DEEP AND ACCELERATED LEARNING IN ADAPTIVE CONTROL: A LYAPUNOV-BASED APPROACH

By

DUC MINH LE

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2022

To my father, Duoc (1961-2015), my mother, Linh, and my brothers Duy and Minh

ACKNOWLEDGMENTS

I thank my advisor, Dr. Warren Dixon, for his mentorship during my PhD career. During this journey, Dr. Dixon has been a role model that has shown me what it means to love your profession, persevere, and strive for continual growth by challenging yourself day in and day out. I also extend my gratitude to my committee members, Dr. Matthew Hale, Dr. Yu Wang, and Dr. John Shea for their recommendations and insights. I thank my colleagues in the Nonlinear Control and Robotics Lab, colleagues at the University of Florida, and research collaborators, who have challenged and encouraged me throughout my career. Finally, thank you to my friends and family for your relentless support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

a.e.	Almost Everywhere
CL	Concurrent Learning
DNN	Deep Neural Network
FE	Finite Excitation
ICL	Integral Concurrent Learning
LIP	Linear-in-the-parameters
NN	Neural Network
PE	Persistent Excitation
ReLu	Rectified Linear Unit
ResNet	Residual Neural Network
RMS	Root Mean Square
RMSE	Root Mean Square Error
SD	Standard Deviation

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

DEEP AND ACCELERATED LEARNING IN ADAPTIVE CONTROL: A LYAPUNOV-BASED APPROACH

By

Duc Minh Le

December 2022

Chair: Warren E. Dixon
Major: Mechanical Engineering

Adaptive control has become a prevalent technique used to achieve a control objective, such as trajectory tracking, in nonlinear systems subject to model uncertainties. Typically, an adaptive feedforward term is developed to compensate for model uncertainties, and closed-loop adaptation laws are developed to adjust the feedforward term in real-time. However, there are limitations in performance as adaptive control results typically achieve asymptotic convergence rates. Hence there is motivation for adaptation designs with faster learning capabilities such as accelerated learning methods. Accelerated gradient-based optimization methods have gained significant interest due to their improved transient performance and faster convergence rates. Accelerated gradient-based methods are discrete-time algorithms that alter their search direction by using a weighted sum from the previous iteration to add a momentum-based term and accelerate convergence. Recent results make connections between discrete-time accelerated gradient methods and continuous-time analogues. These connections lead to new insights on algorithm design based accelerated gradient methods. This dissertation aims to develop new adaptive control designs based on accelerated gradient methods using Lyapunov-based methods.

Chapter 2 provides a new data-driven adaption design based on the accelerated gradient method. Accelerated gradient methods such as Nesterov's accelerated gradient in numerical optimization have been shown to yield faster convergence than standard

gradient methods. However, these results either assume available measurements of the regression error or do not guarantee convergence of the parameter estimation error unless the restrictive persistence of excitation condition is satisfied. In this chapter, a new integral concurrent learning (ICL)-based accelerated gradient adaptive update law is developed to achieve trajectory tracking and real-time parameter identification for general uncertain Euler-Lagrange systems. The accelerated gradient adaptation is a higher-order scheme composed of two coupled adaptation laws. A Lyapunov-based method is used to guarantee the closed-loop error system yields global exponential stability under a less restrictive finite excitation condition. A comparative simulation study is performed on a two-link robot manipulator to demonstrate the efficacy of the developed method. Results show the higher-order scheme outperforms standard and ICL-based adaption by 19.6% and 11.1%, respectively, in terms of the root mean squared parameter estimation errors.

Chapter 3 develops a NN-based adaptation law based on accelerated gradient methods. In Chapter 2, the system's uncertainties were assumed to be linear-in-the-parameters (LIP) and was used to facilitate the analysis. Instead, in this chapter, a NN is used to approximate non-LIP system uncertainties to a prescribed function approximation error. This chapter develops a new NN-based accelerated gradient adaptive controller to achieve trajectory tracking in general nonlinear systems subject to unstructured uncertainties. Higher-order accelerated gradient-based adaptation laws are developed to generate real-time estimates of the NN weights. Due to the inherent nonlinear nested parametrization of NNs and coupling from higher-order adaptation, mathematical challenges are introduced by the injection of NN-based cross-terms in the analysis. A nonsmooth Lyapunov-based method is used to guarantee the closed-loop error system achieves global asymptotic tracking. Simulations are conducted to demonstrate the improved performance from the developed method. Results show the

higher-order adaptation outperforms the standard gradient-based NN adaptation by 32.3% in terms of the root mean squared function approximation error.

Chapter 4 provides a real-time deep neural network (DNN) adaptive control architecture for uncertain control-affine nonlinear systems to track a time-varying desired trajectory. A nonsmooth Lyapunov-based analysis is used to develop adaptation laws for the output-layer weights and develop constraints for inner-layer weight adaptation laws. The developed adaptation laws include a switching mechanism that enables the weight updates for each layer to be arbitrarily switched on and off. The inclusion of the switching mechanism enables the user to freely select when to update each layer weight based on factors such as computational resources. Unlike existing works with DNN-based control, the developed method establishes a framework to simultaneously update the weights of multiple layers for a DNN of arbitrary depth in real-time. The real-time controller and weight update laws enable the system to track a time-varying trajectory while compensating for unknown drift dynamics and parametric DNN uncertainties. A nonsmooth Lyapunov-based analysis is used to guarantee semi-global asymptotic tracking.

Chapter 5 leverages the insights developed from Chapter 4 on the development and analysis of DNN adaptation to build upon the developments of Chapter 3 and develop a new DNN-based adaptation law based on the accelerated gradient methods. To compensate for non-LIP uncertainties, the results in Chapter 3 developed a NN-based accelerated gradient adaptive controller to achieve trajectory tracking for nonlinear systems; however, the development and analysis only considered single hidden layer NNs. In Chapter 5, a generalized DNN architecture with an arbitrary number of hidden layers is considered, and a new DNN-based accelerated gradient adaptation scheme is developed to generate estimates of all the DNN weights in real-time. A nonsmooth Lyapunov-based analysis is used to guarantee the developed accelerated gradient-based DNN adaptation design achieves global asymptotic tracking error convergence for

general nonlinear control affine systems subject to unknown (non-LIP) drift dynamics. Simulations are conducted to demonstrate the improved performance from the developed method. Results show the developed accelerated gradient-based DNN adaptation outperforms gradient-based DNN adaptation by 67.41% and 78.82% in terms of the root mean squared tracking and function approximation errors, respectively.

Chapter 6 concludes the dissertation by highlighting the contributions of the developments from each chapter. Additionally, future works are discussed in this chapter.

CHAPTER 1
INTRODUCTION

1.1 Background

Adaptive control has become a prevalent technique used to achieve a control objective, such as trajectory tracking, in nonlinear systems subject to model uncertainties [1–3]. An adaptive feedforward model is typically developed to compensate for system uncertainties, and closed-loop adaptation laws are used to update the feedforward model in real-time. However, there are limitations in performance as adaptive control results typically achieve asymptotic convergence rates. Hence there is motivation for adaptation designs with faster learning capabilities such as accelerated learning methods. Accelerated gradient-based optimization methods have gained significant interest due to their improved transient performance and faster convergence rates; however, these methods have traditionally been used to design discrete-time update laws. Accelerated gradient-based methods are discrete-time algorithms that alter their search direction by using a weighted sum from the previous iteration to add a momentum-based term and accelerate convergence. Recent results make connections between discrete-time accelerated gradient methods and continuous-time analogues. These connections lead to new insights on algorithm design based accelerated gradient methods. This dissertation focuses on the development and analysis of accelerated gradient-based adaptive control designs for general uncertain nonlinear systems.

Classical adaptive methods typically achieve a control objective, such as trajectory tracking, but the parameter estimates may not converge. To guarantee parameter estimates converge to the true parameters, persistence of excitation (PE) is often required [2–4]. The PE condition is a restrictive assumption that requires sufficient excitation of the system for all time, and the condition is generally not verifiable online for a nonlinear system. To eliminate the restrictive PE condition and guarantee convergence of the parameter estimation errors, less restrictive initial excitation conditions

have been developed in [5–8]. Similarly, results such as [9] and [10] exploit a finite excitation (FE) condition that requires the system to be initially excited for a finite amount of time. Moreover, the FE condition can be verified online by examining the minimum singular value of a function of the regressor matrix. Results in [9] and [10] developed a data-driven method called concurrent learning (CL) that is adopted in this dissertation. Concurrent to real-time execution, sampled input-output data of the system is collected and stored. These CL methods require state derivative measurements that are generally not available and are required to be generated numerically or estimated as in [11]. To eliminate the dependence on unmeasurable state derivatives, results in [12] developed an integral form of the CL method (ICL). Before the FE condition is met, the parameter estimates may not approach the true parameters. Moreover, the parameter estimates may exhibit poor transient performance characterized by oscillatory behavior and slow convergence [13, 14].

Similarly, first-order methods in numerical optimization algorithms are known to yield poor transient performance [15].[1] To improve transient performance in parameter estimates in an optimization setting, a higher-order accelerated gradient method was developed by Nesterov in the seminal work in [16]. Higher-order accelerated gradient-based optimization methods, such as Nesterov's method [16] and the Heavy-ball method [17], have gained significant interest due to the improved transient performance and faster convergence rates compared to first-order methods [15]. The results in [18–21] make connections between discrete-time accelerated gradient methods and continuous-time analogues. Motivated by the improved convergence properties of

[1] First-order optimization methods can be characterized by a recursive difference equation driven by a cost function's gradient. In higher-order accelerated methods, a weighted sum from the previous iteration is used to add a momentum-based term and accelerate convergence.

accelerated gradient methods, insights from [18–21] lead to the design and analysis of accelerated gradient algorithms using Lyapunov-based methods [22–25].

The results in [22] develop an accelerated gradient-based adaptation scheme to estimate a system's parametric uncertainty in applications for model-reference adaptive control with linear dynamics. However, the result in [22] only guarantees asymptotic convergence in the tracking error while the parameter estimation errors are only guaranteed to remain bounded. Results in [23] develop a data-driven accelerated gradient-based method to achieve convergence in the parameter estimates, but this method is only applicable to parameter estimation in linear regression models where a tracking objective is not considered. Moreover, the result in [23] requires measurements of the regression error which is typically not measurable in the context of adaptive control due to the need for higher-order state derivatives. The results in [24] and [25] generalize the accelerated gradient-based adaptation to uncertain nonlinear systems. Results in [24], apply the variational framework in [19] to develop accelerated gradient-based adaptation for general nonlinear systems. Results in [24] also exploit connections between adaptive control and optimization, which provide insight on adaptation design, by comparing gradient- and mirror descent-based adaptation with and without the accelerated gradient method. However, similar to [22], the results in [24] only achieve asymptotic tracking error convergence. Moreover, it is not clear whether the tracking error convergence is uniform or exponential.

In Chapter 2, and in [25], a new higher-order ICL-based control and adaptive update law is developed for a general class of uncertain Euler-Lagrange systems. The uncertainties in this chapter are assumed to be structured uncertainties, i.e., the linear-in-the-parameters (LIP) assumption is satisfied. A torque-filtering method is used to reconstruct a measurable form of the regression error. The torque-filtering method eliminates the dependency on unknown state derivatives that are required to compute and store sampled data for ICL. The higher-order adaptation laws are designed as

two coupled first-order differential equations. Unlike previous ICL results, the coupling introduces ICL cross-terms in the stability analysis that challenges the ability to obtain parameter convergence. To compensate for the complexities of higher-order adaptation, auxiliary estimates and regression errors are introduced into the control input and adaptation laws. A Lyapunov-based stability analysis guarantees global asymptotic tracking. Furthermore, if a FE condition is satisfied, the developed higher-order ICL-based adaptive scheme yields global exponential convergence of the tracking error and parameter estimation errors. To demonstrate the efficacy of the developed method, comparative simulations were performed on a two-link robot manipulator. Results show the higher-order scheme outperforms standard and ICL-based adaption by 19.6% and 11.1%, respectively, in terms of the root mean squared parameter estimation errors.

Although results in [22–25] show improved performance from the accelerated-gradient based adaptation, the underlying assumption throughout the aforementioned results require the system's uncertainty to satisfy the LIP assumption. Hence, scenarios in which a system's dynamics exhibits unstructured or unknown (non-LIP) uncertainties motivates the use of neural network (NN)-based adaptive control techniques. NNs are universal function approximators that are capable of approximating continuous functions to a prescribed accuracy [26–28]. Alternative to real-time adaptation methods, numerical optimization-based methods have been used to train NNs by generating estimates of the NN weights such that a cost function is minimized over a training data set [29]. Although NN training algorithms provide a method to estimate NN weights, training is typically performed offline and then employed with open-loop control, which lacks stability guarantees. Moreover, the offline training often requires an extensive data set and significant computational resources which may require a long duration of time to train.

Due to the function approximation capabilities of NNs, NN-based adaptive control is a useful technique used to achieve control objectives (e.g., trajectory tracking)

in nonlinear systems subject to non-LIP uncertainties [30]. Results such as [31–35] approximate model uncertainties with a closed-loop adaptive NN feedforward component. In these results, Lyapunov-based methods are used to develop real-time adaptation laws to estimate the weights of a NN while also providing stability guarantees. However, the NN adaptation schemes in the aforementioned results are gradient-based, which, as previously mentioned in [13] and [14], may exhibit poor transient performance with oscillations and slow convergence in the weight estimates.

Motivated by the improved convergence properties of accelerated gradient methods, Chapter 3 and [36] develop a new higher-order NN-based adaptive controller for trajectory tracking in nonlinear systems subject to unstructured or unknown (non-LIP) uncertainties. A NN model is used to approximate the uncertainties in the system. An accelerated gradient approach is used to develop higher-order adaptation laws for real-time estimation of the weights of a NN. The higher-order adaptation scheme is structured as two coupled first-order differential equations; the first adaptation law is used to generate auxiliary estimates of the NN weights, and these estimates are coupled to the second adaptation law which generates the true estimate of the NN weights. Consequently, the coupled structure of the developed higher-order adaptation laws impose some mathematical challenges in facilitating a Lyapunov-based stability analysis. To compensate for perturbing effects from cross-terms introduced from higher-order adaptation, auxiliary estimates are designed into the control input. Additionally, the switching analysis in [25] is adopted in this chapter to allow for activation functions with discontinuous gradients, e.g., rectified linear unit (ReLU) activation functions. A nonsmooth Lyapunov-based analysis is used to guarantee the tracking errors achieve global asymptotic tracking. Comparative simulations are conducted to demonstrate the improved performance from the developed method. Results show the higher-order adaptation outperforms the standard gradient-based NN adaptation by 32.3% in terms of the root mean squared function approximation error.

Conventional NNs (i.e., NNs with only a single hidden layer) can approximate functions to a prescribed accuracy [37] and [27]; however, recent evidence indicates deep neural networks (DNNs) exploit more complex learning features that can potentially improve function approximation performance [38]. Although DNNs may potentially approximate the nonlinear dynamics of a system more accurately, it is difficult to derive real-time adaptation laws for DNNs with multiple layers because the uncertain ideal weights are nested within a collection of nonlinear activation functions.

Results in [33, 39–41] leverage Lyapunov-based analysis to develop multi-timescale DNN-based controllers containing real-time and offline iterative learning components. The output-layer weights of the DNN are adjusted online (i.e., in real-time) using NN-based adaptive control techniques. Concurrent to real-time execution, data is collected and DNN training algorithms, such as gradient descent and stochastic gradient descent (see [39–41] and [29, Ch. 8]), are used to iteratively update the inner-layer DNN weights. Since DNN learning algorithms are performed iteratively, the inner-layer weights are not updated continuously in real-time. The benefit of iterative learning is that the system performance improves with the quality of the DNN estimate. However, improving the quality of the DNN estimate may require a large training data set to capture the nonlinearities of the dynamics and significant computational resources to adjust the inner-layer weights. The DNN-based methods in [33, 39–41] also raise questions regarding the inner-layer weights updates such as: when to collect data, what is the most efficient way to retrain the inner-layer weights, when should the inner-layer weights be updated in the implemented adaptation law, etc.

While such open questions are topics for further investigation, Chapter 4 and [34] investigate general characteristics and structures of inner-layer adaptation laws. Specifically, this chapter develops general constraints on the inner-layer adaptation laws to update the inner-layer weight estimates in real-time. A Lyapunov-based analysis is used to develop a continuous adaptation law to estimate the output-layer weights. However,

unlike previous methods, this chapter provides a first insight into the development of Lyapunov-based adaptive update laws for both the inner-layer DNN weights as well as the output-layer weights. Inspired by modular adaptive control designs in [42–45], general constraints on the inner-layer DNN weight update laws are developed that enable modular design and selection of update laws. The developed DNN-based modular adaptive architecture allows more flexibility when selecting inner-layer DNN weight update laws.

In arbitrary width and depth DNNs, there may be hundreds or thousands of inner-layer weights. Simultaneously updating all the inner-layer weights online may be computationally intractable in real-time or undesired. Hence, the developed method provides a switched framework that provides design guidelines that can be used in future research efforts to guide inner-layer weight adaptive update laws. In doing so, the inner-layer weight update laws may be arbitrarily switched on and off to allocate computational resources while updating the desired weights. Additionally, inner-layer weights may dropout, or be selectively turned off to prevent over-fitting and improve overall function approximation performance [46].

Results such as [32], [33], and [47] develop a robust sliding mode method to achieve asymptotic tracking with NN feedforward controllers. Like the aforementioned results, the developed method uses a sliding mode control term to yield asymptotic tracking in the presence of the NN reconstruction error, but also uses switched adaptation laws for the inner-layer weights. Hence, this chapter leverages a nonsmooth Lyapunov-like analysis [48] to guarantee asymptotic tracking of a desired trajectory. Unlike existing works, the developed modular adaptive architecture incorporates the inner-layer weights of an arbitrarily deep DNN into a Lyapunov-based analysis to develop and characterize a general class of suitable inner-layer DNN adaptation laws. Moreover, sufficient conditions are developed to guarantee the tracking objective is

achieved, despite the arbitrary number of inner-layers and switched update laws. Comparative numerical simulation results are included to demonstrate the efficacy of the developed method and show the benefits of real-time all-layer DNN weight adaptation.

As previously mentioned, DNNs are universal function approximators that are capable of approximating continuous functions to a prescribed accuracy [26–28]. Typically, numerical optimization-based methods have been used to train DNNs by generating estimates of the DNN weights such that a cost function is minimized over a training data set [29]. Although DNN training algorithms provide a method to estimate DNN weights, training is typically performed offline and then held constant during implementation. The resulting controller uses the DNN as a fixed approximate model, with no guarantees on how close the model approximates the actual model experienced during real-time implementation nor what effects such a model mismatch may have on the stability of the controller. Moreover, such offline training often requires a large data set, which could be expensive or not possible to obtain.

Recent results in [33–35, 39, 40] develop the first DNN-based adaptive controllers that enable continuous learning based on weight adaptation laws that are derived from a Lyapunov-based stability analysis. In [33, 39, 40], Lyapunov-based adaptation laws are developed to adjust the output layer of a fully-connected DNN in real-time while the inner layers are updated discretely using data-driven offline training algorithms. More recent results in [34, 35, 49] develop the first Lyapunov-based adaptation laws that can be used to enable continuous learning by all the weights of a DNN in real-time. As in Chapter 4, the results in [34] use a modular approach to develop constraints on the DNN weight adaptation laws, but this approach lacks constructive insights for the adaptive update law. Results in [35] provide the first insights on Lyapunov-derived weight adaptation design for fully-connected DNNs which is then generalized to residual NNs (ResNets) in [49]. The adaptation laws in [33–35, 39, 40, 49] use gradient-based adaptation laws that are designed to cancel cross-terms resulting from a first-order Taylor's series

approximation. However, as mentioned previously, gradient-based adaptation laws may exhibit poor transient performance with oscillations and slow convergence in the weight estimates.

In Chapter 3 and [36], the accelerated gradient strategy is used to develop a NN-based adaptive controller to compensate for non-LIP model uncertainties. Specifically, an accelerated gradient-based adaptation law is developed to estimate NN weights in real-time. However, the NN model considered in Chapter 3 is limited to NNs with only a single hidden layer. Chapter 5 leverages the insights on the development and analysis of DNN-based adaptation via Lyapunov-based techniques in Chapter 4 to extend the accelerated gradient-based NN adaptive control scheme in Chapter 3. Specifically, Chapter 5 generalizes the NN model to deep architectures with an arbitrary number of hidden layers. In comparison to Chapter 5, Chapter 3 generalizes the mathematical development, control design, and stability analysis to account for DNN architectures and unknown exogenous disturbances in the system dynamics. Additionally, new comparative simulations are conducted to demonstrate the improved performance from the developed method in comparison to a baseline DNN adaptive controller. Results show the developed accelerated gradient-based DNN adaptation outperforms gradient-based DNN adaptation by 67.41% and 78.82% in terms of the root mean squared tracking and function approximation errors, respectively.

In Chapter 5, a new DNN-based accelerated gradient adaptive controller is developed for trajectory tracking for general nonlinear control-affine systems subject to non-LIP uncertainties. The constructive variational framework in [19] is used to design higher-order accelerated gradient-based adaptation laws for real-time estimation of all the weights of the DNN. The higher-order adaptation scheme is structured as two coupled first-order differential equations. Despite the potential improvements from the accelerated gradient strategy, the inherent coupled structure poses mathematical challenges with Lyapunov-based analysis. However, the perturbing effects resulting from the

DNN-based cross-terms injected into the analysis by the accelerated gradient-based adaptation strategy are compensated for by the developed control input design. Additionally, Lyapunov-based analyses of the DNN-based adaptation poses challenges due to the nested nonlinear parametrization of DNN architectures. However, a recursive relation of the DNN architecture, similar to [35], is developed to facilitate the analysis and derivation of a general expression for the adaptation design that accounts for general fully-connected DNNs with an arbitrary number of hidden layers. In comparison to [35], Chapter 5 restructures the analysis and control development to facilitate the derivation of a general expression to compute the DNN adaptation law. An example is also provided to show the utility of the generalized adaptation law. A nonsmooth Lyapunov-based analysis is used to analyze the new accelerated gradient-based DNN adaptive control design. The tracking errors are guaranteed to achieve global asymptotic tracking error convergence despite the presence of non-LIP system uncertainties and exogenous disturbances.

1.2 Notation

Mathematical preliminaries and notation used throughout the dissertation is summarized in this section. Let $\mathbb{R}$ and $\mathbb{Z}$ denote the set of real numbers and integers, respectively. Let $\mathbb{R}_{\geq 0} \triangleq [0, \infty)$ and $\mathbb{R}_{>0} \triangleq (0, \infty)$ denote the set of positive and strictly positive real numbers, respectively. Similarly, let $\mathbb{Z}_{\geq 0}$ and $\mathbb{Z}_{>0}$ denote the set of positive and strictly positive integers, respectively. The Euclidean norm of a vector $x \in \mathbb{R}^n$ is denoted by $\|x\| \triangleq \sqrt{x^T x}$. The vector space of essentially bounded Lebesgue measurable functions is denoted by $\mathcal{L}_\infty$. For $n, m \in \mathbb{Z}_{>0}$, let $\mathbb{R}^{n\times m}$ and I_n denote the space of $n \times m$ dimensional matrices and the $n \times n$ dimensional identity matrix, respectively. Let $0_{n\times m}$ denote an $n \times m$ matrix of zeros. The right-to-left matrix product operator is denoted by $\overleftarrow{\prod}$, i.e., given suitable matrices A_i for all $i = 1, \ldots, m$, $\overleftarrow{\prod_{i=1}^{m}} A_i \triangleq A_m \ldots A_2 A_1$ and $\overleftarrow{\prod_{i=a}^{m}} A_i \triangleq I$ if $a > m$. Given matrices $A \in \mathbb{R}^{p\times q}$ and $B \in \mathbb{R}^{r\times s}$, the

Kronecker product is denoted by $A \otimes B \in \mathbb{R}^{pr \times qs}$. Let vec $(\cdot)$ denote the vectorization operator that transforms a matrix into a column vector, i.e., for a matrix $A = [a_{i,j}] \in \mathbb{R}^{n \times m}$, vec $(A) \triangleq [a_{1,1}, \ldots, a_{1,m}, \ldots a_{n,1}, \ldots, a_{n,m}]^T \in \mathbb{R}^{nm}$. Given a square matrix $A \in \mathbb{R}^{n \times n}$, the trace of A is denoted by tr (A). The Frobenius norm of a matrix $A \in \mathbb{R}^{n \times m}$ is denoted by $\|A\|_F = \sqrt{\text{tr}\,(A^T A)}$. The minimum and maximum eigenvalue of a matrix $A \in \mathbb{R}^{n \times n}$ is denoted by $\lambda_{min}(A) \in \mathbb{R}$ and $\lambda_{max}(A) \in \mathbb{R}$, respectively. Given matrices $A \in \mathbb{R}^{k \times l}$, $B \in \mathbb{R}^{l \times m}$, and $C \in \mathbb{R}^{m \times n}$, the vectorization operator satisfies the following property [50, Proposition 7.1.9]

$$\text{vec}\,(ABC) = \left(C^T \otimes A\right) \text{vec}\,(B)\,. \tag{1–1}$$

CHAPTER 2
INTEGRAL CONCURRENT LEARNING-BASED ACCELERATED GRADIENT ADAPTIVE CONTROL OF UNCERTAIN EULER-LAGRANGE SYSTEMS

This chapter considers a trajectory tracking problem for general uncertain Euler-Lagrange system where the model uncertainties are assumed to satisfy the LIP assumption that is standard in adaptive control. Leveraging recent insights on continuous-time analogues of accelerated gradient methods, this chapter develops a data-driven ICL-based accelerated gradient adaptive update law to achieve trajectory tracking and real-time parameter identification for general Euler-Lagrange systems subject to LIP uncertainties. The accelerated gradient adaptation is a higher-order scheme composed of two coupled adaptation laws. Other similar results either assume available measurements of the regression error or do not guarantee convergence of the parameter estimation error unless the restrictive PE condition is satisfied. However, the results in this chapter guarantees parameter convergence under a less restrictive FE condition. Additionally, the developments in this chapter use a torque filtering method to reconstruct a measurable form of the parameter estimation error. A Lyapunov-based method is used to guarantee the closed-loop error system yields global exponential stability under a less restrictive finite excitation condition. A comparative simulation study is performed on a two-link robot manipulator to demonstrate the efficacy of the developed method. Results show the higher-order scheme outperforms standard and ICL-based adaption by 19.6% and 11.1%, respectively, in terms of the root mean squared parameter estimation errors.

2.1 Problem Formulation

2.1.1 Dynamic Model

Consider a general uncertain nonlinear Euler-Lagrange system modeled as [51, Ch. 2]

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F\dot{q} = \tau, \tag{2–1}$$

where $q, \dot{q}, \ddot{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized position, velocity, and acceleration, respectively, $M : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ denotes a generalized inertia matrix, $V_m : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times n}$ denotes a generalized centripetal-Coriolis matrix, $G : \mathbb{R}^n \to \mathbb{R}^n$ represents a generalized vector of potential forces, $F \in \mathbb{R}^{n \times n}$ represents generalized dissipation effects, and $\tau : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the control input. The subsequent development is based on the assumption that q and $\dot{q}$ are measurable. The dynamic model in (2–1) satisfies the following properties and assumption [51, Ch. 2].

Property 1. The inertia matrix $M(q)$ for all $q \in \mathbb{R}^n$ is positive-definite and satisfies $m_1 \|\xi\|^2 \leq \xi^T M(q) \xi \leq m_2 \|\xi\|^2$ for all $\xi \in \mathbb{R}^n$, where $m_1, m_2 \in \mathbb{R}_{>0}$ denote known constants.

Property 2. The inertia and centripetal-Coriolis matrices satisfy the following skew-symmetric relation $\xi^T \left(\dot{M}(q) - 2V_m(q, \dot{q}) \right) \xi = 0$ for all $q, \dot{q}, \xi \in \mathbb{R}^n$.

