ADAPTIVE DEEP LEARNING FOR CONTROL OF UNKNOWN SYSTEMS

By

EMILY J. GRIFFIS

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

2024

To mother, Karen, my father, Michael, and my siblings, Jacklyn and Daniel

ACKNOWLEDGMENTS

I thank my advisor, Dr. Warren Dixon, for his mentorship and inspiration during my PhD career. Dr. Dixon has continuously given invaluable guidance, unwavering encouragement, and a relentless challenge to exceed my goals throughout the course of my dissertation. I am also profoundly thankful to my dissertation committee members, Dr. Scott Banks, Dr. Amor Menezes, and Dr. Alina Zare, for their advice, insights, and generous dedication to my research endeavors. Additionally, I would like to express my heartfelt appreciation to my family and friends for their unconditional love, empowerment, and support throughout my doctoral pursuits. Their belief in me has been a constant source of strength and motivation. I am truly grateful for their help and encouragement. This dissertation would not have been possible without the love and guidance of these remarkable individuals, and for that, I am profoundly grateful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

DNN	Deep Neural Network
DRNN	Deep Recurrent Neural Network
GRU	Gated Recurrent Unit
Lb	Lyapunov-based
LSTM	Long Short-Term Memory
NN	Neural Network
OFB	Output Feedback
RL	Reinforcement Learning
RNN	Recurrent Neural Network
UUB	Uniformly Ultimately Bounded

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

ADAPTIVE DEEP LEARNING FOR CONTROL OF UNKNOWN SYSTEMS

By

Emily J. Griffis

May 2024

Chair: Warren E. Dixon
Major: Mechanical Engineering

Adaptive neural network (NN)-based controllers have become increasingly popular in recent years due to their real-time function approximation capabilities. While most adaptive control results only consider single-hidden layer NNs, recent developments focus on deep learning with feedback control and establish that DNNs are exponentially more efficient than shallow NNs regarding the total number of neurons required to achieve the same accuracy in function approximation [2]. However, the developed adaptation methods are restricted to feedforward NNs, which are static structures and therefore only have access to current state information. Previous results establish that the presence of a memory capable of accessing previous state information both reduces the required data set for training and leads to faster learning. Unlike feedforward NNs, recurrent NNs (RNNs) are a dynamic mapping. Thus, RNNs have an internal memory that can leverage dependencies in a sequence and increase approximation capabilities, thus improving performance [3]. Therefore, motivation exists to develop adaptive deep RNN (DRNN) control architectures. However, the nonlinearities and internal dynamics of these models make deriving stability-driven adaptation results mathematically difficult. Despite these difficulties, this dissertation presents the first stability-driven online learning methods for long short-term memory (LSTM) and deep RNN (DRNN) architectures.

Chapter 1 provides a literature survey of common machine learning techniques, introduces the research section of the dissertation, and provides an outline of the remaining chapters. Section 1.1 provides a general background and motivation for machine learning-based control methods. Section 1.2 provides a background on supervised learning and analyses results in supervised learning-based control development. Section 1.3 covers NN-based control in more depth and surveys results that develop control schemes using shallow NNs. Section 1.4 analyses current results for deep learning in controls. Section 1.5 outlines the remaining chapters of the dissertation.

Chapter 2 develops an adaptive Lyapunov-based (Lb-) LSTM architecture and controller for general Euler-Lagrange systems. Specifically, a continuous-time Lb-LSTM NN is constructed and implemented in the controller as a feedforward term to adaptively estimate uncertain model dynamics. Despite the technical challenges posed by the complexity of the LSTM cell structure, stability-driven adaptation laws adjust the Lb-LSTM weights in real-time and allow the developed architecture to adapt to system uncertainties without any offline training requirements. A Lyapunov-based stability analysis is performed to guarantee uniform ultimate boundedness (UUB) of the tracking errors and LSTM state and weight estimation errors.

Chapter 3 provides a Lb-LSTM architecture for system identification of a class of nonlinear systems. RNNs contain temporal operations that allow them to retain information from previous states, making them well-suited for identification of dynamical systems. The developed LSTM observer adjusts in real-time to system uncertainties through a Lyapunov-derived weight adaptation law, where a filtered estimation error is designed and implemented in the weight adaptation law to relax full-state feedback requirements. A Lyapunov-based stability analysis is performed to ensure asymptotic convergence of the estimation errors and stability of the adaptive Lb-LSTM architecture.

Chapter 4 develops an output feedback (OFB) controller for uncertain nonlinear systems using DRNN. Inspired by the dynamic nature and memory capabilities of

RNNs, a DRNN observer is designed to adaptively estimate the unknown states of the system and is incorporated into a control framework. Unlike the preceding chapters, the developed OFB controller is designed to achieve a two-fold control objective: asymptotic estimation of the unmeasurable states and asymptotic tracking control. The developed Lyapunov-based adaptation laws adjust the weights of the Lb-DRNN using the system output in real-time. Through a Lyapunov-based stability analysis, the developed observer and the overall control design are proven to guarantee asymptotic stability, ensuring reliable and robust control performance.

Chapter 5 summarizes the preceding chapters in the dissertation. Additionally, Chapter 5 presents future research directions based on the work presented in the previous chapters.

CHAPTER 1
LITERATURE REVIEW AND DISSERTATION OUTLINE

1.1 Machine Learning

Machine learning methods leverage data to complete a task, making them popular for complex problems such as image classification and voice recognition [4, 5]. Machine learning algorithms can be split into three subcategories: reinforcement learning, unsupervised learning, and supervised learning [5] (Figure 1-1). In reinforcement

Figure 1-1. Outline of the machine learning subsections covered in this chapter. Subtopics of each section are represented by the branches in the diagram.

learning (RL), an agent learns to make decisions that maximize a predefined reward.

Thus, RL focuses on rewarding desirable behavior and punishing undesirable actions. Examples of RL algorithms include deep RL, Q-learning, and actor-critic methods. When implemented in closed-loop controllers, RL provides a model-free method for solving optimal control problems. Results such as [6–18] focus on the development of RL-based controllers for nonlinear systems.

Unsupervised learning uses machine learning algorithms to cluster unlabeled datasets. Unsupervised learning can be split into exclusive clustering and overlapping clustering methods. Unlabeled input data makes unsupervised learning useful for datasets in which common properties between data points are unclear. Popular uses of unsupervised learning methods include language recognition, spam email identification, and voice recognition [19]. Since the data points are not labeled, unsupervised learning algorithms cannot be used to solve regression problems, e.g., time series regression, which makes them difficult to implement in many closed-loop control applications. Unlike unsupervised learning, supervised learning can be used for both regression and classification problems. Supervised learning uses labeled datasets, i.e., each data point contains both features (i.e., inputs) and an associated label (i.e., output). A training dataset (composed of both input and output data) is used to adjust the model based on an error loss function. Thus, supervised learning algorithms can be implemented to solve regression problems for control design of systems with time-series input data.

1.2 Supervised Learning

Common algorithms for regression-based supervised learning include linear regression, polynomial regression, fuzzy logic, and neural networks (NNs) [5]. Since linear regression finds the linear relationship between the input and output variables, it is unable to accurately model complex functions. Control designs in [20, 21] use linear regression models to estimate uncertain system dynamics, but assume linearity-in-the-parameters. Therefore, these methods are not well-suited for more complex or more uncertain systems. Motivated by the need to accurately model more complex functions,

polynomial regression fits a polynomial curve to a dataset rather than a line. Results such as [22–24] implement polynomial regression algorithms to generate a feedforward estimate in various control schemes. However, polynomial regression methods are limited to modeling functions that can be expressed as a polynomial model, which is restrictive for many closed-loop control applications.

Results in [25–48] develop fuzzy-logic-based controllers (FLC). FLC was initially designed as a model free approach. While this approach had great success in practical applications, the theoretical contributions faced criticism due to a lack of a stability analysis and controller design. In response, some focus shifted towards model-based fuzzy logic control (FLC). FLC does not need an accurate dynamics model and is more robust to umodeled disturbances compared to many other adaptive control methods. Unlike linear and polynomial regression algorithms, fuzzy systems are capable of approximating any nonlinear system within an arbitrary small approximation error given a sufficient number of rules. This makes them useful for complex problems where accurate models may not be feasible.

Like fuzzy systems, NNs are capable of general function approximation and are not restricted to specific models. Specifically, NNs possess a universal function approximation property that allows them to be implemented in feedback control without requirements for linearity in the system parameters [49]. The more complex architecture and nested nonlinearities of NNs allow them to model more complex functions when compared to other regression models (e.g., linear regression). The universal function approximation property makes NNs well-suited for modeling uncertain system dynamics in control schemes.

1.3 Neural Networks

A single hidden layer feedforward NN with fully-connected layers is shown in Fig. 1-2 [50]. The single hidden-layer NN structure is characterized by having two layers of adjustable weights, the outer-layer and the hidden-layer. It has no internal feedback

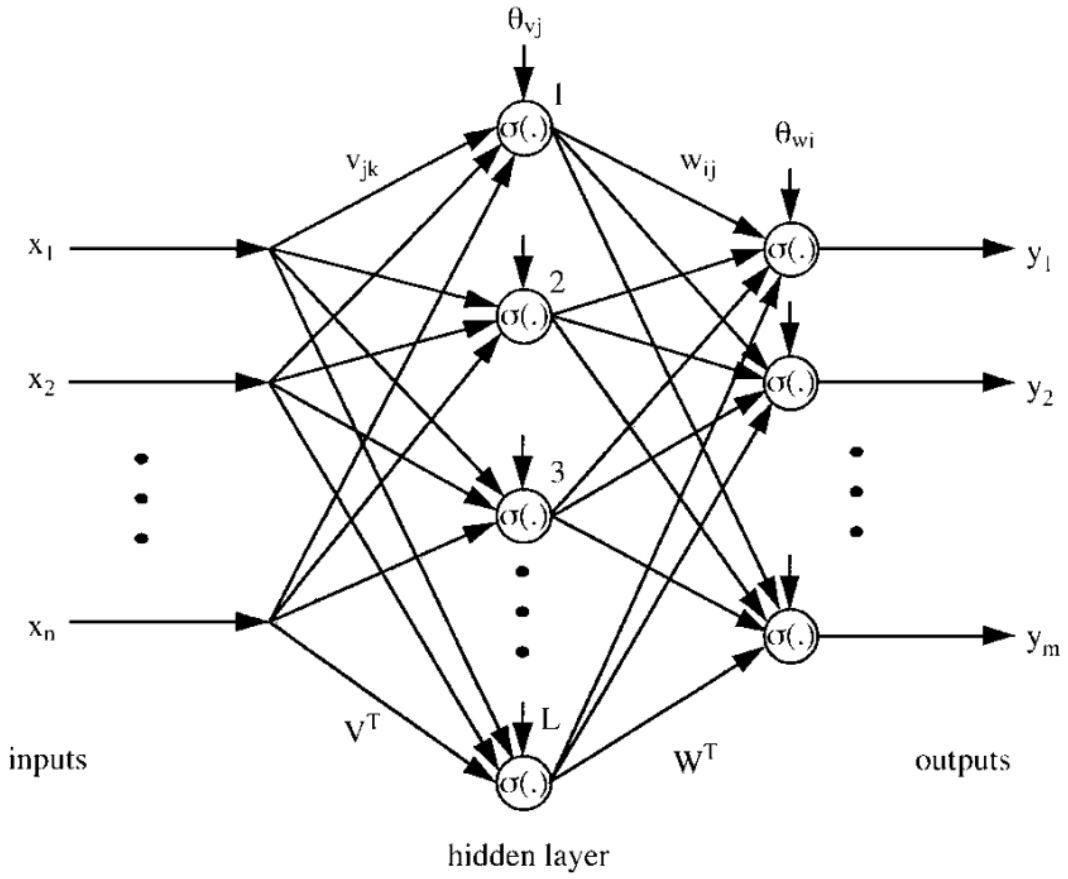

Figure 1-2. Architecture of a single hidden layer neural network [50].

connections, and is subsequently termed *feedforward*. It has no internal dynamics, and so is said to be *static.* A shallow NN is defined as [50]

$$y = W^\top \phi\left(V^\top x\right), \tag{1–1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the input, $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes the output, $\phi : \mathbb{R}^L \to \mathbb{R}^L$ denotes a vector of activation functions, and $V^\top \in \mathbb{R}^{L\times n}$ and $W^\top \in \mathbb{R}^{m\times L}$ denote weight matrices. The dimension $L \in \mathbb{R}_{>0}$ denotes the number of neurons in the hidden layer. Common activation functions include ReLu, the sigmoid function, hyperbolic tangent, and other logarithmic-curve-type functions.

Using the model defined in (1–1), results such as [33, 50–82] have investigated NN-based control schemes to compensate for nonlinearities in the system dynamics.

The use of NNs is motivated by the fact that NNs are universal approximators, i.e., NNs are able to approximate any continuous function on a compact set to some prescribed accuracy [49]. Thus, NNs can be used to estimate parametric uncertainties that do not satisfy the linear-in-the-parameters assumption required in most adaptive control methods. However, NNs are nonlinear in the inner-layer weights. Therfore, a challenge for NN-based closed-loop feedback control is providing online learning algorithms for the NN weight estimates that yield guaranteed stability. Moreover, since the universal function approximation property is restricted to continuous functions over a compact set, additional challenges arise for dynamical systems that do not readily meet these requirements.

Control methods such as [54–57, 68, 69, 73–80] train the NN weights using offline optimization techniques. These offline optimization methods adjust the NN weights by minimizing a loss function over a collected dataset. The resulting feedforward NN term is implemented in the controller as an open-loop function approximator. Such offline methods pose limitations since large, sufficiently rich datasets are typically required for accurate function approximation. Moreover, the resulting accuracy is only based on offline training and does not adjust to real-time data. Due to the lack of continual learning, offline trained models may not accurately model the dynamics during task execution, whereas online methods consider real-time data, involve a closed-loop implementation, and provide stability guarantees under appropriate weight adaptation laws. Rather than base function approximation performance solely on offline learning, results such as [33, 50–53, 58–67, 70–72, 81, 82] develop adaptive NN-based control schemes for real-time learning of the NN weights. Specifically, the developed control schemes include an adaptive closed-loop feedforward term to compensate for unknown, nonlinear terms in the model dynamics and robustifying terms account for approximation error. Online tuning of the weights eliminates the need for an offline learning phase and ensures good performance even with random weight initialization.

Many adaptive control methods use optimization techniques to adjust the NN weights based on a loss function [55–57, 59, 60, 81]. Although such online training methods have promising empirical results, they lack stability guarantees. In contrast to optimization training techniques, results such as [33, 50–54, 58, 61–80, 82] focus on the design of adaptive NN architectures based on Lyapunov stability theory. Lyapunov-derived weight adaptation laws allow for the NN feedforward estimate to adjust based on an analytical update law in real-time. Although NN-based adaptive control design has been well-established, many of the developed adaptive NN architectures only apply for single hidden layer NNs (i.e., the architecture defined in (1–1)), which is restrictive. Moreover, while various adaptive NN architectures have proven stability guarantees, convergence of the NN weight estimates remains an open problem.

Unlike feedforward NNs which are memoryless and static, RNNs are a type of dynamic NN that can capture the temporal dynamic behavior of an unknown system. A single hidden layer RNN can be defined as [83]

$$\dot{h} = -bh + W^{\top}\phi\left(V^{\top}h\left(t\right) + U^{\top}x\left(t\right)\right), \tag{1–2}$$

where $h : \mathbb{R}_{>0} \to \mathbb{R}^m$ denotes the hidden state, $x : \mathbb{R}_{>0} \to \mathbb{R}^n$ denotes the input, $\phi : \mathbb{R}^L \to \mathbb{R}^L$ denotes a vector of activation functions, and $U \in \mathbb{R}^{L\times n}$, $V \in \mathbb{R}^{L\times m}$, and $W \in \mathbb{R}^{m\times L}$ denote weight matrices. From the hidden state h, RNNs contain connections between the nodes that retain state information for later use through temporal operations present in the RNN architecture (Fig. 1-3) [83]. As a result, RNNs are a dynamic mapping and are better suited for dynamical system identification than feedforward NNs [3].

Results such as [1, 3, 84–95] employ RNNs for the identification of nonlinear systems and propose several training methods for adaptation of the RNN weights. Specifically, these results formulate RNNs as observers to estimate hidden states and identify the dynamics of the system. Robustifying terms account for residual error

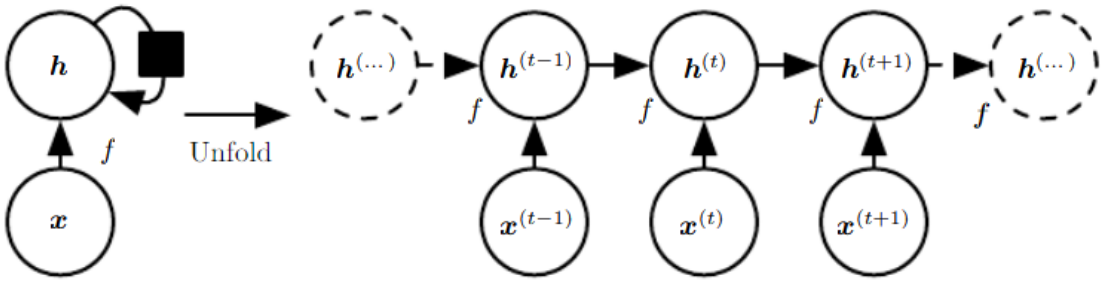

Figure 1-3. An RNN with no output. This RNN processes information from the input x by incorporating it into a hidden state h that is passed forward in time. The left and right diagrams indicate an unfolded and folded representation respectively, where the black box indicates a single time step delay [83].

due to the RNN function approximation mismatch. Compared to traditional model-based observers, RNN-based observers relax model knowledge requirements and can be implemented for estimation of an unknown system. Previous methods employ optimization techniques to train the weights based on some loss function [3, 89–92, 95]. Although these training methods have been successfully used in empirical studies, they lack analytical results concerning stability guarantees. In contrast to offline optimization training techniques, results such as [1, 84–88, 93, 94] focus on the design and analysis of Lyapunov-derived adaptive RNN architectures. Specifically, these results develop adaptive RNN architectures to estimate unknown, nonlinear dynamics. The developed RNN observers adjust in real-time through weight adaptation laws designed using Lyapunov-based stability analyses. Due to the complex nature and nested nonlinearity of deep architectures, these methods are restricted to either RNNs satisfying a linear-in-the-parameters assumption or RNNs with a single hidden layer.

1.4 Deep Learning

Although NNs with a single hidden layer are capable of approximating general nonlinear functions, a growing amount of empirical and experimental results indicate that DNNs provide improved performance [4]. Moreover, DNNs are exponentially more efficient than shallow NNs regarding the total number of neurons required to achieve

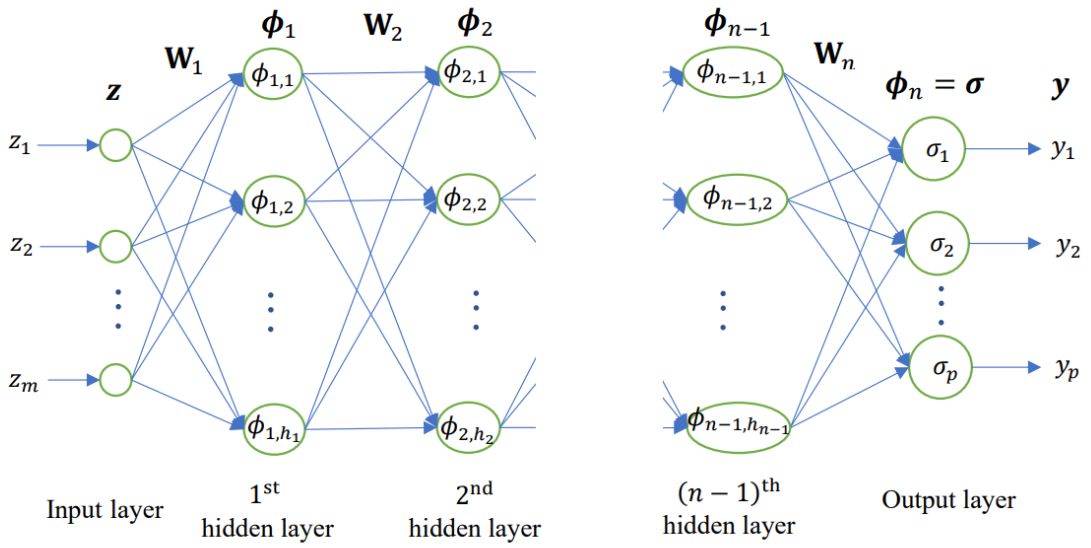

Figure 1-4. Architecture of a feedforward deep neural network [96].

the same accuracy in function approximation [2]. Thus, motivation exists to investigate DNN-based control methods. A feedforward DNN can be modeled as (Figure (1-4)) [96]

$$\Phi\left(x, W_0, W_1, W_2, ..., W_k\right) = W_k^\top \phi_k \circ ... \circ W_1^\top \phi_1 \circ W_0^\top x_a, \tag{1–3}$$

where $x_a \triangleq \left[x^\top \; 1\right]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n+1}$ denotes the augmented input, and $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the input. The smooth activation functions and weight matrices at the j^{th} layer are denoted by $\phi_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ and $W_j^\top \in \mathbb{R}^{L_{j+1} \times L_j}$ for all $j \in \{1, ..., k\}$ and $j \in \{0, ..., k\}$, respectively, where $L_j \in \mathbb{R}_{>0}, \forall j \in \{0, ..., k\}$ denotes the number of neurons in the j^{th} layer. To incorporate a bias term, x_a and ϕ_j are augmented with 1 for all $j \in \{1, ..., k\}$.

Results such as [97] and [98] develop DNN-based closed-loop controllers for uncertain, nonlinear systems. Specifically, a DNN estimate is implemented as a feedforward term to compensate for model uncertainty. Training of the DNN weights is generally accomplished offline using optimization techniques. Hence, these results have the same limitations as the offline training methods developed for single-hidden layer NNs

in Section 1.3. Therefore, results such as [97] and [98] are likely not robust to factors such as unmodeled disturbances, time-varying behavior, or sudden changes in system dynamics. Typically, control architectures that use offline trained DNN estimates are based on empirical studies and lack analytical results that yield stability guarantees.

Motivated to address these limitations, results such as [99–105] focus on the development of DNN-based control schemes with stability-derived, real-time weight adaptation. The inner-layer weights of DNNs are embedded inside nonlinear activation functions. The inner-layer embedding presents a major challenge toward the derivation of stability results for DNN-based controllers and the development of adaptive weight update laws based on stability analyses. Prior to the results in [99] and [100], many DNN-based controllers showed improved performance empirically, but lacked performance guarantees due to the probability nature of DNNs. In [99] and [100], Lyapunov-based stability analyses ensure stability of the developed control schemes, making them suitable for safety-critical applications. However, those results are specific to linear systems with known A and B matrices and matched system uncertainty. While some nonlinear systems can be linearized about an equilibrium point, this approach isn't applicable for many nonlinear systems. Thus, results in [101] extend the adaptive weight update law and DNN architecture developed in [99] and [100] to general uncertain, nonlinear systems. The control development in [99–101] includes training techniques for updates of the DNN concurrent to real-time. DNN training algorithms can be used to update the inner-layers of the DNN concurrent to real-time as an alternative to adaptive inner-layer weight update laws. Switching due to batch updates of the inner-layer weights of the DNN necessitates a nonsmooth stability analysis. However, such designs result in real-time adaptation of only the outer-layer weights.

In [102], an adaptive DNN-based controller is developed that establishes real-time modular weight updates for multiple layers of a DNN of arbitrary depth. The modular weight adaptation law in [102] allows for various update laws for the inner-layer weights

to be selected and designed for learning of the inner-layer weights while guaranteeing tracking performance. However, while this modular approach allows for significant flexibility in the design of the weight adaptation law, it does not provide any guidance regarding which designs would be most efficient or most optimal.

The work in [104] presents the first DNN-based control scheme with real-time Lyapunov-derived weight adaptation laws for all layers of the DNN feedforward estimate. A significant contribution is that the approach overcomes difficulties presented by the nested nonlinearities in the DNN architecture and considers a DNN of an arbitrary depth. This work is later extended to adaptive residual NN (ResNet) architectures in [103] to add robustness to vanishing perturbations to the online weight adaptations. Lyapunov-based stability analyses in [103] and [104] show asymptotic convergence of the tracking errors and stability of the overall closed-loop error systems and adaptive weight update laws, but as seen in shallow NN-based adaptive control schemes, guaranteed convergence of the DNN weight estimates remains an open problem.

Feedforward NNs have a static structure that only has access to current state information. Previous findings have demonstrated that incorporating a memory component capable of accessing previous state information reduces the required training data and accelerates the learning process [106–108]. Building on these insights, results in [109] augment static NN-based controllers with an external memory, resulting in faster learning and improved function approximation. Although this approach introduces a working memory to the NN, the NN remains static and feedforward, with the augmented memory being external to the NN architecture. RNNs are a dynamic type of NN that are specifically designed to handle sequential or temporal data. Unlike feedforward NNs which process inputs independently and have no internal memory, RNNs have a recurrent connection that allows them to maintain information from previous time steps. This internal memory allows RNNs to capture and model time-varying and accumulative effects present in certain dynamic systems that feedforward NNs cannot. This ability to capture

dynamic behavior makes RNNs better suited to construct state observers for output feedback control of uncertain nonlinear systems compared to traditional feedforward NNs.

Recent work in [110] and [111] focuses on the implementation of dynamic DNNs in control schemes. Like RNNs with a single-hidden layer, dynamic DNNs can capture time dependencies from nonlinear time-series data more effectively than a static NN. This property, combined with the improved function approximation performance of deep architectures motivates the development of dynamic DNN-based control architectures. Results in [110] and [111] implement dynamic DNNs for system identification during closed-loop control. The control design in [111] implements long short-term memory (LSTM) units in the developed dynamic DNN architecture. The architecture in [110] is composed of both LSTM units and gated recurrent units (GRU). In contrast to traditional RNN units implemented in [1, 3, 84–95], LSTMs and GRUs regulate the flow of information through operations that let them 'forget' data from previous time steps that is deemed less important (Figure 1-5) [112]. These operations make LSTMs and GRUs

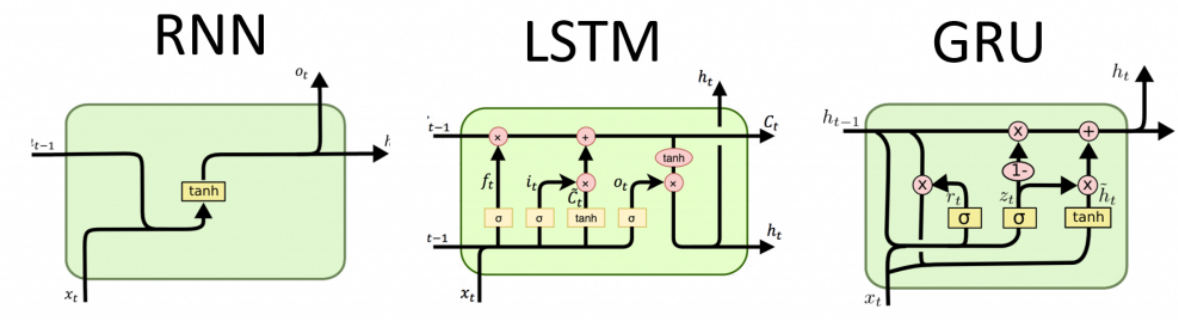

Figure 1-5. Comparison of the RNN, LSTM, and GRU structures [112].

more computationally effective. However, this added complexity also makes real-time adaptation of these architectures significantly more challenging.

