UNIVERSITY OF CALIFORNIA

Los Angeles

Statistical Machine Learning-Based Predictive Control of Nonlinear Processes

A dissertation submitted in partial satisfaction of the

requirement for the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Aisha Alnajdi

2024

ABSTRACT OF THE DISSERTATION

Statistical Machine Learning-Based Predictive Control of Nonlinear Processes

by

Aisha Alnajdi

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2024

Professor Panagiotis D. Christofides, Chair

Data are an essential factor in the fourth industrial revolution, demanding engineers and scientists to leverage and analyze their potential for significantly improving the efficiency of industrial processes and their control systems. In classical industrial process control systems, the models are constructed using linear data-driven approaches, where parameters are adjusted based on experimental or simulated data. In certain critical control loops focused on optimizing profits, first-principles models are used to describe the fundamental physicochemical phenomena, incorporating a small set of parameters derived from industrial or simulation data. However, despite the effectiveness of these classical modeling methods in many studies, there persists a significant challenge when modeling large-scale, complex non-

linear systems within the field of process engineering. Traditional approaches often fall short of accurately representing the complexities and nonlinear dynamics inherent in large-scale industrial processes. Therefore, there are continuous efforts to conduct extensive studies on effective tools for model development and evaluation techniques. This is crucial because process models play a central role in advanced control strategies, particularly, model-based control systems such as model predictive control (MPC) and economic MPC (EMPC) frameworks. Therefore, accurate construction and evaluation of these models will contribute to achieving the desired performance and ensuring operational efficiency, ultimately leading to robust and reliable control systems.

Machine learning techniques have proven to be an effective modeling tool in many engineering applications. More specifically, machine learning models have been used to model large-scale, complex nonlinear systems. These models are then integrated into MPC to achieve closed-loop stability. Among the many types of machine learning techniques, recurrent neural networks (RNNs) are widely used to model nonlinear processes involving time series data. This is due to their special structure, which allows useful previous information to be retained.

In addition to complexities arising from nonlinearities and the large-scale nature of practical industrial processes, and challenges in modeling these systems, time delays pose significant challenges in nonlinear control systems. These delays can arise due to various sources such as transportation lags, sensor and actuator response times. Such delays can lead to instability, oscillations, and overall degradation in the performance of the control system. Hence, addressing these delays is crucial for maintaining the system's stability

and optimizing its performance. Besides time-delay systems, there are also systems that experience different time-scale multiplicity, known as two-time scale systems. These types of systems require specific techniques to handle and design efficient model-based controllers to achieve closed-loop stability. Additionally, this dissertation includes an assessment of generalization error bounds for different types of machine learning models to evaluate their performance and reliability in various scenarios.

In response to the factors highlighted, this dissertation presents the integration of machine learning techniques with model predictive control to stabilize the dynamics of nonlinear chemical processes. The dissertation begins with a comprehensive overview of its motivation, background, and structure. Then, it discusses the use of machine learning models within a model predictive control framework to stabilize a nonlinear system with time-delays. Additionally, the design of a machine learning-based predictor to compensate the effect of inputs delays is discussed. The closed-loop stability of the system achieved with Lyapunov-based model predictive controllers is investigated through theoretical analysis. Subsequently, a theoretical framework for deriving generalization error bounds for RNNs, partially connected recurrent RNNs, and long short-term memory (LSTM) RNNs are introduced. Next, we study generalization error bounds for models capturing the dynamics of two-time-scale systems and present simulation studies to address the modeling criteria of these systems under MPC frameworks, along with the necessary assumptions to achieve closed-loop stability. Throughout the dissertation, control methods are validated through their application in numerical simulations of nonlinear chemical processes, highlighting their effectiveness, performance and reliability.

The dissertation of Aisha Alnajdi is approved.

Dante Simonetti

Jason Speyer

Lieven Vandenberghe

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2024

Contents

List of Figures

List of Tables

Acknowledgements

All praise is due to Allah, by whose favor good deeds are accomplished.

I would like to deliver my profound gratitude to my advisor, Professor Panagiotis D. Christofides, for his support, encouragement, and guidance throughout my academic journey. Professor Christofides exemplifies excellence as a researcher, mentor, instructor, and role model. I consider myself fortunate to be his student, as this Ph.D. experience has laid a strong foundation for my future career. I would also like to extend my sincere gratitude to my doctoral committee: Professor Dante Simonetti, Professor Lieven Vandenberghe, and Professor Jason Speyer for their time, insightful comments, and valuable advice.

I am also grateful for my family and friends for their support and encouragement. In particular, I am deeply grateful to my dear husband, Abdullah, for his prayers, love, and care, as well as the steadfast support and motivation throughout this journey. Additionally, I wish to extend my deepest gratitude to my wonderful parents, Maali and Musaad, for their prayers, unwavering support, encouragement, and unconditional love. My heartfelt thanks also go to Sultan, Asmaa, Afnan, and Anas for their constant belief in me and their encouragement to help me achieve my best. I would also like to extend my thanks to my aunt Kholoud and my friend Maab for their continuous motivation and support.

In addition, I would like to thank all of my colleagues in the Christofides research group, including: Professor Zhe Wu, Dr. Scarlett Chen, Dr. Fahim Abdullah, Professor Mohammed Alhajeri, Dr. Yi Ming Ren, Dr. Matthew Tom, Dr. Sungil Yun, Dr. Junwei Luo, Dr. Berkay Citmaci, Henrik Wang, Feiyang Ou, Dominic Peters, Atharva Suryavanshi, Yash Kadakia,

Xiaodong Cui, Yifei Wang, Parth Jagdish Chheda, Arthur Khodaverdian, Julius Suherman, and Dhruv Gohil.

Additionally, financial support from Kuwait University is gratefully acknowledged. Their contribution has been instrumental in the completion of this dissertation, and I am deeply appreciative of their support.

Curriculum Vitae

Education

Kuwait University — September 2013 - June 2018
B. Sc., Electrical Engineering — **Al-Asimah, Kuwait**

University of California, Los Angeles — September 2019 - March 2021
M. Sc., Electrical and Computer Engineering — **Los Angeles, California**

Expereince

Ministry of Education — March 2019 - June 2019
Electrical engineer — **Hawalli, Kuwait**

Publications

1. Wu, Z., **A. Alnajdi**, Q. Gu, and P. D. Christofides, "Statistical machine-learning–based predictive control of uncertain nonlinear processes," AIChE J., **68**, e17642, 2022.

2. Kadakia, Y., **A. Alnajdi**, F. Abdullah, and P. D. Christofides, "Encrypted Distributed Model Predictive Control with State Estimation for Nonlinear Processes," *Dig. Chem. Eng.*, **9**, 100133, 2023.

3. Kadakia, Y., **A. Alnajdi**, F. Abdullah, and P. D. Christofides, "Encrypted Decentralized Model Predictive Control of Nonlinear Processes with Delays," *Chem. Eng. Res. & Des.*, **200**, 312–324, 2023.

4. Kadakia, Y., A. Suryavanshi, **A. Alnajdi**, F. Abdullah, and P. D. Christofides, "Integrating Machine Learning Detection and Encrypted Control for Enhanced Cybersecurity of Nonlinear Processes," *Comp. & Chem. Eng.*, **180**, 108498, 2024.

5. Kadakia, Y., A. Suryavanshi, **A. Alnajdi**, F. Abdullah, and P. D. Christofides, "Encrypted Model Predictive Control of a Nonlinear Chemical Process Network," *Processes*, **11 (8)**, 2501, 2023.

6. **Alnajdi, A.**, F. Abdullah, A. Suryavanshi and P. D. Christofides, "Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems," *Mathematics*, **11 (18)**, 3827, 2023.

7. Suryavanshi, A., **A. Alnajdi**, M. S. Alhajeri, F. Abdullah, and P. D. Christofides, "Encrypted Model Predictive Control Design for Security to Cyber-Attacks," *AIChE J.*, **69**, e18104, 2023.

8. **Alnajdi, A.**, A. Suryavanshi, M. S. Alhajeri, F. Abdullah, and P. D. Christofides, “Machine Learning-Based Predictive Control of Nonlinear Time-Delay Systems: Closed-loop Stability and Delay Compensation,” *Dig. Chem. Eng.*, **7**, 100084, 2023.

9. Alhajeri, M. S., **A. Alnajdi**, F. Abdullah, and P. D. Christofides, “On Generalization Error of Neural Network Models and its Application to Predictive Control of Nonlinear Processes,” *Chem. Eng. Res. & Des.*, **189**, 664–679, 2023.

10. Kadakia, Y. A., F. Abdullah, **A. Alnajdi**, and P. D. Christofides, "Encrypted distributed model predictive control of nonlinear processes," Control Eng. Pract., **145**, 105874, 2024.

Chapter 1

Introduction

1.1 Motivation

In the field of process control, understanding the behavior of the process dynamics is essential for developing an efficient control system. Hence, having a reliable model to describe these dynamics, especially in time-varying operations, is an ongoing challenge for process control engineers. Early modeling approaches involve using mathematical tools to describe process dynamics. This requires extensive knowledge of the physicochemical mechanisms of the system along with well-established scientific principles such as conservation principles (e.g., mass, energy, momentum, charge) and constitutive relations (e.g., transfer rate equations, reaction kinetics, thermodynamical and volume balance relations). Such models are referred to as first-principles models or "white box" models [1]. When accurately constructed, first-principles models offer valuable insights into the system's input/output relationships. However, in large-scale nonlinear chemical processes, constructing first-principles is a challenging task. Most industrial processes are extremely sophisticated, characterized

by significant nonlinearities, a multitude of variable interactions, scalability issues, and other barriers. These complexities make it difficult to apply scientific principles and laws to develop accurate and reliable first-principles models. As a result, classical modeling approaches may struggle to capture the full scope of the system's behavior, leading to poor performance when these models are used in control system frameworks. Therefore, employing advanced modeling techniques is crucial for developing reliable models. These techniques effectively bridge the gap between theoretical model predictions and practical engineering applications, resulting in more accurate and efficient control strategies. The widespread deployment of sensors, the advancements in affordable computing technology, and the growth of wireless network infrastructures have all been key factors in advancing to the fourth industrial revolution. Over the past few decades, there has been a remarkable increase in data generation and computational capabilities, enabling the evolution of big data analytics and the development of artificial intelligence technologies [2, 3]. This rapid progress has shifted the focus towards developing data-driven models for complex nonlinear processes rather than relying on first-principles modeling approaches.

The term "big data" describes exponentially large and complex datasets beyond the scope of classical data processing tools and methods for effective handling and analysis. Hence, machine learning techniques, as a branch of artificial intelligence, provide efficient tools for capturing meaningful insights from big data and facilitating predictive analytics. These techniques enable the identification of hidden patterns, the generation of data-driven forecasts, and the automation of decision-making procedures [4]. Machine learning is regarded as the cornerstone of artificial intelligence and data science. Advancements in ma-

chine learning algorithms, coupled with the availability of affordable yet powerful computing resources and user-friendly programming environments, have enabled the development of numerous open-source libraries (e.g., Keras, TensorFlow, etc.) for machine learning applications [5–7]. Therefore, machine learning has garnered significant interest not only within engineering, but also across a wide range of scientific disciplines, including health care, manufacturing, education, and marketing, among others [8]. With the available variety of machine learning tools, recurrent neural networks (RNNs) and their variants have demonstrated notable success in modeling large-scale, complex nonlinear processes [e.g., 9–12]. RNNs have proven to be efficient when modeling complex dynamic systems with nonlinear input-output relations that involve time-series independencies. They have been employed in many model-based control strategies, demonstrating high accuracy and achieving desired performance while ensuring closed-loop stability. A prominent application of these strategies is model predictive control (MPC) systems that employ machine learning techniques to enhance predictive capabilities and adapt to complex, nonlinear system behaviors in real-time [e.g., 13–18]. The incorporation of machine learning enables MPC to effectively optimize performance and manage system constraints, resulting in improved control and robustness in dynamic systems.

Among the many challenges associated with implementing machine learning-based MPC in real-life scenarios, time-delay systems pose a major challenge. Specifically, input delays in nonlinear systems can often lead to instability and diminished control performance [19]. Therefore, to ensure stability and maintain effective control in closed-loop systems, it is essential to develop approaches in order to compensate for the effect of these delays. By

addressing time delays within machine learning-based MPC frameworks, we can enhance reliability and performance in control systems. In addition to time-delay systems, there are other real-life applications that require further study in the context of machine learning-based MPC. Two-time-scale systems are a type of systems that are characterized by multiple time scales and require specialized methods for designing efficient MPC to ensure stability [20]. Moreover, studying the generalization error bounds of machine learning models provides a deep theoretical understanding of how well the model will perform across different scenarios, providing insights into its potential on new unseen data [21].

1.2 Background

In recent decades, advanced process control strategies have gained considerable popularity in both research and real-life applications. During the 1960s, scientists and control engineers focused on developing advanced control structures beyond conventional proportional-integral-derivative (PID) controllers [22, 23]. PID controllers are widely used in many real-world engineering applications. Moreover, a significant amount of research has been dedicated to the subject of PID designing and tuning [24]. Despite the widespread usage of PID controllers in many industrial applications, their performance can fall short for certain systems, such as those with long dead times. To address this, several methods have been explored to improve PID control in such scenarios. One notable example is the Smith Predictor [25]. Although the Smith Predictor is based on a simple theoretical approach to addressing time-delays, its practical implementation can be complicated [24]. Furthermore,

despite the availability of several well-structured methods and guidelines for configuring PID controllers, determining the optimal settings can be challenging, particularly for systems that are nonlinear or time-variant [26, 27].

In the late 1970s, the works [28, 29] have gained significant attention, establishing the groundwork of MPC theory [26]. MPC is an advanced control strategy where the control problem is formulated as an optimization task. It aims to minimize a defined objective function subject to a set of constraints. MPC computes optimal control actions by predicting future system behavior and solving the optimization problem based on these predictions. MPC offers several key advantages over classical PID control frameworks, including its ability to deal with non-minimum phase processes. Additionally, MPC can take into account several constraints based on the desired performance, including those related to the system's physical structure as well as constraints ensuring stability, safety, and profitability. By incorporating the process model within MPC, it becomes a more robust control framework that adapts to the variations in the system parameters. Moreover, MPC has the ability to handle large-scale, multi-input multi-output (MIMO) systems and is capable of managing nonlinear dynamics [23, 30].

Machine learning has achieved resounding success in many areas of chemical engineering beyond control processes, including the petrochemicals industry, fuel and energy sectors, health and safety, and pharmacy industry, among others [31]. The integration of machine learning techniques within the context of MPC has generated significant interset in the field of process control, leading to the development and application of machine learning based-MPC. This is largely due to the availability of extensive datasets, advanced computa-

tional resources, and enhanced sensing technologies [32]. In particular, neural networks have demonstrated remarkable success in modeling complex, nonlinear systems. They have been widely used to construct process models, especially when accurate first-principles models are difficult to obtain and engineers lack complete knowledge of the theoretical laws governing the process. This is particularly important, as the availability of an accurate process model is a crucial requirement for a successful implementation of MPC. The neural network is capable of capturing the relationship between the process inputs and outputs within its structure without revealing the precise details on how exactly this relationship is modeled, making it function as a "black box" [31]. Among different types of neural networks, RNNs and their variations (e.g., Long-short-term memory recurrent neural network (LSTM RNN), Gated recurrent unit (GRU)) offer distinct advantages, especially when handling dynamic and sequential data. Unlike standard neural networks, RNNs are designed to model processes that involve time-dependent information. The unique structure of RNNs allows them to utilize both current and past information to create a form of memory within its internal states [31, 33, 34]. Hence, RNNs are highly suitable for modeling nonlinear chemical processes, where capturing temporal dependencies and patterns is essential.

Over the years, many applications of RNNs in modeling chemical processes have been highlighted in the literature [e.g., 9, 35–40]. Moreover, scientists have successfully addressed many types of systems and challenges within the realm of machine learning-based MPC, providing useful theoretical insights and explanations. Several efforts have been done to study model predictive control of time delay systems, providing both theoretical analysis and applications [e.g., 41–45]. However, this dissertation focuses on discussing machine learning-

based MPC for nonlinear time-delay systems, emphasizing closed-loop stability analysis and delay compensation. Additionally, with the widespread use of machine learning in numerous engineering applications, it is essential to evaluate of these models under different scenarios to achieve the desired closed-loop performance within MPC. Studying the generalization error bounds for machine learning models has been a focal point for many researchers, as understanding and analyzing these bounds contribute to constructing a reliable and effective machine learning-based MPC. Various investigations have been done on generalization error bounds for different machine learning models [e.g., 21, 46, 47]. This dissertation investigates the generalization error bounds of partially-connected recurrent neural networks (PCRNNs) and LSTM RNNs through statistical learning theory. It also explores the generalization error bounds for RNNs and establishes system stability results for uncertain nonlinear processes. Furthermore, the dissertation examines the generalization error bound for models of two-time-scale systems.

1.3 Dissertation objectives and structure

This dissertation presents the integration of machine learning methods with MPC frameworks to enhance the stability of different types of large-scale, complex nonlinear systems. It offers a comprehensive overview of theoretical strategies and analyses for control, modeling, and evaluation. The proposed theoretical approaches are demonstrated through various examples of applications to nonlinear chemical processes. The main objectives of this dissertation can be summarized as follows:

1. To develop machine learning-based MPC for nonlinear processes with disturbance and establish closed-loop stability using statistical machine learning theory.

2. To design machine learning-based MPC for nonlinear processes with time delays, propose a method to compensate the effect of delays, and establish closed-loop stability.

3. To develop generalization error bounds for PCRNNs and LSTMs using statistical machine learning theory.

4. To study the use of neural networks to model two-time-scale systems, investigate the generalization error for these models.

5. To evaluate and compare the performance of two neural network models for a two-time-scale process—one for the full system and one for the slow states only—within an MPC scheme.

The remainder of this dissertation is organized as follows: The purpose of **Chapter 2** is to study machine-learning-based model predictive control of nonlinear systems with time delays. The proposed approach involves initially building a machine learning model (i.e., Long Short Term Memory (LSTM)) to capture the process dynamics in the absence of time delays. Then, an LSTM-based model predictive controller (MPC) is designed to stabilize the nonlinear system without time delays. Closed-loop stability results are then presented, establishing robustness of this LSTM-based MPC towards small time delays in the states. To handle input delays, we design an LSTM-based MPC with an LSTM-based predictor that compensates for the effect of input delays. The predictor is used to predict future

states using the process measurement, and then the predicted states are used to initialize the LSTM-based MPC. Stabilization of the time-delay system with both state and input delays around the steady state is achieved through the featured design. The approach is applied to a chemical process example, and its performance and robustness properties are evaluated via simulations.

In **Chapter 3**, machine-learning-based predictive control schemes for nonlinear processes subject to disturbances are presented, and closed-loop system stability properties are established using statistical machine learning theory. Specifically, a generalization error bound via the Rademacher complexity method is derived for the recurrent neural networks (RNN) that are developed to capture the dynamics of the nominal system. Then, the RNN models are incorporated in Lyapunov-based model predictive controllers, under which closed-loop stability properties are studied for the nonlinear systems subject to two types of disturbances: bounded disturbances and stochastic disturbances with unbounded variation. A chemical reactor example is used to demonstrate the implementation and evaluate the performance of the proposed approach.

In **Chapter 4**, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have frequently been used to approximate nonlinear dynamic systems utilizing time-series data. The training error of neural networks may often be made suitably modest; however, the accuracy can be further improved by incorporating prior knowledge in the construction of machine learning-based models. Specifically, physics-based RNN modeling has yielded more reliable RNN models than traditional RNNs. Yet, a framework for constructing and assessing the generalization ability of such RNN models as well as LSTM

models to be utilized in model predictive control (MPC) systems is lacking. In this work, a methodological framework to quantify the generalization error bounds is developed for partially-connected RNNs and LSTM models. The partially-connected RNN model is then utilized to predict the state evolution in an MPC scheme. It is illustrated through open-loop and closed-loop simulations of a nonlinear chemical process of two reactors in series that the proposed approach provides a flexible framework for leveraging both prior knowledge and data, thereby improving the performance significantly when compared to a fully-connected modeling approach under Lyapunov-based MPC.

