ and $\phi_i' \triangleq \partial\phi_i/\partial\Phi_{i-1}$.

In the following sections, let Φ_i^* represent the output of the i^{th} layer with the ideal weight vector θ^*. Additionally, define $\phi_i^* \triangleq \phi_i(\Phi_{i-1}^*)$ and $\phi_i^{*'} \triangleq \partial\phi_i^*/\partial\Phi_{i-1}^*$. Similarly, $\hat{\Phi}_i$ denotes the output of the i^{th} layer with the estimated weight vector $\hat{\theta}$, and define $\hat{\phi}_i \triangleq \phi_i(\hat{\Phi}_{i-1})$ and $\hat{\phi}_i' \triangleq \partial\hat{\phi}_i/\partial\hat{\Phi}_{i-1}$, respectively.

IV. Weight Adaptation Laws

A. Weight Optimizer Design

Consider a positive definite objective function defined as

$$J(\xi;\hat{\theta}) \triangleq \frac{1}{2}\xi^TW\xi$$

where $W = W^T > 0$ is a 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. The corresponding constrained optimization problem is formulated as

$$\begin{aligned} &\min_{\hat{\theta}}\ J(\xi;\hat{\theta}) \\ &\text{s.t. } c_j(\hat{\theta}) \le 0, \quad j \in \mathcal{I}, \end{aligned} \tag{14}$$

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

$$L(\xi,\hat{\theta},[\lambda_j]_{j\in\mathcal{A}}) \triangleq J(\xi;\hat{\theta}) + \sum_{j\in\mathcal{A}}\lambda_jc_j(\hat{\theta})$$

where λ_j denotes the Lagrange multiplier for each constraint, and $\mathcal{A} \triangleq \{j \in \mathcal{I} \mid c_j \ge 0\}$ represents the active set.

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

$$\dot{\hat{\theta}} = -\alpha\frac{\partial L}{\partial\hat{\theta}} = -\alpha\left(\frac{\partial J}{\partial\hat{\theta}} + \sum_{j\in\mathcal{A}}\lambda_j\frac{\partial c_j}{\partial\hat{\theta}}\right), \tag{15}$$

$$\dot{\lambda}_j = \beta_j\frac{\partial L}{\partial\lambda_j} = \beta_jc_j, \qquad \forall j \in \mathcal{A}, \tag{16}$$

$$\lambda_j = \max(\lambda_j, 0), \qquad \forall j \in \mathcal{A}, \tag{17}$$

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 $\mathcal{A}$. The Lagrange multipliers associated with inequality constraints are non-negative, i.e., $\lambda_j \ge 0$, 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 [21], where the adaptation law for Lagrange multipliers is given by $\lambda_j \leftarrow \max(\lambda_j - c_j/\mu, 0)$, with $\mu \in \mathbb{R}_{>0}$ being the penalty parameter.

At steady state, where $\dot{\hat{\theta}} = 0$ and $\dot{\lambda}_j = 0$, the KKT conditions are satisfied, i.e., $\partial L/\partial\hat{\theta} = 0$, $c_j \leq 0$, $\lambda_j \geq 0$, and $\lambda_j c_j = 0$ [21, Chap. 12 T. 12.1]. In other words, the proposed optimizer updates the DNN weights and Lagrange multipliers in a way that satisfies the KKT conditions. These conditions represent the first-order necessary conditions for optimality, guiding the updates toward candidates for a locally optimal point.

B. Calculation of the Exact Gradient of Objective Function

The adaptation law for $\hat{\theta}$ involves the gradient of the objective function with respect to $\hat{\theta}$ (i.e., $\partial J/\partial\hat{\theta}$); see (15). Since the objective function depends on the state ξ of a dynamic system, obtaining the gradient is not straightforward. Therefore, the forward sensitivity method from [29] is employed to calculate the exact gradient of the objective function.

By partially differentiating (11), the sensitivity equation of ξ with respect to $\hat{\theta}$ is first obtained as

$$\dot{\eta} = A_\xi \eta - B_\xi \frac{\partial h}{\partial \tau}\frac{\partial \hat{\Phi}}{\partial \hat{\theta}} \tag{18}$$

where $\eta \triangleq \partial\xi/\partial\hat{\theta} \in \mathbb{R}^{2n\times\Xi}$. Since the initial value of ξ is independent to $\hat{\theta}$, $\eta|_{t=0}$ is a zero matrix. The gradient of the objective function with respect to $\hat{\theta}$ is then obtained as

$$\frac{\partial J}{\partial \hat{\theta}} = \frac{\partial \xi}{\partial \hat{\theta}}^T W\xi = \eta^T W\xi \in \mathbb{R}^{\Xi}. \tag{19}$$

Equations (18) and (19) can be decomposed for each layer as

$$\begin{aligned}\dot{\eta} &= \begin{bmatrix}\eta_k & \eta_{k-1} & \cdots & \eta_0\end{bmatrix}' \\ &= A_\xi \begin{bmatrix}\eta_k & \cdots & \eta_0\end{bmatrix} - B_\xi \frac{\partial h}{\partial \tau}\begin{bmatrix}(I_{l_{k+1}} \otimes \hat{\phi}_k^T) & \cdots & (\cdot)\end{bmatrix}\end{aligned} \tag{20}$$

and

$$\frac{\partial J}{\partial \hat{\theta}} = \begin{bmatrix}\partial J/\partial\hat{\theta}_k \\ \vdots \\ \partial J/\partial\hat{\theta}_0\end{bmatrix} = \begin{bmatrix}\eta_k^T \\ \vdots \\ \eta_0^T\end{bmatrix} W\xi$$

where $\eta_i \triangleq \partial\xi/\partial\hat{\theta}_i \in \mathbb{R}^{2n\times\Xi_i}$. The exact gradient of the objective function is calculated based on (19), with the value of η obtained by simulating the sensitivity equation (18).

The proposed controller is implemented using Algorithm 1. For implementation in the discrete-time domain, it is recommended to use a sufficiently small sampling time T_s. If a large T_s is used, α and β_j should satisfy the Armijo condition [21, Chap. 3 eq. (3.4)] to ensure that the objective function decreases.

Algorithm 1 Weight Optimizer Implementation.