In [110] and [111], the DNN weights are trained offline using machine learning optimization algorithms. Thus, while the authors in [110] and [111] provide promising

empirical results, they do not provide analytical results such as stability guarantees. However, the empirical results in [110] and [111] corroborate previous analytical results that establish the function approximation performance of DNNs as well as the ability of recurrent units in NNs to capture dynamical behavior. While real-time weight adaptation has not been investigated for dynamic DNN control architectures, advantages of stability-derived online learning for feedforward NNs has been established. These factors motivate investigation into adaptive dynamic DNN-based control schemes.

1.5 Dissertation Outline

Chapter 2 develops an adaptive Lb-LSTM architecture and controller for general Euler-Lagrange systems, where the adaptation law is derived from Lyapunov-based methods (hence, we refer to the architecture as Lb-LSTM). Specifically, the first continuous-time representation for the LSTM cell is developed. The continuous-time LSTM estimate is then implemented in the controller as a feedforward term to adaptively estimate uncertain model dynamics. Despite the technical challenges posed by the complexity of the LSTM cell structure, stability-driven adaptation laws adjust the Lb-LSTM weights in real-time and allow the developed architecture to adapt to system uncertainties without any offline training requirements. Thus, this is the first stability-driven adaptation result for deep RNN control architectures. A Lyapunov-based stability analysis is performed to guarantee uniform ultimate boundedness (UUB) of the tracking errors and LSTM state and weight estimation errors. To demonstrate the performance of the adaptive Lb-LSTM controller, simulations were performed and compared to the adaptive DNN-based controller in [104] using three different DNN architectures. The simulation results indicate significant improvements in tracking and function approximation performance when compared to various feedforward DNN architectures. Specifically, the developed Lb-LSTM controller yielded twofold and fourfold faster tracking error and function approximation error convergence, respectively, when compared to a baseline DNN architecture of a similar size.

In Chapter 3, an adaptive Lb-LSTM architecture is designed using the continuous-time Lb-LSTM architecture developed in Chapter 2 and is implemented in an observer to estimate unmeasurable states in a class of nonlinear systems. The developed observer leverages the dynamic structure of LSTMs to produce an adaptive estimate of the unknown system states. Since the unknown observer error is not available for online learning, a dynamic filter is designed to construct an auxiliary error that is implementable in the weight adaptation law. Despite the challenges posed by the complex structure of the LSTM cell and the fact that the system error is unknown, a Lyapunov stability-driven adaptation law is developed for all weights of the LSTM. Thus, the developed Lb-LSTM observer is able to learn the system dynamics in real-time and adapt to model uncertainties without any offline training requirements. A nonsmooth Lyapunov-based stability analysis is performed that guarantees asymptotic convergence of the estimation errors and stability of the Lb-LSTM architecture. To validate the developed observer design, simulations were performed to estimate the angular velocity states of a two-link robot manipulator. The Lb-LSTM observer yielded a 41.13% improvement in the estimation error when compared to the adaptive shallow RNN observer in [1].

Chapter 4 develops an output feedback (OFB) controller for uncertain nonlinear systems using adaptive deep RNNs (DRNNs). Inspired by the dynamic nature and memory capabilities of RNNs, a DRNN observer is designed to adaptively estimate the unknown states of the system and is incorporated into a control framework. Unlike the preceding chapters, the developed OFB controller is designed to achieve a two-fold control objective: asymptotic estimation of the unmeasurable states and asymptotic tracking control and is therefore the first OFB result using adaptive DRNNs. The weights of the DRNN adjust online using Lyapunov-based stability-driven adaptation laws based on the tracking and observer errors. The developed adaptation laws allow the Lb-DRNN to adapt, learn, and control the system based on the system output in real-time.

Through a Lyapunov-based stability analysis, the developed observer and the overall control design are proven to guarantee asymptotic stability, ensuring reliable and robust control performance. Validation simulation experiments on an unmanned underwater vehicle system yielded a 27.98% and 89.94% improvement in normalized linear and angular tracking error, respectively.

Chapter 5 summarizes the preceding chapters in the dissertation. Additionally, Chapter 5 presents future research directions based on the work presented in the previous chapters.

1.6 Notation and Preliminaries

To facilitate the subsequent presentation, common notation used in the remaining chapters is introduced in this section. The space of essentially bounded Lebesgue measurable functions is denoted by $\mathcal{L}_\infty$. The right-to-left matrix product operator is represented by $\overset{\curvearrowleft}{\prod}$, i.e., $\overset{\curvearrowleft m}{\underset{p=a}{\Pi}} A_p = A_m \dots A_{a+1} A_a$ and $\overset{\curvearrowleft m}{\underset{p=a}{\Pi}} A_p = I$ if $a > m$, where I denotes the identity matrix. The Kronecker product is denoted by $\otimes$. Function compositions are denoted using the symbol $\circ$, e.g., given suitable functions g and h, $(g \circ h)(x) = g(h(x))$. The Hadamard (element-wise) product is denoted by $\odot$ and satisfies the following properties [113, Definition 9.3.1]. Given any $b, c \in \mathbb{R}^n$, $b \odot c = D_b c$ and therefore, $\frac{\partial}{\partial c}(b \odot c) = D_b$, where $D_b \in \mathbb{R}^{n \times n}$ denotes a diagonal matrix with the vector b as its main diagonal. The sum of a point $a \in \mathbb{R}$ and a set $B \subset \mathbb{R}$ is defined as $a + B \triangleq \{a + b : b \in \mathbb{R}\}$. The Filippov set-valued map defined in [114, Equation 2b] is denoted by $K[\cdot]$. Consider a Lebesgue measurable and locally essentially bounded function $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. Then, the function $y : \mathcal{I} \to \mathbb{R}^n$ is called a *Filippov solution* of $\dot{y} = h(y, t)$ on the interval $\mathcal{I} \subseteq \mathbb{R}_{\geq 0}$ if y is absolutely continuous on $\mathcal{I}$ and $\dot{y} \overset{a.a.t}{\in} K[h](y, t)$, where $a.a.t.$ denotes almost all time. The vectorization operator is denoted by $\mathrm{vec}(\cdot)$, i.e., given $A \triangleq [a_{i,j}] \in \mathbb{R}^{n \times m}$, $\mathrm{vec}(A) \triangleq [a_{1,1}, \dots, a_{n,1}, \dots, a_{1,m}, \dots, a_{n,m}]^\top$. The vectorization operator satisfies the following properties [113, Proposition 7.1.9]. Given

any $A \in \mathbb{R}^{p \times a}$, $B \in \mathbb{R}^{a \times r}$, and $C \in \mathbb{R}^{r \times s}$,

$$\operatorname{vec}(ABC) = \left(C^{\top} \otimes A\right) \operatorname{vec}(B), \tag{1–4}$$

and consequently,

$$\frac{\partial}{\partial \operatorname{vec}(B)} \operatorname{vec}(ABC) = C^{\top} \otimes A. \tag{1–5}$$

CHAPTER 2
LYAPUNOV-BASED LONG SHORT-TERM MEMORY (LB-LSTM) NEURAL NETWORK-BASED CONTROL

2.1 Introduction

RNNs are a dynamic mapping that can capture time-varying, accumulative effects in a sequence that static, feedforward NNs cannot. LSTMs are a type of RNNs that have gained recent popularity because the cell structure allows them to retain long-term information more efficiently than traditional RNNs. Existing results develop LSTM-based controllers to compensate for uncertainties in nonlinear systems. However, these results use discrete-time LSTMs with offline-trained weights. In this chapter, a Lb-LSTM controller is developed for general Euler-Lagrange systems. To make the architecture better suited for control of continuous-time systems, a continuous-time representation of the LSTM cell is developed for the first time. The Lb-LSTM is implemented in the control design to adaptively estimate uncertain model dynamics, where the weight estimates of the LSTM cell are updated using Lyapunov-based adaptation laws. The Lyapunov-based adaptation laws are the first stability-driven online learning result for LSTMs and allow the LSTM cell to adapt to system uncertainties without requiring offline training. A Lyapunov-based stability analysis yields UUB of the tracking errors and LSTM state and weight estimation errors. Simulations indicate the developed Lb-LSTM-based controller yielded twofold and fourfold faster tracking error and function approximation error convergence, respectively, when compared to a baseline DNN architecture of a similar size.

2.2 System Dynamics and Control Objective

Consider a general uncertain Euler-Lagrange system modeled as

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

where $q,\dot{q},\ddot{q}\in\mathbb{R}^{n}$ denote the generalized position, velocity, and acceleration, respectively, and $M:\mathbb{R}^{n}\rightarrow\mathbb{R}^{n\times n}$, $V_{m}:\mathbb{R}^{n}\times\mathbb{R}^{n}\rightarrow\mathbb{R}^{n\times n}$, $G:\mathbb{R}^{n}\rightarrow\mathbb{R}^{n}$, $F:\mathbb{R}^{n}\rightarrow\mathbb{R}^{n\times n}$,

and $\tau : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denote unknown, continuous generalized inertial effects, generalized centripetal-Coriolis effects, generalized vector of potential forces, generalized dissipation effects, and the control input, respectively. The model dynamics in (2–1) are assumed to satisfy the following assumption.

Assumption 2.1. The inertial and centripetal-Coriolis effects satisfy

$$\xi^{\top}\left(\dot{M}\left(q\right)-2V_m\left(q,\dot{q}\right)\right)\xi=0$$

for all $q,\dot{q},\xi\in\mathbb{R}^n$ [115].

The control objective is to design an adaptive Lb-LSTM controller to achieve UUB tracking of a desired trajectory $q_d\in\mathbb{R}^n$. To quantify the control objective, a tracking error $e\in\mathbb{R}^n$ and an auxiliary tracking error $r\in\mathbb{R}^n$ are defined as

$$e\triangleq q_d-q, \tag{2–2}$$

$$r\triangleq\dot{e}+\alpha e, \tag{2–3}$$

respectively, where $\alpha\in\mathbb{R}_{>0}$ denotes a user-selected constant. The desired trajectory q_d is designed to be sufficiently smooth, i.e., $q_d,\dot{q}_d,\ddot{q}_d$ can be bounded as $\|q_d\|\leq\overline{q}_d,\|\dot{q}_d\|\leq\overline{\dot{q}}_d,\|\ddot{q}_d\|\leq\overline{\ddot{q}}_d$, where $\overline{q}_d,\overline{\dot{q}}_d,\overline{\ddot{q}}_d\in\mathbb{R}_{>0}$ denote known constants and $\dot{q}_d\in\mathbb{R}^n$ and $\ddot{q}_d\in\mathbb{R}^n$ denote the first and second time-derivatives of q_d, respectively.

Taking the time-derivative of (2–3), multiplying by $M\left(q\right)$, and using (2–1) and (2–2) yields

$$M\left(q\right)\dot{r}=g\left(x\right)-\tau-V_m\left(q,\dot{q}\right)r, \tag{2–4}$$

where $x\triangleq[q^{\top},\dot{q}^{\top},q_d^{\top},\dot{q}_d^{\top},\ddot{q}_d^{\top}]^{\top}\in\mathbb{R}^{5n}$ denotes a concatenated vector and the function $g\left(x\right)\in\mathbb{R}^n$ denotes the unknown system dynamics defined as $g\left(x\right)\triangleq M\left(q\right)\left(\ddot{q}_d+\alpha\dot{e}\right)+V_m\left(q,\dot{q}\right)\left(\dot{q}_d+\alpha e\right)+F\left(\dot{q}\right)+G\left(q\right)$.

2.3 Control Development

Long short-term memory (LSTM) NNs have grown in recent popularity due to their ability to leverage both long-term and short-term dependencies in time sequences for faster learning and improved performance. LSTMs are a type of RNN that are designed to overcome the limitations of traditional RNNs in capturing long-term sequence dependencies in time series data due to vanishing gradient. This is done by introducing a more complex memory cell structure with gating mechanisms that regulate the flow of information. The key components of the LSTM cell are the cell state, hidden state, and the gate units (i.e., the forget, cell, input, and output gates). The cell state allows information to flow relatively unchanged through the network, which helps preserve long-term dependencies. The forget gate controls what information to discard from the cell state, and the input and cell gates determine what new information is stored in the cell state. The output gate determines what information in included in the hidden state, which serves as the output of the LSTM cell. These features motivates the development of an LSTM-based controller that can estimate and compensate for the unknown model dynamics in (2–4).

The LSTM cell architecture developed in [116] is in discrete-time and is converted to a continuous-time model in (2–5) to make it more appropriate for controlling a continuous-time system. The gains b_c and b_h in (2–5) are a result of constructing a continuous-time model and can be tuned accordingly to enhance the performance of the continuous-time LSTM. Therefore, based on the design of continuous-time RNNs [85] and using Euler's method, an LSTM NN (Fig. 2-1) can be modeled in continuous-time as [116]

$$f\left(z, W_f\right) = \sigma_g \circ W_f^\top z,$$
$$i\left(z, W_i\right) = \sigma_g \circ W_i^\top z,$$
$$o\left(z, W_o\right) = \sigma_g \circ W_o^\top z,$$

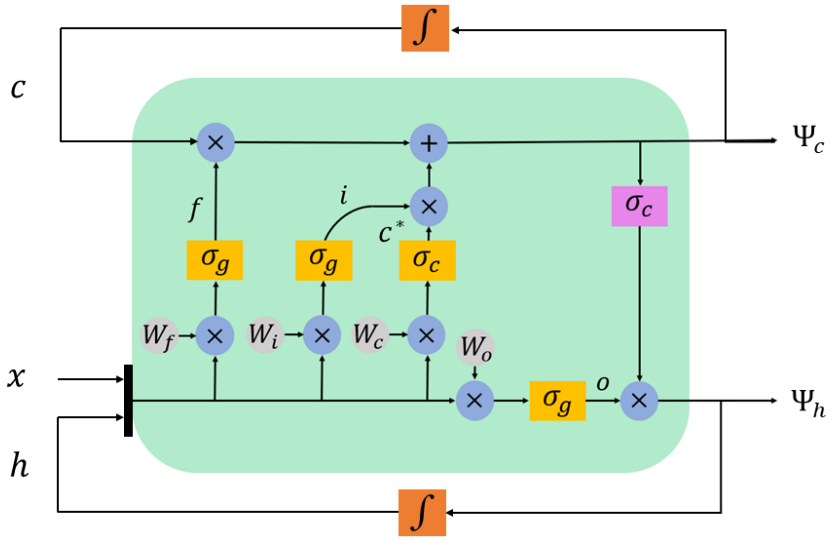

Figure 2-1. LSTM model in (2–5), where the green box represents the LSTM cell.

$$
\begin{aligned}
c^{*}\left(z, W_{c}\right) &= \sigma_{c} \circ W_{c}^{\top} z, \\
\dot{c} &= -b_{c} c + b_{c} \Psi_{c}\left(x, c, h, \theta\right), \\
\dot{h} &= -b_{h} h + b_{h} \Psi_{h}\left(x, c, h, \theta, W_{o}\right),
\end{aligned}
\tag{2–5}
$$

where $b_c, b_h \in \mathbb{R}_{>0}$ denote user-selected constants, and $c \in \mathbb{R}^{l_2}$ and $h \in \mathbb{R}^{l_2}$ denote the cell state and hidden state, respectively, where $h\left(0\right) = c\left(0\right) = 0$ and $l_2 \in \mathbb{R}_{>0}$ denotes the number of neurons. The concatenated state vector $z \in \mathbb{R}^{l_1}$ is defined as $z \triangleq \left[x^{\top}, h^{\top}, 1\right]^{\top}$, where $x \in \mathbb{R}^{5n}$ denotes the LSTM input and $l_1 \triangleq 5n + l_2 + 1$. The state z is augmented with a 1 to incorporate a bias term. The forget gate, input gate, cell gate, and output gate are denoted by $f\left(z, W_f\right) \in \mathbb{R}^{l_2}$, $i\left(z, W_i\right) \in \mathbb{R}^{l_2}$, $c^{*}\left(z, W_c\right) \in \mathbb{R}^{l_2}$, and $o\left(z, W_o\right) \in \mathbb{R}^{l_2}$, respectively. Like the hidden state h, the cell state c is passed from one time step to the next. The gate output c^{*} (typically referred to as $\tilde{c}$ in literature) is not passed to the next time step and represents the output of one of the internal gates within the LSTM cell. The sigmoid and tanh activation functions are denoted

by $\sigma_g : \mathbb{R}^{l_2} \to \mathbb{R}^{l_2}$ and $\sigma_c : \mathbb{R}^{l_2} \to \mathbb{R}^{l_2}$, respectively, and the weight matrices are denoted by $W_f^\top, W_c^\top, W_i^\top, W_o^\top \in \mathbb{R}^{l_2 \times l_1}$, where $\theta \triangleq [W_c^\top, W_i^\top, W_f^\top]^\top \in \mathbb{R}^{l_2 \times 3l_1}$. The functions $\Psi_c(x, c, h, \theta) \in \mathbb{R}^{l_2}$ and $\Psi_h(x, c, h, \theta, W_o) \in \mathbb{R}^{l_2}$ are defined as $\Psi_c(x, c, h, \theta) \triangleq f(z, W_f) \odot c + i(z, W_i) \odot c^*(z, W_c)$ and $\Psi_h(x, c, h, \theta, W_o) \triangleq o(z, W_o) \odot (\sigma_c \circ \Psi_c(x, c, h, \theta))$, respectively.

To ensure the output of the LSTM has the appropriate dimensions, a fully-connected layer is added to the LSTM cell. To add generality to the LSTM model, a feedforward component is added to the output of the LSTM. The resulting LSTM model allows for a direct transmission of the input information through the feedforward component while leveraging the internal memory capabilities of LSTMs. Thus, the output of the LSTM $\Phi(x, c, h, \theta, W_o, W_h, W_{FF}) \in \mathbb{R}^n$ can be modeled as

$$\Phi = W_h^\top \left(\Psi_h(x, c, h, \theta, W_o) + \sigma \circ W_{FF}^\top x \right), \tag{2–6}$$

where $\sigma : \mathbb{R}^{l_2} \to \mathbb{R}^{l_2}$ denotes a vector of smooth activation functions, and $W_h^\top \in \mathbb{R}^{n \times l_2}$ and $W_{FF}^\top \in \mathbb{R}^{l_2 \times 5n}$ denote the output weight matrix and weight matrix of the feedforward NN component, respectively.

The universal function approximation property states that the function space of (2–5) is dense in $\mathcal{C}(\mathcal{Z})$, where $\mathcal{C}(\mathcal{Z})$ denotes the space of continuous functions over the set $\mathcal{Z} \subseteq \mathbb{R}^{l_1}$, where $z \in \mathcal{Z}$ [117, Theorem 1.1][1] . Therefore, for any prescribed $\bar{\varepsilon} \in \mathbb{R}_{>0}$, there exist ideal weight matrices $W_c^\top$, $W_i^\top$, $W_f^\top$ $W_o^\top$, $W_h^\top$, and $W_{FF}^\top$ such that the system dynamics $g(x)$ can be modeled using the LSTM architecture in (2–5) as

$$g(x) = \Phi(x, c, h, \theta, W_o, W_h, W_{FF}) + \varepsilon(x). \tag{2–7}$$

[1] Since the subspace of LSTMs in (2–6) involving the feedforward term $W_h^\top \sigma \circ W_{FF}^\top x$ is dense in $\mathcal{C}(\mathcal{Z})$, the space of LSTMs is also dense.

It is assumed that there exists a known constant $\overline{W} \in \mathbb{R}_{>0}$ such that the ideal weights can be bounded as $\|W_j\|_F \leq \overline{W}$ for all $j \in \{c, i, f, o, h, FF\}$ [104].

To compensate for the unknown LSTM model dynamics in (2–5), auxiliary cell and hidden state estimation errors are introduced in this section. The auxiliary cell and hidden state estimation errors $\tilde{c} \in \mathbb{R}^{l_2}$ and $\tilde{h} \in \mathbb{R}^{l_2}$ are defined as

$$\tilde{c} \triangleq c - \hat{c} + \eta_c, \tag{2–8}$$

$$\tilde{h} \triangleq h - \hat{h} + \eta_h, \tag{2–9}$$

respectively, where $\hat{c} \in \mathbb{R}^{l_2}$ and $\hat{h} \in \mathbb{R}^{l_2}$ denote the estimated cell state and hidden state, respectively, and $\eta_c \in \mathbb{R}^{l_2}$ and $\eta_h \in \mathbb{R}^{l_2}$ are designed as

$$\dot{\eta}_c \triangleq -k_{1,c}\eta_c - K_{2,c}r, \tag{2–10}$$

$$\dot{\eta}_h \triangleq -k_{1,h}\eta_h - K_{2,h}r, \tag{2–11}$$

where $k_{1,c}, k_{1,h} \in \mathbb{R}_{>0}$ denote user-selected constants, and $K_{2,c}, K_{2,h} \in \mathbb{R}^{l_2 \times n}$ denote user-selected matrices.

The following lemma establishes boundedness properties of the cell state and hidden state of the LSTM model in (2–5), which is essential for the ensuing development.

Lemma 2.1. *Consider the LSTM model in* (2–5)*. The hidden state h and cell state c can be bounded as*

$$\|h\| \leq \frac{b_h\sqrt{l_2}}{\sqrt{2\left(b_h - \frac{1}{2}\right)}}, \quad \|c\| \leq \frac{\sqrt{l_2}}{\sqrt{2\left(b_c - b_c\sqrt{l_2} - \frac{1}{2}\right)}}.$$

Proof. Consider the hidden state dynamics in (2–5), where the input Ψ_h can be bounded as $\|\Psi_h\| \leq \sqrt{l_2}$ by design of the sigmoid and tanh activation functions. Consider the candidate Lyapunov function $\mathcal{V}_h : \mathbb{R}^{l_2} \to \mathbb{R}_{\geq 0}$ defined as $\mathcal{V}_h(h) \triangleq \frac{1}{2}h^\top h$. Taking the derivative, using (2–5), bounding, and applying the Gronwall inequality yields $\mathcal{V}_h \leq \mathcal{V}_h(h(t_0))\exp\left(-2\left(b_h - \frac{1}{2}\right)(t - t_0)\right) + \frac{b_h^2 l_2}{4\left(b_h - \frac{1}{2}\right)}$. Therefore, provided $b_h \geq \frac{1}{2}$, initializing h as $h(t_0) = 0$ yields $\|h\| \leq \frac{b_h\sqrt{l_2}}{\sqrt{2\left(b_h - \frac{1}{2}\right)}}$. Similarly, to prove boundedness of

the cell state c, consider the candidate Lyapunov function $\mathcal{V}_c : \mathbb{R}^{l_2} \to \mathbb{R}_{\geq 0}$ defined as $\mathcal{V}_c = \frac{1}{2} c^\top c$. By design of the sigmoid and tanh activation functions, $\|f\| \leq \sqrt{l_2}$ and $\|i \odot c^*\| \leq \sqrt{l_2}$. Taking the derivative of the candidate Lyapunov function, substituting the cell state dynamics in (2–5), bounding, and applying the Gronwall inequality yields $\mathcal{V}_c(c(t)) \leq \mathcal{V}_c\left(c\left(t_0\right)\right) \exp\left(-2\left(b_c - b_c\sqrt{l_2} - \frac{1}{2}\right)\left(t - t_0\right)\right) + \frac{l_2}{4\left(b_c - b_c\sqrt{l_2} - \frac{1}{2}\right)}$. Therefore, provided $b_c \geq \frac{1}{2\left(1+\sqrt{l_2}\right)}$, initializing c as $c\left(t_0\right) = 0$ yields $\|c\| \leq \frac{\sqrt{l_2}}{\sqrt{2\left(b_c - b_c\sqrt{l_2} - \frac{1}{2}\right)}}$. □

2.3.1 Control Design

Let the adaptive estimates of the LSTM weights be denoted as $\hat{\theta} \triangleq [\widehat{W}_c^\top, \widehat{W}_i^\top, \widehat{W}_f^\top]^\top \in \mathbb{R}^{3l_1 \times l_2}$, $\widehat{W}_o^\top \in \mathbb{R}^{l_2 \times l_1}$, $\widehat{W}_h^\top \in \mathbb{R}^{n \times l_2}$, and $\widehat{W}_{FF}^\top \in \mathbb{R}^{l_2 \times 5n}$. Based on the adaptive weight estimates, an Lb-LSTM adaptive feedforward term $\widehat{\Phi} \triangleq \Phi\left(x, \hat{c}, \hat{h}, \hat{\theta}, \widehat{W}_o, \widehat{W}_h, \widehat{W}_{FF}\right)$ is constructed and the control input is designed as

$$\tau \triangleq \widehat{\Phi} + k_r r - K_{2,c}\eta_c - K_{2,h}\eta_h + e, \tag{2–12}$$

where $k_r, k_s \in \mathbb{R}_{>0}$ denote user-selected constants. Substituting the LSTM model in (2–7) and the control input in (2–12) into (2–4) yields the closed-loop error system

$$M\left(q\right)\dot{r} = \widetilde{\Phi} + j_e + \varepsilon\left(x\right) - V_m\left(q, \dot{q}\right)\left(r\right) - k_r r + K_{2,c}\eta_c + K_{2,h}\eta_h - e, \tag{2–13}$$

where the function $j_e\left(x, c, h, \theta, W_o, W_h\right) \in \mathbb{R}^n$ is defined as $j_e \triangleq \Phi\left(x, c, h, \theta, W_o, W_h, W_{FF}\right) - \Phi\left(x, \hat{c}, \hat{h}, \theta, W_o, W_h, W_{FF}\right)$.