In **Chapter 5**, A general form of nonlinear two-time-scale systems is presented, where singular perturbation analysis is used to separate the dynamics of the slow and the fast subsystems. Machine learning techniques are utilized to approximate the dynamics of both subsystems. Specifically, a recurrent neural network (RNN) and feed forward neural network (FNN) are used for predicting the slow and the fast state vectors, respectively. Moreover, we investigate the generalization error bounds for these machine learning models approximating the dynamics of two-time-scale systems. Next, under the assumption that the fast states are asymptotically stable, our focus shifts towards designing a Lyapunov-based model predictive control (LMPC) scheme that exclusively employs the RNN that predicts the dynamics of the slow states. Additionally, we derive sufficient conditions to guarantee closed-loop stability of the system under sample-and-hold implementation of the controller. A nonlinear chemical process example is used to demonstrate the theory. In particular, two RNN models are constructed, one to model the full two-time-scale system and the other to predict solely the slow state vector. Both models are integrated within an LMPC scheme, and we compare

their closed-loop performance while assessing the computational time required to execute the LMPC optimization problem.

Chapter 6 summarizes the main results of this dissertation.

Chapter 2

Machine Learning-Based Predictive Control of Nonlinear Time-Delay Systems: Closed-loop Stability and Input Delay Compensation

2.1 Introduction

Machine learning algorithms have generated considerable interest in the field of control of nonlinear process systems. This is because of their ability to capture the system's dynamics and to model large-scale, complex, nonlinear systems. Moreover, the existence of large data sets, powerful computers, and the variety of machine learning training algorithms have contributed to the recent surge of machine learning being applied to numerous engineering applications. Although, historically, first-principles modeling approaches have been widely

adapted in modeling chemical processes, they can be difficult and/or time-consuming to derive when dealing with large-scale, complex, nonlinear processes. In contrast, machine learning techniques have made a significant impact in the field of nonlinear control systems and have shown great success in modeling large-scale, complex, nonlinear processes (e.g., [21, 48–54]). Researchers in the field have started adapting the science of machine learning since the 90's [55, 56], when they started introducing the concept of machine learning to the field of chemical engineering, and many promising contributions and applications have been observed since then [57].

A wide variety of machine learning techniques are used for modeling nonlinear systems, and one of the powerful and efficient tools is the long short-term memory (LSTM) recurrent neural networks. LSTMs were introduced in [58] and are a type of recurrent neural network (RNN) with a unique structure that allows it to model dynamical systems and overcome numerical issues commonly encountered in traditional RNNs. The incorporation of LSTM models into advanced model-based control strategies such as model predictive control (MPC) comes with notable success. For example, in [51], a distributed MPC was designed and implemented using an LSTM model. Moreover, LSTMs were also used to design a decentralized MPC in [50]. LSTMs were proven to be efficient when it comes to dealing with noisy data and controlling nonlinear processes. For example, in [59], LSTMs were used to model a large-scale, chemical process using noisy, industrial data from Aspen Plus Dynamics, and its closed-loop performance under LSTM-based MPC was studied.

The unique structure of LSTM networks enables them to model systems that require long-time interval dependencies such as nonlinear time-delay systems, which require robust-

ness considerations with regards to closed-loop stability and performance criteria. Time delays are a common phenomenon that occurs naturally in chemical processes. A common reason for state delays is transportation lag when materials flow through a pipe. These types of delays are usually reflected as state delays in the first-principles model of the process. Additionally, input delays are another common type of delays in chemical processes, typically caused by control actuator dynamics. Moreover, delays can also arise due to the approximation of complex reaction mechanisms and/or nonlinear higher-order dynamics. Therefore, investigating the stability, robustness, and performance properties of closed-loop time-delay systems is an important topic in the field of control systems.

A nonlinear time-delay system is usually represented by a differential difference equation (DDE). The behavior of DDEs is different from classical ordinary differential equations (ODEs) in several manners. One important difference between them is that, when solving an ODE, an initial condition is required, whereas, for a DDE, the history of states and inputs has to be stored; in other words, an initial history function has to be specified or computed for both states and manipulated inputs. As a matter of fact, time-delayed systems, even if the delay is small, constitute infinite-dimensional systems regardless of the dimension of the state vector of the system (see [60] for more details on DDEs). Significant efforts have been undertaken to study nonlinear time-delay systems (DDEs) and investigate their stability properties. Most relevant to our present work, in [41], an economic MPC for nonlinear time-delay systems was designed using a first-principles model, with closed-loop stability results being derived using input-to-state stability theorems incorporated with Lyapunov-Razumikhin type arguments. In earlier works, [61, 62] established fundamental results on

feedback control and robustness of nonlinear time-delay systems with applications to process systems. For the case of controlling nonlinear systems with time-varying measurement delays, several works in the literature have discussed this case. For example, [63] studied the stability of differential-functional equations with discrete and distributed delays. Additionally, [64] provided an overview of the stability of linear systems with time-varying delays and reviewed recent works that have been conducted in this topic.

Several common approaches to control nonlinear time-delay systems can be found in the literature. In the case of small time-delay values, earlier approaches include assigning the values of the time-delays to zero and proceeding with the control design using the resulting ODE systems. This control technique can be effective with acceptable closed-loop stability and performance when the nonlinear system suffers from small state delays. Yet, for larger values of time-delays, and in particular when input delays are present, it is necessary to utilize other approaches that are more involved and can compensate for the effect of input delays. Such approaches include using a predictor with the controller design, to predict future values of the state that can be used in the controller. In 1957, the classical Smith predictor was proposed, and it has become one of the most popular predictor structures used for linear time-delay systems [65]. The Smith predictor has proven to be effective in many theoretical aspects and engineering applications. Moreover, various results are found in the literature, adapting the design of the Smith predictor to produce more variations of predictor designs in order to address different types of linear and nonlinear systems with input delay. For instance, [66, 67] presented designs of predictors that can handle nonlinear systems with input delay through the use of conceptional insights from the Smith predictor approach.

In the context of data-based modeling for feedback controller design, machine learning techniques have been incorporated in MPC with resounding success due to their notable accuracy, efficiency, and ability to capture complex systems' dynamics, as successfully demonstrated via numerous chemical process applications in [68]. Specifically, [48, 49] provide fundamental theoretical and practical insights of machine learning-based MPC and necessary stability analyses. The first conceptualization of MPC can be dated to 1978 [29], and, since then, many contributions and applications have been made in both industry and academia. MPC has proven to be efficient in a diverse array of applications due to its ability to handle multiple inputs, outputs, and constraints by solving an optimization problem that minimizes a desired objective function of the inputs and outputs [52, 53] subject to constraints while incorporating measurement feedback into the calculations. With a sufficiently accurate process model, MPC can also handle systems where only noisy data is available [69]. The control actions of MPC are computed through repeatedly solving an optimization problem in a finite time horizon, with both state and input constraints. This guarantees the stability and boundedness of the trajectories of the nonlinear system at all times.

In light of the above considerations, in this chapter, we apply machine learning to develop a model—specifically an LSTM model—using data from the process model without the time delays and use this LSTM model to design a model predictive controller that renders the process model without the time delays stable. Subsequently, we established that the LSTM-based MPC ensures stability of the closed-loop system under sufficiently small state delays. Additionally, we design an LSTM-based predictor to compensate for input delays. Finally, an application of the LSTM-based MPC to a chemical reactor under both

state and input delays is presented.

2.2 Preliminaries

2.2.1 Notation

A time-dependent vector is denoted by $x(t) \in \mathbb{R}^n$. x^T denotes the transpose of vector x. The Euclidean norm of a vector is denoted by $|\cdot|$, and the infinity norm of a function $\phi \in C([a,b],\mathbb{R}^n)$ is represented by $\|\cdot\|$, such that $\|\phi\| := \max_{a\le s\le b}|\phi(s)|$, where $C([a,b],\mathbb{R}^n)$ is the space of continuous functions mapping the interval $[a,b]$ to $\mathbb{R}^n$. Set subtraction is denoted by '\', such that $A\backslash B := \{x \in R^n | x \in A, x \notin B\}$. $S(\Delta)$ denotes the family of piecewise constant, right-continuous functions with period Δ.

2.2.2 Class of Systems

The following family of differential difference equations (DDEs) describes the class of nonlinear time-delay systems considered in this work:

$$\dot{x}(t) = F(x,u) = f\Big(x(t), x(t-d_1), u(t-d_2)\Big) \tag{2.1}$$

$x(t)$ is the n-dimensional state vector, and $u(t)$ is the m-dimensional control input vector bounded by $u \in U$. The set U, defined as $U := \{|u_i| \le u_{\max,i}, i = 1,...,m\}$. The vector $f(\cdot)$ is a locally Lipschitz vector function of its arguments. Under the assumption that $f(0,0,0) = 0$, the origin is a steady state of Eq. (2.1). $d_1 > 0$ is the value of the state delay and $d_2 > 0$ is the value of the input delay. Moreover, without loss of generality, the

initial time is taken to be zero (i.e., $t_0 = 0$), and the initial data is denoted as ϕ_x, where $\phi_x \in C([-d_1, 0], \mathbb{R}^n)$. Additionally, the symbol ϕ_u represents the initial input function, where $\phi_u \in C([-d_2, 0], \mathbb{R}^m)$. Hence, ϕ_u is bounded and is assumed to be piecewise continuous over its domain. The system of Eq. (2.1) can be expressed as a perturbed form of the system without delays in the following form:

$$\dot{x}(t) = F(x, u, \xi) = f(x(t), x(t) + \xi_1(t), u(t) + \xi_2(t)) \tag{2.2a}$$

$$\xi_1(t) = x(t - d_1) - x(t) \tag{2.2b}$$

$$\xi_2(t) = u(t - d_2) - u(t) \tag{2.2c}$$

where $\xi^T := [\xi_1^T \quad \xi_2^T] \in D \times U \subset \mathbb{R}^{n+m}$ is the bounded perturbation vector, and D is an open neighborhood around the origin.

Remark 2.1. *In this work, differential difference equations (DDEs) are used to describe the general class of nonlinear time-delay systems. A number of different methods to describe nonlinear time-delay systems exist in the literature, such as first order plus dead time and second order plus dead time models, which are specific and assume certain (linear) model structures. Therefore, nonlinear differential difference equations with constant delays were chosen as the class of systems in this work to make the analysis more general. However, other works that describe nonlinear time-delay systems with functional differential equations can be found, and our results may be extended to such model structures as well. For example, describing systems with multiple state and input delays and systems with time-varying delays can be done using functional differential equations.*

2.2.3 Stabilization via control Lyapunov function

Taking into consideration the ODE system in Eq. (2.2), we assume that there exists a locally Lipschitz feedback controller $\Phi(x) \in U$, such that the origin of the nominal system of Eq. (2.2) (i.e., with $\xi(t) \equiv 0$) is exponentially stable. Hence, the system of Eq. (2.2) is stabilizable, and there exists a continuously differentiable Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ such that the following inequalities hold:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{2.3a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x), 0) \leq -c_3|x|^2, \tag{2.3b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq c_4|x| \tag{2.3c}$$

given that c_i are positive constants, where $i = 1, 2, 3, 4$, for all $x \in \mathbb{R}^n \subset D$. Since the system $F(x, u, \xi)$ has a Lipschitz property, together with the bounded behaviour of the input u and the perturbation ξ, there exist positive constants M, L_x, $L_x^{'}$, L_ξ, and $L_\xi^{'}$ such that the following inequalities hold for all $x, x' \in D$ and $u \in U$:

$$|F(x, u, \xi)| \leq M \tag{2.4a}$$

$$|F(x, u, \xi) - F(x', u, 0)| \leq L_x|x - x'| + L_\xi|\xi| \tag{2.4b}$$

$$\left|\frac{\partial V(x)}{\partial x} F(x, u, \xi) - \frac{\partial V(x')}{\partial x} F(x', u, 0)\right| \leq L_x^{'}|x - x'| + L_\xi^{'}|\xi| \tag{2.4c}$$

Additionally, the closed loop stability region of the nonlinear system of Eq. (2.2) is characterized by the region Ω_ρ, where $\Omega_\rho := \{x \in D | V(x) \leq \rho\}$

2.2.4 Long Short-Term Memory Recurrent Neural Networks

An LSTM network is a special type of RNN that is composed of a number of gates and cells, which gives it a unique structure. LSTMs were introduced to overcome a number of limitations found in classical RNNs, primarily the vanishing gradient problem. This phenomenon occurs very often in classical RNNs, where the product of the gradients in the loss function get smaller in value as we proceed through the layers of the network, causing the loss function to have a value near zero for older data points. Hence, training the network becomes harder, and analyzing data over long time periods becomes a challenging task. Additionally, LSTMs are well known for modeling systems that require long time dependencies. The LSTM network is designed to predict future states of the process using the past state measurements and future control actions. Therefore, the input sequence of the LSTM network is denoted by $p \in \mathbb{R}^{(n+m)\times T}$, while the output of the LSTM network is denoted by $\hat{x} \in \mathbb{R}^{n\times T}$, where T is the number of time steps or repeating LSTM modules within one sampling period. Figure 2.1 illustrates the LSTM structure.

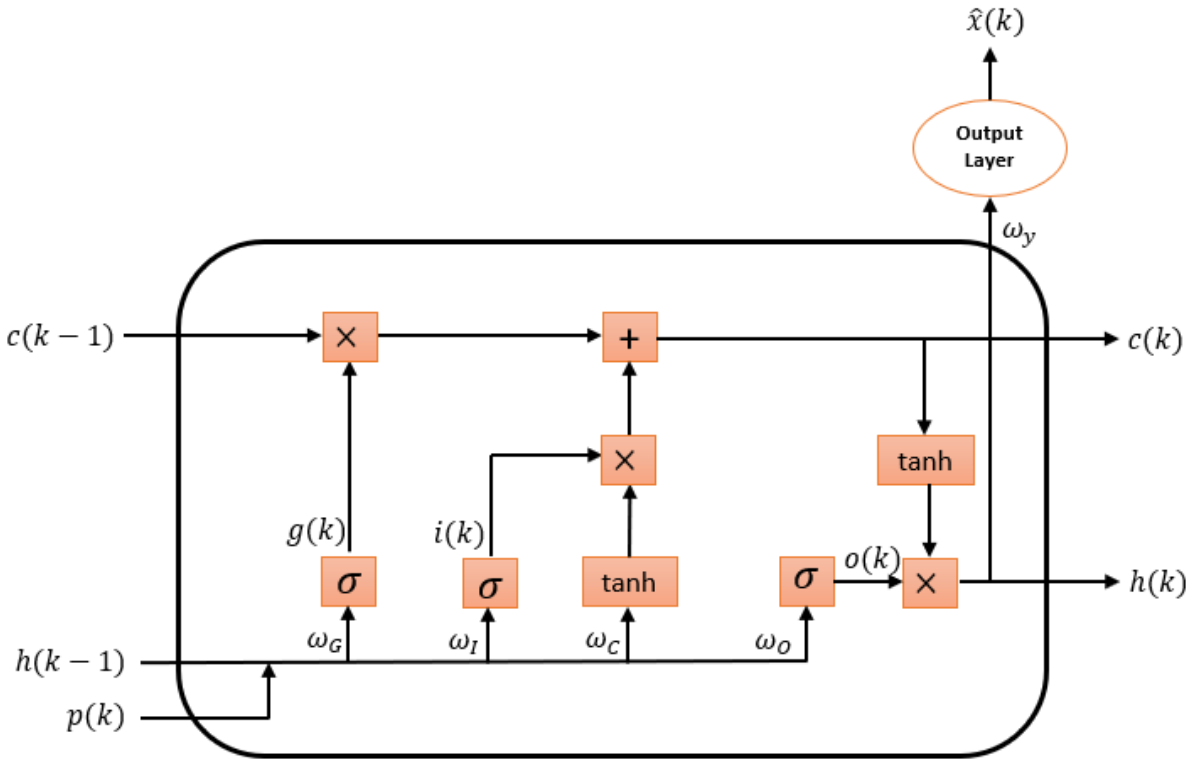

Figure 2.1: LSTM structure

The LSTM unit is expressed through the following equations:

$$g(k) = \sigma\left(\omega_g^p p(k) + \omega_g^h h(k-1) + b_g\right) \tag{2.5a}$$

$$i(k) = \sigma\left(\omega_i^p p(k) + \omega_i^h h(k-1) + b_i\right) \tag{2.5b}$$

$$c(k) = i(k)\tanh\left(\omega_c^p p(k) + \omega_c^h h(k-1) + b_c\right) + g(k)c(k-1) \tag{2.5c}$$

$$o(k) = \sigma\left(\omega_o^p p(k) + \omega_o^h h(k-1) + b_o\right) \tag{2.5d}$$

$$\hat{x}(k) = \omega_y h(k) + b_y \tag{2.5e}$$

$$h(k) = o(k)\tanh(c(k)) \tag{2.5f}$$

where the input sequence is denoted by $p(k)$, and the vector $\hat{x}(k) \in \mathbb{R}^{n\times T}$ represents the

LSTM network output, with $k = 1, \ldots, T$. The weight matrix and bias vector for the output are ω_y and b_y, respectively. Additionally, $h(k)$ is the internal state, while $g(k), i(k)$, and $o(k)$ represent the outputs from the forget gate, the input gate, and the output gate, respectively. $\omega_g^p, \omega_g^h, \omega_i^p, \omega_i^h, \omega_o^p$, and ω_o^h are the weight matrices for the input vector p and the hidden state vector h within the forget gate, the input gate and the output gate, respectively. b_g, b_i, and b_o are the bias vectors for the forget gate, the input gate, and the output gate, respectively. Moreover, $c(k)$ represents the cell state, which is in charge of storing and passing the essential information through successive LSTM units. More precisely, the first term in Eq. (2.5c) is in charge of storing the new, important information coming from the input gate $i(k)$ in the cell state $c(k)$ that is to be passed to the next LSTM unit. In contrast, the second term in Eq. (2.5c) uses the forget gate $g(k)$ to compute the information that should be discarded from the previous state $c(k-1)$. Additionally, the weight matrices associated with the cell state are represented by ω_c^p and ω_c^h, where ω_c^p indicates the weight matrix for the input vector, and ω_c^h is the weight matrix for the hidden state vector. b_c represents the bias vector associated with the cell state. σ and $tanh$ are the nonlinear *sigmoid* and hyperbolic tangent activation functions, respectively. The LSTM input sequence is $p \in \mathbb{R}^{(n+m)\times T}$, which consists of the past state measurements x and the manipulated inputs u.

In this study, we develop an LSTM network model with the following form as a continuous-time nonlinear system:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T z \tag{2.6}$$

where $\hat{x} \in \mathbb{R}^n$ is the LSTM state vector, and $u \in \mathbb{R}^m$ is the manipulated input. $A \in \mathbb{R}^{n\times n}$ and

$\Theta \in \mathbb{R}^{(n+m+1)\times n}$ are the weight matrices, and $z = [z_1...z_{n+m+1}]^T = [\sigma(\hat{x}_1)...\sigma(\hat{x}_n)\ u_1...u_m\ 1]^T \in \mathbb{R}^{n+m+1}$ is a vector associated with the network states $\hat{x}$ and the manipulated input u.

Remark 2.2. *Many applications in the control engineering field require machine learning models that can deal with sequential data. The inputs and outputs of a chemical process are often considered as time-series data. Sequential neural network models, such as recurrent neural networks (RNNs), gated recurrent units (GRUs) and long short-term memory networks (LSTMs), are types of machine learning models that are well-suited to model nonlinear dynamical systems (including the nonlinear time-delay systems considered in our work) using sequential time-series input-output data. The LSTM network is a strong candidate given its potential to overcome problems that occur in RNNs and its proven ability to model systems that require long time dependencies such as nonlinear time-delay systems, which require robustness considerations with regards to closed-loop stability and performance criteria.*

2.2.5 Data generation and model training process

The goal is to design a stabilizing control law for the nonlinear time-delayed system of Eq. (2.1) under small-time delays or, in other words, small and bounded perturbations. In this work, we will be adapting a machine learning method, specifically LSTM neural networks. The first step in the development of an LSTM model is to generate data. We follow the data generation technique described in [48], which is to run extensive open-loop simulations of the nonlinear system of Eq. (2.2) without the time delays. The aim is to capture the dynamics of the system for all $x \in \Omega_\rho$ and $u \in U$, sweeping over all possible combinations of initial conditions $x_0 \in \Omega_\rho$ and inputs $u \in U$. Moreover, we note that, in

the data generation process as well as when simulating the system of Eq. (2.2) without time delays, the input $u \in U$ is applied in a sample-and-hold manner, where it is fed to the system of Eq. (2.2) without time delays as a piecewise constant function $u(t) = u(t_k), \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ (Δ is the sampling period). Then, we integrate the system of Eq. (2.2) without time delays using the explicit Euler method with a sufficiently small integration time step $h_c \ll \Delta$. Therefore, we are able to create a set of time-series data for the state x within a chosen level set of the Lyapunov function, denoted as the operating region Ω_ρ. This procedure yields a large data set of diverse trajectories to be passed to the model training phase. Subsequently, we train the LSTM network using the generated data and the neural network library Keras. The architecture of the LSTM network of Eq. (2.6) is designed to be such that, given the current state measurement and future manipulated inputs, the LSTM is able to predict the future states for at least one sampling period ahead, i.e., $x(t) \in [t_k, t_k+\Delta)$. As a result, we obtain an LSTM model that has the ability to capture the dynamics of the system with sufficiently small modeling error. We point out that the gathered data set is divided into three main sets—the training, validation, and testing sets, each for its own purpose.