Assumption 2.1. The uncertain dynamics in (2–1) are linear-in-the-parameters and can be expressed as

$$M(q)\ddot{q} + V_m(q, \dot{q})\dot{q} + G(q) + F\dot{q} = \Psi(q, \dot{q}, \ddot{q})\theta^*, \tag{2–2}$$

for all $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$, where $\Psi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ denotes a known regressor matrix, and $\theta^* \in \mathbb{R}^m$ denotes a vector of unknown constant parameters.

2.1.2 Control Objective

The control objective is to track a user-defined desired trajectory $q_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite parametric uncertainties in the dynamic model. The desired trajectory and its first two time derivatives are assumed to be continuous and bounded, i.e., $q_d, \dot{q}_d, \ddot{q}_d \in \mathcal{L}_\infty$ for all $t \in \mathbb{R}_{\geq 0}$. Note that the desired trajectory need not be persistently exciting and can be set to a constant for the regulation problem.

To quantify the tracking objective, the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and auxiliary tracking error $r : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are defined as

$$e \triangleq q_d - q, \tag{2–3}$$

$$r \triangleq \dot{e} + \alpha e, \tag{2–4}$$

respectively, where $\alpha \in \mathbb{R}_{\geq 0}$ denotes a user-defined parameter. Additionally, the control objective is to simultaneously perform real-time parameter estimation to compensate for the parametric model uncertainties in (2–1). To quantify the parameter identification objective, the parameter estimation error $\tilde{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ is defined as

$$\tilde{\theta} \triangleq \theta^* - \hat{\theta}, \tag{2–5}$$

where $\hat{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes the parameter estimate.

2.2 Control Development

2.2.1 Controller and Adaptive Update Law

This section introduces the controller design and higher-order ICL-based adaptive update laws. To facilitate the subsequent stability analysis, a measurable form of the parameter estimation error is constructed using the torque-filtering method in [4]. Let the filtered control input $\tau_f : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ be defined as

$$\tau_f \triangleq f * \tau, \tag{2–6}$$

where the filter $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is defined as $f \triangleq \beta \exp(-\beta t), \beta \in \mathbb{R}_{>0}$ denotes a user-defined parameter, and $*$ denotes the convolution operator. The filtered control input in (2–6) can be generated by $\dot{\tau}_f + \beta \tau_f = \beta \tau$ with $\tau_f(0) = 0_n$ [4, Ch. 8.7]. Using (2–1) and (2–2), the filtered control input in (2–6) can be expressed as

$$\tau_f = \Psi_f(q, \dot{q})\, \theta^*, \tag{2–7}$$

where the filtered regressor $\Psi_f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is defined as $\Psi_f \triangleq f * \Psi$. The regressor matrix $(q, \dot{q}, \ddot{q}) \mapsto \Psi(q, \dot{q}, \ddot{q})$ depends on the unmeasurable acceleration $\ddot{q}$. To obtain a measurable form of the filtered regressor Ψ_f that is independent of $\ddot{q}$, the regressor matrix can be re-written as $\Psi = \dot{h} + g$, where $(q, \dot{q}, \ddot{q}) \mapsto \dot{h}(q, \dot{q}, \ddot{q})$ and

$(q, \dot{q}) \mapsto g(q, \dot{q})$ are known functions. Then the filtered regressor can be expressed as

$$\Psi_f = f * \dot{h} + f * g. \tag{2–8}$$

To eliminate the dependence on $\ddot{q}$, the following property of convolutions is used [52, Eq. 6.6.7]

$$f * \dot{h} = \dot{f} * h + f(0) h - f h(0), \tag{2–9}$$

where the function $(q, \dot{q}) \mapsto h(q, \dot{q})$ is known and measurable. Let $\psi_{f,1} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ and $\psi_{f,2} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ be defined as $\psi_{f,1} \triangleq \dot{f} * h$ and $\psi_{f,2} \triangleq f * g$, respectively. Using (2–8) and (2–9), the measurable form of the filtered regressor is

$$\Psi_f = \psi_{f,1}(t) + \psi_{f,2}(t) + \beta h(t) - \beta \exp(-\beta t) h(0),$$

where $\psi_{f,1}$ can be generated by $\dot{\psi}_{f,1} + \beta \psi_{f,1} = -\beta^2 h$ with $\psi_{f,1}(0) = 0_{n \times m}$, and $\psi_{f,2}$ can be generated by $\dot{\psi}_{f,2} + \beta \psi_{f,2} = \beta g$ with $\psi_{f,2}(0) = 0_{n \times m}$.

Based on the subsequent stability analysis, the controller is designed as

$$\tau \triangleq kr + Y\hat{\theta} + e - 2Y\left(\hat{\theta} - \nu\right), \tag{2–10}$$

where $k \in \mathbb{R}_{>0}$ denotes a user-defined parameter, $\nu : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes an auxiliary parameter estimate, and Y denotes a known regressor matrix that is defined subsequently in the closed-loop error development.

Based on the subsequent stability analysis, the higher-order ICL-based adaptive update laws $\dot{\nu} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ and $\dot{\hat{\theta}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ are designed as

$$\dot{\nu} \triangleq \Gamma \left(Y^T r + k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \left(\tau_{f,i} - \Psi_{f,i}^T \nu \right) \right), \tag{2–11}$$

$$\dot{\hat{\theta}} \triangleq -\Gamma \left(k_2 \left(\hat{\theta} - \nu \right) - k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \left(\tau_{f,i} - \Psi_{f,i}^T \nu \right) \right), \tag{2–12}$$

where $k_1, k_2 \in \mathbb{R}_{>0}$ are user-defined parameters, and $\Gamma \in \mathbb{R}^{m\times m}$ denotes a user-defined positive definite learning gain matrix. In (2–11) and (2–12), the terms $\tau_{f,i} \triangleq \tau_f(t_i)$ and $\Psi_{f,i} \triangleq \Psi_f(t_i)$ for all $t_i \in \mathbb{R}_{\geq 0}$ are sampled data points at $N \in \mathbb{Z}_{\geq 0}$ discrete time instants that are collected concurrent to real-time execution.[1] The update laws in (2–11) and (2–12) are the implementable forms. Using (2–7), the adaptive update laws in (2–11) and (2–12) can also be expressed as

$$\dot{\nu} = \Gamma \left(Y^T r + k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i} \left(\theta^* - \nu \right) \right), \tag{2–13}$$

$$\dot{\hat{\theta}} = -\Gamma \left(k_2 \left(\hat{\theta} - \nu \right) - k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i} \left(\theta^* - \nu \right) \right), \tag{2–14}$$

respectively, which will be used in the subsequent stability analysis.[2] In (2–13) and (2–14), $\sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i}$ is the ICL term that contains the recorded input-output data generated by the dynamics.

2.2.2 Closed-Loop Error System

Taking the time derivative of the auxiliary tracking error in (2–4), pre-multiplying by $M(q)$, and substituting in (2–1), the open-loop error system can be expressed as

$$M(q)\dot{r} = Y(q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d)\theta^* - \tau - V_m(q, \dot{q})r, \tag{2–15}$$

where the known regressor $Y : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times m}$ and unknown parameters θ^* are defined based on the relation $Y\theta^* = M(q)(\ddot{q}_d + \alpha\dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \alpha e) + G(q) + F\dot{q}$. Substituting the controller in (2–10) into (2–15) yields the closed-loop error system

$$M\dot{r} = -kr + Y\tilde{\theta} - V_m r - e + 2Y\left(\hat{\theta} - \nu\right). \tag{2–16}$$

[1] See [12] for discussions on data collection and implementation of ICL.

[2] The update laws in (2–13) and (2–14) are used only for the subsequent stability analysis, whereas the update laws in (2–11) and (2–12) are used in implementation.

2.3 Stability Analysis

To facilitate the subsequent stability analysis, the following FE condition is assumed.[3]

Assumption 2.2. [12] There exists a time $T \in \mathbb{R}_{>0}$ such that $\lambda_{min}\left(\sum_{i=1}^{N} \Psi_{f,i}^{T}\Psi_{f,i}\right) \geq \gamma$, where $\gamma \in \mathbb{R}_{>0}$ is a user-defined parameter.

Before the FE condition is satisfied, the developed controller and adaptive update laws are shown to ensure the closed-loop error system remains bounded and achieves the tracking objective. To verify the FE condition is met, the minimum singular value of the sum of discrete regression matrices is checked to verify it is positive. After Assumption 2.2 is satisfied, the tracking and parameter estimation errors are shown to be contained within an exponentially converging envelope.

To facilitate the subsequent stability analysis, let $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n+2m}$ denote a concatenated state vector defined as $z \triangleq \left[e^T, r^T, (\theta^* - \nu)^T, \left(\hat{\theta} - \nu\right)^T\right]^T$. Consider the candidate Lyapunov function $V : \mathbb{R}^{2n+2m} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined as

$$V(z,t) \triangleq \frac{1}{2} r^T M r + \frac{1}{2} e^T e + \frac{1}{2} (\theta^* - \nu)^T \Gamma^{-1} (\theta^* - \nu) + \frac{1}{2} \left(\hat{\theta} - \nu\right)^T \Gamma^{-1} \left(\hat{\theta} - \nu\right), \quad \text{(2–17)}$$

which satisfies the inequality

$$c_1 \|z\|^2 \leq V(z,t) \leq c_2 \|z\|^2, \quad \text{(2–18)}$$

where the known constants $c_1, c_2 \in \mathbb{R}_{>0}$ are defined as $c_1 \triangleq \frac{1}{2}\min\{1, m_1, \lambda_{min}\{\Gamma^{-1}\}\}$ and $c_2 \triangleq \frac{1}{2}\max\{1, m_2, \lambda_{max}\{\Gamma^{-1}\}\}$, where m_1 and m_2 were defined in Property 1.

[3] To facilitate excitation in the system, a transient dither signal can be added to the desired trajectory. To satisfy the FE condition and guarantee the parameter estimation errors converge, the system must only be initially excited for a finite period of time. The FE condition is far less restrictive as the addition of a dither signal to the desired trajectory may be required for all time to satisfy the PE condition.

Theorem 1. *Consider a system modeled by the dynamics in (2–1) that satisfies Properties 1 and 2. Let Assumption 2.1 hold. The controller in (2–10) and higher-order adaptive update laws in (2–13) and (2–14) ensure the closed-loop error system in (2–16) yields global asymptotic tracking in the sense that* $\lim_{t\to\infty} \|e(t)\| = 0$, $\lim_{t\to\infty} \|r(t)\| = 0$, *and* $\lim_{t\to\infty} \left\|\hat{\theta}(t) - \nu(t)\right\| = 0$ *for all initial conditions* $z(0) \in \mathbb{R}^{2n+2m}$.

Proof. Taking the time derivative of (2–17), using (2–4) and (2–5), applying Property 2, and substituting the closed-loop error system in (2–16) and adaptive update laws in (2–13) and (2–14) yields

$$\dot{V} = -kr^T r - \alpha e^T e - k_2 \left(\hat{\theta} - \nu\right)^T \left(\hat{\theta} - \nu\right) - k_1 (\theta^* - \nu)^T \left(\sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i}\right) (\theta^* - \nu). \quad \text{(2–19)}$$

Since the term $\sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i}$ is positive semi-definite, (2–19) can be upper bounded as $\dot{V} \leq -kr^T r - \alpha e^T e - k_2 \left(\hat{\theta} - \nu\right)^T \left(\hat{\theta} - \nu\right)$. From (2–17) and the fact that $\dot{V} \leq 0$, $V \in \mathcal{L}_\infty$, which implies $z \in \mathcal{L}_\infty$, and hence, $e, r, \nu, \hat{\theta} \in \mathcal{L}_\infty$. Using (2–3)–(2–5) implies $q, \dot{q}, \hat{\theta} \in \mathcal{L}_\infty$, respectively. Using (2–4), the fact that $e, r \in \mathcal{L}_\infty$ implies $\dot{e} \in \mathcal{L}_\infty$. The fact that $q, \dot{q}, q_d, \dot{q}_d, \ddot{q}_d \in \mathcal{L}_\infty$ implies $Y \in \mathcal{L}_\infty$. Using (2–16), the fact that $e, r, Y, \hat{\theta}, \nu \in \mathcal{L}_\infty$ implies $\dot{r} \in \mathcal{L}_\infty$. Using (2–10), the fact that $e, r, Y, \hat{\theta}, \nu \in \mathcal{L}_\infty$ implies the controller $\tau \in \mathcal{L}_\infty$ is bounded. Using (2–11), the fact that $r, Y, \nu \in \mathcal{L}_\infty$ implies the auxiliary parameter estimate update law $\dot{\nu} \in \mathcal{L}_\infty$ is bounded. Using (2–12), the fact that $\hat{\theta}, \nu \in \mathcal{L}_\infty$ implies the parameter estimate update law $\dot{\hat{\theta}} \in \mathcal{L}_\infty$ is bounded. Since $e, r, \left(\hat{\theta} - \nu\right) \in \mathcal{L}_\infty$ and $\dot{e}, \dot{r}, \left(\dot{\hat{\theta}} - \dot{\nu}\right) \in \mathcal{L}_\infty$, then $e, r, \left(\hat{\theta} - \nu\right)$ are uniformly continuous. Moreover, using (2–19) implies $e, r, \left(\hat{\theta} - \nu\right) \in \mathcal{L}_2$. Then by invoking Barbalat's Lemma (i.e, Corollary A.7 in [1]), $\lim_{t\to\infty} \|e(t)\| = 0$, $\lim_{t\to\infty} \|r(t)\| = 0$, and $\lim_{t\to\infty} \left\|\hat{\theta}(t) - \nu(t)\right\| = 0$.

■

In Theorem 1, the developed controller and adaptive update laws are shown to ensure the closed-loop error system is bounded and achieves the tracking objective. By Assumption 2.2, there exists a finite period of time in which the FE condition is satisfied.

Then exponential convergence bounds on the tracking and parameter estimation errors can be established as in the following theorem.

Theorem 2. *Consider a system modeled by the dynamics in (2–1) that satisfies Properties 1 and 2. Let Assumptions 2.1 and 2.2 hold. The controller in (2–10) and higher-order adaptive update laws in (2–13) and (2–14) ensures the equilibrium point* $z = 0_{2n+2m}$ *of the closed-loop error system in (2–16) is globally exponentially stable.*

Proof. Following the proof for Theorem 1, we can obtain (2–19). By Assumption 2.2, the minimum eigenvalue of $\lambda_{min}\left(\sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i}\right) \geq \gamma$ is strictly positive for all $t \in [T, \infty)$. Then (2–19) can be upper bounded as $\dot{V} \leq -k \|r\|^2 - \alpha \|e\|^2 - k_1 \gamma \|\theta^* - \nu\|^2 - k_2 \left\|\hat{\theta} - \nu\right\|^2$ for all $t \in [T, \infty)$. Then using (2–17) yields

$$\dot{V} \leq -2\lambda V, \forall t \in [T, \infty),$$

where $\lambda \triangleq \frac{1}{2c_2} \min(k, \alpha, k_1\gamma, k_2)$ denotes a known constant. Invoking the Comparison Lemma in [53, Lemma 3.4] and using (2–18) yields

$$\|z(t)\| \leq \sqrt{\frac{c_2}{c_1}} \|z(T)\| \exp(\lambda T) \exp(-\lambda t), \forall t \in [T, \infty). \tag{2–20}$$

From (2–18) and the fact that $\dot{V} \leq 0, V \in \mathcal{L}_\infty$ implies $\|z(t)\| \leq \sqrt{\frac{c_2}{c_1}} \|z(0)\|$ for all $t \in \mathbb{R}_{\geq 0}$. Then upper bounding (2–20) yields the exponential convergence bound

$$\|z(t)\| \leq \frac{c_2}{c_1} \|z(0)\| \exp(\lambda T) \exp(-\lambda t), \tag{2–21}$$

for all $t \in \mathbb{R}_{\geq 0}$ and initial conditions $z(0) \in \mathbb{R}^{2n+2m}$. Hence, the equilibrium point $z = 0_{2n+2m}$ is globally exponentially stable.

■

2.4 Simulation

To demonstrate the performance of the developed method, a simulation study was conducted on a two-link planar revolute robot with $M : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$, $V_m : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$,

and $F \in \mathbb{R}^{2\times 2}$ modeled as [12]

$$M(q) = \begin{bmatrix} p_1 + 2p_3c_2, & p_2 + p_3c_2 \\ p_2 + p_3c_2, & p_2 \end{bmatrix}, \tag{2–22}$$

$$V_m(q,\dot{q}) = \begin{bmatrix} -p_3s_2\dot{q}_2, & -p_3s_2(\dot{q}_1 + \dot{q}_2) \\ p_3s_2\dot{q}_1, & 0 \end{bmatrix}, \tag{2–23}$$

$$F = \begin{bmatrix} f_1, & 0 \\ 0, & f_2 \end{bmatrix}. \tag{2–24}$$

In (2–22)–(2–24), the angular joint state $q : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ is defined as $q \triangleq [q_1, q_2]^T$, c_2 denotes $\cos(q_2)$, and s_2 denotes $\sin(q_2)$. The nominal parameters of the two-link robot model in (2–22)–(2–24) are $p_1 = 3.473$, $p_2 = 0.196$, $p_3 = 0.242$, $f_1 = 5.3$, and $f_2 = 1.1$.

Three simulation experiments were conducted. Each simulation was conducted for 30 s with initial conditions $q(0) = [1.22, -0.52]^T$ rad and $\dot{q}(0) = [0, 0]^T$ rad/sec. The desired trajectory $q_d(t) \triangleq [q_{d1}, q_{d2}]^T$ was selected as

$$q_d \triangleq \begin{bmatrix} \cos(0.5t) \\ 2\cos(t) \end{bmatrix}.$$

The first simulation was performed with a standard gradient-based adaptive design given by [52]

$$\begin{aligned} \tau &\triangleq kr + Y\hat{\theta} + e, \\ \dot{\hat{\theta}} &\triangleq \Gamma Y^T r. \end{aligned} \tag{2–25}$$

In the second simulation, the ICL adaptive design in [12] was used. The control input was the same as in (2–25), and the implementable form of the ICL adaptive update law was

$$\dot{\hat{\theta}} \triangleq \Gamma \left(Y^T r + k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \left(\tau_i - \Psi_{f,i}\hat{\theta} \right) \right), \tag{2–26}$$

Table 2-1. Simulation Parameters

Adaptation Law	Γ	k	k_1	k_2	β	γ
Standard Adaptive in [52]	$20I_{5\times5}$	-	-	-	-	-
ICL Adaptive in [12]	$20I_{5\times5}$	1	0.1	-	0.03	1
Developed Method	$20I_{5\times5}$	1	0.1	0.1	0.03	1

which yields the analytic form $\dot{\hat{\theta}} \triangleq \Gamma \left(Y^T r + k_1 \sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i} \tilde{\theta} \right)$. The developed higher-order ICL-based adaptive control design in (2–10)–(2–12) was used in the third simulation. In the second and third simulation, data was collected and stored during real-time execution until the FE condition was satisfied. The FE condition was verified online by checking if the minimum eigenvalue condition $\lambda_{min} \left(\sum_{i=1}^{N} \Psi_{f,i}^T \Psi_{f,i} \right) \geq \gamma$ was satisfied, where $\gamma = 1$. The ICL-based terms in the adaptation laws were not executed until the FE condition was satisfied. To provide a comprehensive comparison between the performance from each adaptive control scheme, the user-defined design constants were selected equally in each case where applicable. The parameters used in each case are shown in Table 2-1.

Figure 2-1 illustrates the evolution of the normalized parameter estimation error trajectories for each case. To compare the performance between the methods, the root mean square (RMS) of the parameter estimation error was calculated. The RMS parameter estimation error corresponding to the standard adaptive, ICL adaptive, and developed method were 3.12, 3.00, and 2.75, respectively. The developed method had the lowest RMS parameter estimation error and is 11.77% and 8.38% lower than the standard adaptive and ICL adaptive methods, respectively. The parameter estimates over the duration of the simulation with the standard adaptive method did not converge to the true parameters, as expected since only tracking error convergence is guaranteed. In both simulations for the ICL adaptive and developed method, the parameter estimates converged to the true parameters at approximately 10 s when the FE condition was satisfied. However, the developed method exhibited improved transient performance compared to the ICL adaptive method, which had high oscillatory

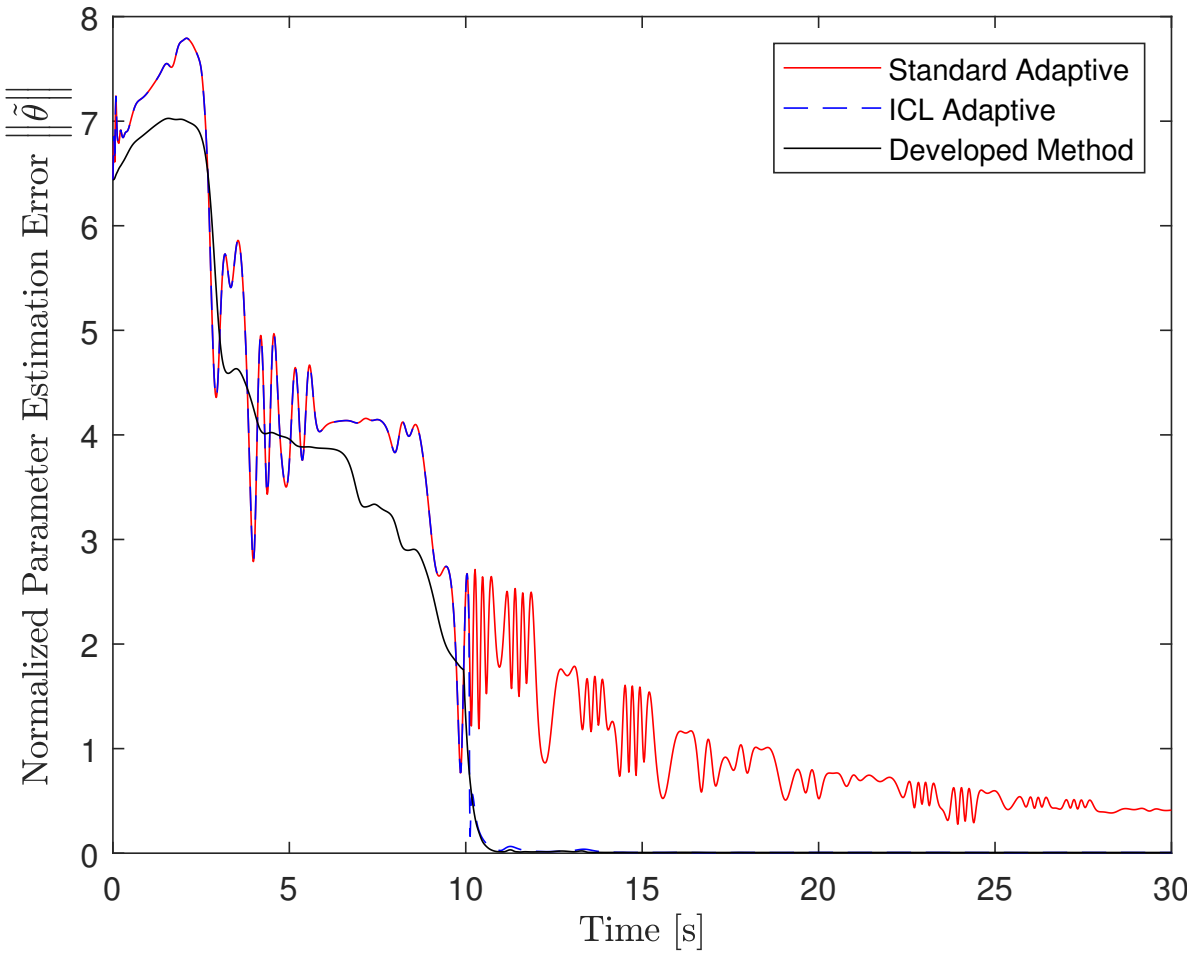

Figure 2-1. Evolution of the normalized parameter estimation error trajectories $\left\|\tilde{\theta}\right\|$ for each simulation. The red line represents the simulation using the standard adaptive method in (2–25). The blue line represents the simulation using the ICL adaptive method in (2–26). The black line represents the simulation using the developed method in (2–10)–(2–12).

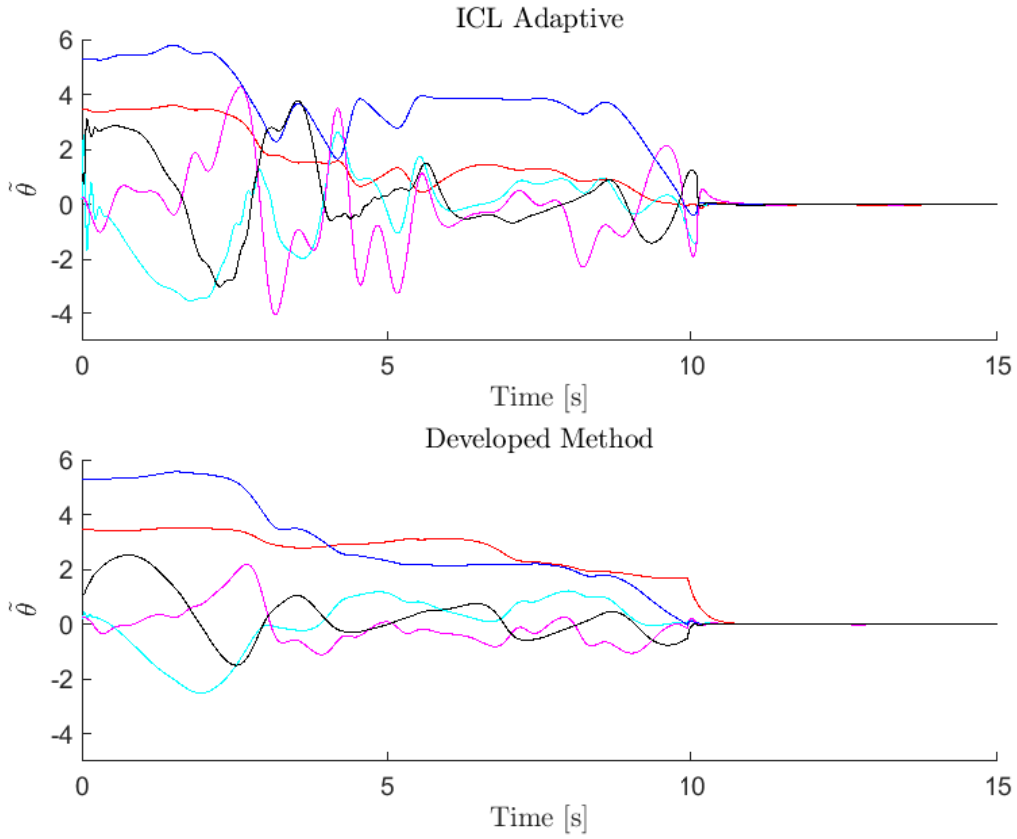

Figure 2-2. (top): Parameter estimation error $\tilde{\theta}$ using the ICL adaptive method. (bottom): Parameter estimation error using the developed method.