2.3.2 Weight Adaptation Laws

Using the LSTM model in (2–5), the estimated cell state $\hat{c}$ and estimated hidden state $\hat{h}$ evolve according to

$$\dot{\hat{c}} = -b_c\hat{c} + b_c\left(f\left(\hat{z}, \widehat{W}_f\right) \odot c + i\left(\hat{z}, \widehat{W}_i\right) \odot c^*\left(\hat{z}, \widehat{W}_c\right)\right), \tag{2–14}$$

$$\dot{\hat{h}} = -b_h\hat{h} + b_h\left(o\left(\hat{z}, \widehat{W}_o\right) \odot \sigma_c \circ \Psi_c\left(x, \hat{c}, \hat{h}, \hat{\theta}\right)\right), \tag{2–15}$$

respectively, where $\hat{z} \triangleq [x^\top, \hat{h}^\top, 1]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n+l_2+1}$ denotes the augmented input of the LSTM estimate. To facilitate the subsequent stability analysis, let the shorthand notations $\widetilde{\Psi}_c \in \mathbb{R}^{l_2}$ and $\widetilde{\Psi}_h \in \mathbb{R}^{l_2}$ be defined as $\widetilde{\Psi}_c \triangleq \Psi_c\left(x, \hat{c}, \hat{h}, \theta\right) - \widehat{\Psi}_c$, and $\widetilde{\Psi}_h \triangleq \Psi_h\left(x, \hat{c}, \hat{h}, \theta, W_o\right) - \widehat{\Psi}_h$, respectively, where $\widehat{\Psi}_c \triangleq \Psi_c\left(x, \hat{c}, \hat{h}, \hat{\theta}\right)$ and $\widehat{\Psi}_h \triangleq \Psi_h\left(x, \hat{c}, \hat{h}, \hat{\theta}, \widehat{W}_o\right)$. Taking the derivative on both sides of (2–8) and (2–9) and substituting in the LSTM model in (2–5) and the auxiliary error dynamics in (2–10) and (2–11) yields

$$\dot{\tilde{c}} = -b_c\tilde{c} + b_c\widetilde{\Psi}_c + g_e + \dot{\eta}_c, \tag{2–16}$$

$$\dot{\tilde{h}} = -b_h\tilde{h} + b_h\widetilde{\Psi}_h + f_e + \dot{\eta}_h, \tag{2–17}$$

where the functions $f_e\left(x, \tilde{c}, \tilde{h}, \theta, W_o\right) \in \mathbb{R}^{l_2}$ and $g_e\left(x, \tilde{c}, \tilde{h}, \theta\right) \in \mathbb{R}^{l_2}$ are defined as $f_e \triangleq b_h\Psi_h\left(x, c, h, \theta, W_o\right) - b_h\Psi_h\left(x, \hat{c}, \hat{h}, \theta, W_o\right)$ and $g_e \triangleq b_c\Psi_c\left(x, c, h, \theta\right) - b_c\Psi_c\left(x, \hat{c}, \hat{h}, \theta\right)$, respectively. Furthermore, let $\widetilde{\Phi} \triangleq \Phi\left(x, \hat{c}, \hat{h}, \theta, W_o, W_h\right) - \widehat{\Phi}$.

Analytic, stability-driven weight adaptation laws are developed in this section to adjust the weights of the LSTM in real-time using the tracking error, providing a continual learning method. While the adaptation laws ensure online learning of the LSTM, offline training methods (e.g., Adam, Levenberg-Marquardt algorithm (LM)) could be used to initialize the LSTM if pretraining data is available. If no data is available, the LSTM can also be initialized with random weights or using pretraining data taken from a similar system (i.e., transfer learning). Based on the subsequent stability analysis, the weight adaptation laws are designed as

$$\begin{aligned}
&\text{vec}\left(\dot{\hat{\theta}}\right) \triangleq \text{proj}_\theta\left(\Gamma_\theta\left(b_c\widehat{\Psi}_c'^\top\eta_c + b_h\widehat{\Psi}_{h,\theta}'^\top\eta_h + \widehat{\Phi}_\theta'^\top r - \gamma_\theta\text{vec}\left(\hat{\theta}\right)\right)\right),\\
&\text{vec}\left(\dot{\widehat{W}}_o\right) \triangleq \text{proj}_{W_1}\left(\Gamma_o\left(b_h\widehat{\Psi}_{h,W_o}'^\top\eta_h + \widehat{\Phi}_{W_o}'^\top r - \gamma_o\text{vec}\left(\widehat{W}_o\right)\right)\right),\\
&\text{vec}\left(\dot{\widehat{W}}_h\right) \triangleq \text{proj}_{W_2}\left(\Gamma_h\left(\widehat{\Phi}_{W_h}'^\top r - \gamma_h\text{vec}\left(\widehat{W}_h\right)\right)\right),\\
&\text{vec}\left(\dot{\widehat{W}}_{FF}\right) \triangleq \text{proj}_{W_3}\left(\Gamma_{FF}\widehat{\Phi}_{W_{FF}}'^\top r - \gamma_{FF}\text{vec}\left(\widehat{W}_{FF}\right)\right),
\end{aligned} \tag{2–18}$$

where $\gamma_\theta, \gamma_o, \gamma_h, \gamma_{FF} \in \mathbb{R}_{>0}$ denote user-selected constants, $\Gamma_\theta \in \mathbb{R}^{3l_1l_2\times 3l_1l_2}$, $\Gamma_o \in \mathbb{R}^{l_1l_2\times l_1l_2}$, $\Gamma_h \in \mathbb{R}^{l_2n\times l_2n}$, and $\Gamma_{FF} \in \mathbb{R}^{5l_2n\times 5l_2n}$ denote user-selected positive-definite gain matrices, the short-hand notations $\widehat{\Psi}'_c$, $\widehat{\Psi}'_{h,\theta}$, $\widehat{\Psi}'_{h,W_o}$, $\widehat{\Phi}'_\theta$, $\widehat{\Phi}'_{W_o}$, $\widehat{\Phi}'_{W_h}$, and $\widehat{\Phi}'_{W_{FF}}$ denote the Jacobians $\widehat{\Psi}'_c \triangleq \frac{\partial \widehat{\Psi}_c}{\partial \text{vec}(\hat{\theta})}$, $\widehat{\Psi}'_{h,\theta} \triangleq \frac{\partial \widehat{\Psi}_h}{\partial \text{vec}(\hat{\theta})}$, $\widehat{\Psi}'_{h,W_o} \triangleq \frac{\partial \widehat{\Psi}_h}{\partial \text{vec}(\widehat{W}_o)}$, $\widehat{\Phi}'_\theta \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\hat{\theta})}$, $\widehat{\Phi}'_{W_o} \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\widehat{W}_o)}$, $\widehat{\Phi}'_{W_h} \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\widehat{W}_h)}$, and $\widehat{\Phi}'_{W_{FF}} \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\widehat{W}_{FF})}$, respectively, and $\text{proj}(\cdot)$ denotes the projection operator defined in [118, Appendix E, Eq. E.4]. The projection operators $\text{proj}_\theta(\cdot)$, $\text{proj}_{W_1}(\cdot)$, $\text{proj}_{W_2}(\cdot)$, and $\text{proj}_{W_2}(\cdot)$ in (2–18) are used to ensure $\hat{\theta}(t) \in \mathcal{B}_\theta \triangleq \{\varsigma \in \mathbb{R}^{3l_1l_2} : \|\varsigma\| \leq \sqrt{3\overline{W}}\}$, $\widehat{W}_o(t) \in \mathcal{B}_{W_1} \triangleq \{\varsigma \in \mathbb{R}^{l_1l_2} : \|\varsigma\| \leq \overline{W}\}$, $\widehat{W}_h(t) \in \mathcal{B}_{W_2} \triangleq \{\varsigma \in \mathbb{R}^{l_2n} : \|\varsigma\| \leq \overline{W}\}$, and $\widehat{W}_{FF}(t) \in \mathcal{B}_{W_3} \triangleq \{\varsigma \in \mathbb{R}^{5nl_2} : \|\varsigma\| \leq \overline{W}\}$, respectively.

Remark 2.1. The terms η_c and η_h are introduced and implemented in the auxiliary cell and hidden state estimation errors $\tilde{c}$ and $\tilde{h}$ to allow the weight adaptation laws in (18) to adaptively compensate for the uncertainty in the internal dynamics inherent in the LSTM cell through the terms $b_c\widehat{\Psi}'^{\top}_c\eta_c$, $b_h\widehat{\Psi}'^{\top}_{h,\theta}\eta_h$, and $b_h\widehat{\Psi}'^{\top}_{h,W_o}\eta_h$.

The Jacobians $\widehat{\Psi}'_c$, $\widehat{\Psi}'_{h,\theta}$, and $\widehat{\Phi}'_\theta$ can be represented as $\widehat{\Psi}'_c \triangleq [\widehat{\Psi}'_{c,W_c}, \widehat{\Psi}'_{c,W_i}, \widehat{\Psi}'_{c,W_f}]$, $\widehat{\Psi}'_{h,\theta} \triangleq [\widehat{\Psi}'_{h,W_c}, \widehat{\Psi}'_{h,W_i}, \widehat{\Psi}'_{h,W_f}]$, and $\widehat{\Phi}'_\theta \triangleq [\widehat{\Phi}'_{W_c}, \widehat{\Phi}'_{W_i}, \widehat{\Phi}'_{W_f}]$, respectively, where $\widehat{\Psi}'_{c,W_j} \triangleq \frac{\partial \widehat{\Psi}_c}{\partial \text{vec}(\widehat{W}_j)}$, $\widehat{\Psi}'_{h,W_j} \triangleq \frac{\partial \widehat{\Psi}_h}{\partial \text{vec}(\widehat{W}_j)}$, and $\widehat{\Phi}'_{W_j} \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\widehat{W}_j)}$ for all $j \in \{c, i, f\}$. Using (2–5), (2–14), (2–15), the chain rule, the properties of the Hadamard product, and the properties of vectorization, the terms $\widehat{\Psi}'_{c,W_c}$, $\widehat{\Psi}'_{c,W_i}$, and $\widehat{\Psi}'_{c,W_f}$ can be expressed as

$$\begin{aligned}
\widehat{\Psi}'_{c,W_c} &= \text{diag}\left(\sigma_g\left(\widehat{W}_i^\top \hat{z}\right)\right)\sigma'_c\left(\widehat{W}_c^\top \hat{z}\right)\left(I_{l_2} \otimes \hat{z}^\top\right), \\
\widehat{\Psi}'_{c,W_i} &= \text{diag}\left(\sigma_c\left(\widehat{W}_c^\top \hat{z}\right)\right)\sigma'_g\left(\widehat{W}_i^\top \hat{z}\right)\left(I_{l_2} \otimes \hat{z}^\top\right), \\
\widehat{\Psi}'_{c,W_f} &= \text{diag}\left(\hat{c}\right)\sigma'_g\left(\widehat{W}_f^\top \hat{z}\right)\left(I_{l_2} \otimes \hat{z}^\top\right),
\end{aligned} \tag{2–19}$$

respectively. Similarly, using (2–19), the terms $\widehat{\Psi}'_{h,W_j}$ and $\widehat{\Psi}'_{h,W_o}$ can be expressed as

$$\begin{aligned}
\widehat{\Psi}'_{h,W_j} &= \text{diag}\left(\sigma_g\left(\widehat{W}_o^\top \hat{z}\right)\right)\sigma'_c\left(\widehat{\Psi}_c\right)\widehat{\Psi}'_{c,W_j}, \\
\widehat{\Psi}'_{h,W_o} &= \text{diag}\left(\sigma_c\left(\widehat{\Psi}_c\right)\right)\left(\sigma'_g\left(\widehat{W}_o^\top \hat{z}\right)\right)\left(I_{l_2} \otimes \hat{z}^\top\right),
\end{aligned} \tag{2–20}$$

for all $j \in \{c, i, f\}$, respectively. Using (2–6) and the chain rule, the Jacobians $\widehat{\Phi}'_{W_j}$, $\widehat{\Phi}'_{W_o}$, $\widehat{\Phi}'_{W_h}$, and $\widehat{\Phi}'_{W_{FF}}$ can be expressed as $\widehat{\Phi}'_{W_j} = \widehat{W}_h^\top \widehat{\Psi}'_{h,W_j}$, $\widehat{\Phi}'_{W_o} = \widehat{W}_h^\top \widehat{\Psi}'_{h,W_o}$, $\widehat{\Phi}'_{W_h} = I_n \otimes \widehat{\Psi}_h^\top$, and $\widehat{\Phi}'_{W_{FF}} = \widehat{W}_h^\top \sigma' \left(\widehat{W}_{FF}^\top x \right) \left(I_{l_2} \otimes x^\top \right)$, for all $j \in \{c, i, f\}$, respectively. NNs such as the LSTM model in (2–5) are nonlinear in terms of the weights. Moreover, the LSTM model has added complexity due to the three gate units present in the cell architecture. To address the resulting mathematical challenges, a first-order Taylor Series approximation-based error model of the LSTM in (2–5) and (2–6) is given by

$$\begin{aligned}
\widetilde{\Psi}_c &= \widehat{\Psi}'_c \text{vec}\left(\tilde{\theta}\right) + \mathcal{O}_c^2\left(\tilde{\theta}\right), \\
\widetilde{\Psi}_h &= \widehat{\Psi}'_{h,W_o} \text{vec}\left(\widetilde{W}_o\right) + \widehat{\Psi}'_{h,\theta} \text{vec}\left(\tilde{\theta}\right) + \mathcal{O}_h^2\left(\tilde{\theta}, \widetilde{W}_o\right), \\
\widetilde{\Phi} &= \widehat{\Phi}'_{W_h} \text{vec}\left(\widetilde{W}_h\right) + \widehat{\Phi}'_{W_{FF}} \text{vec}\left(\widetilde{W}_{FF}\right) + \widehat{\Phi}'_{W_o} \text{vec}\left(\widetilde{W}_o\right) \\
&\quad + \widehat{\Phi}'_\theta \text{vec}\left(\tilde{\theta}\right) + \mathcal{O}_\Phi^2\left(\tilde{\theta}, \widetilde{W}_o, \widetilde{W}_h, \widetilde{W}_{FF}\right),
\end{aligned} \tag{2–21}$$

where $\mathcal{O}_c^2\left(\tilde{\theta}\right) \in \mathbb{R}^{l_2}$, $\mathcal{O}_h^2\left(\tilde{\theta}, \widetilde{W}_o\right) \in \mathbb{R}^{l_2}$, and $\mathcal{O}_\Phi^2\left(\tilde{\theta}, \widetilde{W}_o, \widetilde{W}_h, \widetilde{W}_{FF}\right) \in \mathbb{R}^n$ denotes the higher-order terms. Using Lemma 2.1, the higher-order terms can be bounded as $\left\|\mathcal{O}_c^2\left(\tilde{\theta}\right)\right\|, \left\|\mathcal{O}_h^2\left(\tilde{\theta}, \widetilde{W}_o\right)\right\|, \left\|\mathcal{O}_\Phi^2\left(\tilde{\theta}, \widetilde{W}_o, \widetilde{W}_h, \widetilde{W}_{FF}\right)\right\| \leq \overline{\mathcal{O}}$, where $\overline{\mathcal{O}} \in \mathbb{R}_{>0}$ denotes a known constant.

2.4 Stability Analysis

To facilitate the subsequent stability analysis, let the concatenated state vector $\zeta : \mathbb{R}_{\geq 0} \to \mathbb{R}^\psi$ and constant $\kappa \in \mathbb{R}_{>0}$ be defined as $\zeta \triangleq [e^\top, r^\top, \eta_c^\top, \tilde{c}^\top, \eta_h^\top, \tilde{h}^\top, \text{vec}(\tilde{\theta})^\top, \text{vec}(\widetilde{W}_h)^\top, \text{vec}(\widetilde{W}_o)^\top, \text{vec}(\widetilde{W}_{FF})^\top]^\top$ and $\kappa \triangleq \min\{\frac{b_c}{2} - \frac{k_{1,c}}{2} - \frac{\|K_{2,c}\|_F}{2}, \frac{b_h}{2} - \frac{k_{1,h}}{2} - \frac{\|K_{2,h}\|_F}{2}, \frac{k_r}{2} - \frac{\|K_{2,c}\|_F}{2} - \frac{\|K_{2,h}\|_F}{2}, \frac{k_{1,c}}{4}, \frac{k_{1,h}}{4}, \gamma_\theta, \gamma_h, \gamma_{FF}, \gamma_o, \alpha\}$, respectively, where $\psi \triangleq 2n + 4l_2 + 4l_1 l_2 + 6nl_2$. Additionally, let the auxiliary function $\widetilde{N} : \mathbb{R}^\psi \to \mathbb{R}$ be defined as $\widetilde{N} \triangleq r^\top j_e + \tilde{c}^\top g_e + \tilde{h}^\top f_e + b_c \widehat{\Psi}'_c \text{vec}\left(\tilde{\theta}\right)^\top (c - \hat{c}) + \left(b_h \widehat{\Psi}'_{h,W_o} \text{vec}\left(\widetilde{W}_o\right) + b_h \widehat{\Psi}'_{h,\theta} \text{vec}\left(\tilde{\theta}\right)\right)^\top \left(h - \hat{h}\right)$, where $\widetilde{N}$ represents a group of terms that appear in the subsequent stability analysis. Applying the mean value theorem-based inequality [119, Appendix A] on the terms

$r^\top j_e$, $\tilde{c}^\top g_e$, and $\tilde{h}^\top f_e$, and bounding $\|r\|$, $\|\tilde{c}\|$, $\left\|\tilde{h}\right\|$, and $\|z\|$ terms with $\|\zeta\|$, the auxiliary function $\widetilde{N}$ can be bounded as $\left\|\widetilde{N}\right\| \leq \rho\left(\|\zeta\|\right)\|\zeta\|^2$, where $\rho\left(\cdot\right)$ denotes an invertible, strictly non-increasing function. Let the open and connected sets $\mathcal{D} \subset \mathbb{R}^{\psi}$ and $\Upsilon \subseteq \mathcal{Z}$ be defined as $\mathcal{D} \triangleq \left\{\varsigma \in \mathbb{R}^{\psi} : \|\varsigma\| < \sqrt{\frac{\beta_1}{\beta_2}}\rho^{-1}\left(\kappa - \lambda\right)\right\}$ and $\Upsilon = \left\{\varsigma \in \mathcal{Z} : \|\varsigma\| < \overline{z}\right\}$, respectively, where $\lambda \in \mathbb{R}_{>0}$ denotes a user-selected constant, $\delta \triangleq \frac{\left(\overline{\mathcal{O}^2}+\overline{\varepsilon}\right)^2}{2k_r} + \frac{\left(2b_c\overline{\mathcal{O}^2}\right)^2}{k_{1,c}} + \frac{\left(2b_h\overline{\mathcal{O}^2}\right)^2}{k_{1,h}} + \frac{\left(b_c\overline{\mathcal{O}^2}\right)^2}{2b_c} + \frac{\left(b_h\overline{\mathcal{O}^2}\right)^2}{2b_h} + 6\gamma_\theta\overline{W} + 2\left(\gamma_h+\gamma_o+\gamma_{FF}\right)\overline{W}^2$, and $\overline{z} \triangleq \left(2+\alpha\right)\omega + 2\overline{q}_d + 2\overline{\dot{q}}_d + \overline{\ddot{q}}_d + \frac{\sqrt{l_2}}{\sqrt{2\left(b_h-\frac{1}{2}\right)}} + 1$. The developed adaptive LSTM-based architecture in (2–12) and (2–18) is shown to be uniformly ultimately bounded (UUB) in the following theorem.

Theorem 2.1. *Consider the model dynamics in* (2–1) *with Assumption 2.1. The Lb-LSTM controller in* (2–12) *and the weight adaptation laws in* (2–18) *ensure the states* ζ *are UUB in the sense that* $\|\zeta\| \leq \sqrt{\frac{\beta_2}{\beta_1}\|\zeta\left(t_0\right)\|^2 e^{-\frac{\lambda}{\beta_1}\left(t-t_0\right)} + \frac{\delta}{\lambda}\left(1-e^{-\frac{\lambda}{\beta_1}\left(t-t_0\right)}\right)}$ *provided the sufficient gain conditions* $\kappa \geq \lambda + \rho\left(\sqrt{\frac{\beta_2}{\beta_1}}\left(\|y(t_0)\| + 2\sqrt{l_2} + 6\overline{W} + 6\overline{W}^2\right)\right)$, $b_h \geq \frac{1}{2}$, $b_c \geq \frac{1}{2\left(1+\sqrt{l_2}\right)}$ *are satisfied, where* $y \triangleq \left[e^\top, r^\top, \eta_c^\top, \eta_h^\top\right]^\top$, $\beta_1 \triangleq \lambda_{min}\{1, \Gamma_\theta^{-1}, \Gamma_o^{-1}, \Gamma_{FF}^{-1}, \Gamma_h^{-1}\}$ *and* $\beta_2 \triangleq \lambda_{max}\{1, \Gamma_\theta^{-1}, \Gamma_o^{-1}, \Gamma_{FF}^{-1}, \Gamma_h^{-1}\}$.

Proof. Consider the Lyapunov candidate function $\mathcal{V}_L : \mathbb{R}^{\psi} \to \mathbb{R}_{\geq 0}$

$$\begin{aligned}\mathcal{V}_L\left(\zeta\right) \triangleq\ & \frac{1}{2}\eta_c^\top\eta_c + \frac{1}{2}\eta_h^\top\eta_h + \frac{1}{2}\tilde{c}^\top\tilde{c} + \frac{1}{2}\tilde{h}^\top\tilde{h} + \frac{1}{2}e^\top e + \frac{1}{2}r^\top Mr \\ & + \frac{1}{2}\text{vec}\left(\widetilde{W}_h\right)^\top\Gamma_h^{-1}\text{vec}\left(\widetilde{W}_h\right) + \frac{1}{2}\text{vec}\left(\widetilde{W}_o\right)^\top\Gamma_o^{-1}\text{vec}\left(\widetilde{W}_o\right) \\ & + \frac{1}{2}\text{vec}\left(\tilde{\theta}\right)^\top\Gamma_\theta^{-1}\text{vec}\left(\tilde{\theta}\right) + \frac{1}{2}\text{vec}\left(\widetilde{W}_{FF}\right)^\top\Gamma_h^{-1}\text{vec}\left(\widetilde{W}_{FF}\right),\end{aligned} \tag{2–22}$$

which can be bounded as $\beta_1\|\zeta\|^2 \leq \mathcal{V}_L\left(\zeta\right) \leq \beta_2\|\zeta\|^2$. Substituting (2–2), (2–3), (2–13), (2–16), and (2–17) into the time derivative of $\mathcal{V}_L$ and canceling cross-terms yields

$$\begin{aligned}\dot{\mathcal{V}}_L =\ & -\alpha e^\top e - k_r r^\top r - k_{1,c}\eta_c^\top\eta_c - k_{1,h}\eta_h^\top\eta_h + \tilde{h}^\top\left(f_e + \dot{\eta}_h\right) - \tilde{h}^\top\left(b_h\tilde{h} - b_h\widetilde{\Psi}_h\right) \\ & - \tilde{c}^\top\left(b_c\tilde{c} - b_c\widetilde{\Psi}_c - g_e - \dot{\eta}_c\right) + r^\top\left(\widetilde{\Phi} + j_e + \varepsilon\left(x_d\right)\right) - \text{vec}\left(\widetilde{W}_o\right)^\top\Gamma_o^{-1}\text{vec}\left(\widehat{W}_o\right)\end{aligned}$$

$$- \text{vec}\left(\widetilde{W}_h\right)^\top \Gamma_h^{-1} \text{vec}\left(\widehat{W}_h\right) - \text{vec}\left(\tilde{\theta}\right)^\top \Gamma_\theta^{-1} \text{vec}\left(\dot{\hat{\theta}}\right) - \text{vec}\left(\widetilde{W}_{FF}\right)^\top \Gamma_{FF}^{-1} \text{vec}\left(\widehat{W}_{FF}\right). \tag{2–23}$$

Using [118, Lemma E.1.IV], $-\widetilde{V}^\top \Gamma^{-1} \text{proj}(\kappa) \leq -\widetilde{V}^\top \Gamma^{-1} \kappa$, where the estimation error $\widetilde{V} \in \mathbb{R}^m$ is defined as $\widetilde{V} \triangleq V - \widehat{V}$ for some $V, \widehat{V}, \in \mathbb{R}^m$ such that $\|V\| \leq \overline{V}$ and $\text{proj}(\cdot)$ ensures $\widehat{V}(t) \in \mathcal{B}_V \triangleq \{\varsigma \in \mathbb{R}^m : \|\varsigma\| \leq \overline{V}\}$, where $\overline{V} \in \mathbb{R}_{>0}$ denotes a known constant. Therefore, substituting in (2–10) and (2–11), the weight adaptation laws in (2–18), the first order Taylor series approximation in (2–21), and the definition of $\widetilde{N}$, and using the facts that $\widetilde{\Psi}_c^\top \tilde{c} = \widetilde{\Psi}_c^\top (c - \hat{c} + \eta_c)$ and $\widetilde{\Psi}_h^\top \tilde{h} = \widetilde{\Psi}_h^\top \left(h - \hat{h} + \eta_h\right)$, yields

$$\begin{aligned}
\dot{\mathcal{V}}_L \leq & -\alpha e^\top e - k_r r^\top r - b_c \tilde{c}^\top \tilde{c} - b_h \tilde{h}^\top \tilde{h} - k_{1,c} \eta_c^\top \eta_c - k_{1,h} \eta_h^\top \eta_h \\
& + r^\top \left(\mathcal{O}^2\left(\tilde{\theta}, \widetilde{W}_o, \widetilde{W}_h, \widetilde{W}_{FF}\right) + \varepsilon(x)\right) + b_h \eta_h \mathcal{O}^2\left(\tilde{\theta}, \widetilde{W}_o\right) + b_c \eta_c \mathcal{O}^2\left(\tilde{\theta}\right) \\
& - \tilde{c}^\top \left(k_{1,c} \eta_c + K_{2,c} r\right) - \tilde{h}^\top \left(k_{1,h} \eta_h + K_{2,h} r\right) + \widetilde{N} + b_h \mathcal{O}^2\left(\tilde{\theta}, \widetilde{W}_o\right)^\top \left(h - \hat{h}\right) \\
& + b_c \mathcal{O}^2\left(\tilde{\theta}\right)^\top (c - \hat{c}) - \text{vec}\left(\tilde{\theta}\right)^\top \left(-\gamma_\theta \text{vec}\left(\hat{\theta}\right)\right) - \text{vec}\left(\widetilde{W}_o\right)^\top \left(-\gamma_o \text{vec}\left(\widehat{W}_o\right)\right) \\
& + \gamma_h \text{vec}\left(\widetilde{W}_h\right)^T \text{vec}\left(\widehat{W}_h\right) + \gamma_{FF} \text{vec}\left(\widetilde{W}_{FF}\right)^\top \text{vec}\left(\widehat{W}_{FF}\right).
\end{aligned} \tag{2–24}$$

