Thesis for Master's Degree

Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Euler-Lagrange Systems

Myeongseok Ryu

School of Mechanical and Robotics Engineering

Gwangju Institute of Science and Technology

2025

석 사 학 위 논 문

오일러-라그랑주 시스템을 위한 제약 최적화 기반 신경망 적응 제어

유명석

기 계 로 봇 공 학 부

광 주 과 학 기 술 원

2025

Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Euler-Lagrange Systems

Advisor: Kyunghwan Choi

by

Myeongseok Ryu

School of Mechanical and Robotics Engineering

Gwangju Institute of Science and Technology

A thesis submitted to the faculty of the Gwangju Institute of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the School of Mechanical and Robotics Engineering

Gwangju, Republic of Korea

December 11, 2024

Approved by

Professor Kyunghwan Choi

Committee Chair

Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Euler-Lagrange Systems

Myeongseok Ryu

Accepted in partial fulfillment of the requirements for

the degree of Master of Science

December 11, 2024

Dedicated to my family.

MS/ME 20231022 Myeongseok Ryu. Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Euler-Lagrange Systems . School of Mechanical and Robotics Engineering. 2025. 72p. Advisor: Prof. Kyunghwan Choi.

Abstract

This thesis 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 an ideal stabilizing control law which compensates for lumped system uncertainties while addressing both types of constraints. The weight adaptation laws are derived from constrained optimization theory, ensuring first-order optimality conditions at steady state. The controller's stability is rigorously analyzed using Lyapunov theory, ensuring bounded tracking errors and DNN weights. Two numerical simulations were constructed to compare CoNAC with other benchmark controllers. The simulations demonstrated effectiveness of CoNAC in tracking error regulation and satisfaction of constraints.

MS/ME 20231022 유명석. 오일러-라그랑주 시스템을 위한 제약 최적화 기반 신경망 적응 제어 . 기계로봇공학부. 2025. 72p. 지도교수: 최경환.

국 문 요 약

본 논문은 신경망 가중치와 제어 입력의 제약 조건이 있는 불확실한 오일러-라그랑주 시스템을 위한 제약 최적화 기반 신경망 적응 제어기를 제안한다. 깊은 신경망은 부여된 두 제약조건을 만족하는 동시에 시스템의 불확실성을 보상하며 원하는 제어 법칙을 근사하도록 사용되었다. 가중치의 적응 법칙은 제약 최적화 이론으로 부터 유도되었으며, 정상 상태에서 1차 최적성 조건을 만족시킨다. 제어기의 안정성은 리아푸노프 이론을 사용하여 분석되었으며, 추종 오차와 신경망 가중치의 크기가 제한됨을 보여준다. 제안된 제어기를 다른 벤치마크 제어기와 비교하기 위해 두 개의 시뮬레이션이 시행되었다. 시뮬레이션을 통하여 추종 성능과 제약 조건 만족에 제안된 제어기가 다른 제어기에 비하여 좋은 지표를 가진다는 것을 보였다.

Contents

List of Figures

Chapter 1
Introduction

1.1 Background

Neural Networks in Control Design

Recent advances in deep learning field have shown that neural networks (NNs) can be used in a wide range of applications. A fundamental idea of deep learning is to utilize the well-known universal approximation capability of the NNs, which allows them to approximate any smooth function over a compact set with minimal approximation error. Using this property, various architectures of NNs have been introduced such as convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for time-series data, and long short-term memory (LSTM) networks for sequential data. These advances have motivated many researchers in the control field to leverage NNs for control design.

In general, the NNs are used in control design as a parameter estimator or controller. In other words, the NNs are trained to produce corresponding output to certain input (*e.g.* real system parameters or desired control input from current system output). Hence, designers need to make a dataset which contains input-output pairs, to train the NNs. Random control inputs are typically applied to collect the input-output pairs by exploring the interested operating domain. Then, the NNs are trained to approximate the input-output mapping using the dataset via supervised learning methods to solve regression problem. For example, as the parameter estimators, 1-Dimensional convolutional neural network (1D-CNN) is used for friction potential estimation [1]. Similarly, CNN is used to estimate the road conditions (*e.g.* dry or icy condition) by extracting features of road image from built-in camera [2]. A time delay neural network (TDNN) is used to estimate tire-road friction coefficient for vehicle control [3]. In addition, for motor systems, specific architectures of NNs are used to estimate for the nonlinearity of voltage source inverter and synchronous machine [4].

On the other hand, as the controller, the NNs are trained to imitate given desired control input. Thus, designers should design desired controller as the output of input-output pair for training as well. In [5,6], CNNs are used to produce steering angle from raw pixel data of camera for vehicle control. This literature shows that end-to-end controller can be achieved by using NNs. As an optimal feedforward torque control of synchronous machines, NNs are used to produce optimal reference current [7–9] which is calculated using nonlinear programming (NLP). The literature reported that the NNs can provide the optimal reference current with less computational cost than existing methods. This is because, in general, NLP needs high computational cost to solve the optimization problem exhibiting unsuitability for real-time control. In contrast, once the NNs are trained, they can be used for real-time control, since they consist of simple matrix affine operations and element-wise nonlinear functions. Especially, in [9], multi-objective hyperparameter optimization is utilized to obtain optimal structure of NN regarding floating point operations (FLOPS) which can be considered as computational cost, and current errors, since motor systems require high frequency of control input.

In conclusion, if the NNs are well-trained, the NNs can be used as an estimator or controller in the control system, with smaller computational cost (if online-train is not conducted) and sufficient accuracy. However, if an operating point goes outside the interested operating domain, the NNs can not provide the accurate estimation or control input. This is because, the NNs are train through the dataset which is collected in the interested operating domain (i.e. they are trained for interpolation not extrapolation of the dataset). Furthermore, the NNs are inherently black-box models whose input-output mapping is not interpretable [10,11]. This causes stability and safety problems of the system, since the possibility of unexpected behavior (*e.g.* excessively large or non-proper control input) of the controller exists. Therefore, the stability analysis of the control system with NNs should be conducted to ensure the stability and safety in a perspective of control theory.

Neuro-Adaptive Control

From 1980s, as a branch of adaptive control, neuro-adaptive control (NAC) has been developed to leverage the NNs for control design to approximate unknown system dynamics or entire control laws [12,13]. The conventional adaptive control is a control method that adapts the control parameters to compensate for uncertainties in the system dynamics, since, in practice, real systems often contains uncertainties due to unmodeled dynamics, parameter variations, or external disturbances which can significantly degrade control performance and lead to instability. Similarly, NAC methods approximate the uncertainties via NNs and adapts the weights of the NNs. For more details of conventional adaptive control, the reader is referred to [14, 15].

The adaptive control methods typically ensure the stability of the control system including adaptation law, in the sense of Lyapunov by conducting Lyapunov stability analysis or deriving adaptation law using Lyapunov stability analysis. As a branch of adaptive control, the adaptation laws of NAC methods also ensure the stability. This means that the NAC methods have online adaptation (train) capability with stability guarantees. It is notable, that the adaptation laws of NAC methods (which will be presented in the Section 1.2 with a simple example) use the current observed error as the feedback signal to be backpropagated for adaptation (i.e. generally it is prediction error of NNs). Since the NN's weights are adapted according to current observed error which is dependent on current NN's weights, the offline adaptation methods can not be conducted. This feature makes NAC methods be more close to reinforcement learning methods which attempts to maximize expectation value of reward (i.e. reward is typically defined as sign changed tracking error.) and need to implemented in real-time to obtain current reward.

Besides, since NAC is based on control theory, relatively simpler NN architectures and adaptation methods are used due to their brief mathematical expression. Most widely utilized architectures are single hidden layer neural networks (SHLNNs) [16–21] and radial basis function neural networks (RBFNNs) [22–27]. Recently, deep NNs (DNNs) are utilized in NAC with stability guarantees [28]. The DNNs are more effective for complex system approximation than shallow NNs, since it offer greater expressive power with same number of neurons [29]. Additionally, variations of DNNs, such as long short-term memory (LSTM) networks for time-varying dynamics [30] and physics-informed neural networks (PINNs) for leveraging physical system knowledge [31], have further extended the capabilities of neuro-adaptive control systems.

These various NNs are typically employed to improve the conventional control methods. In [16, 20], the NNs are used to compensate for system uncertainties in the feedback linearization control. Similarly, the NNs are used to approximate the unknown dynamics in the backstepping control for higher-order system in [18, 25, 26, 32]. Besides, sliding mode control (SMC), impedance control and admittance control are also developed with NNs in [24], [22] and [23], respectively. A few of the NAC methods are utilized NNs to approximate the entire control law [19]. Aforementioned NAC methods have shown the effectiveness of the NNs in control design to approximate system uncertainties and control law.

However, there are some limitations in the existing NAC methods. In following sections, the limitations are presented, and the research objective is suggested.

1.2 Simple Example of Neuro-Adaptive Control

In this section, a simple example of neuro-adaptive control (NAC) is presented. Consider a control affine system as follows:

$$\dot{x} = f(x) + h(u)$$

where $x \in \mathbb{R}^n$ is the state, $f : \mathbb{R}^n \to \mathbb{R}^n$ is the unknown dynamics, $h : \mathbb{R}^n \to \mathbb{R}^n$ is control input saturation function, and $u \in \mathbb{R}^n$ is the control input. The objective of this control problem is to design a control law u such that the state x converges to the origin. The unknown dynamics $f(x)$ can be approximated via NNs according to the universal approximation theorem presented in Theorem 2.3.1, as follows:

$$f(x) = \Phi(x; \theta^*) + \epsilon$$

where $\Phi(\cdot)$ denotes a universal approximator (*e.g.* SHLNNs, RBFNNs or DNNs), θ^* is the ideal weight and ϵ is the approximation error. Note that θ^* is supposed to be constant ($\dot{\theta}^* = 0$) and bounded.

Ignoring control input saturation, a desired feedback-linearization based stabilizing control law can be developed as follows:

$$u^* = -\Phi(x; \theta^*) - \epsilon - kx$$

where k is a positive control gain. Using the estimation of ideal weight $\hat{\theta}$, the control law can be approximated as follows:

$$u = -\Phi(x; \hat{\theta}) - kx$$

where $\hat{\theta}$ is the estimated weight. The closed-loop dynamics can be described as follows:

$$\dot{x} = \Phi^* + \epsilon + h(-kx - \hat{\Phi})$$

where $\Phi^* = \Phi(x;\theta^*)$ and $\hat{\Phi} = \Phi(x;\hat{\theta})$.

The adaptation law can be derived based on the Lyapunov stability analysis. To simplify the derivation, ignore the control input saturation. Then, taking the time derivative of the Lyapunov function candidate $\mathcal{V} = (1/2)x^T x + (1/2\alpha)\tilde{\theta}^T\tilde{\theta}$ where $\tilde{\theta} \triangleq \hat{\theta} - \theta^*$ and $\alpha \in \mathbb{R}_{>0}$ denotes adaptation gain (learning rate), yields:

$$\begin{aligned}
\dot{\mathcal{V}} =& x^T(-kx + \Phi^* - \hat{\Phi} + \epsilon) + \frac{1}{\alpha}\tilde{\theta}^T\dot{\hat{\theta}} \\
=& -kx^T x + x^T(\Phi^* - \hat{\Phi} + \epsilon) + \frac{1}{\alpha}\tilde{\theta}^T\dot{\hat{\theta}} \\
=& -kx^T x + x^T(-\frac{\partial\hat{\Phi}}{\partial\hat{\theta}}\tilde{\theta} + \mathcal{O}(\|\tilde{\theta}\|^2) + \epsilon) + \frac{1}{\alpha}\tilde{\theta}^T\dot{\hat{\theta}} \\
=& -kx^T x + \tilde{\theta}^T\left(\frac{1}{\alpha}\dot{\hat{\theta}} - \frac{\partial\hat{\Phi}}{\partial\hat{\theta}}^T x\right) + x^T\Delta
\end{aligned}$$

where $\mathcal{O}(\|\tilde{\theta}\|^2)$ is the higher order term, and $\Delta \triangleq \mathcal{O}(\cdot) + \epsilon$ is the lumped disturbance term. Assuming that Δ is sufficiently small, the adaptation law which realizes $\dot{\mathcal{V}} \approx -kx^T x < 0$ can be derived as follows:

$$\dot{\hat{\theta}} = \alpha\frac{\partial\hat{\Phi}}{\partial\hat{\theta}}^T x.$$

However, Δ can dominate the system when an initial (or on certain domain) estimation error $\tilde{\theta}$ is large. In result, the parameter drift can occur due to Δ (i.e. the parameter estimation $\hat{\theta}$ increases to infinity over time (see [14])). In practical physical applications, the parameter drift is crucial since it makes the control input reach to limitation of actuators (i.e. the amplitude of control input is dependent on weights). This can destruct the system performance and lead to the system instability.

1.3 Research Objective

To address the above issues (boundedness of weights and control input saturation), constrained optimization offers a promising approach. By formulating the NAC problem as an optimization problem with constraints, it is possible to adapt the NN weights (minimize an objective function (e.g., tracking error)) satisfying the constraints regarding with both weight boundedness and control input saturation. Constrained optimization provides a theoretical framework for defining optimality of the problem and presents numerical methods for finding solutions [33]. To the best of the authors' knowledge, no prior work has applied constrained optimization theory to adaptive control systems with real-time weight adaptation. Only a few of literature are introduced constrained optimization based methods to train NNs more efficiently [34–36].

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 NAC. In summary, the objective of this thesis is to develop a NAC that can ensures boundedness of the weights and satisfaction of the input saturation via optimization method.

1.4 Outline of the Thesis

The remainder of this thesis is organized as follows. Preliminaries for the thesis are presented in Chapter 2. Using the preliminaries the proposed constrained optimization-based neuro-adaptive controller (CoNAC) is presented over Chapter 3 and Chapter 4. Each chapter, at first, introduces the exiting methods concerning weight boundedness, and input saturation respectively. Then, the details of the proposed CoNAC are presented, and simulation results are reported. Finally, Chapter 5 concludes the thesis and suggests future work.

Chapter 2
Preliminaries

This chapter presents the necessary background and mathematical tools required for the subsequent chapters. Since the proposed controller (constrained optimization-based neuro-adaptive controller (CoNAC)) is based on adaptive control and constrained optimization theory and deep neural networks (DNNs), this chapter covers the following topics: control theory, constrained optimization theory, and the approximation theory.

2.1 Related Control Theory

In this thesis, the Euler-Lagrange systems will be used as target system since many engineering systems can be modeled using Euler-Lagrange systems (*e.g.* aerospace, robotics, and automotive applications). Using system matrices $M(q) \in \mathbb{R}^{n\times n}$, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$, and $G(q) \in \mathbb{R}^n$, and external forces $F(q) \in \mathbb{R}^n$, the Euler-Lagrange system can be rewritten as

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

where $q \in \mathbb{R}^n$ denotes generalized state variables and $\tau \in \mathbb{R}^n$ denotes generalized control input. The Euler-Lagrange system can be transformed into a second-order control-affine system by defining the state variables $x_1 \triangleq q$ and $x_2 \triangleq \dot{q}$ as

$$\begin{aligned}\dot{x}_1 =& x_2\\ \dot{x}_2 =& f(x,t) + g(x,t)u\end{aligned}$$

where $x \triangleq [x_1, x_2]^T \in \mathbb{R}^{2n}$, $u \triangleq \tau$, $f(x,t) \triangleq M^{-1}(-Cx_2 - G - F)$ and $g(x,t) \triangleq M^{-1}$ are system functions.

Using backstepping control approach, the system can be broken down into lower dimension subsystem by generating the auxiliary control input to regulate the higher dimension original system [37]. Considering tracking error with a given smooth reference signal $r_1(t) \in \mathbb{R}^n$, the tracking error and its time-derivative are defined as

$$e_1 \triangleq x_1 - r_1, \quad \dot{e}_1 = \dot{x}_1 - \dot{r}_1 = x_2 - \dot{r}_1.$$

Using the Lyapunov function defined as $\mathcal{V}_1 \triangleq (1/2)e_1^T e_1$, the time-derivative of $\mathcal{V}_1$ yields

$$\dot{\mathcal{V}}_1 = e_1^T \dot{e}_1 = e_1^T (x_2 - \dot{r}_1)$$

The auxiliary control input $r_2 \in \mathbb{R}^n$ which realizes $\dot{\mathcal{V}}_1 < 0$ can be defined as $r_2 \triangleq -k_1 e_1 + \dot{r}_1$ for some constant $k_1 \in \mathbb{R}_{>0}$. Let $e_2 \triangleq x_2 - r_2 = x_2 - (-k_1 e_1 + \dot{r}_1)$ denotes the tracking error of x_2.

Then, the time-derivative of the Lyapunov function $\mathcal{V}_2 \triangleq (1/2)e_1^T e_1 + (1/2)e_2^T e_2$ yields

$$\begin{aligned}\dot{\mathcal{V}}_2 =& e_1^T (x_2 - \dot{r}_1) + e_2^T \dot{e}_2 \\ =& e_1^T (-k_1 e_1 + e_2) + e_2^T \dot{e}_2 \\ =& -k_1 e_1^T e_1 + e_1^T e_2 + e_2^T (f + gu - \dot{r}_2) \\ & .\end{aligned}$$

Therefore, the stabilizing control input u that realizes $\dot{\mathcal{V}}_2 = -k_1 e_1^T e_1 - k_2 e_2^T e_2 < 0$ can be designed as

$$u \triangleq g^{-1}(-e_1 - k_2 e_2 + \dot{r}_2 - f)$$

where $k_2 \in \mathbb{R}_{>0}$ is a positive constant. Note that, the stabilizing control input u requires the knowledge of the system functions $f(x)$ and $g(x)$.

On the other hand, the bounded input bounded output (BIBO) stability defined in Theorem 2.1.1, will be used for the stability analysis in the later chapters.