Input: ξ, $\hat{\theta}$, λ_j, η
Output: $\hat{\theta}$, λ_j, η
1: Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{j\}$ for all $c_j \geq 0$;
2: Determine update matrix $\dot{\eta}$ using (18);
3: Update $\eta \leftarrow \eta + \dot{\eta} \cdot T_s$;
4: Determine update directions $\dot{\hat{\theta}}$, $[\dot{\lambda}_j]_{j\in\mathcal{A}}$ using (15), (16);
5: Update weight vector $\hat{\theta} \leftarrow \hat{\theta} + \dot{\hat{\theta}} \cdot T_s$;
6: Update multipliers $[\lambda_j]_{j\in\mathcal{A}} \leftarrow [\lambda_j]_{j\in\mathcal{A}} + [\dot{\lambda}_j]_{j\in\mathcal{A}} \cdot T_s$;
7: $[\lambda_j]_{j\in\mathcal{A}} \leftarrow \max([\lambda_j]_{j\in\mathcal{A}}, 0)$;
8: Set $\mathcal{A} \leftarrow \mathcal{A} - \{j\}$ for all $\lambda_j = 0$;
9: **RETURN**;

V. Constraint Candidates

This section introduces potential weight and input constraints that can be used in the proposed neuro–adaptive controller. The controller can handle any combination of the following constraints, provided they meet the specified assumptions.

Assumption 1. *The constraint functions $c_j(\hat{\theta})$, $\forall j \in \mathcal{I}$ are convex in the τ-space and satisfy $c_j(0) \leq 0$ and $c_j(\theta^*) \leq 0$.*

Assumption 2. *The selected constraints satisfy the Linear Independence Constraint Qualification (LICQ) [21, Chap. 12 Def. 12.1].*

A. Weight Norm Constraint

The weight norm constraint $\mathbf{c}_\theta \triangleq [c_{\theta_i}]_{i\in[0,\cdots,k]} \in \mathbb{R}^{k+1}$ limits the maximum norm of each layer's weight vector, where

$$c_{\theta_i} = \|\hat{\theta}_i\|^2 - \bar{\theta}_i^2 \leq 0 \tag{21}$$

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

$$\frac{\partial \mathbf{c}_\theta}{\partial \hat{\theta}} \triangleq \begin{bmatrix}(\partial c_{\theta_0}/\partial\hat{\theta})^T \\ \vdots \\ (\partial c_{\theta_k}/\partial\hat{\theta})^T\end{bmatrix} = 2\cdot\begin{bmatrix}0 & 0 & \cdots & \hat{\theta}_0^T \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \hat{\theta}_{k-1}^T & \cdots & 0 \\ \hat{\theta}_k^T & 0 & \cdots & 0\end{bmatrix} \in \mathbb{R}^{(k+1)\times\Xi}. \tag{22}$$

B. Input Bound Constraint

Most physical systems have control input limits due to electrical and mechanical limitations. These are expressed as $\mathbf{c}_{\overline{\tau}} \triangleq [c_{\overline{\tau}_i}]_{i\in[1,\cdots,n]}$ and $\mathbf{c}_{\underline{\tau}} \triangleq [c_{\underline{\tau}_i}]_{i\in[1,\cdots,n]}$, where

$$\begin{aligned}c_{\overline{\tau}_i} &= \tau_{(i)} - \tau_{\overline{\tau}_i} \leq 0 \\ c_{\underline{\tau}_i} &= \tau_{\underline{\tau}_i} - \tau_{(i)} \leq 0\end{aligned} \tag{23}$$

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

$$\begin{aligned}\frac{\partial c_{\overline{\tau}}}{\partial \hat{\theta}} &\begin{bmatrix}(\partial c_{\overline{\tau}_1}/\partial\hat{\theta})^T\\ \vdots \\ (\partial c_{\overline{\tau}_n}/\partial\hat{\theta})^T\end{bmatrix} = -\frac{\partial \hat{\Phi}}{\partial \hat{\theta}}\\ &= -\begin{bmatrix}(I_{l_{k+1}}\otimes \hat{\phi}_k^T) & \cdots & (\cdot)\end{bmatrix} \in \mathbb{R}^{n\times \Xi},\\ \frac{\partial c_{\underline{\tau}}}{\partial \hat{\theta}} &\triangleq \begin{bmatrix}(\partial c_{\underline{\tau}_1}/\partial\hat{\theta})^T\\ \vdots \\ (\partial c_{\underline{\tau}_n}/\partial\hat{\theta})^T\end{bmatrix} = +\frac{\partial \hat{\Phi}}{\partial \hat{\theta}}\\ &= +\begin{bmatrix}(I_{l_{k+1}}\otimes \hat{\phi}_k^T) & \cdots & (\cdot)\end{bmatrix} \in \mathbb{R}^{n\times \Xi}.\end{aligned} \tag{24}$$

C. 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:

$$c_u = \|\tau\|^2 - \bar{\tau}^2 \leq 0 \tag{25}$$

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 [30]. The gradients of c_u with respect to $\hat{\theta}$ are given by

$$\frac{\partial c_u}{\partial \hat{\theta}} = -\sum_{i=1}^{n} 2\tau_{(i)}\left(\mathrm{row}_i\left(-\frac{\partial \hat{\Phi}}{\partial \hat{\theta}}\right)\right)^T = \tau^T(I_{l_{k+1}}\otimes\hat{\phi}_k^T) \in \mathbb{R}^{\Xi}. \tag{26}$$

It should be noted that constraints (23) and (25) cannot be imposed simultaneously, as their gradients (24) and (26) are linearly dependent, violating the LICQ condition.

VI. Stability Analysis

Before conducting the stability analysis, let $\tilde{\theta} \triangleq [\tilde{\theta}_i]_{i\in[0,\cdots,k]}$, where $\tilde{\theta}_i \triangleq \hat{\theta}_i - \theta_i^*$ represents the weight estimation error. The following Lemmas are introduced for the stability analysis.

Lemma 1. *If Assumptions 1 and 2 are satisfied, the angle between $\partial c_j/\partial\hat{\theta}_k$ and $\hat{\theta}_k$ is positive when c_j is active set, i.e., $(\partial c_j/\partial\hat{\theta}_k)^T\hat{\theta}_k > 0$.*

Proof. Since $\tau = -\hat{\Phi}$, using [25, Proposition 7.1.9], a linear map $T(\cdot): \hat{\theta}_k \to \tau$ can be derived:

$$\begin{aligned}\tau =& -\hat{\Phi} = -\mathrm{vec}(\hat{\Phi}) = -\mathrm{vec}(\hat{V}_k\hat{\phi}_k)\\ =& -(I_{l_{k+1}}\otimes\hat{\phi}_k^T)\mathrm{vec}(\hat{V}_k) = -(I_{l_{k+1}}\otimes\hat{\phi}_k^T)\hat{\theta}_k = T(\hat{\phi}_k)\hat{\theta}_k.\end{aligned} \tag{27}$$

Therefore, the convexity of the input constraints in τ-space (assumed in Assumption 1) holds in $\hat{\theta}_k$-space, implying that $(\partial c_j/\partial\hat{\theta}_k)^T\hat{\theta}_k > 0$.