Using Young's inequality and the facts that $\left\|\widetilde{N}\right\| \leq \rho(\|\zeta\|)\|\zeta\|^2$, $\hat{\theta} = \theta - \tilde{\theta}$, $\widehat{W}_o = W_o - \widetilde{W}_o$, $\widehat{W}_{FF} = W_{FF} - \widetilde{W}_{FF}$, and $\widehat{W}_h = W_h - \widetilde{W}_h$, (2–24) can be bounded as $\dot{\mathcal{V}}_L \leq -(\kappa - \rho(\|\zeta\|))\|\zeta\|^2 + \delta$. From (2–22), $\|\zeta\| \leq \sqrt{\frac{\mathcal{V}_L}{\beta_1}}$, and therefore $\dot{\mathcal{V}}_L$ can be bounded as $\dot{\mathcal{V}}_L \leq -\left(\kappa - \rho\left(\sqrt{\frac{\mathcal{V}_L}{\beta_1}}\right)\right)\frac{\mathcal{V}_L}{\beta_1} + \delta$. Selecting κ according to Theorem 1 ensures $\|\zeta(t_0)\|$ is bounded as $\|\zeta(t_0)\| < \sqrt{\frac{\beta_1}{\beta_2}}\rho^{-1}(\kappa - \lambda)$. Thus, when all trajectories are initialized in $\mathcal{D}$, $\dot{\mathcal{V}}_L$ can be further bounded as $\dot{\mathcal{V}}_L \leq -\frac{\lambda}{\beta_1}\mathcal{V}_L + \delta$, which implies $\mathcal{V}_L(t) \leq \mathcal{V}_L(t_0)e^{-\frac{\lambda}{\beta_1}(t-t_0)} + \frac{\delta\beta_1}{\lambda}(1 - e^{-\frac{\lambda}{\beta_1}(t-t_0)})$. Then, [120, Def. 4.6] can be invoked to conclude that ζ is UUB such that

$$\|\zeta\| \leq \mu \triangleq \sqrt{\frac{\beta_2}{\beta_1}\|\zeta(t_0)\|^2 e^{-\frac{\lambda}{\beta_1}(t-t_0)} + \frac{\delta}{\lambda}\left(1 - e^{-\frac{\lambda}{\beta_1}(t-t_0)}\right)}.$$

To show $z \in \mathcal{Z}$, and therefore the universal function approximation property holds, let $\xi \triangleq [e^\top, r^\top]^\top$ and let $\omega = \rho^{-1}(\kappa - \lambda)$. Thus, if $\|\zeta(t_0)\| \leq \omega\sqrt{\frac{\beta_1}{\beta_2}}$, then $\|\xi(t)\| \leq \omega$, and

therefore $\|e(t)\| \leq \omega$ and $\|r(t)\| \leq \omega$. Hence, using (2–2), (2–3), and Lemma 2.1, $\|z\|$ can be bounded as $\|z\| \leq (2+\alpha)\omega + 2\overline{q}_d + 2\overline{\dot{q}}_d + \overline{\ddot{q}}_d + \frac{\sqrt{l_2}}{\sqrt{2\left(b_h - \frac{1}{2}\right)}} + 1$ provided the sufficient gain conditions $b_h \geq \frac{1}{2}$, $b_c \geq \frac{1}{2\left(1+\sqrt{l_2}\right)}$ are met for Lemma 2.1 to hold. Therefore, if $\zeta(t_0) \in \mathcal{D}$, then $z \in \Upsilon \subseteq \mathcal{Z}$. Since $\zeta \in L_\infty$, $q, \dot{q} \in \mathcal{L}_\infty$. That and the fact that $\hat{c}, \hat{h}, \hat{\theta}, \widehat{W}_o, \widehat{W}_h, \widehat{W}_{FF} \in \mathcal{L}_\infty$ by design imply $\tau \in \mathcal{L}_\infty$. □

2.5 Simulations

To demonstrate the performance and efficacy of the developed Lb-LSTM control design, simulations were performed on two-link robot manipulator, which is modeled as [119, Eqn. (80)]

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= M^{-1}(x_1)\left(\left(-V(x_1, x_2) - F_d\right)x_2 + u\right) - F_s(x_2), \end{aligned} \tag{2–25}$$

where $x_1 \triangleq [x_{11}\ x_{12}]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ and $x_2 \triangleq [x_{21}\ x_2]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ denote the angular position and velocity of the two links, respectively, and $F_d \in \mathbb{R}^{2\times 2}$, $F_s : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$, $M : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$, and $V : \mathbb{R}^4 \to \mathbb{R}^{2\times 2}$ denote the dynamic friction, static friction, inertia matrix and the centripetal-Coriolis matrix, respectively, as defined in [119].

To demonstrate the advantages of using the Lb-LSTM architecture instead of a feedforward DNN architecture in the adaptive controller, the results are compared with the DNN-based adaptive controller developed in [104] as the baseline. The baseline adaptive DNN-based controller in [104] is $\tau \triangleq \widehat{\Phi}_{DNN} + k_r r + e$, where the DNN estimate $\widehat{\Phi}_{DNN}$ was updated according to the weight adaptation laws defined in [104, Eqns. (7)-(8)]. The LSTM model in (2–5) was used with tanh activation functions for the feedforward term and $l_2 = 12$ neurons and was compared to 3 baseline fully-connected DNN architectures, DNN1, DNN2, and DNN3, with 1, 2, and 5 hidden layers each, respectively, with tanh activation functions. DNN1 and DNN2 had 12 neurons in each layer and DNN3 had 14 neurons. The weights of all NNs were randomly initialized with a uniform distribution with values ranging between -1 and 1. The gains were selected

as $\alpha = 15$, $k_r = 50$, $k_{1,c} = 5$, $k_{1,h} = 5$, $K_{2,c} = 0.1 \cdot [I_2 \; 0_{2\times 10}]^\top$, $K_{2,h} = 0.1 \cdot [I_2 \; 0_{2\times 10}]^\top$, $b_c = 5$, $b_h = 1$, $\Gamma_\theta = 40 \cdot I_{3l_1l_2}$, $\Gamma_o = 40 \cdot I_{l_1l_2}$, $\Gamma_h = 40 \cdot I_{l_2n}$, $\Gamma_{FF} = 40 \cdot I_{5l_2n}$, and $\gamma_\theta = \gamma_o = \gamma_h = \gamma_{FF} = 0.01$ for the adaptive LSTM controller. For the baseline controllers, the gains were selected as $\alpha = 15$, $k_r = 50$, and $\Gamma_j = 24 \cdot I_{L_jL_{j+1}}$ $\forall j \in \{0, ..., k\}$. For a fair comparison, the same robust control gains were used for each controllers. The NN gains and parameters (e.g., the learning gains and activation functions) were empirically adjusted to achieve the best performance for each network. For all simulations, the desired trajectory $q_d(t) \triangleq [q_{d,1}, q_{d,2}]^\top \in \mathbb{R}^2$ was selected as $q_d \triangleq \begin{bmatrix} \frac{\pi}{3}\sin\left(\frac{\pi}{4}t\right) \\ \frac{\pi}{2}\sin\left(\frac{\pi}{2}t\right) \end{bmatrix} \in \mathbb{R}^2$ [rad], and the simulations were performed for 25 s with the initial conditions $q(0) = [1.0472, -0.5236]^\top$ [rad] and $\dot{q}(0) = [0, 0]^\top$ [rad/s].

The results of DNN1 are shown in Fig. 2-2 and are compared to the proposed Lb-LSTM controller in Table 2-1. The results of the DNN2 are shown in Fig. 2-3 and are compared to the proposed Lb-LSTM controller in Table 2-1. When compared to the adaptive feedforward NN architectures, the adaptive LSTM architecture resulted in a significant improvement in both tracking and function approximation error performance, with reduced control input. Although all four adaptive NN architectures compensated for the uncertainty in the dynamics and achieved tracking, the LSTM provided improved tracking performance with a significant improvement in both function approximation performance and control effort, when compared to the feedforward NN architectures (Table 2-1). Moreover, the LSTM provided twofold and fourfold faster tracking and function approximation error convergence, respectively, compared to DNN3 with better transient behavior (Fig. 2-4). When compared to the adaptive controller DNN3, the LSTM-based controller and developed weight adaptation law resulted in 25.1% and 68.4% improvement in the tracking error and function approximation error, respectively, while requiring 33.6% reduced control effort, as shown in Table 2-1.

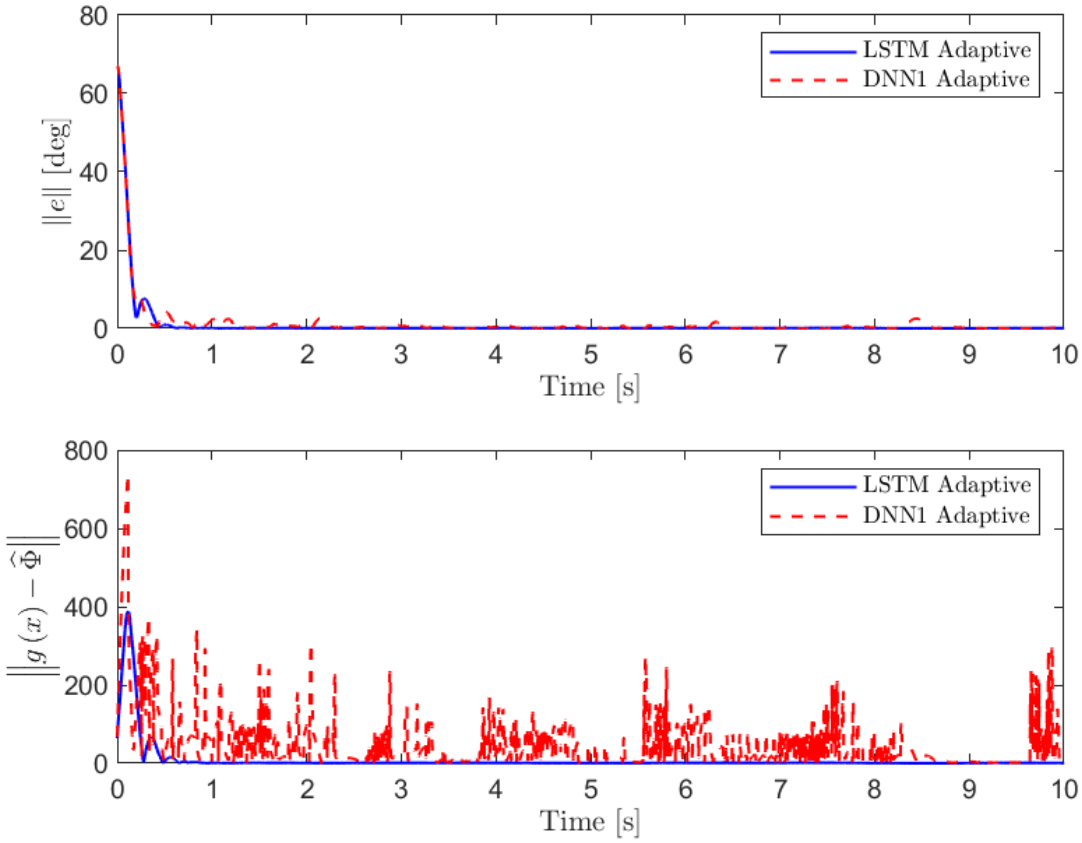

Figure 2-2. (a) Norm of tracking error over time for DNN1 compared to the LSTM controller, (b) Norm of function approximation error over time for DNN1 compared to the LSTM controller. To better exhibit transient performance, only the first 10 s of the simulation are shown.

Table 2-1. Performance Comparison Results

NN Architecture	$\|e\|$[deg]	$\left\|g(x) - \widehat{\Phi}\right\|$	$\|\tau\|$[N·m]
DNN1	0.6374	36.46	33.36
DNN2	0.6360	22.21	19.59
DNN3	0.5302	11.84	9.260
LSTM	0.3970	3.748	6.147

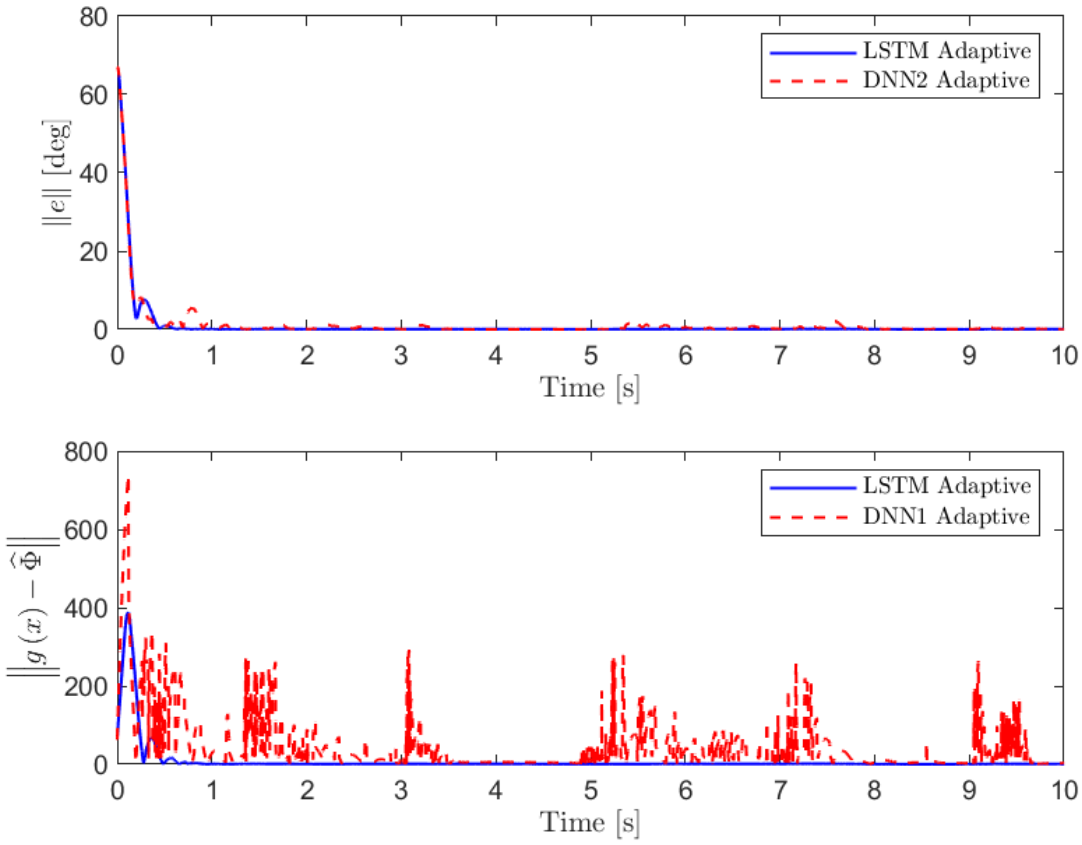

Figure 2-3. (a) Norm of tracking error over time for DNN2 compared to the LSTM controller, (b) Norm of function approximation error over time for the DNN2 compared to the LSTM controller. To better exhibit transient performance, only the first 10 s of the simulation are shown.

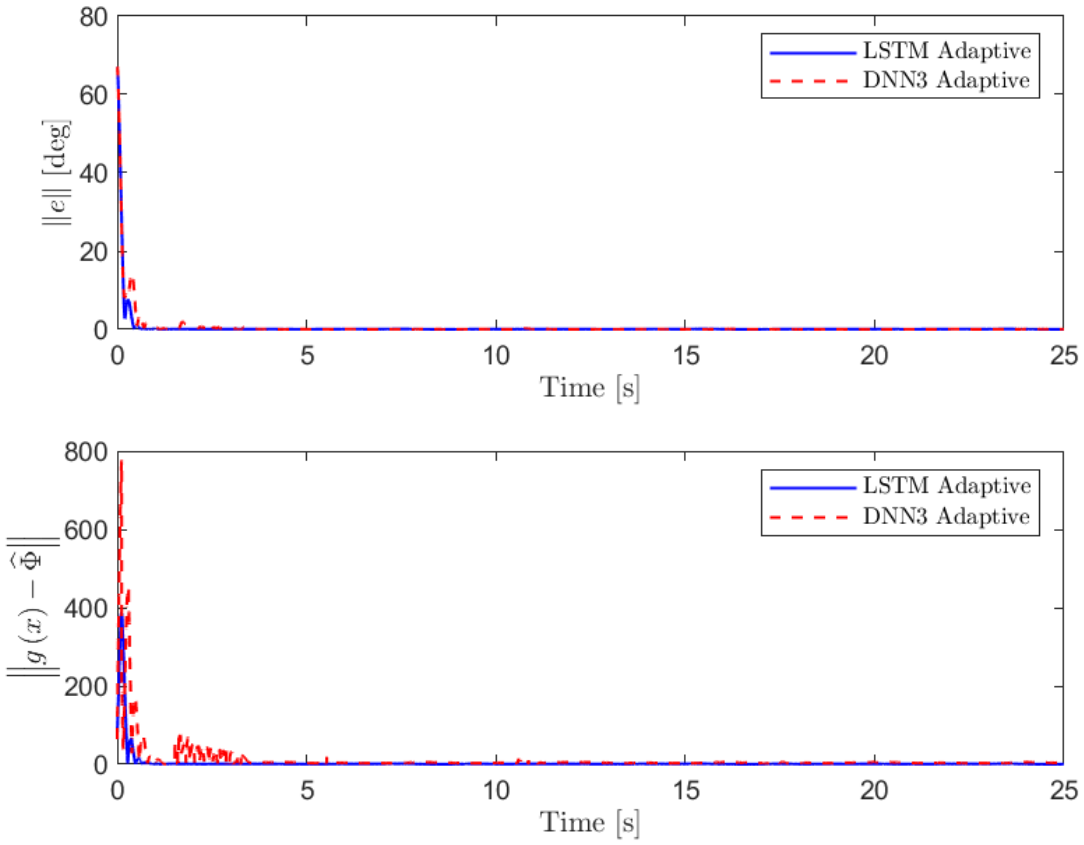

Figure 2-4. (a) Norm of the tracking error over time for the developed adaptive LSTM controller and the baseline adaptive controller DNN3. (b) Norm of the function approximation error over time for the developed adaptive LSTM controller and the baseline adaptive controller DNN3.

2.6 Conclusions

An adaptive LSTM-based controller was developed for general uncertain Euler-Lagrange systems. Leveraging the dynamic structure and internal memory inherent in LSTMs, the developed continuous-time Lb-LSTM architecture is able to leverage time dependencies in the system dynamics and capture time-varying accumulative effects in the system dynamics that static, feedforward NNs cannot. Stability-driven weight adaptation laws are developed for the Lb-LSTM weights in real-time, eliminating the need for offline pre-training. However, the Lb-LSTM control architecture was developed for a tracking control problem, and therefore the system error implemented in the adaptation laws are known. Thus, additional difficulties arise when developing an adaptive Lb-LSTM architecture when the system error is unknown, e.g., the state estimation error in the system identification problem investigated in Chapter 3.

CHAPTER 3
LYAPUNOV-BASED LONG SHORT-TERM MEMORY (LB-LSTM) NEURAL NETWORK-BASED ADAPTIVE OBSERVER

3.1 Introduction

LSTMs excel at capturing short- and long-term dependencies, making them powerful tools for system identification and state estimation. However, due to mathematical challenges involved in developing adaptation methods for LSTMs, their training is predominantly limited to offline methods. The arising difficulty is due in part to the nonlinear structure of the LSTM cell, which makes deriving stable, online learning algorithms difficult. Additional difficulties arise due to the fact that the state estimation error is unknown and can therefore not be used in the adaptation law. This chapter develops a Lyapunov-based (Lb-) LSTM observer for state estimation in nonlinear systems using the continuous-time LSTM model developed in Chapter 2. The Lb-LSTM weights adapt in real-time using Lyapunov-based stability-driven adaptation laws, and therefore, this is the first online learning result for LSTM-based observers with stability guarentees. To compensate for the fact that the estimation error is unknown, a dynamic filter is developed and implemented in the adaptation law. A nonsmooth Lyapunov-based stability analysis ensures state estimation error convergence and stability of the overall Lb-LSTM architecture. To validate the developed observer design, simulations were performed to estimate the unknown angular velocity states of a two-link robot manipulator. The developed method yielded a 41.13% improvement in the root mean square estimation error when compared to an adaptive RNN observer.

3.2 System Dynamics

Consider a second-order nonlinear system modeled as

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= g\left(x, u\right), \end{aligned} \tag{3–1}$$

where $x \triangleq \left[x_1^\top \; x_2^\top\right]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n}$ and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denote the generalized state and control input of the system, respectively, and $g : \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}^n$ denotes an unknown function. The development in this chapter is restricted to second-order systems for the ease of illustration, but can be extended for n^{th} order systems using the observer development in [121]. The following assumptions facilitate the subsequent observer development.

Assumption 3.1. The unknown function g is continuously differentiable.

Assumption 3.2. The system is assumed to be bounded-input bounded-output stable. Furthermore, the control input is assumed to be sufficiently smooth such that $\|u\| \leq \overline{u}$ and $\|\dot{u}\| \leq \overline{\dot{u}}$, where $\overline{u}, \overline{\dot{u}} \in \mathbb{R}_{>0}$ denote known constants. Therefore, the state can be bounded as $\|x\| \leq \overline{x}$, and there is a known, compact set $\mathcal{Z} \subseteq \mathbb{R}^{2n} \times \mathbb{R}^m$ such that $z \in \mathcal{Z}$, where $z \triangleq \left[x^\top \; u^\top\right]^\top$ and $\overline{x} \in \mathbb{R}_{>0}$ denotes a known constant.

Assumption 3.3. The system dynamics in (3–1) are observable.

Assumption 3.4. The state x_1 is assumed to be known.

3.3 Observer Development

Since only the first state x_1 is available for state feedback, the objective is to design an adaptive Lyapunov-based (Lb-) LSTM observer to estimate the unknown system dynamics. Let $\hat{x} \triangleq \left[\hat{x}_1^\top \; \hat{x}_2^\top\right]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n}$ denote the observer state estimate. To quantify the objective of the observer, an estimation error $\tilde{x}_1 : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and an auxiliary estimation error $r : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are defined as

$$\tilde{x}_1 \triangleq x_1 - \hat{x}_1, \tag{3–2}$$

$$r \triangleq \dot{\tilde{x}}_1 + \alpha \tilde{x}_1 + \eta, \tag{3–3}$$

respectively, where $\alpha \in \mathbb{R}_{>0}$ denotes a user-selected constant and $\eta : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the output of a dynamic filter designed to compensate for the lack of availability of r since x_2 is unknown. Based on the subsequent stability analysis, the dynamic filter is

designed as [1]

$$\begin{aligned}\eta &\triangleq p-(\alpha+k_r)\,\tilde{x}_1,\\ \dot{p} &\triangleq -(k_r+2\alpha)\,p-\nu+\left((\alpha+k_r)^2+1\right)\tilde{x}_1,\\ \dot{\nu} &\triangleq p-\alpha\nu-(\alpha+k_r)\,\tilde{x}_1,\end{aligned} \tag{3–4}$$

where $k_r \in \mathbb{R}_{>0}$, $p : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and $\nu : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denote a user-defined constant, an internal filter variable, and auxiliary filter output, respectively. The filter variables p and ν are initialized such that $p(0) = (\alpha + k_r)\,\tilde{x}_1(0)$ and $\nu(0) = 0$. The developed dynamic filter in (3–4) uses $\tilde{x}_1$ as an input, yielding the filter outputs ν and η. The internal variable p of the filter is utilized to generate the output η, circumventing the need for the unmeasurable derivative of the estimation error $\dot{\tilde{x}}_1$. From (3–3) and (3–4), the dynamic filter can be related to the unmeasurable auxiliary estimation error r as

$$r = \dot{e} + \alpha e, \tag{3–5}$$

where $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is an auxiliary error defined as

$$e \triangleq \tilde{x}_1 + \nu. \tag{3–6}$$

3.3.1 Adaptive Long Short-Term Memory (LSTM) Architecture

The improved memory and dynamical behavior of LSTMs make them well-suited for estimating dynamic system states, where long-term memory and accurate representation of accumulative effects are crucial for making informed predictions. Thus, integrating an LSTM model into the observer design can improve predictive accuracy and enable robust modeling. Using the LSTM cell model developed in Chapter 2 (Fig. 2-1) an LSTM can be modeled in continuous-time as

$$\begin{aligned}\dot{c} &= -b_c c + b_c \Psi_c\left(\zeta, c, \theta\right),\\ \dot{h} &= -b_h h + b_h \Psi_h\left(\zeta, c, \theta\right).\end{aligned} \tag{3–7}$$

The concatenated state vector $\zeta \in \mathbb{R}^{l_1}$ is augmented with a 1 to incorporate a bias term and is defined as $\zeta \triangleq [z^\top, h^\top, 1]^\top$, where $z \in \mathbb{R}^{2n+m}$ denotes the LSTM input, $l_1 \triangleq 2n + m + l_2 + 1$, and $l_2 \in \mathbb{R}_{>0}$ denotes the user-selected number of neurons in the weight matrices. The weight matrices are denoted by $W_c^\top, W_i^\top, W_f^\top, W_o^\top \in \mathbb{R}^{l_2 \times l_1}$, and $W_h^\top \in \mathbb{R}^{n \times l_2}$, where $\theta \triangleq [\text{vec}\,(W_c)^\top, \text{vec}\,(W_i)^\top, \text{vec}\,(W_f)^\top, \text{vec}\,(W_o)^\top, \text{vec}\,(W_h)^\top]^\top \in \mathbb{R}^{4l_2 l_1 + l_2 n}$. The functions $\Psi_c(\zeta, c, \theta) \in \mathbb{R}^{l_2}$ and $\Psi_h(\zeta, c, \theta) \in \mathbb{R}^{l_2}$ in the cell and hidden state dynamics are defined as

$$\Psi_c(\zeta, c, \theta) \triangleq f(\zeta, W_f) \odot c + i(\zeta, W_i) \odot c^*(\zeta, W_c),$$
$$\Psi_h(\zeta, c, \theta) \triangleq o(\zeta, W_o) \odot (\sigma_c \circ \Psi_c(\zeta, c, \theta)),$$

respectively. To ensure the output of the LSTM has the appropriate dimensions, a fully-connected layer is added to the LSTM cell using the output weight matrix W_h. Thus, the output of the LSTM $\Phi(\zeta, c, \theta) \in \mathbb{R}^n$ can be modeled as

$$\Phi(\zeta, c, \theta) = W_h^\top \Psi_h(\zeta, c, \theta). \tag{3–8}$$

Using the universal function approximation property, the system dynamics $g(x, u)$ can be modeled using the LSTM architecture in (3–7) as $g(x, u) = \Phi(\zeta, c, \theta) + \varepsilon(z)$, where $\varepsilon : \mathbb{R}^{2n+m} \to \mathbb{R}^n$ denotes a function reconstruction error that can be bounded as $\|\varepsilon\|_{z \in \mathcal{Z}} \leq \bar{\varepsilon}$, where $\bar{\varepsilon} \in \mathbb{R}_{>0}$ denotes a bounding constant. Therefore, taking the time-derivative of (3–3) and using (3–1) and (3–2) yields

$$\dot{r} = \Phi(\zeta, c, \theta) + \varepsilon(z) - \dot{\hat{x}}_2 + \alpha \dot{\tilde{x}}_1 + \dot{\eta}, \tag{3–9}$$

where $\dot{\eta}$ can be determined by taking the time derivative of η and using (3–3) and (3–4) to yield

$$\dot{\eta} = -(\alpha + k_r) r - \alpha \eta + \tilde{x}_1 - \nu. \tag{3–10}$$

3.3.2 Observer Design

While LSTMs have improved memory capabilities compared to other NN architectures, their application has not been explored for real-time state estimation. Offline approaches remain static and do not allow updates of the NN weights irrespective of system performance, resulting in a lack of robustness to uncertainty in the dynamics. In contrast, adaptive NN-based observers dynamically update the weights online through stability-driven methods. Motivated by the adaptability to changing conditions and improved robustness of adaptive NN architectures, a Lb-LSTM using the shorthand notation $\widehat{\Phi} \triangleq \Phi\left(\hat{\zeta}, \hat{c}, \hat{\theta}\right)$ is constructed and an observer is designed as

$$\begin{aligned} \dot{\hat{x}}_1 &\triangleq \hat{x}_2, \\ \dot{\hat{x}}_2 &= \widehat{\Phi} + k_s \text{sgn}\left(e\right) + \chi, \end{aligned} \tag{3–11}$$

where $k_s \in \mathbb{R}_{>0}$ denotes a user-selected constant, $\hat{\zeta} \triangleq [\hat{z}^\top, \hat{h}^\top, 1]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$, $\hat{z} \triangleq [\hat{x}^\top \ u^\top]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{2n+m}$ denotes the input of the LSTM estimate, $\hat{\theta} \triangleq [\text{vec}(\widehat{W}_c)^\top, \text{vec}(\widehat{W}_i)^\top, \text{vec}(\widehat{W}_f)^\top, \text{vec}(\widehat{W}_o)^\top, \text{vec}(\widehat{W}_h)^\top]^\top : \mathbb{R}_{\geq 0} \to \mathbb{R}^{4l_2l_1+l_2n}$ denotes the adaptive weight estimates, and $\chi \in \mathbb{R}^n$ denotes an auxiliary term defined as $\chi \triangleq -\left(3\alpha + k_r\right)\eta + \left(\alpha^2 + 2\right)\tilde{x}_1 - \nu$. Substituting the observer in (3–11) into (3–9) and adding and subtracting $\Phi\left(\hat{\zeta}, \hat{c}, \theta\right)$ yields

$$\begin{aligned} \dot{\tilde{x}}_1 &= \dot{x}_1 - \dot{\hat{x}}_1, \\ \dot{r} &= \widetilde{\Phi} + \varepsilon\left(z\right) - k_s \text{sgn}\left(e\right) - \chi + \alpha\dot{\tilde{x}}_1 + \dot{\eta} + N_1, \end{aligned} \tag{3–12}$$

where $\widetilde{\Phi} \triangleq \Phi\left(\hat{\zeta}, \hat{c}, \theta\right) - \widehat{\Phi}$ and $N_1 \triangleq \Phi\left(\zeta, c, \theta\right) - \Phi\left(\hat{\zeta}, \hat{c}, \theta\right)$. Using the bounds on the tanh and sigmoid activation functions, the auxiliary function N_1 can be bounded as $\|N_1\| \leq C_1$, where $C_1 \triangleq 2\overline{W}\sqrt{l_2}$.