Theorem 2.1.1. *(see [38, Theorem 1.9]) Let the closed-loop transfer function $H(x)$ be the exponentially stable and strictly proper. Then $y = H \star u \in L^\infty$, $\dot{y} \in L^\infty$, and y is uniformly continuous, if $u \in L^\infty$.*

In other word, $\|x\|$ is bounded, for a linear system $\dot{x} = Ax + Bu$ where A is Hurwitz matrix and $\|B\|_F$ and $\|u\|$ are bounded. Without loss of generality, the BIBO stability can be extended to matrix state $x \in \mathbb{R}^{n \times m}$ as well.

2.2 Mathematical Review

2.2.1 Matrix Algebra

In this thesis, we will use the following notations for matrix algebra.

- $x_{(i)}$ denotes the i-th element of vector $x \in \mathbb{R}^n$.

- $A_{(i,j)}$ denotes the element in the i-th row and j-th column of matrix $A \in \mathbb{R}^{n\times m}$.
- $\mathrm{row}_j(A)$ denotes the j-th row of matrix $A \in \mathbb{R}^{n\times m}$.
- $\lambda_{\min}(A)$ denotes the minimum eigenvalue of matrix $A \in \mathbb{R}^{n\times n}$.

For a matrix $A \in \mathbb{R}^{n\times m}$, the **vec** operator is defined as

$$\mathrm{vec}(A) \triangleq \begin{bmatrix}\mathrm{row}_1(A^T) & \mathrm{row}_2(A^T) & \cdots & \mathrm{row}_n(A^T)\end{bmatrix} \in \mathbb{R}^{nm}.$$

Moreover, for the brief expression of neural networks (NNs) and their gradient, **Kronecker product** will be used which is defined in Definition 2.2.1.

Definition 2.2.1 (see [39, Definition 7.1.2])**.** Using the **vec** operator, the **Kronecker product** of two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$ is defined as

$$A \otimes B \triangleq \begin{bmatrix} A_{(1,1)}B & A_{(1,2)}B & \cdots & A_{(1,m)}B \\ A_{(2,1)}B & A_{(2,2)}B & \cdots & A_{(2,m)}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{(n,1)}B & A_{(n,2)}B & \cdots & A_{(n,m)}B \end{bmatrix} \in \mathbb{R}^{np\times mq}.$$

The **Kronecker product** has the following proposition.

Proposition 2.2.1 (see [39, Proposition 7.1.9])**.** *For matrix $A \in \mathbb{R}^{n\times m}$ and vector $x \in \mathbb{R}^n$, we have the following property*

$$A^T x = vec(A^T x) = vec(x^T A) = (I_m \otimes x^T)vec(A).$$

The proof of Proposition 2.2.1 can be found in [39]. Using Proposition 2.2.1 the gradient with respect the vectorized A can be computed as

$$\frac{\partial(A^T x)}{\partial \mathrm{vec}(A)} = I_m \otimes x^T.$$

The reader is referred to [39, Chapter 7] for more details about the **Kronecker Product** and its properties.

2.2.2 Constrained Optimization Theory

The general formulation of the constrained optimization problem can be represented as

$$\underset{x}{\text{minimize}} \quad f(x), \qquad \text{subject to} \begin{cases} c_j(x) = 0, & \forall j \in \mathcal{E} \\ c_j(x) \leq 0, & \forall j \in \mathcal{I} \end{cases}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ denotes the objective function, $x \in \mathbb{R}^n$ denotes the optimization variables, and $\mathcal{E}$ and $\mathcal{I}$ are the set of equality and inequality constraints, respectively. The objective of the constrained optimization problem is to find the optimal point x^* that locally or globally minimizes the objective function $f(x)$, satisfying the constraints $c_j(x)$. Generally, the imposed constraints in an active set $\mathcal{A} \triangleq \mathcal{E} \cup \{j \in \mathcal{I} \mid c_j \geq 0\}$ are supposed to satisfy the Linear Independence Constraint Qualification (LICQ) which is defined in Definition 2.2.2.

Definition 2.2.2 (see [33, Definition 12.1])**.** If the gradients of the active constraints $\nabla c_j(x^*)$, $j \in \mathcal{A}$, are linearly independent, the set of constraints $\{c_j\}$ satisfies the *Linear Independence Constraint Qualification* (LICQ) at x^*.

Using the Lagrange multipliers λ_j for each constraint $c_j(x)$, the Lagrangian function is defined as

$$L(x, [\lambda_j]_{j \in \mathcal{E} \cup \mathcal{I}}) = f(x) + \sum_{j \in \mathcal{E}} \lambda_j c_j(x) + \sum_{j \in \mathcal{I}} \lambda_j c_j(x).$$

Then, the constrained optimization problem can be reformulated as the min-max problem as

$$\min_{x} \max_{[\lambda_j]_{j \in \mathcal{E} \cup \mathcal{I}}} L(x, [\lambda_j]_{j \in \mathcal{E} \cup \mathcal{I}}).$$

The conditions of optimality are defined by the first-order necessary condition and the second-order necessary and sufficient conditions. The first-order necessary condition which is often known as the *Karush-Kuhn-Tucker conditions* (KKT) is that the gradient of the Lagrangian function L at (x^*, λ^*) should be zero. The second-order necessary condition is that the Hessian matrix of L at (x^*, λ^*) should be positive semi-definite while the second-order sufficient condition is that the Hessian matrix should be positive definite (i.e. L is convex in the neighbor of the point (x^*, λ^*)). The constrained optimization problem typically attempts to find local solution (x^*, λ^*) which satisfy the first-order necessary condition of optimality, since there is no guarantee that L is convex function (i.e. for the global optimality second-order condition is required). The

KKT conditions for the constrained optimization problem is stated in [33] as following Theorem 2.2.1.

Theorem 2.2.1 (see [33, Theorem 12.1])**.** *Let x^* be a local solution of the constrained optimization problem. Then, there exists a Lagrange multiplier vector λ^* such that the following conditions are satisfied:*

$$\begin{aligned}
\nabla_x L(x^*, \lambda^*) &= 0 \\
c_j(x^*) &= 0, \quad \forall j \in \mathcal{E} \\
c_j(x^*) &\leq 0, \quad \forall j \in \mathcal{I} \\
\lambda_j^* &\geq 0, \quad \forall j \in \mathcal{I} \\
\lambda_j^* c_j(x^*) &= 0, \quad \forall j \in \mathcal{E} \cup \mathcal{I}.
\end{aligned}$$

For the details of the optimization theory, the reader can refer to [33] and [40].

2.2.3 Preservation of Convexity

Definition 2.2.3 (see [40, Chapter 2.1.4])**.** A set C is **convex** if the line segment between any two points in C lies entirely in C. That is, for all x, $y \in C$ and $\lambda \in [0, 1]$, we have

$$\lambda x + (1 - \lambda)y \in C.$$

Definition 2.2.4 (see [40, Chapter 3.1.1])**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if **dom** of f is a convex set and if for all x, $y \in$ **dom** f and $\lambda \in [0, 1]$, we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

The convexity is very useful property in optimization theory and controller design to find optimal control parameters, since the every local solution points are global solution point satisfying the second-order necessary and sufficient conditions [33, Theorem 2.5]. Furthermore, if the optimal point of the convex function $f(x)$ is the origin, the opposite direction of the gradient of f at x is the descent direction. In other word, the angle between the gradient at x and the vector x is positive as following Lemma 2.2.1.

Lemma 2.2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and x be a point in the domain of f. If the origin is the optimal point of the function f, then the angle between the gradient of f at x and the vector x is positive, implying that $\nabla f^T x > 0$.*

Proof. Let the origin be the isolated optimal point that minimizes the function f such that $f(x) > f(0)$. Then, the following inequality holds:

$$\begin{aligned}\nabla f^T(-x) =& \frac{d}{d\delta} f(x - \delta x)\bigg|_{\delta=0} \\ =& \lim_{\delta\to 0} \frac{f(x + \delta(0 - x) - f(x)}{\delta} \\ \leq& \lim_{\delta\to 0} \frac{\delta f(0) + (1-\delta) f(x) - f(x)}{\delta} \\ =& f(0) - f(x) < 0\end{aligned}.$$

It implies that the angle between the gradient of f at x and the vector x is positive. □

In [40, Chapter 2.3.2], the authors stated that the affine functions preserve the convexity of the function as follows:

> Recall that a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if it is a sum of a linear function and a constant, i.e. if it has the form $f(x) = Ax + b$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Suppose $S \in \mathbb{R}^n$ is convex and $f : R \in n \to \mathbb{R}^m$ is an affine function. Then the image of S under f,
>
> $$f(S) = \{f(x)|x \in S\},$$
>
> is convex.

This property will be utilized in the stability analysis of the controller in Chapter 4.

2.3 Deep Neural Networks

The capability of NNs to approximate functions is based on the approximation theory [41]. In other words, the NNs can approximate any sufficiently smooth function on a compact set with arbitrary accuracy according to the universal approximation theorem defined in Theorem 2.3.1.

Theorem 2.3.1 (see [13, 42])**.** *Let f be a sufficiently smooth function defined on a compact set $x \in \Omega \in \mathbb{R}^n$. Then, for any $\epsilon \in \mathbb{R}_{>0}$, there exists an ideal weight vector θ^* in a single hidden layer NN (SHLNN) with the sigmodal activation function $\Phi(x;\theta^*)$ that approximates f with ϵ-accuracy in $x \in \Omega$ such that* $\sup_{x\in\Omega} \|\Phi(x;\theta^*) - f(\cdot)\| = \epsilon < \infty$.

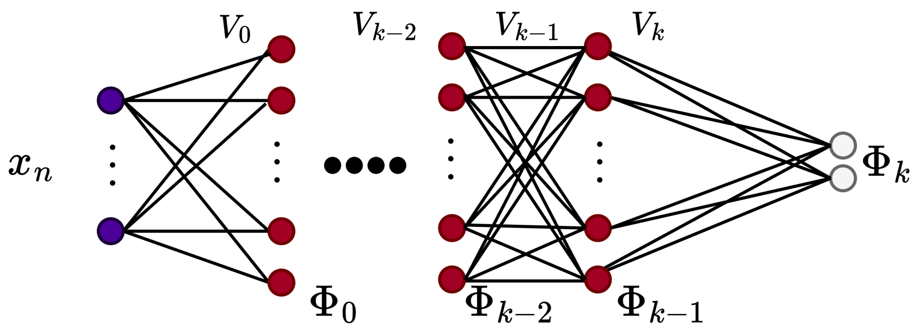


Figure 2.1: Architecture of the deep neural network (DNN).

Besides, the universal approximation property of deep NNs (DNNs) is also reported in [43]. Generally, the ideal vector is assumed to be constant and bounded such that $\|\theta^*\| \le \bar{\theta} < \infty$.

Mathematical Expression of Deep Neural Network

Most literature which utilizes the SHLNNs in the controller, have shown that the SHLNNs can approximate the uncertain system functions or control law with satisfactory performance index. However, the DNNs are exponentially more expressive than the SHLNNs to the same accuracy in terms of the total number of weights [29]. Therefore, the DNNs will be utilized in the controller in this thesis. Note that the SHLNN can be considered as the simplest architecture of the DNN with a single hidden layer.

In general, it was open problem to leverage the DNNs in controllers due to the nonlinearity and mathematically complex architecture of the DNNs. In [28], Omkar Sudhir Patil *et al.* proposed the novel DNN based neuro-adaptive control for control-affine nonlinear systems. The architecture of the DNN in the controller is described in Fig. 2.1 and is defined as

$$\Phi(x_n;\theta) \triangleq \underbrace{V_k^T\phi_k(\underbrace{V_{k-1}^T \cdots \phi_2(\underbrace{V_1^T\phi_1(\underbrace{V_0^T x_n}_{\Phi_0})}_{\Phi_1})\cdots}_{\Phi_{k-1}}))}_{\Phi_k} \tag{2.1}$$

where x_n denotes the NN input vector, $V_i \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$ 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 element-wise 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. For a better understanding, (2.1) also can be represented recursively as

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

where Φ_i denote each layer's output (i.e. the last layer's output is equal to the output of DNN such that $\Phi_k = \Phi(x_n; \theta)$).

One of the widely used activation functions for large DNNs is from the ReLU family [44], which effectively avoids the gradient vanishing problem during error backpropagation. The gradient vanishing problem occurs when the gradient of the activation function is close to zero, since the gradient of each layer is multiplied using chain rule to backpropagate the error to the inner layers (i.e. deeper NNs have high possibility of gradient vanishing). 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 are bounded such that $\|\phi_i(\cdot)\| < \infty$ and $\|\nabla\phi_i(\cdot)\|_F < \infty$. In this thesis, 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 $\|\nabla\sigma(\cdot)\| < 1$. Note that, the number of hidden layers should be limited around 5 to avoid the gradient vanishing issue, since the gradient vanishing problem is not addressed.

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 augmentation 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.

Gradient of Deep Neural Network

In the derivation of adaptation law, the gradient of the DNN with respect to the weights is required. The gradient of $\Phi(x_n;\theta)$ with respect to θ is defined as

$$\frac{\partial \Phi}{\partial \theta} = \begin{bmatrix} \frac{\partial \Phi}{\partial \theta_k} & \frac{\partial \Phi}{\partial \theta_{k-1}} & \cdots & \frac{\partial \Phi}{\partial \theta_0} \end{bmatrix} \in \mathbb{R}^{n\times \Xi} \tag{2.2}$$

where

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

where $\phi_i \triangleq \phi_i(\Phi_{i-1})$ and $\phi_i' \triangleq \partial\phi_i/\partial\Phi_{i-1}$. The gradient can be obtained using the chain rule and Proposition 2.2.1.

Chapter 3
CoNAC for Uncertain Euler-Lagrange Systems Under Weight Constraints

3.1 Introduction

In this chapter, a novel constrained optimization-based neuro-adaptive control (Co NAC) is presented for uncertain Euler-Lagrange systems with a weight norm constraint. As presented in Section 1.2, one of the common issues in neuro-adaptive control (NAC) is that the boundedness of neural network's (NN) weights is not guaranteed. Since the amplitude of control input of NAC is dependent on the weights, the unbounded weights may lead to instability and severe safety issues. The satisfaction of the boundedness of the weights are reformulated into weight norm constraints, which are then incorporated into the CoNAC.

3.2 Problem Formulation

Consider an uncertain Euler-Lagrange system modeled as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(q) = \tau \tag{3.1}$$

where $q \in \mathbb{R}^n$ and $\tau \in \mathbb{R}^n$ denotes the generalized coordinate and the control input, respectively; $M(q) \in \mathbb{R}^{n\times n}$, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$, and $G(q) \in \mathbb{R}^n$ denote the unknown system function matrices; and $F(q) \in \mathbb{R}^n$ denotes the external force.

Using the user-designed matrices $M_0 > 0, C_0$ and G_0, (3.1) can be represented as

$$M_0\ddot{q} + C_0\dot{q} + G_0 = \tau + f(q,\dot{q},\ddot{q}) \tag{3.2}$$

where $f(q,\dot{q},\ddot{q}) \triangleq -(M-M_0)\ddot{q} - (C-C_0)(q,\dot{q})\dot{q} - (G-G_0) - F(q)$ denotes the residual unknown term.

As presented in Section 1.2, the parameter drift (i.e. the weights of NNs can diverse) may occur due to the lumped disturbance term. Hence, the objective of the control design is to make q track the continuously differentiable desired trajectory $q_d(t) : \mathbb{R} \to \mathbb{R}^n$ under the unknown terms f while ensuring boundedness of weights of NN in the controller.

3.3 Exiting Works for Boundedness of Weights

Most studies modify their adaptation laws to ensure the boundedness of the weights.

3.3.1 Projection Operator

In [27,28,30], the projection operator is utilized to prevent the weight divergence, by projecting the adaptation direction on some convex set of the weights. The projection operator is defined in [37, Appendix E, eq. (E.4)] and represented as

$$\mathrm{Proj}_{\Omega}(y) = \begin{cases} \Gamma y & \begin{array}{l}\text{if } x \in \Omega \text{ or if} \\ x \in \delta\Omega \text{ and } \nabla c^T \Gamma y \geq 0\end{array} \\ \Gamma y - \Gamma \frac{\nabla c \nabla c^T}{\nabla c^T \Gamma \nabla c} \Gamma y & \text{otherwise} \end{cases} \tag{3.3}$$

where $\Gamma = \Gamma^T > 0$ is adaptation gain matrix and $y \in \mathbb{R}^n$ denotes the update direction of optimization variable $x \in \mathbb{R}^n$. Moreover, $\Omega \triangleq \{x \mid c(x) \leq 0\}$ denotes a convex set defined by $c(\cdot)$ is a convex function and $\delta\Omega$ denotes a boundary of Ω. The convex function $c(\cdot)$ is typically selected as $(1/2)x^T x \leq \bar{x}^2$ where $\bar{x} \in \mathbb{R}_{>0}$ is a maximum norm of x. However, in the literature, the projection operator is only applied to theoretically guarantee the boundedness of the weights, by selecting the convex set as large as possible. This is because, the authors attempts to estimate the globally ideal weights whose magnitude is unknown.

3.3.2 σ and ϵ-modifications

In adaptive control theory, the σ-modification [26] and the ϵ-modification [17,19] are widely used to regulate the magnitude of the weights by adding a stabilizing function in the adaptation law as follows:

$$y_{\sigma} = \Gamma(y + \lambda x), \quad y_{\epsilon} = \Gamma(y + \rho \|e\| x)$$

where $y_i,\ i \in [\sigma, \epsilon]$ denote adaptation laws of σ-modification and ϵ-modification, respectively, e denotes tracking error, and λ and ρ denote parameters of σ-modification and ϵ-modification, respectively. These methods make the invariance set of the estimation error of the weights over time. The existing methods have shown their effectiveness in ensuring the boundedness of the weights via numerical simulations. However, the weights are biased to the origin by the stabilizing function, which may degrade the performance of the controller. This means that there is a trade-off between the boundedness of the weights and the optimality of the weights. Moreover, they lack theoretical analysis regarding the optimality of the adapted weights.

Interestingly, similar approaches that regulate the magnitude of the weights have also been introduced in the deep learning literature. One of the approaches is L_2-regularization, which adds the squared magnitudes of the weights to the objective function [45,46]. Then, the adaptation process attempts to reduce not only the original objective function, but also the magnitude of the weights. By regulating the magnitude of the weights, the stability of the adaptation process can be enhanced, and overfitting can be prevented. However, L_2-regularization also involves same limitations as σ and ϵ-modifications.