□

Lemma 2. *If $c_j(\hat{\theta}),\ \forall j \in \mathcal{I}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}$ satisfies Assumption 1, then $\|\partial c_j/\partial\hat{\theta}_i\|$, for all $i \in [k-1,\cdots,0]$, is bounded, provided the norms of $\hat{\theta}_i$, for all $i \in [k,\cdots,i+1]$, remain bounded.*

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

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

where $\partial\tau/\partial\hat{\Phi} = -I_n$, which is bounded. From the linear mapping in (27), τ is bounded as long as $\hat{\theta}_k$ is bounded (by the condition of the lemma), and $\|\phi_k(\cdot)\|$ is bounded due to the properties of the activation functions. By Assumption 1, the function c_j is convex. The convex function has a bounded derivative with respect to τ, since τ is a bounded variable (i.e., $\partial c_j/\partial\tau$ is bounded). Furthermore, $\partial\hat{\Phi}/\partial\hat{\theta}_i$ is bounded, provided that the norms of $\hat{\theta}_i,\ \forall i \in [k,\cdots,i+1]$, are bounded. This can be verfied by using the definition of $\partial\hat{\Phi}/\partial\hat{\theta}_i$ given in (13). Consequently, $\|\partial c_j/\partial\hat{\theta}_i\|,\ \forall j \in \mathcal{I}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}$, is bounded, when $\hat{\theta}_i,\ \forall i \in [k,\cdots,i+1]$ are bounded.

□

The following theorem shows that $\hat{\theta}$ and ξ are bounded.

Theorem 1. *For the dynamical system in* (1)*, the neuro–adaptive controller* (9) *and weight adaptation laws* (15)*,* (16)*, and* (17) *ensure the boundedness of the augmented error ξ and the weight estimate $\hat{\theta}$. This holds with the weight norm constraint* (21) *and input constraints satisfying Assumption 1 and 2, provided that the control gains k_q and k_z satisfy* (32)*.*

Proof. The boundednesses of $\hat{\theta}$, ξ, and η are analyzed recursively from the last k^{th} layer to the first layer of $\hat{\Phi}$. The step-by-step analysis is described as follows.

Step 1: Boundedness of $\hat{\theta}_k, \eta_k$, and ξ

The boundedness of ξ follows from (11), using the result from [31, Chap. 4 T. 1.9], since A_ξ is a stable matrix and the term $B_\xi(\Phi^* - h(\hat{\Phi}) + \epsilon)$ is bounded due to $\|B_\xi\|_F$, $\|V_k^*\|_F$, $\|\phi(\cdot)\|$, $\|h(\cdot)\|$, and $\|\epsilon\| < \infty$.

Assume that all constraints are actve in $\mathcal{A}$ without loss of generality. The dynamics of η_k and $\hat{\theta}_k$ can be decomposed from (15) and (20) as

$$\begin{aligned}\dot{\eta}_k =& A_\xi\eta_k - B_\xi\frac{\partial h}{\partial\tau}\frac{\partial\hat{\Phi}}{\partial\hat{\theta}_k} = A_\xi\eta_k - B_\xi\frac{\partial h}{\partial\tau}(I_{l_{k+1}}\otimes\hat{\phi}_k^T)\\ \dot{\hat{\theta}}_k =& -\alpha\left[\eta_k^T W\xi + \sum_{j\in\mathcal{A}}\lambda_j\frac{\partial c_j}{\partial\hat{\theta}_k}\right]\end{aligned}$$

According to [31, Chap. 4 T. 1.9], $\|\eta_k\|_F$ is bounded, since A_ξ is a stable matrix and the term $-B_\xi(\partial h/\partial\tau)(I_{l_{k+1}}\otimes\hat{\phi}_k^T)$ is also bounded.

Let $\mathcal{V}: \mathbb{R}^{2n}\times\mathbb{R}^{\Xi}\to\mathbb{R}_{>0}$ denote the Lyapunov function:

$$\mathcal{V} = \frac{1}{2}\xi^T P\xi + \frac{1}{2\alpha}\hat{\theta}_k^T\hat{\theta}_k, \tag{28}$$

with the Lyapunov equation $A_\xi^T P + P A_\xi = -Q$, where $A_\xi < 0$, $P = P^T > 0$, and $Q > 0$. Taking the time derivative of $\mathcal{V}$ yields:

$$\begin{aligned}\dot{\mathcal{V}} = &-\frac{1}{2}\xi^T Q \xi + \xi^T P B (V_k^{*T}\phi_k^* - h(\tau) + \epsilon) \\ &- \hat{\theta}_k^T \left(\eta_k W \xi + \sum_{j\in\mathcal{A}} \lambda_j \frac{\partial c_j}{\partial \hat{\theta}_k} \right)\end{aligned} \tag{29}$$

By applying the boundedness of $\Delta \triangleq PB(V_k^{*T}\phi_k^* - h(\tau) + \epsilon)$ and $M \triangleq \eta_k W$, where $\|\Delta\| \le \bar{\Delta} < \infty$ and $\|M\|_F \le \bar{M} < \infty$, this simplifies to:

$$\begin{aligned}\dot{\mathcal{V}} \le &\left(-\frac{\lambda_{\min}(Q)}{2} + \frac{\bar{M}}{2} \right) \|\xi\|^2 + \bar{\Delta}\|\xi\| \\ &+ \frac{\bar{M}}{2}\|\hat{\theta}_k\|^2 - \sum_{j\in\mathcal{A}} \lambda_j \hat{\theta}_k^T \frac{\partial c_j}{\partial \hat{\theta}_k}.\end{aligned} \tag{30}$$