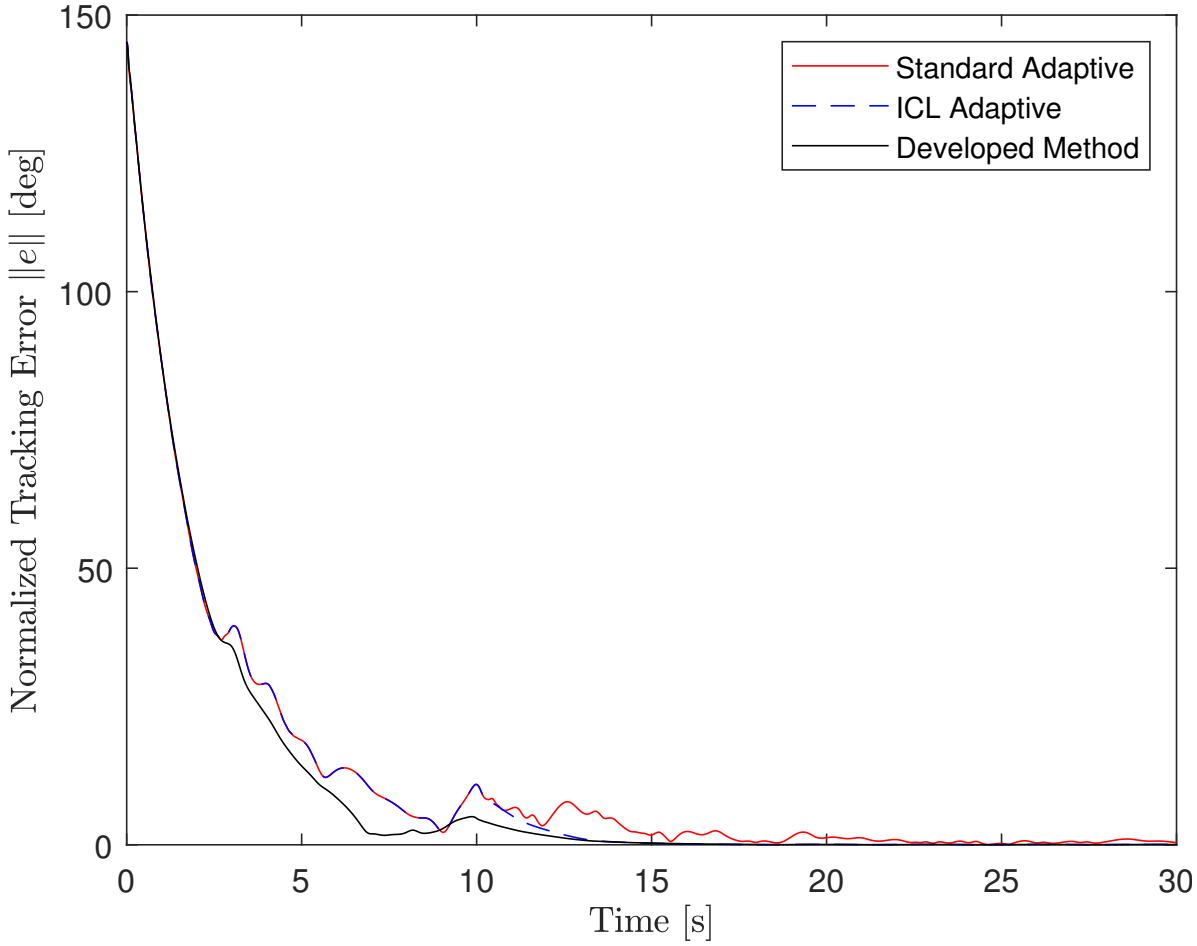

Figure 2-3. Evolution of the normalized tracking error trajectory $\|e\|$ for each simulation.

Table 2-2. RMS Errors

RMS	Standard Adaptive	ICL Adaptive	Developed Method
e [deg]	27.06	26.99	26.53
e_{0-10} [deg]	46.67	46.67	45.93
e_{ss} [deg]	3.07	1.99	1.03
$\tilde{\theta}$	3.12	3.00	2.75
$\tilde{\theta}_{0-10}$	5.19	5.19	4.76
$\tilde{\theta}_{ss}$	1.05	0.19	0.09

behavior. From 0–10 s, the RMS parameter estimation error for the ICL adaptive method was 5.19, whereas the developed method had a 8.29% decreased RMS error of 4.76. Additionally, the ICL adaptive simulation exhibited higher parameter estimation error from approximately 3–6 s than the developed method. To better compare the transient performance, the parameter estimation errors from the ICL adaptive and the developed method are shown in Figure 2-2.

Figure 2-3 illustrates the evolution of the normalized tracking error trajectory for each simulation. To compare the tracking performance, the RMS tracking error was computed. The RMS tracking errors corresponding to the simulations for the standard adaptive, ICL adaptive, and developed method are 27.06, 26.99, and 26.53 deg, respectively. The tracking error performance across all simulations were similar. However, the tracking error trajectories in the simulations with the standard adaptive and ICL adaptive methods have higher oscillatory behavior. The control input in (2–25) is dependent on $\hat{\theta}$. Consequently, oscillatory behavior in the parameter estimates may induce oscillations in the state trajectories, as seen from approximately 3–10 s.

Table 2-2 summarizes the simulations results in this section. In the leftmost column, e, e_{0-10}, and e_{ss} denote the RMS of the tracking error, tracking error from 0–10 s, and steady-state tracking error, respectively. Additionally, $\tilde{\theta}$, $\tilde{\theta}_{0-10}$, and $\tilde{\theta}_{ss}$ denote the RMS of the total, transient, and steady-state parameter estimation error, respectively.

2.5 Conclusion

Motivated by the improved transient performance, this chapter developed a new higher-order ICL-based adaptive update law. A general uncertain Euler-Lagrange dynamic system was considered. A Lyapunov-based analysis was used to guarantee the tracking and parameter estimation objectives were achieved. Under an FE condition, the closed-loop system yields global exponential stability of the tracking and parameter identification errors. To demonstrate the efficacy of the developed method, comparative simulations were performed on a two-link robot manipulator in Section 2.4. The simulation study showed the developed higher-order ICL adaptation outperformed the standard adaptive and the ICL adaptive schemes by 19.6% and 11.1%, respectively, in terms of the root mean squared parameter estimation errors.

The model uncertainty in this chapter was assumed to have structure, i.e., the uncertainty was LIP and assumed Assumption 2.1. Further motivation exists to investigate scenarios that consider a more general class of non-LIP model uncertainties that are examined in Chapter 3.

CHAPTER 3
ACCELERATED GRADIENT APPROACH FOR NEURAL NETWORK-BASED ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

In Chapter 2, the development assumed the system's uncertainties were LIP. However, in many practical scenarios, systems may have uncertainties in which there is no structure (non-LIP) or the uncertainty is unknown. To compensate for non-LIP system uncertainty, this chapter develops a new NN-based accelerated gradient adaptive controller to achieve trajectory tracking in general nonlinear control affine systems subject to non-LIP uncertainties. Higher-order accelerated gradient-based adaptation laws are developed to generate real-time estimates of both the unknown ideal output- and hidden-layer weights of a NN. A nonsmooth Lyapunov-based method is used to guarantee the closed-loop error system achieves global asymptotic tracking. Simulations are conducted to demonstrate the improved performance from the developed method. Results show the higher-order adaptation outperforms the standard gradient-based NN adaptation by 32.3% in terms of the root mean squared function approximation error.

3.1 Problem Formulation

3.1.1 Dynamic Model and Control Objective

Consider a control-affine nonlinear system modeled as

$$\dot{x} = f(x) + u, \tag{3–1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the state, $f : \mathbb{R} \to \mathbb{R}^n$ denotes the unknown differentiable drift dynamics, and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes a control input. The control objective is to track a user-defined desired trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite uncertainty of the drift dynamics in (3–1). Note that, unlike Chapter 2, the system uncertainties considered in this chapter are non-LIP and do not satisfy Assumption 2.1. To quantify the control objective, the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined as

$$e \triangleq x - x_d. \tag{3–2}$$

The desired trajectory and its time derivative are assumed to be continuous and bounded, i.e., $x_d \in \Omega$ and $\dot{x}_d \in \mathcal{L}_\infty$, for all $t \in \mathbb{R}_{\geq 0}$, where $\Omega \subset \mathbb{R}^n$ denotes a known compact set.

3.2 Control Development

3.2.1 Neural Network Function Approximation

NN function approximation methods are well-suited for systems with unknown or unstructured uncertainties, i.e., the uncertainty does not satisfy the typical LIP assumption (Assumption 2.1) in adaptive control (cf., [1, 2, 4]). To compensate for the unknown drift dynamics in (3–1), a NN-based feedforward estimate of the drift dynamics is introduced in this section. Let the NN architecture $\Phi : \mathbb{R}^n \times \mathbb{R}^{(L+1)\times n} \times \mathbb{R}^{(n+1)\times L} \to \mathbb{R}^n$ be defined as $\Phi\left(x_d, W, V\right) \triangleq W^T \sigma\left(V^T \overline{x}_d\right)$, where $\overline{x}_d \triangleq \left[x_d^T, 1\right]^T \in \mathbb{R}^{n+1}$ denotes a concatenated state vector to account for weight biases, $W \in \mathbb{R}^{(L+1)\times n}$ denotes the output-layer weight matrix, $V \in \mathbb{R}^{(n+1)\times L}$ denotes the hidden-layer weight matrix, $\sigma : \mathbb{R}^n \to \mathbb{R}^{L+1}$ denotes the vector of activation functions, and $L \in \mathbb{Z}_{>0}$ denotes the user-defined number of neurons in the hidden-layer. The vector of activation functions can be composed of various activation functions, and hence, may be represented as $\sigma \triangleq \left[\varsigma_1, \ldots, \varsigma_L, 1\right]^T$, where $\varsigma_i : \mathbb{R} \to \mathbb{R}$, for all $i \in \{1, \ldots, L\}$, denotes a piecewise continuously differentiable activation function. Let $\mathbb{C}\left(\Omega\right)$ denote the space where $f : \Omega \to \mathbb{R}^n$ is continuous. The universal function approximation theorem in [54, Thm 3.2] states the function space of NNs are dense in $\mathbb{C}\left(\Omega\right)$. Then for any $f \in \mathbb{C}\left(\Omega\right)$ and prescribed $\overline{\varepsilon} \in \mathbb{R}_{>0}$, there exists a constant $L \in \mathbb{Z}_{>0}$ and ideal weight matrices $W^* \in \mathbb{R}^{(L+1)\times n}$ and $V^* \in \mathbb{R}^{(n+1)\times L}$ such that $\sup_{x_d \in \Omega} \left\| f\left(x_d\right) - \Phi\left(x_d, W^*, V^*\right) \right\| \leq \overline{\varepsilon}$. Then the unknown drift dynamics in (3–1) is modeled as

$$f\left(x_d\right) = W^{*T} \sigma\left(V^{*T} \overline{x}_d\right) + \varepsilon\left(x_d\right), \qquad \forall x_d \in \Omega, \tag{3–3}$$

where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ denotes the unknown bounded function approximation error. The function approximation error is bounded such that $\sup_{x_d \in \Omega} \|\varepsilon(x_d)\| \leq \overline{\varepsilon}$. To facilitate the subsequent development, the following assumption is made.

Assumption 3.1. [55, Assumption 9.1] The ideal NN weights can be bounded as $\|W^*\|_F \leq \overline{W}$ and $\|V^*\|_F \leq \overline{V}$, where $\overline{W}, \overline{V} \in \mathbb{R}_{>0}$ are known constants.

3.2.2 Control Input and Weight Adaptation Laws

This section introduces the design of the controller and the higher-order accelerated gradient-based NN weight adaptation laws. The higher-order adaptation scheme is composed of two coupled first-order adaptation laws is used to estimate the NN weights in real-time. Based on the subsequent stability analysis, the higher-order adaptation laws for the output-layer weight estimate are designed as

$$\dot{\hat{\omega}} \triangleq \text{proj}\left(\Gamma_\omega \sigma\left(\hat{V}^T \overline{x}_d\right) e^T\right), \tag{3–4}$$

$$\dot{\hat{W}} \triangleq -\text{proj}\left(\Gamma_\omega \Gamma_W \tilde{W}\right), \tag{3–5}$$

where $\hat{\omega} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(L+1)\times n}$ and $\hat{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(L+1)\times n}$ denote an auxiliary and actual estimate of W^*, respectively, $\Gamma_\omega, \Gamma_W \in \mathbb{R}^{(L+1)\times(L+1)}$ denote user-defined positive definite learning gain matrices, and $\tilde{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(L+1)\times n}$ is defined as

$$\tilde{W} \triangleq \hat{W} - \hat{\omega}. \tag{3–6}$$

The operator $\text{proj}(\cdot)$ in (3–4) and (3–5) denotes the projection operator as defined in [1, eq. E.2] and is used to ensure the weight estimates remain bounded, i.e., $\hat{\omega}(t), \hat{W}(t) \in \mathcal{W}$ for all $t \in \mathbb{R}_{\geq 0}$, where $\mathcal{W} \triangleq \left\{w \in \mathbb{R}^{(L+1)\times n} : \|w\|_F \leq \overline{W}\right\}$ denotes a known convex set and $\overline{W}$ is known by Assumption 3.1. The higher-order adaptation laws for the hidden-layer weight estimate are designed as

$$\dot{\hat{\nu}} \triangleq \text{proj}\left(\Gamma_\nu \overline{x}_d e^T \hat{W}^T \sigma'\left(\hat{V}^T \overline{x}_d\right)\right), \tag{3–7}$$

$$\dot{\hat{V}} \triangleq -\text{proj}\left(\Gamma_\nu \Gamma_V \tilde{V}\right), \tag{3–8}$$

where $\hat{\nu} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(n+1)\times L}$ and $\hat{V} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(n+1)\times L}$ denote an auxiliary and actual estimate of V^*, respectively, $\Gamma_\nu, \Gamma_V \in \mathbb{R}^{(n+1)\times(n+1)}$ denote user-defined positive definite learning gain matrices, and $\tilde{V} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(n+1)\times L}$ is defined as

$$\tilde{V} \triangleq \hat{V} - \hat{\nu}. \tag{3–9}$$

Similar to (3–4) and (3–5), the projection operator is used to ensure $\hat{\nu}(t), \hat{V}(t) \in \mathcal{V}$ for all $t \in \mathbb{R}_{\geq 0}$, where $\mathcal{V} \triangleq \left\{v \in \mathbb{R}^{(n+1)\times L} : \|v\|_F \leq \overline{V}\right\}$ denotes a convex set. Based on the subsequent stability analysis, the control input is designed as

$$u \triangleq \dot{x}_d - k_e e - k_s \text{sgn}(e) - \rho(\|e\|) e - \hat{W}^T \sigma\left(\hat{V}^T \overline{x}_d\right) + \mu, \tag{3–10}$$

where $k_e, k_s \in \mathbb{R}_{>0}$ denote user-defined constants, $\text{sgn}(\cdot)$ denotes the vector signum function, and $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a known strictly increasing function that satisfies $\|f(x) - f(x_d)\| \leq \rho(\|e\|)\|e\|$ for all $x \in \mathbb{R}^n$ and $x_d \in \Omega$ [56, Lem. 5]. The auxiliary term $\mu : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ in (3–10) is designed to cancel cross-terms introduced from higher-order adaptation in the subsequent analysis and is defined as

$$\mu \triangleq 2\tilde{W}^T \sigma\left(\hat{V}^T \overline{x}_d\right) + 2\hat{W}^T \sigma'\left(\hat{V}^T \overline{x}_d\right) \tilde{V}^T \overline{x}_d. \tag{3–11}$$

3.2.3 Closed-Loop Error System

To facilitate the subsequent stability analysis, the closed-loop error system is developed in this section. Let

$$\tilde{\omega}^* \triangleq W^* - \hat{\omega}, \tag{3–12}$$

$$\tilde{W}^* \triangleq W^* - \hat{W}, \tag{3–13}$$

$$\tilde{\nu}^* \triangleq V^* - \hat{\nu}, \tag{3–14}$$

$$\tilde{V}^* \triangleq V^* - \hat{V}, \tag{3–15}$$

where $\tilde{\omega}^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(L+1)\times n}$ and $\tilde{\nu}^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(n+1)\times L}$ denote the weight estimation errors between the auxiliary weight estimates and ideal weights for the output- and hidden-layers, respectively, and $\tilde{W}^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(L+1)\times n}$ and $\tilde{V}^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{(n+1)\times L}$ denote the weight estimation errors between the actual weight estimates and ideal weights for the output- and hidden-layers, respectively. Taking the time derivative of (3–2), adding and subtracting $f(x_d)$, and substituting in (3–1) and (3–3), the open-loop error system can be expressed as

$$\dot{e} = f(x) - f(x_d) + W^{*T}\sigma\left(V^{*T}\overline{x}_d\right) + \varepsilon(x_d) + u - \dot{x}_d. \tag{3–16}$$

Substituting (3–10) into (3–16) yields the closed-loop error system

$$\begin{aligned}\dot{e} = {} & f(x) - f(x_d) + W^{*T}\sigma\left(V^{*T}\overline{x}_d\right) + \varepsilon(x_d) \\ & - \hat{W}^T\sigma\left(\hat{V}^T\overline{x}_d\right) - k_e e - k_s \text{sgn}(e) - \rho(\|e\|)e + \mu.\end{aligned} \tag{3–17}$$

The first-order Taylor's series approximation of $\sigma\left(V^{*T}\overline{x}_d\right)$ yields [31, eq. (22)]

$$\sigma\left(V^{*T}\overline{x}_d\right) = \sigma\left(\hat{V}^T\overline{x}_d\right) + \sigma'\left(\hat{V}^T\overline{x}_d\right)\tilde{V}^{*T}\overline{x}_d + \mathcal{O}^2\left(\tilde{V}^{*T}\overline{x}_d\right), \tag{3–18}$$

where $\sigma' : \mathbb{R}^L \to \mathbb{R}^{(L+1)\times L}$ denotes the gradient of the vector of activation functions (i.e., $\sigma'(y) \triangleq \frac{\partial \sigma}{\partial z}(z)\mid_{z=y} \forall y \in \mathbb{R}^L$), and $\mathcal{O}^2(\cdot)$ denotes higher-order terms in the first-order Taylor series approximation.[1] Adding and subtracting the terms $W^{*T}\sigma\left(\hat{V}^T\overline{x}_d\right)$ and $\hat{W}^T\sigma'\left(\hat{V}^T\overline{x}_d\right)\tilde{V}^{*T}\overline{x}_d$ in (3–17) and using (3–18) yields the closed-loop error system

$$\dot{e} = \chi + \hat{W}^T\sigma'\left(\hat{V}^T\overline{x}_d\right)\tilde{V}^{*T}\overline{x}_d + \tilde{W}^{*T}\sigma\left(\hat{V}^T\overline{x}_d\right) - k_e e - k_s \text{sgn}(e) - \rho(\|e\|)e + \mu, \tag{3–19}$$

[1] Given suitable functions f and g, the notation $f(x) = \mathcal{O}^k(g(x))$ means that there exits constants $c, x_0 \in \mathbb{R}_{>0}$ such that $\|f(x)\| \leq c\|g(x)\|^k$ for all $x \geq x_0$.

where $\chi : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an auxiliary function defined as $\chi \triangleq f(x) - f(x_d) + \tilde{W}^{*T}\sigma'\left(\hat{V}^T\overline{x}_d\right)\tilde{V}^{*T}\overline{x}_d + W^{*T}\mathcal{O}^2\left(\tilde{V}^{*T}\overline{x}_d\right) + \varepsilon(x_d)$.

3.3 Stability Analysis

The closed-loop control design introduced in Section 3.2 employs a discontinuous robust sliding mode term and may have potential discontinuities in σ' depending on the choice of activation functions (e.g., ReLU activation functions). As a result, the closed-loop system is nonsmooth and does not admit classical solutions. Hence, a nonsmooth Lyapunov-based analysis is used to analyze generalized solutions of the resulting closed-loop system and ensure the tracking objective is achieved [48] (cf., [1, Thm. A.8] for LaSalle-Yoshizawa invariance principles for smooth nonautonomous systems). Specifically, the switching analysis in [35] for NN-based adaptive control is adopted to model the closed-loop system as a state-dependent switched system composed of a finite collection of smooth functions. To facilitate the subsequent analysis, let $\varrho \in \mathcal{P}$ denote a switching index, where $\mathcal{P} \subset \mathbb{Z}_{\geq 0}$ denotes a set of possible switching indices. Then the NN function approximation in (3–3) can be represented as $f(x_d) = W^{*T}\sigma_\varrho\left(V^{*T}\overline{x}_d\right) + \varepsilon_\varrho(x_d)$, where $x_d \mapsto \sigma_\varrho$ is smooth for each $\varrho \in \mathcal{P}$ with the corresponding function reconstruction error $\varepsilon_\varrho(x_d)$. Additionally, the functions σ', χ, and μ can be represented by the switched functions σ'_ϱ, χ_ϱ, and μ_ϱ, respectively, where the switched functions are continuous for each $\varrho \in \mathcal{P}$. Then the closed-loop error system in (3–19) and the adaptation laws in (3–4) and (3–7) can be represented as

$$\dot{e} = \chi_\varrho + \hat{W}^T\hat{\sigma}'_\varrho\tilde{V}^{*T}\overline{x}_d + \tilde{W}^{*T}\hat{\sigma}_\varrho - k_e e - k_s \text{sgn}(e) - \rho(\|e\|)e + \mu_\varrho, \tag{3–20}$$

$$\dot{\hat{\omega}} = \text{proj}\left(\Gamma_\omega\hat{\sigma}_\varrho e^T\right), \tag{3–21}$$

$$\dot{\hat{\nu}} = \text{proj}\left(\Gamma_\nu\overline{x}_d e^T\hat{W}^T\hat{\sigma}'_\varrho\right), \tag{3–22}$$

for all $\varrho \in \mathcal{P}$, where the shorthand notation $\sigma_\varrho = \sigma_\varrho\left(V^{*T}\overline{x}_d\right)$, $\hat{\sigma}_\varrho = \sigma_\varrho\left(\hat{V}^T\overline{x}_d\right)$, and $\hat{\sigma}'_\varrho = \sigma'_\varrho\left(\hat{V}^T\overline{x}_d\right)$ is introduced for notional brevity. It is assumed that the bound $\sup_{x_d\in\Omega,\varrho\in\mathcal{P}} \|\varepsilon_\varrho(x_d)\| \leq \overline{\varepsilon}$ is satisfied. By the use of the projection algorithm in (3–5) and (3–8), the output- and hidden-layer weight estimates can be upper bounded as $\left\|\hat{W}(t)\right\|_F \leq \overline{W}$ and $\left\|\hat{V}(t)\right\|_F \leq \overline{V}$ for all $t \in \mathbb{R}_{\geq 0}$. Hence, using (3–13), (3–15), and Assumption 3.1, it follows that $\left\|\tilde{W}^*(t)\right\|_F \leq 2\overline{W}$ and $\left\|\tilde{V}^*(t)\right\|_F \leq 2\overline{V}$, for all $t \in \mathbb{R}_{\geq 0}$. Moreover, since $x_d, \hat{V} \in \mathcal{L}_\infty$ and the fact that σ'_ϱ and $\mathcal{O}^2_\varrho$ are continuous, it follows that σ'_ϱ and $\mathcal{O}^2_\varrho$ can be upper bounded by known constants for all $\varrho \in \mathcal{P}$. Then χ_ϱ can be upper bounded as

$$\|\chi_\varrho\| \leq c_1 + c_2\|e\| + \rho(\|e\|)\|e\|, \qquad \forall \varrho \in \mathcal{P}, \tag{3–23}$$

where $c_1, c_2 \in \mathbb{R}_{>0}$ are known constants, and $\rho(\cdot)$ was defined previously in (3–10).

For notational brevity, let $\Psi \in \mathbb{Z}_{>0}$ be defined as $\Psi \triangleq n + 2n(L+1) + 2L(n+1)$. Let the concatenated state be defined as $z \triangleq \left[e^T, \text{vec}\left(\tilde{W}\right)^T, \text{vec}\left(\tilde{\omega}^*\right)^T, \text{vec}\left(\tilde{V}\right)^T, \text{vec}\left(\tilde{\nu}^*\right)^T\right]^T \in \mathbb{R}^\Psi$, and let $\dot{z} = h_\varrho(z)$, for all $\varrho \in \mathcal{P}$, denote a collection of subsystems, where $h_\varrho : \mathbb{R}^\Psi \to \mathbb{R}^\Psi$. Then the corresponding switched system is represented as

$$\dot{z} = h_{p(z)}(z), \tag{3–24}$$

where $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$ denotes a Filippov solution to (3–24), $p : \mathbb{R}^\Psi \to \mathcal{P}$ denotes a state-dependent switching signal, and $h_\varrho(z)$ is defined as

$$h_\varrho(z) \triangleq \begin{bmatrix} g_{cl} \\ \text{vec}\left(\Gamma_\omega \hat{\sigma}_\varrho e^T + \text{proj}\left(\Gamma_\omega \Gamma_W \tilde{W}\right)\right) \\ -\text{vec}\left(\text{proj}\left(\Gamma_\omega \hat{\sigma}_\varrho e^T\right)\right) \\ \text{vec}\left(\Gamma_\nu \overline{x}_d e^T \hat{W}^T \hat{\sigma}'_\varrho + \text{proj}\left(\Gamma_\nu \Gamma_V \tilde{V}\right)\right) \\ -\text{vec}\left(\text{proj}\left(\Gamma_\nu \overline{x}_d e^T \hat{W}^T \hat{\sigma}'_\varrho\right)\right) \end{bmatrix},$$

for all $\varrho \in \mathcal{P}$, where $g_{cl} \triangleq \chi_\varrho + \hat{W}^T \hat{\sigma}'_\varrho \tilde{V}^{*T} \overline{x}_d + \tilde{W}^{*T} \hat{\sigma}_\varrho - k_e e - k_s \text{sgn}(e) - \rho(\|e\|) e + \mu_\varrho$. In the following theorem, nonsmooth Lyapunov-based analysis techniques developed in [48] are used to establish invariance properties of (3–24) to ensure the tracking objective is achieved.

Theorem 3. *Consider a system modeled with the dynamics in (3–1) and let Assumption 3.1 hold. Then the control input in (3–10) and higher-order weight adaptation laws in (3–4), (3–5), (3–7), and (3–8) ensure global asymptotic tracking in the sense that* $\lim_{t\to\infty} \|e(t)\| = 0$, $\lim_{t\to\infty} \left\| vec\left(\tilde{W}\right) \right\| = 0$, *and* $\lim_{t\to\infty} \left\| vec\left(\tilde{V}\right) \right\| = 0$, *provided the following sufficient gain conditions are satisfied*

$$k_s > c_1, \quad k_e > c_2, \tag{3–25}$$

where c_1 *and* c_2 *are known constants defined in (3–23).*

Proof. Consider a candidate common Lyapunov function $V_L : \mathbb{R}^\Psi \to \mathbb{R}_{\geq 0}$ defined as

$$V_L(z) \triangleq \frac{1}{2} e^T e + \frac{1}{2} \text{vec}\left(\tilde{W}\right)^T \left(I_{L+1} \otimes \Gamma_\omega^{-1}\right) \text{vec}\left(\tilde{W}\right) + \frac{1}{2} \text{vec}\left(\tilde{\omega}^*\right)^T \left(I_{L+1} \otimes \Gamma_\omega^{-1}\right) \text{vec}\left(\tilde{\omega}^*\right) + \frac{1}{2} \text{vec}\left(\tilde{V}\right)^T \left(I_{n+1} \otimes \Gamma_\nu^{-1}\right) \text{vec}\left(\tilde{V}\right) + \frac{1}{2} \text{vec}\left(\tilde{\nu}^*\right)^T \left(I_{n+1} \otimes \Gamma_\nu^{-1}\right) \text{vec}\left(\tilde{\nu}^*\right), \tag{3–26}$$

which satisfies the inequality $\underline{\alpha}(\|z\|) \leq V_L(z) \leq \overline{\alpha}(\|z\|)$, where $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ are continuous positive definite functions. Let $F_\varrho : \mathbb{R}^\Psi \rightrightarrows \mathbb{R}^\Psi$ denote the Filippov regularization of (3–24) and be defined as $F_\varrho \triangleq \text{K}[h_\varrho](z)$, where the calculus of $\text{K}[\cdot]$ is defined in [57]. Then the generalized time derivative of (3–26) can be computed as $\dot{\overline{V}}_L \triangleq \max_{p \in \partial V_L(z)} \max_{q \in F'_\varrho(z)} p^T q$ [48, Def. 3], where $F'_\varrho(z) \supseteq F_\varrho(z)$ denotes a bound on the regularization of (3–24), and ∂V_L denotes Clarke's generalized gradient of V_L [58, pp. 39]. Since $z \mapsto V_L$ is continuously differentiable, $\partial V_L(z) = \{\nabla V_L(z)\}$. Additionally, the time derivative of V_L exists for almost all time, i.e., $\dot{V}_L(z(t)) \overset{\text{a.e.}}{\in} \dot{\overline{V}}_L(z(t))$, where the notation $\overset{\text{a.e.}}{(\cdot)}$ denotes the relation holds for almost all time.