3.4 CoNAC with Weight Norm Constraint

Without loss of generality a single hidden layer neural network (SHLNN) is utilized in this chapter for better intuition and simplicity. Note that SHLNN is the simplest case of DNN presented in Section 2.3. The architecture of the CoNAC is illustrated in Fig. 3.1, consisting of: a reference generator, the SHLNN that functions as NAC, and a weight optimizer for the SHLNN. The reference generator is designed based on backstepping control (BSC), presented in previous Section 2.1 , to generate a tracking reference for both q and $\dot{q}$.

3.4.1 Control Law Development

The system dynamics (3.2) can be represented as

$$\begin{aligned} \dot{q} &= z, \\ \dot{z} &= -M_0^{-1}C_0 z - M_0^{-1}G_0 + M_0^{-1}h(\tau) + M_0^{-1}f, \end{aligned} \tag{3.4}$$

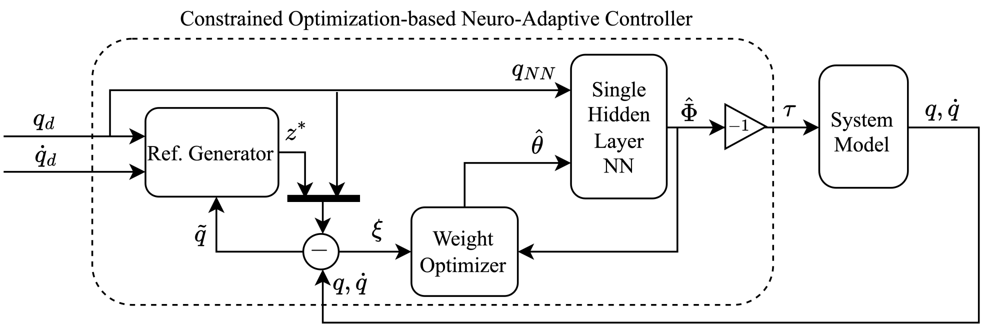


Figure 3.1: Architecture of the constrained optimization-based neuro-adaptive controller (CoNAC).

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

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

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

$$\tilde{z} \triangleq z - z^* = z - (-k_q\tilde{q} + \dot{q}_d). \tag{3.5}$$

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

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

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:

$$\tau^* \triangleq -M_0 \cdot (k_z\tilde{z}) + (-M_0\tilde{q} + C_0 z + G_0 - f + M_0\dot{z}^*). \tag{3.6}$$

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.

As introduced in Section 2.3, the SHLNN which is simple version of DNN is represented as

$$\Phi(q_{NN};\theta) \triangleq V_1^T\phi(V_0^T q_{NN})$$

where $q_{NN} \in \mathbb{R}^{l_0+1}$ denotes the NN input vector, $V_i \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$, $i \in [0,1]$ denotes the weight matrix of i^{th} layer and $\phi : \mathbb{R}^{l_1} \to \mathbb{R}^{l_1+1}$ denotes the activation function layer. The element-wise activation function layer consists of nonlinear function $\sigma(\cdot)$ and augmented 1 to combine the bias term in weight matrix (i.e. $\phi(x) = [\sigma(x_{(1)}), \cdots, \sigma(x_{(l_1)}), 1]^T$). For further simplicity, let $\theta \triangleq [\theta_1^T, \theta_0^T]^T \in \mathbb{R}^{\Xi}$ denote the total weight vector, where $\theta_i \triangleq \text{vec}(V_i) \in \mathbb{R}^{\Xi_i}$ denote the vectorized weights. $\Xi_i = (l_i+1)\cdot l_{i+1}$ and $\Xi = \Xi_0 + \Xi_1$ denote the number of each layer and total weights, respectively.

Using this SHLNN, the desired controller τ^* can be approximated through ideal weight vector θ^* for a compact subset $\Omega_{NN} \in \mathbb{R}^{l_0+1}$ to ϵ-accuracy according to Theorem 2.3.1 such that $\sup_{q_{NN}\in\Omega_{NN}} \|\Phi(q_{NN};\theta^*) - \tau^*\| = \epsilon < \infty$. The ideal weight vector θ^* is typically assumed to be bounded. In this thesis, θ^* is defined as a local optimal point, rather than a global optimal point. Then using the estimated weight vector $\hat{\theta} = [\hat{\theta}_1^T, \hat{\theta}_0^T]^T$ of $\theta^* = [\theta_1^{*T}, \theta_0^{*T}]^T$, the desired controller $\tau^* \approx -\Phi(q_{NN};\theta^*) - \epsilon$ can be approximated as follows:

$$\tau \triangleq -\Phi(q_{NN};\hat{\theta}). \tag{3.7}$$

For further sections, let $\Phi^* \triangleq \Phi(q_{NN};\theta^*)$ and $\phi^* \triangleq \phi(V_0^{*T}q_{NN})$, and $\hat{\Phi} \triangleq \Phi(q_{NN};\hat{\theta})$, $\hat{\phi} \triangleq \phi(\hat{V}_0^T q_{NN})$ and $\hat{\phi}' = \partial\hat{\phi}/\partial(\hat{V}_0^T q_{NN})$.

Using (3.4), (3.5), (3.6), and (3.7), the error dynamics can be derived as

$$\begin{aligned}\dot{\tilde{q}} &= -k_q\tilde{q} + \tilde{z}\\ \dot{\tilde{z}} &= -\tilde{q} - k_z\tilde{z} + M_0^{-1}(\Phi^* - \hat{\Phi} + \epsilon).\end{aligned} \tag{3.8}$$

The error dynamics (3.8) can be represented as a first-order system:

$$\dot{\xi} = A_\xi\xi + B_\xi(\Phi^* - \hat{\Phi} + \epsilon) \tag{3.9}$$

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

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

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

3.4.2 Weight Adaptation Laws

Weight Optimizer Design

Consider a positive definite objective function defined as

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

where $W = W^T > 0$ is a weighting matrix. The weight norm constraints $c_j,\ j \in \mathcal{I}$ presented in following Section 3.5, are imposed during the weight adaptation process, where $\mathcal{I}$ denotes the set of the imposed inequality constraints. The corresponding constrained optimization problem is formulated as

$$\underset{\hat{\theta}}{\text{minimize}} \quad J(\xi; \hat{\theta}), \qquad \text{subject to } c_j(\hat{\theta}) \leq 0,\ \forall j \in \mathcal{I}. \tag{3.10}$$

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

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

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

The adaptation laws for $\hat{\theta}$ and $[\lambda]_{j\in\mathcal{A}}$ are derived to solve the dual problem of (3.10) (i.e. $\min_{\hat{\theta}} \max_{[\lambda]_{j\in\mathcal{A}}} L(\xi, \hat{\theta}, [\lambda]_{j\in\mathcal{A}})$), as follows:

$$\dot{\hat{\theta}} = -\alpha \frac{\partial L}{\partial \hat{\theta}} = -\alpha \bigg(\frac{\partial J}{\partial \hat{\theta}} + \sum_{j\in\mathcal{A}} \lambda_j \frac{\partial c_j}{\partial \hat{\theta}} \bigg), \tag{3.11a}$$

$$\dot{\lambda}_j = \beta_j \frac{\partial L}{\partial \lambda_j} = \beta_j c_j, \qquad \forall j \in \mathcal{A}, \tag{3.11b}$$

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

where $\alpha \in \mathbb{R}_{>0}$ denotes the adaptation gain (also known as 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. When a constraint c_j becomes active (i.e. violated), the corresponding Lagrange multiplier λ_j increases from zero to address the violation by adjusting the weights' adaptation direction $\dot{\hat{\theta}}$. 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 augmented Lagrangian method (ALM) in [33], 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 defined in Theorem 2.2.1, are satisfied, i.e. $\partial L/\partial\hat{\theta} = 0$, $c_j \leq 0$, $\lambda_j \geq 0$, and $\lambda_j c_j = 0$. In other words, the proposed optimizer updates the SHLNN 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.

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 (3.11a). 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 [47] is employed to calculate the exact gradient of the objective function.

By partially differentiating (3.9), the sensitivity equation of ξ with respect to $\hat{\theta}$ is first obtained as

$$\dot{\eta} = A_\xi \eta - B_\xi \frac{\partial \hat{\Phi}}{\partial \hat{\theta}} \tag{3.12}$$

Algorithm 1: Weight optimizer implementation.

```
  input  : $\xi$, $\hat{\theta}$, $\lambda_j$, $\eta$
  output: $\hat{\theta}$, $\lambda_j$, $\eta$
1 Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{j\}$ for all $c_j \geq 0$;
2 Determine update matrix $\dot{\eta}$ using (3.12);
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 (3.11a), (3.11b);
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 $\lambda_j = 0$;
```

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

$$\frac{\partial J}{\partial \hat{\theta}} = \frac{\partial \xi}{\partial \hat{\theta}}^T W \xi = \eta^T W \xi \in \mathbb{R}^{\Xi}. \tag{3.13}$$

Equations (3.12) and (3.13) can be decomposed for each layer as

$$\begin{aligned}\dot{\eta} =& \begin{bmatrix} \eta_1 & \eta_0 \end{bmatrix}' \\ =& A_\xi \begin{bmatrix} \eta_1 & \eta_0 \end{bmatrix} - B_\xi \begin{bmatrix} (I_{l_2} \otimes \hat{\phi}^T) & \hat{V}_1^T \hat{\phi}' (I_{l_1} \otimes {q_{NN}}^T) \end{bmatrix}.\end{aligned}$$

and

$$\frac{\partial J}{\partial \hat{\theta}} = \begin{bmatrix} \partial J / \partial \hat{\theta}_1 \\ \partial J / \partial \hat{\theta}_0 \end{bmatrix} = \begin{bmatrix} \eta_1^T \\ \eta_0^T \end{bmatrix} W \xi$$

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 (3.13), with the value of η obtained by simulating the sensitivity equation (3.12).

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 [33, Chap. 3 eq. (3.4)] to ensure that the objective function decreases.

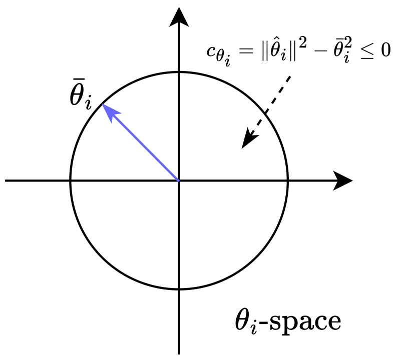

$c_{\theta_i} = \|\hat{\theta}_i\|^2 - \bar{\theta}_i^2 \leq 0$
$\bar{\theta}_i$
θ_i-space

Figure 3.2: Weight norm constraints.

3.5 Weight Norm Constraint

The satisfaction of the weights' boundedness is reformulated as a weight norm constraints as shown in Fig. 3.2. The weight norm constraints are represented as follows:

$$c_{\theta_1}(\hat{\theta}) \triangleq \|\hat{\theta}_1\|^2 - \bar{\theta}_1^2 \leq 0,$$
$$c_{\theta_0}(\hat{\theta}) \triangleq \|\hat{\theta}_0\|^2 - \bar{\theta}_0^2 \leq 0,$$

where $\bar{\theta}_i^2 \in \mathbb{R}_{>0}$ denotes the predefined weight norm bound for each layer $i \in [0, 1]$.

The gradient of the constraints can be obtained easily as follows:

$$\frac{\partial c_{\theta_0}}{\partial \hat{\theta}} = \begin{bmatrix} 0_{\Xi_1 \times 1} \\ 2\hat{\theta}_0 \end{bmatrix}, \quad \frac{\partial c_{\theta_1}}{\partial \hat{\theta}} = \begin{bmatrix} 2\hat{\theta}_1 \\ 0_{\Xi_0 \times 1} \end{bmatrix}.$$

3.6 Stability Analysis

The following theorem proves the boundedness of the tracking error and the weight estimation of the weights.

Theorem 3.6.1. *For the dynamical system in* (3.1)*, the proposed controller* (3.7) *and the adaptation law* (3.11) *ensure the boundedness of the tracking error* ξ *and the weight estimation* $\hat{\theta}$*, provided that control gains* k_q *and* k_z *satisfy* (3.15)*.*

Proof. The boundedness will be proved from the last layer to the first layer.

Step 1: Boundedness of $\hat{\theta}_1, \eta_1, \xi$

For convenience, assume that all constraints are in the active set without loss of generality. If the constraints are not in the active set, the boundedness cannot be guaranteed, but weights will be adapted to reduce the objective function until the constraints are violated.

The dynamics of ξ can be represented as

$$\dot{\xi} = A_\xi \xi + B_\xi(-\hat{V}_1^T \hat{\phi} + w(t))$$

where $w(t) \triangleq V_1^{*T}\phi^* + \epsilon$ is a lumped residual term, which is bounded as $\|w(t)\| \leq \bar{w} < 0$. On the other hand, the dynamics of η_1 and $\hat{\theta}_1$ are represented as

$$\begin{aligned}\dot{\eta}_1 =& A_\xi \eta_1 - B_\xi(I_{l_2} \otimes \hat{\phi}^T) \\ \dot{\hat{\theta}}_1 =& -\alpha(\eta_1^T W \xi + 2\lambda_{\theta_1}\hat{\theta}_1).\end{aligned}$$

According to Theorem 2.1.1, the boundedness of η_1 can be obtained, since A_ξ is stable and the residual term $-B_\xi(I_{l_2} \otimes \hat{\phi}^T)$ is bounded.

Define the Lyapunov function $\mathcal{V}_1 = (1/2)\xi^T P \xi + (1/2\alpha)\tilde{\theta}_1^T \tilde{\theta}_1$, with the Lyapunov equation $A_\xi^T P + P A_\xi = -Q$, where $A_\xi < 0, P = P^T > 0$, and $Q > 0$. Using Proposition 2.2.1 (i.e. $\hat{V}_1^T \hat{\phi} = \text{vec}(\hat{V}_1^T \hat{\phi}) = \text{vec}(\hat{\phi}^T \hat{V}_1) = (I_{l_2} \otimes \hat{\theta}^T)\text{vec}(\hat{V}_1) = (I_{l_2} \otimes \hat{\phi}^T)\hat{\theta}_1)$, the time derivative of $\mathcal{V}_1$ is

$$\begin{aligned}\dot{\mathcal{V}}_1 =& \frac{1}{2}\xi^T(A_\xi^T P + P A_\xi)\xi + \xi^T P(-B_\xi \hat{V}_1^T \hat{\phi} + B_\xi w(t)) + \hat{\theta}_1^T\Big(- \eta_1^T W \xi - 2\lambda_{\theta_1}\hat{\theta}_1\Big) \\ =& -\frac{1}{2}\xi^T Q \xi - \xi^T P B_\xi (I_{l_2} \otimes \hat{\phi}^T)\hat{\theta}_1 + \xi^T \Delta - \hat{\theta}_1^T \eta_1^T W \xi - 2\lambda_{\theta_1}\hat{\theta}_1^T \hat{\theta}_1 \\ \leq& -(1/2)\lambda_{\min}(Q)\|\xi\|^2 + \bar{\Delta}\|\xi\| + \bar{M}\|\xi\|\|\hat{\theta}_1\| - 2\lambda_{\theta_1}\|\hat{\theta}_1\|^2 \\ \leq& \Big(- \frac{\lambda_{\min}(Q)}{2} + \frac{\bar{M}}{2}\Big)\|\xi\|^2 + \bar{\Delta}\|\xi\| + \Big(- 2\lambda_{\theta_1} + \frac{\bar{M}}{2}\Big)\|\tilde{\theta}_1\|^2\end{aligned} \tag{3.14}$$

where $\Delta \triangleq P B_\xi w(t)$ and $M \triangleq -P B_\xi (I_{l_2} \otimes \hat{\phi}^T) + W\eta_1$ which are bounded such that $\|\Delta\| \leq \bar{\Delta} < \infty$ and $\|M\|_F \leq \bar{M} < \infty$, respectively.

By defining $P = I_n$, the eigenvalues of $Q = -A_\xi^T - A_\xi$ are $2k_q$ and $2k_z$, since A_ξ is a skew-symmetric matrix except for the diagonal entries. According to (3.14), if k_q and k_z are provided that

$$\min(k_q, k_z) > \bar{M}/2, \tag{3.15}$$

and if λ_{θ_1} is increased sufficiently large such that $2\lambda_{\theta_1} > \bar{M}/2$, due to the violation of the c_{θ_1}, the tracking error is bounded in

$$\Theta_\xi = \left\{ \xi \;\middle|\; \|\xi\| \leq \frac{\bar{\Delta}}{\lambda_{\min}(Q) - \bar{M}/2} \right\},$$

and the weight estimation θ_1^* is bounded in

$$\Theta_{\hat{\theta}_1} = \{\hat{\theta} \mid \|\hat{\theta}\| \leq \bar{\theta}_1\}.$$

The Lagrange multiplier λ_{θ_1} is also bounded, since λ_{θ_1} update halts once $\hat{\theta}_1$ approaches into the compact set $\Theta_{\hat{\theta}_1}$, satisfying the constraint c_{θ_1}.