Remark 2.3. *The LSTM model can be trained using data from the delayed process. However, training on the basis of the ODE process model without delays allows us to further evaluate the robustness of this model with respect to applying it in an MPC that is used to control the delayed process, despite the training data being from the system without delays.*

Remark 2.4. *Referring to the nonlinear system of Eq.* (2.2)*, the existence of perturbation or*

disturbances in the system indicates that we may be dealing with noisy data in the training phase, which can affect the accuracy of the developed LSTM model. In some cases, the LSTM model can fail to make the correct predictions due to the existence of perturbation in the data set. Additionally, some studies indicate that the perturbed data makes the LSTM model more robust and enhances the performance of the model (see [70] for more details on this topic). It is important to highlight that, in this chapter, we generated data and trained the LSTM model based on the nominal system of Eq. (2.2) *(i.e., with* $\xi = 0$*). Using machine learning techniques to control perturbed nonlinear systems is a topic that requires further studies. However, in further sections, we will show that the LSTM model developed based on the nominal system of Eq.* (2.2)*, when incorporated in a Lyapunov-based model predictive controller (LMPC), is able to stabilize the perturbed nonlinear system of Eq.* (2.2)*, under the condition of bounded perturbation, and eventually under sufficiently small state delays.*

2.3 Robustness of LSTM-based LMPC to Small Time State Delays

In this section, we will focus on the closed loop stability analysis of the perturbed nonlinear system of Eq. (2.2), taking into consideration sufficiently small state delays only (i.e., $d_2 = 0$ and, hence, $\xi_2 = 0$). However, the stabilization of the perturbed system of Eq. (2.2) in the presence of both state and input delays will be achieved using a predictor feedback LSTM-based LMPC methodology in Section 2.5. Additionally, knowing that the state delays are represented through the perturbation term $\xi_1(t) = x(t-d_1)-x(t)$, the upper

bound of the perturbation can be written as follows:

$$|\xi(t)| = |\xi_1(t)| = |x(t - d_1) - x(t)| \leq d_1 \|x_d(t)\| \tag{2.7}$$

where $\|x_d(t)\|$ is the max-norm of $x_d(t) \in C([-d_1, 0], \mathbb{R}^n)$ (i.e., $\|x_d(t)\| = \max_{\theta \in [-d_1, 0]} |x(t - \theta)|$).

2.3.1 Stabilization of LSTM models via control Laypunov function

Taking into consideration the LSTM model developed in Eq. (2.6), we assume that there exists a locally Lipschitz feedback controller $\Phi_{nn}(x) \in U$ such that exponential stability of the LSTM model is attained at the origin. This implies that there exists a continuously differentiable Lyapunov function $\hat{V} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ such that the following inequalities hold:

$$\hat{c}_1 |x|^2 \leq \hat{V}(x) \leq \hat{c}_2 |x|^2 \tag{2.8a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{c}_3 |x|^2 \tag{2.8b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{c}_4 |x| \tag{2.8c}$$

given that $\hat{c}_i$ are positive constants, for all $x \in \mathbb{R}^n \subset \hat{D}$ where, $i = 1, 2, 3, 4$, and $\hat{D}$ is an open neighborhood around the origin. The LSTM model of Eq. (2.6) has a stability region denoted as $\Omega_{\hat{\rho}}$, characterized as a compact set embedded in $\hat{D}$ as follows: $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. In addition, there exist positive constants M_{nn} and L_{nn} such that the following

inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|F_{nn}(x,u)| \leq M_{nn} \tag{2.9a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x,u) - \frac{\partial \hat{V}(x')}{\partial x} F_{nn}(x',u) \right| \leq L_{nn}|x - x'| \tag{2.9b}$$

The proposition shown below illustrates that the feedback control input $u = \Phi_{nn}(x) \in U$ can stabilize the nominal system of Eq. (2.2), under a sufficiently small modeling error.

Proposition 2.1. *(c.f Proposition 2 in [48]) Consider the LSTM model of Eq.* (2.6) *that satisfies the stabilizability criteria of Eq.* (2.8) *and is exponentially stable around the origin under the control law $u = \Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$. Then, the origin of the perturbed nonlinear system of Eq.* (2.2) *is exponentially stable around the origin for all $x \in \Omega_{\hat{\rho}}$ under the condition that there exists a positive real number γ, where $\gamma < \hat{c}_3/\hat{c}_4$. Additionally, γ is the upper bound of the modeling error between the nominal system of Eq.* (2.2) *with $\xi = 0$ and the LSTM model (i.e., $\nu = |F(x,u,0) - F_{nn}(x,u)| \leq \gamma|x|$ for all $x \in \Omega_{\hat{\rho}}$).*

Proof. We proceed by following the proof of Proposition 2 in [48]. The goal is to prove that the nominal system of Eq. (2.2) is exponentially stable around the origin for all $x \in \Omega_{\hat{\rho}}$. This can be achieved by showing that $\dot{\hat{V}}(x)$ is negative for the nominal system of Eq. (2.2) under the stabilizing control law $u = \Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$. Using the inequalities in

Eqs. (2.8b) and (2.8c), $\dot{\hat{V}}(x)$ can be computed as follows:

$$\begin{aligned}\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{nn}(x), 0) \\ &= \frac{\partial \hat{V}}{\partial x}(F_{nn}(x, \Phi_{nn}(x)) + F(x, \Phi_{nn}(x), 0) - F_{nn}(x, \Phi_{nn}(x)) \\ &\leq -\hat{c}_3|x|^2 + \hat{c}_4|x|(F(x, \Phi_{nn}(x), 0) - F_{nn}(x, \Phi_{nn}(x))) \\ &= -\hat{c}_3|x|^2 + \gamma\hat{c}_4|x|^2\end{aligned} \tag{2.10}$$

By letting $\gamma < \frac{\hat{c}_3}{\hat{c}_4}$, we achieve $\dot{\hat{V}}(x) \leq -\tilde{c}_3|x|^2 \leq 0$ where $\tilde{c}_3 = -\hat{c}_3 + \gamma\hat{c}_4 \geq 0$ and, consequently, closed-loop stability of the nominal system of Eq. (2.2) around the origin under the control law $\Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$. □

2.3.2 Sample-and-hold implementation of Lyapunov-based controller

It is important to highlight that the LSTM-based LMPC is designed using the LSTM model generated in Eq. (2.6), where the control actions are executed in sample-and-hold fashion. In order to study the robustness of the LSTM-based LMPC to small time-delays in the states, we note that, in the next two propositions, we will consider state delays only (i.e., $\xi_2 = 0$). Given that the state delays are sufficiently small, this implies that the perturbation ξ_1 is bounded. The following proposition shows that the error between the state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ and the predicted state by the LSTM model Eq. (2.6) is bounded.

Proposition 2.2. *(c.f Proposition 3 in [48]) Consider the perturbed nonlinear system*

of Eq. (2.2) *with* $\xi_2 = 0$ *in the presence of bounded disturbances (i.e.,* $|\xi(t)| = |\xi_1(t)| \leq d_1\|x_d(t)\|, \|x_d(t)\| = \max_{\theta\in[-d_1,0]} |x(t-\theta)|$*) and the LSTM model of Eq.* (2.6) *with the same initial condition* $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$*. There exists a class* $\mathcal{K}$ *function* $f_\xi(\cdot)$ *and a positive constant* κ *such that the following inequalities hold for all* $x, \hat{x} \in \Omega_{\hat{\rho}}$ *and* $|\xi(t)| = |\xi_1(t)| \in D \subset \mathbb{R}^n$*:*

$$|x(t) - \hat{x}(t)| \leq f_\xi(t) := \frac{L_\xi\, d_1\|x_d(t)\| + \nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{2.11a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{2.11b}$$

Proof. Let $e(t) = x(t) - \hat{x}(t)$ represent the error vector between the state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ and the state of the LSTM model of Eq. (2.6). The following bound can be found for the time-derivative of $e(t)$:

$$\begin{aligned} |\dot{e}(t)| &= |F(x,u,\xi) - F_{nn}(\hat{x},u)| \\ &\leq |F(x,u,\xi) - F(\hat{x},u,0)| + |F(\hat{x},u,0) - F_{nn}(\hat{x},u)| \end{aligned} \tag{2.12}$$

Using Eq. (2.4b), for all $x, \hat{x} \in \Omega_{\hat{\rho}}$ and $|\xi(t)| = |\xi_1(t)| \in D \subset \mathbb{R}^n$, we can bound the first term in Eq. (2.12) as follows:

$$\begin{aligned} |F(x,u,\xi) - F(\hat{x},u,0)| &\leq L_x|x(t) - \hat{x}(t)| + L_\xi|\xi| \\ &\leq L_x|x(t) - \hat{x}(t)| + L_\xi d_1\|x_d(t)\| \end{aligned} \tag{2.13}$$

We observe that the second term of Eq. (2.13) is equivalent to the modeling error, which is upper bounded by ν_m for all $\hat{x} \in \Omega_{\hat{\rho}}$. Therefore, the bound of the modeling error and the

bound of Eq. (2.13) can be used to further bound $\dot{e}(t)$ as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L_x|x(t) - \hat{x}(t)| + L_\xi|d_1|\|x_d(t)\| + \nu_m \\ &\leq L_x|e(t)| + L_\xi d_1\|x_d(t)\| + \nu_m \end{aligned} \tag{2.14}$$

Integrating the inequality of Eq. (2.14) from zero initial conditions (i.e., $e(0) = 0$), the following upper bound for the error vector can be obtained for all $x, \hat{x} \in \Omega_{\hat{\rho}}$ and $|\xi(t)| = |\xi_1(t)| \in D \subset \mathbb{R}^n$:

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{L_{\xi_1} d_1\|x_d(t)\| + \nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{2.15}$$

Using the Taylor series expansion of $\hat{V}(x)$ around $\hat{x}$, we derive Eq. (2.11b) as follows, for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{2.16}$$

where κ is a positive real number. Additionally, we use Eqs. (2.8a) and (2.8b) to further upper bound $\hat{V}(x)$ as follows:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{2.17}$$

□

The following proposition shows that the closed-loop state trajectory $x(t)$ of the perturbed system of Eq. (2.2) with $\xi_2 = 0$ is bounded in $\Omega_{\hat{\rho}}$ for all times and can be driven to

a small neighborhood around the origin, $\Omega_{\rho_{\min}}$, under the controller $\Phi_{nn}(x) \in U$ executed in sample-and-hold fashion.

Proposition 2.3. *(c.f Proposition 4 in [48]) Consider the nonlinear system of Eq.* (2.2) *with* $\xi_2 = 0$ *under the controller* $\Phi_{nn}(\hat{x}) \in U$ *that meets the conditions of Eq.* (2.8) *and stabilizes the LSTM model of Eq.* (2.6)*. The controller is executed in sample-and-hold, i.e.,* $\Phi_{nn}(\hat{x}(t_k)), \forall t \in [t_k, t_{k+1})$*, where* $t_{k+1} := t_k + \Delta$*. Then, there exist* $\epsilon_w > 0$*,* $\Delta > 0$ *and* $\hat{\rho} > \rho_{\min} > \rho_{nn} > \rho_s$ *that satisfy*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}M_{nn}\Delta \leq -\epsilon_s \tag{2.18a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}M_F\Delta \leq -\epsilon_w \tag{2.18b}$$

and

$$\rho_{nn} := \max\{\hat{V}\Big(\hat{x}(t+\Delta)\Big) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{2.19a}$$

$$\rho_{\min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}f_\xi(\Delta) + \kappa\Big(f_\xi(\Delta)\Big)^2 \tag{2.19b}$$

such that for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$*, the following inequality holds:*

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{2.20}$$

and the state $x(t)$ *of the perturbed nonlinear system of Eq.* (2.2) *is bounded in* $\Omega_{\hat{\rho}}$ *for all times and ultimately bounded in* $\Omega_{\rho_{\min}}$*.*

Proof. First, we need to show that $\hat{V}(x)$ is decreasing under the controller $u(t) = \Phi_{nn}(x(t_k))$

for $t \in [t_k, t_{k+1})$. Consider $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, where $x(t_k)$ is the state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$, and $\hat{x}(t_k)$ is the state of the LSTM model in Eq. (2.6). The time-derivative of $\hat{V}(x)$ for all $t \in [t_k, t_{k+1})$ is computed as follows:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
&= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\Big) \\
&\quad + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))\Big) \\
&\quad - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\Big)
\end{aligned} \tag{2.21}$$

Using the inequalities in Eqs. (2.8a) and (2.8b), we obtain the following:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) \leq &- \frac{\hat{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
&- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k)))
\end{aligned} \tag{2.22}$$

Using the Lipschitz inequalities in Eq. (2.9), we can further bound $\dot{\hat{V}}(\hat{x}(t))$ by the following:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &\leq - \frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}|\hat{x}(t) - \hat{x}(t_k)| \\
&\leq - \frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn} M_{nn} \Delta
\end{aligned} \tag{2.23}$$

Hence, if Eq. (2.18a) is satisfied, the following inequality holds for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_s \tag{2.24}$$

Integrating the above inequality over $t \in [t_k, t_{k+1})$ yields the desired result, $\hat{V}(\hat{x}(t_{k+1})) \leq \hat{V}(\hat{x}(t_k)) - \epsilon_s \Delta$. Hence, we have shown that, if Eq. (2.18a) holds, then $\dot{\hat{V}}(x(t))$ is negative for

any $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. This implies that the closed-loop state of the LSTM model of Eq. (2.6) under the sample-and-hold implementation of the controller $u = \Phi_{nn}(\hat{x})$ is bounded within the region $\Omega_{\hat{\rho}}$ and moves toward the origin. Additionally, the region $\Omega_{\rho_{nn}}$ in Eq. (2.19a) is introduced for the case where $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$. In this case, Eq. (2.24) may not hold, and the state $\hat{x}(t_k)$ may leave the region Ω_{ρ_s} within one sampling period. Therefore, $\Omega_{\rho_{nn}}$ is designed to guarantee that the closed-loop state $\hat{x}(t_k)$ of the LSTM model will be bounded in the region $\Omega_{\rho_{nn}}$ within one sampling period, for all $t \in [t_k, t_{k+1})$, $u \in U$ and $\hat{x}(t_k) \in \Omega_{\rho_s}$, because, if $\hat{x}(t_{k+1})$ leaves Ω_{ρ_s}, the controller $u = \Phi_{nn}\left(x(t_{k+1})\right)$ reactivates, such that Eq. (2.24) will be satisfied again at $t = t_{k+1}$, and the state will be driven back toward Ω_{ρ_s} over the next sampling period. Thus far, we can conclude that the state of the LSTM system of Eq. (2.6) is ultimately bounded in $\Omega_{\rho_{nn}}$ for all $x_0 \in \Omega_{\hat{\rho}}$.

The next step is to show that the controller $u = \Phi_{nn}(x) \in U$, applied in sample-and-hold fashion, is able to bound the states of the perturbed nonlinear system of Eq. (2.2) with sufficiently small state delays and with $\xi_2 = 0$, in some neighborhood around the origin. Therefore, we need to show that $\hat{V}(x)$ for the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ is decreasing under the controller $u(t) = \Phi_{nn}(x(t_k))$ for $t \in [t_k, t_{k+1})$ and $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. The time-derivative of $\hat{V}(x(t))$ is calculated as:

$$\begin{aligned}
\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), \xi) \\
&= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0) \\
&\quad + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), \xi) \\
&\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0)
\end{aligned} \tag{2.25}$$

where the first term can be further bounded using the inequality in Eq. (2.10) as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) \leq &-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), \xi) \\ &- \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0)\end{aligned} \tag{2.26}$$

Since f and Φ_{nn} in the perturbed nonlinear system, $F(x(t), \Phi_{nn}(x(t_k)), \xi)$, are locally Lipschitz vector functions, there exists a $\gamma_1^* \in \mathcal{K}$ such that:

$$\begin{aligned}|\xi(t)| = |\xi_1(t)| &= \left| \int_{t-d_1}^{t} f\Big(x(s), x(t-s), \Phi_{nn}(x(s))\Big)\, \mathrm{d}s \right| \\ &\leq d_1 \gamma_1^*(\|x_d(t)\|)\end{aligned} \tag{2.27}$$

where $\|x_d(t)\| = \max_{s\in[-2d_1,0]} |x(t+s)|$. Applying the inequality of Eq. (2.27) and the Lipschitz condition of Eq. (2.4), we obtain the following bound for $\dot{\hat{V}}(x(t))$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}|x(t) - x(t_k)| + L_\xi^{'}|\xi| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} M_F \Delta + L_\xi^{'} d_1 \gamma_1^*(\|x_d(t)\|)\end{aligned} \tag{2.28}$$

Hence, if Eq. (2.18b) is satisfied, the following inequality holds for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and for all $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{2.29}$$

By integrating the above inequality over $t \in [t_k, t_{k+1})$, it is shown that Eq. (2.20) holds, and $\dot{\hat{V}}(x(t))$ is negative for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. Hence, the state of the closed-loop perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ is bounded within the region $\Omega_{\hat{\rho}}$ for all times and can be driven towards the origin in every sampling period under the control law $u = \Phi_{nn}(x)$.

For the case where $x(t_k) \in \Omega_{\rho_s}$, we recall Eq. (2.11a), where the error between the state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ and the state of the LSTM model of Eq. (2.6) is bounded by the term f_ξ. We introduce a compact set $\Omega_{\rho_{\min}} \supset \Omega_{\rho_{nn}}$ satisfying Eq. (2.19b). This ensures that, if the state of the LSTM model of Eq. (2.6) is bounded in $\Omega_{\rho_{nn}}$, then the state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ will be bounded within $\Omega_{\rho_{\min}}$ during one sampling period. If the state $x(t)$ enters $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_s}$, it is shown that Eq. (2.29) is satisfied, which implies that the state will be driven towards the origin.

Finally, we conclude that the closed loop state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$ is always bounded in $\Omega_{\hat{\rho}}$ and ultimately bounded within a small region around the neighborhood (i.e., $\Omega_{\rho_{\min}}$) under the control law $u = \Phi_{nn}(x) \in U$, provided that the assumptions in Proposition 2.3 are satisfied. □

2.4 LSTM-based Model Predictive Control

The trained LSTM model is then incorporated into a Lyapunov-based MPC (LMPC), where it will be used to evaluate future states in the MPC algorithm. The resulting LSTM-based LMPC computes the optimal control actions by solving the following optimization

problem [48, 49]:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t), u(t))dt \tag{2.30a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{2.30b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{2.30c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{2.30d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k)),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{2.30e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{2.30f}$$

In the LMPC formulation, $\tilde{x}(t)$ is the predicted state trajectory, and the number of sampling periods in the prediction horizon is denoted by N. The LSTM-based LMPC computes the optimal control action, $u^*(t)$, over the whole prediction horizon $t \in [t_k, t_{k+N})$. The controller sends the optimal control action $u^*(t_k)$ computed for the first sampling period within the prediction horizon to be applied to the process, after which the resultant real-time state from the process $x(t_k)$ is sent back to the LSTM-based LMPC to resolve the optimal input trajectory at the next sampling time. The cost function of the optimization problem is shown in Eq. (2.30a), and it minimizes the time-integral of $L_{MPC}(\tilde{x}(t), u(t))$ over the prediction horizon. The first constraint of the optimization problem is that of Eq. (2.30b), which uses the LSTM model to predict the states. The second constraint is Eq. (2.30c), which limits the inputs that may be applied, over the whole prediction horizon. Equation (2.30d) is the state measurement at $t = t_k$, which is the initial condition to integrate $\tilde{x}(t)$ from when

integrating Eq. (2.30b). The closed loop trajectory converges towards the steady state value if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ due to the constraint of Eq. (2.30e). Otherwise, if $x(t_k)$ enters the region $\Omega_{\rho_{nn}}$, the constraint in Eq. (2.30f) ensures that the states predicted by the LSTM model remain trapped inside the region $\Omega_{\rho_{nn}}$ for the whole prediction horizon. Additionally, if the assumptions in Proposition 2.2 and Proposition 2.3 are satisfied with small time-delays in the states of the system, stability results derived in [48] show that using the LSTM-based LMPC of Eq. (2.30) guarantees that the closed-loop state of the perturbed nonlinear system of Eq. (2.2) with $\xi_2 = 0$, under the control law $u = \Phi_{nn} \in U$, is bounded within the stability region $\Omega_{\hat{\rho}}$ and ultimately bounded within a small region around the origin, $\Omega_{\rho_{\min}}$, for all $t \geq 0$ and any initial state $x_0 \in \Omega_{\hat{\rho}}$.