3.3.3 Weight Adaptation Laws

Based on the subsequent stability analysis, the weight adaptation law is designed as

$$\dot{\hat{\theta}} \triangleq \Gamma \widehat{\Phi}'^{\top} e, \tag{3–13}$$

where $\Gamma \in \mathbb{R}^{(4l_1l_2+l_2n)\times(4l_1l_2+l_2n)}$ denotes a user-selected positive-definite adaptation gain matrix and the short-hand notation $\widehat{\Phi}'$ denotes the Jacobian $\widehat{\Phi}' \triangleq \frac{\partial \widehat{\Phi}}{\partial \hat{\theta}}$.

The Jacobian $\widehat{\Phi}'$ can be represented as $\widehat{\Phi}' \triangleq [\widehat{\Phi}'_{W_c}, \widehat{\Phi}'_{W_i}, \widehat{\Phi}'_{W_f}, \widehat{\Phi}'_{W_o}, \widehat{\Phi}'_{W_h}]$, where $\widehat{\Phi}'_{W_j} \triangleq \frac{\partial \widehat{\Phi}}{\partial \text{vec}(\widehat{W}_j)}$ for all $j \in \{c, i, f, o, h\}$. Using (3–8) and the chain rule, the Jacobians $\widehat{\Phi}'_{W_j}$ and $\widehat{\Phi}'_{W_h}$ can be expressed as $\widehat{\Phi}'_{W_j} = \widehat{W}_h^{\top} \widehat{\Psi}'_{h,W_j}$ and $\widehat{\Phi}'_{W_h} = I_n \otimes \Psi_h^{\top}\left(\hat{\zeta}, \hat{c}, \hat{\theta}\right)$, for all $j \in \{c, i, f, o\}$, respectively, where $\widehat{\Psi}'_{h,W_j} \triangleq \frac{\partial \Psi_h\left(\hat{\zeta},\hat{c},\hat{\theta}\right)}{\partial \text{vec}(\widehat{W}_j)}$ $\forall j \in \{c, i, f, o\}$. Using (3–7), the properties of the Hadamard product, the properties of the vectorization operator, and the chain rule, the terms $\widehat{\Psi}'_{h,W_j}$ and $\widehat{\Psi}'_{h,W_o}$ can be expressed as

$$\widehat{\Psi}'_{h,W_j} = \text{diag}\left(\sigma_g\left(\widehat{W}_o^{\top}\hat{\zeta}\right)\right)\sigma_c'\left(\Psi_c\left(\hat{\zeta},\hat{c},\hat{\theta}\right)\right)\widehat{\Psi}'_{c,W_j},$$
$$\widehat{\Psi}'_{h,W_o} = \text{diag}\left(\sigma_c\left(\widehat{\Psi}_c\right)\right)\left(\sigma_g'\left(\widehat{W}_o^{\top}\hat{\zeta}\right)\right)\left(I_{l_2} \otimes \hat{\zeta}^{\top}\right),$$

for all $j \in \{c, i, f\}$, respectively, where $\widehat{\Psi}'_{c,W_j} \triangleq \frac{\partial \Psi_c\left(\hat{\zeta},\hat{c},\hat{\theta}\right)}{\partial \text{vec}(\widehat{W}_j)}$ $\forall j \in \{c, i, f\}$. Likewise, using (3–7), the terms $\widehat{\Psi}'_{c,W_c}$, $\widehat{\Psi}'_{c,W_i}$, and $\widehat{\Psi}'_{c,W_f}$ can be expressed as

$$\widehat{\Psi}'_{c,W_c} = \text{diag}\left(\sigma_g\left(\widehat{W}_i^{\top}\hat{\zeta}\right)\right)\sigma_c'\left(\widehat{W}_c^{\top}\hat{\zeta}\right)\left(I_{l_2} \otimes \hat{\zeta}^{\top}\right),$$
$$\widehat{\Psi}'_{c,W_i} = \text{diag}\left(\sigma_c\left(\widehat{W}_c^{\top}\hat{\zeta}\right)\right)\sigma_g'\left(\widehat{W}_i^{\top}\hat{\zeta}\right)\left(I_{l_2} \otimes \hat{\zeta}^{\top}\right),$$
$$\widehat{\Psi}'_{c,W_f} = \text{diag}\left(\hat{c}\right)\sigma_g'\left(\widehat{W}_f^{\top}\hat{\zeta}\right)\left(I_{l_2} \otimes \hat{\zeta}^{\top}\right),$$

respectively, where $\sigma_j'(y) \triangleq \frac{\partial}{\partial z}\sigma_j(z)\big|_{z=y}$, $\forall j \in \{c, g\}$, $y \in \mathbb{R}^{l_2}$.

3.4 Stability Analysis

To address the mathematical issues arising due to the nonlinear parameterization, a first-order Taylor Series approximation of the LSTM in (3–7) and (3–8) is constructed,

given by $\widetilde{\Phi} = \widehat{\Phi}'\tilde{\theta} + \mathcal{O}^2\left(\tilde{\theta}\right)$, where $\mathcal{O}^2\left(\tilde{\theta}\right) \in \mathbb{R}^n$ denotes the higher-order terms. Thus, substituting this into (3–12) yields the closed-loop error system

$$\begin{aligned} \dot{\tilde{x}}_1 &= \dot{x}_1 - \dot{\hat{x}}_1, \\ \dot{r} &= \widehat{\Phi}'\tilde{\theta} + N_2 - k_s \text{sgn}\left(e\right) - \chi + \alpha\dot{\tilde{x}}_1 + \dot{\eta}, \end{aligned} \tag{3–14}$$

where $N_2 \triangleq N_1 + \mathcal{O}^2\left(\tilde{\theta}\right) + \varepsilon\left(z\right)$.

To facilitate the stability analysis, let a candidate Lyapunov function $\mathcal{V}_L : \mathbb{R}^{\psi} \to \mathbb{R}_{\geq 0}$ be defined as

$$\mathcal{V}_L\left(\xi\right) \triangleq \frac{1}{2}\eta^{\top}\eta + \frac{1}{2}\nu^{\top}\nu + \frac{1}{2}\tilde{x}_1^{\top}\tilde{x}_1 + \frac{1}{2}r^{\top}r + P + \frac{\alpha}{2}\tilde{\theta}^{\top}\Gamma^{-1}\tilde{\theta}, \tag{3–15}$$

where the concatenated state vector $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\psi}$ is defined as $\xi \triangleq [\tilde{x}_1^{\top},\ r^{\top},\ \eta^{\top},\ \nu^{\top},\ \tilde{\theta}^{\top},\ \sqrt{P}]^{\top}$, $\psi \triangleq 4n + 4l_1l_2 + l_2n + 1$, and $P : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denotes a subsequently designed P-function. The candidate Lyapunov function in (3–15) can be bounded as $\beta_1 \left\|\xi\right\|^2 \leq \mathcal{V}_L\left(\xi\right) \leq \beta_2 \left\|\xi\right\|^2$, where $\beta_1 \triangleq \min\left\{\frac{1}{2}, \frac{\alpha}{2}\lambda_{\min}\left\{\Gamma\right\}\right\}$ and $\beta_2 \triangleq \max\left\{1, \frac{\alpha}{2}\lambda_{\max}\left\{\Gamma\right\}\right\}$. Let the open and connected sets $\mathcal{D} \subset \mathbb{R}^{\psi}$ and $\mathcal{S} \subset \mathbb{R}^{\psi}$ be defined as $\mathcal{D} \triangleq \left\{\varsigma \in \mathbb{R}^{\psi} : \left\|\varsigma\right\| < \sqrt{\frac{\beta_1}{\beta_2}}\omega\right\}$ and $\mathcal{S} = \left\{\varsigma \in \mathbb{R}^{\psi} : \left\|\varsigma\right\| < \omega\right\}$, respectively, where $\omega \in \mathbb{R}_{>0}$ denotes a bounding constant. The universal function approximation property only holds on the compact domain $\mathcal{Z}$. Therefore, the following stability analysis must guarantee $z\left(t\right) \in \mathcal{Z}$ for all $t \geq 0$ which is achieved by a stability result that constrains ξ to a compact domain, specifically that $\xi\left(t\right) \in \mathcal{S}$ for all $t \geq 0$ by initializing $\xi\left(0\right) \in \mathcal{D}$.

Taking the time-derivative of $\mathcal{V}_L$ using the chain rule for nonsmooth systems in [122, Theorem 2.2], substituting in the closed-loop dynamics in (3–14), and canceling the coupling terms yields

$$\begin{aligned} \dot{\mathcal{V}}_L \overset{a.a.t.}{\in} & r^{\top}\left(\widehat{\Phi}'\tilde{\theta} + N_2 - k_s K\left[\text{sgn}\right]\left(e\right) - \chi + \alpha\dot{\tilde{x}}_1 + \dot{\eta}\right) \\ & + \tilde{x}_1^{\top}\dot{\tilde{x}}_1 + \eta^{\top}\dot{\eta} + \nu^{\top}\dot{\nu} + \dot{P} - \alpha\tilde{\theta}^{\top}\Gamma^{-1}\dot{\hat{\theta}}. \end{aligned} \tag{3–16}$$

Substituting (3–6) and the weight adaptation law in (3–13) into (3–16) yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\in} r^\top \left(N_2 - k_s K \left[\text{sgn}\right](e) - \chi + \alpha \dot{\tilde{x}}_1 + \dot{\eta} + N_1\right) + \tilde{x}_1^\top \dot{\tilde{x}}_1 + \eta^\top \dot{\eta} + \nu^\top \dot{\nu} + \dot{P} + \dot{e}^\top \widehat{\Phi}' \tilde{\theta}. \tag{3–17}$$

Using the design of the dynamic filter in (3–4)-(3–6) and canceling like terms yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\in} r^\top \left(N_2 - k_s K \left[\text{sgn}\right](e) - k_r r\right) - \left(\alpha + k_r\right) \tilde{x}_1^\top \tilde{x}_1 - \alpha \eta^\top \eta + \nu^\top \left(\alpha \tilde{x}_1 - \alpha \nu\right) + \dot{e}^\top \widehat{\Phi}' \tilde{\theta} + \dot{P}. \tag{3–18}$$

Convergence of the estimation errors using the developed adaptive LSTM architecture and overall observer design is guaranteed in the following theorem.

To facilitate the subsequent stability analysis, let $N_3 \triangleq \widehat{\Phi}' \tilde{\theta}$. Using Assumptions 3.1 and 3.2, Lemma 1 in [123], and the facts that $N_2 + N_3 = g\left(x, u\right) - \Phi\left(\hat{\zeta}, \hat{c}, \hat{\theta}\right)$ and the LSTM Φ is continuously differentiable by design, the bounds

$$\left\|N_2\right\| \leq \kappa_1, \left\|N_3\right\| \leq \kappa_2, \left\|\dot{N}_2 + \dot{N}_3\right\| \leq \kappa_3, \tag{3–19}$$

hold when $\xi \in \mathcal{S}$, where κ_1, κ_2, $\kappa_3 \in \mathbb{R}_{>0}$ are known positive bounding constants.

Theorem 3.1. *Consider the system in* (3–1)*. Let Assumptions 3.1-3.4 hold. The Lb-LSTM observer in* (3–11) *and the weight adaptation law in* (3–13) *ensure asymptotic estimation error convergence in the sense that* $\left\|x_2 - \hat{x}_2\right\| \to 0$ *as* $t \to \infty$*, provided* $\xi\left(0\right) \in \mathcal{D}$*, the following gain condition is satisfied.*

$$k_s \geq \kappa_1 + \kappa_2 + \frac{1}{\alpha - 1}\left(\alpha \kappa_2 + \kappa_3\right), \quad \alpha > 1. \tag{3–20}$$

Proof. Consider the Lyapunov candidate function in (3–15). The P-function in (3–15) is designed as

$$P\left(t\right) \triangleq \mathrm{e}^{-t} * \left(\left(\alpha - 1\right)\left(k_s \left\|e\right\|_1 - e^\top \left(N_2 + N_3\right)\right)\right) + k_s \left\|e\right\|_1$$

$$+ \mathrm{e}^{-t} * \left(\alpha e^{\top} N_3 + e^{\top} \left(\dot{N}_2 + \dot{N}_3 \right) \right) - e^{\top} \left(N_2 + N_3 \right). \tag{3–21}$$

Using [124, Lemma 4] and (3–19), it can be shown that $P(t) \geq 0$ for all $t \geq 0$, provided the sufficient gain conditions in (3–20) are satisfied.

Therefore, substituting the time-derivative of (3–21) into (3–18) and using Young's inequality, (3–18) can be further bounded as

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} -\lambda \left\| y \right\|^2,$$

when $\xi \in \mathcal{S}$, where $\lambda \triangleq \min\left\{k_r, \frac{\alpha}{2} + k_r, \alpha, \frac{\alpha}{2}, 1\right\}$ and $y \triangleq \left[r^{\top},\ \tilde{x}_1^{\top},\ \eta^{\top},\ \nu^{\top},\ \sqrt{P}\right]^{\top}$ denotes a concatenated state vector. To show $\xi \in \mathcal{S}$ for all $t \geq 0$, using the fact that $\dot{\mathcal{V}}_L(\xi(t)) \overset{a.a.t.}{\leq} 0$ and (3–15) implies $\xi(t)$ can be bounded as $\|\xi(t)\| \leq \sqrt{\frac{\beta_2}{\beta_1}} \|\xi(0)\|$ when $\xi \in \mathcal{S}$. Thus, if $\|\xi(0)\| \leq \omega \sqrt{\frac{\beta_1}{\beta_2}}$, then $\|\xi(t)\| \leq \omega$ for all $t \geq 0$. Therefore, if the states ξ are initialized such that $\xi(0) \in \mathcal{D}$, then $\xi \in \mathcal{S}$ for all $t \geq 0$. Since $\xi \in \mathcal{S}$ when $\xi(0) \in \mathcal{D}$, the bounds in (3–19) hold. To show $z \in \mathcal{Z}$ and the universal function approximation property holds, let the open and connected set $\Upsilon \subseteq \mathcal{Z}$ be defined as $\Upsilon = \{\varsigma \in \mathcal{Z} : \|\varsigma\| < \overline{x} + (3+\alpha)\omega + \overline{u}\}$. Using the fact that $\|\xi(t)\| \leq \omega$ for all $t \geq 0$, it can be shown that $\|\tilde{x}_1(t)\| \leq \omega$, $\|\eta(t)\| \leq \omega$, and $\|r(t)\| \leq \omega$ for all $t \geq 0$. Hence, using (3–2) and (3–3), $\hat{z}$ can be bounded as $\|\hat{z}\| \leq \overline{x} + (3+\alpha)\omega + \overline{u}$. Therefore, if $\xi(0) \in \mathcal{D}$, then $\hat{z} \in \Upsilon \subseteq \mathcal{Z}$. Using (3–15) and the fact that $\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} 0$ implies $\tilde{x}_1, \nu, \eta, r, P, \tilde{\theta} \in \mathcal{L}_\infty$. Therefore, the observer $\hat{x} \in \mathcal{L}_\infty$ and $\hat{\theta} \in \mathcal{L}_\infty$. Since $\hat{x}, \hat{\theta} \in \mathcal{L}_\infty$ and the function Φ is continuously differentiable, $\dot{\hat{\theta}} \in \mathcal{L}_\infty$. The extension of LaSalle-Yoshizawa corollary in [125, Corollary 1] can be invoked to show $\|\tilde{x}_1\| \to 0$, $\|\nu\| \to 0$, $\|\eta\| \to 0$, and $\|r\| \to 0$ as $t \to \infty$. Therefore, using (3–3) and (3–4), it can be further shown that $\|x_2 - \hat{x}_2\| \to 0$ as $t \to \infty$. □

3.5 Simulation Results

Comparative simulations were performed to demonstrate the performance of the developed Lb-LSTM observer, where the results were compared to the adaptive

Table 3-1. Performance Comparison

Architecture	$\|x_2 - \hat{x}_2\|$ [deg/s]	$\|\tilde{x}_1\|$ [deg]
RNN	0.3856	0.1065
LSTM	0.2270	0.0595
Percent Improvement	41.13%	44.09%

shallow RNN observer in [1]. Simulations were performed to estimate the unknown angular velocity states of the two-link robot manipulator system modeled in (2–25). A proportional derivative (PD) controller was selected as $u = 15\left(x_1 - x_{1d}\right) + 5\left(\hat{x}_2 - x_{2d}\right)$ to track the desired position trajectory $x_{d,1} = \begin{bmatrix} \frac{\pi}{6}\sin\left(\frac{\pi}{2}t\right) & \frac{\pi}{6}\sin\left(\frac{\pi}{2}t\right) \end{bmatrix}^{\top}$. Each simulation was performed for 50 seconds with a step size of 0.001 seconds, and noise was added to the joint angle measurements from a uniform distribution $U\left(-0.5, 0.5\right)$ [deg]. The observer and dynamic filter gains in (3–3), (3–4), and (3–11) were selected as $k_r = 20$, $k_s = 0.05$, $\alpha = 60$, and $b_c = b_h = 10$. For a fair comparison, the same robust gains and dynamic filter was used for both observers, and the comparative observer was constructed by replacing the LSTM estimate in (3–11) with the adaptive shallow RNN estimate developed in [1]. The LSTM and RNN estimates were composed of $l_2 = 12$ neurons each with $l_1 = 19$ for the LSTM. The LSTM and RNN weights were randomly selected from a uniform distribution $U\left(-2, 2\right)$, with learning gains of $\Gamma = 20 \cdot I_{4l_1l_2+l_2n}$ for the LSTM and $\Gamma_{W_f} = 20 \cdot I_{24}$ and $\Gamma_{V_{f1}} = 20 \cdot I_{96}$ for the shallow RNN. The performance results of the two simulations are shown in Table 3-1 and Figure 3-1. The developed Lb-LSTM observer yielded a 41.13% improvement in the root mean square estimation error. While the estimation errors settled for both observers after approximately 1 s, the Lb-LSTM observer yielded a significant improvement in the steady state performance. As evident from Figure 3-1, the adaptive shallow RNN observer produced small, frequent spikes in the estimation error, which contributed to a higher root mean square estimation error. Ultimately, the developed Lb-LSTM architecture and adaptive observer design resulted in significant improvements in estimation accuracy of the the unknown state x_2 when compared to the baseline adaptive shallow RNN observer.

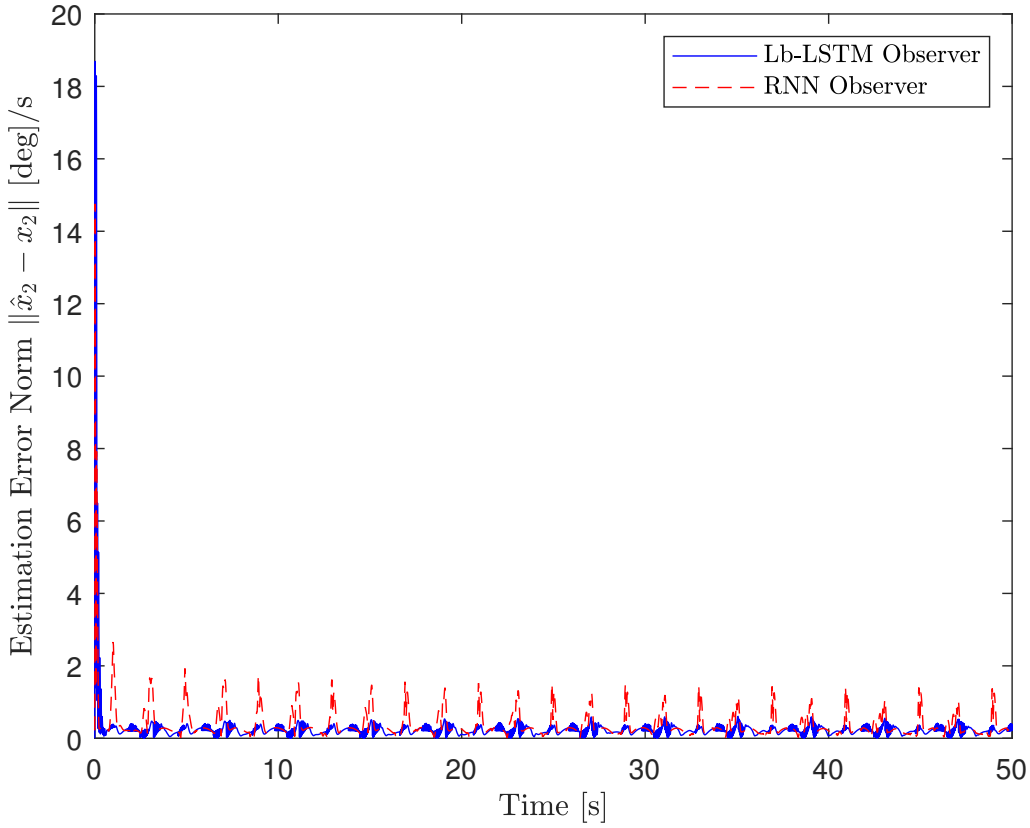

Figure 3-1. Plot of the estimation error norm $\|\hat{x}_2 - x_2\|$ over time for the developed Lb-LSTM observer compared to the adaptive shallow RNN observer in [1].

3.6 Conclusions

In this chapter, an Lb-LSTM observer is designed for nonlinear system state estimation. The developed Lb-LSTM architecture adapts in real-time through Lyapunov stability-driven adaptation laws, making it the first LSTM observer to learn the system dynamics online in a stability-driven manner. A nonsmooth Lyapunov-based stability analysis is performed to guarantee convergence of the state estimation error and stability of the overall Lb-LSTM observer design. Comparative simulations are provided to estimate the unknown angular velocity states of a two-link robot manipulator. The simulation results show the developed method yielded a 41.13% improvement in the root mean square estimation error when compared to the adaptive shallow RNN observer in [1]. However, the developed observer design and adaptation law does consider the control design or tracking error in the stability analysis, motivating the OFB-based controller in Chapter 4. OFB-based control would bring additional challenges due to the fact that the adaptation law would have to integrate two error objectives: one for the tracking control problem and one for system identification.

CHAPTER 4
ADAPTIVE OUTPUT FEEDBACK CONTROL USING LYAPUNOV-BASED DEEP RECURRENT NEURAL NETWORKS (LB-DNNS)

4.1 Introduction

Motivated by the dynamic behavior of RNNs, this chapter leverages the adaptive RNN-based control development in Chapter 2 and the adaptive RNN-based observer development in Chapter 3 to develop an adaptive Lb-DRNN OFB controller for uncertain nonlinear systems. Specifically, an Lb-DRNN observer is designed to adaptively estimate the unknown states of the system and is integrated into an OFB control framework. To ensure real-time adaptation, the DRNN weights are dynamically adjusted through Lyapunov-based adaptation laws using both tracking and estimation error feedback, making this the first online learning result for a DRNN-based OFB controller. A Lyapunov-based stability analysis proves asymptotic estimation and tracking error convergence. Validation simulation experiments were performed on an unmanned underwater vehicle (UUV) system that resulted in a 27.98% and 89.94% improvement in linear and angular tracking error, respectively, when compared to a shallow RNN-based OFB controller.