Step 2: Boundedness of $\hat{\theta}_0, \eta_0$

The dynamics of η_0 and $\hat{\theta}_0$ are represented as

$$\begin{aligned} \dot{\eta}_0 =& A_\xi \eta_0 - B_\xi \hat{V}_1^T \hat{\phi}'(I_1 \otimes q_{NN}{}^T) \\ \dot{\hat{\theta}}_0 =& -\alpha\left(\eta_0^T W \xi + 2\lambda_{\theta_0}\hat{\theta}_0\right). \end{aligned}$$

Also, according to Theorem 2.1.1, η_0 is bounded since A_ξ is a stable matrix and $-B_\xi \hat{V}_1^T \hat{\phi}'(I_1 \otimes q_{NN}{}^T)$ is bounded. To obtain the invariance set of $\hat{\theta}_0$, taking the time-derivative of the Lyapunov function $\mathcal{V}_0 = (1/2\alpha)\hat{\theta}_0^T\hat{\theta}_0$ yields:

$$\begin{aligned} \dot{\mathcal{V}}_0 =& \hat{\theta}_0^T(-\eta_0 W\xi - 2\lambda_{\theta_0}\hat{\theta}_0) \\ \leq& \|\hat{\theta}_0\|\|\eta_0 W\xi\| - 2\lambda_{\theta_0}\hat{\theta}_0^T\hat{\theta}_0 \\ \leq& -2\lambda_{\theta_0}\|\hat{\theta}_0\|^2 + \|\eta_0 W\xi\|\|\hat{\theta}_0\|. \end{aligned}$$

Then, the invariance set can be represented as

$$\Theta_{\hat{\theta}_0} = \{\hat{\theta}_0 \mid \|\hat{\theta}_0\| \leq \|\eta_0 W\xi\|/\lambda_{\theta_0}\}.$$

If λ_{θ_0} is generated sufficiently large due to the violation of c_{θ_0}, the invariance set $\Theta_{\hat{\theta}_0}$ converges to $\{\hat{\theta}_0 \mid \|\hat{\theta}_0\| \leq \bar{\theta}_0\}$, once the constraint c_{θ_0} is satisfied. Therefore, the Lagrange multiplier λ_{θ_0} is also bounded.

$\square$

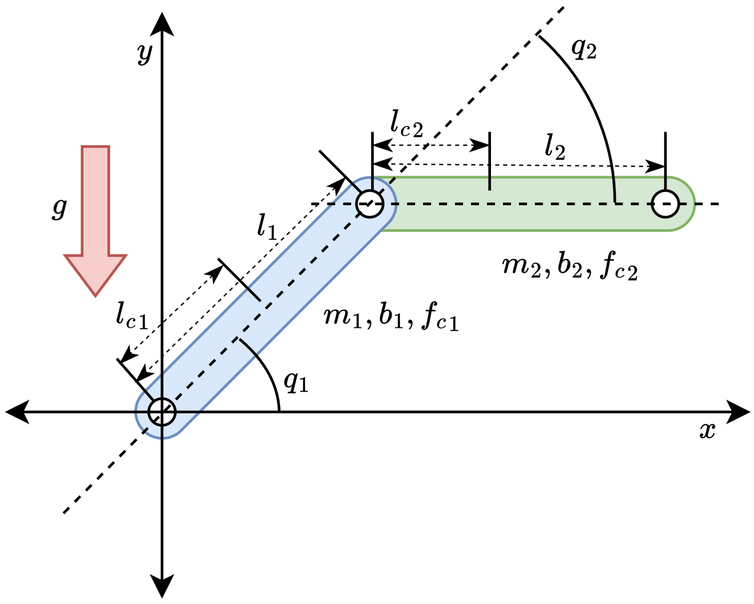


Figure 3.3: Two-link manipulator model.

Remark 3.6.1. In the constrained optimization method, the corresponding method of L_2-regularization method is the quadratic penalty method, which replaces the constrained optimization problem into an unconstrained optimization problem by adding the penalty term $(1/2\mu)\sum_{r\in\mathcal{A}} c_r^2$ in the objective function. The penalty parameter $\mu \in \mathbb{R}_{>0}$ usually decreases over implementation for the convergence of the optimization process. However, the decreased penalty term μ may alter the original objective function as the penalty term dominates the objective function. Therefore, L_2-regularization inherently has the analogous drawback of the quadratic penalty method in the selection of the regularization coefficient λ.

3.7 Simulation Validation

3.7.1 Setup

The two-link manipulator model in [48] is employed for the simulation demonstration. In the system, the parameters $q_p, q_{d_p}, \tau_p, m_p, l_p, l_{cp}, b_p$, and f_{cp} 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 values of the system parameters

Table 3.1: 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)
θ_1, b_2	Viscous coefficient	2.288 (Nms)	0.172 (Nms)
f_{c1}, f_{c2}	Friction coefficient	7.17 (Nm)	1.734 (Nm)

are given in Table 3.1. The reference signal of the $q = [q_1, q_2]^T$ is defined as follows:

$$q_d = \begin{bmatrix} q_{d1} \\ q_{d2} \end{bmatrix} = \begin{bmatrix} +\cos(\pi/2t) + 1 \\ -\cos(\pi/2t) - 1 \end{bmatrix}.$$

For the comparative study, three controllers were selected: the neuro-adaptive controller with L_2-regularization (NAC-L2) and with ϵ-modification (NAC-eMod), and the proposed controller with constrained optimization (CoNAC). The performances of the selected controllers are compared based on the tracking performances and the dependencies of the parameters λ, ρ, and β_j of NAC-L2, NAC-eMod, and CoNAC, respectively. The square root of integrated squared error (ISE) (i.e. $\sqrt{\int_0^T \|\xi\|^2 \, dt}$, where T denotes a simulation termination time) is utilized to evaluate the tracking performances. The parameter dependencies of the controllers were examined via various values of the parameters. The values ranged from 0.001 to 1 across 10 samples.

The control laws of all three controllers were the same as defined in (3.7). The adaptation law of NAC-L2 is derived by adding the squared weight term $(1/2)\lambda\hat{\theta}^T\hat{\theta}$ to the objective function such that $J_{L_2} = J + (1/2)\lambda\hat{\theta}^T\hat{\theta}$, where $\lambda \in \mathbb{R}_{>0}$ denotes the L_2-coefficient. The adaptation law obtained via the gradient descent method is subsequently adjusted by adding stabilizing term $-\alpha\lambda\hat{\theta}$ as follows:

$$\dot{\hat{\theta}} = \frac{\partial J_{L_2}}{\partial \hat{\theta}} = -\alpha\left(\frac{\partial J}{\partial \hat{\theta}} + \lambda\hat{\theta}\right).$$

Note that this adaptation law derived based on L_2-regularization method in deep learning is inherently the same as σ-modification in the adaptive control theory which adds the term $-\alpha\sigma\hat{\theta}$, where $\sigma \in \mathbb{R}_{>0}$. For NAC-eMod, Similar to σ-modification, the stabi-

Table 3.2: Quantitative comparison of square root of tracking ISE.

	NAC-L2	**NAC-eMod**	**CoNAC** (proposed)
Maximum	11.1753e-3	0.5603e-3	0.3439e-3
Upper quartile	1.7284e-3	0.5566e-3	0.3261e-3
Median	0.5898e-3	0.5519e-3	0.3240e-3
Lower quartile	0.5533e-3	0.5470e-3	0.3238e-3
Minimum	0.5434e-3	0.5434e-3	0.3235e-3

lizing function $-\alpha\rho\|\tilde{z}\|\hat{\theta}$ is added to the adaptation law as follows:

$$\dot{\hat{\theta}} = -\alpha\left(\frac{\partial J}{\partial\hat{\theta}} + \rho\|\tilde{z}\|\hat{\theta}\right)$$

where $\rho \in \mathbb{R}_{>0}$ denotes the ϵ-modification coefficient. By $\|\tilde{z}\|$, the stabilizing function proportionally increases as the tracking error $\tilde{z}$ increases. Therefore, the adaptation attempts to reduce the tracking error mainly without the effect of the stabilizing function, if the tracking error is sufficiently regulated. The adaptation law of CoNAC is presented in (3.11). Owing to the stabilizing functions, the weights of NAC-L2 and NAC-eMod are biased, since the stabilizing functions drive the weights toward the origin.

All controllers had the same control parameters except their crucial parameters (i.e. λ, ρ and β_j) as $k_q = 1.1$, $k_z = 10$, $M_0 = I_2$ and $W = \text{diag}([5, 1, 15, 15])$. The parameters of the NNs were set $l_0 = 2$, $l_1 = 16$, $l_2 = 2$, and $\alpha = 10^3$ and the same random seed was applied for the weight initialization. The NN input vector was set to the desired trajectory q_d, with the augmented 1 to incorporate the bias term in the weight matrix, such that $q_{NN} = [q_d^T, 1]^T$. For CoNAC, the parameters of the weight norm constraints were set as $\bar{\theta}_0 = 10$ and $\bar{\theta}_1 = 20$. The sampling time of the simulation and the simulation termination time were set to $T_s = 10^{-4}$ and $T = 10$, respectively.

3.7.2 Results

As shown in Fig. 3.4, the maximum square root of tracking ISE of CoNAC is smaller than the minimum square root of tracking ISEs of NAC-L2 and NAC-eMod for all variations of the parameters. This is because NAC-L2 and NAC-eMod bias the weights to the origin, due to the presence of the stabilizing functions. A quantitative

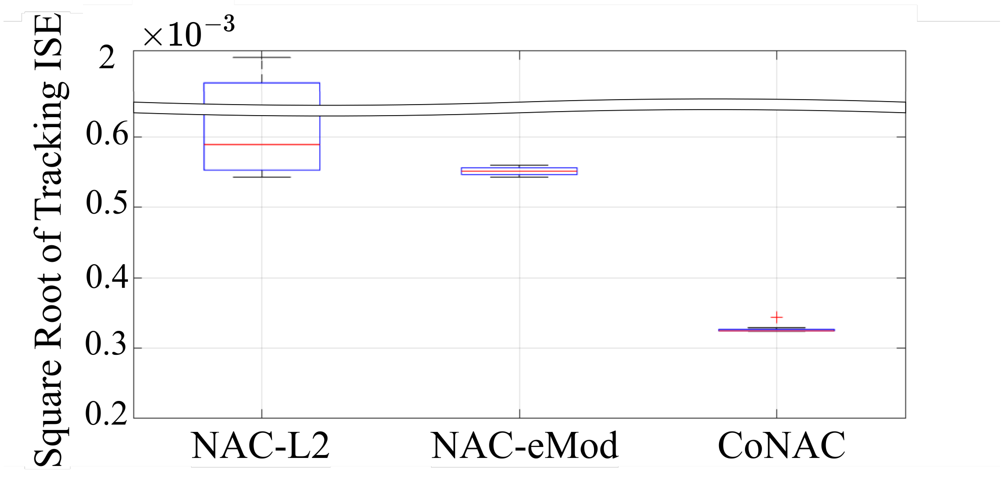


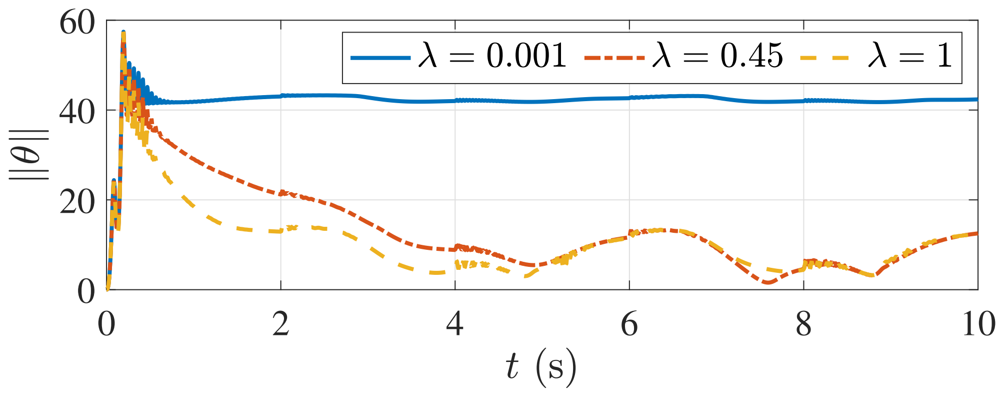
Figure 3.4: Box-and-Whisker plot of the square root of the tracking ISEs of NAC-L2, NAC-eMod and CoNAC across various parameter values.

comparison of the square root of tracking ISE is provided in Table 3.2.

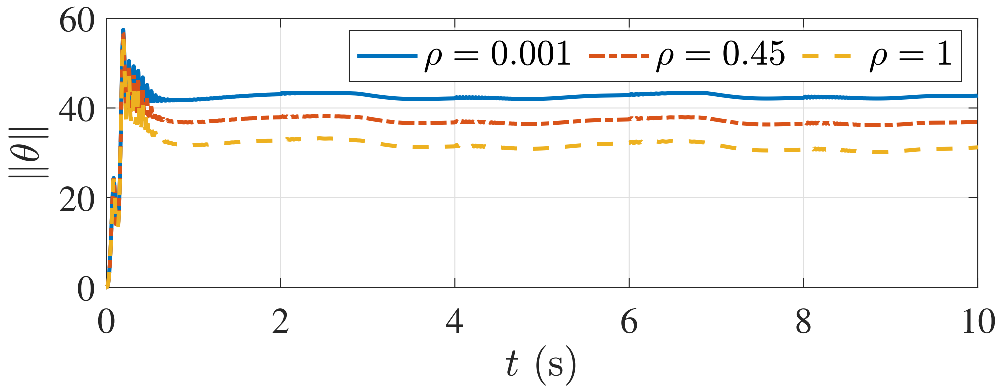
For the detailed analysis, three values of the parameters (i.e. $\lambda, \rho, \beta_j \in [0.001, 0.45, 1]$) were selected as described in Fig. 3.5 and Fig. 3.6. As shown in Fig. 3.5a, increasing λ reduces the weight norm of NAC-L2 by the stabilizing function $-\alpha\lambda\hat{\theta}$. Moreover, the high dependency of NAC-L2 to the L_2-regularization coefficient λ also can be observed. Since the weight norm is decreased, NAC-L2 cannot generate sufficient control inputs, resulting in a larger square root of tracking ISE, as shown in Fig. 3.6a.

On the other hand, NAC-eMod exhibits lower dependency on the ϵ-modification coefficient ρ as shown in Fig. 3.5b and Fig. 3.6b. This is because stabilizing function $-\alpha\rho\|\tilde{z}\|\hat{\theta}$ can be decreased once the tracking error $\tilde{z}$ is sufficiently regulated. However, the bias of the weights to the origin still exists as described in Fig. 3.5b (i.e. smaller weight norms are observed as ρ is increased.). Therefore, similar to NAC-L2, the biased weights produce insufficient control input, resulting in a relatively larger square root of tracking ISE than that of CoNAC, as described in Table 3.2.

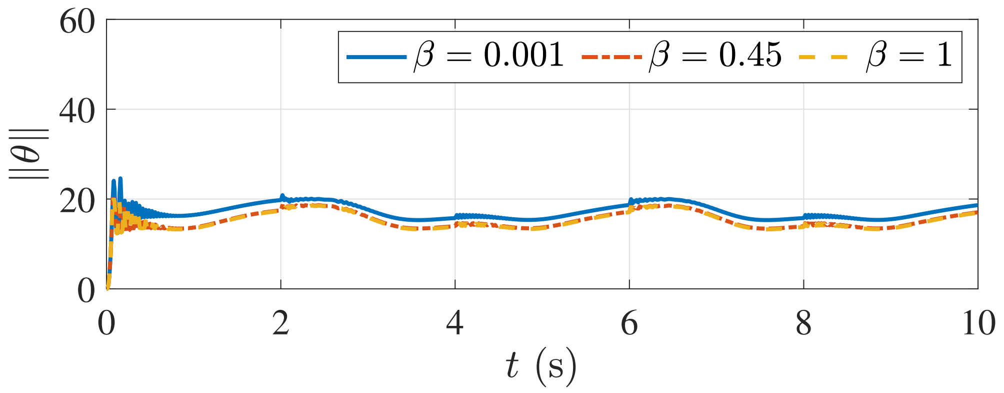
Finally, the weight norm of CoNAC is smaller than those of NAC-L2 and NAC-eMod as shown in Fig. 3.5c with better tracking performances. Even if the large β_j is provided, CoNAC can adjust the adaptation direction to satisfy the weight norm constraints faster, according to (3.11b). Therefore, the lowest dependency on the update rate β_j is observed in CoNAC as shown in Fig. 3.5c and Fig. 3.6c. Note that β_j of CoNAC is the update rate for Lagrange multipliers while λ and ρ are the coefficients of the stabilizing function that generates the biases of the weights. However, considering the



(a) NAC-L2



(b) NAC-eMod

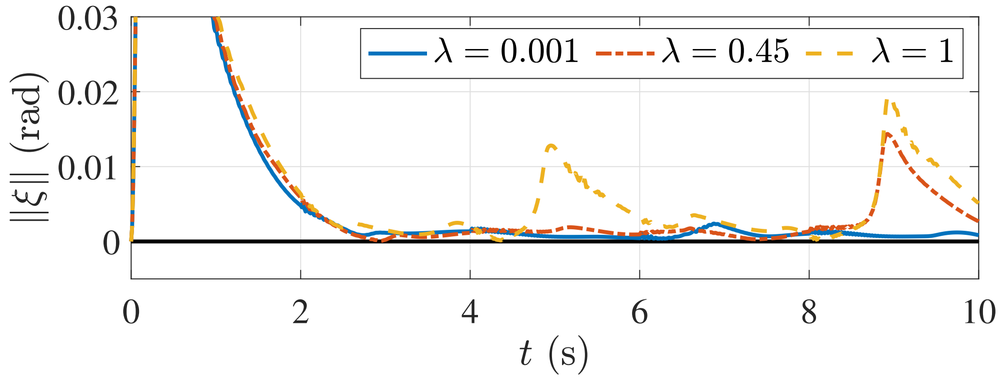


(c) CoNAC

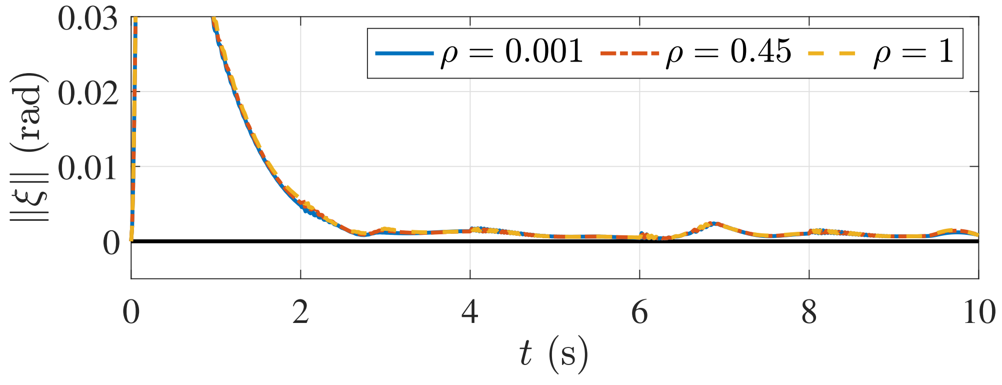
Figure 3.5: Weight norms of NAC-L2, NAC-eMod and CoNAC.

implementation using a digital computer, excessively large β_j should be avoided.