2.5 Predictor feedback LSTM-based LMPC methodology

The LSTM-based LMPC is proven to be robust for systems that have sufficiently small state delays. Robustness can be enhanced by tuning some parameters in the LSTM-based LMPC controller [41], such as the weights in the cost function of Eq. (2.30a) or the parameter ρ_{nn}. Hence, the parameters can be chosen to ensure a margin of robustness of the closed-loop system in the presence of state delays. On the other hand, input delays are more challenging, and require further modifications in the controller structure.

In this section, we will present a predictor feedback LSTM-based LMPC methodology, and how the predictor is incorporated within the closed-loop system to compensate for the

effect of input delays. As the name indicates, it is an LSTM-based predictor, in the sense that it uses an LSTM model to predict the evolution of the future states of the process up to a future time equal to the input delay. Specifically, at sampling time t_k, the predictor is used to predict the future state at time $t_k + d_2$, utilizing past state values and the input trajectory that has been calculated previously over t_k to $t_k + d_2$. Additionally, an LMPC formulation with the shifted timescale, $\bar{t}_k = k\Delta + d_2$, is used to calculate the future input trajectory from $\bar{t}_k$ to $\bar{t}_{k+N}$:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{\bar{t}_k}^{\bar{t}_{k+N}} L_{MPC}(\tilde{x}(t), u(t))\, dt \tag{2.31a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{2.31b}$$

$$u(t) \in U, \ \forall\ t \in [\bar{t}_k, \bar{t}_{k+N}) \tag{2.31c}$$

$$\tilde{x}(\bar{t}_k) = \bar{x}(\bar{t}_k) \tag{2.31d}$$

$$\dot{\hat{V}}(x(\bar{t}_k), u) \leq \dot{\hat{V}}(x(\bar{t}_k), \Phi_{nn}(x(\bar{t}_k))),$$

$$\text{if } x(\bar{t}_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{2.31e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall\ t \in [\bar{t}_k, \bar{t}_{k+N}), \ \text{if } x(\bar{t}_k) \in \Omega_{\rho_{nn}} \tag{2.31f}$$

With respect to the LSTM-based predictor initialization, it is important to clarify the following: specifically, we need to assume initial data for both the states and inputs. For states, we assume the initial state data, from time $-d_1$ to 0, to be equal to the value of the states at time $t_k = 0$. The inputs are assumed to be at their steady state values from time 0 to d_2.

Subsequently, at sampling time t_k or, in other words, $\bar{t}_k - d_2$, the predictor receives the state measurement $x(\bar{t}_k - d_2)$ $\Big($or $x(t_k)\Big)$ along with past input measurements from time $\bar{t}_k - d_2$ to $\bar{t}_k$ as its inputs. The LSTM-based predictor is then used to predict the future state value $\bar{x}(\bar{t}_k)$. Subsequently, the output of the predictor is sent to the LSTM-based LMPC to initialize it and compute the optimal input trajectory along the whole prediction horizon. The computed control action $u^*(\bar{t}_k|\bar{t}_k - d_2)$ is then sent to the process to be applied from time $\bar{t}_k$ to $\bar{t}_{k+1}$, which yields the output, i.e., the process state $x(\bar{t}_{k+1} - d_2)$ $\Big($or $x(t_{k+1})\Big)$. Figure 2.2 illustrates the proposed LSTM-based LMPC framework. Additionally, the LSTM-predictor used in this study is a closed-loop predictor (i.e., at each sampling time, a new measurement $x(t_k)$ is sent to it from the process). Hence, unlike open-loop predictors, closed-loop ones play an effective role when trying to control processes with open-loop unstable equilibrium points [41]. Figure 2.3 shows the LSTM-based predictor block in the feedback loop of the closed-loop system.

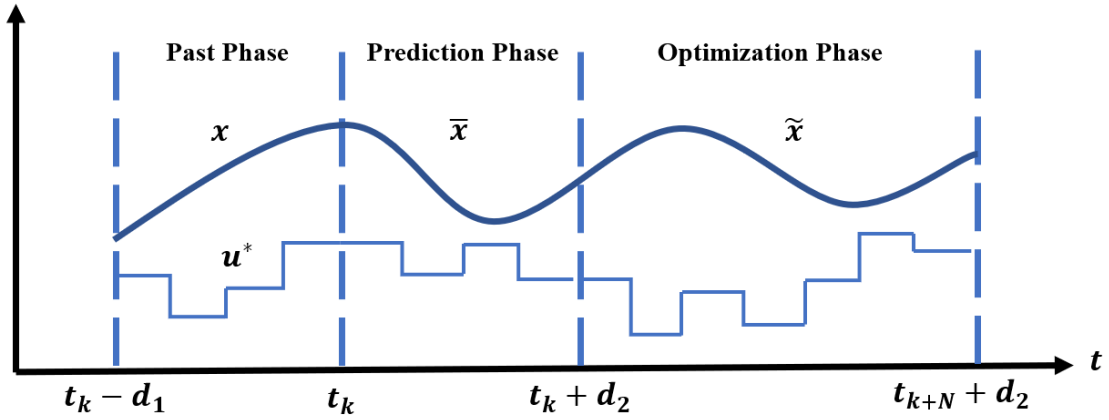

Figure 2.2: Time phases of the states and the control action of the predictor-based control system.

We summarize the implementation of the LSTM based predictor in the following algo-

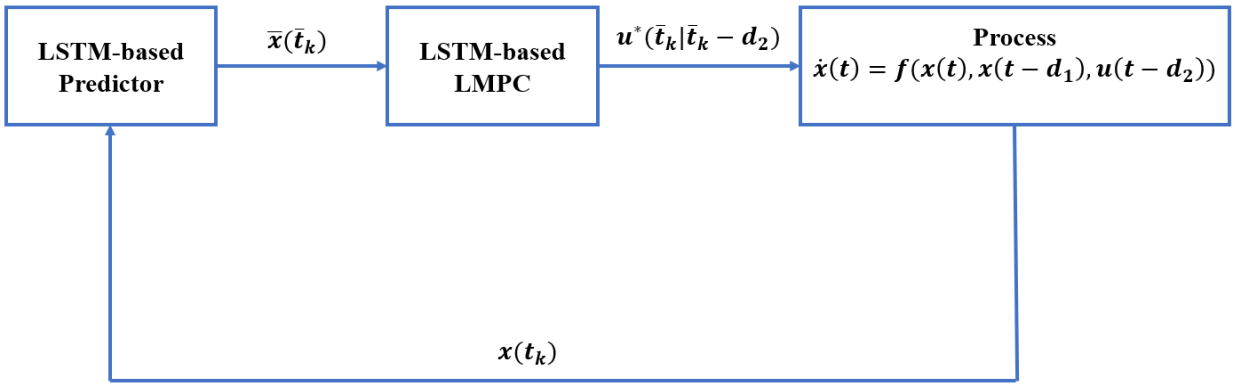

Figure 2.3: Flow diagram of the closed-loop system with the predictor block.

rithm:

Algorithm 2.1. *LSTM-based predictor, MPC feedback implementation.*

1. At sampling time t_k (i.e., $\bar{t}_k - d_2$), the predictor receives $x(\bar{t}_k - d_2)$ and past input measurements from time $\bar{t}_k - d_2$ to $\bar{t}_k$.

2. The predictor predicts the future state $\bar{x}(\bar{t}_k)$.

3. The LSTM-based LMPC is then initialized with the predicted state $\bar{x}(\bar{t}_k)$, and the optimal control input trajectory is computed.

4. The computed control action, $u^*(\bar{t}_k|\bar{t}_k - d_2)$, is then applied to the process from $\bar{t}_k$ to $\bar{t}_{k+1}$.

5. Set $k \longleftarrow k+1$ and go to step 1.

2.6 Application to a Chemical Process Example

To illustrate the use of the LSTM-based LMPC and the LSTM-based predictor for stabilizing a nonlinear system in the presence of small time-delays, we consider the chemical

rector in [41]. In a well-mixed, non-isothermal continuous stirred tank reactor (CSTR), the irreversible, exothermic, and elementary second order reaction transforming a reactant A to a desired product B ($A \rightarrow B$) takes place. Figure 2.4 shows the process flow diagram of the CSTR.

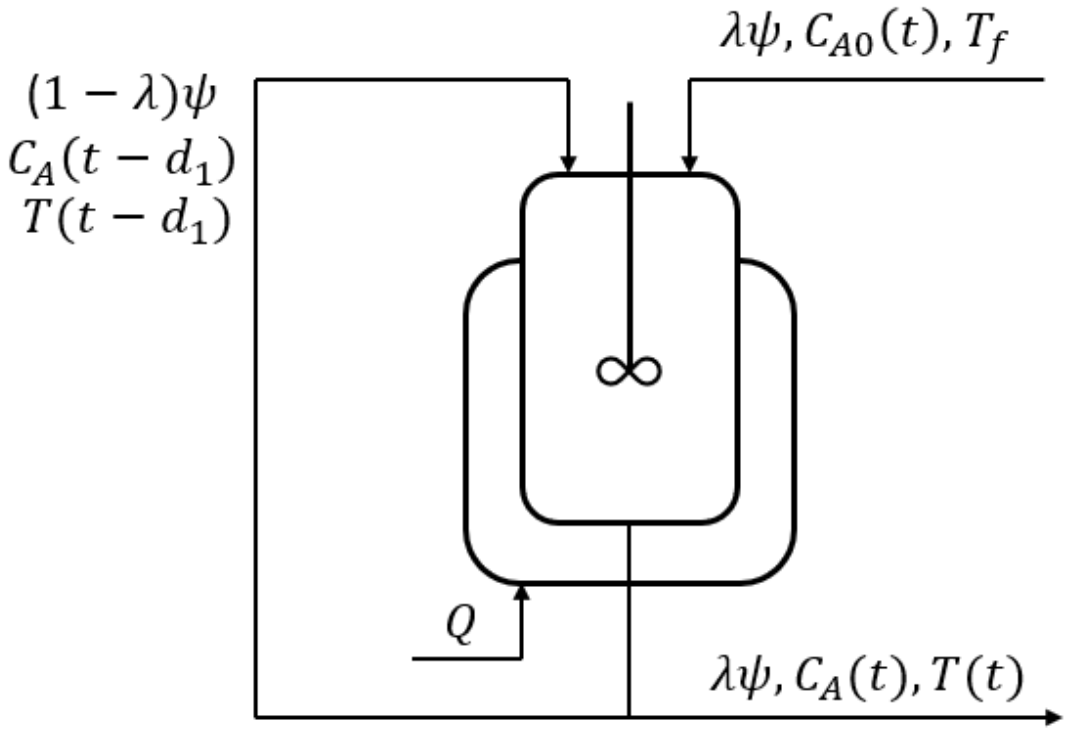

Figure 2.4: Process flow diagram of the CSTR with the recycle stream.

The inlet stream enters the reactor with a flow rate $\lambda\psi$, feed concentration C_{A0}, and a feed temperature T_f. The outlet stream of the reactor is split into two streams. The first stream is the product of the reactor, with a volumetric flow rate $\lambda\psi$, concentration C_A, and temperature T. The second stream is a recycle stream with flow rate $(1-\lambda)\psi$ that is carried back to the reactor. Specifically, the unprocessed portion of chemical A is reused through the recycle steam, where it carries a splitting fraction $(1-\lambda)$ of the outlet stream back to

the reactor. This recycle stream causes a transportation lag. Hence, a time-delay of value d_1 appears in the dynamics of the process. C_{A0} and Q are the feed concentration and the heat rate, respectively, which are the manipulated inputs of the CSTR. The control actuators' dynamics and their operation with dead-times cause an input delay of value d_2, which appear in the process dynamics.

The first-principles model of the second-order CSTR with a recycle stream is described by the following material and energy balance equations:

$$\begin{aligned}\dot{C}_A(t) = &\frac{(1-\lambda)\psi}{V_R}C_A(t-d_1) + \frac{\lambda\psi}{V_R}C_{A0}(t-d_2) \\ &- \frac{\psi}{V_R}C_A(t) - k_0 \exp\left(\frac{-E}{RT(t)}\right)C_A^2(t) \qquad (2.32a)\\ \dot{T}(t) = &\frac{(1-\lambda)\psi}{V_R}T(t-d_1) + \frac{\lambda\psi}{V_R}T_f - \frac{\psi}{V_R}T(t) \\ &- \frac{\Delta H k_0}{\rho_L C_p}\exp\left(\frac{-E}{RT(t)}\right)C_A^2(t) + \frac{Q(t-d_2)}{V_R\rho_L C_p} \qquad (2.32b)\end{aligned}$$

The vector $x^T = [C_A \quad T]$ represents the state vector, where C_A is the concentration of reactant A, and T is the temperature of the reactor. The notations and the parameter values are illustrated in Table 2.1. The inputs of the reactor are bounded as follows: $C_{A0} \in [0.5, 7.5]\,\text{kmol/m}^3$ and $Q \in \left[-8\times10^4, 8\times10^4\right]$ kJ/h. Additionally, the LSTM-based MPC is designed to drive the system to the steady state, $x_s = (C_{As}, T_s) = \left(2.96\,\text{kmol/m}^3, 320\,\text{K}\right)$, which is open-loop asymptotically stable. This is achieved under the input values $C_{A0s} = 4\,\text{kmol/m}^3$ and $Q_s = 12.2\times10^3\ kJh^{-1}$. By converting both the state and input variables to deviation variables, the steady state of the system is shifted to the origin. For the development of the LSTM model, we follow the technique illustrated in Section 2.2.5, where 10^5

data points are generated through extensive open-loop simulations of the nonlinear system of Eq. (2.32) (without the delays) using the explicit Euler method with a sufficiently small time step of $h_c = 10^{-4}\ h$ and sampling period $\Delta = 0.01\ h$. The data set is then split into 80,000 points for training and 20,000 points for validation. Then, using the machine learning library Keras, we construct the LSTM model of Eq. (2.6), consisting of 600 LSTM units. The loss function was chosen to be the mean squared error (MSE). Additionally, during each epoch of the training, both the training and the validation loss were calculated simultaneously. Moreover, early stopping was used, in which the stopping criterion was defined on the basis of the validation data loss and chosen as 2×10^{-6}, i.e., training would terminate once the validation loss was below 2×10^{-6}. Once the early stopping criterion was satisfied, the training was stopped and the final values of the training and validation loss were reported to be 5.4×10^{-5} and 1.4×10^{-6}, respectively; both considered to be sufficiently small. The trained LSTM model is then incorporated into a Lyapunov-based MPC (LMPC). The model's performance is then evaluated not using a test set but rather with respect to its ability to lead to an LMPC that gives satisfactory closed-loop performance and is capable of handling the delays in states and inputs, as this is the fundamental criterion when designing an MPC. The LSTM-based LMPC predicts optimal control laws for a prediction horizon of $N = 3$. Moreover, the Lyapunov function of the CSTR system is defined as

$$V(x) = (x - x_s)^T P (x - x_s) \tag{2.33}$$

where the matrix P is given by

$$P = \begin{bmatrix} 500 & 20 \\ 20 & 1 \end{bmatrix}$$

In the following subsections, we will show the results of the designed LSTM-based LMPC and its ability to stabilize the system. Moreover, we will show simulation results to demonstrate the predictor feedback LSTM-based LMPC methodology and its ability to compensate for the effect of input time-delays.

Concentration of chemical A	C_A
Reactor temperature	T
Feed concentration	C_{A0}
Heat removal rate from the reactor	Q
Splitting fraction	$\lambda = 0.7$
Reaction rate constant	$k_0 = 1 \times 10^9\ m^3 kmol^{-1} h^{-1}$
Feed temperature	$T_f = 300\ K$
Density	$\rho_L = 1 \times 10^3$
Heat capacity	$C_p = 4.18\ kJkg^{-1}K^{-1}$
Reactor volume	$V_R = 1\ m^3$
Flow rate	$\psi = 6\ m^3 h^{-1}$
Heat of reaction	$\Delta H = -7.8 \times 10^4\ kJkmol^{-1}$
Activation energy	$E/R = 5.7 \times 10^4/8.314\ K$

Table 2.1: Notation and parameter values of the CSTR with recycle stream.

2.6.1 LSTM-based LMPC closed-loop simulation results

We first conduct closed-loop simulations for the CSTR under the LSTM-based LMPC with different values of input time delays as they are known to have a significant impact on the state trajectories and stability. Hence, in these simulations, the value of the state delay was fixed at $d_1 = 0.01\ h$ for all simulations results. Figure 2.5 shows the closed-loop trajectories of the CSTR under the LSTM-based LMPC with time delays of $d_1 = 0.01\ h$

and $d_2 = 0.01\ h$. We observe that the trajectories converge to the steady-state values and are stabilized by the controller. This shows that the designed controller is robust to small time-delays, and that closed-loop stability is achieved. In this particular application, when the value of the input delay, d_2, is higher than $0.01\ h$, stability is lost, and we notice oscillations and fluctuations in the closed-loop trajectories as seen in Figs. 2.6 and 2.7, which correspond to input time-delays of $d_2 = 0.02\ h$, and $d_2 = 0.03\ h$, respectively. Moreover, from the trajectories of Figs. 2.6 and 2.7, we observe that increasing the value of the input delay d_2 increases the amplitude of the oscillations around the steady state as well. Hence, the closed-loop trajectories oscillate around the steady state, causing the system to become unstable.

2.6.2 Predictor feedback LSTM-based LMPC closed-loop simulation results

In this section, we will demonstrate the results of closed-loop simulations using the predictor feedback LSTM-based LMPC design. This method was proposed to overcome the performance deterioration that arises due to larger time delays, particularly to compensate for the effect of larger input time-delays (i.e., $d_2 > 0.01\ h$). The scheme was applied for the cases in Section 2.6.1 where we found oscillations in the closed loop stability under the LSTM-based LMPC. Figures 2.8 and 2.9 show the closed-loop trajectories of the CSTR under the predictor feedback LSTM-based LMPC with input time delays of $d_2 = 0.02\ h$ and $d_2 = 0.03\ h$, respectively, while maintaining $d_1 = 0.01\ h$ in all simulations. From the results, we observe, under larger values of input time-delays, significant improvement in the

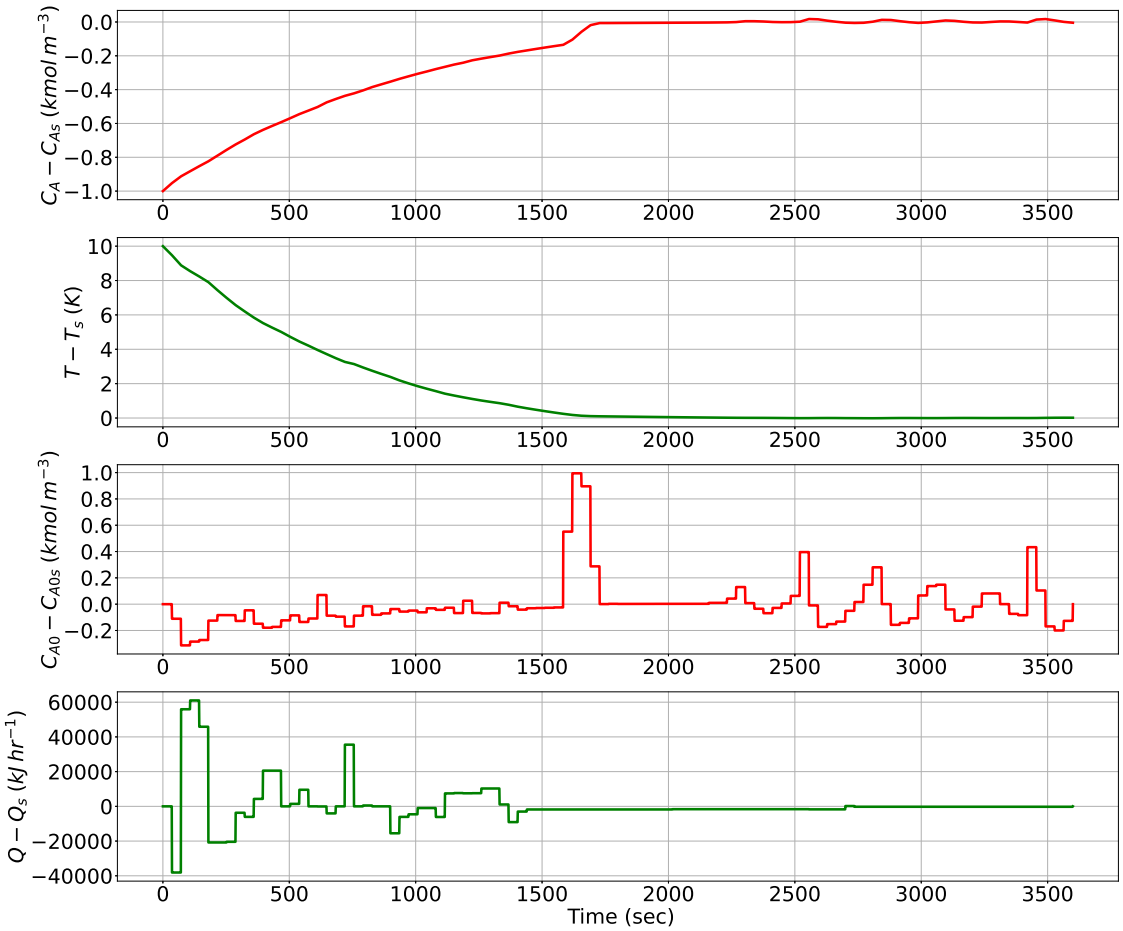

Figure 2.5: Closed-loop state and input trajectories under LSTM-based LMPC with time delays: $d_1 = 0.01\ h$ and $d_2 = 0.01\ h$.

closed-loop performance under the proposed control system. This is achieved through the use of an LSTM-based predictor together with the LSTM-based LMPC in the feedback loop. As the trajectories converge to their steady state values without oscillations, the process is considered to be stabilized.