Taking the generalized time derivative of (3–26), using (3–6), (3–9), and (3–12)–(3–15), and performing some algebraic manipulation yields[2]

$$\dot{\overline{V}}_L = e^T \mathsf{K}\left[\dot{e}\right] + \mathrm{vec}\left(-\tilde{W}^* - 2\tilde{W}\right)^T \mathrm{vec}\left(\Gamma_\omega^{-1}\mathsf{K}\left[\dot{\hat{\omega}}\right]\right) + \mathrm{vec}\left(-\tilde{V}^* - 2\tilde{V}\right)^T \mathrm{vec}\left(\Gamma_\nu^{-1}\mathsf{K}\left[\dot{\hat{\nu}}\right]\right)$$
$$+ \mathrm{vec}\left(\tilde{W}\right)^T \mathrm{vec}\left(\Gamma_\omega^{-1}\mathsf{K}\left[\dot{\hat{W}}\right]\right) + \mathrm{vec}\left(\tilde{V}\right)^T \mathrm{vec}\left(\Gamma_\nu^{-1}\mathsf{K}\left[\dot{\hat{V}}\right]\right). \tag{3–27}$$

Substituting (3–5), (3–8), and (3–20)–(3–22) into (3–27) yields

$$\dot{\overline{V}}_L = e^T\left(\mathsf{K}\left[\chi_\varrho\right] - k_e e - k_s \mathsf{K}\left[\mathrm{sgn}\right](e) - \rho\left(\|e\|\right)e - 2\tilde{W}^T\mathsf{K}\left[\hat{\sigma}_\varrho\right] - 2\hat{W}^T\mathsf{K}\left[\hat{\sigma}'_\varrho\right]\tilde{V}^T\overline{x}_d\right)$$
$$+ e^T\mathsf{K}\left[\mu_\varrho\right] - \mathrm{vec}\left(\tilde{W}\right)^T\mathrm{vec}\left(\mathsf{K}\left[\mathrm{proj}\right]\left(\Gamma_W\tilde{W}\right)\right) - \mathrm{vec}\left(\tilde{V}\right)^T\mathrm{vec}\left(\mathsf{K}\left[\mathrm{proj}\right]\left(\Gamma_V\tilde{V}\right)\right). \tag{3–28}$$

Since χ_ϱ, $\hat{\sigma}_\varrho$, and $\hat{\sigma}'_\varrho$ for all $\varrho \in \mathcal{P}$ are continuous functions, $\mathsf{K}\left[\chi_\varrho\right] = \{\chi_\varrho\}$, $\mathsf{K}\left[\hat{\sigma}_\varrho\right] = \{\hat{\sigma}_\varrho\}$, and $\mathsf{K}\left[\hat{\sigma}'_\varrho\right] = \left\{\hat{\sigma}'_\varrho\right\}$ for all $\varrho \in \mathcal{P}$. The projection algorithm in (3–4), (3–5), (3–7), and (3–8) ensures the states in (3–6) and (3–9) can be bounded as $\left\|\tilde{W}(t)\right\|_F \leq 2\overline{W}$ and $\left\|\tilde{V}(t)\right\|_F \leq 2\overline{V}$, for all $t \in \mathbb{R}_{\geq 0}$, respectively. Lemma E.1 in [1] states $-\left(\theta^* - \hat{\theta}\right)^T M\mathrm{proj}(y) \leq -\left(\theta^* - \hat{\theta}\right)^T My$ for all $\theta^*, \hat{\theta} \in \Theta \subset \mathbb{R}^m$, $M \in \mathbb{R}^{m\times m}$, and $y \in \mathbb{R}^m$, where M denotes a positive definite matrix, and Θ denotes a convex set. Note that $K\left[\mathrm{proj}\right](y)$ computes the set of convex combinations of $\mathrm{proj}(y)$ and y at the points of discontinuity. Therefore $-\left(\theta^* - \hat{\theta}\right)^T M\mathsf{K}\left[\mathrm{proj}\right](y) \leq -\left(\theta^* - \hat{\theta}\right)^T My$, and hence, the terms with the $\mathrm{proj}(\cdot)$ operator in (3–28) can be bounded as

$$-\mathrm{vec}\left(\tilde{W}\right)^T\mathrm{vec}\left(\mathsf{K}\left[\mathrm{proj}\right]\left(\Gamma_W\tilde{W}\right)\right) \leq -\mathrm{vec}\left(\tilde{W}\right)^T\mathrm{vec}\left(\Gamma_W\tilde{W}\right), \tag{3–29}$$

$$-\mathrm{vec}\left(\tilde{V}\right)^T\mathrm{vec}\left(\mathsf{K}\left[\mathrm{proj}\right]\left(\Gamma_V\tilde{V}\right)\right) \leq -\mathrm{vec}\left(\tilde{V}\right)^T\mathrm{vec}\left(\Gamma_V\tilde{V}\right). \tag{3–30}$$

[2] Given matrices $A \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{q\times r}$, and $C \in \mathbb{R}^{r\times s}$, $\mathrm{vec}(ABC) = \left(C^T \otimes A\right)\mathrm{vec}(B)$ [59, Prop. 7.1.9].

Then substituting μ_ϱ into (3–28), using (3–23), (3–29), (3–30), and the facts that $e^T \mathrm{K}\left[\text{sgn}\left(e\right)\right] = \left\{\left\|e\right\|_1\right\}$ and $-k_s \left\|e\right\|_1 \leq -k_s \left\|e\right\|$, (3–28) can be bounded as

$$\dot{\overline{V}}_L \overset{\text{a.e.}}{\leq} -\left(k_e - c_2\right)\left\|e\right\|^2 - \lambda_W \left\|\tilde{W}\right\|_F^2 - \lambda_V \left\|\tilde{V}\right\|_F^2 - \left(k_s - c_1\right)\left\|e\right\|, \tag{3–31}$$

where $\lambda_W \in \mathbb{R}_{>0}$ and $\lambda_V \in \mathbb{R}_{>0}$ denote the known minimum eigenvalues of Γ_W and Γ_V, respectively. Provided the sufficient gain conditions in (3–25) are satisfied, (3–31) can be upper bounded as $\dot{\overline{V}}_L \overset{\text{a.e.}}{\leq} -\lambda \left\|e\right\|^2 - \lambda_W \left\|\tilde{W}\right\|_F^2 - \lambda_V \left\|\tilde{V}\right\|_F^2$, where $\lambda \in \mathbb{R}_{>0}$ denotes a known constant. From (3–26) and the fact that $\dot{\overline{V}}_L \leq 0$, it follows that $V_L \in \mathcal{L}_\infty$, which implies $z \in \mathcal{L}_\infty$, and hence, $e, \tilde{W}, \tilde{\omega}^*, \tilde{V}, \tilde{\nu}^* \in \mathcal{L}_\infty$. Using (3–2), the fact that $e, x_d \in \mathcal{L}_\infty$ implies $x \in \mathcal{L}_\infty$. Using the projection algorithm in (3–4), (3–5), (3–7), and (3–8) ensures $\hat{\omega}, \hat{W}, \hat{\nu}, \hat{V} \in \mathcal{L}_\infty$. Using (3–13) and (3–15), the fact that $\hat{W}, \hat{V} \in \mathcal{L}_\infty$ implies $\tilde{W}^*, \tilde{V}^* \in \mathcal{L}_\infty$, respectively. Using (3–4), the fact that $\hat{V}, x_d, e \in \mathcal{L}_\infty$ implies $\dot{\hat{\omega}} \in \mathcal{L}_\infty$. Using (3–7), the fact that $x_d, e, \hat{W}, \hat{V} \in \mathcal{L}_\infty$ implies $\dot{\hat{\nu}} \in \mathcal{L}_\infty$. Using (3–5) and (3–8), the fact that $\tilde{W}, \tilde{V} \in \mathcal{L}_\infty$ implies $\dot{\tilde{W}}, \dot{\tilde{V}} \in \mathcal{L}_\infty$, respectively. Using (3–11), the fact that $\tilde{W}, \hat{W}, \tilde{V}, x_d \in \mathcal{L}_\infty$ implies $\mu \in \mathcal{L}_\infty$. Using (3–10), the fact that $x_d, \dot{x}_d, e, \hat{W}, \hat{V}, \mu \in \mathcal{L}_\infty$ implies $u \in \mathcal{L}_\infty$. Invoking the LaSalle-Yoshizawa theorem extension for nonsmooth systems in [48, Thm. 2], $\lim_{t\to\infty} \left\|e\right\| = 0$, $\lim_{t\to\infty} \left\|\text{vec}\left(\tilde{W}\right)\right\| = 0$, and $\lim_{t\to\infty} \left\|\text{vec}\left(\tilde{V}\right)\right\| = 0$.

■

3.4 Simulation

Comparative simulations were conducted on the two-state nonlinear system described in [60] to demonstrate the performance of the developed method. The unknown drift dynamics in (3–1) was modeled as

$$f\left(x\right) = \begin{bmatrix} -x_1 + x_2 \\ -\frac{1}{2}x_1 - \frac{1}{2}x_2\left(1 - \left(\cos\left(2x_1\right) + 2\right)^2\right) \end{bmatrix}, \tag{3–32}$$

where $x \triangleq \left[x_1, x_2\right]^T \in \mathbb{R}^2$ denotes the system state.

Two simulation experiments were performed, where each simulation was conducted for 30 s with initial condition $x(0) = [1, 0.5]^T$. The desired trajectory $x_d \triangleq [x_{d,1}, x_{d,2}]^T$ was selected as $x_d(t) = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$. The NN used in each simulation had $L = 20$ neurons in the hidden-layer and contained sigmoid activation functions. The first simulation was performed with a baseline NN adaptive controller which used a typical gradient-based adaptive scheme given by

$$\dot{\hat{W}} \triangleq \Gamma_W \hat{\sigma} e^T, \tag{3–33}$$

$$\dot{\hat{V}} \triangleq \Gamma_V \overline{x}_d e^T \hat{W}^T \hat{\sigma}', \tag{3–34}$$

$$u \triangleq \dot{x}_d - \hat{W}^T \hat{\sigma} - k_e e - k_s \text{sgn}(e). \tag{3–35}$$

The second simulation was performed with the developed method in (3–4), (3–5), (3–7), (3–8), and (3–10). In both simulations, the hidden- and output-layer weight estimates were initialized randomly from the normal distribution $\mathcal{N}(0, 1)$. The parameters used in each simulation are summarized in Table 3-1.

Table 3-1. Simulation Parameters

Adaptation Law	Γ_ω	Γ_ν	Γ_W	Γ_V	k_e	k_s
Standard NN Adaptive	-	-	$50I_{21}$	$50I_3$	5	0.5
Developed Method	$50I_{21}$	$50I_3$	$0.9I_{21}$	$0.9I_3$	5	0.5

Table 3-2. RMS Tracking Error, Control Effort, and Function Approximation Error

RMS	Standard NN Adaptive	Developed Method
$\overline{e}$	0.0674	0.0457
$\overline{u}$	2.6881	2.5320
$\overline{\tilde{f}}$	1.4546	0.9845
$\overline{e}_\tau$	0.3223	0.2169
$\overline{u}_\tau$	6.8375	5.2455
$\overline{\tilde{f}}_\tau$	6.3851	3.4789

Table 3-2 summarizes the performance in each simulation. In the leftmost column, $\overline{e}$, $\overline{u}$, and $\overline{\tilde{f}}$ denote the root mean square (RMS) tracking error, control effort, and function approximation error, respectively. The subscript $(\cdot)_\tau$ denotes the transient period before the tracking errors reach a steady-state. As shown in Figure 3-1, the tracking errors converge to the origin and reach steady-state after approximately 1.3 s. The RMS tracking error for the standard NN and developed methods were 0.0674 and 0.0457, respectively. The developed method had a 32.2% decrease in the RMS tracking error with a 5.8% decrease in the RMS control effort in comparison to the standard NN adaptive method.

Figure 3-2 illustrates the normalized function approximation error for each simulation. The simulation with the developed method showed a 32.3% decrease in the RMS function approximation error in comparison to the simulation with the standard NN adaptive controller. The improved transient performance from the accelerated gradient approach in the developed method is evident as the RMS function approximation error is 45.5% lower than the errors with the standard NN adaptation during the transient period from 0 to approximately 1.3 s. Additionally, the RMS tracking error and control effort showed a 32.8% and 23.3% decrease, respectively, during the transient period using the developed method. To better illustrate the improved transient performance from the

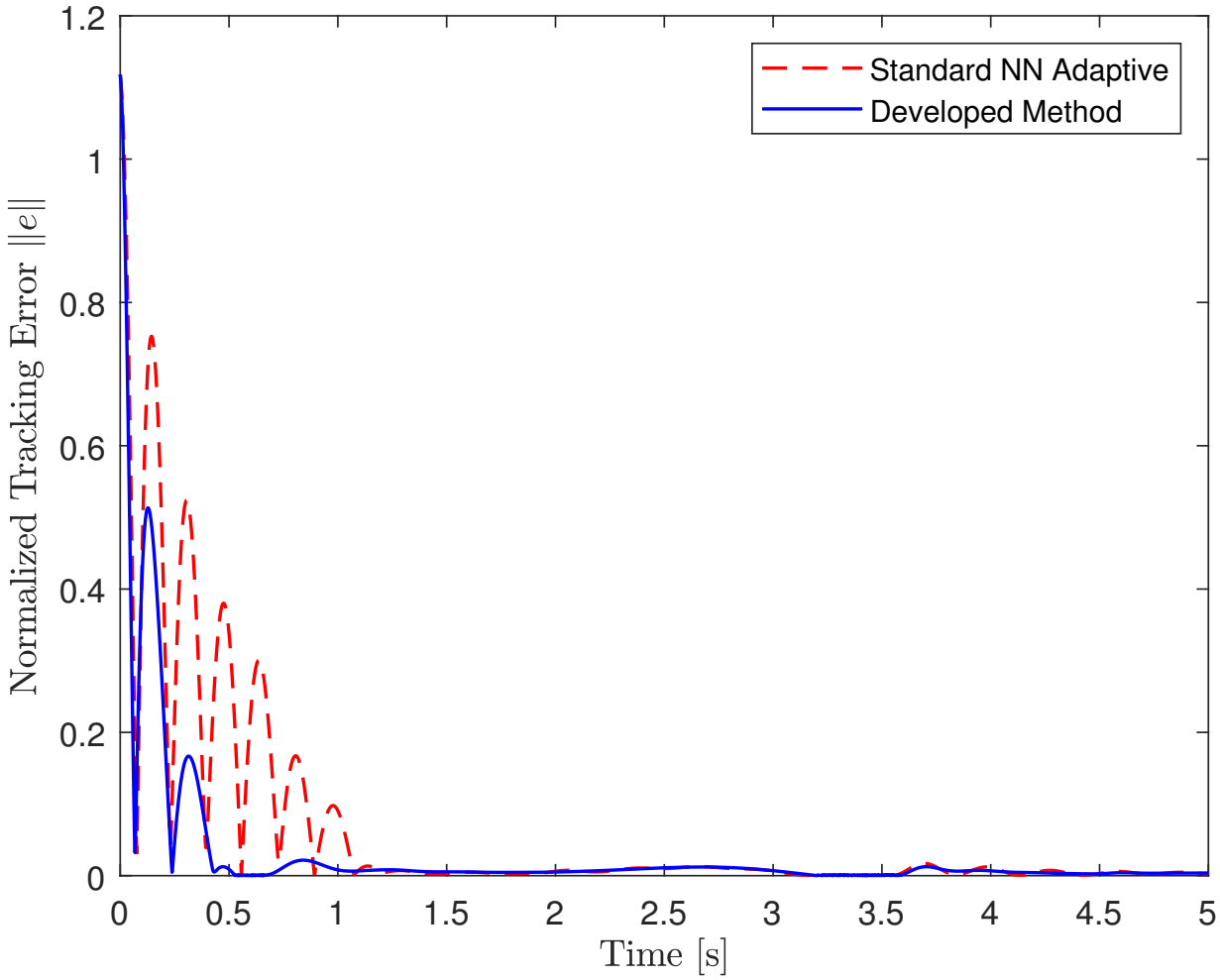

Figure 3-1. Evolution of the normalized tracking errors $\|e\|$. The simulation with the typical gradient-based NN adaptation scheme in (3–33)–(3–35) is shown in the dashed red line, and the simulation using the developed higher-order NN-based adaptation scheme in (3–4), (3–5), (3–7), (3–8), and (3–10) is shown in the solid blue line.

developed higher-order adaptation, Figure 3-3 illustrates the evolution of the output-layer weight estimates. The weight estimates generated from the standard NN adaptation and the developed method are shown in Figure 3-3a and Figure 3-3b, respectively. The output-layer weight estimates from the standard NN adaptation exhibit oscillatory behavior during the transient period from approximately 0–1.3 s. However, the weight estimates from the developed method had improved transient performance as seen by

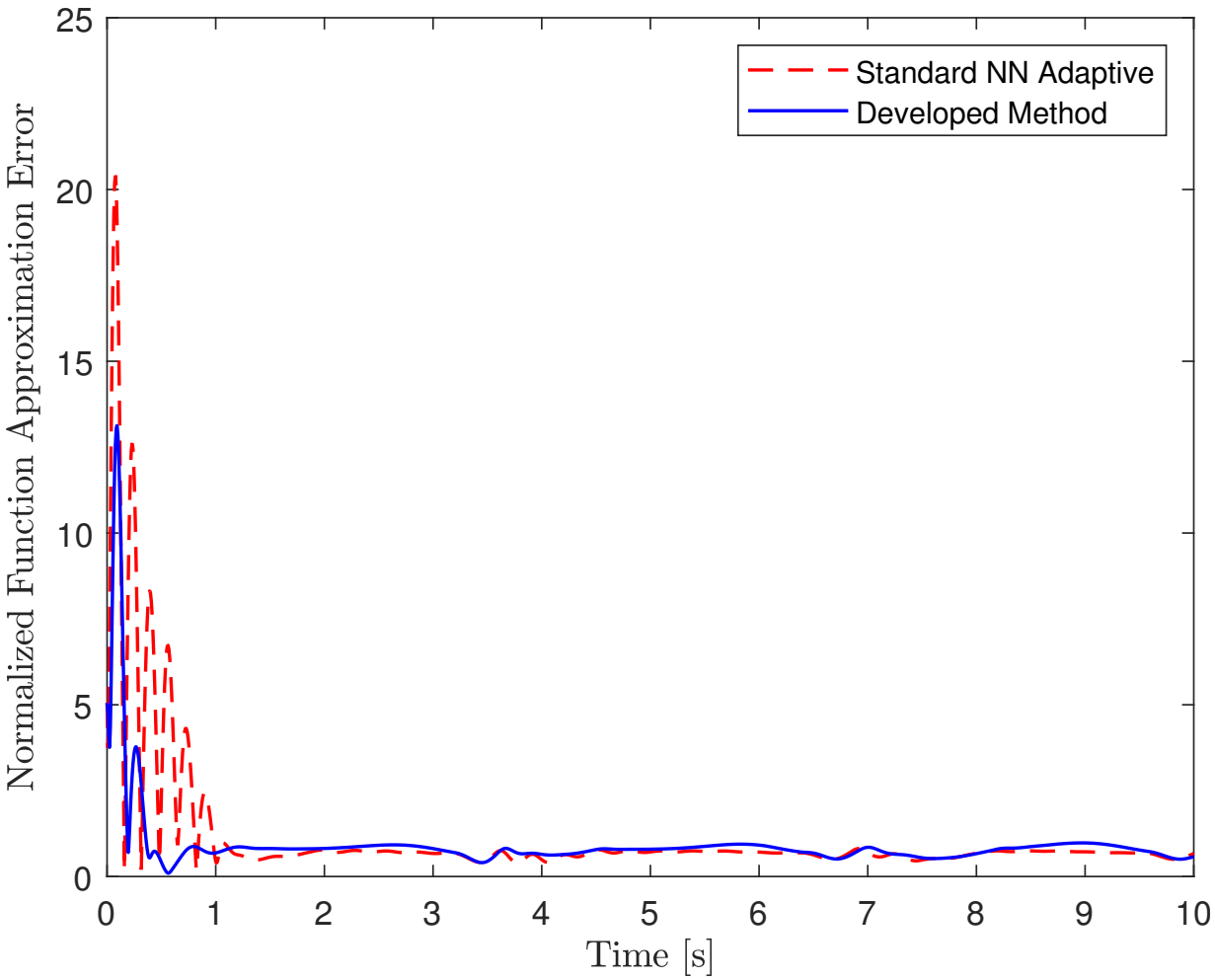

Figure 3-2. Evolution of the normalized function approximation errors $\left\| f\left(x_d\right) - \hat{f}\left(x_d\right) \right\|$ for each simulation.

the reduction of the oscillatory components in the weight estimates, and hence, resulting in improved real-time learning performance from the higher-order NN-based adaptation.

3.5 Conclusion

Recent connections in adaptive control to continuous-time analogues of Nesterov's accelerated gradient method have led to the development of new real-time adaptation laws based on accelerated gradient methods. In Chapter 2, a data-driven accelerated gradient-based adaptation law was for general uncertain Euler-Lagrange systems. However, the development and analysis in Chapter 2 assumed the system uncertainties satisfied the LIP assumption in Assumption 2.1. This chapter extended the developments in

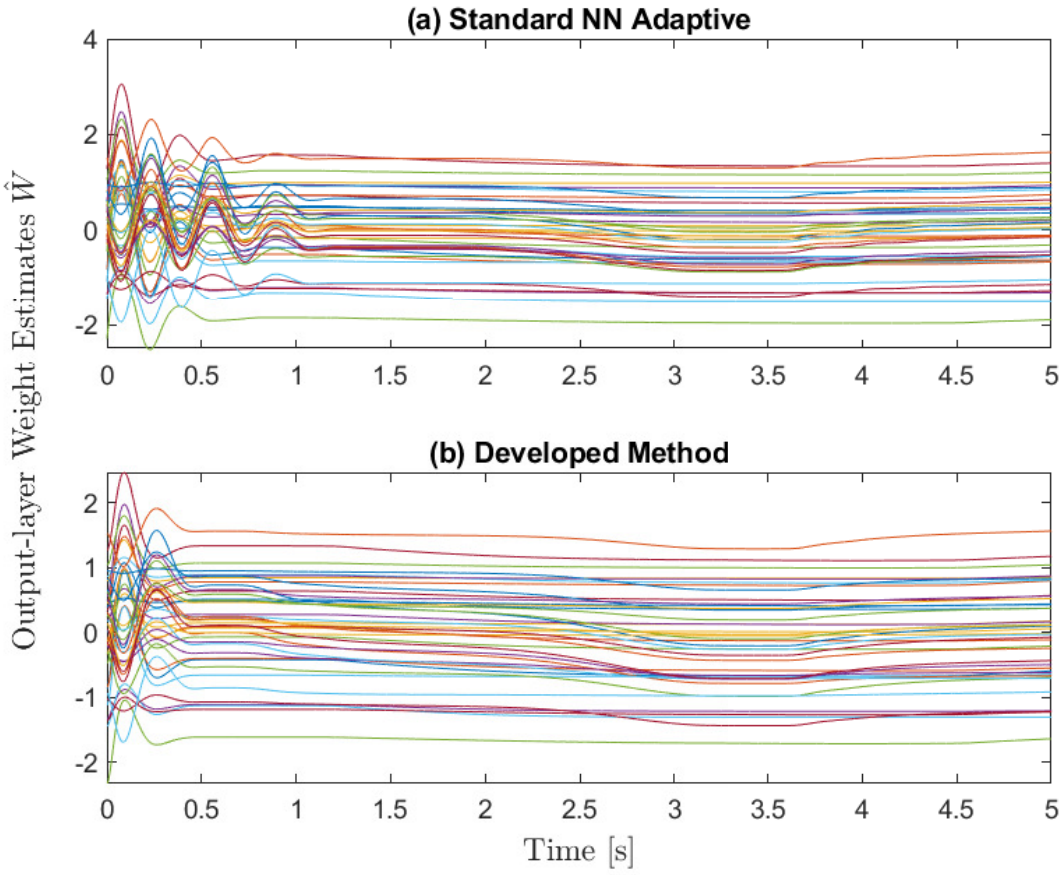

Figure 3-3. (a) Evolution of the output-layer weight estimates using the standard NN adaptation law in (3–33). (b) Evolution of the output-layer weight estimates using the developed higher-order adaptation laws in (3–4) and (3–5).

Chapter 2 by developing an accelerated gradient approach to design a higher-order NN-based adaptive control scheme for trajectory tracking of general nonlinear control affine systems subject to non-LIP uncertainties. A nonsmooth Lyapunov-based analysis was performed to show the developed methods yields global asymptotic tracking. Comparative simulations were conducted on a two-state nonlinear system to demonstrate the improved performance from higher-order NN weight adaptation laws. The simulations showed the developed higher-order adaptation outperformed the standard NN gradient adaptive scheme and had a 32.3% decrease in the RMS function approximation error.

CHAPTER 4
REAL-TIME MODULAR DEEP NEURAL NETWORK-BASED ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

In Chapter 3, an accelerated gradient-based NN adaptive controller was developed for trajectory tracking in general uncertain nonlinear control affine systems. However, the NN architecture in Chapter 3 is limited to NNs with only a single hidden layer. To advance the developments of Chapter 3 to DNN architectures, this chapter focuses on the development and analysis of DNN-based adaptive controller via Lyapunov-based techniques. Specifically, in this chapter, a real-time DNN adaptive control architecture is developed for uncertain control-affine nonlinear systems to track a time-varying desired trajectory. A Lyapunov-based analysis is used to develop adaptation laws for the output-layer weights and develop constraints for inner-layer weight adaptation laws. Unlike existing works in neural network and DNN-based control, the developed method establishes a framework to simultaneously update the weights of multiple layers for a DNN of arbitrary depth in real-time. The real-time controller and weight update laws enable the system to track a time-varying trajectory while compensating for unknown drift dynamics and parametric DNN uncertainties. A nonsmooth Lyapunov-based analysis is used to guarantee semi-global asymptotic tracking. Comparative numerical simulation results are included to demonstrate the efficacy of the developed method.

4.1 Problem Formulation

4.1.1 System Dynamics

Consider a control-affine nonlinear dynamic system modeled as

$$\dot{x} = f(x) + g(x)\, u, \tag{4–1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the state, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes unknown, locally Lipschitz drift dynamics, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denotes the known control effectiveness

matrix,[1] and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes the control input. Similar to Chapter 3, the system uncertainties considered in this chapter are non-LIP and do not satisfy Assumption 2.1. To facilitate the subsequent control design, the following assumption is made on the control effectiveness matrix. The control effectiveness matrix $g(x)$ is assumed to be full-row rank for all $x \in \mathbb{R}^n$. The right pseudo inverse of $g(x)$ is denoted by $g^+ : \mathbb{R}^n \to \mathbb{R}^{m \times n}$, where $g^+(\cdot) \triangleq g^T(\cdot)\left(g(\cdot)\, g^T(\cdot)\right)^{-1}$ is assumed to be bounded given a bounded argument.

4.1.2 Control Objective

The control objective is to track a user-defined time-varying trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite unknown system drift dynamics. The desired trajectory and its time derivative are assumed to be continuous and bounded, i.e., $x_d, \dot{x}_d \in \mathcal{L}_\infty$. The tracking objective is quantified by the tracking error e defined in (3–2).