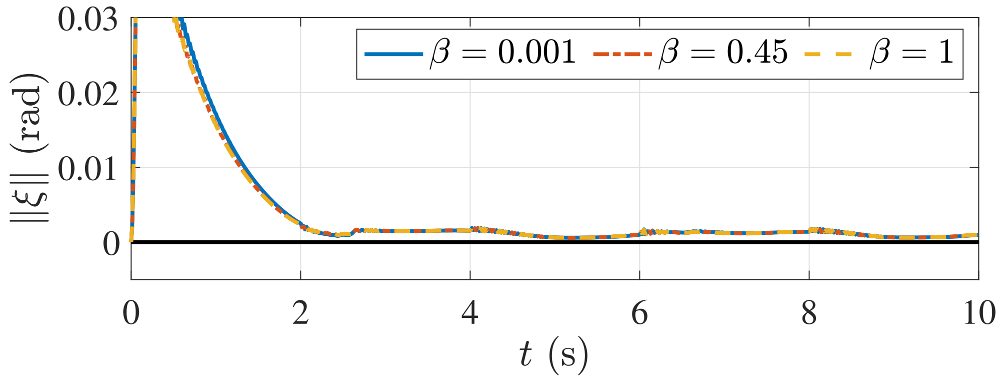
The details of the satisfaction of the weight norm constraints are shown in Fig. 3.7 for CoNAC with $\beta_j = 0.001$. As the weight norms of each layer reach the constraint boundary, the corresponding Lagrange multipliers are generated. Using the Lagrange multipliers, the adaptation direction is adjusted toward the constraint satisfactory



(a) NAC-L2



(b) NAC-eMod

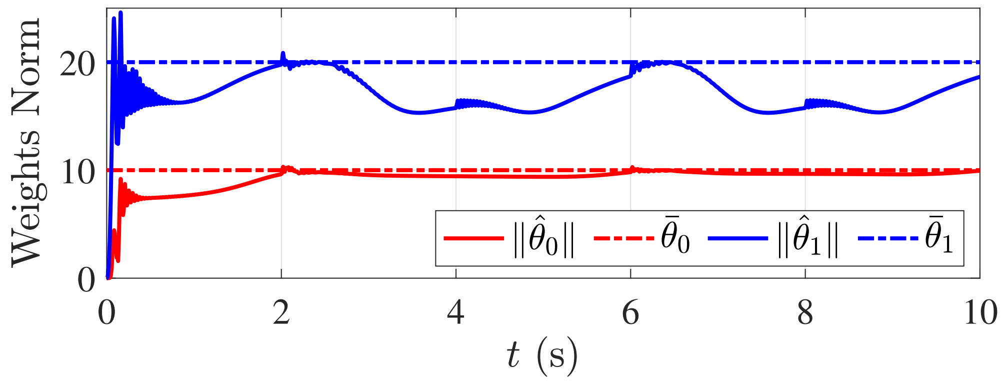


(c) CoNAC

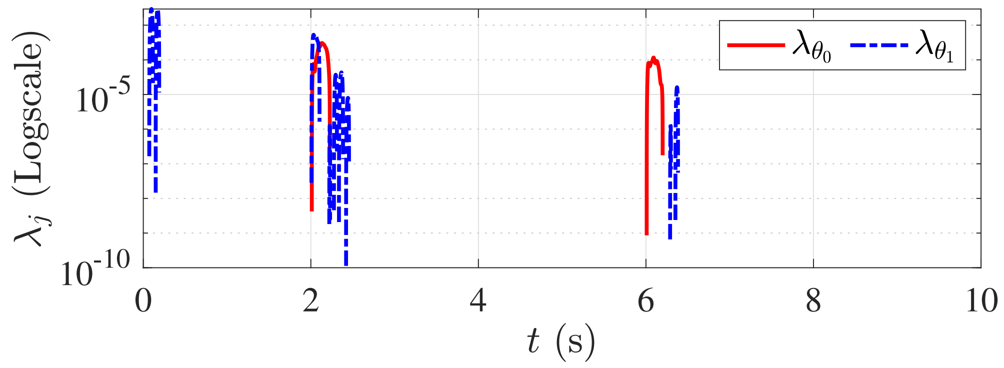
Figure 3.6: Tracking errors of NAC-L2, NAC-eMod and CoNAC.

point. The Lagrange multipliers disappear when the constraints are satisfied, and the weights are adapted to optimize the original objective function without weight bias.

Furthermore, it is important to note that CoNAC shows enhanced tracking performance with smaller weights than NAC-L2 and NAC-eMod. This implies that the weights in CoNAC approach the different local optimal solution points from those of NAC-L2 and NAC-eMod. Therefore, if the physical analysis of the system is available



(a) Weight norm



(b) Lagrange multipliers

Figure 3.7: Weight norms and Lagrange multipliers of CoNAC ($\beta = 0.001$).

to predict the feasible maximum control inputs, CoNAC can find the local optimal solution without unnecessary large control input by imposing the proper weight norm constraints.

3.8 Conclusion

In this chapter, a constrained optimization-based neuro-adaptive controller (Co NAC) with a single hidden layer neural network (SHLNN) and weight norm constraint is presented for the uncertain Euler-Lagrange systems. The boundedness of the weights is handled by formulating a constrained optimization problem with weight norm inequality constraints. Using the constrained optimization approach, the adaptation laws of the weights and Lagrange multipliers are derived. The boundedness of the tracking error and the weight estimation are analyzed via Lyapunov analysis. The simulation results demonstrated that the proposed controller outperforms the existing methods in terms of tracking performance and parameter dependency. In next chapter, CoNAC will be extended to handle input constraints and DNN.

Chapter 4
CoNAC for Uncertain Euler–Lagrange Systems Under Weight and Input Constraints

4.1 Introduction

In this chapter, the constrained optimization-based neuro-adaptive controller (CoNAC) framework is extended to address the input saturation problem. Besides the boundedness of neural network (NN) weights, the other major issue is satisfying input constraints, particularly in systems where actuators are subject to physical limitations [49]. The unpredictable outputs of NNs can sometimes lead to excessively large control inputs which may destroy the actuators or the system itself. This problem is exacerbated in neuro-adaptive controllers (NACs) 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. The control input saturation is reformulated into convex input constraints, then incorporated into the CoNAC. In addition, CoNAC presented in Chapter 3, is extended by substituting the single hidden layer neural network (SHLNN) by deep neural network (DNN).

4.2 Problem formulation

Consider an uncertain Euler-Lagrange system modeled as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(q) = h(\tau) \tag{4.1}$$

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 continuous and bounded, i.e. $\|\partial h/\partial \tau\|_F \in L_\infty$.

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 4.5 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, (4.1) can be reformulated as

$$M_0\ddot{q} + C_0\dot{q} + G_0 = h(\tau) + f(q, \dot{q}, \ddot{q}) \tag{4.2}$$

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 unknown function f while addressing the imposed constraints (*e.g.* weight boundedness and input saturation).

4.3 Existing Works to Prevent Input Saturation

To address input saturation, one may consider the projection operator in Section 3.3.1 as one of solutions. However, the projection operator can not be simply applied since the NNs are highly nonlinear and non-convex function. Note that the operator is used to project adaptation direction on given convex set.

On the other hand, 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 [19, 20, 50], auxiliary states were generated whenever input saturation was detected, and the auxiliary states were incorporated into the adaptation law to adjust the NN weights accordingly. For details, the auxiliary systems are generally designed as

$$\dot{\zeta} = A_\zeta \zeta + B_\zeta \Delta\tau$$

where ζ is the auxiliary state, A_ζ is Hurwitz matrix, B_ζ is control gain matrix, and $\Delta\tau_{(i)} = \tau_{(i)} - \tau_{\text{sat},(i)}$ is saturated control input of each control channel where $\tau_{\text{sat},(i)}$ denotes maximum or minimum value of $\tau_{(i)}$. Thus, the auxiliary state ζ is generated when the control input τ exceeds the saturation limit τ_{sat}. Using the auxiliary state,

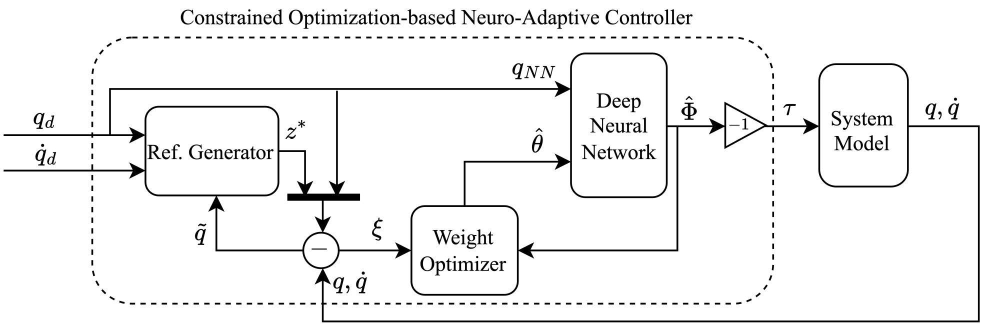


Figure 4.1: Architecture of the constrained optimization-based neuro-adaptive controller (CoNAC).

the feedback signal for the adaptation law is modified as

$$\dot{\hat{\theta}} = \gamma(\xi) \;\rightarrow\; \dot{\hat{\theta}} = \gamma(\xi + \zeta)$$

where $\gamma(\cdot)$ denotes arbitrary adaptation law and ξ denotes tracking error which is typically used as feedback signal for adaptation process for NACs. This approach helps 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 [22, 23, 51]. In addition, the NN was used to approximate the desired control input, which compensate for input saturation in [17].

However, these approaches typically handle input bound constraints on a per-input basis, and may not account for more complex and nonlinear constraints, like input norm constraints, which are commonly found in physical systems such as robotic actuators or motor systems due to their power limitations.

4.4 CoNAC with Weight and Input Constraints

The architecture of the proposed CoNAC is illustrated in Fig. 4.1, consisting of: a reference generator, a DNN that functions as NAC, and a weight optimizer for the DNN. The simple version of the CoNAC is introduced in Chapter 3 ahead with the SHLNN instead of the DNN and the weight norm constraints. We will extend it to the DNN with the weight and input constraints in this chapter.

4.4.1 Control Law Development

The system dynamics (4.2) can be represented as

$$\begin{aligned}\dot{q} &= z,\\ \dot{z} &= -M_0^{-1}C_0 z - M_0^{-1}G_0 + M_0^{-1}h(\tau) + M_0^{-1}f,\end{aligned} \tag{4.3}$$

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

Similar to the control law development in Section 3.4.1, the reference generator generates the desired trajectory $z^* \triangleq -k_q\tilde{q} + \dot{q}_d$ for $z \triangleq \dot{q}$, where $\tilde{q} \triangleq q - q_d$ and $k_q \in \mathbb{R}_{>0}$. Then the desired stabilizing controller can be designed as

$$\tau^* = -M_0(k_z\tilde{z} + \tilde{q}) + C_0 z + G_0 - f + M_0\dot{z}^* \tag{4.4}$$

where $\tilde{z} \triangleq z - z^*$, $k_z \in \mathbb{R}_{>0}$, and $k_z \in \mathbb{R}_{>0}$. Note that the control law τ^* cannot be realized because of f.

The DNN presented in Section 2.3, is defined as

$$\Phi(q_{NN};\theta) \triangleq \underbrace{V_k^T\phi_k(\underbrace{V_{k-1}^T\cdots\phi_2(\underbrace{V_1^T\phi_1(\underbrace{V_0^T q_{NN}}_{\Phi_0})}_{\Phi_1})\cdots)}_{\Phi_{k-1}})}_{\Phi_k}$$

where q_{NN} denotes the NN input vector, $V_i \in \mathbb{R}^{(l_i+1)\times l_{i+1}}$ 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. For the simplicity, the weights are vectorized in $\theta \triangleq [\theta_i]_{i\in[k,\cdots,0]} \in \mathbb{R}^{\Xi}$ consisting of vectorized weights of each layer such that $\theta_i \triangleq \text{vec}(V_i) \in \mathbb{R}^{\Xi_i}$, where $\Xi \triangleq \sum_{i\in[k,\cdots 0]}$ denotes the total number of weights and $\Xi_i \triangleq (l_i+1)l_{i+1}$ denote the number of weights in the i^{th} layer, respectively.

According to Theorem 2.3.1, the desired control law τ^* can be approximated by the DNN with the ideal weight vector θ^* on a compact subset $\Omega_{NN} \in \mathbb{R}^l$ to ϵ-accuracy, such that $\sup_{q_{NN}\in\Omega_{NN}} \|\Phi(q_{NN};\theta^*) - \tau^*\| = \epsilon < \infty$. The ideal weight θ^* is typically assumed to be bounded, i.e. $\|\theta^*\| \leq \bar{\theta} < \infty$. In this thesis, θ^* is defined as a local optimal point, rather than a global optimal point.

Then, the desired control law can be represented as follows:

$$\tau^* = -\,(\Phi^* + \epsilon),$$

which is estimated online by

$$\tau = -\,\hat{\Phi}, \tag{4.5}$$

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

Using (4.3), (4.4), (4.5), and the definition of $\tilde{z}$ the error dynamics can be derived as

$$\begin{aligned} \dot{\tilde{q}} &= -\,k_q\tilde{q} + \tilde{z} \\ \dot{\tilde{z}} &= -\,\tilde{q} - k_z\tilde{z} + M_0^{-1}(\Phi^* - h(\hat{\Phi}) + \epsilon). \end{aligned} \tag{4.6}$$

The error dynamics (4.6) can be represented as a first-order system:

$$\dot{\xi} = A_\xi\xi + B_\xi(\Phi^* - h(\hat{\Phi}) + \epsilon) \tag{4.7}$$

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 $\|B_\xi\|_F < \infty$. For further sections, $\phi_i^* \triangleq \phi_i(\Phi_{i-1}^*)$ and ${\phi^*}_i' = \partial\phi_i^*/\partial\Phi_{i-1}^*$, and $\hat{\phi}_i \triangleq \phi_i(\hat{\Phi}_{i-1})$ and $\hat{\phi}_i' = \partial\hat{\phi}_i/\partial\hat{\Phi}_{i-1}$.

4.4.2 Weight Adaptation Laws

The adaptation laws are same as the previous Section 3.4.2 and represented as follows:

$$\dot{\hat{\theta}} = -\alpha\frac{\partial L}{\partial\hat{\theta}} = -\alpha\left(\frac{\partial J}{\partial\hat{\theta}} + \sum_{j\in\mathcal{A}}\lambda_j\frac{\partial c_j}{\partial\hat{\theta}}\right), \tag{4.8a}$$

$$\dot{\lambda}_j = \beta_j\frac{\partial L}{\partial\lambda_j} = \beta_jc_j, \qquad \forall j \in \mathcal{A}, \tag{4.8b}$$

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

where $L \triangleq J(\xi;\hat{\theta}) + \sum_{j\in\mathcal{A}} \lambda_j c_j(\hat{\theta})$ denotes Lagrangian function consisting of the original objective function $J \triangleq (1/2)\xi^T\xi$, the inequality constraint $c_j,\ j \in \mathcal{I}$ and Lagrange multiplier λ_j, where $\mathcal{I}$ is the set of imposed constraints, $\mathcal{A} \triangleq \{j \in \mathcal{I} \mid c_j \geq 0\}$ represents the active set. Moreover, $\alpha \in \mathbb{R}_{>0}$ denotes the adaptation gain (learning rate) and $\beta_j \in \mathbb{R}_{>0}$ denotes the update rate of the Lagrange multipliers in $\mathcal{A}$.

As presented in Section 3.4.2, the gradient of the objective function with respect to the weights can be represented as

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

where $W \in \mathbb{R}^{2n\times 2n}$ is a positive weight matrix, and $\partial\xi/\partial\hat{\theta}$ is the sensitivity of the weights to the augmented error. The sensitivity of the weights can be obtained by simulating the sensitivity equation as follows:

$$\begin{aligned} \dot{\eta} &= \begin{bmatrix} \eta_k & \eta_{k-1} & \cdots & \eta_0 \end{bmatrix}' \\ &= A_\xi \begin{bmatrix} \eta_k & \cdots & \eta_0 \end{bmatrix} - B_\xi \frac{\partial h}{\partial \tau} \begin{bmatrix} (I_{l_{k+1}} \otimes \hat{\phi}_k^T) & \cdots & (\cdot) \end{bmatrix} \end{aligned} \tag{4.9}$$

with zero initial value of η, since the initial ξ is independent of the weights.

The adaptation is implemented using Algorithm 2. 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 [33, Chap. 3 eq. (3.4)] to ensure that the objective function decreases.

4.5 Constraint Candidates

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

Assumption 4.5.1. *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 4.5.2. *The selected constraints satisfy the Linear Independence Constraint Qualification (LICQ) defined in Definition 2.2.2.*

Algorithm 2: Weight optimizer implementation.

```
input  : $\xi$, $\hat{\theta}$, $\lambda_j$, $\eta$
output: $\hat{\theta}$, $\lambda_j$, $\eta$
1 Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{j\}$ for all $c_j \geq 0$;
2 Determine update matrix $\dot{\eta}$ using (4.9);
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 (4.8a), (4.8b);
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 $\lambda_j = 0$;
```

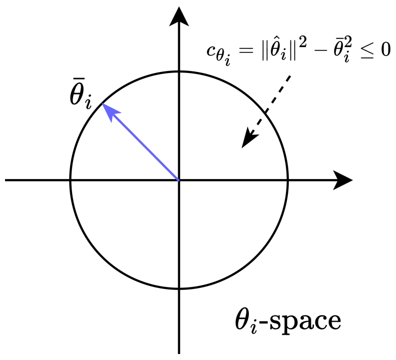

$c_{\theta_i} = \|\hat{\theta}_i\|^2 - \bar{\theta}_i^2 \leq 0$
$\bar{\theta}_i$
θ_i-space

Figure 4.2: Weight norm constraints.