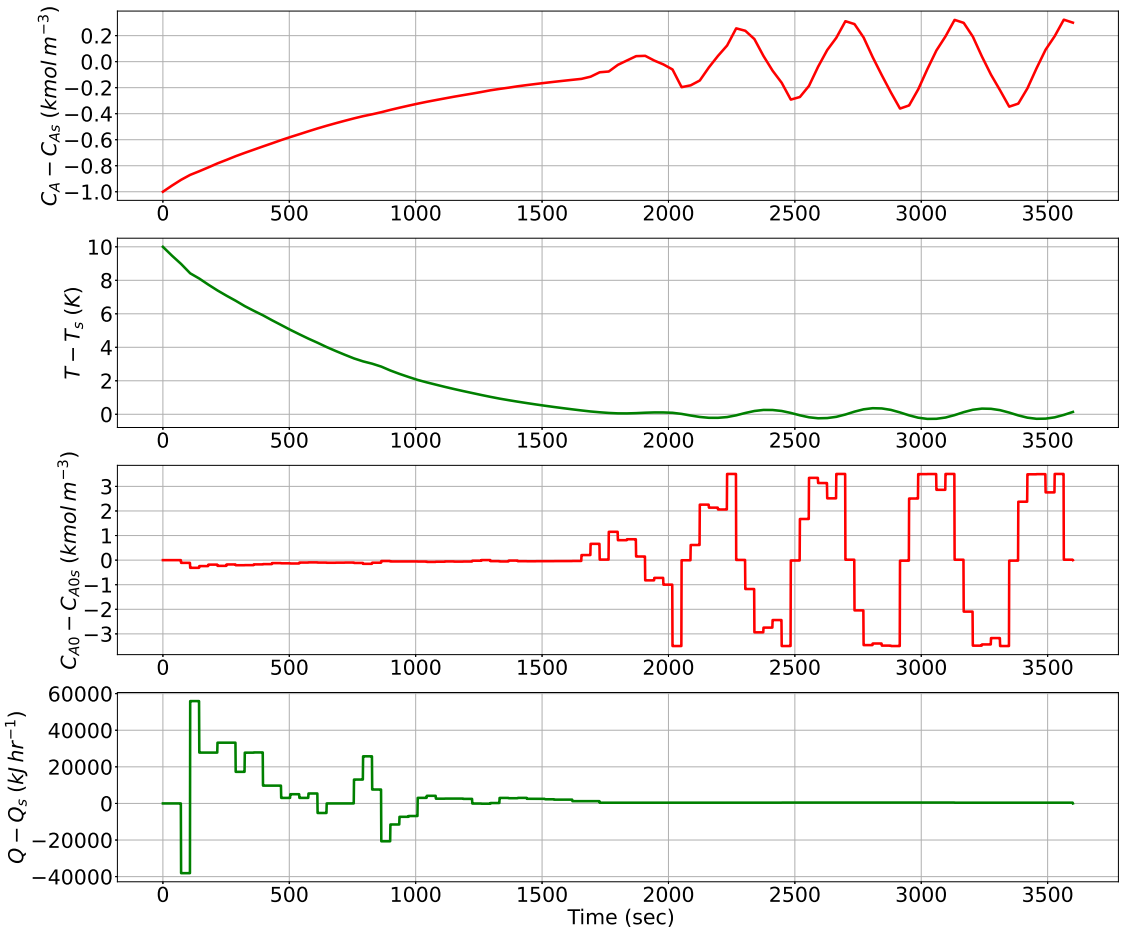

Figure 2.6: Closed-loop state and input trajectories under LSTM-based LMPC with time delays $d_1 = 0.01\ h$ and $d_2 = 0.02\ h$.

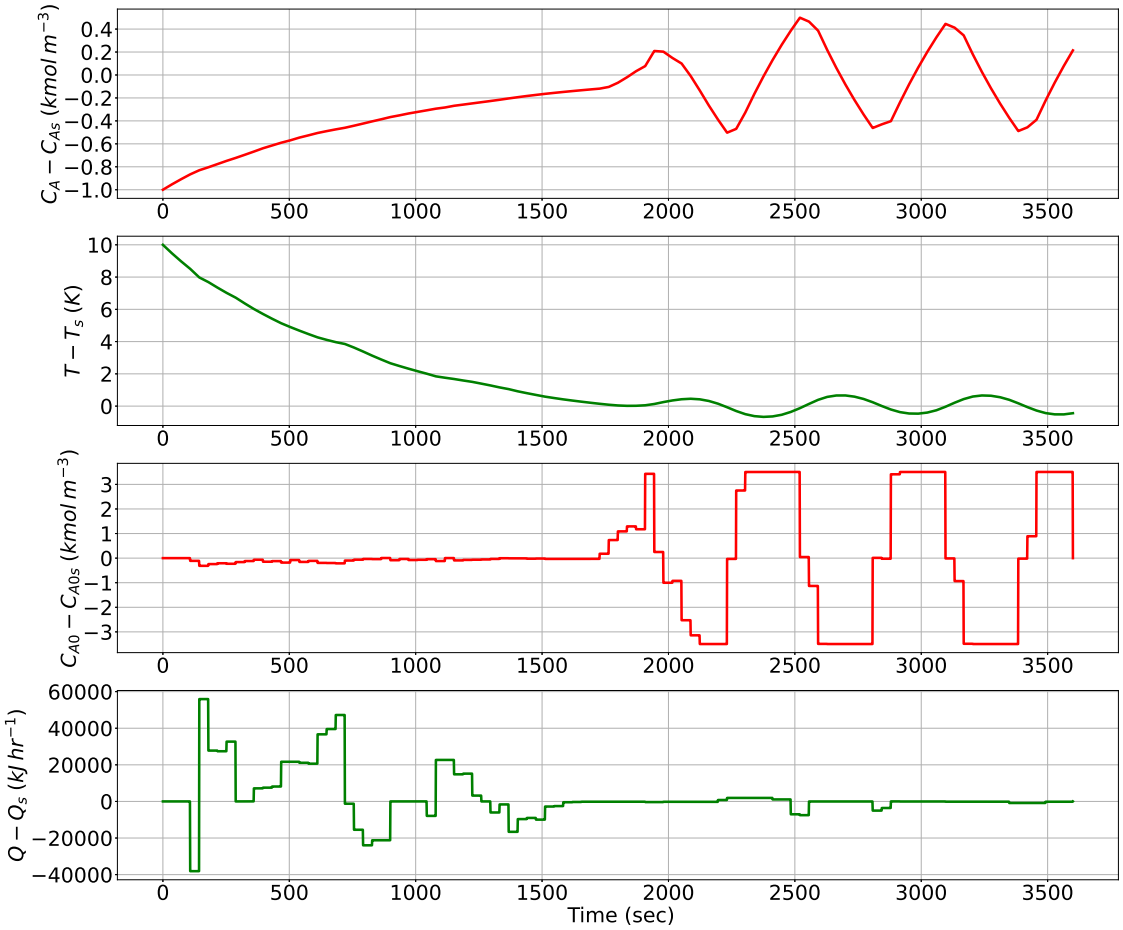

Figure 2.7: The closed-loop trajectories of the CSTR under LSTM-based LMPC with time delays: $d_1 = 0.01\ h$ and $d_2 = 0.03\ h$.

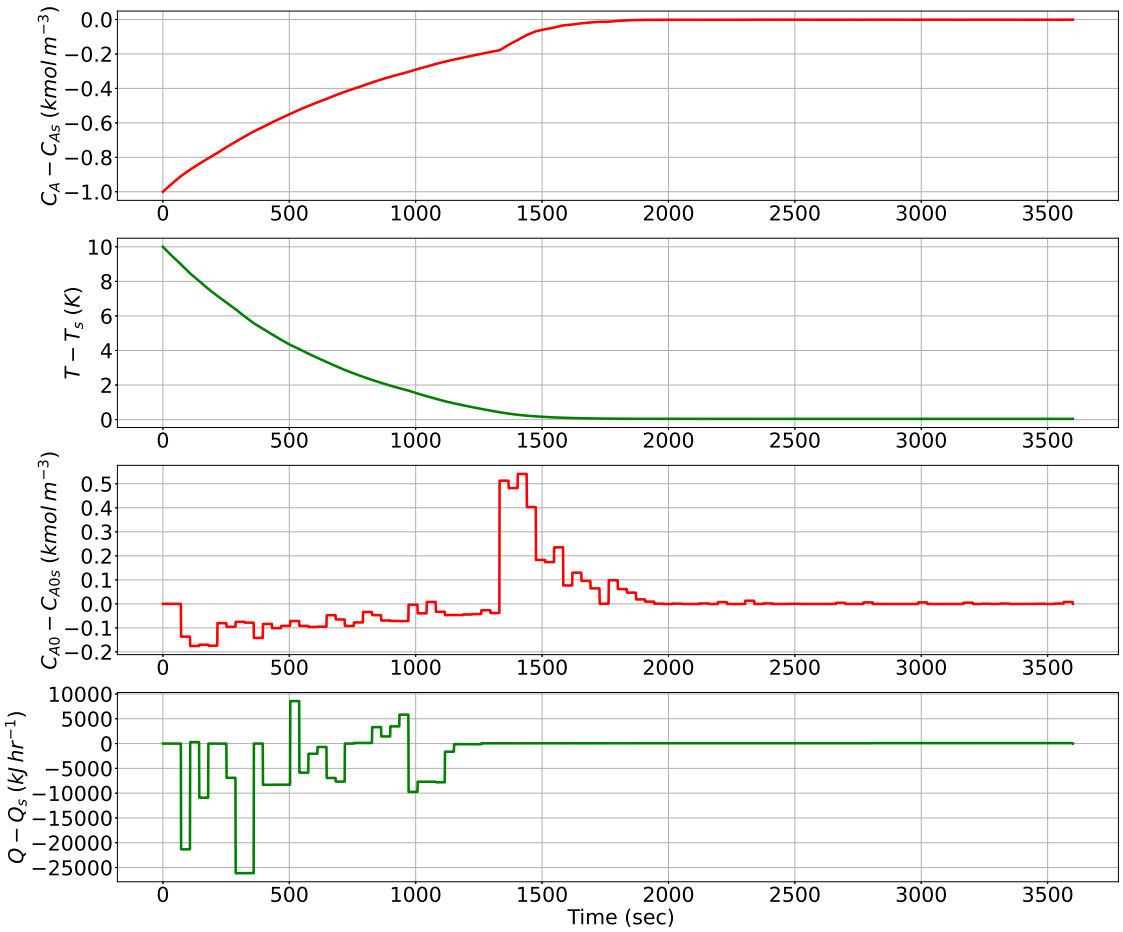

Figure 2.8: Closed-loop state and input trajectories under the predictor feedback LSTM-based LMPC, where the time delays: $d_1 = 0.01\ h$ and $d_2 = 0.02\ h$.

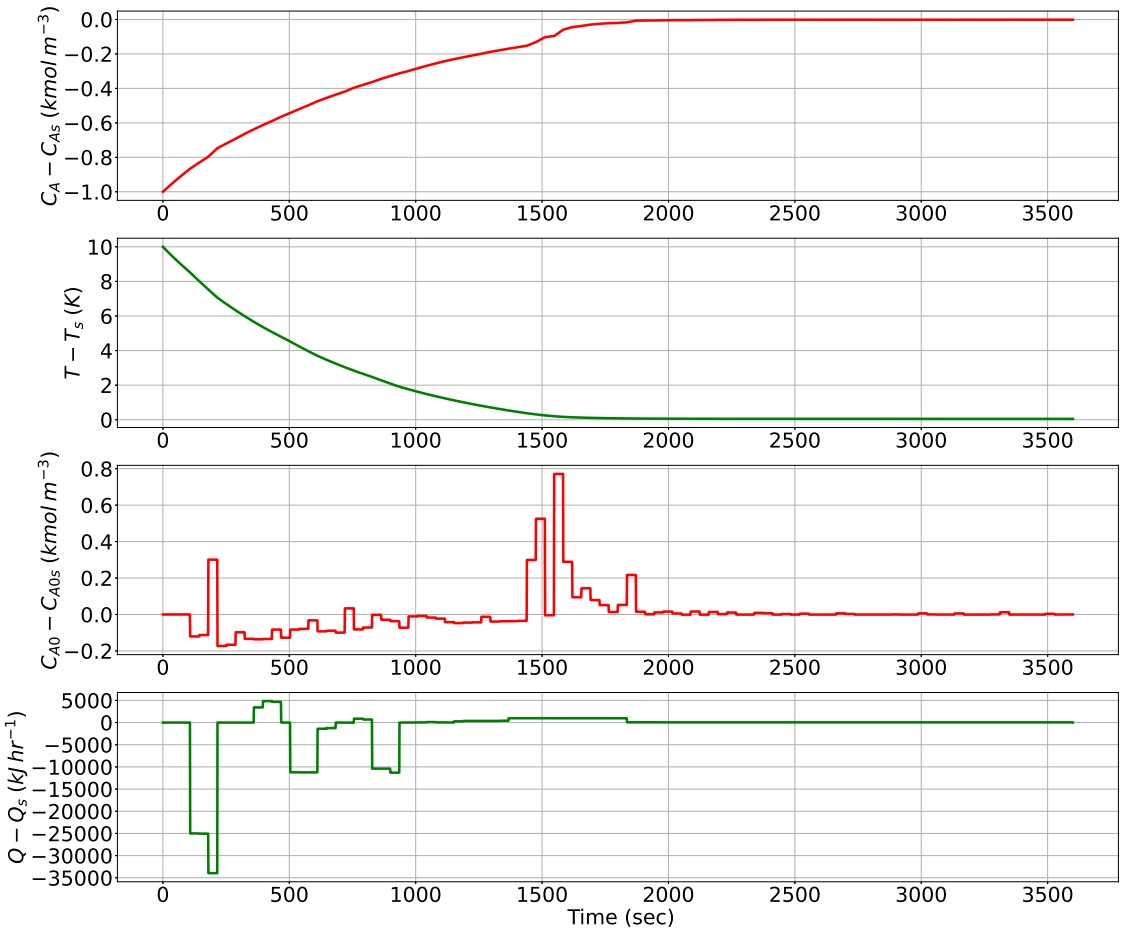

Figure 2.9: Closed-loop state and input trajectories under the predictor feedback LSTM-based LMPC, where the time delays: $d_1 = 0.01\ h$ and $d_2 = 0.03\ h$.

Chapter 3

Statistical Machine-Learning-based Predictive Control of Uncertain Nonlinear Processes

3.1 Introduction

Machine learning has gained increasing attention in modeling nonlinear systems due to powerful learning strategies, the availability of big data sets, and the development of computing resources. While the training error of machine learning models could be sufficiently low with good-quality datasets and a careful tuning of model hyper-parameters, identifying the generalization error for machine learning models on unseen data remains challenging. Generalization error bound provides an efficient way to measure the effectiveness of training and accuracy of machine learning models. The generalization error bound relies on various factors, including data sample size, bounds of weight matrices, and the number of neurons and layers. Many recent works have been done to obtain the generalization error bounds for

the implementation of neural networks in classification problems with single output [46, 71–73]. Additionally, in [74], a margin-based multi-class generalization bound was derived for the neural networks based on their margin-normalized spectral complexity. In [21, 47], the generalization error for RNNs was developed for multiclass classification problems, and regression problems of multi-input and multi-output (MIMO) nonlinear systems, respectively.

Additionally, model predictive controllers (MPC) using machine learning models have been studied in recent years, with successful applications to a number of chemical engineering problems [48, 75, 76]. As machine learning models can capture complex process dynamics, machine-learning-based MPCs have demonstrated their superior closed-loop performance when compared with the MPCs using (usually linear) data-driven models in traditional industrial process control systems. However, machine learning models are typically approximations of the nominal system dynamics, and thus, how to deal with uncertainty in processes within machine-learning-based MPCs is an important issue that requires further study.

Motivated by the above considerations, we develop RNN-based MPC schemes for nonlinear systems with model uncertainty in this manuscript. While MPC of stochastic nonlinear systems has been studied in literature, for example [77–81], RNN-based MPC of stochastic nonlinear systems is still in its infancy. In this work, we perform a probabilistic closed-loop stability analysis for the nonlinear systems subject to two common types of disturbances (unknown-but-bounded disturbances and stochastic disturbances with unbounded variation) under RNN-MPC based on the generalization error bound derived for RNN models. The theoretical study also provides a guidance showing how to improve machine learning models in a systematic way in order to achieve desired accuracy in both open-loop and closed-loop

simulations. The rest of this chapter is organized as follows: in Section "Preliminaries", the notations, the nonlinear systems and the recurrent neural network formulation are presented. In Section "RNN Generalization Error", a generalization error bound is derived for RNNs through Rademacher complexity approach. In Section "Probabilistic stability analysis", closed-loop stability results are developed for the nonlinear systems subject to bounded, and unbounded, stochastic disturbances, respectively. Finally, in Section "Application to a chemical process example", we use a chemical reactor as an example to illustrate the relation between training sample size and the RNN generalization error as well as the probability of closed-loop system stability.

3.2 Preliminaries

3.2.1 Notation

The transpose of $\mathbf{x}$ is denoted by $\mathbf{x}^T$. The Lie derivative is $L_f V(\mathbf{x}) := \frac{\partial V(\mathbf{x})}{\partial x} f(\mathbf{x})$. The operator $|\cdot|$ denotes the Euclidean norm of a vector. $|\cdot|_Q$ denotes the weighted Euclidean norm of a vector, where Q is a positive definite matrix. The Frobenius norm of A is denoted by $\|A\|_F$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. Given a set $\mathcal{D}$, the boundary of $\mathcal{D}$ is denoted by by $\partial\mathcal{D}$, and the interior of $\mathcal{D}$ is denoted by $\mathcal{D}^o$. The first hit time (or the hitting time) of a set X is defined as the first time that the state trajectory hits the boundary of X, and is denoted by τ_X. Also, we define $\tau_X(t) = \min\{\tau_X, t\}$ and $\tau_{X,T}(t) = \min\{\tau_X, T, t\}$, where T is the operation time.

$\mathbf{R}_+$ represents nonnegative real numbers. A function $f(x)$ belongs to class C^k if for all

$i = 1, 2, ..., k$, the ith derivative of f exists and is continuous. A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is L-Lipschitz continuous, if for all $a, b \in \mathbf{R}^n$, $|f(a) - f(b)| \leq L|a - b|$ holds, $L \geq 0$. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to a class $\mathcal{K}$ function if it is zero only when evaluated at zero, and is strictly increasing. $\mathbb{E}[X]$ is the expected value of a random variable X, and $\mathbb{P}(A)$ is the probability of the event A occurring.