4.2 Problem Formulation

4.2.1 Model Dynamics

Consider a second order nonlinear system modeled as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + g(x_1)\, u, \end{aligned} \tag{4–1}$$

where $x \triangleq \left[x_1^\top \; x_2^\top\right]^\top \in \mathbb{R}^{2n}$ and $u \in \mathbb{R}^m$ denote the generalized state and control input of the system, respectively, $f : \mathbb{R}^{2n} \to \mathbb{R}^n$ denotes a continuously differentiable function, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denotes a continuous function. While the control effectiveness matrix is assumed to be known, the control development in [101] can be used to account for an uncertain, linearly parametrizable control effectiveness g. Like Chapter

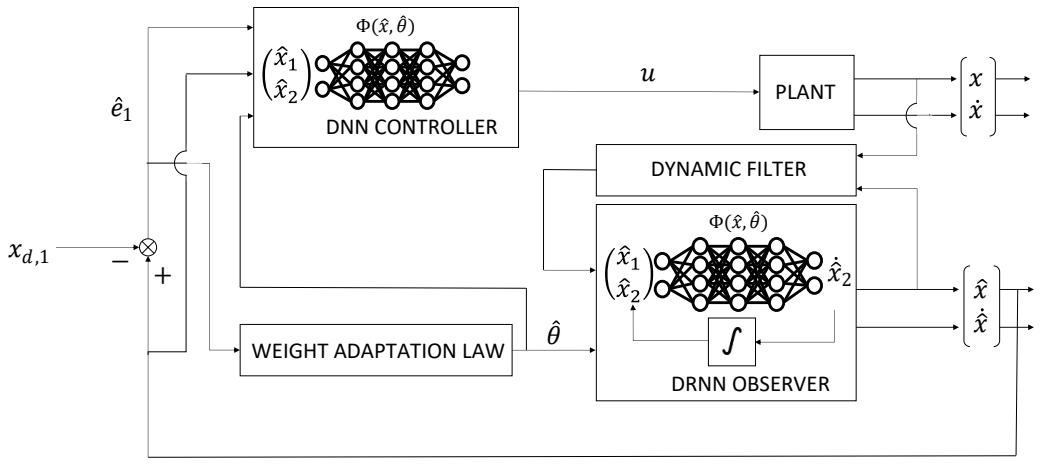

Figure 4-1. Block diagram of adaptive DRNN-based output feedback controller.

3, a second order nonlinear system was considered for ease of exposition and the development in [121] can be used with the developed adaptive DRNN OFB controller for control of N-th order strict nonlinear systems with an unmeasurable state x_N. The control objective is to design an adaptive DRNN controller to track a desired trajectory $x_{d,1} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ despite the system uncertainty and unavailability of the state x_2. It is assumed that the desired trajectory $x_{d,1}$ is designed to be sufficiently smooth such that $|x_{d,1}(t)| \leq \overline{x_d}$, $|\dot{x}_{d,1}(t)| \leq \overline{\dot{x}_d}$, and $|\ddot{x}_{d,1}(t)| \leq \overline{\ddot{x}_d}$ for all $t \in \mathbb{R}_{\geq 0}$, where $\overline{x}_d, \overline{\dot{x}_d}, \overline{\ddot{x}_d} \in \mathbb{R}_{>0}$ denote known constants. The model in (4–1) is assumed to satisfy Assumptions 3.3 -3.4 and the following assumption.

Assumption 4.1. The function g is known and full row rank.

4.2.2 Deep Recurrent Neural Network Model

An adaptive DRNN architecture is developed to estimate the unknown model dynamics in (4–1) in real-time and is implemented in a subsequently designed output feedback-based controller, as shown in Fig. 4-1. Similar to the continuous-time LSTM representation developed in Chapter 2, a continuous-time DRNN can be modeled using the discrete-time DRNN representation in [126] as

$$\dot{h} = -bh + W_k^\top \phi_k \circ ... \circ W_1^\top \phi_1 \circ W_0^\top y, \tag{4–2}$$

where $b \in \mathbb{R}_{>0}$ denotes a user-selected time constant, $y \in \mathbb{R}^{L_{k+1}+2n+1}$ denotes a concatenated state vector defined as $y \triangleq \left[h^\top \; x_a^\top\right]^\top$, $x_a \in \mathbb{R}^{2n+1}$ denotes the augmented input defined as $x_a \triangleq \left[x^\top \; 1\right]^\top$, and $h \in \mathbb{R}^{L_{k+1}}$ denotes the hidden state. The weight matrices and smooth activation functions[1] at the j^{th} layer are denoted by $W_j^\top \in \mathbb{R}^{L_{j+1}\times L_j}$ and $\phi_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ for all $j \in \{0, ..., k\}$ and $j \in \{1, ..., k\}$, respectively, where $\theta \triangleq \left[\text{vec}\left(W_0\right)^\top \; ... \; \text{vec}\left(W_k\right)^\top\right]^\top \in \mathbb{R}^{\sum_{j=0}^{k} L_j L_{j+1}}$ and $L_j \in \mathbb{R}_{>0}$ for all $j \in \{0, ..., k\}$ denotes the number of neurons in the j^{th} layer. Since the hidden state is an input to the first hidden layer, $W_0^\top \in \mathbb{R}^{L_1\times(L_{k+1}+2n+1)}$. To incorporate a bias term, x_a and ϕ_j are augmented with 1 for all $j \in \{1, ..., k\}$. For DNNs with multiple types of activation functions at each layer, ϕ_j may be modeled as $\phi_j \triangleq \left[\varsigma_{j,1}, \varsigma_{j,2}, ..., \varsigma_{j,L_{j-1}}, 1\right]^\top$, where $\varsigma_{j,i} : \mathbb{R} \to \mathbb{R}$ for all $j \in \{1, ..., k\}$ and $i \in \{1, ..., L_j\}$ denotes the activation function at the i^{th} node of the j^{th} layer.

To facilitate the subsequent analysis, a recursive representation of the adaptive Lb-DRNN architecture can be modeled as

$$\Phi_j = \begin{cases} W_j^\top \phi_j\left(\Phi_{j-1}\right), & j = \{1, ..., k\}, \\ W_j^\top y, & j = 0, \end{cases} \tag{4–3}$$

where $\Phi_j\left(x, \theta\right) \in \mathbb{R}^{L_{j+1}}$ denotes the output of the j^{th} layer defined as $\Phi_j \triangleq W_j^\top \phi_j \circ W_{j-1}^\top \phi_{j-1} \circ ... \circ W_1^\top \phi_1 \circ W_0^\top y$ for all $j \in \{0, ..., k\}$. From (4–3), the Lb-DRNN in (4–2) can be represented as $\dot{h} = -bh + \Phi\left(x, \theta\right)$, where $\Phi\left(x, \theta\right) \triangleq \Phi_k\left(x, \theta\right)$.

The Lb-DRNN architecture in (4–2) can then be used to model the unknown system dynamics in (4–1) using the unknown state x_2 as the hidden state h. Using the universal function approximation property, the dynamics in (4–1) can be modeled using an

[1] The adaptive DRNN architecture in (4–2) does not consider nonsmooth activation functions for notational simplicity. However, the switched analysis in [104] can be used with the developed method to incorporate nonsmooth activation functions into the RNN architecture.

adaptive DRNN architecture as

$$\dot{x}_2 = -bx_2 + \Phi\left(x, \theta\right) + \varepsilon\left(x\right) + g\left(x_1\right) u. \tag{4–4}$$

4.3 Control Development

Since the state x_2 is not available and the dynamics in (4–1) are unknown and unstructured, an adaptive DRNN observer is constructed to facilitate the control development. Thus, an adaptive DRNN observer is designed to estimate x_2. To quantify the control objectives, the estimation error $\tilde{x}_1 \in \mathbb{R}^n$ and tracking error $e_1 \in \mathbb{R}^n$ are defined as

$$\begin{aligned} \tilde{x}_1 &\triangleq x_1 - \hat{x}_1, \\ e_1 &\triangleq x_1 - x_{d,1}, \end{aligned} \tag{4–5}$$

respectively, for state estimates $\hat{x} \triangleq \left[\hat{x}_1^\top \ \hat{x}_2^\top\right]^\top$. Using the estimation and tracking errors, auxiliary estimation and tracking errors $\xi,\ r \in \mathbb{R}^n$ are defined as

$$\begin{aligned} \xi &\triangleq \dot{\tilde{x}}_1 + \alpha \tilde{x}_1 + \eta, \\ r &\triangleq \dot{e}_1 + \alpha e_1 + \eta, \end{aligned} \tag{4–6}$$

respectively, where $\alpha \in \mathbb{R}_{>0}$ denotes a user-selected constant and $\eta \in \mathbb{R}^n$ denotes the output of a dynamic filter introduced to compensate for the lack of direct measurements of the state x_2. Based on the subsequent stability analysis, the dynamic filter is designed as [127]

$$\begin{aligned} \eta &\triangleq p - \left(\alpha + k_r\right) \tilde{x}_1, \\ \dot{\nu} &\triangleq p - \alpha\nu - \left(\alpha + k_r\right) \tilde{x}_1, \\ \dot{p} &\triangleq -\left(k_r + 2\alpha\right) p - \nu + \left(\left(\alpha + k_r\right)^2 + 1\right) \tilde{x}_1 + e_1, \end{aligned} \tag{4–7}$$

where $k_r \in \mathbb{R}_{>0}$ denotes a user-selected constant, $p \in \mathbb{R}^n$ denotes an internal filter variable, and $\nu \in \mathbb{R}^n$ denotes the auxiliary output of the filter. The filter variables p and ν are initialized such that $p(0) = (\alpha + k_r)\tilde{x}_1(0)$ and $\nu(0) = 0$, respectively.

The filter in (4–7) uses the estimation error $\tilde{x}_1$ and tracking error e_1 as inputs and produces the two filter outputs ν and η. The internal filter variable p is used to generate the signal η without involving the unavailable derivative of the estimation error $\dot{\tilde{x}}_1$. From (4–6) and (4–7), the dynamic filter can be related to the unknown auxiliary estimation error ξ and unknown auxiliary tracking error r as

$$\begin{aligned} \xi &= \dot{e}_{\text{es}} + \alpha e_{\text{es}}, \\ r &= \dot{e}_{\text{tr}} + \alpha e_{\text{tr}}, \end{aligned} \tag{4–8}$$

where $e_{\text{es}} \in \mathbb{R}^n$ and $e_{\text{tr}} \in \mathbb{R}^n$ are auxiliary errors defined as $e_{\text{es}} \triangleq \tilde{x}_1 + \nu$ and $e_{\text{tr}} \triangleq e_1 + \nu$, respectively. To facilitate the subsequent stability analysis, taking the time derivative of η and using (4–6) and (4–7) yields

$$\dot{\eta} = -(\alpha + k_r)\xi - \alpha\eta + \tilde{x}_1 + e_1 - \nu. \tag{4–9}$$

4.3.1 Observer Design

Based on the subsequent stability analysis, the adaptive Lb-DRNN observer for the uncertain nonlinear system in (4–1) is designed as

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 \\ \dot{\hat{x}}_2 &= -b\hat{x}_2 + \Phi\left(\hat{x}, \hat{\theta}\right) + g(x_1)u + \beta_1 \text{sgn}(e_{\text{es}}) + \chi, \end{aligned} \tag{4–10}$$

where sgn$[\cdot]$ denotes the element-wise sign function, $\hat{\theta} \triangleq \left[\text{vec}\left(\widehat{W}_0\right)^\top, ..., \text{vec}\left(\widehat{W}_k\right)^\top\right]^\top$, $\beta_1 \in \mathbb{R}_0$ denotes a user-defined constant, and $\widehat{W}_j^\top \in \mathbb{R}^{L_{j+1} \times L_j}$ denotes the weight estimates for all $j \in \{0, ..., k\}$. The auxiliary term $\chi \in \mathbb{R}^n$ is designed based on the

stability analysis as

$$\chi \triangleq -\left(\gamma\left(k_r+\alpha\right)+2\alpha\right)\eta+\left(\gamma-\alpha\right)\nu+\tilde{x}_1-\nu, \tag{4–11}$$

where $\gamma \in \mathbb{R}_{>0}$ denotes a user-selected constant. As is typical in observer design approaches (see [128]), the state x_1 is also approximated, even though it is measurable, as a means to provide feedback for the construction of the estimate for the unmeasurable state x_2.

Taking the derivative of ξ and substituting (4–4)-(4–6), (4–10), and (4–11) and adding and subtracting $\Phi\left(\hat{x},\theta\right)$ yields

$$\begin{aligned}\dot{\xi} = &-b\dot{\tilde{x}}_1+\Phi\left(\hat{x},\theta\right)-\Phi(\hat{x},\hat{\theta})+\Phi\left(x,\theta\right)-\Phi\left(\hat{x},\theta\right)+\varepsilon\left(x\right)\\ &-\beta_1\text{sgn}\left(e_{\text{es}}\right)-\chi+\alpha\left(\xi-\alpha\tilde{x}_1-\eta\right)+\dot{\eta}.\end{aligned} \tag{4–12}$$

To facilitate the subsequent development and to address the mathematical challenges posed by the nested nonlinearity of the DRNN architecture, a first-order Taylor series approximation-based error model is evaluated as [129]

$$\Phi\left(\hat{x},\theta\right)-\Phi\left(\hat{x},\hat{\theta}\right)=\widehat{\Phi}'\tilde{\theta}+\mathcal{O}^2\left(\hat{x},\tilde{\theta}\right), \tag{4–13}$$

where $\mathcal{O}^2\left(\hat{x},\tilde{\theta}\right)$ denotes higher-order terms. Substituting (4–13) into (4–12) yields the closed-loop error system

$$\dot{\xi}=N_1+\widehat{\Phi}'\tilde{\theta}-\beta_1\text{sgn}\left(e_{\text{es}}\right)-\chi+\alpha\left(\xi-\alpha\tilde{x}_1-\eta\right)+\dot{\eta}, \tag{4–14}$$

where the auxiliary term $N_1 \in \mathbb{R}^n$ is defined as $N_1 \triangleq -b\dot{\tilde{x}}_1+\Phi\left(x,\theta\right)-\Phi\left(\hat{x},\theta\right)+\mathcal{O}^2\left(\hat{x},\tilde{\theta}\right)+\varepsilon\left(x\right)$.

4.3.2 Control Design

An OFB controller is designed using the developed adaptive Lb-DRNN architecture and observed state estimates $\hat{x}$. The Lb-DRNN weights are adjusted online using subsequently designed stability-driven weight adaptation laws that allow the developed

OFB control design to estimate the unknown states and system dynamics in real-time. Based on the subsequent stability analysis, the control input is designed as

$$u \triangleq g\left(x_1\right)^{+}\left[-\left(-b\hat{x}_2+\Phi\left(\hat{x},\hat{\theta}\right)\right)-\beta_2 \text{sgn}\left(e_{\text{tr}}\right)+\ddot{x}_{d,1}-\left(k_r+\alpha\right)\left(\dot{\hat{e}}_1+\alpha\hat{e}_1\right)-\alpha^2 e_1-\nu\right], \tag{4–15}$$

where $\hat{e}_1 \triangleq \hat{x}_1 - x_{d,1}$ and $\beta_2 \in \mathbb{R}_0$ denotes a user-defined control gain. Taking the time-derivative of r, substituting (4–4)-(4–6) and (4–15), adding and subtracting $\Phi\left(\hat{x},\theta\right)$, and using the Taylor series-based approximation in (4–13) yields the closed-loop error system

$$\dot{r} = N_1 + \widehat{\Phi}'\tilde{\theta} - \left(k_r+\alpha\right)\left(\dot{\hat{e}}_1+\alpha\hat{e}_1\right)+\dot{\eta}-\beta_2 \text{sgn}\left(e_{\text{tr}}\right)+\alpha\left(r-\eta\right)+\nu. \tag{4–16}$$

4.3.3 Adaptive Weight Update Laws

In this section, a Lyapunov stability-driven weight adaptation law is developed for the DRNN architecture. The weight adaptation law allows the developed output feedback controller to adaptively compensate for the uncertain model dynamics of the system while ensuring stability guarantees. Based on the subsequent stability analysis, the DRNN weight adaptation law is designed as

$$\dot{\hat{\theta}} \triangleq \Gamma\widehat{\Phi}'^{\top}\left(e_{\text{es}}+e_{\text{tr}}\right), \tag{4–17}$$

where $\Gamma \in \mathbb{R}^{\sum_{j=0}^{k} L_j L_{j+1} \times \sum_{j=0}^{k} L_j L_{j+1}}$ and $\widehat{\Phi}' \in \mathbb{R}^{2n \times \sum_{j=0}^{k} L_j L_{j+1}}$ denote a user-selected positive-definite gain matrix and a shorthand notation denoting the Jacobian $\widehat{\Phi}' \triangleq \left[\widehat{\Phi}'_0, ..., \widehat{\Phi}'_k\right]$, where the shorthand notation $\widehat{\Phi}'_j$ is defined as $\widehat{\Phi}'_j \triangleq \frac{\partial \Phi_j\left(\hat{x},\hat{\theta}\right)}{\partial \hat{\theta}}$, for all $j \in \{0, ..., k\}$. Using the chain rule, the DRNN model in (4–3), and the properties of the vectorization operator, the terms $\widehat{\Phi}'_0$ and $\widehat{\Phi}'_j$ can be expressed as

$$\widehat{\Phi}'_0 \triangleq \left(\prod_{l=1}^{\overset{\curvearrowleft}{k}} \widehat{W}_l^{\top}\hat{\phi}'_l\right)\left(I_{L_1} \otimes \hat{x}_a^{\top}\right)$$

$$\widehat{\Phi}_j' \triangleq \left(\prod_{l=j+1}^{\overset{\curvearrowleft}{k}} \widehat{W}_l^\top \hat{\phi}_l' \right) \left(I_{L_{j+1}} \otimes \hat{\phi}_j^\top \right), \ \forall j \in \{1, ..., k\},$$

respectively, where $\hat{x}_a \in \mathbb{R}^{2n+1}$ denotes the augmented RNN input $\hat{x}_a \triangleq \left[\hat{x}^\top 1\right]^\top$, and the shorthand notations $\hat{\phi}_j$ and $\hat{\phi}_j'$ are defined as $\hat{\phi}_j \triangleq \phi_j\left(\Phi_{j-1}\left(\hat{x}, \hat{\theta}\right)\right)$ and $\hat{\phi}_j' \triangleq \phi_j'\left(\Phi_{j-1}\left(\hat{x}, \hat{\theta}\right)\right)$ for all $j \in \{1, ..., k\}$, respectively.

4.4 Stability Analysis

To facilitate the subsequent development, let the function $\mathcal{V}_L : \mathbb{R}^\psi \to \mathbb{R}_{\geq 0}$ be defined as

$$\mathcal{V}_L(\zeta) \triangleq \frac{\gamma}{2}\tilde{x}_1^\top \tilde{x}_1 + \frac{1}{2}\xi^\top \xi + \frac{\gamma}{2} e_1^\top e_1 + \frac{1}{2} r^\top r + \frac{\gamma}{2}\eta^\top \eta + \frac{\gamma}{2}\nu^\top \nu + P + \frac{\alpha}{2}\tilde{\theta}^\top \Gamma^{-1}\tilde{\theta}, \tag{4–18}$$

where the concatenated state vector $\zeta \in \mathbb{R}^\psi$ is defined as $\zeta \triangleq \left[z^\top \ \sqrt{P}\ \tilde{\theta}^\top\right]^\top$, $z \triangleq \left[\tilde{x}_1^\top \ \xi^\top \ e_1^\top \ r^\top \ \eta^\top \ \nu^\top\right]^\top$, and $\psi \triangleq 6n + 1 + \sum_{j=0}^{k} L_j L_{j+1}$. The term $P : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a subsequently designed positive P-function used to account for mismatches that appear in the function $\mathcal{V}_L$ due to the weight adaptation law in (4–17) being expressed in terms of the known errors e_{es} and e_{tr} rather than the unknown auxiliary estimation and tracking errors ξ and r. The function $\mathcal{V}_L$ in (4–18) satisfies

$$\lambda_1 \|\zeta\|^2 \leq \mathcal{V}_L \leq \lambda_2 \|\zeta\|^2, \tag{4–19}$$

where the auxiliary constants λ_1 and λ_2 are defined as $\lambda_1 \triangleq \frac{1}{2}\min\left\{1, \gamma, \alpha\lambda_{\min}\left\{\Gamma^{-1}\right\}\right\}$ and $\lambda_2 \triangleq \frac{1}{2}\max\left\{1, \gamma, \alpha\lambda_{\max}\left\{\Gamma^{-1}\right\}\right\}$, respectively. Since the closed-loop error system has a discontinuous right hand side, a Filippov regularization is applied on the system to facilitate the subsequent analysis [130]. Let $\partial\mathcal{V}_L$ denote the Clarke gradient of $\mathcal{V}_L$ defined in [131, p. 39]. Since $\zeta \mapsto \mathcal{V}_L(\zeta)$ is continuously differentiable, $\partial\mathcal{V}_L(\zeta) = \{\nabla\mathcal{V}_L(\zeta)\}$, where ∇ denotes the standard gradient operator. Based on the chain rule in [122, Thm 2.2], it can be verified that $t \to \mathcal{V}_L(\zeta(t))$ satisfies the differential inclusion $\dot{\mathcal{V}}_L \overset{a.a.t.}{\in} \bigcap_{\xi \in \partial\mathcal{V}_L(\varsigma)} \xi^\top(\psi, t) K[h](\varsigma, t)$ for $\varsigma \triangleq \left[z^\top \ P\ \tilde{\theta}^\top\right]^\top$. Taking the derivative of $\mathcal{V}_L(\zeta)$,

substituting in (4–7), (4–9), and (4–14), and canceling cross-terms yields

$$\dot{\mathcal{V}}_L \overset{a.a.t}{\in} -\alpha\gamma\tilde{x}_1^\top\tilde{x}_1 + \gamma e_1^\top\dot{e}_1 + r^\top\dot{r} - \alpha\gamma\nu^\top\nu - k_r\xi^\top\xi \\ + \xi^\top\left(\widehat{\Phi}'\tilde{\theta} - \beta_1 K\left[\text{sgn}\right](e_{\text{es}}) + e_1 + N_1\right) - \gamma\eta^\top(\alpha\eta + e_1) + \dot{P} - \alpha\tilde{\theta}^\top\Gamma^{-1}\dot{\hat{\theta}}.$$

Substituting in (4–6) and (4–9), the closed-loop tracking error system in (4–16), and the weight adaptation law in (4–17) yields

$$\dot{\mathcal{V}}_L \overset{a.a.t}{\in} -\alpha\gamma\tilde{x}_1^\top\tilde{x}_1 - \alpha\gamma\nu^\top\nu - \alpha\gamma e_1^\top e_1 - k_r r^\top r + \dot{P} + \xi^\top\left(\widehat{\Phi}'\tilde{\theta} - \beta_1 K\left[\text{sgn}\right](e_{\text{es}}) + e_1 + N_1\right) \\ - k_r\xi^\top\xi - \gamma\eta^\top\alpha\eta - \alpha\tilde{\theta}^\top\widehat{\Phi}'^\top(e_{\text{es}} + e_{\text{tr}}) + r^\top\left(N_1 + \widehat{\Phi}'\tilde{\theta} - \beta_2 K\left[\text{sgn}\right](e_{\text{tr}}) + \tilde{x}_1\right). \tag{4–20}$$

To facilitate the subsequent analysis, let the sets $\mathcal{S} \subset \mathbb{R}^\psi$ and $\mathcal{D} \subset \mathbb{R}^\psi$ be defined as $\mathcal{S} \triangleq \left\{\sigma \in \mathcal{D} : \|\sigma\| \leq \sqrt{\frac{\lambda_1}{\lambda_2}}\omega\right\}$ and $\mathcal{D} \triangleq \left\{\sigma \in \mathbb{R}^\psi : \|\sigma\| < \omega\right\}$, respectively, for some bounding constant $\omega \in \mathbb{R}_{>0}$. The following theorem establishes asymptotic tracking and estimation error convergence for the developed adaptive DRNN observer and overall output feedback control design.

Theorem 4.1. *Consider the system in* (4–1) *and let Assumptions 3.3 -3.4 and 4.1 hold. The adaptive Lb-DRNN observer in* (4–10)*, controller in* (4–15)*, and Lb-DRNN weight adaptation law in* (4–17) *ensure asymptotic estimation and tracking error convergence in the sense that* $\|e_{tr}\| \to 0$*,* $\|e_{es}\| \to 0$*,* $\|x_2 - \hat{x}_2\| \to 0$*, and* $\|x_2 - \dot{x}_{d,1}\| \to 0$ as $t \to \infty$*, provided* $\zeta(0) \in \mathcal{S}$ *and the following sufficient gain conditions are satisfied*

$$\beta_1, \beta_2 > \xi_1 + \xi_2 + \frac{1}{\alpha - \lambda_P}(\alpha\xi_2 + \xi_3), \\ \alpha \geq \frac{1}{2\gamma k_r}. \tag{4–21}$$

Proof. Consider the candidate Lyapunov function in (4–18). Based on the stability development, the P-function is designed as

$$P(t) \triangleq \mathrm{e}^{-\lambda_P t} * \left(\alpha(e_{\text{es}} + e_{\text{tr}})^\top N_2 + (\alpha - \lambda_P)(\beta_1\|e_{\text{es}}\|_1 + \beta_2\|e_{\text{tr}}\|_1)\right.$$

$$- (\alpha - \lambda_P) (e_{\text{es}} + e_{\text{tr}})^{\top} (N_1 + N_2) + (e_{\text{es}} + e_{\text{tr}})^{\top} \left(\dot{N}_1 + \dot{N}_2 \right) \Big)$$
$$+ \beta_1 \|e_{\text{es}}\|_1 + \beta_2 \|e_{\text{tr}}\|_1 - (e_{\text{es}} + e_{\text{tr}})^{\top} (N_1 + N_2), \tag{4–22}$$

where $\lambda_P \in \mathbb{R}_{>0}$ denotes a constant and $N_2 \in \mathbb{R}^n$ denotes an auxiliary term defined as $N_2 \triangleq \widehat{\Phi}' \tilde{\theta}$. Using the fact that the ideal weights are assumed to be bounded and the continuous differentiability of the auxiliary terms N_1 and N_2, N_1 and N_2 and the time-derivative of $N_1 + N_2$ can be bounded as $\|N_1\| \leq \xi_1$, $\|N_2\| \leq \xi_2$, and $\left\| \dot{N}_1 + \dot{N}_2 \right\| \leq \xi_3$, respectively, when $\zeta \in \mathcal{D}$, for known constants $\xi_1,\ \xi_2,\ \xi_3 \in \mathbb{R}_{>0}$. Therefore, using similar arguments as in [124, Lemma 4] and Chapter 3, it can be shown that P remains positive for all time $t \in \mathbb{R}_{\geq 0}$ provided the sufficient gain conditions in (4–21) are satisfied. The P-function in (4–22) is a Filippov solution to

$$\dot{P} \in -\lambda_P P - \xi^{\top} (N_1 - \beta_1 K [\text{sgn}] (e_{\text{es}})) - r^{\top} (N_1 - \beta_2 K [\text{sgn}] (e_{\text{tr}})) - (\dot{e}_{es} + \dot{e}_{tr})^{\top} N_2. \tag{4–23}$$

The terms $t \to \xi^{\top} K [\text{sgn}] (e_{\text{es}})$ and $t \to r^{\top} K [\text{sgn}] (e_{\text{tr}})$ are set-valued only for the set of time instants $T_1 = \{t \in [0, \infty) | \exists i \in \{1, 2, ..., n\}$ s.t. $e_{\text{es},i} (t) = 0 \wedge \xi_i (t) \neq 0\}$ and $T_2 = \{t \in [0, \infty) | \exists i \in \{1, 2, ..., n\}$ s.t. $e_{\text{tr},i} (t) = 0 \wedge r_i (t) \neq 0\}$, respectively. According to [124, Lemma 1], the sets T_1 and T_2 have Lebesgue measure zero. Using this, (4–8), (4–20), (4–23), the definition of N_2, and canceling like terms yields

$$\dot{\mathcal{V}}_L \overset{a.a.t}{\in} -\alpha\gamma \tilde{x}_1^{\top} \tilde{x}_1 - \alpha\gamma \nu^{\top} \nu - \alpha\gamma e_1^{\top} e_1 - \alpha\gamma \eta^{\top} \eta - k_r r^{\top} r - k_r \xi^{\top} \xi - \lambda_P P + \xi^{\top} e_1 + r^{\top} \tilde{x}_1, \tag{4–24}$$

when $\zeta \in \mathcal{D}$. Using Young's inequality, (4–24) can be further bounded as

$$\dot{\mathcal{V}}_L \leq -\lambda_3 \|z\|^2 - \lambda_P P,\ \forall \zeta \in \mathcal{D}, \tag{4–25}$$

where $\lambda_3 \triangleq \min \left\{ \alpha\gamma - \frac{1}{2k_r}, \alpha\gamma, \frac{k_r}{2} \right\}$. Therefore, provided the sufficient gain condition in (4–21) is satisfied, $\dot{\mathcal{V}}_L \leq 0, \forall \zeta \in \mathcal{D}$.