4.5.1 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 as shown in Fig. 4.2, where

$$c_{\theta_i} = \|\hat{\theta}_i\|^2 - \bar{\theta}_i^2 \leq 0 \tag{4.10}$$

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

$$\frac{\partial \mathbf{c}_\theta}{\partial \hat{\theta}} \triangleq \begin{bmatrix} (\partial c_{\theta_0}/\partial \hat{\theta})^T \\ \vdots \\ (\partial c_{\theta_k}/\partial \hat{\theta})^T \end{bmatrix} = 2 \cdot \begin{bmatrix} 0 & 0 & \cdots & \hat{\theta}_0^T \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \hat{\theta}_{k-1}^T & \cdots & 0 \\ \hat{\theta}_k^T & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{(k+1)\times\Xi}. \tag{4.11}$$

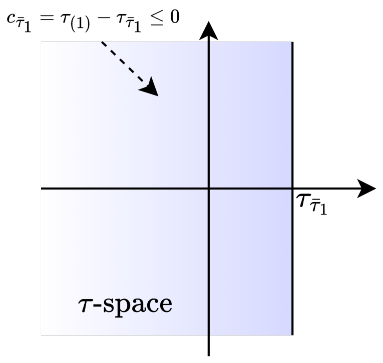

$c_{\bar{\tau}_1} = \tau_{(1)} - \tau_{\bar{\tau}_1} \leq 0$
$\tau_{\bar{\tau}_1}$
τ-space

Figure 4.3: Input bound constraints.

4.5.2 Input Bound Constraint

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

$$c_{\bar{\tau}_i} = \tau_{(i)} - \tau_{\bar{\tau}_i} \leq 0, \quad c_{\underline{\tau}_i} = \tau_{\underline{\tau}_i} - \tau_{(i)} \leq 0 \tag{4.12}$$

with $\tau_{\bar{\tau}_i}$ and $\tau_{\underline{\tau}_i}$ representing the maximum and minimum control input bounds, respectively. For the case of $c_{\bar{\tau}_1}$ is shown in Fig. 4.3. The gradients of $\mathbf{c}_{\bar{\tau}}$ and $\mathbf{c}_{\underline{\tau}}$ with respect to $\hat{\theta}$ are given by

$$\begin{aligned}
\frac{\partial c_{\bar{\tau}}}{\partial \hat{\theta}} &\triangleq \begin{bmatrix} (\partial c_{\bar{\tau}_1}/\partial\hat{\theta})^T \\ \vdots \\ (\partial c_{\bar{\tau}_n}/\partial\hat{\theta})^T \end{bmatrix} = -\frac{\partial \hat{\Phi}}{\partial \hat{\theta}} = -\begin{bmatrix} (I_{l_{k+1}} \otimes \hat{\phi}_k^T) & \cdots & (\cdot) \end{bmatrix} \in \mathbb{R}^{n\times\Xi}, \\
\frac{\partial c_{\underline{\tau}}}{\partial \hat{\theta}} &\triangleq \begin{bmatrix} (\partial c_{\underline{\tau}_1}/\partial\hat{\theta})^T \\ \vdots \\ (\partial c_{\underline{\tau}_n}/\partial\hat{\theta})^T \end{bmatrix} = +\frac{\partial \hat{\Phi}}{\partial \hat{\theta}} = +\begin{bmatrix} (I_{l_{k+1}} \otimes \hat{\phi}_k^T) & \cdots & (\cdot) \end{bmatrix} \in \mathbb{R}^{n\times\Xi}.
\end{aligned} \tag{4.13}$$

4.5.3 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:

$$c_u = \|\tau\|^2 - \bar{\tau}^2 \leq 0 \tag{4.14}$$

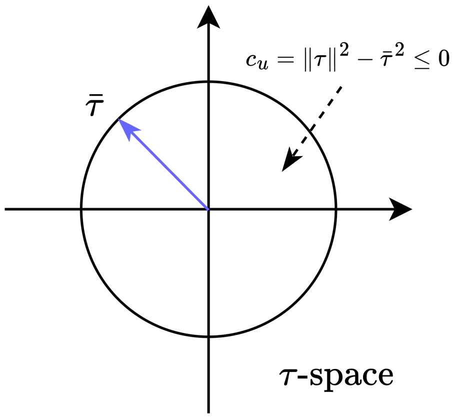


Figure 4.4: Input norm constraints.

with $\bar{\tau} \in \mathbb{R}_{>0}$ denoting the maximum allowable control input magnitude as shown in Fig. 4.4. This input norm constraint is also commonly applied in current and torque control problems for electric motors [52]. The gradients of c_u with respect to $\hat{\theta}$ are given by

$$\frac{\partial c_u}{\partial \hat{\theta}} = -\sum_{i=1}^{n} 2\tau_{(i)} \left(\mathrm{row}_i \left(-\frac{\partial \hat{\Phi}}{\partial \hat{\theta}} \right) \right)^T = \tau^T (I_{l_{k+1}} \otimes \hat{\phi}_k^T) \in \mathbb{R}^{\Xi}. \tag{4.15}$$

It should be noted that constraints (4.12) and (4.14) cannot be imposed simultaneously, as their gradients (4.13) and (4.15) are linearly dependent, violating the LICQ condition (see Definition 2.2.2).

4.6 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 4.6.1. *If Assumptions 4.5.1 and 4.5.2 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 Proposition 2.2.1, a linear map $T(\cdot) : \hat{\theta}_k \to \tau$ can be derived as follows:

$$\begin{aligned} \tau =& -\hat{\Phi} = -\mathrm{vec}(\hat{\Phi}) = -\mathrm{vec}(\hat{V}_k \hat{\phi}_k) \\ =& -(I_{l_{k+1}} \otimes \hat{\phi}_k^T)\mathrm{vec}(\hat{V}_k) = -(I_{l_{k+1}} \otimes \hat{\phi}_k^T)\hat{\theta}_k = T(\hat{\phi}_k)\hat{\theta}_k. \end{aligned} \tag{4.16}$$

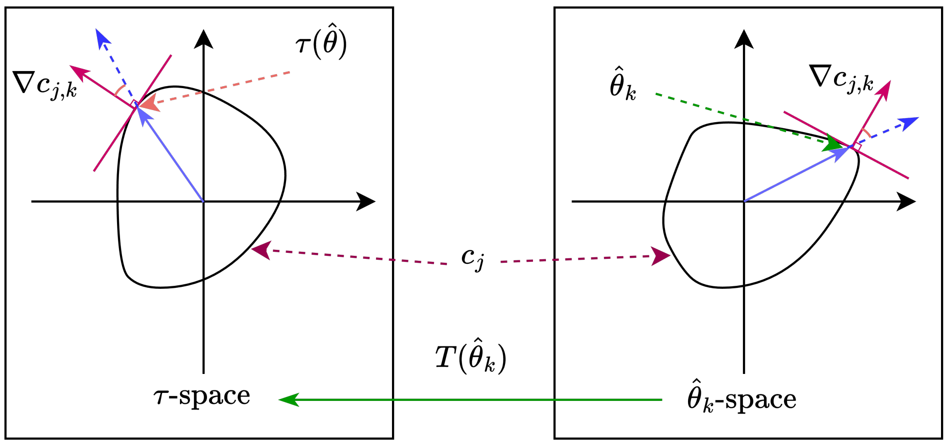


Figure 4.5: Convexity of input constraints.

Therefore, since the linear map is affine map, the convexity of the input constraints in τ-space (assumed in Assumption 4.5.1) holds in $\hat{\theta}_k$-space as well as discussed in Section 2.2.3, implying that $(\partial c_j/\partial\hat{\theta}_k)^T\hat{\theta}_k > 0$ (see Lemma 2.2.1). The preservation of convexity is illustrated in Fig. 4.5.

□

Lemma 4.6.2. *If $c_j(\hat{\theta})$, $\forall j \in \mathcal{I}\backslash\{\theta_i\}_{i\in[0,\cdots,k]}$ satisfies Assumption 4.5.1, 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 (4.16), 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 4.5.1, 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 verified using the definition of $\partial\hat{\Phi}/\partial\hat{\theta}_i$ given in (2.2). 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 4.6.1. *For the dynamical system in* (4.1)*, the neuro-adaptive controller* (4.5) *and weight adaptation laws* (4.8) *ensure the boundedness of the augmented error ξ and the weight estimate $\hat{\theta}$. This holds with the weight norm constraint* (4.10) *and input constraints satisfying Assumption 4.5.2 and 4.5.2, provided that the control gains k_q and k_z satisfy* (4.18)*.*

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 (4.7), using Theorem 2.1.1, 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 in active set $\mathcal{A}$ without loss of generality. The dynamics of η_k and $\hat{\theta}_k$ can be decomposed from (4.8a) and (4.9) 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 Theorem 2.1.1, $\|\eta_k\|_F$ 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:

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

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:

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

By applying the boundedness of $\Delta \triangleq PB(V_k^{*T}\phi_k^* - h(\tau) + \epsilon)$ and $M \triangleq \eta_k W$, where

$\|\Delta\| \leq \bar{\Delta} < \infty$ and $\|M\|_F \leq \bar{M} < \infty$, this simplifies to:

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

Representing the term $\partial c_{\theta_k}/\partial\hat{\theta}_k$ in the last inequality as $\partial c_{\theta_k}/\partial\hat{\theta}_k = 2\hat{\theta}_k$ and using the result provided in (4.11), $\dot{\mathcal{V}}$ can be rewritten as

$$\begin{aligned}\dot{\mathcal{V}} \leq& (\cdot) + \left(-2\lambda_{\theta_k} + \bar{M}/2\right)\|\hat{\theta}_k\|^2 - \underbrace{\sum_{j\in\mathcal{A}\setminus\{\theta_i\}_{i\in[0,\cdots,k]}} \lambda_j \hat{\theta}_k^T \frac{\partial c_j}{\partial \hat{\theta}_k}}_{>0,\text{ by Lemma4.6.1 and Assumption4.5.2}} \\ \leq& \left(-\lambda_{\min}(Q)/2 + \bar{M}/2\right)\|\xi\|^2 + \bar{\Delta}\|\xi\| + \left(-2\lambda_{\theta_k} + \bar{M}/2\right)\|\hat{\theta}_k\|^2\end{aligned} \tag{4.17}$$

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 (4.17), if the control gains k_q and k_z are chosen sufficiently large to satisfy the condition:

$$\min(k_q, k_z) > \bar{M}/2 \tag{4.18}$$

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

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

(as dictated by (4.8b) 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

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

and

$$\Theta_{\hat{\theta}_k} = \{\hat{\theta}_k \mid \|\hat{\theta}_k\| \leq \bar{\theta}_k\}.$$

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 λ_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, λ_{θ_r}, $\forall r \in [0, \cdots, k-1]$, will be examined in Step i. Based on Assumption 4.5.1 and Lemma 4.6.1, the equality constraints $c_j \leq 0$, $\forall j \in \mathcal{A} \setminus \{\theta_i\}_{i\in[0,\cdots,k]}$, for convex 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 η_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 (2.2). 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 Theorem 2.1.1, $\|\eta_i\|_F$ 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 4.6.2. 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.

□

4.7 Simulation Validation

4.7.1 Setup

The proposed CoNAC was validated using a two-link manipulator model, as depicted in Fig. 4.6, adapted from [48]. The parameters $q_p, q_{dp}, \tau_p, m_p, l_p, l_{cp}, b_p$, and f_{cp} 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 values of the system model parameters are provided in Table 4.1. The desired trajectory for $q = [q_1, q_2]^T$ was defined as

$$q_d(t) = \begin{bmatrix} q_{d1} \\ q_{d2} \end{bmatrix} = \begin{bmatrix} +\cos(0.49\pi t) + 1 \\ -\cos(0.49\pi t) - 1 \end{bmatrix}.$$

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 [53] and is given by

$$\mathrm{SSF}_L^U(\|\tau\|) = \frac{\|\tau\|}{(1 + (\|\tau\|/\bar{\tau})^p)^{1/p}}. \tag{4.19}$$

with p being the smoothing factor. The effect of p and the boundedness of $\|\partial h/\partial\tau\|_F$ is shown in Fig. 4.7. 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 (4.14) 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

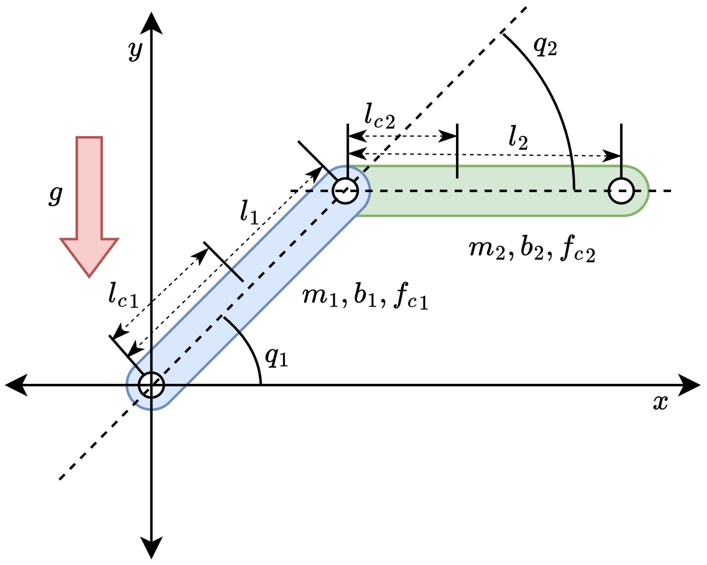


Figure 4.6: Two-link manipulator model.

Table 4.1: 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)

third was DNN-BSC augmented with an auxiliary system presented in [19,20,50] (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 4.2.

The BSC used the control law defined in (4.4). Since BSC did not consider the unknown system dynamics, the approximation term $\hat{f}$ was set to zero (i.e. $\hat{f} = [0, 0]^T$). 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

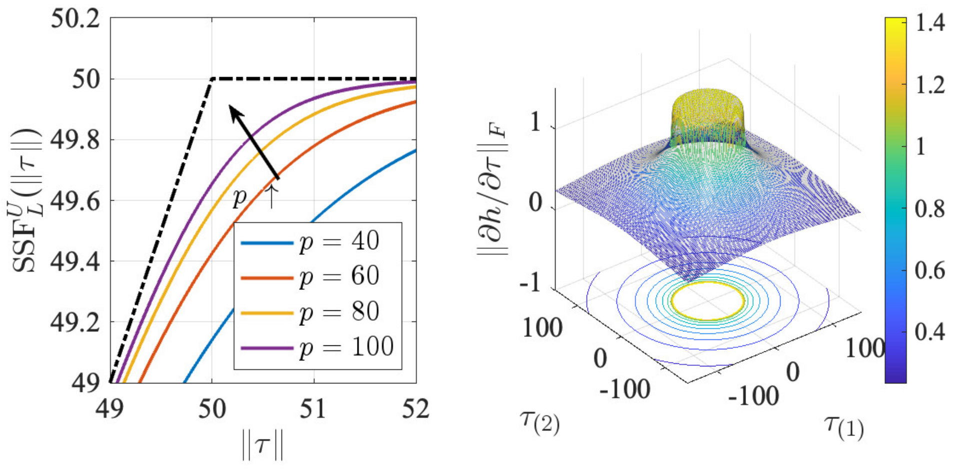


Figure 4.7: Effect of parameter p in the control input saturation function h and boundedness of $\|\partial h/\partial \tau\|_F$.

Table 4.2: 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)

presented in [28], was defined by

$$\dot{\hat{\theta}} = \mathrm{Proj}_\Omega[\alpha(\partial\hat{\Phi}/\partial\hat{\theta})M_0^{-1}\tilde{z}], \tag{4.20}$$

where $\mathrm{Proj}_\Omega(\cdot)$ is the projection operator defined in (3.3), 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 (4.10).

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 (4.14), an approximated version of the input norm constraint was used as an input bound constraint (4.12) 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. 4.12. The auxiliary system is defined as $\dot{\zeta} = A_\zeta \zeta + B_\zeta \Delta\tau, \quad \zeta|_{t=0} = 0$, where $\zeta \in \mathbb{R}^n$ denotes the auxiliary state, $A_\zeta = -[20, 0; 0; 20], B_\zeta = [10, 0; 0, 10]$, and $\Delta\tau_{(i)} = \tau_{(i)} - \mathrm{sat}(\tau_{(i)}, \overline{\tau}_i, \underline{\tau}_i)$. The auxiliary state variables were used in the adaptation law (4.20) by substituting $\tilde{z}$ with $\tilde{z} + \zeta$.

The proposed CoNAC directly approximated the control law using the DNN as defined in (4.5). The update rates for the Lagrange multipliers were set as $\beta_j = 0.1$. 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. $q_{NN} = [{q_d}^T, 1]^T$ 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. (4.19)). 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 $T_s = 10^{-4}$.