3.2.2 Class of Systems

The following state-space model represents the class of continuous-time nonlinear systems considered in this work:

$$\dot{x} = F(x, u) := f(x) + g(x)u, \; x(t_0) = x_0 \tag{3.1}$$

where the n-dimensional state vector is denoted by $x \in \mathbf{R}^n$, and $u \in \mathbf{R}^k$ denotes the k-dimensional manipulated input vector bounded by $u \in U$. The set U defines the maximum $u_{\max}$ and the minimum value $u_{\min}$ for input vectors, i.e., $U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^k$. The vector $f(\cdot)$ and the matrix functions $g(\cdot)$ are sufficiently smooth with dimensions $n \times 1$, and $n \times k$, respectively. We assume that $f(0) = 0$ without loss of generality, and therefore, the origin is a steady-state of Eq. 3.1. Additionally, we assume that $t_0 = 0$ (i.e., the initial time is zero).

We assume that there exists a feedback controller $u = \Phi(x) \in U$ under which the origin can be rendered exponentially stable. The stabilizability assumption implies that there is a

$\mathcal{C}^1$ Lyapunov function $V(x)$ such that for all x in D the following inequalities hold:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{3.2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3|x|^2, \tag{3.2b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq c_4|x| \tag{3.2c}$$

where D is an open neighborhood around the origin, and c_i, $i = 1, 2, 3, 4$ are positive constants. We follow the method in [48] to generate the data by carrying out extensive open-loop simulation for the system of Eq. 3.1 with various inputs $u \in U$ and initial conditions x_0 to develop a set of time-series data for$x \in \Omega_\rho$, where Ω_ρ is a level set of Lyapunov function (i.e., $\Omega_\rho := \{x \in \mathbf{R}^n \mid V(x) \leq \rho\}$, $\rho > 0$) utilized as the operating region. Then, we develop recurrent neural network (RNN) models for capturing system dynamics and predicting state evolution. Specifically, the RNN models predict future states $x(t)$, $t > t_k$ based on the current state measurements $x(t_k)$ at time $t = t_k$, and the manipulated inputs $u(t)$, $t > t_k$.

3.2.3 Recurrent Neural Networks

In this section, we consider a general RNN model developed with m sequences of data $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$, where $\mathbf{y}_{i,t} \in \mathbf{R}^{d_y}$ is the RNN output, and $\mathbf{x}_{i,t} \in \mathbf{R}^{d_x}$, $i = 1, ..., m$ and $t = 1, ..., T$ (T is the time length) is the RNN input, to capture the system dynamics of Eq. 3.1. The RNN input/state/output vectors are written in boldface to differentiate the notations from those for the nonlinear system of Eq. 3.1. Additionally, to simplify the discussion, we develop the RNN model of Eqs. 3.3-3.4 to predict future states for one sampling period (denoted by

Δ) with internal steps $T = \frac{\Delta}{h_c}$, where h_c is the integration time step used by the Explicit Euler method to solve the continuous-time system of Eq. 3.1, and Δ is the sampling period within which the control action $u(t)$ remains unchanged (i.e., for all $t = 1, ..., T$). The RNN model predicts one sampling period forward, including all the internal states every h_c time step. As a result, for $t = 1, ..., T$, the predicted states are the RNN output $\mathbf{y}_{i,t}$, and the inputs and the current state measurements are RNN input $\mathbf{x}_{i,t}$.

The time-series data is generated independently following the data distribution over $\mathbf{R}^{d_x \times T} \times \mathbf{R}^{d_y \times T}$. Then, we develop the dataset by generating m data sequences of the same distribution. To simplify the discussion, a one-hidden-layer RNN (Fig. 3.1) is considered. The RNN states in the hidden layers $\mathbf{h}_i \in \mathbf{R}^{d_h}$ are

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) \tag{3.3}$$

where the weight matrices $W \in \mathbf{R}^{d_h \times d_x}$ and $U \in \mathbf{R}^{d_h \times d_h}$ are associated with the input and hidden state vectors, respectively. The element-wise nonlinear activation function is denoted by σ_h (e.g., ReLU). The output layer $\mathbf{y}_{i,t}$ is calculated using the following equation:

$$\mathbf{y}_{i,t} = \sigma_y(V\mathbf{h}_{i,t}) \tag{3.4}$$

where the activation function σ_y and the weight matrix $V \in \mathbf{R}^{d_y \times d_h}$ are associated with the output layer.

We have the following standard assumptions on the RNN model and datasets.

Assumption 3.1. *The RNN inputs are bounded, i.e.,* $\left|\mathbf{x}_{i,t}\right| \leq B_X$, *for all* $i = 1, ..., m$ *and*

$t = 1, ..., T$.

Assumption 3.2. *The Frobenius norms of the weight matrices are bounded as follows:*

$$\|W\|_F \leq B_{W,F}, \|V\|_F \leq B_{V,F}, \|U\|_F \leq B_{U,F} \tag{3.5}$$

Assumption 3.3. *All the datasets (i.e., training, validation, and testing) are drawn from the same distribution.*

Assumption 3.4. *σ_h is a 1-Lipschitz continuous activation function, and is positive-homogeneous in the sense that $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ holds for all $\alpha \geq 0$ and $z \in \mathbf{R}$.*

Remark 3.1. *Assumptions 3.1-3.4 are standard assumptions in machine learning theory. Specifically, Assumptions 3.1-3.2 assume the boundedness of RNN inputs and weight matrices. This is consistent with the practical implementation of RNN training that only a finite class of RNN hypotheses are searched for the optimal solution. Assumption 3.3 is also a basic assumption that is widely adopted in machine learning modeling works. The RNN models trained from the process operational data will be tested against data that come from the same target distribution. Assumption 3.4 requires positive-homogeneity of the activation function. For example, Rectified Linear Unit (ReLu), a popular nonlinear activation function in the machine learning domain, is a candidate of activation function that meets this assumption.*

3.3 RNN Generalization Error

The RNN learning algorithms provide no information on the generalization performance for unseen testing data since they are evaluated on training data only. Therefore, the generalization error is used to measure the neural network's predictive capability for any data not utilized in training. Specifically, an upper bound is developed in this section for the RNN

generalization error. Then we demonstrate that with high probability, this error is bounded if the development of RNN models meets a few requirements.

3.3.1 Preliminaries

Let $\mathcal{H}$ be the hypothesis class of RNN functions $h(\cdot)$ that map a d_x-dimensional input $\mathbf{x} \in \mathbf{R}^{d_x}$ to a d_y-dimensional output $\mathbf{y} \in \mathbf{R}^{d_y}$. The predicted output of the RNN model and the loss function are denoted by $\mathbf{y}_t = h(\mathbf{x}_t)$ and $L(\mathbf{y}_t, \bar{\mathbf{y}}_t)$, respectively, where $L(\mathbf{y}, \bar{\mathbf{y}})$ calculates the squared difference between the predicted output $\mathbf{y}$ and the true output $\bar{\mathbf{y}}$. We have the following error definitions for training RNN models.

Definition 3.1. *Given a data distribution D, and a function h that predicts y (output) based on x (input), the* ***generalization error*** *or* ***expected loss / error*** *is*

$$\mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y)\rho(x, y)dxdy \tag{3.6}$$

where the joint probability distribution for x and y is represented as $\rho(x, y)$, and the vector space for all possible outputs and inputs are denoted by Y and X, respectively.

Since in most cases ρ is an unknown distribution, we utilize empirical error as an approximation measure for the expected error. The empirical error is calculated as follows.

Definition 3.2. *Given a dataset $S = (s_1, ..., s_m)$, $s_i = (x_i, y_i)$, with m data samples drawn from the data distribution D, the* ***empirical risk*** *or* ***error*** *is*

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i) \tag{3.7}$$

Since the RNN data is generated within a compact set, the RNN predicted outputs $\mathbf{y}_t$ and the true outputs $\bar{\mathbf{y}}_t$ are assumed to be bounded by $r_t > 0$, $t = 1, ..., T$, i.e., $|\mathbf{y}_t|, |\bar{\mathbf{y}}_t| \leq r_t$. Thus, the following inequality from local Lipschitz continuity holds for the loss function of mean squared error (MSE) for all $|\mathbf{y}_t| \leq r_t$, and $|\bar{\mathbf{y}}_t| \leq r_t$.

$$|L(\mathbf{y}_2, \bar{\mathbf{y}}) - L(\mathbf{y}_1, \bar{\mathbf{y}})| \leq L_r |\mathbf{y}_2 - \mathbf{y}_1| \tag{3.8}$$

where the local Lipschitz constant is denoted by L_r.

3.3.2 Rademacher Complexity

Rademacher complexity is used to quantify the richness of a function class in computational learning theory. The definition of empirical Rademacher Complexity is presented below.

Definition 3.3. *Given a set of data samples $S = \{s_1, ..., s_m\}$, and a hypothesis class $\mathcal{F}$ of real-valued functions, the definition of empirical Rademacher complexity of $\mathcal{F}$ is*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i f(s_i) \right] \tag{3.9}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_m)^T$, and ϵ_i are Rademacher random variables that are independent and identically distributed (i.i.d.) and satisfy $\mathbb{P}(\epsilon_i = -1) = \mathbb{P}(\epsilon_i = 1) = 0.5$.

The Rademacher complexity is used to derive the generalization error bound in the following lemma.

Lemma 3.1 (c.f. Theorem 3.3 in [82]). *Let $\mathcal{H}$ be the hypothesis class that maps $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_t\} \in \mathbf{R}^{d_x \times t}$ (i.e., the first t-time-step inputs) to $\boldsymbol{y}_t \in \mathbf{R}^{d_y}$ (i.e., the t-th output), and $\mathcal{G}_t$ be loss function set with $\mathcal{H}$.*

$$\mathcal{G}_t = \{g_t : (\boldsymbol{x}, \bar{\boldsymbol{y}}) \rightarrow L(h(\boldsymbol{x}), \bar{\boldsymbol{y}}), h \in \mathcal{H}\} \tag{3.10}$$

where $\bar{\boldsymbol{y}}$ and $\boldsymbol{x}$ are the true output vector and the RNN input vector, respectively. Then, given a dataset consisting of m i.i.d. data samples, the inequality below holds in probability for all $g_t \in \mathcal{G}_t$ over the data samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$:

$$\mathbb{E}[g_t(\boldsymbol{x}, \boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{3.11}$$

The RHS of Eq. 3.11 represents the upper bound for the generalization error, which relies on various factors. Specifically, the first term of the RHS in Eq. 3.11 represents the empirical risk, the second term represents the Rademacher complexity, and an error function of the samples size m and the confidence δ is represented in the last term. Note that the last and the first terms can be computed once a set of training data of size m and the confidence δ are given. Therefore, our goal is to derive the Rademacher complexity bound for $\mathcal{R}_S(\mathcal{G}_t)$.

3.3.3 Generalization Error Bound

We first present a few lemmas to provide preliminary results following the proof technique in [21].

Lemma 3.2 (c.f. Lemma 4 in [21]). *Given a dataset $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, of m i.i.d. data samples, $i = 1, ..., m$, and a real-valued function class $\mathcal{H}_k$ that corresponds to the k-th com-*

ponent of the class $\mathcal{H}$ of vector-valued functions, the scaled empirical Rademacher complexity $m\mathcal{R}_S(\mathcal{H}_k) = \mathbb{E}[\sup_{h\in\mathcal{H}_k}\sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i)]$ satisfies the the following inequality:

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_k) &= \frac{1}{\lambda}\log\exp\left(\lambda\mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i)\right]\right) \\ &\leq \frac{1}{\lambda}\log\left(\mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\exp(\lambda\sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i))\right]\right) \end{aligned} \tag{3.12}$$

where λ is an arbitrary positive real number.

Additionally, since the RNN models of Eqs. 3.3-3.4 are essentially complex nonlinear functions that are difficult to measure the learning capacity, the following lemma provides a useful tool to peel off RNN weights and nonlinear activation functions through layers.

Lemma 3.3 (c.f. Lemma 6 in [21])**.** *Given any monotonically increasing and convex function $p : \mathbf{R} \to \mathbf{R}_+$, and a vector-valued RNN function class $\mathcal{H}$ with a positive-homogeneous, 1-Lipschitz, activation function $\sigma_h(\cdot)$, the inequality below holds:*

$$\begin{aligned} &\mathbb{E}\left[\sup_{||W||_F\leq B_{W,F}, ||U||_F\leq B_{U,F}, h\in\mathcal{H}} p\left(\left|\sum_{i=1}^m \epsilon_i \boldsymbol{h}_{i,t}\right|\right)\right] \\ &\leq 2\mathbb{E}\left[\sup_{h\in\mathcal{H}} p\left(B_{W,F}\left|\sum_{i=1}^m \epsilon_i \boldsymbol{x}_{i,t}\right| + B_{U,F}\left|\sum_{i=1}^m \epsilon_i \boldsymbol{h}_{i,t-1}\right|\right)\right] \end{aligned} \tag{3.13}$$

Based on Lemma 3.3, the following lemma derives a Rademacher complexity bound for the real-valued RNN function class $\mathcal{H}_k$ that corresponds to the k-th output of the class $\mathcal{H}$ of vector-valued functions.

Lemma 3.4 (c.f. Lemma 7 in [21]). *Given a dataset $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, of m i.i.d. data samples, $i = 1, ..., m$, and a class of real-valued functions, $\mathcal{H}_{k,t}$, $k = 1, ..., d_y$, that corresponds to the k-th output at t-th time step, and satisfy Assumptions 3.1-3.4, the Rademacher complexity can be bounded using the following inequality:*

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \leq \frac{MB_X(1 + \sqrt{2\log(2)t})}{\sqrt{m}} \tag{3.14}$$

where $M = \frac{1-(B_{U,F})^t}{1-B_{U,F}} B_{W,F} B_{V,F}$.

Finally, we consider the class of loss function for the vector-valued RNN models, and use the contraction inequality in [83] to further bound the RNN generalization error as follows.

Theorem 3.1 (c.f. Theorem 1 in [21]). *Given a dataset $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$ with i.i.d. data samples, $i = 1, ..., m$, and the loss function class $\mathcal{G}_t$ associated with the RNN function class $\mathcal{H}_t$ that predicts outputs at the t-th time step, with probability at least $1 - \delta$ over S, the following inequality holds for the RNN models with the activation functions and weight matrices that satisfy Assumptions 3.1-3.4.*

$$\mathbb{E}[g_t(\boldsymbol{x}, \boldsymbol{y})] \leq \mathcal{O}\left(L_r d_y \frac{MB_X(1 + \sqrt{2\log(2)t})}{\sqrt{m}} \right) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{3.15}$$

where M is given in Eq. 3.14, B_X is the RNN input bound defined in Eq. 3.1, L_r is the local Lipschitz constant defined in Eq. 3.8, and d_y is the RNN output dimension.

3.4 Probabilistic Stability Analysis

The RNN models are incorporated within MPC in this section to provide the prediction of future states. We develop the MPC scheme using RNN models (RNN-MPC), and study the system stability properties for the nonlinear system of Eq. 3.1 subject to disturbances. We demonstrate that under RNN-MPC, the state of the closed-loop system remains inside the stability region in probability for all times in the presence of process disturbances.

The single-hidden-layer RNN model is represented as a continuous-time nonlinear system for simplifying the analysis of its stability properties [48]:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T z \tag{3.16}$$

where $u \in \mathbf{R}^k$ is the manipulated input, and $\hat{x} \in \mathbf{R}^n$ is the RNN state vector. A and Θ are the weight matrices, and z is a vector associated with the $\hat{x}$ and u. The readers are referred to [21] for the details of Eq. 3.16. In the following sections, the Lyapunov-based MPC schemes using RNN models are designed to stabilize the nonlinear system in a probabilistic manner in the presence of process disturbances. Specifically, we consider two types of disturbances: unknown-but-bounded disturbances and stochastic disturbances with unbounded variation, and establish the probabilistic closed-loop stability results for the nonlinear systems under RNN-MPC.

3.4.1 Nonlinear systems with bounded disturbances

The class of continuous-time nonlinear systems subject to bounded disturbances is described by the following ordinary differential equation:

$$\dot{x} = F(x, u, w) := f(x) + g(x)u + h(x)w,\ x(t_0) = x_0 \tag{3.17}$$

where the notations are the same as those in Eq. 3.1. The disturbance vector w is bounded by $W := \{w \in \mathbf{R}^q \mid |w| \leq w_m,\ w_m \geq 0\}$. $h(x)$ is a sufficiently smooth matrix function of dimension $n \times q$. Based on the boundedness of x, u and w, and the Lipschitz property of $F(x, u, w)$, there exist positive constants M_F, $L_x, L_x^{'}$, $L_w, L_w^{'}$ such that for all $w \in W$, $u \in U$, $x, x' \in \Omega_\rho$, the following inequalities hold:

$$|F(x', u, 0) - F(x, u, w)| \leq L_w|w| + L_x|x - x'| \tag{3.18a}$$

$$\left|\frac{\partial V(x')}{\partial x}F(x', u, 0) - \frac{\partial V(x)}{\partial x}F(x, u, w)\right| \leq L_w^{'}|w| + L_x^{'}|x - x'| \tag{3.18b}$$

$$|F(x, u, w)| \leq M_F \tag{3.18c}$$

Under the assumption of exponential stabilization of the origin of the RNN model of Eq. 3.16 for states in an open set $\hat{D}$ around the origin by a feedback controller $u = \Phi_{nn}(x) \in U$, a $\mathcal{C}^1$ Lyapunov function $\hat{V}(x)$ can be found such that for all states x in $\hat{D}$, the following inequalities hold:

$$\left|\frac{\partial \hat{V}(x)}{\partial x}\right| \leq \hat{c}_4|x|, \tag{3.19a}$$

$$\frac{\partial \hat{V}(x)}{\partial x}F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{c}_3|x|^2, \tag{3.19b}$$

$$\hat{c}_1|x|^2 \leq \hat{V}(x) \leq \hat{c}_2|x|^2 \tag{3.19c}$$

where $\hat{c}_i$, $i = 1, 2, 3, 4$ are positive constants. Similarly, we characterize the stability region for the RNN model of Eq. 3.16 as a compact set embedded in $\hat{D}$ as follows: $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. The following proposition demonstrates that for the RNN model developed with a sufficiently small modeling error, the nominal system of Eq. 3.17 with $w(t) \equiv 0$ can be stabilized under $u = \Phi_{nn}(x) \in U$ with high probability.

Proposition 3.1. *Consider the nominal system of Eq. 3.17 with $w(t) \equiv 0$ and an RNN model that is trained with m i.i.d. data samples and satisfies the conditions in Theorem 3.1. Under the stabilization assumption of Eq. 3.19, if there exists a positive real number γ that satisfies $\gamma < \hat{c}_3/\hat{c}_4$, and constrains the modeling error, i.e., $|F_{nn}(x, u) - F(x, u, 0)| \leq \gamma|x|$, for all $u \in U$ and $x \in \Omega_{\hat{\rho}}$, then for all $x \in \Omega_{\hat{\rho}}$, the origin of the nominal system is rendered exponentially stable with probability at least $1 - \delta$ under $u = \Phi_{nn}(x) \in U$.*

Proof. Following the proof of Proposition 2 in [48], the time derivative of $\hat{V}$ is obtained as follows using Eq. 3.19b and Eq. 3.19a:

$$\begin{aligned} \dot{\hat{V}} =& \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{nn}(x), 0) \\ \leq& \hat{c}_4|x| \cdot |F(x, \Phi_{nn}(x), 0) - F_{nn}(x, \Phi_{nn}(x))| - \hat{c}_3|x|^2 \end{aligned} \tag{3.20}$$

where the term $|F(x, \Phi_{nn}(x), 0) - F_{nn}(x, \Phi_{nn}(x))|$ is the modeling mismatch between the nominal system of Eq. 3.17 and the RNN model. Using the generalization error results

derived in Theorem 3.1, we have the following bound for the modeling error:

$$|F_{nn}(x, \Phi_{nn}(x)) - F(x, \Phi_{nn}(x), 0)| \leq E_M \tag{3.21}$$

where E_M represents the generalization error that is bounded by the RHS of Eq. 3.15.