By representing the term $\partial c_{\theta_k}/\partial\hat{\theta}_k$ in the last inequality as $\partial c_{\theta_k}/\partial\hat{\theta}_k = 2\hat{\theta}_k$ using the result provided in (22), $\dot{\mathcal{V}}$ can be rewritten as

$$\begin{aligned}\dot{\mathcal{V}} \le &(\cdot) + \Big(-2\lambda_{\theta_k} + \bar{M}/2 \Big) \|\hat{\theta}_k\|^2 \\ &- \underbrace{\sum_{j\in\mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}} \lambda_j \hat{\theta}_k^T \frac{\partial c_j}{\partial \hat{\theta}_k}}_{>0, \text{ by Lemma1 and Assumption2}} \\ \le &\Big(-\lambda_{\min}(Q)/2 + \bar{M}/2 \Big) \|\xi\|^2 + \bar{\Delta}\|\xi\| \\ &+ \Big(-2\lambda_{\theta_k} + \bar{M}/2 \Big) \|\hat{\theta}_k\|^2\end{aligned} \tag{31}$$

Define $P = I_2$, leading to $Q = -A_\xi^T - A_\xi$. Consequently, the minimum eigenvalues $\lambda_{\min}(Q)$ is determined by $2\min(k_q, k_z)$, as follows from the structure of the matrix A_ξ. From (31), if the control gains k_q and k_z are chosen sufficiently large to satisfy the condition:

$$\min(k_q, k_z) > \bar{M}/2 \tag{32}$$

and if the Lagrange multiplier λ_{θ_k} for the weight norm constraint of the k^{th} layer is increased sufficiently, such that

$$-2\lambda_{\theta_k} + \bar{M}/2 < 0, \tag{33}$$

(as dictated by (16) when the corresponding constraint is violated, i.e., $c_{\theta_k} = \|\hat{\theta}_k\|^2 - \bar{\theta}_k^2 > 0$), then both ξ and $\hat{\theta}_k$ will remain bounded within the compact sets Θ_ξ and $\Theta_{\hat{\theta}_k}$, defined as

$$\Theta_\xi = \left\{ \xi \;\middle|\; \|\xi\| \le \frac{2\bar{\Delta}}{\lambda_{\min}(Q) - \bar{M}} \right\} \tag{34}$$

and

$$\Theta_{\hat{\theta}_k} = \{\hat{\theta}_k \mid \|\hat{\theta}_k\| \le \bar{\theta}_k\}. \tag{35}$$

The increase of the Lagrange multiplier λ_{θ_k} will halt once $\hat{\theta}_k$ reaches the compact set $\Theta_{\hat{\theta}_k}$. Thus, the Lagrange multiplier λ_{θ_k} is bounded.

The boundedness of the Lagrange multipliers $\lambda_j,\ \forall j \in \mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}$, can be accessed by considering the convexity of the constraints in $\hat{\theta}_k$-space. The boundedness of the remaining Lagrange multipliers associated with the weight norm constraints, $\lambda_{\theta_r},\ \forall r \in [0,\cdots,k-1]$, will be examined in Step i. Based on Assumption 1 and Lemma 1, the equality constraints $c_j \le 0,\ \forall j \in \mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}$, formconvex sets Θ_{c_j} in $\hat{\theta}_k$-space. Let $\Theta_c \triangleq \cap_{j\in\mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}}\Theta_{c_j}$ represent the intersection of these convex sets, which also contains the origin. If λ_j increases sufficiently such that $\dot{\mathcal{V}} \approx -\sum_{j\in\mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}} \lambda_j \hat{\theta}_k^T(\partial c_j/\partial\hat{\theta}_k) < 0$, then $\mathcal{V}$, and consequently $\|\hat{\theta}_k\|$, will decrease until $\hat{\theta}_k$ reaches $\Theta_\xi \cap \Theta_{\hat{\theta}_k} \cap \Theta_c$. Once $\hat{\theta}_k$ hits the boundary of Θ_c, the Lagrange multipliers will cease to increase, thus ensuring their boundedness.

Step i: Boundedness of $\hat{\theta}_i$ and $\eta_i,\ i \in [k-1,\cdots,0]$

The boundedness of the Frobenius norm of the gradient $\partial\hat{\Phi}/\partial\hat{\theta}_i$ can be obtained from its definition in (13). This relies on the fact that the boundedness of $\|\hat{\theta}_i\|,\ \forall i \in [k,\cdots,i+1]$, was already shown in the previous step (i.e., Step 1 to Step $i+1$) and the activation functions $\hat{\phi}_i$ and their derivative $\hat{\phi}_i'$ are bounded.

The dynamics of η_i and $\hat{\theta}_i$ for all $i \in [k-1,\cdots,0]$ are represented as

$$\dot{\eta}_i = A_\xi \eta_i - B_\xi \frac{\partial h}{\partial \tau}\frac{\partial \hat{\Phi}}{\partial \hat{\theta}_i}$$

and

$$\dot{\hat{\theta}}_i = -\alpha \left[\eta_i^T W \xi + 2\lambda_{\theta_i}\hat{\theta}_i + \sum_{j\in\mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}} \lambda_j \frac{\partial c_j}{\partial \hat{\theta}_i} \right].$$

According to [31, Chap. 4 T. 1.9], $\|\eta_i\|_F$ is bounded because A_ξ is a stable matrix and the terms $\|B_\xi\|, \|(\partial h/\partial\tau)\|_F$, and $\|\partial\hat{\Phi}/\partial\hat{\theta}_i\|_F$ are bounded. When λ_{θ_i} is generated due to a violation of the weight norm constraint, $\hat{\theta}_i$ remains bounded because the term $-2\alpha\lambda_{\theta_i}$ is stable, and the residual terms $\|\eta_i W\xi\|$, λ_j, and $\|(\partial c_j/\partial\hat{\theta}_i)\|_F$ for all $j \in \mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}$ are bounded, as demonstrated in Step 1 and by Lemma 2. The boundedness of each λ_{θ_i} also can be obtained, assuming that λ_{θ_i} is sufficiently increased regulating $\|\theta_i\|$ into the origin until c_{θ_i} is satisfied.

Therefore, starting from the boundedness of ξ, $\hat{\theta}_k$, and η_k in the k^{th} layer, the boundednesses of $\hat{\theta}_i$ and η_i for the remaining layers $i \in [0,\cdots,k-1]$ can be established recursively, down to the input layer ($i = 0$). Thus, $\hat{\theta}$, ξ, and η are bounded because $\hat{\theta}_i, \eta_i,\ \forall i \in [0,\cdots,k]$ and ξ are bounded. Furthermore, since the estimated weight vector $\hat{\theta}$ is bounded, the weight estimation error $\tilde{\theta}$ is also bounded, as θ^* is bounded according to the universal approximation theorem.