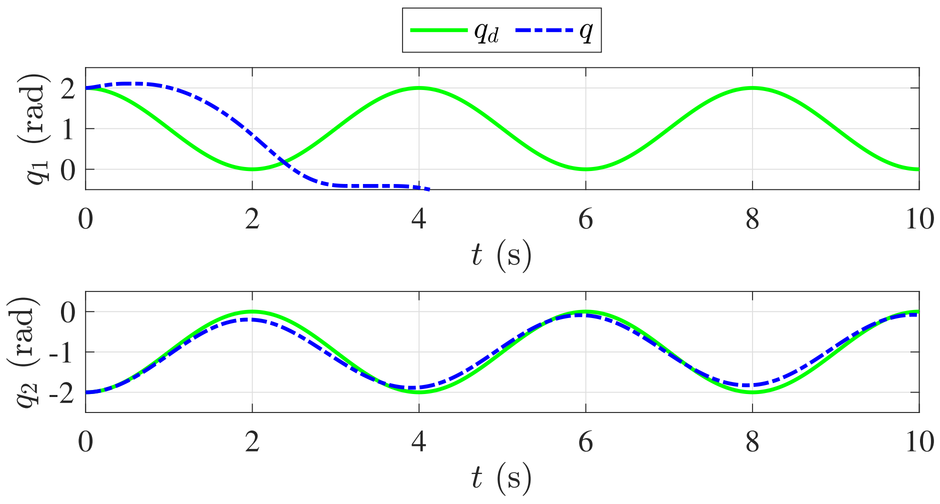
4.7.2 Results

System Uncertainty Handling

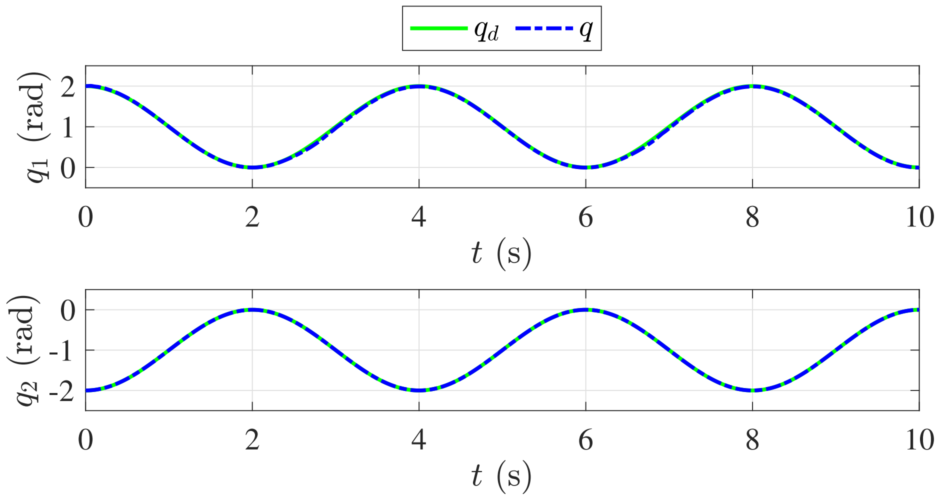
The tracking results of the selected controllers are shown in Fig. 4.8 and Fig. 4.9. To demonstrate the effectiveness of using DNNs for compensating the lumped system uncertainty function f, the gains k_q and k_z for BSC were intentionally selected as small values, resulting in a weak ability to handle these uncertainties. As a result, BSC failed to track the reference trajectory, as shown in Fig. 4.8a.

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. 4.8b. Fig. 4.9a 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 4.7.2.

Finally, CoNAC, which directly approximates the stabilizing control law along with the compensation term, demonstrated satisfactory tracking performance across both



(a) BSC



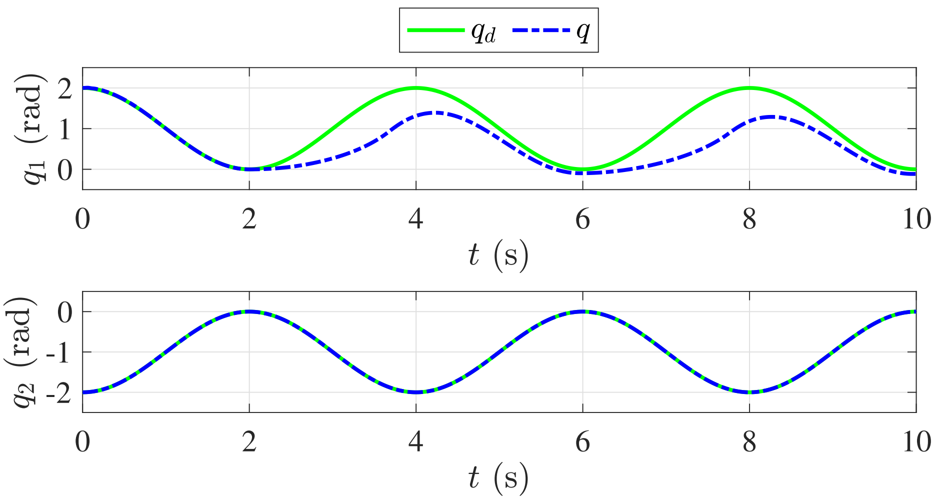
(b) DNN-BSC

Figure 4.8: Comparison of the tracking performance of BSC and DNN-BSC.

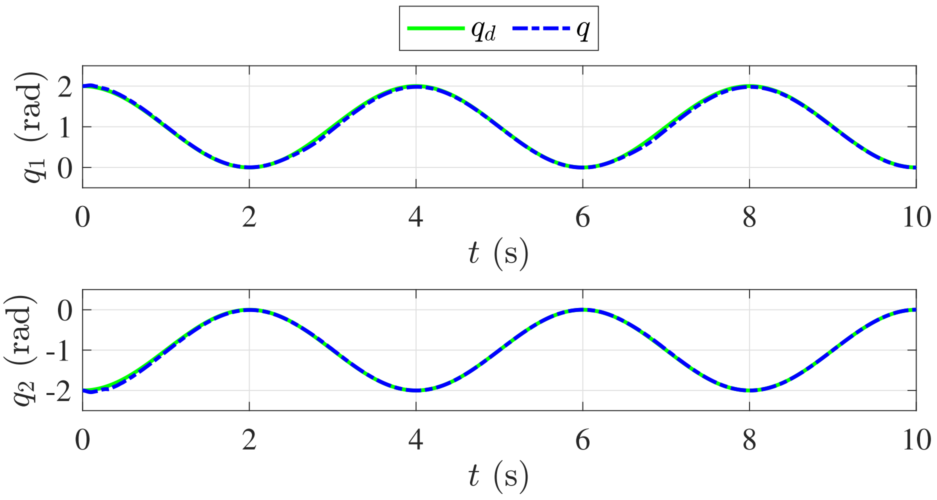
states, as illustrated in Fig. 4.9b.

Input Norm Constraint Handling

The resulting control input τ and physically saturation control input $h(\tau)$ for the selected controllers are shown in Fig. 4.10 and Fig. 4.11. As illustrated in Fig. 4.10a, 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. 4.10b. This failure to account for the input norm constraint led to oscillations in the control input τ. The DNN



(a) DNN-BSC-A

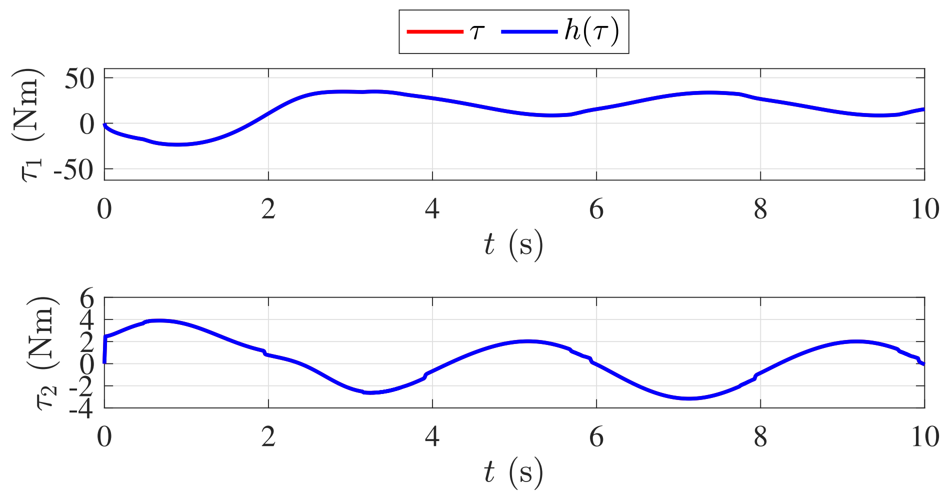


(b) CoNAC

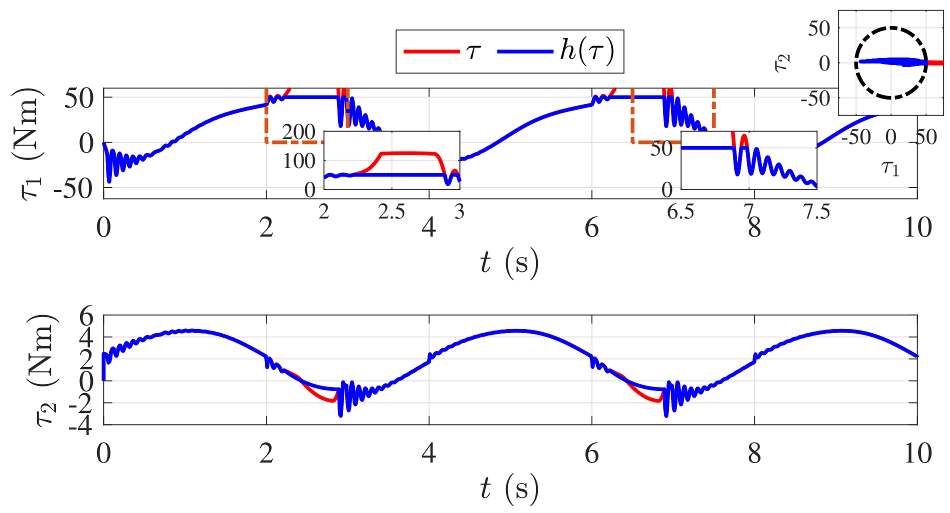
Figure 4.9: Comparison of the tracking performance of DNN-BSC-A and CoNAC.

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. 4.11a and Fig. 4.11b, 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



(a) BSC

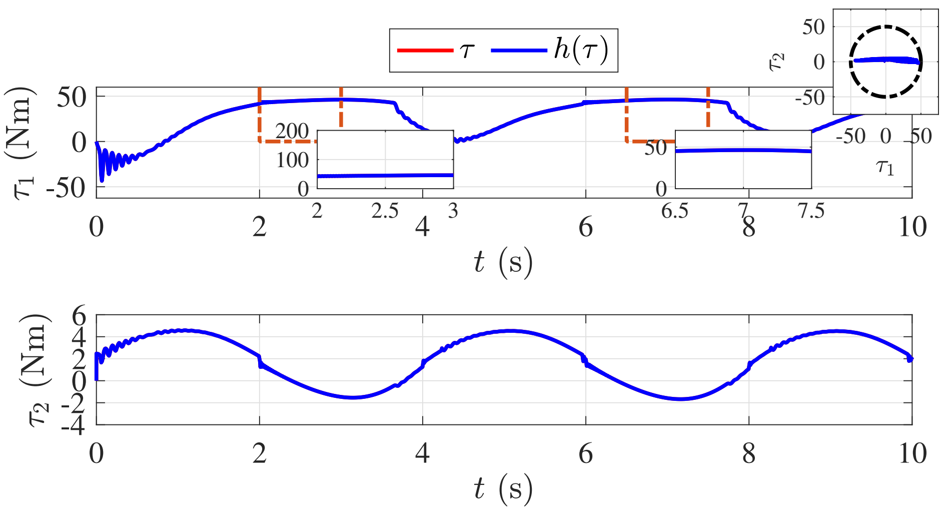


(b) DNN-BSC

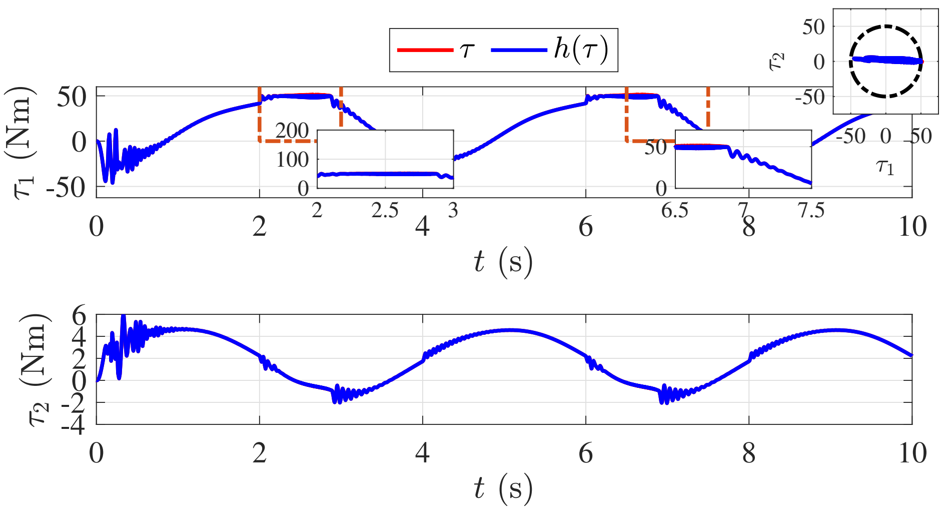
Figure 4.10: Comparison of the control input τ and the physically saturated control input $h(\tau)$ of BSC and DNN-BSC.

the input norm constraint with an input bound constraint, creating a rectangular constraint in the τ-space (see Fig. 4.12). 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. 4.12, where DNN-BSC-A exhibited minor violations of the input bound constraint. In contrast, CoNAC satisfied the constraint



(a) DNN-BSC-A



(b) CoNAC

Figure 4.11: Comparison of the control input τ and the physically saturated control input $h(\tau)$ of DNN-BSC-A and CoNAC.

without being affected by such dynamics, as its Lagrange multiplier adjusted as soon as the constraint was violated.

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. 4.13 and Fig. 4.14. All three controllers—DNN-BSC, DNN-BSC-A, and CoNAC—maintained weight norms within the imposed weight norm constraints.

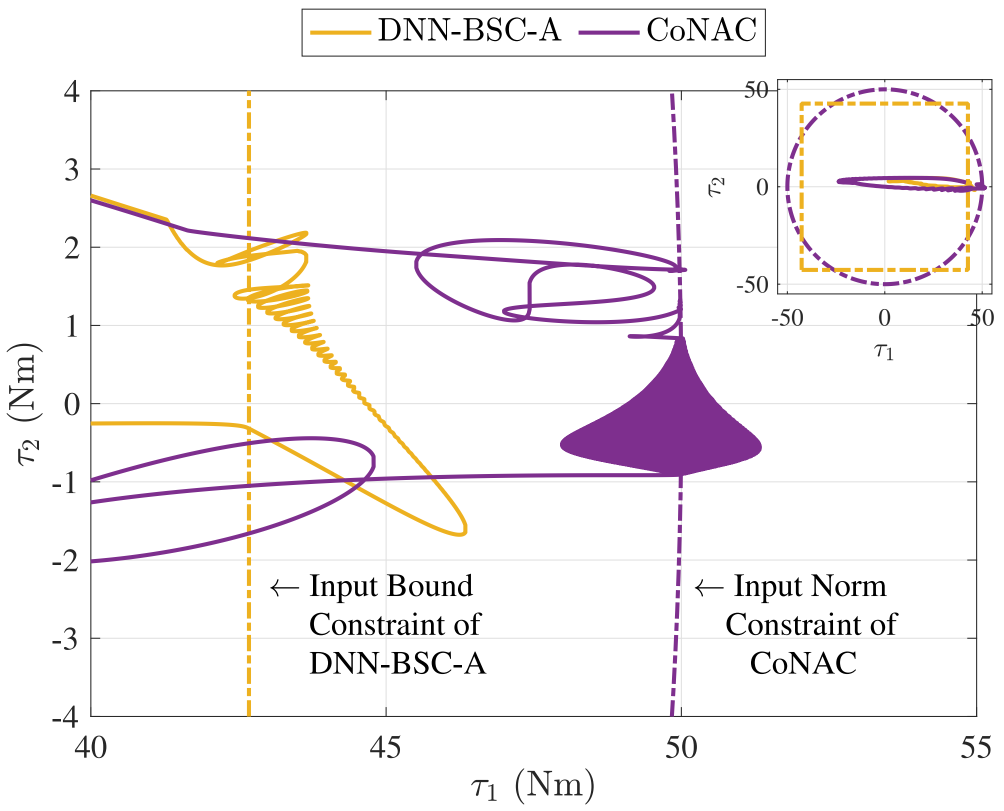
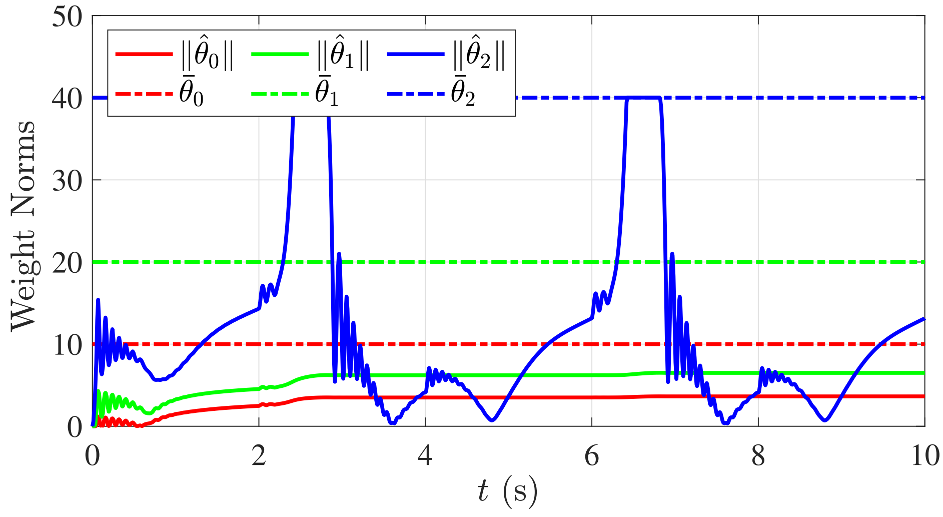


Figure 4.12: Control input paths of DNN-BSC and CoNAC during the time interval from 5 s to 8 s.