To show $\zeta(t) \in \mathcal{D}$ for all $t \geq 0$, the fact that $\mathcal{V}_L$ is nonincreasing, can be used to show $\|\zeta(t)\| \leq \sqrt{\frac{\mathcal{V}_L(t)}{\lambda_1}} \leq \sqrt{\frac{\mathcal{V}_L(0)}{\lambda_1}} \leq \sqrt{\frac{\lambda_2}{\lambda_1}}\|\zeta(0)\|$. Therefore, if $\|\zeta(0)\| \leq \sqrt{\frac{\lambda_1}{\lambda_2}}\omega$, then $\|\zeta\| < \omega$ for all $t \geq 0$. Thus, the states ζ should be initialized such that $\zeta(0) \in \mathcal{S}$ in order to guarantee that $\zeta(t) \in \mathcal{D}$ for all $t \in [0, \infty)$. To show $x \in \mathcal{X}$ so that the universal function approximation property holds, let the set $\Upsilon \subseteq \mathcal{X}$ be defined as $\Upsilon \triangleq \left\{\varsigma \in \mathcal{X} : \|\varsigma\| < \overline{x_d} + \overline{\dot{x}_d} + (3 + 2\alpha + \alpha^2)\omega\right\}$. Thus, if $\|\zeta(0)\| \in \mathcal{S}$, then $\|\zeta(t)\| \leq \omega$, and therefore $\|e_1(t)\| \leq \omega$, $\|\eta(t)\| \leq \omega$, and $\|r(t)\| \leq \omega$. Hence, using (4–5) and (4–6), $\|x\|$ can be bounded as $\|x\| \leq \overline{x_d} + \overline{\dot{x}_d} + (3 + 2\alpha + \alpha^2)\omega$. Therefore, if $\zeta(0) \in \mathcal{S}$, then $x \in \Upsilon \subseteq \mathcal{X}$ and thus the universal function approximation property holds.

Since $\zeta \in \mathcal{L}_\infty$, $x_1, \hat{x}_1, e_{\text{tr}}, e_{\text{es}} \in \mathcal{L}_\infty$. Since $e_{\text{tr}}, e_{\text{es}}, \widehat{\Phi}' \in \mathcal{L}_\infty$ and Φ is a smooth function, $\dot{\hat{\theta}} \in \mathcal{L}_\infty$. Using (4–18), (4–19), and (4–25), the LaSalle-Yoshizawa corollary in [125, Corollary 1] can be invoked to show $\|\tilde{x}_1\| \to 0$, $\|e_1\| \to 0$, $\|\nu\| \to 0$, $\|\eta\| \to 0$, $\|\xi\| \to 0$, and $\|r\| \to 0$ as $t \to \infty$. From (4–5) and (4–6), it can be further shown that $\|e_{\text{tr}}\| \to 0$, $\|e_{\text{es}}\| \to 0$, $\|x_2 - \hat{x}_2\| \to 0$, and $\|x_2 - \dot{x}_{d,1}\| \to 0$ as $t \to \infty$. □

4.5 Simulations

To demonstrate the performance and efficacy of the developed adaptive DRNN-based OFB controller, comparative simulations were performed with a shallow RNN (henceforth referred to as "SRNN") [84] and a central difference observer (henceforth referred to as "CD") as baselines for comparison. The simulations were performed on an unmanned underwater vehicle (UUV) system modeled as [132]

$$\begin{aligned} \dot{x}_1 &= \dot{\eta} \\ \dot{x}_2 &= -\overline{M}^{-1}(\eta)\left(\overline{C}(\eta, \dot{\eta}, \nu)\dot{\eta} + \overline{D}(\eta, \nu)\dot{\eta}\right) + \overline{M}^{-1}(\eta)\tau_n, \end{aligned} \tag{4–26}$$

where $x_1 = \eta \in \mathbb{R}^6$ denotes a vector of position and orientation with coordinates in the earth-fixed frame, $x_2 = \dot{\eta} \in \mathbb{R}^6$ denotes a vector of linear and angular velocities with coordinates in the earth-fixed frame, and $\nu \in \mathbb{R}^6$ denotes a vector of linear and angular

velocities with coordinates in the body-fixed frame. The inertial effects, centripetal-Coriolis effects, hydrodynamic damping effects, and control input in the earth-fixed frame can be represented by $\overline{M} : \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$, $\overline{C} : \mathbb{R}^6 \times \mathbb{R}^6 \times \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$, $\overline{D} : \mathbb{R}^6 \times \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$, and $\tau_n : \mathbb{R}_{\geq 0} \to \mathbb{R}^6$, respectively. The velocities in the body-fixed frame can be related to the velocities in the earth-fixed frame using the relation

$$\dot{\eta} = J(\eta)\,\nu, \tag{4–27}$$

where $J : \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$ is a Jacobian transformation matrix relating the two frames [132, Equation (2)]. Using the kinematic transformation in (4–27), the earth–fixed dynamics in (4–26) can be expressed using body-fixed dynamics as $\overline{M} = J^{-\top} M J^{-1}$, $\overline{C} = J^{-\top}\left[C(\nu) - MJ^{-1}\dot{J}\right]J^{-1}$, $\overline{D} = J^{-\top} D(\nu) J^{-1}$, and $\tau_n = J^{-\top}\tau_b$, where $M \in \mathbb{R}^{6\times 6}$, $C : \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$, $D : \mathbb{R}^6 \to \mathbb{R}^{6\times 6}$, and $\tau_b : \mathbb{R}_{\geq 0} \to \mathbb{R}^6$ denote the inertial effects, centripetal-Coriolis effects, hydrodynamic damping effects, and control input in the body-fixed frame, respectively. The inertial effects, centripetal-Coriolis effects, and hydrodynamic damping effects in the body-fixed effects can be expressed as [133, Equation (2.246)]

$$M = \text{diag}\{m_1, m_2, m_3, m_4, m_5, m_6\}$$

$$D = \text{diag}\{d_{11} + d_{12}|\nu(1)|, d_{21} + d_{22}|\nu(2)|, d_{31} + d_{32}|\nu(3)|,$$
$$d_{41} + d_{42}|\nu(4)|, d_{51} + d_{52}|\nu(5)|, d_{61} + d_{62}|\nu(6)|\}$$

$$V_m = \begin{bmatrix} 0 & 0 & 0 & 0 & m_3\nu_3 & -m_2\nu_2 \\ 0 & 0 & 0 & -m_3\nu_3 & 0 & m_1\nu_1 \\ 0 & 0 & 0 & m_2\nu_2 & -m_1\nu_1 & 0 \\ 0 & m_3\nu_3 & -m_2\nu_2 & 0 & m_6\nu_6 & -m_5\nu_5 \\ -m_3\nu_3 & 0 & m_1\nu_1 & -m_6\nu_6 & 0 & m_4\nu_4 \\ m_2\nu_2 & -m_1\nu_1 & 0 & m_5\nu_5 & -m_4\nu_4 & 0 \end{bmatrix},$$

where the following numerical values of mass, inertia, and damping parameters were used (Table 4-1).

Table 4-1. UUV System Parameters [133, Equation (2.247)]

$m_1 = 215$ kg	$d_{11} = 70$ Nm·sec	$d_{41} = 30$ Nm·sec
$m_2 = 265$ kg	$d_{12} = 100$ N·sec^2	$d_{42} = 50$ N·sec^2
$m_3 = 265$ kg	$d_{21} = 100$ Nm·sec	$d_{51} = 50$ Nm·sec
$m_4 = 40$ kg·m^2	$d_{22} = 200$ N·sec^2	$d_{52} = 100$ N·sec^2
$m_5 = 80$ kg·m^2	$d_{31} = 200$ Nm·sec	$d_{61} = 50$ Nm·sec
$m_6 = 80$ kg·m^2	$d_{32} = 50$ N·sec^2	$d_{62} = 100$ N·sec^2.

The desired trajectory was selected as a helical trajectory defined as $x_1 = [5\cos(0.1t)\text{ m}, 5\sin(0.1t)\text{ m}, 0.1t\text{ m}, 0\text{ rad}, 0\text{ rad}, -0.05t\text{ rad}]^\top$. The control gains were selected as $b = 1,\ k_r = 2,\ \alpha = 5,\ \beta_1 = 0.001,\ \beta_2 = 0.001$, and $\Gamma = 0.5 \cdot I_{592}$. The simulation was run for 150 seconds with a step size of 0.001 seconds and initial conditions $x_1 = [4\text{ m}, 0.5\text{ m}, 0\text{ m}, 0\text{ rad}, 0.2\text{ rad}, 0\text{ rad}]^\top$ and $x_2 = [0\text{ m}, 0\text{ m}, 0\text{ m}, 0\text{ rad}, 0\text{ rad}, 0\text{ rad}]^\top$.

The DRNN was composed of $k = 8$ layers with 8 neurons in each layer and hyperbolic tangent activation functions. The SRNN baseline used the same controller and observer design, but with $k = 2$ layers and $L = 17$ neurons to ensure that approximately the same number of individual weights was used for both RNNs. The second comparison simulation used a central difference observer with the controller in (4–15) without the feedforward DNN, i.e., $u = g(x_1)^+ \left[\ddot{x}_{d,1} - (k_r + \alpha)\left(\dot{\hat{e}}_1 + \alpha\hat{e}_1\right) - \alpha^2 e_1\right]$. For a fair comparison, the same robust control gains were used for all three simulations. To better emulate real-world systems, noise was added to the position measurements from a uniform distribution of $U(-0.001, 0.001)$, and the control input was saturated at 200 N or 200 N/m. The RNN weight estimates were randomly initialized from a uniform distribution of $U(-0.5, 0.5)$.

The performance results of the three simulations are shown in Table 4-2 and Figures 4-2-4-4. As shown in Figure 4-2, the RNN-based OFB controllers yield significant improvements in the estimation error performance when compared to the central difference observer. Specifically, the DRNN-based OFB controller yielded a 99.72% and 99.84% improvement in linear and angular estimation errors, respectively, when compared to the CD observer. The CD observer was significantly less robust to

Table 4-2. Simulation Performance Results

Architecture		$\|e_1\|$	$\|\tilde{x}_2\|$	$\|\tau_b\|$
SRNN	linear	0.1215 [m]	0.0103 [m/s]	92.19 [N]
	angular	0.0815 [rad]	0.0081 [rad/s]	12.72 [N/m]
CD	linear	0.1115 [m]	1.7339 [m/s]	301.37 [N]
	angular	0.0252 [rad]	1.7387 [rad/s]	336.43 [N/m]
DRNN	linear	0.0875 [m]	0.0049 [m/s]	91.84 [N]
	angular	0.0082 [rad]	0.0027 [rad/s]	12.68 [N/m]

measurement noise and therefore yielded chattering, oscillatory behavior in the estimation error which degraded steady state performance. This oscillatory behavior in the estimation errors led to similar behavior in the control inputs in Figure 4-4, which saturate frequently throughout the simulation, and also created chattering behavior in the tracking error (as shown in Figure 4-3). Thus, the DRNN-based observer design was significantly more robust to measurement noise.

While the SRNN-based OFB controller yielded noticeably better estimation error performance when compared to the CD observer, the DRNN-based OFB controller improved the normalized estimation error by 52.43% and 66.67% for the linear and angular estimation errors, respectively. Moreover, the DRNN-based OFB controller yielded significant improvements in the tracking error performance. As shown in Figure 4-3, both OFB controllers yielded similar transient tracking error performance and settled after approximately 5 seconds. However, the DRNN architecture significantly improved steady state behavior and the tracking error for the DRNN converged to a considerably smaller value than that of the SRNN. Specifically, the DRNN architecture improved the linear and angular tracking error by 27.98% and 89.94%, respectively, when compared to the shallow architecture with a comparable control effort.

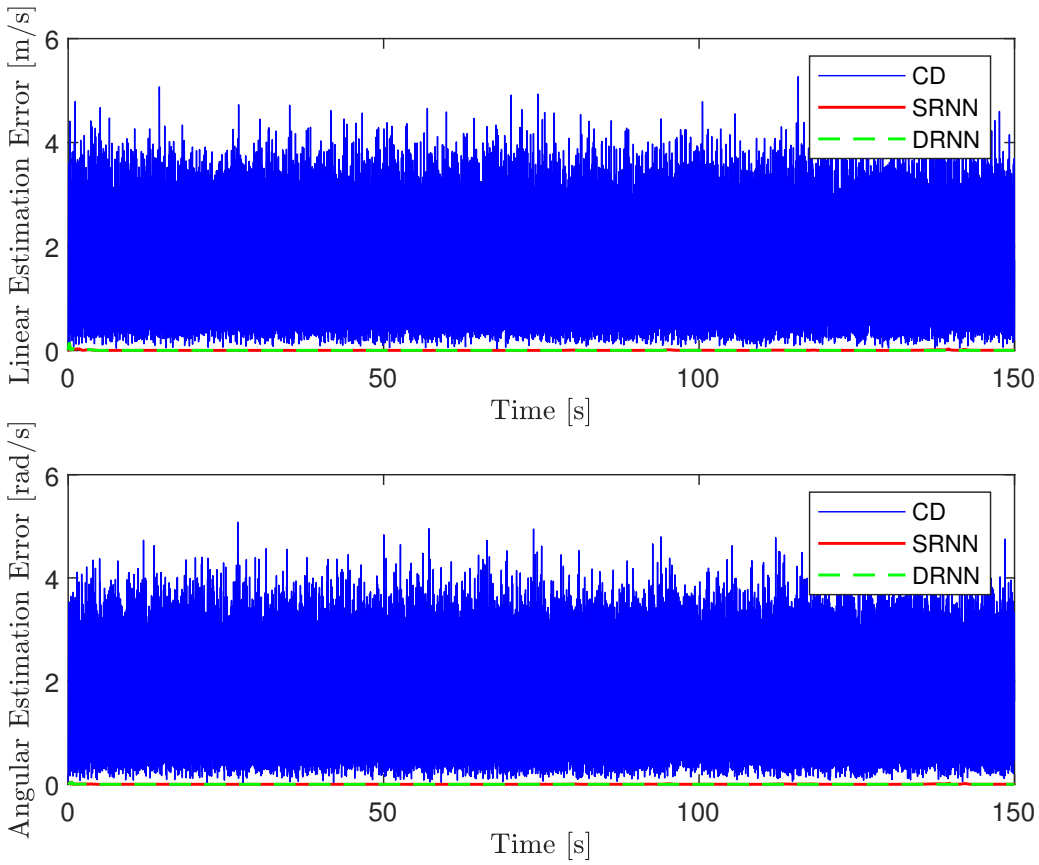

Figure 4-2. Plots of the norm of the linear (top) and angular (bottom) estimation errors over time of the DRNN OFB controller compared to the SRNN OFB controller and CD observer.

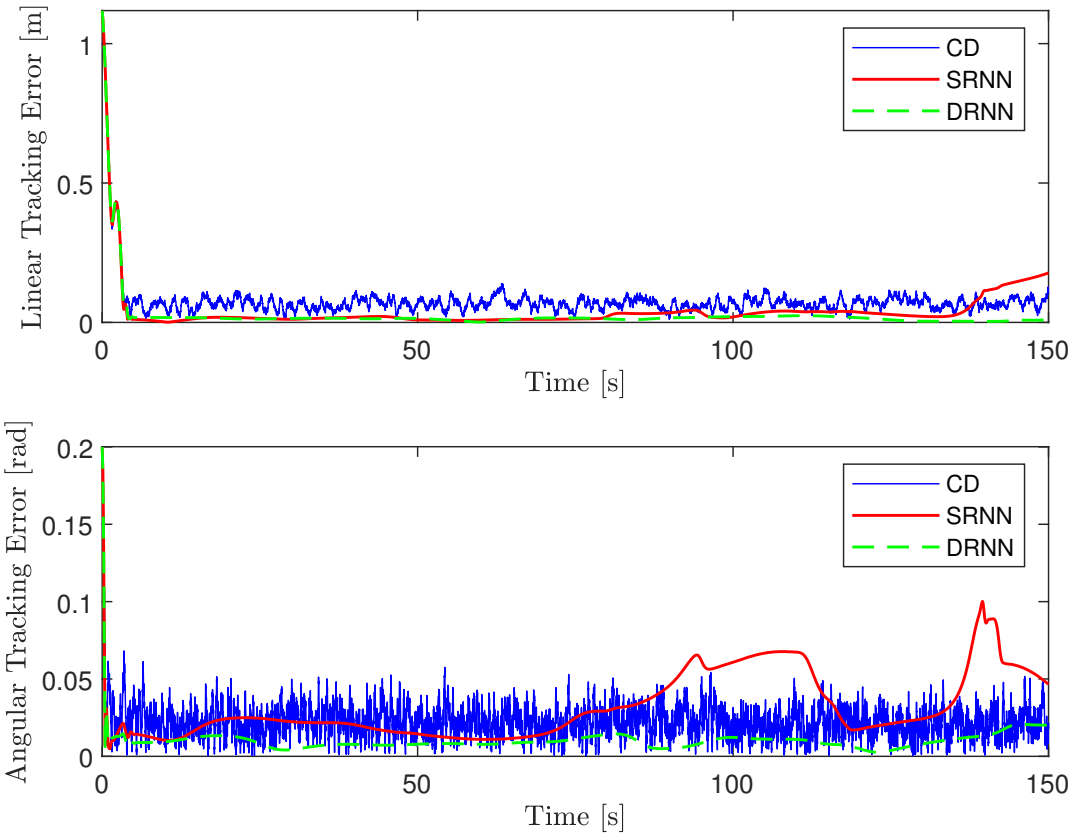

Figure 4-3. Plots of the norm of the linear (top) and angular (bottom) tracking errors over time of the DRNN OFB controller compared to the SRNN OFB controller and CD observer.

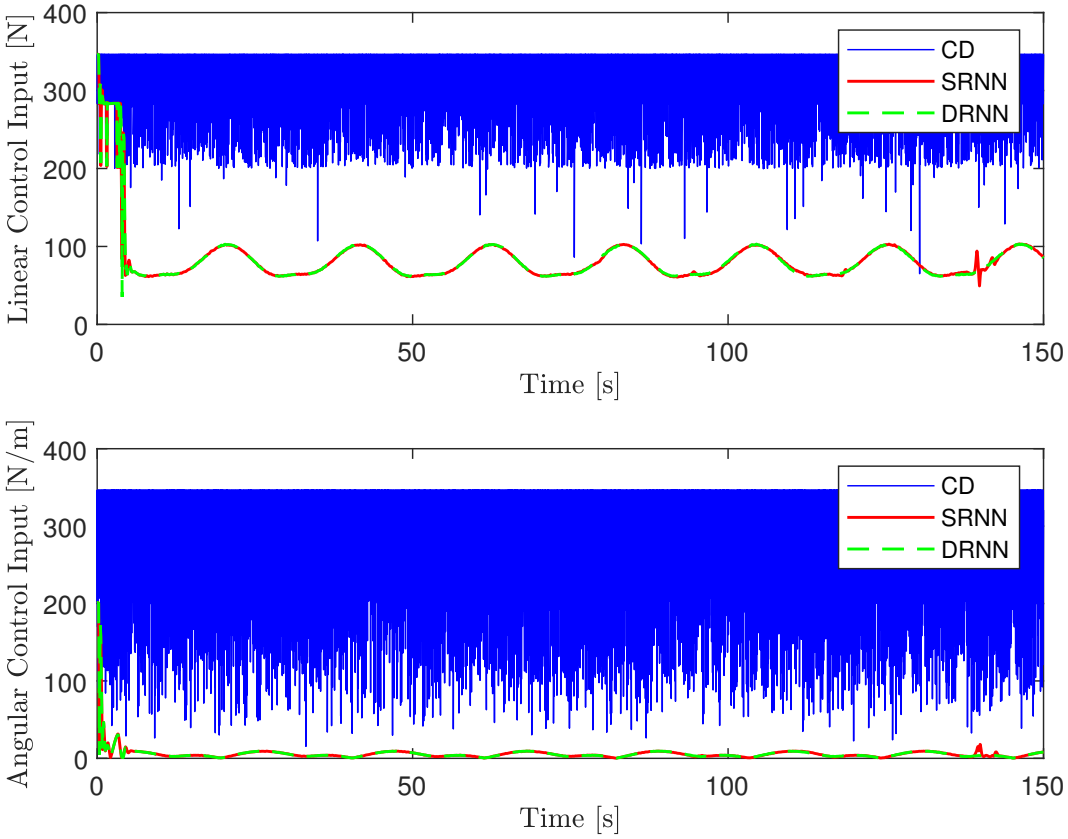

Figure 4-4. Plots of the norm of the linear (top) and angular (bottom) control input over time of the DRNN OFB controller compared to the SRNN OFB controller and CD observer.

4.6 Conclusions

An adaptive Lb-DRNN-based output feedback controller is developed for a class of uncertain nonlinear systems. Motivated by the dynamic nature and internal memory structure of RNNs, an Lb-DRNN observer is developed to adaptively estimate the unknown states of the system, which are implemented in the developed control design. However, the developed OFB controller is designed using the traditional DRNN architecture. Future work could integrate the adaptive LSTM-based control development in Chapter 2 and the adaptive LSTM-based observer development in Chapter 3 together into as Lb-LSTM OFB control architecture.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

DNN-based control methods are becoming more popular due to the improved performance of DNNs over traditional shallow NNs. However, previous analytic, stability-driven, adaptive DNN-based control methods are restricted to feedforward NNs. Thus, Chapter 2 develops an adaptive Lb-LSTM architecture and controller for general Euler-Lagrange systems, where the adaptation law is derived from Lyapunov-based methods. The novelty of Chapter 2 is the development of not only the first adaptive LSTM architecture, but also the first continuous-time representation of the LSTM cell. Specifically, a continuous-time Lb-LSTM NN is constructed and implemented in the controller as a feedforward term to adaptively estimate uncertain model dynamics. Despite the technical challenges posed by the complexity of the LSTM cell structure, stability-driven adaptation laws adjust the Lb-LSTM weights in real-time and allow the developed architecture to adapt to system uncertainties without any offline training requirements. To do so, Jacobians are computed of the LSTM cell dynamics with respect to the weights. Compared to typical offline LSTM methods (which can be used to provide an initial condition for the controller), the developed method provides online, continued learning using an analytical adaptation law, thereby providing significantly more robustness against changes in parameters or reference signal. Thus, the developed Lb-LSTM controller is able to learn the system dynamics in real-time and adapt to model uncertainties without any offline training requirements. A Lyapunov-based stability analysis is performed to guarantee UUB of the tracking errors and LSTM state and weight estimation errors.
To demonstrate the performance of the adaptive Lb-LSTM controller, simulations were performed and compared to the adaptive DNN-based controller in [104] using three different DNN architectures. The simulation results indicate significant improvements in tracking and function approximation performance when compared to various feedforward DNN architectures.

Since RNNs involve a hidden state, they implicitly involve state estimation, making them amenable for constructing state observers for uncertain nonlinear systems. Thus, in Chapter 3, the adaptive Lb-LSTM architecture developed in Chapter 2 is used to develop a novel LSTM-based observer to estimate unmeasurable states in a class of nonlinear systems. Since the unknown observer error is not available for online learning, a dynamic filter is designed to construct an auxiliary error that is implementable in the weight adaptation law. A nonsmooth Lyapunov-based stability analysis is performed that guarantees asymptotic convergence of the estimation errors and stability of the Lb-LSTM architecture and simulation results showed improved performance over existing methods. To validate the developed observer design, simulations were performed to estimate the angular velocity states of a two-link robot manipulator. The Lb-LSTM observer yielded a 41.13% improvement in the estimation error when compared to the adaptive shallow RNN observer in [1].

Chapter 4 leverages the adaptive control design in Chapter 2 and adaptive observer design in Chapter 3 to develop an OFB controller for uncertain nonlinear systems using adaptive DRNNs. Inspired by the dynamic nature and memory capabilities of RNNs, the first adaptive DRNN observer is designed to estimate the unknown states of the system and is incorporated into a control framework. Unlike the preceding chapters, the developed OFB controller is designed to achieve a two-fold control objective: asymptotic estimation of the unmeasurable states and asymptotic tracking control. The weights of the DRNN adjust online using Lyapunov-based stability-driven adaptation laws based on the tracking and observer errors. Through a Lyapunov-based stability analysis, the developed observer and the overall control design are proven to guarantee asymptotic stability, ensuring reliable and robust control performance. Validation simulation experiments on an unmanned underwater vehicle system yielded a 27.98% and 89.94% improvement in normalized linear and angular tracking error, respectively.

Potential future research could include extending the developed LSTM controller and observer design in Chapters 2 and 3 to develop an OFB controller, similar to the one developed in Chapter 4. Extending the OFB control design in Chapter 4 to adaptive LSTMs would bring additional challenges because the developed adaptation laws for the LSTM are more complex due to the more nonlinear structure of LSTMs compared to traditional DRNNs. Developing an adaptive LSTM-based OFB controller would involve incorporating the two-fold control objective of the OFB control design into the online training of the LSTM by integrating both the tracking and state estimation error into the weight adaptation laws. Since the estimation error is unknown, a dynamic filter could be used to construct an auxiliary error that would be implementable in the weight adaptation law.