□

VII. Simulation

A. Setup

The proposed CoNAC was validated using a two-link manipulator model, as depicted in Fig. 2, adapted from [32]. The parameters $q_p, q_{d_p}, \tau_p, m_p, l_p, l_{cp}, b_p$, and f_{c_p} denote the joint

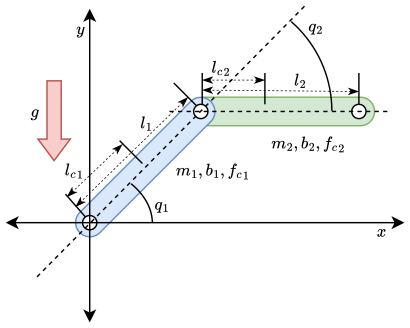


Fig. 2: Two-link manipulator model.

TABLE I: System model parameters.

Symbol	**Description**	**Link 1**	**Link 2**
m_1, m_2	Mass of link	23.902 (kg)	3.88 (kg)
l_1, l_2	Length of link	0.45 (m)	0.45 (m)
l_{c1}, l_{c2}	COM of link	0.091 (m)	0.048 (m)
b_1, b_2	Viscous coefficient	2.288 (Nms)	0.172 (Nms)
f_{c1}, f_{c2}	Friction coefficient	7.17 (Nm)	1.734 (Nm)

angle, desired joint angle, torque, mass, length, center of mass, viscous coefficient, and friction coefficient, respectively, for link $p \in [1, 2]$. The values of the system model parameters are provided in Table I. The desired trajectory for $q = [q_1, q_2]^T$ was defined as

$$q_d(t) = \begin{bmatrix} q_{d1} \\ q_{d2} \end{bmatrix} = \begin{bmatrix} +\cos(0.5\pi t) + 1 \\ -\cos(0.5\pi t) - 1 \end{bmatrix}.$$

The control input saturation function was defined as $h(\tau) \triangleq \tau / \|\tau\| \cdot \mathrm{SSF}_L^U(\|\tau\|)$, where smooth saturation function (SSF) was adopted from [33] and is given by

$$\mathrm{SSF}_L^U(\|\tau\|) = \frac{\|\tau\|}{(1 + (\|\tau\|/\bar{\tau})^p)^{1/p}}. \tag{36}$$

with p being the smoothing factor. The effect of p and the boundedness of $\|\partial h / \partial \tau\|_F$ is shown in Fig. 3. The parameters of the control input saturation function were selected as $p = 100$ and $\bar{\tau} = 50$.

In addition to this physically imposed control input saturation, the input norm constraint (25) was imposed to ensure that the control input τ stays within the unsaturation region of $h(\tau)$ and to prevent inefficient use of the input. With a sufficiently large value for p, the input norm constraint essentially matches the control input saturation function. Note that among the selected controllers given below, only the proposed CoNAC can rigorously handle this input norm constraint.

Four controllers were examined for a comparative study. The first was the Backstepping Controller (BSC), used as the baseline. The second was the DNN-based Backstepping Controller (DNN-BSC), an existing neuro–adaptive control method where a DNN was employed to learn and compensate for the lumped system uncertainty function f in the BSC. While this method addressed the weight norm constraint via a projection operator, it did not account for either the input norm or input bound constraints. The third was DNN-BSC augmented with an auxiliary system presented in [9], [10], [17]

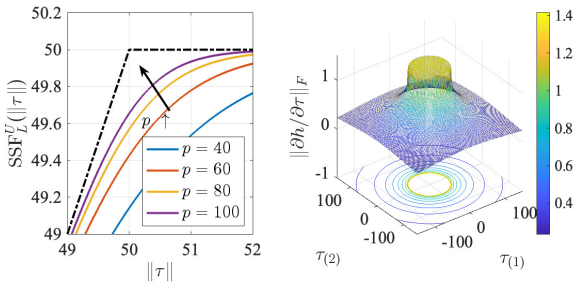


Fig. 3: Effect of parameter p in the control input saturation function h and boundedness of $\|\partial h / \partial \tau\|_F$.

TABLE II: Properties of the controllers used in simulation

	Handling Capability		
	System Uncertainty	Weight Norm Constraint	Input Norm Constraint
BSC	X	X	X
DNN-BSC	O	O (by projection)	X
DNN-BSC-A	O	O (by projection)	△ (by aux. system)
CoNAC	O	O (by optimization)	O (by optimization)

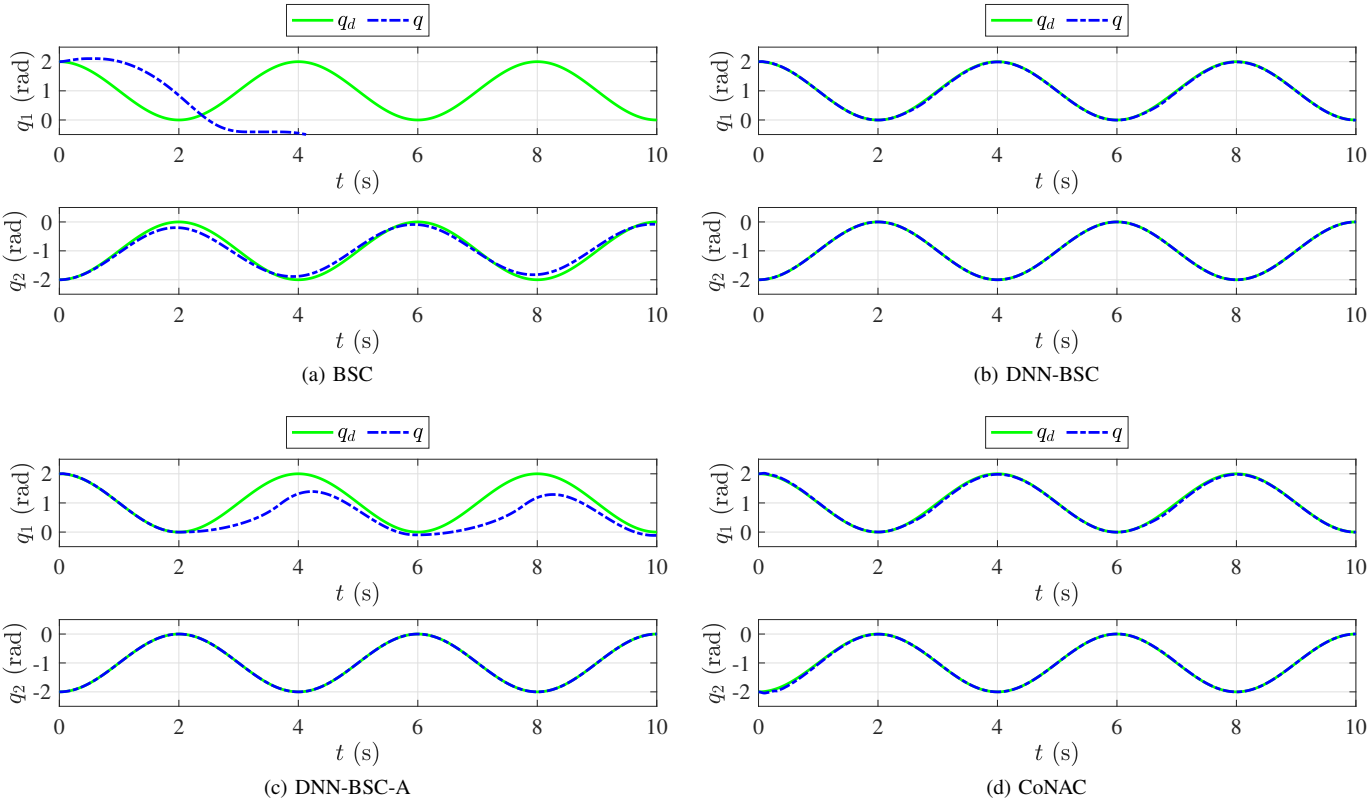
(DNN-BSC-A), which handled the (linear) input saturation constraint but not the nonlinear input norm constraint. As a result, an approximation of the input norm constraint was used as an input bound constraint. Lastly, the proposed controller, CoNAC, rigorously considered system uncertainties, the weight norm constraint, and input constraints within a constrained optimization framework. The properties of these four controllers are summarized in Table II.