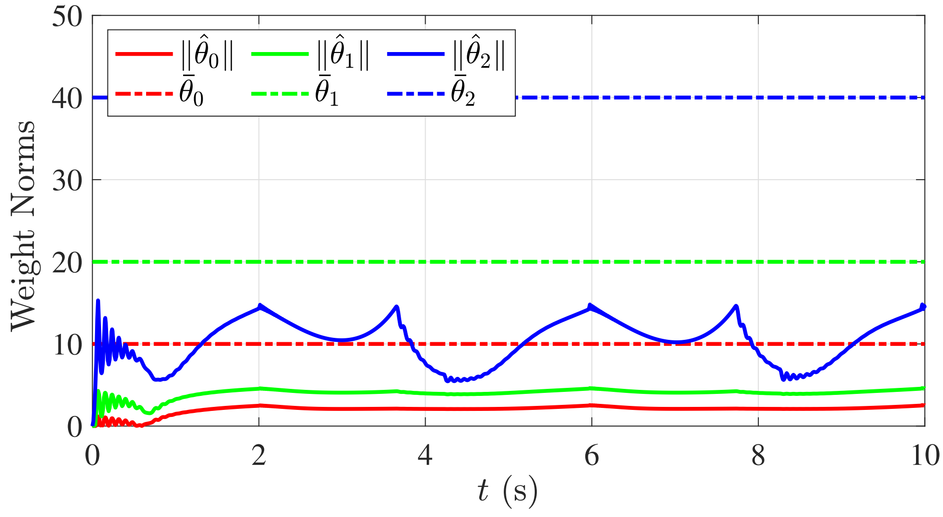
IN DNN-BSC, as shown in Fig. 4.13a, the weight norm of the last layer (i.e. $\|\hat{\theta}_2\|$) fluctuated significantly over time, proportional to the control input norm. This is because the last layer's weights directly determine the control input. When the control input violated the input norm constraint, the last layer's weight norm hit the boundary and stayed there due to the projection operator. However, the projection operator 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. 4.13b. 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.4.14b. 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.4.14a). 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 $\|\hat{\theta}_2\|$



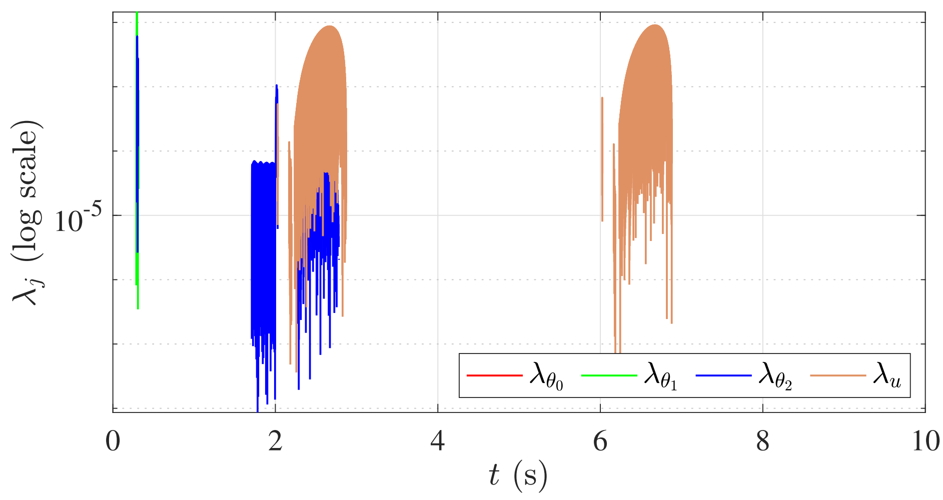
(a) Weight norms of DNN-BSC



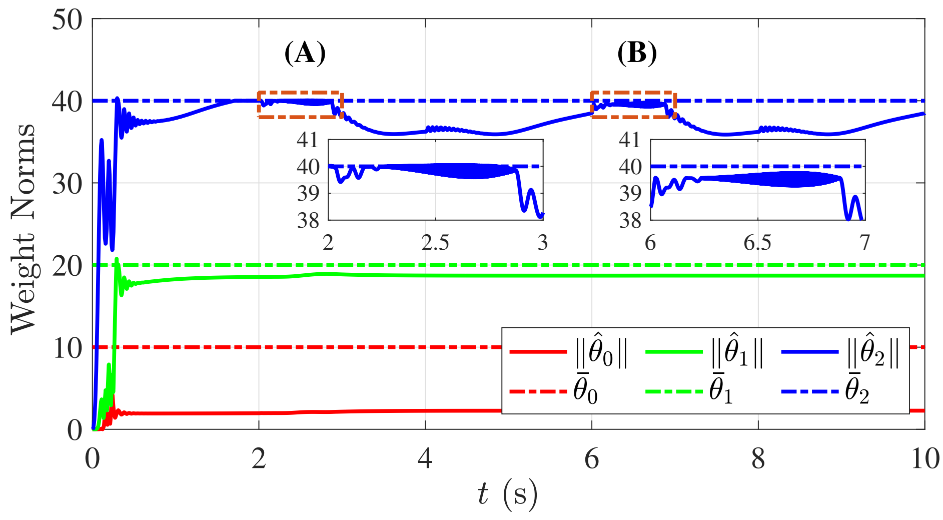
(b) Weight norms of DNN-BSC-A

Figure 4.13: Weight norms of DNN-BSC and DNN-BSC-A.

stabilized within the upper bound by around 6.5 seconds (see Fig. 4.14b (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. 4.14b (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.



(a) Multipliers of CoNAC



(b) Weight norms of CoNAC

Figure 4.14: Lagrange multipliers and weight norms CoNAC.

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.

Comparison Analysis of Computation Times

In addition, the computation times were analyzed and summarized in Table 4.3. The MATLAB's functions `tic` and `toc` were used to measure the computation time at each time step and the average values were calculated. The simulations were conducted on a

Table 4.3: Average computation times.

Computation time	**BSC**	**DNN-BSC**	**DNN-BSC-A**	**CoNAC**
Control (10^{-5} s)	0.087	0.755	0.7477	0.7465
Ratio	0.112	1	0.988	0.987
Train ($\times 10^{-5}$ s)	-	3.877	3.821	3.826
Ratio	-	1	0.986	0.987

MacBook Air (2021 model) with an M1 processor and 8 GB of RAM. The computation times of DNN-BSC were used as the reference for the comparison.

The computing times of the control decisions and training processes of all controllers were below the simulation sampling time $T_s = 10^{-4}$ indicating their suitability for real-time applications, as shown in Table 4.3. Notably, controllers utilizing DNNs exhibited higher computation times than BSC due to the additional processing required for the DNNs. However, compared to the DNN-BSC and DNN-BSC-A, the CoNAC had similar computation times for both the control and training processes. In other words, even though the CoNAC involves the additional process for the constraints and the Lagrange multipliers, its computation times were not significantly increased compared to the DNN-BSC and DNN-BSC-A. Therefore, the CoNAC can achieve the better tracking performance and the constraint handling capability which were discussed in aforementioned sections, without notable increase in the computation times compared to existing methods with DNNs.

4.8 Conclusion

In this chapter, a constrained optimization-based neuro-adaptive controller (CoN AC) for the uncertain Euler-Lagrange system is extended to address both weight norm and input constraints using deep neural network (DNN). The adaptation law is derived through a rigorous optimization framework. The stability of the proposed controller was analyzed using Lyapunov theory, ensuring that the system maintains 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.

Chapter 5
Conclusion and Future Work

In this thesis, the constrained optimization-based neuro-adaptive controller (CoNAC) for uncertain Euler-Lagrange systems is presented. The two simulation validations showed that the CoNAC can satisfy the imposed constraints regarding boundedness of neural network's (NN's) weights and control input saturation, while achieving the desired tracking performance.

However, neuro-adaptive control (NAC) methods including CoNAC have several limitations to be referred as deep learning-based controller. First, NAC methods adapts their weights to reduce objective function using current tracking error. This means online implementation is required to train the NNs since the tracking error is dependent on the current NNs' weights. Moreover, for the same reason, the NNs cannot be trained offline. Second, the gradient vanishing problem still exists in the train process of the NNs. To overcome this issue, simply ReLU activation function can be used. However, the stability should be examined more rigorously than $\tanh(\cdot)$, since ReLU is unbounded and not continuously differentiable.

The following future works are suggested to tackle above limitations. For the first limitation, first, reinforcement learning (RL) approach can be used, since training NNs to minimize objective function is similar as RL which trains to maximize the expected reward typically defined as sign changed tracking error. There are some literature based on optimal control theory [54–56]. They approximate optimal control law which is ideally obtained using Hamilton-Jacobi-Bellman framework or concept of value function. Second, since the ideal desired control law is unavailable, the NAC problem can be reformulated as a system identification problem by re-design NAC to approximate system dynamics instead of control law. In contrast to the control law approximation, the NNs can be trained offline using the system identification data, if the accurate system dynamics or observation is available.

For the second limitation, other novel constrained optimization methods can be used to solve gradient vanishing issue. Except gradient descent-like methods, the existing methods to overcome gradient vanishing issue using constrained optimization approach such as the augmented Lagrangian method (ALM) [34] and the alternating direction method of multipliers (ADMM) [35, 36] are introduced. These methods transform the NN's architecture of the NNs to equality constraints and optimize each layer's weights and output of activation functions. In other words, the output of NNs can be considered as one of the optimization variables in optimization problem. Hence, it may simplify the stability analysis of CoNAC with complex constraints since the nonlinearity of NNs can be omitted in the adaptation derivation.

On the other hand, if the offline adaptation is available, stochastic approach can be used to theoretically utilize novel deep learning methods. Since the recent deep learning methods are based on stochastic methods (*e.g.* , stochastic gradient descent (SGD), drop out, L_2-regularization), stochastic stability analysis should be conducted if such methods are used in system. The stability analysis of system which uses stochastic methods is introduced in [57, 58]. By conducting stability analysis for stochastic systems with NNs and stochastic methods, the stability of the controllers with novel NN methods (*e.g.* convolutional NN, transformer) can be theoretically and rigorously analyzed.

Summary

Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Euler-Lagrange Systems

In this thesis, a constrained optimization-based neuro-adaptive controller (CoNAC) for uncertain Euler-Lagrange systems is presented. The deep neural network (NN) in the CoNAC is used to approximate the uncertainties of the system. Therefore, any prior knowledge of the system uncertainties is not required. To derive the adaptation laws of NN's weights and Lagrange multipliers, the control problem is formulated as a constrained optimization problem. Satisfactions of the weights' boundedness and control input saturation are transformed into constraints in the constrained optimization problem. Using the corresponding Lagrangian function, the adaptation laws are derived, and they satisfy the first-order optimality conditions at the steady state. The stability of the CoNAC is analyzed using the Lyapunov stability theorem with boundedness of weights and tracking errors guaranteed. Two simulation demonstrated that the CoNAC can achieve the sufficient tracking performance while satisfying the constraints on the weights and control input saturation.

References

1. S. Todorovic, A. Wagner, S. Müller, and J. Neubeck, "Neural network based model for friction potential estimation under longitudinal and lateral excitations," in *Journal of Physics: Conference Series*, vol. 2234, p. 012005, IOP Publishing, 2022.

2. Y. Shen, J. Mao, A. Wu, R. Liu, and K. Zhang, "Optimal slip ratio tracking integral sliding mode control for an emb system based on convolutional neural network online road surface identification," *Electronics*, vol. 11, no. 12, p. 1826, 2022.

3. A. M. Ribeiro, A. Moutinho, A. R. Fioravanti, and E. C. de Paiva, "Estimation of tire–road friction for road vehicles: a time delay neural network approach," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 1, p. 4, 2020.

4. S. Wiedemann and C. M. Hackl, "Simultaneous identification of inverter and machine nonlinearities for self-commissioning of electrical synchronous machine drives," *IEEE Transactions on Energy Conversion*, vol. 38, no. 3, pp. 1767–1780, 2023.

5. V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *2017 American Control Conference (ACC)*, pp. 4914–4919, IEEE, 2017.

6. J. Jhung, I. Bae, J. Moon, T. Kim, J. Kim, and S. Kim, "End-to-end steering

controller with cnn-based closed-loop feedback for autonomous vehicles," in *2018 IEEE intelligent vehicles symposium (IV)*, pp. 617–622, IEEE, 2018.

7. M. A. Buettner, N. Monzen, and C. M. Hackl, "Artificial neural network based optimal feedforward torque control of interior permanent magnet synchronous machines: A feasibility study and comparison with the state-of-the-art," *Energies*, vol. 15, no. 5, p. 1838, 2022.

8. N. Monzen and C. M. Hackl, "Artificial neural network based optimal feedforward torque control of electrically excited synchronous machines," in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, pp. 1–8, IEEE, 2023.

9. N. Monzen, F. Stroebl, H. Palm, and C. M. Hackl, "Multi-objective hyperparameter optimization of artificial neural networks for optimal feedforward torque control of synchronous machines," *IEEE Open Journal of the Industrial Electronics Society*, 2024.

10. Y.-h. Sheu, "Illuminating the black box: interpreting deep neural network models for psychiatric research," *Frontiers in Psychiatry*, vol. 11, p. 551299, 2020.

11. C. Rudin and J. Radin, "Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition," *Harvard Data Science Review*, vol. 1, no. 2, pp. 1–9, 2019.

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

13. G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

14. P. Ioannou and B. Fidan, *Adaptive control tutorial.* SIAM, 2006.

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

16. A. Yeşildirek and F. L. Lewis, “Feedback linearization using neural networks,” *Automatica*, vol. 31, no. 11, pp. 1659–1664, 1995.

17. 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.

18. Y. Zhang, P. Y. Peng, and Z. P. Jiang, “Stable neural controller design for unknown nonlinear systems using backstepping,” *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1347–60, 2000.

19. 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.

20. 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.

21. S. You, G. Kim, S. Lee, D. Shin, and W. Kim, "Neural approximation-based adaptive control using reinforced gain for steering wheel torque tracking of electric power steering system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

22. 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.

23. 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.

24. S.-J. Kim and J.-H. Suh, "Adaptive robust rbf-nn nonsingular terminal sliding mode control scheme for application to snake robot's head for image stabilization," *Applied Sciences*, vol. 13, no. 8, p. 4899, 2023.

25. S.-J. Kim, M. Jin, and J.-H. Suh, "A study on the design of error-based adaptive robust rbf neural network back-stepping controller for 2-dof snake robot's head," *IEEE Access*, vol. 11, pp. 23146–23156, 2023.

26. 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.

27. X. Zhou, H. Shen, Z. Wang, H. Ahn, and J. Wang, "Driver-centric lane-keeping assistance system design: A noncertainty-equivalent neuro-adaptive control ap-

proach," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 6, pp. 3017–3028, 2023.

28. 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.

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

30. 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.

31. 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.

32. Z. Liu, C. Wu, X. Shen, W. Yao, J. Liu, and L. Wu, "Adaptive interval type-2 fuzzy neural network-based novel fixed-time backstepping control for uncertain euler-lagrange systems," *IEEE Transactions on Fuzzy Systems*, 2024.

33. J. Nocedal and S. J. Wright, *Numerical optimization.* Springer, 1999.

34. 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, 2021.

35. 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, 2019.

36. 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, 2016.

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

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

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

40. S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

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

42. A. R. Barron, “Universal approximation bounds for superpositions of a sigmoidal function,” *IEEE Transactions on Information theory*, vol. 39, no. 3, pp. 930–945, 1993.

43. P. Kidger and T. Lyons, “Universal approximation with deep narrow networks,” in *Conference on learning theory*, pp. 2306–2327, PMLR, 2020.

44. 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, 2013.

45. A. Lewkowycz and G. Gur-Ari, “On the training dynamics of deep networks with l_2 regularization,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 4790–4799, 2020.

46. D. Wu and J. Xu, “On the optimal weighted l_2 regularization in overparameterized linear regression,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 10112–10123, 2020.

47. B. Sengupta, K. J. Friston, and W. D. Penny, “Efficient gradient computation for dynamical models,” *Neuroimage*, vol. 98, pp. 521–7, 2014.

48. 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, 2013.

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

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

51. 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.

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

53. 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.

54. S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, "A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems," *Automatica*, vol. 49, no. 1, pp. 82–92, 2013.

55. B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

56. B. A. Wallace and J. Si, "Continuous-time reinforcement learning control: A review

of theoretical results, insights on performance, and needs for new designs," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

57. R. Hu and M. Lauriere, "Recent developments in machine learning methods for stochastic control and games," *arXiv preprint arXiv:2303.10257*, 2023.

58. M. Lechner, Žikelić, K. Chatterjee, and T. A. Henzinger, "Stability verification in stochastic control systems via neural network supermartingales," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 7326–7336, 2022.

Acknowledgements

먼저, 본 석사 학위 논문을 작성하는 데 아낌없는 의견 교류와 조언을 해주신 최경환 지도 교수님께 감사의 말씀 드립니다. 교수님께서 석사과정 학생인 저에게 제공한 과분한 연구적 자유와 지원이 없었다면 여기까지 제 생각을 확장할 수 없었을 것입니다. 아울러 모자란 학위 논문을 심사하여 주신 최경환 교수님을 비롯한 안효성 교수님과 허필원 교수님께 감사드립니다. 교수님들께서 지적하여 주신 부분을 보완하여 완성도를 높일 수 있었습니다.

또한, 이 논문은 부모님과 하나뿐인 형제의 무한한 사랑과 무언의 기다림, 지지가 없었다면 존재하지 않았을 것입니다. 단거리의 미래도 예측할 수 없는 인생에서 언제나 돌아갈 수 있는 든든하고 따뜻한 가족이 있다는 것은 큰 행운이라고 생각합니다. 그렇기에, 제 생각을 부족함 없이 써 내려간 이 논문의 마지막 한 장의 한 문단을 빌려, 그동안 잘 표현하지 못한 부모님과 형제에게 감사의 말씀을 전합니다.

그리고, 옆에서 함께 고민하고 논의하여 익숙지 않은 주제와 관점을 제공해 주고 정신적 버팀목이 되어준 연구실 동료들에게 감사드립니다. 특별히, 연구 접근에 대하여 여러 생각을 하게 해준 승훈이 형, 논문 작업을 도와준 동화 형과 지윤이, 언제나 편하게 대할 수 있는 재준이, 그리고 마음속 깊은 의견을 나눌 수 있는 민석이에게 감사드립니다. 마지막으로, 드넓은 학술의 우주에서 아주 조그마한 탐색을 할 수 있도록 그 토대와 동기를 마련해준 선대 연구자들에게 감사드립니다. 그 외, 언급되지 않은 이 논문을 작성하는 데 도움을 주신 모든 분에게 감사드립니다.