4.2 Control Design

4.2.1 Feedforward DNN Estimate

As previously discussed in Chapter 3.2.1, NN-based adaptive control architectures are well-suited for uncertain or unstructured models, as in (4–1) where the drift dynamics $f(\cdot)$ are unknown. In Chapter 3, the NN architecture considered is limited to NNs with only a single hidden layer. The development and analysis in this chapter considers fully-connected DNN architectures with an arbitrary number of hidden layers. Using the universal function approximation property in [27], a DNN-based feedforward estimate of the drift dynamics is developed in this section. Let $\Omega \subset \mathbb{R}^n$ be a compact simply connected set and define $\mathbb{C}(\Omega)$ as the space where $f : \Omega \to \mathbb{R}^n$ is continuous. The universal function approximation property states there exist ideal weights and basis

[1] While the developed method does not account for an uncertain control effectiveness matrix for simplicity and to better focus the result on the unique specific contributions, the method in [33] can be used with the developed method to approximate the uncertain control effectiveness matrix online.

functions such that the drift dynamics $f(x) \in \mathbb{C}(\Omega)$ can be represented as

$$f(x) = W^{*T}\sigma^*(\Phi^*(x)) + \varepsilon(x), \tag{4–2}$$

where $W^* \in \mathbb{R}^{L\times n}$ denotes the unknown ideal output-layer weight matrix of the DNN, $\sigma^* : \mathbb{R}^p \to \mathbb{R}^L$ denotes the unknown vector of ideal activation functions corresponding to the output-layer of the DNN, $L \in \mathbb{Z}_{>0}$ denotes the user-defined number of neurons used in the output-layer, $\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ denotes the unknown function reconstruction error, and $\Phi^* : \mathbb{R}^n \to \mathbb{R}^p$ denotes the inner-layers of the DNN containing unknown ideal weight matrices and activation functions. Specifically, the ideal inner DNN Φ^* can be expressed as

$$\Phi^*(x) \triangleq \left(V_k^{*T}\phi_k^* \circ V_{k-1}^{*T}\phi_{k-1}^* \circ ... \circ V_1^{*T}\phi_1^*\right)\left(V_0^{*T}x\right), \tag{4–3}$$

where $k \in \mathbb{Z}_{>0}$ denotes the user-defined number of inner-layers, $V_j^* \in \mathbb{R}^{L_j\times L_{j+1}}$ for all $j \in \{0,\ldots,k\}$ denotes the j^{th} inner-layer ideal weight matrix, and $\phi_j^* : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ for all $j \in \{1,\ldots,k\}$ denotes the j^{th} inner-layer vector of ideal activation functions, and the symbol $\circ$ denotes function composition, e.g., $(g\circ h)(x) = g(h(x))$. The user-selected parameters $L_j \in \mathbb{Z}_{>0}$ for all $j \in \{1,\ldots,k\}$ denote the number of neurons in the j^{th} inner-layer. Note that $L_0 = n$ and $L_{k+1} = p$.

Based on (4–2), the DNN feedforward estimate of the drift dynamics $\hat{f} : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\hat{f}(x) \triangleq \hat{W}^T\hat{\sigma}\left(\hat{\Phi}(x)\right), \tag{4–4}$$

where $\hat{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L\times n}$ denotes the output-layer weight matrix estimate, $\hat{\sigma} : \mathbb{R}^p \to \mathbb{R}^L$ denotes the user-selected vector of activation functions, and $\hat{\Phi} : \mathbb{R}^n \to \mathbb{R}^p$ denotes the estimated inner-layers of the DNN. The inner-layer DNN estimate is defined as

$$\hat{\Phi}(x) \triangleq \left(\hat{V}_k^T\hat{\phi}_k \circ \hat{V}_{k-1}^T\hat{\phi}_{k-1} \circ ... \circ \hat{V}_1^T\hat{\phi}_1\right)\left(\hat{V}_0^T x\right), \tag{4–5}$$

where $\hat{V}_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j \times L_{j+1}}$ for all $j \in \{0, \ldots, k\}$ denotes the j^{th} inner-layer estimated weight matrix, and $\hat{\phi}_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ for all $j \in \{1, \ldots, k\}$ denotes the j^{th} inner-layer vector of activation functions. The design of the update laws on the weight estimates $\hat{W}$ and $\hat{V}_j$ are subsequently defined. The weight estimate mismatch of the ideal output-layer weight $\tilde{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L \times n}$ and weight estimate mismatch of the ideal inner-layer weights $\tilde{V}_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j \times L_{j+1}}$ for all $j \in \{0, \ldots, k\}$ are defined as

$$\tilde{W} \triangleq W^* - \hat{W}, \tag{4–6}$$

$$\tilde{V}_j \triangleq V_j^* - \hat{V}_j. \tag{4–7}$$

It is assumed there exist known constants $\overline{W^*}, \overline{V^*}, \overline{\sigma^*}, \overline{\hat{\sigma}}, \overline{\varepsilon} \in \mathbb{R}_{\geq 0}$ that upper bound the unknown ideal weights W^*, unknown ideal weights V_j^*, unknown ideal bounded activation functions[2] $\sigma^*(\cdot)$, user-selected bounded activation functions $\hat{\sigma}(\cdot)$, and the function reconstruction error $\varepsilon(\cdot)$, respectively, as $\sup_{x \in \Omega} \|W^*\|_F \leq \overline{W^*}$, $\sup_{x \in \Omega, \forall j} \left\|V_j^*\right\|_F \leq \overline{V^*}$, $\sup_{x \in \Omega} \|\sigma^*(\cdot)\| \leq \overline{\sigma^*}$, $\sup_{x \in \Omega} \|\hat{\sigma}(\cdot)\| \leq \overline{\hat{\sigma}}$, and $\sup_{x \in \Omega} \|\varepsilon(\cdot)\| \leq \overline{\varepsilon}$ [37].

4.2.2 Control Development

Based on the subsequent stability analysis, the control input is designed as

$$u \triangleq g^+(x) \left(\dot{x}_d - k_1 e - k_s \text{sgn}(e) - \hat{f}(x) \right), \tag{4–8}$$

where $k_1, k_s \in \mathbb{R}_{>0}$ are user-defined control gains, and $\text{sgn}(\cdot)$ denotes the signum function. Based on the subsequent Lyapunov-based stability analysis, the output-layer weight estimate update law $\dot{\hat{W}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L \times n}$ is designed as

$$\dot{\hat{W}} \triangleq \Gamma_W \hat{\sigma} \left(\hat{\Phi}(x) \right) e^T, \tag{4–9}$$

[2] For some common activation functions, e.g., hyperbolic tangent functions, sigmoid functions, radial basis functions, $\overline{\sigma^*} = \overline{\hat{\sigma}} = L$.

where $\Gamma_W \in \mathbb{R}^{L\times L}$ denotes a user-defined positive definite gain matrix used to adjust the learning rate of the output-layer weight matrix estimate.

Taking the time derivative of (3–2) and substituting in (4–1), (4–2), (4–4), and (4–8) yields the closed-loop error system

$$\dot{e} = W^{*T}\sigma^*\left(\Phi^*\left(x\right)\right) + \varepsilon\left(x\right) - k_1 e - k_s \text{sgn}\left(e\right) - \hat{W}^T\hat{\sigma}\left(\hat{\Phi}\left(x\right)\right). \tag{4–10}$$

In [33], the inner-layer weights of the DNN were held constant and only updated discretely with data-driven learning algorithms. Common learning algorithms include gradient descent variants (see [39, 40, 61] and [29, Ch. 8]). These algorithms often use training data sets to update DNN weights through an optimization process in which the algorithms seek to minimize a cost function. However, DNN training algorithms often require large amounts of training data and high computational costs [38], making real-time execution intractable. Motivated to execute real-time learning and allow flexibility in user-selection of training algorithms while maintaining stability guarantees, we develop modular inner-layer DNN weight estimate update laws (see [62] for single-hidden-layer NNs).

For all $j \in \{0,\ldots,k\}$, the j^{th} inner-layer weight update law $\dot{\hat{V}}_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j\times L_{j+1}}$ is designed as

$$\dot{\hat{V}}_j \triangleq p_j\left(t\right)\nu_j\left(e,t\right)\mathbf{1}_{\left\{\underline{\hat{V}_j}\leq\left\|\hat{V}_j\right\|_F\leq\overline{\hat{V}_j}\right\}}, \tag{4–11}$$

where $p_j : \mathbb{R}_{\geq 0} \to \{0,1\}$ denotes a switching signal that indicates the active inner-layer weight estimate updates, $\mathbf{1}_{\{\cdot\}}$ is the indicator function, $\underline{\hat{V}_j}, \overline{\hat{V}_j} \in \mathbb{R}$ are user-defined constants where $\overline{\hat{V}_j} \leq \overline{V^*}$, and $\nu_j : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j\times L_{j+1}}$ denotes a user-defined function that satisfies

$$\left\|\nu_j\right\|_F \leq \rho\left(\left\|e\right\|\right)\left\|e\right\|, \tag{4–12}$$

where $\rho : \mathbb{R}^n \to \mathbb{R}^n$ is a positive, globally invertible, and non-decreasing function.

4.3 Stability Analysis

The stability of the closed-loop tracking error system in (4–10) is analyzed to show the tracking objective is achieved in the following theorem.

Theorem 4. *Consider a system modeled by the dynamics in (4–1) with the initial condition $x(0) \in \Omega$. Then the control input in (4–8), output-layer weight adaptation law in (4–9), and the family of potential inner-layer weight adaptation laws that satisfy (4–11) ensure the closed-loop error system in (4–10) yields semi-global asymptotic tracking in the sense that $\lim_{t\to\infty} \|e(t)\| = 0$, provided the following sufficient gain condition is satisfied*

$$k_s > \overline{W^*}\left(\overline{\sigma^*} + \overline{\hat{\sigma}}\right) + \overline{\varepsilon} + 2\overline{V^*}(k+1)\,\rho\left(\sqrt{\frac{\overline{\alpha}}{\underline{\alpha}}}\,\|z(0)\|\right), \tag{4–13}$$

where $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}_{\geq 0}$ are known constants.

Proof. Let $\mathcal{D} \subset \mathbb{R}^{\Psi}$ be a set containing $z = 0_{\Psi\times 1}$ and Ω, where $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\Psi}$ denotes a concatenated state defined as $z \triangleq \left[\, e^T,\ \operatorname{vec}\left(\tilde{W}\right)^T,\ \operatorname{vec}\left(\tilde{V}_0\right)^T,\ \ldots,\ \operatorname{vec}\left(\tilde{V}_k\right)^T \,\right]^T$, and $\Psi \triangleq n(L+1) + \sum_{j=0}^{k} L_j L_{j+1}$ is defined for notional brevity. Consider the candidate Lyapunov function $V_L : \mathcal{D} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined as

$$V_L(z,t) \triangleq \frac{1}{2}e^T e + \frac{1}{2}\operatorname{tr}\left(\tilde{W}^T \Gamma_W^{-1} \tilde{W}\right) + \frac{1}{2}\sum_{j=0}^{k} \operatorname{tr}\left(\tilde{V}_j^T \tilde{V}_j\right), \tag{4–14}$$

which satisfies the inequality $\underline{\alpha}\|z\|^2 \leq V_L(z,t) \leq \overline{\alpha}\|z\|^2$, where $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}_{\geq 0}$ are known constants. Let $\zeta : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\Psi}$ be a Filippov solution to the differential inclusion $\dot{\zeta} \in K[h](\zeta,t)$, where $\zeta(t) = z(t)$, the calculus of $K[\cdot]$ is used to compute Filippov's differential inclusion as defined in [57], and $h : \mathbb{R}^{\Psi} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{\Psi}$ is defined as $h(\zeta,t) \triangleq \left[\, \dot{e}^T,\ \operatorname{vec}\left(\dot{\tilde{W}}\right)^T,\ \operatorname{vec}\left(\dot{\tilde{V}}_0\right)^T,\ \ldots,\ \operatorname{vec}\left(\dot{\tilde{V}}_k\right)^T \,\right]^T$. The generalized time-derivative of V_L along the Filippov trajectories of $\dot{\zeta} = h(\zeta,t)$ is defined by $\dot{\tilde{V}}_L(\zeta,t) \triangleq \bigcap_{\varsigma \in \partial V_L(\zeta,t)} \varsigma^T \begin{bmatrix} K[h](\zeta,t) \\ 1 \end{bmatrix}$, where $\partial V_L(\zeta,t)$ denotes Clarke's generalized gradient of $V_L(\zeta,t)$ [63, Eq. 13]. Since $V_L(\zeta,t)$ is continuously differentiable in ζ, then $\partial V_L(\zeta,t) = \{\nabla V_L(\zeta,t)\}$, where ∇ denotes the gradient operator. Additionally, the time

derivative of V_L exists almost everywhere (a.e.), i.e., $\dot{V}_L(\zeta,t) \overset{\text{a.e.}}{\in} \dot{\tilde{V}}_L(\zeta,t)$ for almost all $t \in \mathbb{R}_{\geq 0}$.

Taking the generalized time derivative of (4–14), using the trace operator property[3] , and substituting the closed-loop error system in (4–10), the output-layer adaptive update law in (4–9), and the inner-layer adaptive update laws in (4–11) yields

$$\dot{\tilde{V}}_L \subseteq e^T\left(W^{*T}\sigma^*\left(\Phi^*(x)\right)+\varepsilon(x)-k_1 e-k_s K\left[\text{sgn}(e)\right]-\hat{W}^T K\left[\hat{\sigma}\left(\hat{\Phi}(x)\right)\right]\right) - e^T\tilde{W}^T K\left[\hat{\sigma}\left(\hat{\Phi}(x)\right)\right]-\sum_{j=0}^{k}\text{tr}\left(K\left[\tilde{V}_j^T p_j(t)\,\nu_j(t)\,\mathbf{1}_{\left\{\underline{\hat{V}_j}\leq\left\|\hat{V}_j\right\|_F\leq\overline{\hat{V}_j}\right\}}\right]\right). \tag{4–15}$$

Adding and subtracting $e^T\left(W^{*T}K\left[\hat{\sigma}\left(\hat{\Phi}(x)\right)\right]\right)$ in (4–15) yields

$$\dot{\tilde{V}}_L \subseteq e^T W^{*T}\sigma^*\left(\Phi^*(x)\right)+e^T\varepsilon(x)-k_1 e^T e-k_s e^T K\left[\text{sgn}(e)\right]-e^T W^{*T}K\left[\hat{\sigma}\left(\hat{\Phi}(x)\right)\right] - \sum_{j=0}^{k}\text{tr}\left(K\left[\tilde{V}_j^T p_j(t)\,\nu_j(t)\,\mathbf{1}_{\left\{\underline{\hat{V}_j}\leq\left\|\hat{V}_j\right\|_F\leq\overline{\hat{V}_j}\right\}}\right]\right). \tag{4–16}$$

By the definition of the calculus $K[\cdot]$, $e^T K\left[\text{sgn}(e)\right]=\|e\|$. Using (4–12), (4–16) can be upper bounded as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\|e\|\left(k_s-\overline{W^*}\left(\overline{\sigma^*}+\overline{\hat{\sigma}}\right)-\overline{\varepsilon}-2(k+1)\,\overline{V^*}\rho\left(\|e\|\right)\right)-k_1\|e\|^2. \tag{4–17}$$

To ensure $k_s > \overline{W^*}\left(\overline{\sigma^*}+\overline{\hat{\sigma}}\right)+\overline{\varepsilon}+2(k+1)\,\overline{V^*}\rho(\|e\|)$, it is required that $\|e\|<\rho^{-1}\left(\frac{k_s-\overline{W^*}\left(\overline{\sigma^*}+\overline{\hat{\sigma}}\right)-\overline{\varepsilon}}{2(k+1)\overline{V^*}}\right)$, which implies $\|z\|<\rho^{-1}\left(\frac{k_s-\overline{W^*}\left(\overline{\sigma^*}+\overline{\hat{\sigma}}\right)-\overline{\varepsilon}}{2(k+1)\overline{V^*}}\right)$. The inequality in (4–17) can be upper bounded as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -k_1\|e\|^2, \quad \forall z\in\mathcal{D}, \tag{4–18}$$

[3] For real column vectors $a,b\in\mathbb{R}^n$, the trace of the outer product is equivalent to the inner product, i.e., $\text{tr}\left(ba^T\right)=a^T b$.

where $\mathcal{D} \triangleq \left\{ z \in \mathbb{R}^{\Psi} : \|z\| < \rho^{-1}\left(\frac{k_s - \overline{W^*}\left(\overline{\sigma^*}+\overline{\tilde{\sigma}}\right)-\overline{\varepsilon}}{2(k+1)\overline{V^*}}\right)\right\}$. Then using (4–14) and (4–16), V_L is positive definite and non-increasing, which implies $\|z(0)\| \leq \sqrt{\frac{V_L(0)}{\underline{\alpha}}}$. Therefore, it is sufficient to show $\|z(0)\| < \sqrt{\frac{\underline{\alpha}}{\overline{\alpha}}}\rho^{-1}\left(\frac{k_s - \overline{W^*}\left(\overline{\sigma^*}+\overline{\tilde{\sigma}}\right)-\overline{\varepsilon}}{2(k+1)\overline{V^*}}\right)$, which implies $\mathcal{S} \triangleq \left\{ z \in \mathcal{D} : \sqrt{\frac{\underline{\alpha}}{\overline{\alpha}}}\rho^{-1}\left(\frac{k_s - \overline{W^*}\left(\overline{\sigma^*}+\overline{\tilde{\sigma}}\right)-\overline{\varepsilon}}{2(k+1)\overline{V^*}}\right)\right\}$ is the region where (4–18) holds, and yields the sufficient gain condition in (4–13).

From (4–14) and (4–18), $V_L \in \mathcal{L}_\infty$, which implies $z \in \mathcal{L}_\infty$, and hence, $e, \tilde{W} \in \mathcal{L}_\infty$. Using (3–2) and (4–6) implies $x \in \mathcal{L}_\infty$ and $\hat{W} \in \mathcal{L}_\infty$, respectively. Using (4–11) and (4–12), $e \in \mathcal{L}_\infty$ implies $\nu_j \in \mathcal{L}_\infty$, and by the use of the indicator function $\mathbf{1}_{\left\{\underline{\hat{V}_j} \leq \left\|\hat{V}_j\right\|_F \leq \overline{\hat{V}_j}\right\}}$, implies $\dot{\hat{V}}_j \in \mathcal{L}_\infty$ for all $j \in \{0, \ldots, k\}$. Using (4–5), the fact that $x, \hat{V}_j \in \mathcal{L}_\infty$ implies $\hat{\Phi} \in \mathcal{L}_\infty$. By design, $\dot{x}_d, \hat{\sigma} \in \mathcal{L}_\infty$. Using (4–8), the fact that $x, e, \dot{x}_d, \hat{W}, \hat{\sigma} \in \mathcal{L}_\infty$ implies $u \in \mathcal{L}_\infty$. Using (4–9), the fact that $\hat{\sigma}, e \in \mathcal{L}_\infty$ implies $\dot{\hat{W}} \in \mathcal{L}_\infty$. By the LaSalle-Yoshizawa theorem extension for nonsmooth systems in [48] and [64], $k_1 \|e\|^2 \to 0$, which implies $\|e(t)\| \to 0$ as $t \to \infty$.

■

4.4 Simulation

To demonstrate the performance of the developed method, simulations are performed on the nonlinear system from [65], where the drift dynamics $f(x)$ are modeled by (3–32) and the control effectiveness is modeled by $g(x) = \operatorname{diag}[5, 3]$. The desired trajectory is $x_d = [3\cos(t), 5\sin(t)]^T$. The initial condition is $x(0) = [3, 0]^T$. The controller gains are selected as $k_s = 0.5$ and $k = 7$.

To illustrate the modularity of the architecture, two simulation studies are conducted for 60 seconds, each with different inner-layer adaptation laws and structures implemented. During the entire 60 seconds, the output-layer update law in (4–9) is active in both studies. Due to the large number of weights in this system, only one inner-layer

weight update law is active at a time.[4] However, the selection of switching signals that dictates when to update each inner-layer DNN may be arbitrarily selected. Computational resources may be allocated to update a subset of the inner-layer weights, or all inner-layers may be updated arbitrarily at a time. Additionally, inner-layer weights may dropout, or be selectively turned off to prevent over-fitting [46].

To reduce the computational load and show the flexibility in selection of the switching signals to update each inner-layer DNN weight estimate, in both simulation studies, the switching signal is arbitrarily designed as

$$p_j(t) = \begin{cases} 1, & t \in [10j, 10(j+1)], \\ 0, & \text{else}, \end{cases} \tag{4–19}$$

for all $j \in \{0, \ldots, 5\}$. Based on the switching signals in (4–19), each inner-layer weight update law is active for 10 seconds during the duration of the simulation and is activated in consecutive order, i.e., $\hat{V}_0$ is active from 0 to 10 seconds, $\hat{V}_1$ is active from 10 to 20 seconds, etc. If the update law is not active (i.e., $p_j(t) = 0$), then its associated weights are not updated.

The DNN is composed of 6 layers and the hyperbolic tangent function is the activation function for each neuron. Layers 1-6 have 12, 10, 15, 15, 12, and 20 neurons, respectively. The outer-layer weight learning parameter is set to $\Gamma_W = 10 \cdot \mathbf{1}_{L \times L}$, where $\mathbf{1}_{n \times m}$ is an $n \times m$ matrix of ones. The bounds on the inner-layer weights are $\underline{\hat{V}_j} = 10^{-6}$ and $\overline{\hat{V}_j} = 250$ for all $j \in \{0, 1, \ldots, 5\}$. The initial conditions for the output-layer weight

[4] There are 955 individual weights in the simulation. As guaranteed by the analysis, the developed method can update each layer separately, i.e., a subset of the total number of weights are updated at a given time. It may be computationally burdensome for some systems to update a large number of weights online. One purpose of this simulation is to highlight the developed method's ability update different DNN layers separately.

estimate $\hat{W}$ and the inner-layer weight estimates $\hat{V}_j$ are randomly selected from a uniform distribution from $[-0.5, 0.5]$.

Various update laws for $\dot{\hat{V}}_{0-5}$ can be selected and designed for learning the inner-layer DNN weights while guaranteeing tracking performance. The constraints in (4–12) provide general guidelines and enable the user to select or design update laws accordingly, such as gradient tuning laws based on back-propagated errors [66] or a Hebbian tuning law [67].

In the first simulation study (Study 1), the inner-layer weight update laws, which are heuristically selected and inspired by the methods in [68] and [69], are selected as

$$\nu_j = \Gamma_{V_j} e \cdot \mathrm{re}\left(\left(\tanh \circ \hat{\phi}_j^{-1} \circ \hat{V}_j^{+T} \hat{\phi}_{j+1}^{-1} \circ \cdots \circ \hat{V}_5^{+T} \hat{\sigma}^{-1}\right)\left(\hat{W}^{+T} \dot{\bar{x}}\right)\right) \hat{V}_j, \tag{4–20}$$

for all $j \in \{0, 1, \ldots, 5\}$, where the inner-layer weight learning parameters are set to $\Gamma_{V_j} = 1000 \cdot \mathbf{1}_{L_j \times 2}$, $\hat{V}_j^+$ is the right-pseudo inverse of $\hat{V}_j$, the $\mathrm{re}\,(\cdot)$ operator outputs element-wise the real component of each entry in $\hat{V}_j$, and $\dot{\bar{x}} \in \mathbb{R}^n$ denotes the numerically generated state derivative. Due to the projection bounds $\underline{\hat{V}_j}$ and $\overline{\hat{V}_j}$, $\hat{V}_j^+$ exists and is bounded. The selected update law satisfies the modular adaptive control constraint in (4–12).

Figure 4-1 shows the tracking error $e \triangleq [e_1, e_2]^T$ with the inner-layer DNN weight update law in (4–20). From initialization to approximately 30 seconds, the system exhibits poor tracking performance with oscillatory behavior. Poor controller performance is likely due to randomly initialized weights which signifies no *a priori* model knowledge in the drift dynamic approximation. However, as each subsequent inner-layer weight matrix $\hat{V}_{0-5}$ is updated, the tracking performance improves and the amplitude of the error signals decreases, which indicates the DNNs improved approximation of the drift dynamics.

Figure 4-2 shows the online adjustment of the inner-layer weights with the inner-layer DNN weight update law in (4–20). Each set of weights is divided by a set of black

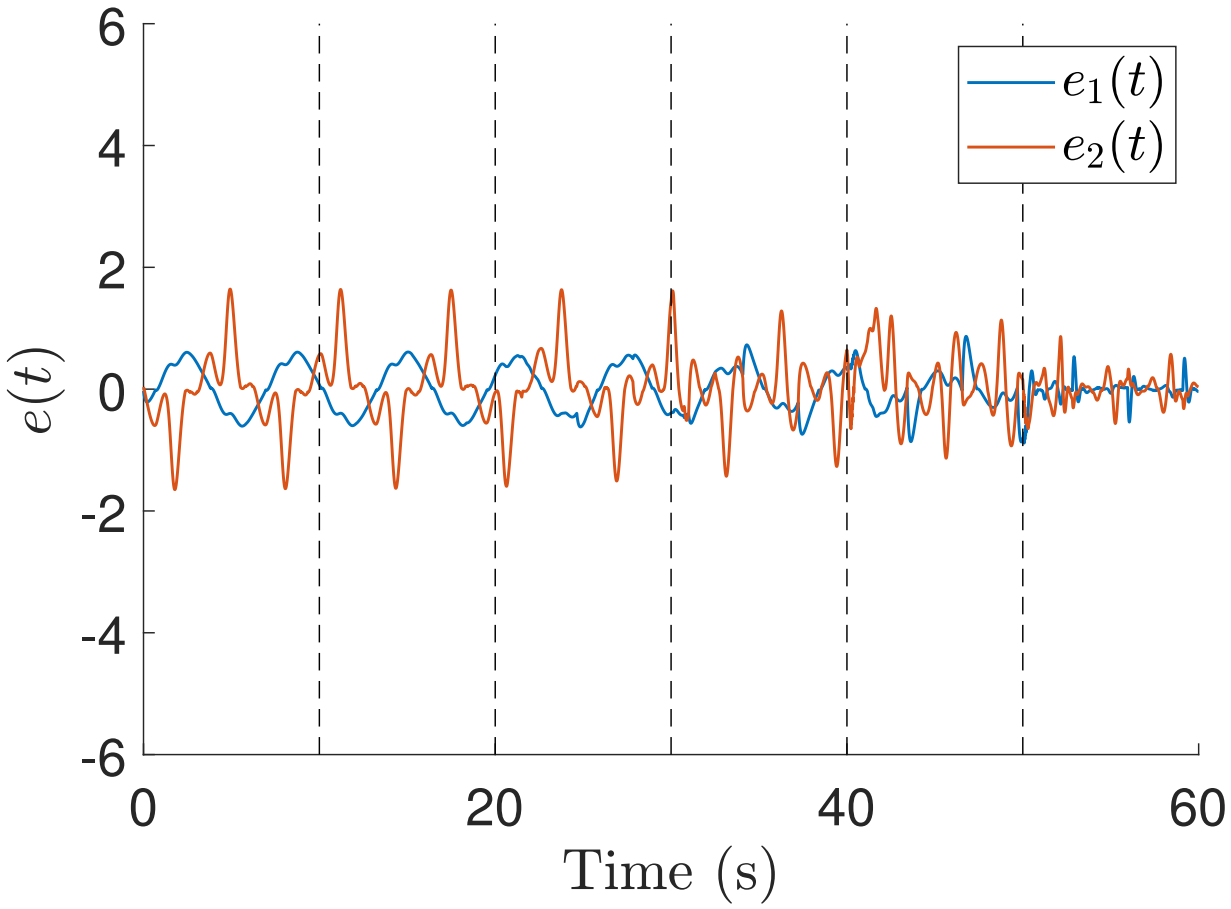

Figure 4-1. Tracking error e in the system with the inner-layer DNN weight update law in (4–20). The vertical lines denote when a new inner-layer update law is activated.

dashed lines. Note that some of the weights are unaffected by the indicator function since the upper bounds $\overline{\hat{V}_j}$ are sufficiently large. However, the weights of layers 3 and 4 are affected by the indicator function. When layers 3 and 4 are activated, their weight estimates quickly reach the bounds defined by the indicator function.