The BSC used the control law defined in (7) with $\hat{f} = [0, 0]^T$. Since BSC did not consider the unknown system dynamics, the approximation term $\hat{f}$ was set to zero. The control law for DNN-BSC was the same as BSC, but the unknown system dynamics were approximated by a DNN i.e., $\hat{f} \approx \hat{\Phi}$. The adaptation law for DNN-BSC, as presented in [8], was defined by

$$\dot{\hat{\theta}} = \mathrm{Proj}_{\Omega}[\alpha(\partial \hat{\Phi} / \partial \hat{\theta}) M_0^{-1} \tilde{z}], \tag{37}$$

where $\mathrm{Proj}_{\Omega}(\cdot)$ is the projection operator [26, Appendix E, eq. (E.4)], which projects an input vector onto a convex set Ω. The convex set was defined as $\Omega \triangleq \{\Omega_0 \cap \cdots \cap \Omega_k\}$, where $\Omega_i \triangleq \{\hat{\theta}_i \mid c_{b_i} \leq 0\}$, $\forall i \in [0, \cdots, k]$, representing the weight norm constraint (21).

The control law for DNN-BSC-A was the same as DNN-BSC, 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 (25), an approximated version of the input norm constraint was used as an input bound constraint (23) with $\overline{\tau}_i = -\underline{\tau}_i = (\bar{\tau}/\sqrt{2} + \bar{\tau})/2$. The comparison between the original input norm constraint and its approximation is shown in Fig. 6. The auxiliary system is defined as $\dot{\zeta} =$



(a) BSC (b) DNN-BSC

(c) DNN-BSC-A (d) CoNAC

Fig. 4: Comparison of the tracking performance across the selected controllers.

$A_\zeta \zeta + B_\zeta \Delta\tau, \quad \zeta|_{t=0} = 0$, where $\zeta \in \mathbb{R}^n$ denotes the auxiliary state, $A_\zeta = -[20, 0; 0; 20]$, $B_\zeta = [10, 0; 0, 10]$, and $\Delta\tau_{(i)} = \tau_{(i)} - \text{sat}(\tau_{(i)}, \overline{\tau}_i, \underline{\tau}_i)$ The auxiliary state variables were used in the adaptation law (37) by substituting $\tilde{z}$ with $\tilde{z} + \zeta$.

The proposed CoNAC directly approximated the control law using the DNN as defined in (9). The update rates for the multipliers were set as $\beta_j = 0.1$. The weight matrix W was selected as $W = \text{diag}([5, 1, 15, 15])]$.

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., $q_{NN} = [{q_d}^T, 1]^T$ 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 $\bar{\theta}_0 = 20, \bar{\theta}_1 = 30, \bar{\theta}_2 = 40$, and $\bar{\tau} = 50$ (from Eq. (36)). The control parameters for all the controllers were set as $k_q = 1.1, k_z = 10, M_0 = I_2, C_0 = I_2, G_0 = [0, 0]^T$. The sampling time of the simulations was selected as $T_s = 10^{-4}$.