Since the generalization error bound of Eq. 3.15 depends on the training sample size, we can find the minimum data sample size $m_N(|x|, h_c, \delta)$ such that the modeling error is upper bounded by $E_M \leq \gamma|x|$. Therefore, by choosing the sample size $m \geq m_N(|x|, h_c, \delta)$, the following equation holds for $\dot{\hat{V}}$, with probability no less than $1 - \delta$.

$$\begin{aligned}
\dot{\hat{V}} =& \frac{\partial \hat{V}}{\partial x}(F_{nn}(x, \Phi_{nn}(x)) + F(x, \Phi_{nn}(x), 0) - F_{nn}(x, \Phi_{nn}(x))) \\
\leq& - \hat{c}_3|x|^2 + |F_{nn}(x, \Phi_{nn}(x)) - F(x, \Phi_{nn}(x), 0)| \cdot \hat{c}_4|x| \\
\leq& - \hat{c}_3|x|^2 + \frac{\hat{c}_3|x|}{\hat{c}_4} \cdot \hat{c}_4|x| \\
=& - \tilde{c}_3|x|^2 \\
<& 0
\end{aligned} \tag{3.22}$$

where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma < 0$ for any $\gamma < \hat{c}_3/\hat{c}_4$. This implies that with a certain probability, $\dot{\hat{V}}$ can be rendered negative (i.e., $\mathbb{P}[\dot{\hat{V}} < 0] \geq 1 - \delta$), and therefore, the state of the nominal system of Eq. 3.17 moves towards the origin under $u = \Phi_{nn}(x) \in U$ for all $x_0 \in \Omega_{\hat{\rho}}$. □

Remark 3.2. *Note that the minimum data sample size $m_N(|x|, h_c, \delta)$ that satisfies $E_M \leq \gamma|x|$ is a function of $|x|$, h_c and δ. Specifically, by substituting the RHS of Eq. 3.15 into E_M, it is straightforward to show that the solution to $E_M \leq \gamma|x|$ is a function of the confidence level δ. Additionally, it is observed from $E_M \leq \gamma|x|$ that the solution depends on the value of x in the way that a smaller x value (i.e., the states closer to the origin) leads to a*

tighter generalization error bound E_M, which requires more data to be used for training. In the extreme case where the state x is sufficiently close to the origin, a large number of data is needed to render E_M sufficiently low in order to meet the condition $E_M \leq \gamma|x|$, which is computationally impracticable in general. Therefore, to reduce the computational time in training RNN models, we do not require this condition to hold in a small neighborhood around the origin, and we will demonstrate in Theorem 3.2 that closed-loop stability remains unaffected under RNN-based MPC despite this loose condition. Lastly, $m_N(\cdot)$ is also a function of h_c since we calculate the modeling error of Eq. 3.21 by approximating the derivatives using finite differences with a sufficiently small time step h_c.

Proposition 3.2. *Consider the RNN model $\dot{\hat{x}} = F_{nn}(\hat{x}, u)$ of Eq. 3.16 developed satisfying Assumptions 3.1-3.4 and the nonlinear system $\dot{x} = F(x, u, w)$ of Eq. 3.17 with bounded disturbances $|w| \leq w_m$. There exists a positive constant κ and a class $\mathcal{K}$ function $f_w(\cdot)$ such that for all $x, \hat{x} \in \Omega_{\hat{\rho}}$, with probability at least $1 - \delta$, the following inequalities hold with $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$ (i.e., the same initial condition):*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + E_M}{L_x}(e^{L_x t} - 1) \tag{3.23a}$$

$$\hat{V}(x) \leq \kappa|x - \hat{x}|^2 + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \hat{V}(\hat{x}) \tag{3.23b}$$

Proof. The results are derived following the proof technique used in Proposition 3 in [48]. Note that in [48], the modeling error is assumed to be bounded by a constant number almost surely (i.e., with probability 1). However, the modeling error $|F(\hat{x}, u, 0) - F_{nn}(x, u)|$ in this work is bounded by E_M with probability at least $1 - \delta$ since we use the generalization error to represent the model mismatch for any states in the stability region including those which are not used in training. The key steps for the proof are presented below. We first define an

error vector $e(t) = x(t) - \hat{x}(t)$ that represents the modeling error between the RNN model state $\dot{\hat{x}} = F_{nn}(\hat{x}, u)$ and the actual nonlinear system state $\dot{x} = F(x, u, w)$ subject to bounded disturbances. The time derivative of error vector is bounded for all $\hat{x}, x \in \Omega_{\hat{\rho}}$, $w(t) \in W$ and $u \in U$ as follows.

$$\begin{aligned} |\dot{e}| &= |F(x, u, w) - F_{nn}(\hat{x}, u)| \\ &\leq |F(x, u, w) - F(\hat{x}, u, 0)| + |F(\hat{x}, u, 0) - F_{nn}(\hat{x}, u)| \\ &\leq L_x|e(t)| + L_w w_m + E_M \end{aligned} \tag{3.24}$$

where the last inequality is obtained using Eq. 3.18 and the modeling error constraint of Eq. 3.21. Since the initial error $e(0)$ is zero for the same initial condition $x_0 = \hat{x}_0$, the evolution of error vector is bounded as follows.

$$|e(t)| \leq \frac{L_w w_m + E_M}{L_x}(e^{L_x t} - 1) \tag{3.25}$$

Additionally, using Taylor series expansion and ignoring higher-order terms, we have

$$\begin{aligned} \hat{V}(x) &\leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \\ &\leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\bar{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \end{aligned} \tag{3.26}$$

where the last inequality is obtained using Eq. 3.19a, Eq. 3.19c, and κ is a positive real number. This completes the proof of Proposition 3.2. □

Subsequently, the RNN models are utilized within MPC to provide the prediction of future state evolution. The optimization problem of RNN-MPC is presented as follows [48,

49]:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t), u(t))dt \tag{3.27a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{3.27b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{3.27c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.27d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{3.27e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{3.27f}$$

where L_{MPC} denotes the objective function of MPC that attains its minimum value at the origin. $\tilde{x}$ is the state predicted by the RNN model $F_{nn}(x, u)$. The RNN-MPC of Eq. 3.27 is applied in a sample-and-hold fashion in which control actions remain unchanged within each sampling period, with control actions optimized over the prediction horizon from t_k to t_{k+N} as a piece-wise function in $S(\Delta)$. The goal of RNN-MPC is to stabilize the nonlinear system at the steady-state in the way that the state is maintained in the stability region $\Omega_{\hat{\rho}}$ at all times, and is driven into a small terminal set around the origin ultimately. Eq. 3.27b is the prediction model, and Eq. 3.27c is the input constraint. The feedback measurement of x at each sampling time is utilized as the initial state for solving Eq. 3.27b. The two Lyapunov-based constraints of Eqs. 3.27e-3.27f ensure stability for the closed-loop system under RNN-MPC. The following theorem establishes the closed-loop stability properties for the uncertain system of Eq. 3.17 subject to bounded disturbances $w \in W$ under RNN-MPC.

Theorem 3.2. *Consider the nonlinear system of Eq. 3.17 under the RNN-MPC of Eq. 3.27 with the controller* $\Phi_{nn}(x)$ *that meets Eq. 3.19. Let* $\Delta > 0$, $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ *and* $\epsilon_w > 0$ *satisfy Eq. 3.28 and 3.29.*

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} M_F \Delta + L_w^{'} w_m \leq -\epsilon_w \tag{3.28}$$

and

$$\rho_{nn} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid u \in U,\ \hat{x}(t) \in \Omega_{\rho_s}\} \tag{3.29a}$$

$$\rho_{min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{3.29b}$$

where $f_w(t) = \frac{E_M + L_w w_m}{L_x}(e^{L_x t} - 1)$. *Then, by choosing the sample size* m *to be greater than the minimum sample size* $m_N(|x|, h_c, \delta)$ *that satisfies* $E_M \leq \gamma|x|$, *for any initial state* $x_0 \in \Omega_{\hat{\rho}}$ *and for each sampling time step, system stability is achieved for the disturbed system of Eq. 3.17 with* $w \in W$ *with probability at least* $1 - \delta$, *under the RNN-MPC of Eq. 3.27 in the sense that* $x(t)$ *ultimately converges to* $\Omega_{\rho_{min}}$, *and is maintained in* $\Omega_{\hat{\rho}}$ *at all times, i.e.,* $x(t) \in \Omega_{\hat{\rho}},\ \forall t \geq 0$.

Proof. The proof follows the proofs of Proposition 3 and Theorem 2 in [21] for the nominal system of Eq. 3.17 with $w(t) \equiv 0$, and we present a proof sketch here to help readers understand the key steps. The main difference is that the system of Eq. 3.17 is subject to sufficiently small bounded disturbances $|w(t)| \leq w_m$ in this work, which needs to be accounted for in the controller design and stability analysis, while in [21] the results were developed for the nominal system of Eq. 3.17 with $w(t) \equiv 0$. We first obtain the time

derivative of $\hat{V}$ for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ under the controller $u = \Phi_{nn}(x) \in U$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) =& \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), w) \\ =& \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), w) \\ &- \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0)\end{aligned} \tag{3.30}$$

Using the results in Proposition 3.1 and Eq. 3.18, we can further bound Eq. 3.30 for $x(t) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, $u \in U$, and $w \in W$ as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) \leq& \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), w) - \frac{\tilde{c}_3}{\hat{c}_2}\rho_s - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0) \\ \leq& -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} M_F \Delta + L_w^{'} w_m\end{aligned} \tag{3.31}$$

Therefore, if Eq. 3.28 is satisfied, with probability at least $1-\delta$, $\dot{\hat{V}}(x(t))$ is rendered negative for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, which implies the convergence of state towards the terminal set, as well as the boundedness of state within $\Omega_{\hat{\rho}}$ in probability. Additionally, $\Omega_{\rho_{min}}$ and $\Omega_{\rho_{nn}}$ of Eq. 3.29 are the two small sets around the origin which contain Ω_{ρ_s} as their subsets (i.e., $\rho_{min} > \rho_{nn} > \rho_s$). Specifically, we do not require the modeling error to be bounded by E_M within Ω_{ρ_s}, and thus, Eq. 3.31 does not hold for the state in Ω_{ρ_s}. This loose condition of modeling error within Ω_{ρ_s} significantly reduces the computational complexity for training RNN models with the data sufficiently close to the origin. The set $\Omega_{\rho_{nn}}$ is characterized to ensure that for any state within Ω_{ρ_s}, the predicted state remains inside $\Omega_{\rho_{nn}}$ under any control actions within the bounds. As a result, the state of the actual nonlinear system of

Eq. 3.17 subject to disturbances $w(t) \in W$ is bounded in the set $\Omega_{\rho_{min}}$ that is characterized accounting for the modeling error and disturbances. □

3.4.2 Nonlinear systems with stochastic disturbances

In addition to the robustness treatment of the process disturbances as bounded uncertain variables, another approach to dealing with model uncertainty is to develop controllers that achieve stability in probability for the closed-loop system by modeling the disturbance terms in a probabilistic manner and taking the distribution information of disturbances into account. Specifically, we consider the nonlinear system with stochastic disturbances in the form of the following stochastic differential equation (SDE):

$$dx(t) = f(x(t))dt + g(x(t))u(t)dt + h(x(t))dw(t) \tag{3.32}$$

where the notations follows those in Eq. 3.1. The disturbance vector $w(t)$ is represented by a standard Wiener process. The steady-state of the nominal system with $w(t) \equiv 0$ is assumed to be at the origin, i.e., $(x_s^*, u_s^*) = (0, 0)$. In Eq. 3.1, $f(x(t)) + g(x(t))u(t)$ and $h(x(t))$ are the deterministic drift and the diffusion matrix, respectively. $h(0)$ is assumed to be zero such that $h(x(t))dw(t)$ (i.e., the disturbance term) vanishes at the origin. Similarly, we assume that $L_g V(x)$, $L_f V(x)$, and $h(x)^T \frac{\partial^2 V(x)}{\partial x^2} h(x)$ are locally Lipschitz. For the system of Eq. 3.32, if for any $\epsilon > 0$, the following conditions hold, then the origin is asymptotically stable in

probability.

$$\lim_{x(0)\to 0} \mathbb{P}(\lim_{t\to\infty} |x(t)| = 0) = 1,$$
$$\lim_{x(0)\to 0} \mathbb{P}(\sup_{t\geq 0} |x(t)| > \epsilon) = 0 \tag{3.33}$$

It is assumed that a stochastic feedback controller $u = \Phi_s(x) \in U$ exists such that for all $x \in D_2 \subset \mathbf{R}^n$ (D_2 is an open neighborhood of the origin), exponential stabilization of the origin of the RNN model of Eq. 3.16 is achieved in probability. The stabilizability assumption implies that a C^2, positive definite stochastic Lyapunov function V exists and meets the following conditions:

$$\mathcal{L}V(x) = \frac{\partial V(x)}{\partial x} F_{nn}(x, \Phi_s(x)) + \frac{1}{2} Tr\{h^T \frac{\partial^2 V}{\partial x^2} h\} \leq -\alpha_1 |x|^2 \tag{3.34}$$

$$h(x)^T \frac{\partial^2 V}{\partial x^2} h(x) \geq 0 \tag{3.35}$$

where $\mathcal{L}V(x)$ represents the infinitesimal generator of the system of Eq. 3.32, and α_1 is a positive real number. $\phi_d = \{x \in \mathbf{R}^n \mid \mathcal{L}V + \kappa V(x) \leq 0,\ \kappa > 0,\ u = \Phi_s(x) \in U\}$ is characterized as a set of initial conditions from which the exponential stabilization of the origin of the RNN model of Eq. 3.16 can be achieved in probability using the controller $u = \Phi_s(x) \in U$. Subsequently, a level set of $V(x)$ inside ϕ_d, i.e., $\Omega_\rho := \{x \in \phi_d \mid V(x) \leq \rho\}$, $\rho > 0$, is chosen as the operating region.

Based on $u = \Phi_s(x) \in U$, the following Lyapunov-based MPC scheme is designed to stabilize the stochastic nonlinear system of Eq. 3.32, where the notations follow those in

Eq. 3.27.

$$\min_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t), u(t))\, dt \tag{3.36a}$$

$$\text{s.t. } \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{3.36b}$$

$$u(t) \in U, \quad \forall\, t \in [t_k, t_{k+N}) \tag{3.36c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.36d}$$

$$\mathcal{L}V(x(t_k), u(t_k)) \leq \mathcal{L}V(x(t_k), \Phi_s(x(t_k))), \text{ if } x(t_k) \in \Omega_\rho \backslash \Omega^{\text{o}}_{\rho_{nn}} \tag{3.36e}$$

$$V(\tilde{x}(t)) < \rho_{nn}, \ \forall t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega^{\text{o}}_{\rho_{nn}} \tag{3.36f}$$

Theorem 3.3 establishes the probabilistic stability properties for the uncertain system of Eq. 3.32 under the RNN-MPC of Eq. 3.36.

Theorem 3.3. *Consider the system of Eq. 3.32 under the MPC of Eq. 3.36 using RNN models that meet the Assumptions 3.1-3.4. By letting $m \geq m_N(|x|, h_c, \delta)$, given any initial condition $x(0) \in \Omega_\rho$, probability $\lambda \in (0, 1]$, and positive real numbers satisfying $\rho > \rho_{min} > \rho_{nn}$ and $\rho_c \in [\rho_{nn}, \rho]$, there exists a sampling period $\Delta > 0$ and probabilities $\beta, \gamma \in [0, 1]$ such that the following inequalities hold for $t \in [t_k, t_{k+1})$, $t_{k+1} := t_k + \Delta$.*

$$\mathbb{P}(\sup_{t\in[t_k,t_{k+1})} V(x(t)) < \rho_{min}) \geq (1-\delta)(1-\beta)(1-\lambda), \quad \forall x(t_k) \in \Omega_{\rho_{nn}} \tag{3.37}$$

$$\mathbb{P}(\tau_{\mathbf{R}^n \backslash \Omega^o_{\rho_{nn}}} < \tau_{\Omega_\rho}) \geq (1-\delta)(1-\gamma)(1-\lambda), \quad \forall x(t_k) \in \Omega_{\rho_c} \backslash \Omega^o_{\rho_{nn}} \tag{3.38}$$

where

$$\frac{\sup_{x\in\partial\Omega_{\rho_{nn}}} V(x)}{\inf_{x\in\mathbf{R}^n\backslash\Omega_{\rho_{min}}} V(x)} \leq \beta \tag{3.39a}$$

$$\sup_{x \in \Omega_{\rho_c} \backslash \Omega^o_{\rho_{nn}}} \frac{V(x)}{\rho} \leq \gamma \tag{3.39b}$$

Proof. The two probabilities in Eqs. 3.37-3.38 can be interpreted as follows. Eq. 3.37 gives the probability that the future state remains inside $\Omega_{\rho_{min}}$ (i.e., the terminal set around the origin) during one sampling period for any initial state inside $\Omega_{\rho_{nn}}$ ($\Omega_{\rho_{nn}}$ is a subset of $\Omega_{\rho_{min}}$ as defined in Eq. 3.29). Eq. 3.38 is the probability that the state hits the boundary of $\Omega_{\rho_{nn}}$ before it leaves the stability region Ω_ρ for any initial states in $\Omega_\rho \backslash \Omega_{\rho_{nn}}$. The proof consists of three parts. In the first part, we demonstrate that the infinitesimal generator $\mathcal{L}V$ for the stochastic system of Eq. 3.32 is rendered negative (Eq. 3.34) under the controller $u = \Phi_s(x) \in U$ for all $x \in \Omega_\rho$ with a certain probability, which is a key step in the derivation of Eqs. 3.37-3.38. Specifically, Eq. 3.34 holds almost surely for the RNN model since the stability region Ω_ρ is characterized using the RNN model and the controller $u = \Phi_s(x) \in U$; however, due to the existence of generalization error, Eq. 3.34 holds for the nonlinear system of Eq. 3.32 in a probabilistic manner (i.e., with probability at least $1-\delta$), which leads to the probability of $1-\delta$ on the RHS of Eqs. 3.37-3.38. Furthermore, the final probability of $\mathcal{L}V$ being negative also needs to account for the impact of stochastic, unbounded disturbance. In the second part, we prove the probabilities in Eq. 3.37 and Eq. 3.38, and demonstrate that the probability terms (i.e., $1-\beta$, $1-\gamma$ and $1-\lambda$) depend on the size of the sets Ω_ρ, $\Omega_{\rho_{nn}}$, $\Omega_{\rho_{min}}$ and the sampling period. Finally, in the third part, we prove probabilistic closed-loop stability for the stochastic nonlinear system of Eq. 3.32 under the RNN-MPC of Eq. 3.36.

Part 1 : We first prove that there exists a sufficiently small sampling period Δ such that $\mathcal{L}V(x(t))$ can be rendered negative for all the states $x(t_k) \in \Omega_\rho \backslash \Omega^o_{\rho_s}$ in probability. Following

the proof of Theorem 1 in [81], we define an event that the disturbance $w(t)$ is bounded within one sampling period as follows: $A_B := \{w \in \mathbf{R}^q \mid \sup_{t\in[t_k,t_k+\Delta)} |w(t)| \leq B\}$. Then, there exists a sufficiently small ball B such that $P(A_B) = 1-\lambda$ holds for any probability $\lambda = (0, 1]$ under the disturbance $w(t)$ following standard Brownian motion. As a result, for $w \in A_B$, the probability $\mathbb{P}(A_W) \geq 1-\lambda$ holds for the event $A_W := \{\sup_{t\in[t_k,t_k+\Delta)} |x(t)-x(t_k)| \leq k_1(\Delta)^r\}$, $k_1 > 0$, $r < \frac{1}{2}$, which states that the state evolution is bounded within one sampling period in the presence of a bounded disturbance [84]. Subsequently, we prove that $\mathcal{L}V(x(t))$ is negative for any $x(t_k) \in \Omega_\rho \backslash \Omega^{\mathrm{o}}_{\rho_s}$ accounting for the modeling error between RNN model and the nominal system of Eq. 3.32. Specifically, by letting the training sample size $m \geq m_N(|x|, h_c, \delta)$ such that the modeling error is constrained by $\gamma|x|$ for any $x(t_k) \in \Omega_\rho \backslash \Omega^{\mathrm{o}}_{\rho_s}$, the following equation holds under $u = \Phi_s(x) \in U$, with probability no less than $1-\delta$.

$$\begin{aligned}
\mathcal{L}V(x(t)) =& \frac{\partial V}{\partial x}(F_{nn}(x, \Phi_s(x)) + F(x, \Phi_s(x), 0) - F_{nn}(x, \Phi_s(x))) + \frac{1}{2}Tr\{h(x)^T \frac{\partial^2 V(x)}{\partial x^2} h(x)\} \\
&\leq -\alpha_1|x|^2 + |F_{nn}(x, \Phi_s(x)) - F(x, \Phi_s(x), 0)| \cdot \hat{c}_4|x| \\
&\leq -\alpha_1|x|^2 + \gamma|x| \cdot \hat{c}_4|x| \\
&= -\tilde{\alpha}_1|x|^2 \\
&< 0
\end{aligned} \tag{3.40}$$

where $\tilde{\alpha}_1 = -\alpha_1 + \hat{c}_4\gamma < 0$ for any $\gamma < \alpha_1/\hat{c}_4$. Eq. 3.40 shows that under the stochastic stabilizing controller $u = \Phi_s(x)$ designed for the RNN model, the infinitesimal generator $\mathcal{L}V(x(t))$ for the stochastic nonlinear system of Eq. 3.32 is rendered negative for $x \in \Omega_\rho \backslash \Omega^{\mathrm{o}}_{\rho_s}$ with probability at least $1-\delta$, provided that the training sample size is chosen appropriately

to ensure a sufficiently small and bounded modeling error. As a result, we can find a positive real number κ such that $\mathcal{L}V(x(t)) \leq -\kappa V(x)$ holds for any $x \in \Omega_\rho \backslash \Omega^o_{\rho_s}$ with probability at least $1-\delta$.