Additional potential efforts include extending the developed Lyapunov-based network architecture design to adaptive physics-informed NN architectures or to other RNN architectures, such as GRUs. Like LSTMs, GRUs are a subarchitecture of RNNs that are designed with additional gate units to provide a more sophisticated memory than traditional RNNs. GRUs have one less gate unit than LSTMs and are therefore a simpler architecture. Therefore, extending the LSTM control and observer developments in Chapters 2 and 3 to develop an adaptive GRU architecture would result in a simpler NN architecture that is still capable of comparable function approximation performance. To do so, the developed weight adaptation laws would need to be rederived to consider the Jacobian of the GRU unit with respect to the weights and account for the lack of a cell state in the GRU architecture.

Transformers are a current state-of-the-art NN model that have outperformed traditional RNNs, LSTMs, and other NN models without requiring the computationally expensive recurrent loop present in RNNs. However, they have not been extensively investigated for controls applications, and there is no present adaptive control result for such an architecture. As shown in Figure 5-1, transformers rely on the self-attention

mechanism, which allows them to capture dependencies between inputs regardless of their positions in the time sequence. This mechanism enables transformers to model long-range dependencies effectively, making them well-suited for tasks involving sequential data. Transformers integrate past information without the use of a recurrent loop by using past input and output data as inputs to the transformer architecture. In real-time control applications, integrating past outputs as inputs to the transformer necessitates obtaining an implementable form of past outputs into the control design, which has not yet been done for transformers. Moreover, transformers are significantly more complex than LSTMs or other DRNN architectures. The more complex architecture complicates the development of stability-driven weight adaptation laws and motivates the incorporation of computer derived gradients into the adaptation laws to simplify the control architecture.

This dissertation provided the first adaptive LSTM architecture for both trajectory tracking and system identification. Moreover, this dissertation presents the first adaptive DRNN for OFB-based control. Simulation results empirically show that the advancements provided significant improvements when compared to state-of-the-art adaptive NN architectures. However, as described, there remains additional work to do.

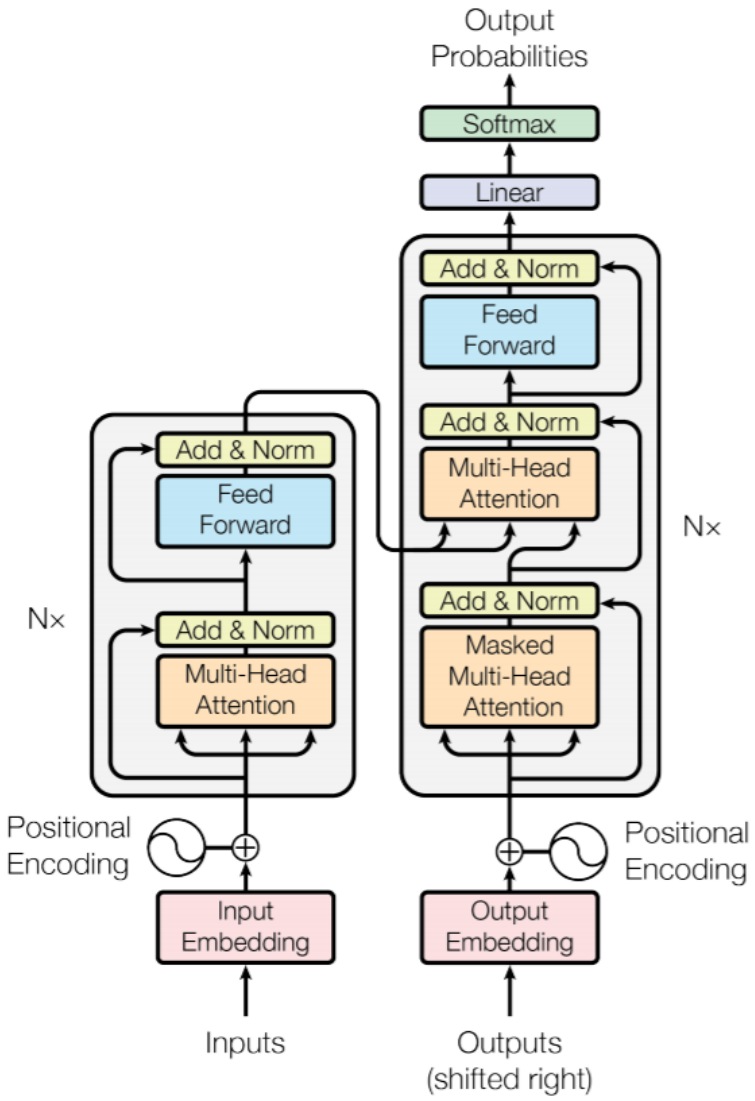

Figure 1: The Transformer - model architecture.

Figure 5-1. Transformer model [134].

REFERENCES

[1] H. T. Dinh, R. Kamalapurkar, S. Bhasin, and W. E. Dixon, "Dynamic neural network-based robust observers for uncertain nonlinear systems," *Neural Netw.*, vol. 60, pp. 44–52, Dec. 2014.

[2] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," in *Int. Conf. Learn. Represent.*, 2018.

[3] K. Funahashi and Y. Nakamura, "Approximation of dynamic systems by continuous-time recurrent neural networks," *Neural Netw.*, vol. 6, pp. 801–806, 1993.

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

[5] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[6] R. Sutton, A. Barto, and R. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Syst. Mag.*, vol. 12, no. 2, pp. 19–22, 1992.

[7] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[8] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, 2016.

[9] P. Deptula, Z. Bell, E. Doucette, W. J. Curtis, and W. E. Dixon, "Data-based reinforcement learning approximate optimal control for an uncertain nonlinear system with control effectiveness faults," *Automatica*, vol. 116, pp. 1–10, June 2020.

[10] 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.

[11] S. Bhasin, N. Sharma, P. Patre, and W. E. Dixon, "Asymptotic tracking by a reinforcement learning-based adaptive critic controller," *J. Control Theory Appl.*, vol. 9, no. 3, pp. 400–409, 2011.

[12] R. Kamalapurkar, J. R. Klotz, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for differential graphical games," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 423–433, 2018.

[13] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," in *Proc. IEEE Conf. Decis. Control*, (Los Angeles, CA), pp. 5083–5088, Dec. 2014.

[14] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 753–758, 2017.

[15] R. Kamalapurkar, H. T. Dinh, P. Walters, and W. E. Dixon, "Approximate optimal cooperative decentralized control for consensus in a topological network of agents with uncertain nonlinear dynamics," in *Proc. Am. Control Conf.*, (Washington, DC), pp. 1322–1327, Jun. 2013.

[16] R. Kamalapurkar, H. Dinh, S. Bhasin, and W. E. Dixon, "Approximate optimal trajectory tracking for continuous-time nonlinear systems," *Automatica*, vol. 51, pp. 40–48, Jan. 2015.

[17] R. Kamalapurkar, J. Klotz, and W. E. Dixon, "Model-based reinforcement learning for on-line feedback-Nash equilibrium solution of N-player nonzero-sum differential games," in *Proc. Am. Control Conf.*, pp. 3000–3005, 2014.

[18] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proc ACM/IEEE Int. Conf. Cyber Phys. Syst.*, pp. 237–248, ACM, 2019.

[19] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised learning of natural languages," *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, pp. 11629–11634, 2005.

[20] L. H. Smith, T. A. Kuiken, and L. J. Hargrove, "Evaluation of linear regression simultaneous myoelectric control using intramuscular emg," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 4, pp. 737–746, 2016.

[21] R. Kopp and R. Orford, "Linear regression applied to system identification for adaptive control systems," *AIAA J.*, vol. 1, no. 10, 1963.

[22] R.-C. Wu, Y.-W. Tseng, and C.-Y. Chen, "Estimating parameters of the induction machine by the polynomial regression," *Appl. Sci.*, vol. 8, no. 7, 2018.

[23] H. Li and S. Yamamoto, "Polynomial regression based model-free predictive control for nonlinear systems," in *Conf Society Instrument Control Engineers Japan*, pp. 578–582, 2016.

[24] H. Li and S. Yamamoto, "A model-free predictive control method based on polynomial regression," in *Int Symposium Control Sys*, 2016.

[25] J. Spooner and K. Passino, "Stable adaptive control using fuzzy systems and neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 3, pp. 339–359, 1996.

[26] D. Kim, H. T. Chung, S. Bhasin, and W. E. Dixon, "Robust composite adaptive fuzzy identification and control for a class of MIMO nonlinear systems," in *Proc. Am. Control Conf.*, (San Francisco, CA), pp. 4947–4952, 2011.

[27] V. Subramanian, T. F. Burks, and W. E. Dixon, "Sensor fusion using fuzzy logic enhanced kalman filter for autonomous vehicle guidance in citrus groves," *Trans. of the Am. Society of Agri. and Bio. Eng.*, vol. 52, no. 5, pp. 1411–1422, 2009.

[28] C. Tao, J. Taur, and M. Chan, "Adaptive fuzzy terminal sliding mode controller for linear systems with mismatched time-varying uncertainties," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 34, no. 1, pp. 255–262, 2004.

[29] M. Roopaei and M. Zolghadri, M. Sina, "Enhanced adaptive fuzzy sliding mode control for uncertain nonlinear systems," *Commun. Nonlinear Sci.*, vol. 14, no. 9–10, pp. 3670–3681, 2009.

[30] S. Tong, Y. Li, and P. Shi, "Fuzzy adaptive backstepping robust control for siso nonlinear system with dynamic uncertainties," *Inform. Sciences*, vol. 179, no. 9, pp. 1319–1332, 2009.

[31] H. Choi, "Adaptive controller design for uncertain fuzzy systems using variable structure control approach," *Automatica*, vol. 45, no. 11, pp. 2646–2650, 2009.

[32] K. Tanaka and M. Sugeno, "Stability analysis and design of fuzzy control-systems," *Fuzzy Set. Syst.*, vol. 45, no. 2, pp. 135–156, 1992.

[33] F. L. Lewis, "Nonlinear network structures for feedback control," *Asian J. Control*, vol. 1, no. 4, pp. 205–228, 1999.

[34] P. P. Angelov and D. P. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 34, no. 1, pp. 484–498, 2004.

[35] B.-T. Ann, J.-J. J. Chen, and M.-S. Ju, "Implementation of an adaptive fuzzy controller for FES-cycling system," in *Proc. 19th Int. Conf. of the IEEE EMBS*, vol. 3, pp. 1119–1120, Oct. 1997.

[36] A. Boulkroune, M. Tadjine, M. MSaad, and M. Farza, "How to design a fuzzy adaptive controller based on observers for uncertain affine nonlinear systems," *Fuzzy Set. Syst.*, vol. 159, pp. 926–948, 2008.

[37] A. Boulkroune, M. Tadjine, M. M'Saad, and M. Farza, "Fuzzy adaptive controller for mimo nonlinear systems with known and unknown control direction," *Fuzzy Set. Syst.*, vol. 161, no. 6, pp. 797–820, 2010.

[38] J.-J. J. Chen, N.-Y. Yu, D.-G. Huang, B.-T. Ann, and G.-C. Chang, "Applying fuzzy logic to control cycling movement induced by functional electrical stimulation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 5, pp. 158–169, June 1997.

[39] Y. Chen and C. Teng, “A model-reference control-structure using a fuzzy neural-network,” *Fuzzy Set. Syst.*, vol. 73, no. 3, pp. 291–312, 1995.

[40] B. Chen, X. Liu, and S. Tong, “Robust fuzzy control of nonlinear systems with input delay,” *Chaos, Solitons & Fractals*, vol. 37, no. 3, pp. 894–901, 2008.

[41] C.-S. Chiu and T.-S. Chiang, “Robust output regulation of t–s fuzzy systems with multiple time-varying state and input delays,” *IEEE Trans. Fuzzy Syst.*, vol. 17, pp. 962–975, Aug. 2009.

[42] A. Farhoud and A. Erfanian, “Fully automatic control of paraplegic FES pedaling using higher-order sliding mode and fuzzy logic control,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 3, pp. 533–542, 2014.

[43] H. Lee, J. Park, and G. Chen, “Robust fuzzy control of nonlinear systems with parametric uncertainties,” *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 2, pp. 369–379, 2001.

[44] Y. Hsu, G. Chen, S. Tong, and H. Li, “Integrated fuzzy modeling and adaptive control for nonlinear systems,” *Inform. Sciences*, vol. 153, no. 1, pp. 217–236, 2003.

[45] Q. Zhou, P. Shi, J. Lu, and S. Xu, “Adaptive output-feedback fuzzy tracking control for a class of nonlinear systems,” *IEEE Trans. Fuzzy Syst.*, vol. 19, pp. 972–982, Oct. 2011.

[46] B. Yoo and W. Ham, “Adaptive fuzzy sliding mode control of nonlinear systems,” *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 2, pp. 315–321, 1998.

[47] S.-C. Tong and N. Sheng, “Adaptive fuzzy observer backstepping control for a class of uncertain nonlinear systems with unknown time-delay,” *Int.. J. Autom. and Comput.*, vol. 7, no. 2, pp. 236–246, 2010.

[48] S. Tong, J. Tang, and T. Wang, “Fuzzy adaptive control of multivariable nonlinear systems,” *Fuzzy Set. Syst.*, vol. 111, no. 2, pp. 153–167, 2000.

[49] 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.

[50] F. L. Lewis, K. Liu, and A. Yesildirek, “Neural net robot controller with guaranteed tracking performance,” *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 703–715, 1995.

[51] F. L. Lewis, A. Yegildirek, and K. Liu, “Multilayer neural-net robot controller with guaranteed tracking performance,” *IEEE Trans. Neural Netw.*, vol. 7, pp. 388–399, Mar. 1996.

[52] S. S. Ge, C. C. Hang, and T. Zhang, “Adaptive neural network control of nonlinear systems by state and output feedback,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, pp. 818–828, December 1999.

[53] S. Ge, C. Hang, and T. Zhang, “Nonlinear adaptive control using neural networks and its application to CSTR systems,” *J. Process Control*, vol. 9, no. 4, pp. 313–323, 1999.

[54] A. Yesildirek and F. L. Lewis, “A neural net controller for robots with Hebbian tuning and guaranteed tracking,” in *Proc. Am. Control Conf.*, (Seattle, WA), pp. 2784–2789, 1995.

[55] H. Gomi and M. Kawato, “Neural network control for a closed-loop system using feedback-error-learning,” *Neural Networks*, vol. 6, no. 7, pp. 933–946, 1993.

[56] J. Noriega and H. Wang, “A direct adaptive neural-network control for unknown nonlinear systems and its application,” *IEEE Transactions on Neural Networks*, vol. 9, pp. 27–34, 1998.

[57] F. Chen and H. Khalil, “Adaptive control of a class of nonlinear discrete-time systemsusing neural networks,” *IEEE Trans. Autom. Control*, vol. 40, no. 5, pp. 791–801, 1995.

[58] W. He, Y. Chen, and Z. Yin, “Adaptive neural network control of an uncertain robot with full-state constraints,” *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 620–629, 2016.

[59] 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.

[60] H. D. Patino and D. Liu, “Neural network-based model reference adaptive control system,” *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 30, pp. 198–204, 2000.

[61] P. M. Patre, W. MacKunis, K. Kaiser, and W. E. Dixon, “Asymptotic tracking for uncertain dynamic systems via a multilayer neural network feedforward and RISE feedback control structure,” *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2180–2185, 2008.

[62] P. Patre, S. Bhasin, Z. Wilcox, and W. E. Dixon, “Composite adaptation for neural network-based controllers,” in *Proc. IEEE Conf. Decis. Control*, (Shanghai, China), pp. 6726–6731, Jan. 2009.

[63] S. A. Nivison and P. Khargonekar, “Improving long-term learning of model reference adaptive controllers for flight applications: A sparse neural network approach,” in *Proc. AIAA Guid. Navig. Control Conf.*, Jan. 2017.

[64] S. A. Nivison and P. Khargonekar, “A sparse neural network approach to model reference adaptive control with hypersonic flight applications,” in *Proc. AIAA Guid. Navig. Control Conf.*, p. 0842, 2018.

[65] R. Sanner and J. J. Slotine, “Gaussian networks for direct adaptive control,” *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 837–864, 1992.

[66] M. Polycarpou, “Stable adaptive neural control scheme for nonlinear systems,” *IEEE Trans. Autom. Control*, vol. 41, no. 3, pp. 447–451, 1996.

[67] G. Chowdhary and E. Johnson, “Recursively updated least squares based modification term for adaptive control,” in *Proc. Am. Control Conf.*, pp. 892–897, Oct. 2010.

[68] S. Lin and A. Goldenberg, “Neural-network control of mobile manipulators,” *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1121–1133, 2001.

[69] C. Yang, Z. Li, R. Cui, and B. Xu, “Neural network-based motion control of an underactuated wheeled inverted pendulum model,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 11, pp. 2004–2016, 2014.

[70] X. Li and C. C. Cheah, “Adaptive neural network control of robot based on a unified objective bound,” *IEEE Trans. Control Sys. Techn.*, vol. 22, no. 3, pp. 1032–1043, 2014.

[71] R. T. Anderson, G. Chowdhary, and E. N. Johnson, *Comparison of RBF and SHL Neural Network Based Adaptive Control*, pp. 183–199. Dordrecht: Springer Netherlands, 2009.

[72] H. Yang and J. Liu, “An adaptive rbf neural network control method for a class of nonlinear systems,” *IEEE J. Autom. Sin.*, vol. 5, pp. 457–462, 03 2018.

[73] Y. Li, W. Chen, J. Chen, X. Chen, J. Liang, and M. Du, “Neural network based modeling and control of elbow joint motion under functional electrical stimulation,” *Neurocomputing*, vol. 340, pp. 171–179, 2019.

[74] H. Y. Zhou, L. K. Huang, Y. M. Gao, Z. L. Vasic, M. Cifrek, and M. Du, “Estimating the ankle angle induced by fes via the neural network-based hammerstein model,” *IEEE Access*, vol. 7, pp. 141277–141286, 2019.

[75] N. Lan, H.-Q. Feng, and P. E. Crago, “Neural network generation of muscle stimulation patterns for control of arm movements,” *IEEE Trans. Rehabil. Eng.*, vol. 2, pp. 213–224, Dec. 1994.

[76] D. Graupe and H. Kordylewski, “Artificial neural network control of FES in paraplegics for patient responsive ambulation,” *IEEE Trans. Biomed. Eng.*, vol. 42, pp. 699–707, July 1995.

[77] J. P. Giuffrida and P. E. Crago, “Functional restoration of elbow extension after spinal-cord injury using a neural network-based synergistic FES controller,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 13, pp. 147–152, June 2005.

[78] G.-C. Chang, J.-J. Lub, G.-D. Liao, J.-S. Lai, C.-K. Cheng, B.-L. Kuo, and T.-S. Kuo, "A neuro-control system for the knee joint position control with quadriceps stimulation," *IEEE Trans. Rehabil. Eng.*, vol. 5, pp. 2–11, Mar. 1997.

[79] K. Kurosawa, R. Futami, T. Watanabe, and N. Hoshimiya, "Joint angle control by FES using a feedback error learning controller," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 13, pp. 359–371, Sept. 2005.

[80] A. Pedrocchi, S. Ferrante, E. De Momi, and G. Ferrigno, "Error mapping controller: a closed loop neuroprosthesis controlled by artificial neural networks," *J. Neuroeng. Rehabil.*, vol. 3, p. 25, Oct. 2006.

[81] J. J. Abbas and H. J. Chizeck, "Neural network control of functional neuromuscular stimulation systems: computer simulation studies," *IEEE Trans. Biomed. Eng.*, vol. 42, pp. 1117–1127, Nov. 1995.

[82] N. Sharma, C. M. Gregory, M. Johnson, and W. E. Dixon, "Closed-loop neural network-based nmes control for human limb tracking," *IEEE Trans. Control Syst. Tech.*, vol. 20, no. 3, pp. 712–725, 2012.

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

[84] H. Dinh, S. Bhasin, R. Kamalapurkar, and W. E. Dixon, "Dynamic neural network-based output feedback tracking control for uncertain nonlinear systems," *ASME J. Dyn. Syst. Meas. Control*, vol. 139, pp. 074502–1–074502–7, July 2017.

[85] E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, and P. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 422–431, 1995.

[86] Y. H. Kim, F. L. Lewis, and C. T. Abdallah, "A dynamic recurrent neural-network-based adaptive observer for a class of nonlinear systems," *Automatica*, vol. 33, pp. 1539–1543, 1997.

[87] Y. H. Kim and F. L. Lewis, "Neural network output feedback control of robot manipulators," *IEEE Trans. Robot. Autom.*, vol. 15, pp. 301–309, 1999.

[88] C. T. A. Young H. Kim, Frank L. Lewis, "A dynamic recurrent neural-network-based adaptive observer for a class of nonlinear systems," *Automatica*, vol. 33, pp. 1539–1543, 1997.

[89] F.-J. L. Rong-Jong Wai, "Adaptive recurrent-neural-network control for linear induction motor," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, 2001.

[90] J. Hanson, M. Raginsky, and E. Sontag, "Learning recurrent neural net models of nonlinear systems," 2020.

[91] J. Hanson and M. Raginsky, "Universal simulation of stable dynamical systems by recurrent neural nets," in *Learn. Dyn. Control*, 2020.

[92] E. D. Sontag, "A learning result for continuous-time recurrent neural networks," *IEEE Syst. Control Lett.*, vol. 34, 1998.

[93] N. Hovakimyan, F. Nardi, A. Calise, and N. Kim, "Adaptive output feedback control of uncertain nonlinear systems using single-hidden-layer neural networks," *IEEE Trans. Neural Netw.*, vol. 13, pp. 1420–1431, Nov. 2002.

[94] F. Abdollahi, H. Talebi, and R. Patel, "A stable neural network-based observer with application to flexible-joint manipulators," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 118–129, 2006.

[95] A. Nikolakopoulou, M. S. Hong, and R. D. Braatz, "Dynamic state feedback controller and observer design for dynamic artificial neural network models," *Automatica*, vol. 146, p. 110622, 2022.

[96] 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.

[97] A. Stroh and J. Desai, "Human-centered deep learning neural network trained myoelectric controller for a powered wheelchair," in *Int. Symp. Measur. Control Robot.*, pp. D2–4–1–D2–4–4, 2019.

[98] S. Edhah, S. Mohamed, A. Rehan, M. AlDhaheri, A. AlKhaja, and Y. Zweiri, "Deep learning based neural network controller for quad copter: Application to hovering mode," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–5, 2019.

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

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

[101] 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.

[102] 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.

[103] 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.

[104] 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.

[105] Z. Bell, R. Sun, K. Volle, P. Ganesh, S. Nivison, and W. E. Dixon, “Target tracking subject to intermittent measurements using attention deep neural networks,” *IEEE Control Syst. Lett.*, vol. 7, pp. 379–384, 2023.

[106] A. Graves, G. Wayne, and I. Danihelka, “Neural Turing machines,” *arXiv*, 2014.

[107] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in *Proc. Int. Conf. Mach. Learn.*, ICML’16, p. 1842â1850, JMLR.org, 2016.

[108] E. Parisotto and R. Salakhutdinov, “Neural map: Structured memory for deep reinforcement learning,” in *Proc. Int. Conf. Learn. Represent.*, 2018.

[109] D. Muthirayan and P. P. Khargonekar, “Memory augmented neural network adaptive controllers: Performance and stability,” *IEEE Trans. Autom. Contr*, vol. 68, no. 2, pp. 825–838, 2023.

[110] R. Patel, M. Zeinali, and K. Passi, “Deep learning-based robot control using recurrent neural networks (lstm; gru) and adaptive sliding mode control,” in *Int. Conf. Control Dyn. Sys. Robot.*, 05 2021.

[111] Q. Teng and L. Zhanga, “Data driven nonlinear dynamical systems identication using multi-step CLDNN,” *AIP Advances*, 2019.

[112] J. Tembhurne and T. Diwan, “Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks,” *Multimedia Tools and Applications*, vol. 80, pp. 1–40, 02 2021.

[113] D. S. Bernstein, *Matrix Mathematics*. Princeton university press, 2009.

[114] 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, pp. 73–82, Jan. 1987.

[115] 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.

[116] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

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

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

[119] M. de Queiroz, J. Hu, D. Dawson, T. Burg, and S. Donepudi, "Adaptive position/force control of robot manipulators without velocity measurements: Theory and experimentation," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 27-B, no. 5, pp. 796–809, 1997.

[120] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 3 ed., 2002.

[121] E. Griffis, O. Patil, W. Makumi, and W. E. Dixon, "Deep recurrent neural network-based observer for uncertain nonlinear systems," in *IFAC World Congr.*, 2023.

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

[123] E. Griffis, O. Patil, Z. Bell, and W. E. Dixon, "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based control," *IEEE Control Syst. Lett.*, vol. 7, pp. 2976–2981, 2023.

[124] O. Patil, A. Isaly, B. Xian, and W. E. Dixon, "Exponential stability with RISE controllers," *IEEE Control Syst. Lett.*, vol. 6, pp. 1592–1597, 2022.

[125] N. Fischer, R. Kamalapurkar, and W. E. Dixon, "LaSalle-Yoshizawa corollaries for nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 58, pp. 2333–2338, Sep. 2013.

[126] K. C. Razvan Pascanu, Caglar Gulcehre, "How to construct deep recurrent neural networks," *Int Conf Learn Representations*, 2014.

[127] H. T. Dinh, S. Bhasin, D. Kim, and W. E. Dixon, "Dynamic neural network-based global output feedback tracking control for uncertain second-order nonlinear systems," in *American Control Conference*, (Montréal, Canada), pp. 6418–6423, Jun. 2012.

[128] Luenberger, "Observing the state of a linear system," *IEEE Trans. Mil. Electron.*, vol. 8, pp. 74–80, 1964.

[129] F. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural net robot controller: structure and stability proofs," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, 1996.

[130] A. F. Filippov, *Differential equations with discontinuous right-hand side*. Kluwer Academic Publishers, 1988.

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

[132] N. Fischer, D. Hughes, P. Walters, E. Schwartz, and W. E. Dixon, "Nonlinear RISE-based control of an autonomous underwater vehicle," *IEEE Trans. Robot.*, vol. 30, pp. 845–852, Aug. 2014.

[133] W. E. Dixon, A. Behal, D. M. Dawson, and S. Nagarkatti, *Nonlinear Control of Engineering Systems: A Lyapunov-Based Approach*. Birkhauser: Boston, 2003.

[134] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Adv. Neural Inf. Process Syst.* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, 2017.

BIOGRAPHICAL SKETCH

Emily Jean Griffis was born in Gainesville, Florida in 1998. She got her love of engineering from her dad. She received her Bachelor of Science (B.S.) degree from the Department of Mechanical and Aerospace Engineering at the University of Florida in May 2020. In Fall 2020, Emily joined the Nonlinear Controls and Robotics Laboratory at the University of Florida under the supervision of Dr. Warren Dixon to pursue her Ph.D. She received her Master of Science (M.S.) degree in mechanical engineering in December 2021. Emily's research focuses on using adaptive control and deep learning to study Lyapunov-based control of nonlinear and uncertain dynamical systems.