To provide a comparison in the selection of inner-layer weight update laws, a second simulation study (Study 2) is performed where the DNN is structured similarly as the first simulation study, and each inner-layer update is activated according to the switching signal in (4–19). To illustrate the modularity in the selection of inner-layer weight update laws, various inner-layer weight update laws are arbitrarily selected to

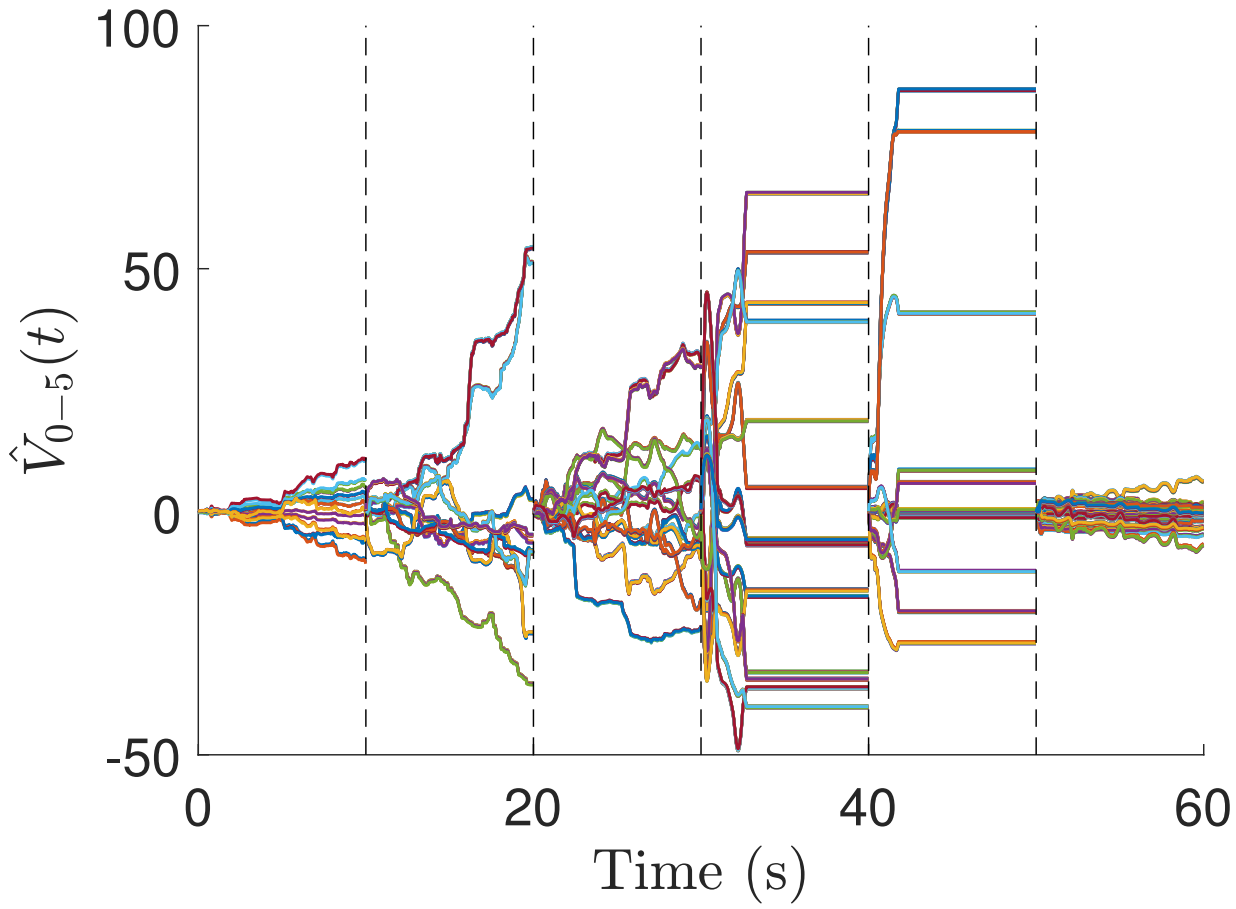

Figure 4-2. Value of the active DNN inner-layer weight estimates $\hat{V}_{0-5}$ when each inner-layer update law is active with the inner-layer DNN weight update law in (4–20). The vertical lines denote when a new inner-layer update law is activated. Based on the switching signals in (4–19), $\hat{V}_0$ is active from 0 to 10 seconds, $\hat{V}_1$ is active from 10 to 20 seconds, etc.

satisfy (4–12) and are selected as

$$\begin{aligned}
\nu_0 &= \Gamma_{V_0} \mathsf{e}^{-\frac{e_1^2}{2}} \mathsf{e}^{-\frac{e_2^2}{2}} \|e\|, \\
\nu_1 &= \Gamma_{V_1} \mathsf{e}^{-\frac{e_1^2}{2}} \tanh(e_2) \|e\|, \\
\nu_2 &= \Gamma_{V_2} \tanh(e_1) \tanh(e_2) \|e\|, \\
\nu_3 &= \Gamma_{V_3} \frac{1}{1+\mathsf{e}^{-e_1}} \mathsf{e}^{-\frac{e_2^2}{2}} \|e\|, \\
\nu_4 &= \Gamma_{V_4} \tanh(e_1) \frac{1}{1+\mathsf{e}^{-e_2}} \|e\|, \\
\nu_5 &= \Gamma_{V_5} \frac{1}{1+\mathsf{e}^{-e_1}} \tanh(e_2) \|e\|,
\end{aligned} \tag{4–21}$$

where the inner-layer weight learning parameters are set to $\Gamma_{V_j} = 1000 \cdot \mathbf{1}_{L_j \times 2}$ for all $j \in \{0, 1, \ldots, 5\}$.

Figure 4-3 shows the tracking error e with the inner-layer DNN weight update law in (4–21). Similar to Study 1, the tracking error results in Figure 4-3 show poor tracking performance at the start of the simulation. However, as the simulation progresses and each subsequent inner-layer is activated, the tracking performance improves.

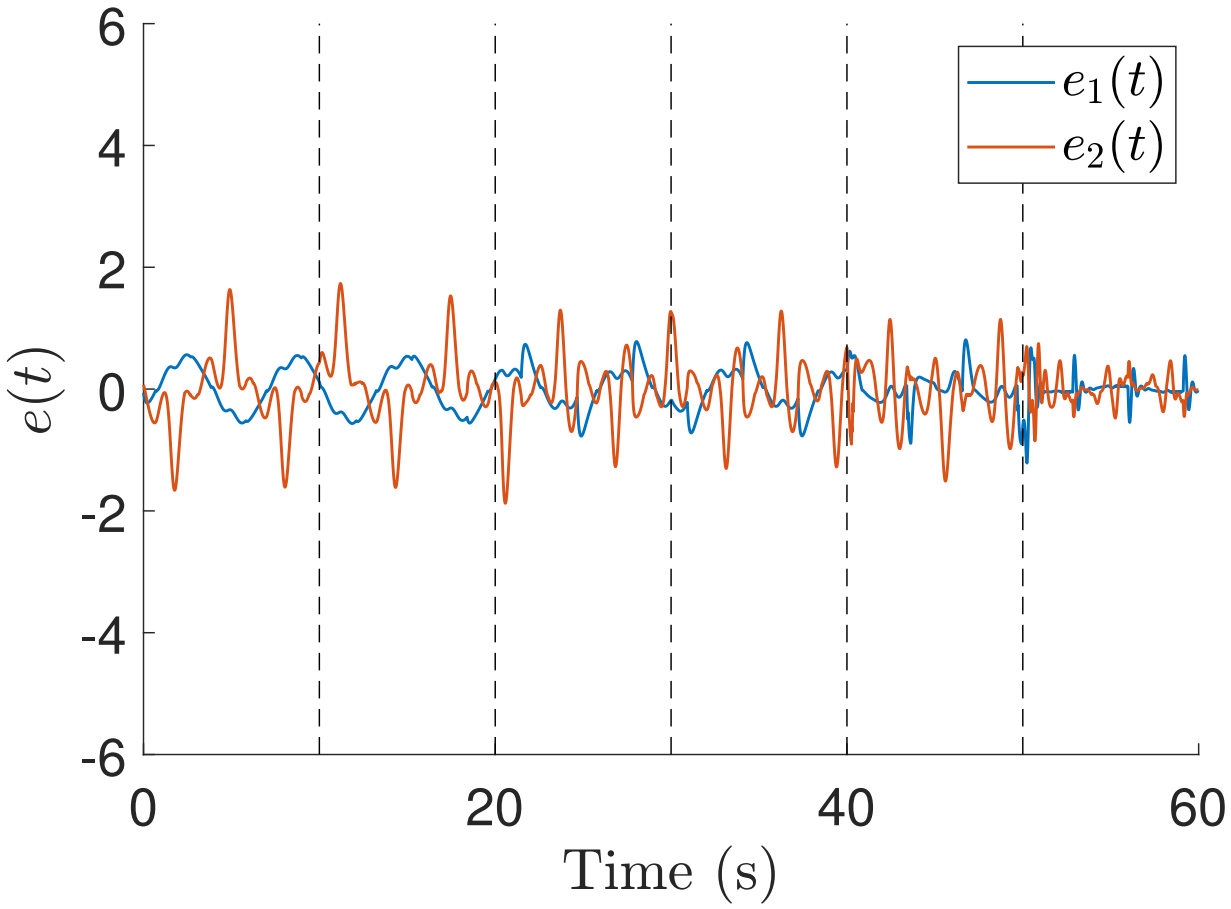

Figure 4-3. Tracking error e in the system with inner-layer DNN weight update law in (4–21). The vertical lines denote when a new inner-layer update law is activated.

Table 4-1 shows the root mean squared (RMS) error and standard deviation (SD) of the tracking error. The leftmost column indicates the active inner-layer weight law, e.g., the first row is data collected while the $\hat{V}_0$ update law is active. Study 1 uses the

heuristically selected update law in (4–20), and the RMS error and SD corresponding to $\hat{V}_0$ are 0.735 and 0.145, respectively. Study 2 uses the update law in (4–21), and the RMS error and SD are 0.705 and 0.145, respectively. In Study 1, as each subsequent inner-layer update law is activated, the tracking performance improves and results in an RMS error and SD of 0.289 and 0.039, respectively. Although a similar trend is seen in Study 2, the inner-layer update law in Study 1 outperforms Study 2 which indicates the inner-layer update law yielded better function approximation performance, as expected. However, the tracking objective is achieved in both studies, despite different inner-layer update laws, as guaranteed by the analysis.

Table 4-1. RMS and SD Error

Active $\hat{V}_j$	Study 1		Study 2	
	RMS Error	SD Error	RMS Error	SD Error
$\hat{V}_0$	0.735	0.145	0.705	0.145
$\hat{V}_1$	0.727	0.142	0.700	0.149
$\hat{V}_2$	0.710	0.136	0.651	0.137
$\hat{V}_3$	0.640	0.116	0.612	0.098
$\hat{V}_4$	0.658	0.087	0.591	0.094
$\hat{V}_5$	0.289	0.039	0.327	0.050

4.5 Conclusion

This chapter developed a DNN-based modular adaptive control update laws and constraints, which was inspired by existing modular adaptive control constraints, to achieve trajectory tracking objectives. Unlike Chapter 3, which used NN architectures with only a single hidden layer, this chapter focused on the development and analysis of DNN-based adaptation laws for general fully-connected DNNs with an arbitrary number of layers. The modular adaptive control framework provides general constraints and enables users to design update laws accordingly. The developed method also allows for different sets of weights to be arbitrarily switched. A simulation study was performed

to compare the performance of different inner-layer weight update laws selected. Inner-layer weight update laws were designed and selected that satisfy the developed design constraints. Despite different inner-layer update laws used and arbitrary switching, the implemented controllers achieved the tracking objective.

CHAPTER 5
ACCELERATED GRADIENT APPROACH FOR DEEP NEURAL NETWORK-BASED ADAPTIVE CONTROL OF UNKNOWN NONLINEAR SYSTEMS

This chapter leverages the insights developed from Chapter 4 on the development and analysis of DNN adaptation to build upon the developments of Chapter 3 and develop a new DNN-based adaptation law based on the accelerated gradient methods. To compensate for non-LIP uncertainties, the results in Chapter 3 developed a NN-based accelerated gradient adaptive controller to achieve trajectory tracking for nonlinear systems; however, the development and analysis only considered single hidden layer NNs. In this chapter, a generalized DNN architecture with an arbitrary number of hidden layers is considered, and a new DNN-based accelerated gradient adaptation scheme is developed to generate estimates of all the DNN weights in real-time. A nonsmooth Lyapunov-based analysis is used to guarantee the developed accelerated gradient-based DNN adaptation design achieves global asymptotic tracking error convergence for general nonlinear control affine systems subject to unknown (non-LIP) drift dynamics and exogenous disturbances. Simulations are conducted to demonstrate the improved performance from the developed method. Results show the developed accelerated gradient-based DNN adaptation outperforms gradient-based DNN adaptation by 67.41% and 78.82% in terms of the root mean squared tracking and function approximation errors, respectively.

5.1 Problem Formulation

5.1.1 Dynamic Model and Control Objective

Consider a control-affine nonlinear system modeled as

$$\dot{x} = f(x) + g(x)\,u + d(t), \tag{5–1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the system state, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown differentiable drift dynamics, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denotes a known control effectiveness, $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an unknown exogenous disturbance, and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes

a control input. To facilitate the subsequent stability analysis, the following assumptions are made.

Assumption 5.1. The exogenous disturbance $d(\cdot)$ can be bounded as $\|d(t)\| \leq \overline{d}$ for all $t \in \mathbb{R}_{\geq 0}$, where $\overline{d} \in \mathbb{R}_{>0}$ denotes a known constant.

Assumption 5.2. The control effectiveness matrix $g(x)$ is full row rank for all $x \in \mathbb{R}^n$.

The control objective is to use an accelerated gradient approach to design a DNN-based adaptive controller to track a user-defined desired trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite uncertainty of the drift dynamics in (5–1). The desired trajectory and its time derivative are assumed to be continuous and bounded, i.e., $x_d(t) \in \Omega$, for all $t \in \mathbb{R}_{\geq 0}$, and $\dot{x}_d \in \mathcal{L}_\infty$ where $\Omega \subset \mathbb{R}^n$ denotes a known compact set. The tracking objective is quantified by the tracking error e defined in (3–2).

5.1.2 Deep Neural Network Architecture and Function Approximation

NN function approximation methods are well-suited for systems with unknown or unstructured uncertainties, i.e., the uncertainty does not satisfy the typical LIP assumption (Assmption 2.1) in adaptive control (cf., [1, 2, 4]). To compensate for the unknown drift dynamics in (5–1), a fully-connected DNN-based feedforward estimate of the drift dynamics is introduced in this section. The DNN architecture considered in this chapter is the same as in (4–2). However, the strategy in Chapter 4 was to develop an adaptation law for the output-layer weights while modular constraints were developed for the inner layer weight adaptation laws. The notation for the DNN architecture is slightly redefined in this section to generalize the architecture with the addition of bias terms, and to also facilitate the subsequent development which contains adaptation laws for all the layers of the DNN. Let the DNN architecture $\Phi : \mathbb{R}^n \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}^{L_{k+1}}$ be defined as

$$\Phi(s, \theta) \triangleq \left(V_k^T \sigma_k \circ \ldots \circ V_1^T \sigma_1\right)\left(V_0^T \overline{s}\right), \tag{5–2}$$

where $\overline{s} \triangleq \left[s^T, 1\right]^T \in \mathbb{R}^{L_0+1}$ denotes a concatenated state input that is augmented by 1 to facilitate the inclusion of bias terms, $\theta \triangleq \left[\text{vec}\left(V_k\right)^T, \ldots, \text{vec}\left(V_0\right)^T\right]^T \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ denotes a concatenated vector of the DNN hidden-layer weights $V_i \in \mathbb{R}^{(L_i+1)\times L_{i+1}}$ for all $i \in \{0, \ldots, k\}$, where $k \in \mathbb{Z}_{>0}$ denotes the number of hidden-layers in the DNN, $L_i \in \mathbb{Z}_{>0}$ for all $i \in \{0, \ldots, k+1\}$ denotes the number of neurons in each hidden layer, and $\sigma_i : \mathbb{R}^{L_i} \to \mathbb{R}^{L_i+1}$ for all $i \in \{1, \ldots, k\}$ denotes a vector of activation functions. The vector of activation functions can be composed of various activation functions, and hence, may be represented as $\sigma_i = [\varsigma_{L_i}, \ldots, \varsigma_1, 1]^T$ for all $i \in \{1, \ldots, k\}$, where $\varsigma_j : \mathbb{R} \to \mathbb{R}$ for all $j \in \{1, \ldots, L_i\}$ denotes a piecewise continuously differentiable activation function.[1] Note that for the subsequent function approximation of the unknown drift dynamics, the input and output dimensions are defined as $L_0 \triangleq L_{k+1} \triangleq n$.

Let $\mathbb{C}(\Omega)$ denote the space of continuous functions on the set Ω. By the universal function approximation theorem in [54, Thm 3.2], the function space of DNNs is dense in $\mathbb{C}(\Omega)$. Then for any function $f \in \mathbb{C}(\Omega)$ and prescribed function reconstruction error bound $\overline{\varepsilon} \in \mathbb{R}_{>0}$, there exist ideal DNN weights $\theta^* \triangleq \left[\text{vec}\left(V_k^*\right)^T, \ldots, \text{vec}\left(V_0^*\right)^T\right]^T \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ and activation functions σ_i for all $i \in \{1, \ldots, k\}$ such that $\sup_{x_d \in \Omega} \|f(x_d) - \Phi(x_d, \theta^*)\| \leq \overline{\varepsilon}$. Then, the DNN architecture in (5–2) models the unknown drift dynamics in (5–1) as

$$f(x_d) = \Phi(x_d, \theta^*) + \varepsilon(x_d), \qquad \forall x_d \in \Omega, \tag{5–3}$$

where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown bounded function approximation error. The function approximation error is bounded such that $\sup_{x_d \in \Omega} \|\varepsilon(x_d)\| \leq \overline{\varepsilon}$. Since the ideal DNN weights θ^* are unknown, real-time adaptive weight estimates $\hat{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ are generated to develop a DNN-based adaptive feedforward component $\Phi\left(x_d, \hat{\theta}\right)$,

[1] Some common choices in activation functions include the sigmoid, hyperbolic tangent, and ReLu activation functions.

where $\hat{\theta} \triangleq \left[\text{vec}\left(\hat{V}_k\right)^T, \ldots, \text{vec}\left(\hat{V}_0\right)^T\right]^T$. For brevity, the following short-hand notations are introduced

$$\Phi^* \triangleq \Phi\left(x_d, \theta^*\right), \quad \hat{\Phi} \triangleq \Phi\left(x_d, \hat{\theta}\right). \tag{5–4}$$

The DNN weight estimation error $\tilde{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ is defined as

$$\tilde{\theta}\left(t\right) \triangleq \theta^* - \hat{\theta}\left(t\right). \tag{5–5}$$

To facilitate the subsequent stability analysis, the following assumption is made.

Assumption 5.3. [31, Assumption 1] The ideal DNN weights can be bounded as $\|\theta^*\| \leq \overline{\theta}$, where $\overline{\theta} \in \mathbb{R}_{>0}$ denotes a known constant.

5.2 Control Development

This section introduces the DNN-based adaptive control design. Section 5.2.1 introduces the closed-loop error system resulting from the DNN-based adaptive controller. The DNN $\hat{\Phi}$ in (5–4) is used to develop an adaptive feedforward term that compensates for the system uncertainty. Then in Section 5.2.2, an accelerated gradient approach is used to design an adaptation law for $\hat{\theta}$ in (5–4) that generates real-time weight estimates for fully-connected DNNs with an arbitrary number of hidden-layers.

5.2.1 Closed-Loop Error System

Based on the subsequent stability analysis, the control input is designed as

$$u \triangleq g^+\left(x\right)\left[\dot{x}_d - k_e e - k_s \text{sgn}\left(e\right) - \hat{\Phi} - \rho\left(\|e\|\right)e + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right)\right], \tag{5–6}$$

where $k_e, k_s \in \mathbb{R}_{>0}$ denote user-defined parameters, $\hat{\nu} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}}$ denotes an auxiliary weight estimate that is subsequently defined, $\text{sgn}\left(\cdot\right)$ denotes the signum function, and $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a known strictly increasing function that satisfies $\|f\left(x\right) - f\left(x_d\right)\| \leq \rho\left(\|e\|\right)\|e\|$ for all $x, x_d \in \mathbb{R}^n$ [56, Lem. 5]. Taking the time derivative of (3–2), adding and subtracting $f\left(x_d\right)$, and substituting in (5–1) and (5–3), the open-loop

error system can be expressed as

$$\dot{e} = \Phi^* + \varepsilon\left(x_d\right) + f\left(x\right) - f\left(x_d\right) + d\left(t\right) - \dot{x}_d + g\left(x\right)u. \tag{5–7}$$

Substituting (5–6) into (5–7) yields the closed-loop error system

$$\dot{e} = -k_e e - k_s \text{sgn}\left(e\right) + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right) + \Phi^* - \hat{\Phi} + \varepsilon\left(x_d\right) + f\left(x\right) - f\left(x_d\right) + d\left(t\right) - \rho\left(\|e\|\right)e. \tag{5–8}$$

The first-order Taylor's series approximation of Φ^* yields [31, Eq. (22)]

$$\Phi^* - \hat{\Phi} = \hat{\Phi}'\tilde{\theta} + \mathcal{O}^2\left(\left\|\tilde{\theta}\right\|\right), \tag{5–9}$$

where $\mathcal{O}^2\left(\cdot\right)$ denotes higher-order terms resulting from the Taylor's series approximation.[2] Then substituting (5–9) into (5–8), the closed-loop error system can be expressed as

$$\dot{e} = -k_e e - k_s \text{sgn}\left(e\right) + \hat{\Phi}'\tilde{\theta} + 2\hat{\Phi}'\left(\hat{\theta} - \hat{\nu}\right) - \rho\left(\|e\|\right)e + \chi, \tag{5–10}$$

where $\chi : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes an auxiliary function defined as $\chi \triangleq \mathcal{O}^2\left(\left\|\tilde{\theta}\right\|\right) + \varepsilon\left(x_d\right) + f\left(x\right) - f\left(x_d\right) + d\left(t\right)$.

5.2.2 Accelerated Gradient-Based Adaptation

This section applies the variational framework in [19] to develop an accelerated higher-order adaptation law which generates real-time weight estimates for the DNN in (5–6). Unlike previous chapters, this section includes the variational framework in [19] to provide a constructive approach on accelerated gradient-based adaptation design. The variational approach in [19] defines a Bregman Lagrangian function and uses principles from the calculus of variations to generate a class of accelerated gradient update

[2] Given suitable functions f and g, the notation $f\left(x\right) = \mathcal{O}^k\left(g\left(x\right)\right)$ means that there exist constants $c, x_0 \in \mathbb{R}_{>0}$ such that $\|f\left(x\right)\| \leq c\left\|g\left(x\right)\right\|^k$ for all $x \geq x_0$.

laws that minimizes a cost functional. Similar to the approach in [19], the Bregman Lagrangian function $\mathscr{L} : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is defined as

$$\mathscr{L}\left(\hat{\theta}, \dot{\hat{\theta}}, t\right) = e^{\overline{\alpha}(t)+\overline{\gamma}(t)}\left(D_h\left(\hat{\theta} + e^{-\overline{\alpha}(t)}\dot{\hat{\theta}}, \hat{\theta}\right) - e^{\overline{\beta}(t)}E\left(\hat{\theta}\right)\right), \tag{5–11}$$

where $\overline{\alpha}, \overline{\beta}, \overline{\gamma} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denote arbitrary continuously differentiable scaling functions, $E : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}$ denotes a user-defined loss function, and $D_h : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \times \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ denotes the Bregman divergence defined as $D_h(p, q) \triangleq \frac{1}{2}\|q - p\|^2$.[3] The scaling functions in (5–11) are selected as $\overline{\alpha} \triangleq 0$, $\overline{\beta} \triangleq \ln(\gamma_1\gamma_2)$, and $\overline{\gamma}(t) \triangleq \gamma_2 t$, where $\gamma_1, \gamma_2 \in \mathbb{R}_{>0}$ are user-defined parameters [22, 24] (cf., [19]). Then, (5–11) can be expressed as $\mathscr{L}\left(\hat{\theta}, \dot{\hat{\theta}}, t\right) = e^{\gamma_2 t}\left(\frac{1}{2}\dot{\hat{\theta}}^T\dot{\hat{\theta}} - \gamma_1\gamma_2 E\left(\hat{\theta}\right)\right)$. Let a cost functional $J : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ be defined as $J\left(\hat{\theta}\right) \triangleq \int \mathscr{L}\left(\hat{\theta}, \dot{\hat{\theta}}, \tau\right) d\tau$. Then by the calculus of variations, the trajectories $t \mapsto \hat{\theta}(t)$ that minimize the cost functional J are the solutions of the Euler-Lagrange equation $\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\hat{\theta}}}\left(\hat{\theta}, \dot{\hat{\theta}}, t\right)\right) - \frac{\partial \mathscr{L}}{\partial \hat{\theta}}\left(\hat{\theta}, \dot{\hat{\theta}}, t\right) = 0$ [19, Eq. 3]. Then computing the terms $\frac{\partial \mathscr{L}}{\partial \dot{\hat{\theta}}}$, $\frac{\partial \mathscr{L}}{\partial \hat{\theta}}$, and $\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\hat{\theta}}}\right)$ in the Euler-Lagrange equation yields the higher-order adaptation law

$$\ddot{\hat{\theta}} + \gamma_2\dot{\hat{\theta}} = -\gamma_2\gamma_1\frac{\partial}{\partial \hat{\theta}}E\left(\hat{\theta}\right). \tag{5–12}$$

Based on the subsequent stability analysis, the loss function is defined as $E\left(\hat{\theta}\right) \triangleq \frac{d}{dt}\left(\frac{1}{2}e^T e\right)$. Using (5–10) yields $\frac{\partial E}{\partial \hat{\theta}} = \hat{\Phi}'^T e$, where $\hat{\Phi}' \triangleq \frac{\partial \hat{\Phi}}{\partial \hat{\theta}} \in \mathbb{R}^{n \times \sum_{i=0}^{k}(L_i+1)L_{i+1}}$. To facilitate the subsequent analysis, let $\gamma_1 \triangleq \gamma_\nu$ and $\gamma_2 \triangleq \gamma_\nu\gamma_\theta$, where $\gamma_\nu, \gamma_\theta \in \mathbb{R}_{>0}$ are user-defined parameters. Then by defining an auxiliary weight estimate as $\hat{\nu} \triangleq \hat{\theta} + \frac{1}{\gamma_2}\dot{\hat{\theta}}$, (5–12) can be expressed by two first-order differential equations as

[3] The formulation in [19] considers a non-Euclidean setting and defines the Bregman divergence as $D_h(p, q) = h(p) - h(q) - \nabla h(q)^T(p - q)$, where $h : \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} \to \mathbb{R}_{\geq 0}$ denotes a distance generating function. To obtain the Bregman divergence function defined in this chapter, $h(x) \triangleq \frac{1}{2}\|x\|^2$.

$$\dot{\hat{\nu}} \triangleq \text{proj}\left(\gamma_\nu \hat{\Phi}'^T e\right), \tag{5–13}$$

$$\dot{\hat{\theta}} \triangleq -\text{proj}\left(\gamma_\nu \gamma_\theta \left(\hat{\theta} - \hat{\nu}\right)\right), \tag{5–14}$$

where the operator $\text{proj}(\cdot)$ is used in (5–13) and (5–14) to ensure the weight estimates remain bounded in the subsequent stability analysis, i.e., $\hat{\nu}(t), \hat{\theta}(t) \in \Theta$ for all $t \in \mathbb{R}_{\geq 0}$, where $\Theta \triangleq \left\{\theta \in \mathbb{R}^{\sum_{i=0}^{k}(L_i+1)L_{i+1}} : \|\theta\| \leq \overline{\theta}\right\}$ denotes a known convex set and $\overline{\theta}$ is known by Assumption 5.3.