B. System Uncertainty Handling

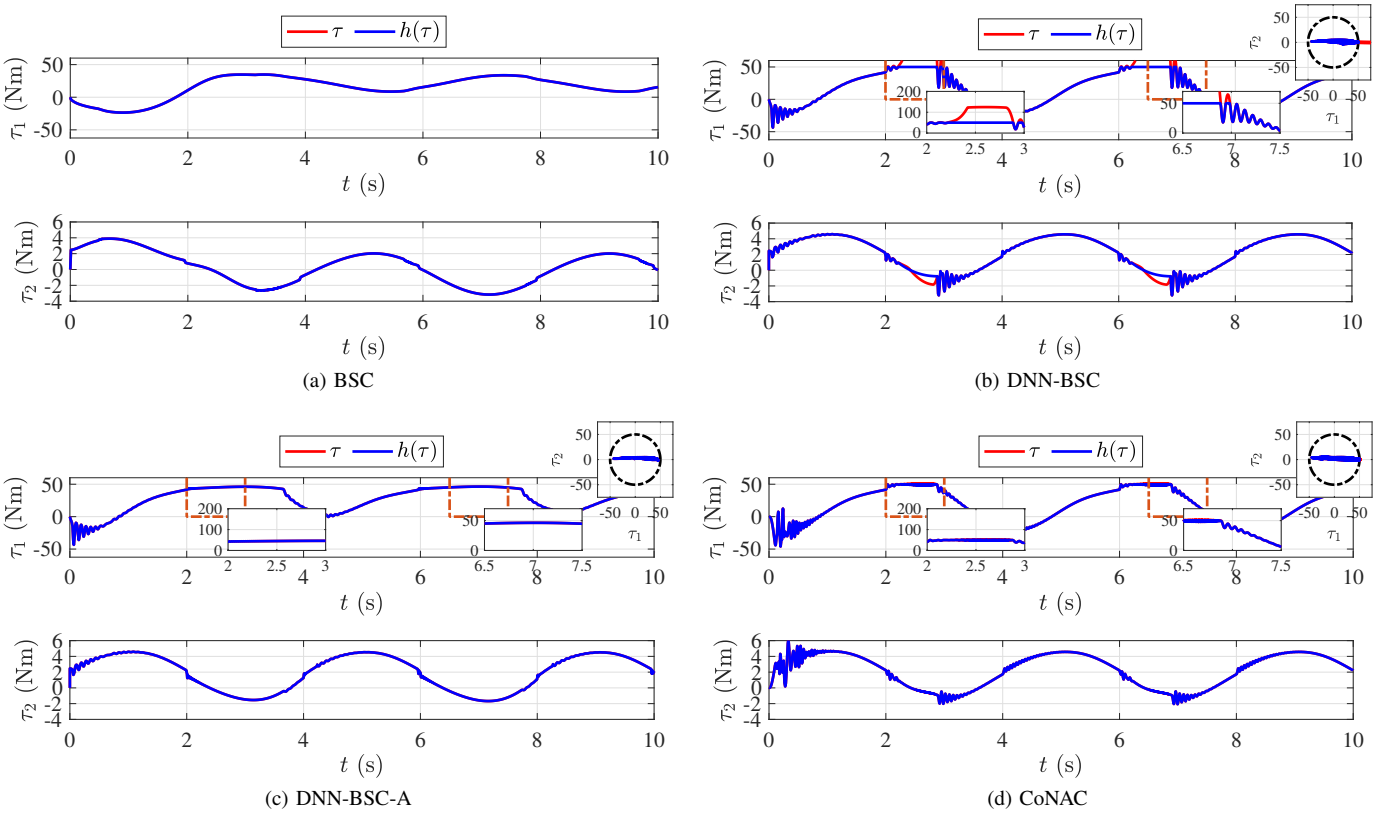
The tracking results of the selected controllers are shown in Fig. 4. To demonstrate the effectiveness of using DNNs for compensating the lumped system uncertainty function f, the gains k_q and k_z for BSC were intentionally selected as small values, resulting in a weak ability to handle these uncertainties. As a result, BSC failed to track the reference trajectory, as shown in Fig. 4a.

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. 4b. Fig. 4c 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 VII-C.

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. 4d.

C. Input Norm Constraint Handling

The resulting control input τ and physically saturation control input $h(\tau)$ for the selected controllers are shown in Fig. 5. As illustrated in Fig. 5a, 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., $\tau > h(\tau)$) at several points; see Fig. 5b. 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.



(a) BSC (b) DNN-BSC

(c) DNN-BSC-A (d) CoNAC

Fig. 5: Comparison of the control input τ and the physically saturated control input $h(\tau)$ across the selected controllers.

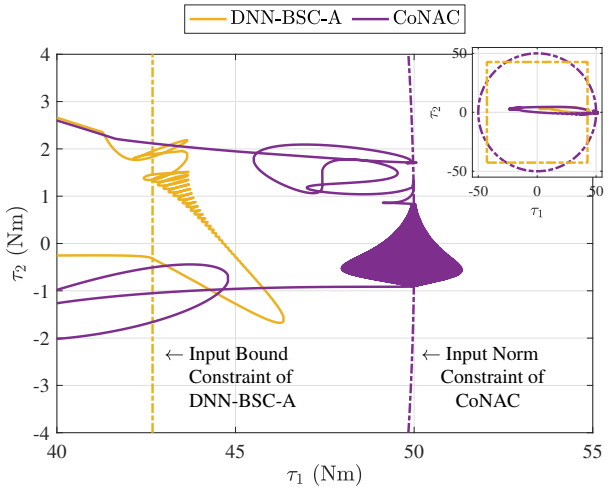


Fig. 6: Control input paths of DNN-BSC and CoNAC during the time interval from 5 s to 8 s.

Such high-frequency oscillations may induce instability in the control system or cause fatigue in the actuators.

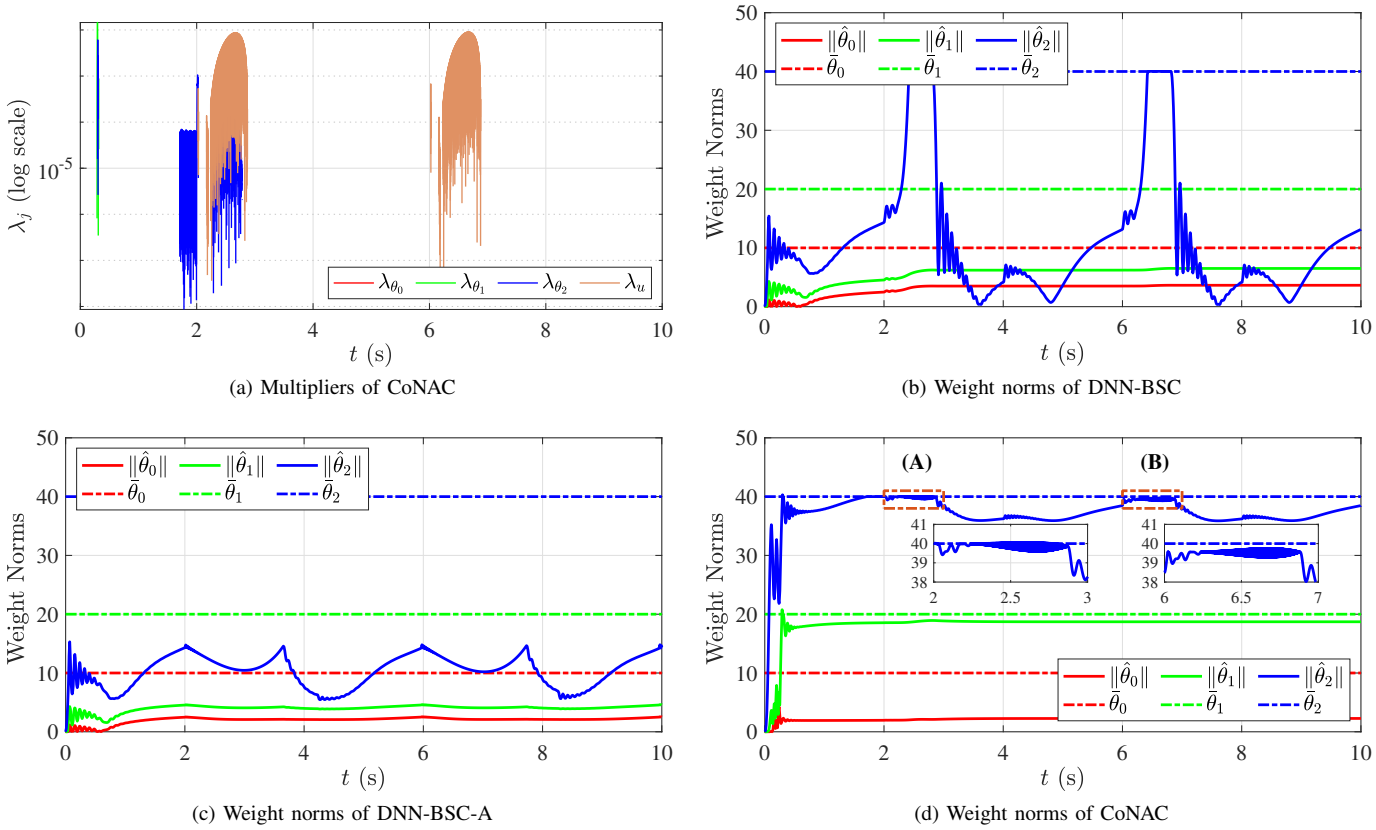
On the other hand, both DNN-BSC-A and CoNAC successfully handled their imposed input constraints, as shown in Fig. 5c and Fig. 5d, respectively, without causing notable oscillations in the control input τ even after the input constraint was activated. However, the tracking performance of DNN-BSC-A was 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. 6). 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. 6, 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 its Lagrange multiplier adjusted as soon as the constraint was violated.