The term $\hat{\Phi}'$ in (5–13) can be computed as follows. To facilitate the subsequent development, let $\hat{\sigma}_i : \mathbb{R}^{L_i} \to \mathbb{R}^{L_i+1}$ be defined recursively as

$$\hat{\sigma}_i \triangleq \begin{cases} \sigma_1\left(\hat{V}_0^T \overline{x}_d\right), & i = 1, \\ \sigma_i\left(\hat{V}_{i-1}^T \hat{\sigma}_{i-1}\right), & i = 2, \ldots, k. \end{cases} \tag{5–15}$$

Using (5–4) and the recursive relation in (5–15), the DNN estimate $\hat{\Phi}$ can be expressed as $\hat{\Phi} = \hat{V}_k^T \hat{\sigma}_k$. Then recalling $\hat{\theta} = \left[\text{vec}\left(\hat{V}_k\right)^T, \ldots, \text{vec}\left(\hat{V}_0\right)^T\right]^T$, $\hat{\Phi}'$ can be expressed as

$$\hat{\Phi}' = \left[\frac{\partial \hat{\Phi}}{\partial \text{vec}\left(\hat{V}_k\right)}, \ldots, \frac{\partial \hat{\Phi}}{\partial \text{vec}\left(\hat{V}_0\right)}\right]. \tag{5–16}$$

Using the chain rule, the vectorization property in (1–1), and the facts that $\frac{\partial \hat{\sigma}_1}{\partial \text{vec}\left(\hat{V}_0\right)} = \hat{\sigma}_1'\left(I_{L_1} \otimes \overline{x}_d^T\right)$ and $\frac{\partial \hat{\sigma}_{i+1}}{\partial \text{vec}\left(\hat{V}_i\right)} = \hat{\sigma}_{i+1}'\left(I_{L_{i+1}} \otimes \hat{\sigma}_i^T\right)$ for all $i = 1, \ldots, k-1$, the terms in (5–16) can be computed as

$$\frac{\partial \hat{\Phi}}{\partial \text{vec}\left(\hat{V}_i\right)} = \begin{cases} \left(\prod_{j=1}^{\overset{\curvearrowleft}{k}} \hat{V}_j^T \hat{\sigma}_j'\right)\left(I_{L_1} \otimes \overline{x}_d^T\right), & i = 0, \\ \left(\prod_{j=i+1}^{\overset{\curvearrowleft}{k}} \hat{V}_j^T \hat{\sigma}_j'\right)\left(I_{L_{i+1}} \otimes \hat{\sigma}_i^T\right), & i = 1, \ldots, k, \end{cases} \tag{5–17}$$

where $\hat{\sigma}_1' \triangleq \frac{\partial \hat{\sigma}_1}{\partial \hat{V}_0^T \overline{x}_d}$ and $\hat{\sigma}_i' \triangleq \frac{\partial \hat{\sigma}_i}{\partial \hat{V}_{i-1}^T \hat{\sigma}_{i-1}}$ for $i = 2, \ldots, k$. The adaptation law in (5–13), (5–16), and (5–17) are expressed for fully-connected DNNs with an arbitrary number of hidden layers. To provide more insight, the following example is provided.

Example 1. Consider the single-hidden-layer NN $\hat{\Phi}\left(x_d, \hat{\theta}\right) = \hat{V}_1^T \sigma_1\left(\hat{V}_0^T \overline{x}_d\right)$ (i.e., $k = 1$), where $\hat{V}_0 \in \mathbb{R}^{(n+1)\times L}$, $\hat{V}_1 \in \mathbb{R}^{(L+1)\times n}$, $L \in \mathbb{Z}_{>0}$, and $\hat{\theta} \triangleq \left[\text{vec}\left(\hat{V}_1\right)^T, \text{vec}\left(\hat{V}_0\right)^T\right]^T \in \mathbb{R}^{2Ln+L+n}$. Then using (5–16) and (5–17), $\hat{\Phi}'$ can be computed as

$$\hat{\Phi}' = \left[\left(I_n \otimes \hat{\sigma}_1^T\right), \hat{V}_1^T \hat{\sigma}_1'\left(I_L \otimes \overline{x}_d^T\right)\right]. \tag{5–18}$$

Using (5–18), the update law in (5–13) yields

$$\dot{\hat{\nu}} = \text{proj}\left(\gamma_\nu \begin{bmatrix} \left(I_n \otimes \hat{\sigma}_1^T\right)^T \\ \left(\hat{V}_1^T \hat{\sigma}_1'\left(I_L \otimes \overline{x}_d^T\right)\right)^T \end{bmatrix} e\right). \tag{5–19}$$

Defining $\hat{\nu} \triangleq \left[\text{vec}\left(\hat{\nu}_1\right)^T, \text{vec}\left(\hat{\nu}_0\right)^T\right]^T$, using (1–1), and applying some algebraic manipulation, (5–19) can be expressed as

$$\dot{\hat{\nu}}_1 = \text{proj}\left(\gamma_\nu \hat{\sigma}_1 e^T\right), \tag{5–20}$$

$$\dot{\hat{\nu}}_0 = \text{proj}\left(\gamma_\nu \overline{x}_d e^T \hat{V}_1^T \hat{\sigma}_1'\right). \tag{5–21}$$

Remark 5. The DNN configuration in Example 1 is a special case where there is only one hidden layer, i.e., $k = 1$. This example provides insights and shows the generality of the adaptation law developed in (5–13). Note that, for the special case, the adaptation laws in (5–20) and (5–21) are equivalent to (3–4) and (3–7), respectively. The adaptation laws in (5–20) and (5–21) are equivalent to the output- and inner-layer weight adaptation laws developed in [31] for single-hidden-layer NNs. Hence, the adaptation law developed in (5–13) can be interpreted as a gradient-based adaptation law that has been generalized to fully-connected DNNs. However, the fundamental difference in the developed method is that the adaptation law in (5–13) generates

auxiliary weight estimates. The auxiliary estimates are coupled to the adaptation law in (5–14) which alters the search direction of the auxiliary estimates to generate the true DNN weight estimates.

Remark 6. The adaptation law developed in (5–12) is dependent on the selection of the loss function. The loss function $E = \frac{1}{2}\frac{d}{dt}\left(e^T e\right)$, defined below (5–12), was selected based on the subsequent stability analysis and yields the higher-order adaptation laws in (5–13) and (5–14). It is well known that including the weight estimation error $\tilde{\theta}$ in the adaptation law design can improve performance, and hence, including a term such as $\frac{1}{2}\tilde{\theta}^T\tilde{\theta}$ in the loss function can be beneficial. However, including the weight estimation error in the adaptation design poses challenges in implementation because the weight estimation error is unknown; therefore, selections in loss functions that incorporate a measurable/computable form of the weight estimation error may be a potential avenue for future works.

5.3 Stability Analysis

The closed-loop control design introduced in Section 5.2 employs a discontinuous robust sliding mode term and may also have potential discontinuities in $\hat{\Phi}'$ depending on the choice of activation functions (e.g., ReLU activation functions). As a result, the closed-loop system is nonsmooth and does not admit classical solutions. Hence, a nonsmooth Lyapunov-based analysis is used to analyze generalized solutions of the resulting closed-loop system and ensure the tracking objective is achieved [48] (cf., [1, Thm. A.8] for LaSalle-Yoshizawa invariance principles for smooth nonautonomous systems). Specifically, the switching analysis in [35] for DNN-based adaptive control is adopted to model the closed-loop system as a state-dependent switched system composed of a finite collection of smooth functions.

To facilitate the subsequent analysis, let $\varrho \in \mathcal{P}$ denote a switching index, where $\mathcal{P} \subset \mathbb{Z}$ denotes a finite set of possible switching indices. Then the DNN function approximation in (5–3) can be represented as $f\left(x_d\right) = \Phi_{\varrho}^* + \varepsilon_{\varrho}\left(x_d\right)$, where $x_d \mapsto \Phi_{\varrho}^*$

is smooth for each $\varrho \in \mathcal{P}$ with the corresponding function reconstruction error $\varepsilon_\varrho(x_d)$. Similarly, the function χ defined below (5–10) can be represented by the switched function

$$\chi_\varrho \triangleq \mathcal{O}_\varrho^2\left(\left\|\tilde{\theta}\right\|\right) + \varepsilon_\varrho(x_d) + f(x) - f(x_d) + d(t), \tag{5–22}$$

which is continuous for each $\varrho \in \mathcal{P}$. By the use of the projection algorithm in (5–14), the DNN weight estimates can be upper bounded as $\left\|\hat{\theta}(t)\right\| \leq \overline{\theta}$ for all $t \in \mathbb{R}_{\geq 0}$. Hence, by Assumption 5.3 and using (5–5), the DNN weight estimation error can be bounded as $\left\|\tilde{\theta}(t)\right\| \leq 2\overline{\theta}$ for all $t \in \mathbb{R}_{\geq 0}$. Moreover, since $\tilde{\theta} \in \mathcal{L}_\infty$ and $\mathcal{O}_\varrho^2$ is continuous in each $\varrho \in \mathcal{P}$, it follows that $\mathcal{O}_\varrho^2$ can be upper bounded by a known constant for all $\varrho \in \mathcal{P}$. The exogenous disturbance $t \mapsto d(t)$ is upper bounded by a known constant by Assumption 5.1. Then χ_ϱ can be upper bounded as

$$\|\chi_\varrho\| \leq c + \rho(\|e\|)\|e\|, \qquad \forall \varrho \in \mathcal{P}, \tag{5–23}$$

where $c \in \mathbb{R}_{>0}$ denotes a known constant and $\rho(\cdot)$ was defined below (5–6).

Based on the development introduced above, the adaptation law in (5–13) and the closed-loop error system in (5–10) can be represented as

$$\dot{\hat{\nu}} = \text{proj}\left(\gamma_\nu \hat{\Phi}_\varrho'^T e\right), \qquad \forall \varrho \in \mathcal{P}, \tag{5–24}$$

$$\dot{e} = -k_e e - k_s \text{sgn}(e) + \hat{\Phi}_\varrho' \tilde{\theta} + 2\hat{\Phi}_\varrho'\left(\hat{\theta} - \hat{\nu}\right) - \rho(\|e\|)e + \chi_\varrho, \qquad \forall \varrho \in \mathcal{P}. \tag{5–25}$$

For notational brevity, let $\Psi \in \mathbb{Z}_{>0}$ be defined as $\Psi \triangleq n + 2\sum_{i=0}^{k}(L_i + 1)L_{i+1}$. Let $z \triangleq \left[e^T, \left(\hat{\theta} - \hat{\nu}\right)^T, (\theta^* - \hat{\nu})^T\right]^T \in \mathbb{R}^\Psi$ denote a concatenated state, and let $\dot{z} = h_\varrho(z,t)$ for all $\varrho \in \mathcal{P}$ denote a collection of subsystems, where $h_\varrho : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$. Then the corresponding switched system is represented as

$$\dot{z} = h_{p(z)}(z,t), \tag{5–26}$$

where $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\Psi}$ denotes a Filippov solution to (5–26), $p : \mathbb{R}^{\Psi} \to \mathcal{P}$ denotes a state-dependent switching signal, and $h_{\varrho}(z,t)$ is defined as

$$h_{\varrho}(z,t) = \begin{bmatrix} f_{\varrho}^{cl}(z,t) \\ -\text{proj}\left(\gamma_{\nu}\gamma_{\theta}\left(\hat{\theta}-\hat{\nu}\right)\right) - \text{proj}\left(\gamma_{\nu}\hat{\Phi}_{\varrho}'^{T}e\right) \\ -\text{proj}\left(\gamma_{\nu}\hat{\Phi}_{\varrho}'^{T}e\right) \end{bmatrix}, \tag{5–27}$$

for all $\varrho \in \mathcal{P}$, where $f_{\varrho}^{cl} : \mathbb{R}^{\Psi} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n}$ is defined as $f_{\varrho}^{cl}(z,t) \triangleq -k_e e - k_s \text{sgn}(e) + \hat{\Phi}_{\varrho}'\tilde{\theta} + 2\hat{\Phi}_{\varrho}'\left(\hat{\theta}-\hat{\nu}\right) - \rho(\|e\|)e + \chi_{\varrho}$. In the following theorem, nonsmooth Lyapunov-based analysis techniques developed in [48] are used to establish invariance properties of (5–26) to ensure the tracking objective is achieved.

Theorem 7. *Consider a system modeled as in (5–1) and let Assumptions 5.1–5.3 hold. Then the control input in (5–6) and higher-order DNN weight adaptation laws in (5–13) and (5–14) ensure global asymptotic tracking in the sense that* $\lim_{t\to\infty}\|e(t)\| = 0$ *and* $\lim_{t\to\infty}\left\|\hat{\theta}(t)-\hat{\nu}(t)\right\| = 0$*, provided the following sufficient gain condition is satisfied*

$$k_s > c, \tag{5–28}$$

where c is a known constant defined in (5–23).

Proof. Consider a candidate common Lyapunov function $V_L : \mathbb{R}^{\Psi} \to \mathbb{R}_{\geq 0}$ defined as

$$V_L(z) \triangleq \frac{1}{2}e^T e + \frac{1}{2\gamma_{\nu}}\left(\hat{\theta}-\hat{\nu}\right)^T\left(\hat{\theta}-\hat{\nu}\right) + \frac{1}{2\gamma_{\nu}}(\theta^*-\hat{\nu})^T(\theta^*-\hat{\nu}), \tag{5–29}$$

which satisfies the inequality $\underline{\alpha}(\|z\|) \leq V_L(z) \leq \overline{\alpha}(\|z\|)$, where $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ are continuous positive definite functions. Let $F_{\varrho} : \mathbb{R}^{\Psi} \rightrightarrows \mathbb{R}^{\Psi}$ denote the Filippov regularization of (5–26) and be defined as $F_{\varrho} \triangleq \text{K}[h_{\varrho}](z,t)$, where the calculus of $\text{K}[\cdot]$ is defined in [57] and the notation $\rightrightarrows$ denotes a set-valued mapping. Then the generalized time derivative of (5–29) can be computed as $\dot{\overline{V}}_L \triangleq \max_{p\in\partial V_L(z)}\max_{q\in F_{\varrho}'(z)} p^T q$ [48, Def. 3], where $F_{\varrho}'(z,t) \supseteq F_{\varrho}(z,t)$ denotes a bound on the regularization of (5–26), and ∂V_L denotes Clarke's generalized gradient of V_L [58, pp. 39]. Since $z \mapsto V_L(z)$ for all $z \in \mathbb{R}^{\Psi}$ is

continuously differentiable, $\partial V_L(z) = \{\nabla V_L(z)\}$, where $\{\cdot\}$ denotes a singleton set and ∇ denotes the gradient. Additionally, the time derivative of V_L exists for almost all time, i.e., $\dot{V}_L(z(t)) \overset{\text{a.e.}}{\in} \dot{\overline{V}}_L(z(t))$, where the notation $\overset{\text{a.e.}}{(\cdot)}$ denotes that the relation holds for almost all time.

Taking the generalized time derivative of (5–29), using (5–5), adding and subtracting $\hat{\theta}^T \frac{1}{\gamma_\nu} \mathsf{K}\left[\dot{\hat{\nu}}\right]$, and performing some algebraic manipulation yields

$$\dot{\overline{V}}_L = e^T \mathsf{K}[\dot{e}] + \left(\hat{\theta} - \hat{\nu}\right)^T \frac{1}{\gamma_\nu} \mathsf{K}\left[\dot{\hat{\theta}}\right] - \left(\tilde{\theta} + 2\hat{\theta} - 2\hat{\nu}\right)^T \frac{1}{\gamma_\nu} \mathsf{K}\left[\dot{\hat{\nu}}\right]. \tag{5–30}$$

Substituting (5–14), (5–24), and (5–25) into (5–30) yields

$$\begin{aligned}\dot{\overline{V}}_L = e^T &\left(-k_e e - k_s \mathsf{K}[\mathsf{sgn}](e) - \rho(\|e\|) e + \mathsf{K}[\chi_\varrho] + \mathsf{K}\left[\hat{\Phi}'_\varrho\right] \tilde{\theta} + 2\mathsf{K}\left[\hat{\Phi}'_\varrho\right] \left(\hat{\theta} - \hat{\nu}\right)\right) \\ &- \left(\hat{\theta} - \hat{\nu}\right)^T \mathsf{K}[\mathsf{proj}]\left(\gamma_\theta \left(\hat{\theta} - \hat{\nu}\right)\right) - \left(\tilde{\theta} + 2\hat{\theta} - 2\hat{\nu}\right)^T \mathsf{K}[\mathsf{proj}]\left(\hat{\Phi}'^T_\varrho e\right).\end{aligned} \tag{5–31}$$

Since χ_ϱ and $\hat{\Phi}_\varrho$ are continuous functions for each $\varrho \in \mathcal{P}$, $\mathsf{K}[\chi_\varrho] = \{\chi_\varrho\}$ and $\mathsf{K}\left[\hat{\Phi}'_\varrho\right] = \left\{\hat{\Phi}'_\varrho\right\}$ for all $\varrho \in \mathcal{P}$. To bound the terms that involve $\mathsf{proj}(\cdot)$ in (5–31), Lemma E.1.IV in [1] is invoked which states $-r^T \Gamma \mathsf{proj}(\tau) \leq -r\Gamma\tau$ for all $r \in \mathcal{R} \subset \mathbb{R}^m$, $\Gamma \in \mathbb{R}^{m\times m}$, and $\tau \in \mathbb{R}^m$, where Γ denotes a positive definite matrix and $\mathcal{R}$ denotes a convex set. By the use of the projection algorithm in (5–13) and (5–14), the trajectories of $\hat{\nu}(t)$ and $\hat{\theta}(t)$, for all $t \in \mathbb{R}_{\geq 0}$, remain in the convex set Θ defined below (5–14). By Assumption 5.3, the ideal DNN weights can be bounded as $\|\theta^*\| \leq \overline{\theta}$. Moreover, note that $\mathsf{K}[\mathsf{proj}](\tau)$ computes the set of convex combinations of $\mathsf{proj}(\tau)$ and τ at the points of discontinuity. Therefore $-r^T \Gamma \mathsf{K}[\mathsf{proj}](\tau) \leq -r^T\Gamma\tau$, and hence, the terms with the $\mathsf{proj}(\cdot)$ operator in (5–31) can be upper bounded as

$$-\left(\hat{\theta} - \hat{\nu}\right)^T \mathsf{K}[\mathsf{proj}]\left(\gamma_\theta \left(\hat{\theta} - \hat{\nu}\right)\right) \leq -\gamma_\theta \left(\hat{\theta} - \hat{\nu}\right)^T \left(\hat{\theta} - \hat{\nu}\right), \tag{5–32}$$

$$\begin{aligned}-\left(\tilde{\theta} + 2\hat{\theta} - 2\hat{\nu}\right)^T \mathsf{K}[\mathsf{proj}]\left(\hat{\Phi}'_\varrho e\right) &\leq -\left(\tilde{\theta} + 2\hat{\theta} - 2\hat{\nu}\right)^T \hat{\Phi}'^T_\varrho e \\ &= -e^T \left(\hat{\Phi}'_\varrho \tilde{\theta} + 2\hat{\Phi}'_\varrho \left(\hat{\theta} - \hat{\nu}\right)\right).\end{aligned} \tag{5–33}$$

Then using (5–23), (5–32), (5–33), and the facts that $e^T \mathrm{K} [\mathrm{sgn}] (e) = \{\|e\|_1\}$ and $-k_s \|e\|_1 \leq -k_s \|e\|$, (5–31) can be upper bounded as

$$\dot{\overline{V}}_L \overset{\text{a.e.}}{\leq} -k_e \|e\|^2 - (k_s - c) \|e\| - \gamma_\theta \left\| \hat{\theta} - \hat{\nu} \right\|^2. \tag{5–34}$$

Provided the sufficient gain condition in (5–28) is satisfied, (5–34) can be upper bounded as $\dot{\overline{V}}_L \overset{\text{a.e.}}{\leq} -k_e \|e\|^2 - \gamma_\theta \left\| \hat{\theta} - \hat{\nu} \right\|^2$. From (5–29) and the fact that $\dot{\overline{V}}_L \overset{\text{a.e.}}{\leq} 0$, it follows that $V_L \in \mathcal{L}_\infty$, which implies $z \in \mathcal{L}_\infty$, and hence $e, \hat{\nu}, \hat{\theta} \in \mathcal{L}_\infty$. Using (3–2), the fact that $e, x_d \in \mathcal{L}_\infty$ implies $x \in \mathcal{L}_\infty$. Using (5–5), the fact that $\theta^*, \hat{\theta} \in \mathcal{L}_\infty$ implies $\tilde{\theta} \in \mathcal{L}_\infty$. The fact that $x_d, \hat{\theta} \in \mathcal{L}_\infty$ implies $\hat{\Phi}, \hat{\Phi}' \in \mathcal{L}_\infty$. Using (5–6), the fact that $x, \dot{x}_d, e, \hat{\Phi}, \hat{\Phi}', \hat{\theta}, \hat{\nu} \in \mathcal{L}_\infty$ implies $u \in \mathcal{L}_\infty$. Using (5–13), the fact that $\hat{\Phi}', e \in \mathcal{L}_\infty$ implies $\dot{\hat{\nu}} \in \mathcal{L}_\infty$. Using (5–14), the fact that $\hat{\theta}, \hat{\nu} \in \mathcal{L}_\infty$ implies $\dot{\hat{\theta}} \in \mathcal{L}_\infty$. Invoking the LaSalle-Yoshizawa theorem extension for nonsmooth systems in [48, Thm. 2] yields $\lim_{t \to \infty} \|e(t)\| = 0$ and $\lim_{t \to \infty} \left\| \hat{\theta}(t) - \hat{\nu}(t) \right\| = 0$. ■

5.4 Simulations

Two comparative simulations were conducted on a two-state nonlinear system to demonstrate the performance of the developed method. The unknown drift dynamics $f(x)$ in (5–1) was modeled by (3–32) where $x \triangleq [x_1, x_2]^T \in \mathbb{R}^2$ denotes the system state. The control effectiveness was modeled as $g = I_2$. The exogenous disturbance in (5–1) was modeled as random noise drawn from the distribution $\mathcal{N}(0, 1)$. Each simulation was conducted for 30 s with initial condition $x(0) = [5.0, 7.0]^T$. The desired trajectory $x_d \triangleq [x_{d,1}, x_{d,2}]^T$ was selected as $x_d(t) = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$ for all $t \in \mathbb{R}_{\geq 0}$. The DNN used in both simulations was configured with $L = 10$ neurons in each of the $k = 5$ hidden layers. The hyperbolic tangent activation function was used across all layers. The first simulation was performed with a baseline DNN adaptive controller which used a typical gradient-based scheme given by [35]

$$u \triangleq \dot{x}_d - k_e e - k_s \mathrm{sgn}(e) - \hat{\Phi} - \rho(\|e\|) e, \tag{5–35}$$

$$\dot{\hat{\theta}} \triangleq \text{proj}\left(\gamma_\theta \hat{\Phi}'^T e\right), \tag{5–36}$$

where $e \triangleq [e_1, e_2]^T$, $\hat{\theta} \triangleq \left[\text{vec}\left(\hat{V}_5\right)^T, \ldots, \text{vec}\left(\hat{V}_0\right)^T\right]^T$, and $\hat{\Phi}'$ was computed using (5–16) and (5–17). The second simulation was performed with the developed method in (5–6), (5–13), and (5–14). The bound for the projection operator was selected as $\overline{\theta} = 1,000$. The robust state feedback term $\rho\left(\|e\|\right)$ in (5–6) was designed as $0.1\left(\|e\| + \|e\|^2\right)$ in both simulations were omitted to better focus on the performance resulting from the DNN-based adaptive terms. The DNN weight estimates $\hat{\theta}\left(0\right)$ (and also $\hat{\nu}\left(0\right)$ in the second simulation) were initialized randomly from the normal distribution $\mathcal{N}\left(0, 1\right)$. The controllers in each simulation were configured similarly and are summarized in Table 5-1.

Table 5-1. Controller Configuration

Parameters	DNN Adaptive [35]	Developed Method
Hidden layers, k	5	5
Neurons, L	5	5
Activation function	Hyperbolic tangent	Hyperbolic tangent
γ_ν	-	40
γ_θ	40	0.6
k_s	0.1	0.1
k_e	5	5

Table 5-2. RMS Tracking Error, Control Effort, and Function Approximation Error

RMS	DNN Adaptive	Developed Method
$\overline{e}$	0.0851	0.0277
$\overline{u}$	4.2761	2.6033
$\overline{\tilde{f}}$	2.9878	0.6029

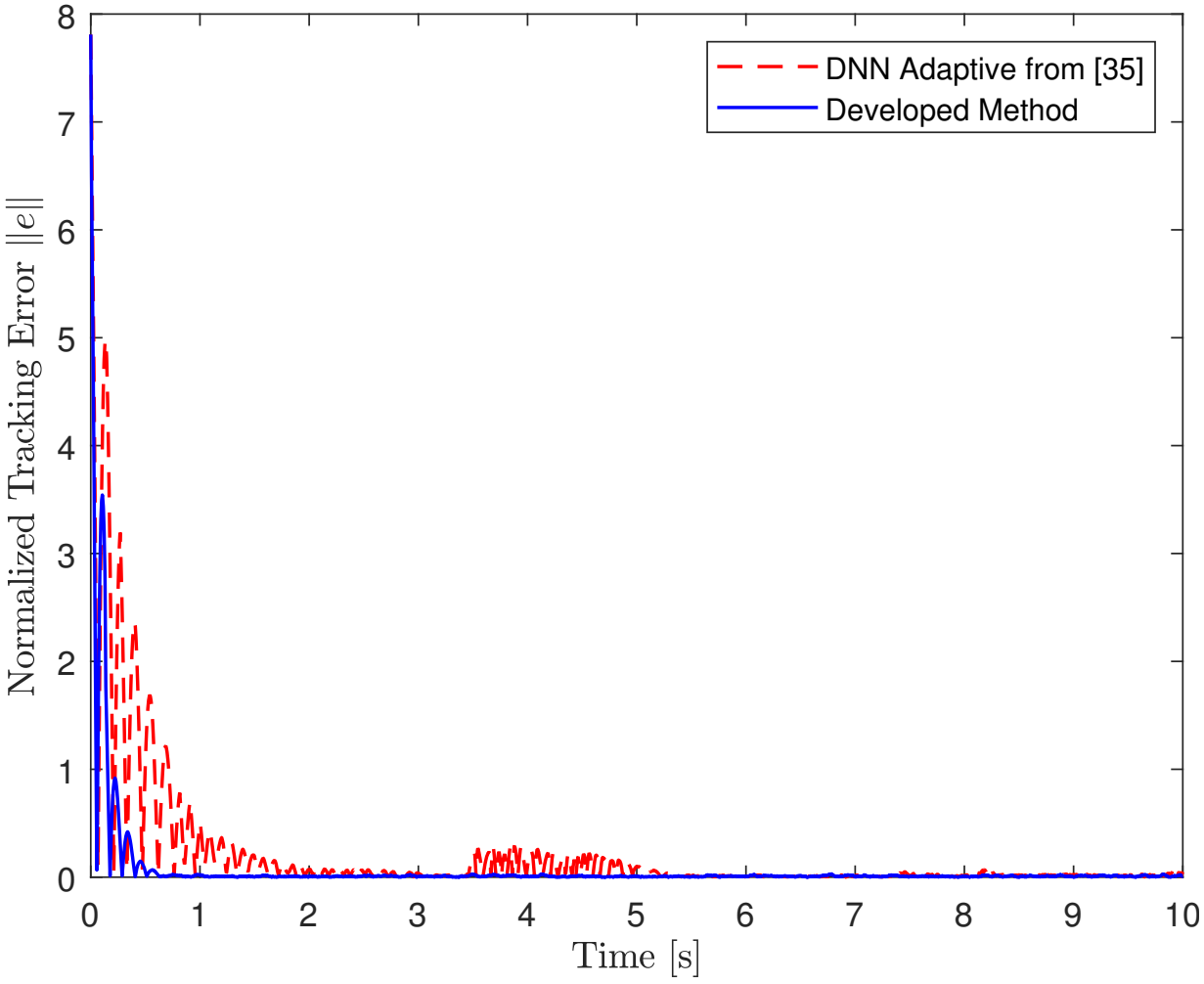

Figure 5-1. Evolution of the normalized tracking errors $\|e\|$. The simulation with the gradient-based DNN adaptation scheme in (5–35) and (5–36) is shown as a dashed red line, and the simulation using the developed higher-order DNN-based adaptation scheme in (5–13) and (5–14) is shown as a solid blue line. The tracking errors are shown over a 10 second window rather than the entirety of the simulation to better exhibit the transient performance.