D. 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. 7. 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. 7b, the weight norm of the last layer (i.e., $\|\hat{\theta}_2\|$) 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



(a) Multipliers of CoNAC

(b) Weight norms of DNN-BSC

(c) Weight norms of DNN-BSC-A

(d) Weight norms of CoNAC

Fig. 7: Lagrange multipliers of CoNAC and weight norms of DNN-BSC, DNN-BSC-A, and CoNAC.

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. 7c. 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.7d. 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.7a). Notably, the weight norms of the first and second layers ($\|\hat{\theta}_0\|$ and $\|\hat{\theta}_1\|$) remained nearly constant throughout the control period. The weight norm of the last layer $\|\hat{\theta}_2\|$ stabilized within the upper bound by around 6.5 seconds (see Fig. 7d (B)), coinciding with the activation of the input norm constraint. This quasi-static behavior of the weight norm (i.e., $d\hat{\theta}/dt = -\alpha\partial L/\partial\hat{\theta} \approx 0$) 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. 7d (A)), the weight norm of the last layer reached the upper limit earlier, despite similar control 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.

VIII. 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.

REFERENCES

[1] P. Ioannou and B. Fidan, *Adaptive control tutorial*. SIAM, 2006.

[2] G. Tao, *Adaptive control design and analysis*, vol. 37. John Wiley & Sons, 2003.

[3] J. A. Farrell and M. M. Polycarpou, *Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches*. John Wiley & Sons, 2006.

[4] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable adaptive neural network control*, vol. 13. Springer Science & Business Media, 2013.

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

[6] J. Liu, *Radial Basis Function (RBF) neural network control for mechanical systems: design, analysis and Matlab simulation*. Springer Science & Business Media, 2013.

[7] S. S. Ge and C. Wang, "Direct adaptive nn control of a class of nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 214–21, 2002.

[8] O. S. Patil, D. M. Le, M. L. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Systems Letters*, vol. 6, pp. 1855–1860, 2021.

[9] K. Esfandiari, F. Abdollahi, and H. Talebi, "A stable nonlinear in parameter neural network controller for a class of saturated nonlinear systems," *International Federation of Automatic Control Proceedings Volumes (IFAC)*, vol. 47, no. 3, pp. 2533–2538, 2014.

[10] K. Esfandiari, F. Abdollahi, and H. A. Talebi, "Adaptive control of uncertain nonaffine nonlinear systems with input saturation using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2311–22, 2015.

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

[12] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions. arxiv 2017," *arXiv preprint arXiv:1705.05502*.

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

[14] R. G. Hart, E. J. Griffis, O. S. Patil, and W. E. Dixon, "Lyapunov-based physics-informed long short-term memory (LSTM) neural network-based adaptive control," *IEEE Control Systems Letters*, 2023.

[15] K. Esfandiari, F. Abdollahi, and H. A. Talebi, *Neural network-based adaptive control of uncertain nonlinear systems*. Springer, 2022.

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

[17] S. P. Karason and A. M. Annaswamy, "Adaptive control in the presence of input constraints," in *1993 american control conference*, pp. 1370–1374, IEEE.

[18] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 334–344, 2015.

[19] E. Arefinia, H. A. Talebi, and A. Doustmohammadi, "A robust adaptive model reference impedance control of a robotic manipulator with actuator saturation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 2, pp. 409–420, 2017.

[20] G. Peng, C. Yang, W. He, and C. P. Chen, "Force sensorless admittance control with neural learning for robots with actuator saturation," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 4, pp. 3138–3148, 2019.

[21] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

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

[23] J. Wang, F. Yu, X. Chen, and L. Zhao, "Admm for efficient deep learning with global convergence," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 111–119.

[24] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *International conference on machine learning*, pp. 2722–2731, PMLR.

[25] D. S. Bernstein, *Matrix mathematics: theory, facts, and formulas*. Princeton university press, 2009.

[26] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[27] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conference on learning theory*, pp. 2306–2327, PMLR.

[28] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International conference on machine learning*, vol. 30, p. 3, Atlanta, GA.

[29] B. Sengupta, K. J. Friston, and W. D. Penny, "Efficient gradient computation for dynamical models," *Neuroimage*, vol. 98, pp. 521–7, 2014.

[30] K. Choi, J. Kim, and K.-B. Park, "Generalized model predictive torque control of synchronous machines," *IEEE/ASME Transactions on Mechatronics*, 2024.

[31] C. A. Desoer and M. Vidyasagar, *Feedback systems: input-output properties*. SIAM, 2009.

[32] E. D. Markus, J. T. Agee, and A. A. Jimoh, "Trajectory control of a two-link robot manipulator in the presence of gravity and friction," in *2013 Africon*, pp. 1–5, IEEE.

[33] J. D. Vasquez-Plaza, J. M. R. Scarpetta, T. M. Hansen, R. Tonkoski, and F. A. Rengifo, "Smooth mathematical representation of the der_a aggregated model," *IEEE Access*, vol. 11, pp. 101398–101408, 2023.