Table 5-2 summarizes the performance in each simulation. In the leftmost column, $\overline{e}$, $\overline{u}$, and $\overline{\tilde{f}}$ denote the root mean square (RMS) tracking error, control effort, and function approximation error, respectively. As shown in Figure 5-1, the tracking errors converge towards the origin. The tracking errors using the baseline DNN adaptive controller in the first simulation converged to a neighborhood of the origin after approximately 5.4 s, whereas the tracking errors using the developed method converged after approximately 0.7 s. The RMS tracking error for the baseline DNN and developed methods were 0.0851 and 0.0277, respectively. The developed method had a 67.41% decrease in the RMS tracking error with a 39.12% decrease in the RMS control effort in comparison to the baseline DNN adaptive method.

Figure 5-2 illustrates the normalized function approximation error for each simulation. The simulation with the developed method showed a 78.82% decrease in the RMS function approximation error in comparison to the simulation with the baseline DNN adaptive controller. To better illustrate the improved transient performance from the developed higher-order adaptation, Figure 5-3 illustrates the evolution of the DNN weight estimates. The weight estimates from the baseline DNN adaptation exhibited oscillatory behavior which degraded the tracking and function approximation performance. As seen at approximately 3.9 s, and many other instances throughout the simulation, many of the weights in the baseline simulation had oscillations which resulted in increases to the tracking and function approximation errors shown in Figures 5-1 and 5-2, respectively. In comparison, the weight estimates from the developed method had improved transient performance as seen by the rapid convergence in the weight estimates and function approximation error after approximately 1.1 s.

5.5 Conclusion

Motivated by the potential improvements in transient performance, an accelerated gradient approach was used to design an accelerated gradient-based DNN adaptive control scheme for trajectory tracking of uncertain nonlinear systems. This chapter

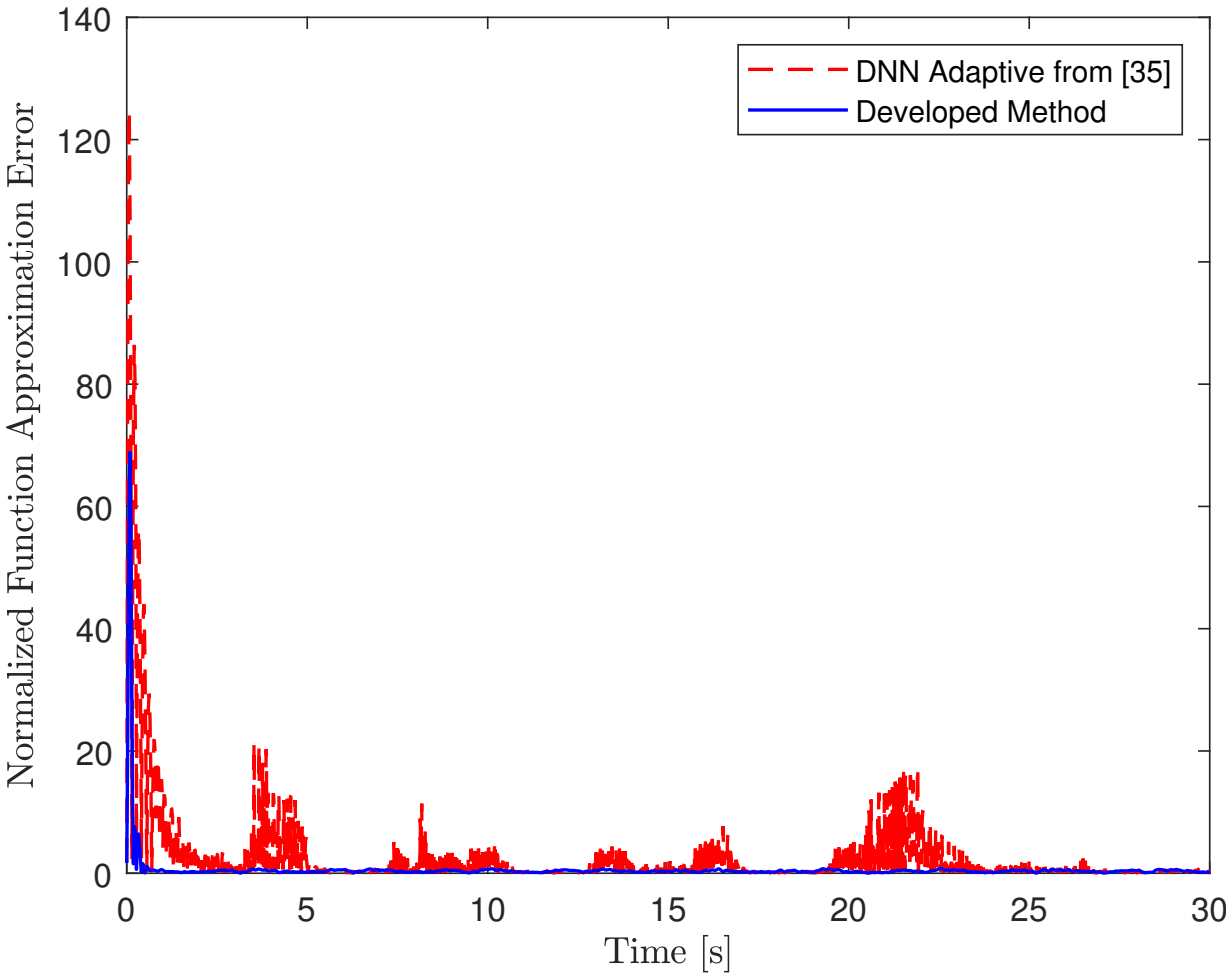

Figure 5-2. Evolution of the normalized function approximation errors $\left\| f\left(x_d\right) - \hat{f}\left(x_d\right) \right\|$ for each simulation.

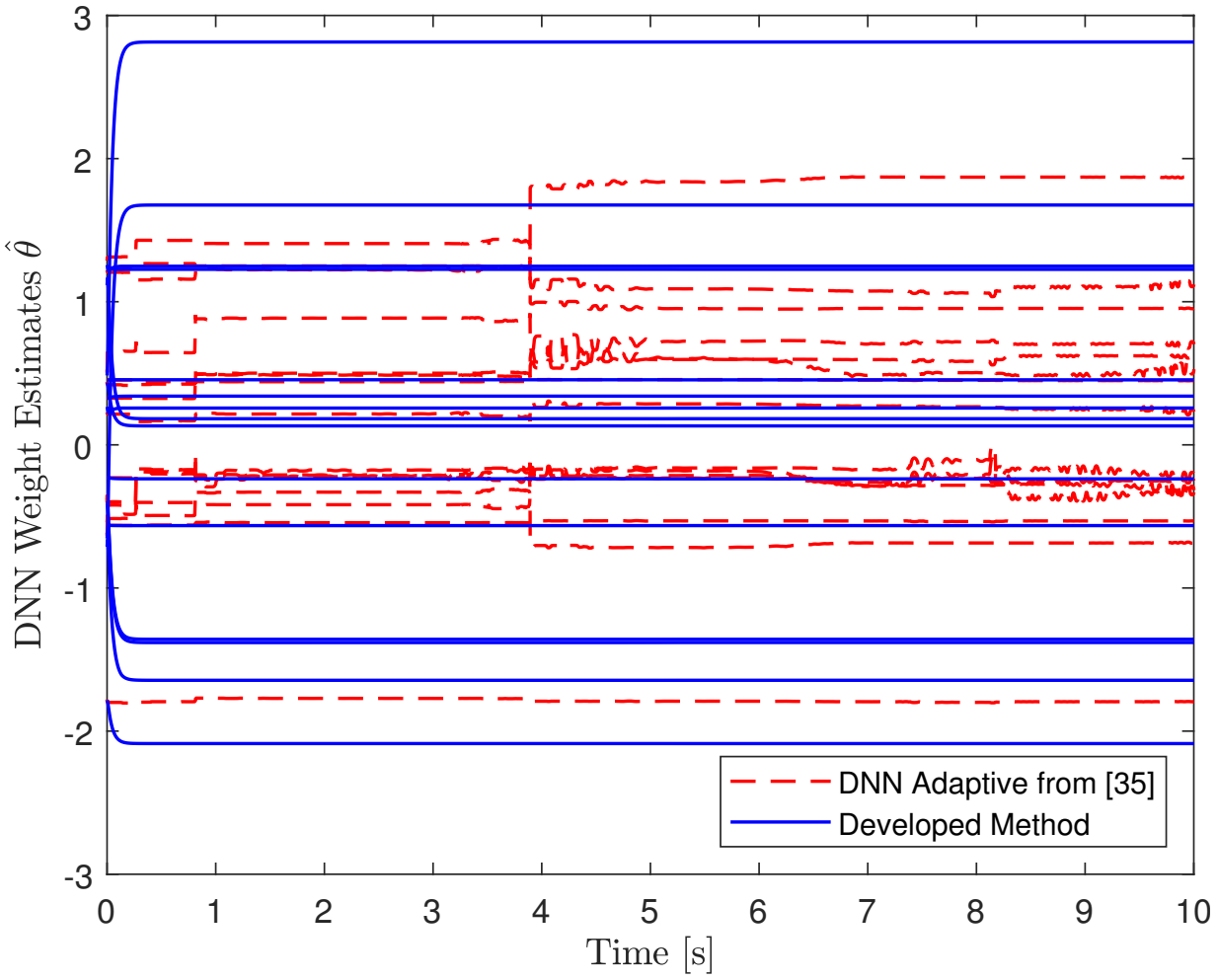

Figure 5-3. Evolution of the DNN weight estimates in each simulation. Each of the corresponding weights were initialized at the same initial condition. To better exhibit the performance in the weight estimates, for both simulations, only 15 of the DNN weight estimates are shown over a 10 second window rather than the entirety of the simulation.

leverages the insights on the development and analysis of DNN-based adaptation via Lyapunov-based techniques in Chapter 4 to extend the accelerated gradient-based NN adaptive control scheme in Chapter 3. Specifically, the NN architectures in Chapter 3 are generalized to account for fully-connected DNN architectures with an arbitrary number of hidden layers. The generalized development was analyzed in this chapter with a nonsmooth Lyapunov-based analysis to show the developed methods yields global asymptotic tracking. Comparative simulations were conducted on a two-state nonlinear system to demonstrate the improved performance from the developed accelerated gradient-based adaptation design. The simulations showed the developed accelerated gradient-based adaptation outperformed the DNN gradient adaptive scheme in [35] and had a 67.41% and 78.82% decrease in the RMS tracking and function approximation errors, respectively.

Future work may involve analyzing new adaptation designs from the selection of the loss function in (5–12). As discussed in Remark 6, selecting a loss function that incorporates computable versions of the function approximation error in the adaptation design may improve performance. However, there are challenges in the analysis and implementation of such adaptation laws due to the unknown nature of the ideal DNN weights.

CHAPTER 6
CONCLUSION

Adaptive control is a common technique used in nonlinear systems to achieve a control objective, such as trajectory tracking, while also learning parametric model uncertainties in real-time. Accelerated gradient-based optimization methods have gained significant interest due to their improved transient performance and faster convergence rates. Accelerated gradient-based methods are discrete-time algorithms that alter their search direction by using a weighted sum from the previous iteration to add a momentum-based term and accelerate convergence. Recent results have made connections between discrete-time accelerated gradient methods and continuous-time analogues. These connections have led to new insights on continuous-time algorithm design based on accelerated gradient methods. This dissertation developed new accelerated gradient-based adaptive control schemes via Lyapunov-based techniques to achieve trajectory tracking in general uncertain nonlinear dynamical systems.

First, in Chapter 2, an ICL-based accelerated gradient adaptive update law was developed to achieve trajectory tracking and real-time parameter identification for general uncertain Euler-Lagrange systems. The system uncertainties in this chapter were assumed to be LIP. The data-driven ICL method used samples data and injects a measurable form of the parameter estimation error into the adaptation law. To eliminate the need for direct measurements of the parameter estimation error, a torque-filtering method was used to reconstruct a measurable form of the parameter estimation error. A Lyapunov-based method was used to guarantee the closed-loop error system achieves global exponential stability under a less restrictive finite excitation condition. A comparative simulation study was performed on a two-link robot which showed the higher-order scheme outperformed standard and ICL-based adaption by 19.6% and 11.1%, respectively, in terms of the root mean squared parameter estimation errors.

Then to eliminate the restrictive LIP assumption, Chapter 3 generalized the development in Chapter 2 to account for general nonlinear systems subject to unstructured or unknown (non-LIP) uncertainty. To compensate for the non-LIP uncertainties, NN-based adaptive control techniques were used to approximate the uncertainty with a real-time adaptive NN-based feedforward term. Specifically, Chapter 3 developed an accelerated gradient-based NN adaptation scheme to achieve trajectory tracking in general nonlinear control affine systems subject to non-LIP uncertainties. The higher-order accelerated gradient-based adaptation laws were developed to generate real-time estimates of both the unknown ideal output- and hidden-layer weights of a NN. A non-smooth Lyapunov-based method was used to analyze and guarantee the closed-loop error system achieves global asymptotic tracking. Comparative simulations were conducted, and the results showed the higher-order adaptation outperformed the standard gradient-based NN adaptation by 32.3% in terms of the root mean squared function approximation error.

The development in Chapter 3 was limited to NN architectures that contain only a single hidden layer. DNN architectures have improved function approximation capabilities. However, in the context of Lyapunov-based methods, DNN-based adaptive control have been challenged by the nested nonlinear parametrized structure inherent to DNNs. Hence to focus on the unique challenges of DNN-based adaptive control, Chapter 4 used a modular approach to develop general constraints on the DNN weight adaptation laws for general nonlinear control affine systems. The modular adaptive constraints provided design guidelines for real-time DNN adaptation while also ensuring stability in the tracking objective. A nonsmooth Lyapunov-based analysis was used to guarantee semi-global asymptotic tracking. Additionally, simulations were conducted to demonstrate the improved performance and utility from DNN adaptation. However, the development in Chapter 4 did not provide a constructive method to design DNN-based adaptation laws.

Hence, Chapter 5 leveraged the insights developed from Chapter 4 on the development and analysis of DNN adaptation to build on the accelerated gradient approach in Chapter 3 to develop a new accelerated gradient-based DNN adaptation law. A generalized DNN architecture with an arbitrary number of hidden layers was considered, and the new accelerated gradient-based DNN adaptation scheme was used to generate real-time estimates of all the DNN weights. A nonsmooth Lyapunov-based analysis was used to guarantee the developed accelerated gradient-based DNN adaptation design achieves global asymptotic tracking error convergence for general nonlinear control-affine systems subject to non-LIP drift dynamics and exogenous disturbances. Simulations were conducted, and the results showed the developed accelerated gradient-based DNN adaptation outperformed gradient-based DNN adaptation by 67.41% and 78.82% in terms of the root mean squared tracking and function approximation errors, respectively.

Future work may involve analyzing new adaptation designs based on the selection of the loss function in (5–12). As discussed in Remark 6, selecting a loss function that incorporates the function approximation error in the adaptation design may improve performance. Due to the unknown nature of the ideal DNN weights, direct measurements of the function approximation error are typically not available. Hence, the torque filtering method in Chapter 3 could be used to generate a measurable/computable form of the function approximation error. However, there are challenges to using the torque filtering method as this method relies on the structured uncertainty from Assumption 2.1 to leverage algebraic relations and reconstruct the parameter estimation error. Moreover, it is not clear on how to select the loss function, and changes in the loss function yield new adaptation laws that require analysis to ensure the stability result is maintained. Hence, future works may also include using the modular approach from Chapter 4 to analyze the accelerated adaptation design in (5–12) for a broad class of loss functions that include the function approximation error.

REFERENCES

[1] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, *Nonlinear and Adaptive Control Design*. New York, NY, USA: John Wiley & Sons, 1995.

[2] P. Ioannou and J. Sun, *Robust Adaptive Control*. Prentice Hall, 1996.

[3] K. Narendra and A. Annaswamy, *Stable Adaptive Systems*. Prentice-Hall, Inc., 1989.

[4] J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[5] S. B. Roy, S. Bhasin, and I. N. Kar, "A UGES switched MRAC architecture using initial excitation," in *IFAC-PapersOnLine*, vol. 50, no. 1, Jul. 2017, pp. 7044–7051.

[6] ——, "Combined MRAC for unknown MIMO LTI systems with parameter convergence," *IEEE Trans. Autom. Control*, vol. 63, no. 1, pp. 283–290, Jan. 2018.

[7] N. Cho, H.-S. Shin, Y. Kim, and A. Tsourdos, "Composite model reference adaptive control with parameter convergence under finite excitation," *IEEE Trans. Autom. Control*, vol. 63, pp. 811–818, 2017.

[8] R. Ortega, S. Aranovskiy, A. A. Pyrkin, A. Astolfi, and A. A. Bobtsov, "New results on parameter estimation via dynamic regressor extension and mixing: Continuous and discrete-time cases," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2265–2272, 2021.

[9] G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *J. Guid. Control Dynam.*, vol. 34, no. 2, pp. 592–607, Mar. 2011.

[10] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adapt. Control Signal Process.*, vol. 27, no. 4, pp. 280–301, 2013.

[11] R. Kamalapurkar, B. Reish, G. Chowdhary, and W. E. Dixon, "Concurrent learning for parameter estimation using dynamic state-derivative estimators," *IEEE Trans. Autom. Control*, vol. 62, no. 7, pp. 3594–3601, July 2017.

[12] A. Parikh, R. Kamalapurkar, and W. E. Dixon, "Integral concurrent learning: Adaptive control with parameter convergence using finite excitation," *Int J Adapt Control Signal Process*, vol. 33, no. 12, pp. 1775–1787, Dec. 2019.

[13] M. Krstić, P. V. Kokotović, and I. Kanellakopoulos, "Transient-performance improvement with a new class of adaptive controllers," *Syst. Control Lett.*, vol. 21, no. 6, pp. 451–461, 1993.

[14] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "On adaptive control with closed-loop reference models: transients, oscillations, and peaking," *IEEE Access*, vol. 1, pp. 703–717, 2013.

[15] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2003, vol. 87.

[16] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O\left(\frac{1}{k^2}\right)$," in *Soviet Mathematics Doklady*, vol. 269, 1983, pp. 543–547.

[17] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. & Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[18] W. Su, S. Boyd, and E. J. Candes, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 5312–5354, 2016.

[19] A. Wibisono, A. C. Wilson, and M. I. Jordan, "A variational perspective on accelerated methods in optimization," *Proc. Natl. Acad. Sci.*, vol. 113, no. 47, pp. E7351–E7358, 2016.

[20] A. C. Wilson, B. Recht, and M. I. Jordan, "A Lyapunov analysis of accelerated methods in optimization," *J. Mach. Learn. Res.*, vol. 22, no. 113, pp. 1–34, 2021.

[21] H. Attouch and F. Alvarez, "The heavy ball with friction dynamical system for convex constrained minimization problems," in *Optimization*. Springer, 2000, pp. 25–35.

[22] J. E. Gaudio, A. M. Annaswamy, M. A. Bolender, E. Lavretsky, and T. E. Gibson, "A class of high order tuners for adaptive systems," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 391–396, 2020.

[23] D. E. Ochoa, J. I. Poveda, A. Subbaraman, G. S. Schmidt, and F. R. Pour-Safaei, "Accelerated concurrent learning algorithms via data-driven hybrid dynamics and nonsmooth odes," in *Proc. Conf. Learn. Dyn. Control*. PMLR, 2021, pp. 866–878.

[24] N. M. Boffi and J.-J. E. Slotine, "Implicit regularization and momentum algorithms in nonlinearly parameterized adaptive control and prediction," *Neural Computation*, vol. 33, no. 3, pp. 590–673, 2021.

[25] D. M. Le, O. S. Patil, P. Amy, and W. E. Dixon, "Integral concurrent learning-based accelerated gradient adaptive control of uncertain euler-lagrange systems," in *Proc. Am. Control Conf.*, Jun. 2022.

[26] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1985.

[27] N. E. Cotter, "The Stone-Weierstrass theorem and its application to neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 4, pp. 290–295, 1990.

[28] M. H. Stone, “The generalized Weierstrass approximation theorem,” *Math. Mag.*, vol. 21, no. 4, pp. 167–184, 1948.

[29] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

[30] F. L. Lewis, “Neural network control of robot manipulators,” *IEEE Expert*, vol. 11, no. 3, pp. 64–75, 1996.

[31] F. L. Lewis, A. Yegildirek, and K. Liu, “Multilayer neural-net robot controller with guaranteed tracking performance,” *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, Mar. 1996.

[32] M. Chen, S. S. Ge, and B. V. E. How, “Robust adaptive neural network conrtol for a class of uncertain MIMO nonlinear systems with input nonlinearities,” *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 796–812, May 2010.

[33] R. Sun, M. Greene, D. Le, Z. Bell, G. Chowdhary, and W. E. Dixon, “Lyapunov-based real-time and iterative adjustment of deep neural networks,” *IEEE Control Syst. Lett.*, vol. 6, pp. 193–198, 2022.

[34] D. Le, M. Greene, W. Makumi, and W. E. Dixon, “Real-time modular deep neural network-based adaptive control of nonlinear systems,” *IEEE Control Syst. Lett.*, vol. 6, pp. 476–481, 2022.

[35] O. Patil, D. Le, M. 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 Syst Lett.*, vol. 6, pp. 1855–1860, 2022.

[36] D. M. Le, O. S. Patil, C. Nino, and W. E. Dixon, “Accelerated gradient approach for neural network-based adaptive control of nonlinear systems,” in *Proc. IEEE Conf. Decis. Control*, 2022.

[37] F. L. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and nonlinear systems*. Philadelphia, PA: CRC Press, 1998.

[38] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[39] G. Joshi and G. Chowdhary, “Deep model reference adaptive control,” in *Proc. IEEE Conf. Decis. Control*. IEEE, 2019, pp. 4601–4608.

[40] G. Joshi, J. Virdi, and G. Chowdhary, “Design and flight evaluation of deep model reference adaptive controller,” in *AIAA Scitech 2020 Forum*, 2020, p. 1336.

[41] ——, “Asynchronous deep model reference adaptive control,” *Proc. PMLR Conf. Robot Learn.*, pp. 4601–4608, November 2020.

[42] P. Patre, W. Mackunis, K. Dupree, and W. E. Dixon, "Modular adaptive control of uncertain Euler-Lagrange systems with additive disturbances," *IEEE Trans. Autom. Control*, vol. 56, no. 1, pp. 155–160, 2011.

[43] M. S. De Queiroz, D. M. Dawson, and M. Agarwal, "Adaptive control of robot manipulators with controller/update law modularity," *Automatica*, vol. 35, pp. 1379–1390, 1999.

[44] W. E. Dixon, M. S. de Queiroz, D. M. Dawson, and T. J. Flynn, "Adaptive tracking and regulation control of a wheeled mobile robot with controller/update law modularity," *IEEE Trans. Control Syst. Technol.*, vol. 12, pp. 138–147, 2004.

[45] M. Krstic and P. V. Kokotovic, "Adaptive nonlinear design with controller-identifier separation and swapping," *IEEE Trans. Autom. Control*, vol. 40, no. 3, pp. 426–440, Mar. 1995.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] H. D. Patino, R. Carelli, and B. R. Kuchen, "Neural networks for advanced control of robot manipulators," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 343–354, 2002.

[48] R. Kamalapurkar, J. A. Rosenfeld, A. Parikh, A. R. Teel, and W. E. Dixon, "Invariance-like results for nonautonomous switched systems," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 614–627, Feb. 2019.

[49] O. S. Patil, D. M. Le, E. Griffis, and W. E. Dixon, "Deep residual neural network (ResNet)-based adaptive control: A Lyapunov-based approach," in *Proc. IEEE Conf. Decis. Control*, 2022.

[50] D. Bernstein, *Matrix Mathematics*. Princeton Univ Pr, 2005.

[51] R. Ortega, A. Loría, P. J. Nicklasson, and H. J. Sira-Ramirez, *Passivity-based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*. Springer, 1998.

[52] F. L. Lewis, D. M. Dawson, and C. Abdallah, *Robot Manipulator Control Theory and Practice*. CRC, 2003.

[53] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[54] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conf. Learn. Theory*. PMLR, 2020, pp. 2306–2327.

[55] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. The Institution of Engineering and Technology, 2013.

[56] R. Kamalapurkar, J. A. Rosenfeld, J. Klotz, R. J. Downey, and W. E. Dixon. (2014) Supporting lemmas for RISE-based control methods. arXiv:1306.3432.

[57] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," *IEEE Trans. Circuits Syst.*, vol. 34, no. 1, pp. 73–82, Jan. 1987.

[58] F. H. Clarke, *Optimization and nonsmooth analysis*. SIAM, 1990.

[59] D. S. Bernstein, *Matrix mathematics*. Princeton university press, 2009.

[60] M. L. Greene, P. Deptula, S. Nivison, and W. E. Dixon, "Sparse learning-based approximate dynamic programming with barrier constraints," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 743–748, Jul. 2020.

[61] G. Joshi, J. Virdi, and G. Chowdhary, "Asynchronous deep model reference adaptive control," in *Conf. Robot Learn.*, 2020.

[62] P. M. Patre, K. Dupree, W. MacKunis, and W. E. Dixon, "A new class of modular adaptive controllers, part ii: Neural network extension for non-lp systems," in *Proc. Am. Control Conf.*, Seattle, WA, USA, Jun. 2008, pp. 1214–1219.

[63] D. Shevitz and B. Paden, "Lyapunov stability theory of nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 39 no. 9, pp. 1910–1914, 1994.

[64] N. Fischer, R. Kamalapurkar, and W. E. Dixon, "LaSalle-Yoshizawa corollaries for nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2333–2338, Sep. 2013.

[65] R. Kamalapurkar, J. Rosenfeld, and W. E. Dixon, "Efficient model-based reinforcement learning for approximate online optimal control," *Automatica*, vol. 74, pp. 247–258, Dec. 2016.

[66] P. J. Werbos, "Back propagation: past and future," in *Proc. Int. Conf. Neural Netw.*, vol. 1, 1989, pp. 1343–1353.

[67] D. O. Hebb, *The organization of behavior*. Wiley Inc., New-York, 1949.

[68] K.-A. Toh, "Analytic network learning," *arXiv preprint arXiv:1811.08227*, 2018.

[69] H.-T. Nguyen, C. C. Cheah, and K.-A. Toh, "An analytic layer-wise deep learning framework with applications to robotics," *arXiv preprint arXiv:2102.03705*, 2021.

BIOGRAPHICAL SKETCH

Duc Minh Le was born in Sarasota, Florida in 1996. He attended the University of Florida where he received dual Bachelor's of Science (B.S.) degrees in mechanical and aerospace engineering. In January 2019, Duc joined the Nonlinear Controls and Robotics (NCR) laboratory under the guidance of Dr. Warren Dixon to pursue his doctoral studies. He received his Master of Science (M.S.) degree in Mechanical Engineering in May 2020, and is currently pursuing a Ph.D. in the department of mechanical and aerospace engineering at the University of Florida. His research focuses on development and application of Lyapunov-based adaptive control techniques.