Subsequently, we prove under sample-and-hold implementation (i.e., $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$), there exists a sufficiently small sampling period Δ such that $\mathcal{L}V(x(t))$ can be rendered negative within one sampling period. Specifically, since $\mathcal{L}V(x) = L_f V(x) + L_g V(x) + \frac{1}{2} Tr\{h(x)^T \frac{\partial^2 V(x)}{\partial x^2} h(x)\}$, and $L_f V(x)$, $L_g V(x)$, $h(x)^T \frac{\partial^2 V(x)}{\partial x^2} h(x)$ are locally Lipschitz, there exist positive real numbers k_3, k_4, k_5 such that the following equations hold.

$$
\begin{aligned}
|L_f V(x(t)) - L_f V(x(t_k))| &\leq k_3 |x(t) - x(t_k)| \\
|L_g V(x(t))u(t_k) - L_g V(x(t_k))u(t_k)| &\leq k_4 |x(t) - x(t_k)| \\
|\frac{1}{2} Tr\{h(x(t))^T \frac{\partial^2 V(x(t))}{\partial x^2} h(x(t))\} - \frac{1}{2} Tr\{h(x(t_k))^T \frac{\partial^2 V(x(t_k))}{\partial x^2} h(x(t_k))\}| &\leq k_5 |x(t) - x(t_k)|
\end{aligned}
\tag{3.41}
$$

Using the results from Eq. 3.40, we obtain the following inequality for $\mathcal{L}V(x(t))$, $\forall x \in \Omega_\rho \backslash \Omega^o_{\rho_s}$ under $u(t) = \Phi_s(x(t_k)) \in U$, $\forall t \in [t_k, t_{k+1})$:

$$
\begin{aligned}
\mathcal{L}V(x(t)) =& \mathcal{L}V(x(t_k)) + (\mathcal{L}V(x(t)) - \mathcal{L}V(x(t_k))) \\
\leq& -\kappa\rho_s + (k_3 + k_4 + k_5)|x(t) - x(t_k)|
\end{aligned}
\tag{3.42}
$$

Therefore, for any $w \in A_B$ with $\mathbb{P}(A_W) \geq 1-\lambda$, and $\Delta < (\frac{\kappa\rho_s - \epsilon}{k_1(k_3+k_4+k_5)})^{(\frac{1}{r})}$, we have $\mathcal{L}V(x(t)) < -\epsilon$, for all $t \in [t_k, t_{k+1})$. We define the event that $\mathcal{L}V(x(t))$ is rendered negative within one sampling period as $A_V := \{\sup_{t\in[t_k,t_{k+1})} \mathcal{L}V(x(t)) < -\epsilon\}$, and the final probabil-

ity of A_V occurring is derived as $\mathbb{P}(A_V) \geq (1-\lambda)(1-\delta)$, given that the modeling error is sufficiently small, and the disturbance is bounded. This completes the proof of *Part* 1.

Part 2 : Subsequently, we prove the main results of the probabilities of Eqs. 3.37-3.38. We first prove Eq. 3.37 which states that for any initial state inside $\Omega_{\rho_{nn}}$, the future state remains inside the terminal set $\Omega_{\rho_{min}}$ in one sampling period. To simply the notations, the conditional expectations and the probabilities given that the event A_V occurs are denoted as $\mathbb{E}^*(\cdot)$ and $\mathbb{P}^*(\cdot)$, respectively. We consider the extreme scenario where the initial state is on the boundary of $\Omega_{\rho_{nn}}$, and show Eq. 3.37 holds in this case. Specifically, using Dynkin's formula, we obtain the expected value of $V(x)$ as follows [79, 81, 85]:

$$\mathbb{E}^*(V(x(\tau_{T,\mathcal{Z}}(t)))) = V(x(t_k)) + \mathbb{E}^*(\int_{t_k}^{t_k+\tau_{T,\mathcal{Z}}(t)} \mathcal{L}V(x(s))ds) \tag{3.43}$$

where $\mathcal{Z} = \Omega_{\rho_{min}} \backslash \Omega^{\mathrm{o}}_{\rho_{nn}}, t \in [t_k, t_{k+1})$, and $T = \infty$. As defined in Section "Notations", we have $\tau_{T,\mathcal{Z}}(t) = \min\{\tau_{\mathcal{Z}}, T, t\}$, where $\tau_{\mathcal{Z}}$ is the hitting time of the set $\mathcal{Z}$. Then, for any $x(t_k) \in \partial\Omega_{\rho_{nn}}$, we have the following inequality using the proof technique in [79, 81].

$$\begin{aligned} \mathbb{E}^*(V(x(\tau_{T,\mathcal{Z}}(t)))) &= \int_{V\geq\tilde{\lambda}} V(x(\tau_{T,\mathcal{Z}}(t)))d\mathbb{P}^* + \int_{V<\tilde{\lambda}} V(x(\tau_{T,\mathcal{Z}}(t)))d\mathbb{P}^* \\ &\geq \tilde{\lambda}\mathbb{P}^*(V(x(\tau_{T,\mathcal{Z}}(t))) \geq \tilde{\lambda}) \end{aligned} \tag{3.44}$$

Let $\tilde{\lambda} = \inf_{x\in\mathbf{R}^n\setminus\Omega_{\rho_{min}}} V(x)$. We have the following inequality for $x(t_k) \in \partial\Omega_{\rho_{nn}}$:

$$\begin{aligned}\mathbb{P}^*(V(x(t))) \geq \rho_{min}, \text{ for some } t \in [t_k, t_{k+1})) &\leq \frac{\mathbb{E}^*(V(x(\tau_{T,\mathcal{Z}}(t))))}{\tilde{\lambda}} \\ &= \frac{V(x(t_k)) + \mathbb{E}^*(\int_{t_k}^{t_k+\tau_{T,\mathcal{Z}}(t)} \mathcal{L}V(x(s))ds)}{\inf_{x\in\mathbf{R}^n\setminus\Omega_{\rho_{min}}} V(x)} \\ &\leq \frac{V(x(t_k))}{\inf_{x\in\mathbf{R}^n\setminus\Omega_{\rho_{min}}} V(x)}\end{aligned} \tag{3.45}$$

The last inequality is obtained using the fact derived in *Part* 1 that $\mathcal{L}V$ is rendered negative with probability at least $(1-\lambda)(1-\delta)$. By taking the complementary events, we obtain the probability $\inf_{x(t_k)\in\partial\Omega_{\rho_{nn}}} \mathbb{P}^*(V(x(t)) < \rho_{min}, \forall t \in [t_k, t_{k+1})) \geq (1-\beta)$, conditioned on the occurrence of event A_V, where β is defined in Eq. 3.39a. Since we have $\mathbb{P}(A_V) \geq (1-\lambda)(1-\delta)$ from *Part* 1, the probability of Eq. 3.37 is obtained using the properties of conditional probability.

Next, we consider the initial state $x(t_k) \in \Omega_\rho\setminus\Omega_{\rho^\circ_{nn}}$, and prove the probability of Eq. 3.38. Specifically, we assume the initial state is on the boundary of Ω_{ρ_c}, $\rho_c \in [\rho_{nn}, \rho]$, which is a set between Ω_ρ and $\Omega_{\rho_{nn}}$, and show that the state will reach the boundary of $\Omega_{\rho_{nn}}$ before leaving Ω_ρ with a certain probability. Let $A_T := \{\tau_{\mathbf{R}^n\setminus\Omega^\circ_{\rho_{nn}}} > \tau_{\Omega_\rho}\}$ denotes the complementary event that the state first hits the boundary of Ω_ρ instead of $\Omega_{\rho_{nn}}$. The following inequality is obtained since the event A_T belongs to the event $\{\frac{V(x(\tau_{\Omega_\rho\setminus\Omega^\circ_{\rho_{nn}}}))}{\rho} \geq 1\}$.

$$\mathbb{P}^*(\tau_{\mathbf{R}^n\setminus\Omega^\circ_{\rho_{nn}}} > \tau_{\Omega_\rho}) \leq \mathbb{P}^*(\frac{V(x(\tau_{\Omega_\rho\setminus\Omega^\circ_{\rho_{nn}}}))}{\rho} \geq 1) \leq \frac{V(x(t_k))}{\rho} \tag{3.46}$$

The last inequality is derived using the results in Eq. 3.45. Therefore, given a positive real number γ satisfying Eq. 3.39b, we have $\mathbb{P}^*(\tau_{\mathbf{R}^n\setminus\Omega^\circ_{\rho_{nn}}} < \tau_{\Omega_\rho}) \geq (1-\gamma)$ by taking the

complementary event of A_T. The final probability of Eq. 3.38 is derived accounting for the conditional probability of A_V.

Part 3 : Finally, consider the stochastic nonlinear system of Eq. 3.32 under the RNN-MPC of Eq. 3.36. When $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}^o}$, the constraint of Eq. 3.36e is activated to optimize control actions such that $\mathcal{L}V$ is no greater than the one using the stabilizing controller $u = \Phi_s(x) \in U$, and thus, is also rendered negative. Therefore, using the results in *Part* 2 which prove that with a certain probability the state will reach the boundary of $\Omega_{\rho_{nn}}$ before leaving Ω_ρ, it follows that the probability under MPC is no worse than the probability of Eq. 3.38. Once the state enters $\Omega_{\rho_{nn}}$, the constraint of Eq. 3.36f is activated to maintain the predicted states within $\Omega_{\rho_{nn}}$. In this case, Eq. 3.37 gives the probability that the state of Eq. 3.32 remains inside $\Omega_{\rho_{min}}$ for the next sampling period. Therefore, for the stochastic nonlinear system of Eq. 3.32 under the RNN-MPC of Eq. 3.36, we derive the probability of closed-loop stability in the sense that the closed-loop state is bounded in Ω_ρ, and is ultimately bounded in $\Omega_{\rho_{nn}}$. This completes the proof of Theorem 3.3.

□

Remark 3.3. *Eqs. 3.37-3.38 in Theorem 3.3 give the the probabilities of closed-loop stability for each sampling period. While the RNN predictions are invoked recursively within MPC to predict for the entire prediction horizon, the probabilities of closed-loop stability remain unaffected since only the first control action is applied for the next sampling period. The RNN-MPC is implemented in a receding horizon manner by recursively solving the optimization problem of Eq. 3.36 with new state measurements received at each sampling time. Therefore, at each time step, Eqs. 3.37-3.38 can be used to estimate the probability of system being stable under RNN-MPC.*

Remark 3.4. *It should be noted that the probabilities of Eqs. 3.37-3.38 represent only the lower bounds for closed-loop stability under RNN-MPC. The actual probability of closed-loop stability could be higher due to a number of reasons: 1) the RNN model is well trained and the modeling error does not reach the upper bound for every time step, 2) in general, the stochastic disturbances in industrial chemical plants fall within a bounded region in most of the time, which can be handled through the robustness of MPC, and 3) the optimality of MPC improves closed-loop performance in terms of fast convergence to the steady-state under the constraint of Eq. 3.36f, which leads to better probability results than those derived under the controller* $u = \Phi_s(x)$ *in Theorem 3.3.*

Remark 3.5. *Theorem 3.3 demonstrates that in addition to the RNN structure in terms of width and depth, and the training sample size that affect the neural network generalization performance (Theorem 3.1), the closed-loop stability for the stochastic system of Eq. 3.32 under RNN-MPC also depends on the sampling time and the size of multiple sets embedded in the stability region* Ω_ρ*. Therefore, all the factors above should be accounted for to improve the overall probability of stability for the nonlinear systems subject to stochastic disturbances.*

Remark 3.6. *In the presence of stochastic disturbances with unbounded variation, one of the benefits of using the RNN-MPC of Eq. 3.36 is that the operating region can be characterized less conservatively by utilizing the probability distribution of disturbances. It was demonstrated in [81] that compared to the RNN-MPC of Eq. 3.36, the closed-loop operating regions were overly conservative using the robust controller design that handles disturbances in a bounded manner, which leads to reduced economic benefits in the context of economic MPC. Therefore, the probabilities in Theorem 3.3 demonstrates the relationship between closed-loop stability and operating region size, which could provide a guidance to characterize the operating region when implementing machine learning models in real chemical processes subject to various process disturbances.*

3.5 Application to a Chemical Process Example

To demonstrate the efficacy of machine-learning-based MPC and study the impact of data sample size on RNN generalization performance and system stability in the presence of bounded disturbances and stochastic disturbances, we present a simulation example using the chemical process example from [49]. Specifically, we consider a continuous stirred tank reactor (CSTR) that is non-isothermal and well-mixed with reactant A transformed into product B ($A \rightarrow B$) in an exothermic, irreversible second-order reaction. A heating jacket is equipped to remove/supply heat at a rate Q. We first consider the case of bounded disturbances, and present the process dynamical model by the following energy and material balance equations:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 + w_1 \tag{3.47a}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} + w_2 \tag{3.47b}$$

where T denotes the temperature in the reactor, and C_A represents the concentration of reactant A. F is the volumetric flow rate, T_0 is the feed temperature, and C_{A0} is the feed concentration of reactant A. V and Q are the volume of the reacting substance in the reactor and the heat input rate, respectively. $w^T = [w_1 \ w_2]$ are bounded disturbances of Gaussian distribution with variance $\sigma_1 = 2.5 \ kmol/m^3$, $\sigma_2 = 70 \ K$, and bounds $|w_1| \leq 2.5 \ kmol/m^3$, $|w_2| \leq 70 \ K$. The definition of all the other parameters and their values are reported in [49].

In the presence of stochastic disturbances, the nonlinear system can be represented in

the following form:

$$dC_A = \frac{F}{V}(C_{A0} - C_A)dt - k_0 e^{\frac{-E}{RT}} C_A^2 dt + \bar{\sigma}_1 (C_A - C_{As}) d\bar{w}_1 \tag{3.48a}$$

$$dT = \frac{F}{V}(T_0 - T)dt + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 dt + \frac{Q}{\rho_L C_p V} dt + \bar{\sigma}_2 (T - T_s) d\bar{w}_2 \tag{3.48b}$$

where $\bar{w}_1,\ \bar{w}_2$ are standard Wiener processes that satisfy $\bar{w}(0) = 0$ and $\bar{w}(t) - \bar{w}(s) \sim \sqrt{t-s}\mathcal{N}(0,1)$ ($\mathcal{N}(0,1)$ is a normal distribution with zero mean and unit variance). To simulate the Wiener process, we discretize the Wiener process with the integration time step h_c as follows: $d\bar{w}_i \sim \sqrt{h_c}\mathcal{N}(0,1),\ i = 1, 2$, and thus, the realization of the Wiener process can be obtained through $\bar{w}_i(t + h_c) = \bar{w}_i(t) + d\bar{w}_i,\ \forall t \geq 0$. The coefficients $C_A - C_{As}$ and $T - T_s$ are added to ensure that the disturbances vanish at the steady-state. The variances $\bar{\sigma}_1 = 2.5\ kmol/m^3$, $\bar{\sigma}_2 = 70\ K$ are used in the simulation of Eq. 3.48. The RNN-MPC is designed to stabilize the reactor at $(T_s,\ C_{As}) = (402\ K,\ 1.95\ kmol/m^3)$, which is an unstable steady-state under the given input values $(Q_s,\ C_{A0s}) = (0\ kJ/hr,\ 4\ kmol/m^3)$. The heat supply/removal rate and the inlet concentration of species A are the two manipulated inputs. All the states and inputs of the process are represented in their deviation variable forms, i.e., $\Delta T = \ T - T_s$, $\Delta C_A = C_A - C_{As}$, $\Delta C_{A0} = C_{A0} - C_{A0_s}$, and $\Delta Q = Q - Q_s$. Additionally, the upper bounds of the manipulated inputs are $|\Delta Q| \leq 5 \times 10^5\ kJ/hr$ and $|\Delta C_{A0}| \leq 3.5\ kmol/m^3$.

The RNN training process follows the standard training method in [49]. In this study, the RNN models are trained using different data sample size (while other parameters and settings remain unchanged), and the generalization performance is evaluated using the test-

ing data. The RNN models are built with 50 neurons in a single hidden layer. The mean squared error (MSE) is used as the loss function. PyIpopt, which is a python connector to the IPOPT software package, is used to solve the MPC optimization problem [86], and Keras is used to build and train RNN models [5]. Fig. 3.2 shows the relationship between the RNN generalization performances and the training sample size, from which it is demonstrated that with less data used for training, the errors for testing and training both increase [21]. Additionally, we calculate the generalization gap using $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] - \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$. The increase of the generalization gap in Fig. 3.2 implies a worse generalization performance for models with less training data.

Subsequently, we simulate the closed-loop system with bounded disturbances and Wiener process disturbances with unbounded variation. Fig. 3.3 shows the probability results obtained through the simulation of various initial conditions (48 initial conditions) within $\Omega_{\hat{\rho}}$ using the six RNN models trained earlier with different sample sizes. The probabilities in Fig. 3.3 are calculated using the following rule: given an initial condition, the closed-loop system under RNN-MPC is unstable if the state trajectory leaves $\Omega_{\hat{\rho}}$ at any time step or escapes from $\Omega_{\rho_{min}}$ after it enters $\Omega_{\rho_{min}}$ due to disturbances and/or modeling error. It is shown in Fig. 3.3 that the probability of closed-loop stability increases with more data used for training, which is consistent with the results shown in the open-loop simulation study (Fig. 3.2). Additionally, it is observed that the probabilities of closed-loop stability under bounded disturbances reach 0.7 for training sample size greater than 1×10^4, and the probabilities under unbounded, stochastic disturbances reach 0.6 with the same number of training sample size. Overall, it is shown that the MPC under bounded disturbances achieves

higher probabilistic of closed-loop stability than that under unbounded, stochastic disturbances with the same variances. This is consistent with the disturbance realizations shown in Fig. 3.4, from which it is demonstrated that the Gaussian disturbance on temperature T is bounded within the $\pm\sigma_2$ region (top figure), while the Wiener process disturbance is unbounded, and has a greater impact on system stability due to its wider range.

The state-space trajectory and the state profiles for the initial state $x_0 = (-1.2,\ 50)$ with the model trained with 7000 training data are shown in Fig. 3.5 and Fig. 3.6. In Fig. 3.5, it is seen that the dynamic trajectory remains inside the stability region $\Omega_{\hat{\rho}}$ all the time, and ultimately converges to the small ball $\Omega_{\rho_{min}}$ in the presence of bounded disturbances (blue, solid line); however, in the presence of stochastic disturbances, the state trajectory leaves $\Omega_{\rho_{min}}$ during its oscillation around the steady-state, and thus is considered unstable in this case. In Fig. 3.6, it is shown that for this particular initial condition, the closed-loop states (i.e., reactor temperature T and reactant concentration C_A) are stabilized at the origin after around 0.06 hr for both disturbances, with slight variation around the steady-state afterwards due to disturbances. The case study demonstrates the relation between RNN training sample size and its generalization performance as well as the probability of closed-loop system stability, which supports the results derived in Theorem 3.1 and Theorem 3.2.

Remark 3.7. *Note that the RNN generalization performance depends on various factors as demonstrated in Theorem 3.1. While we only showed the impact of the sample size of training data on RNN generalization performance in this section due to space limitations, the generalization error is also affected by RNN width/depth, and input time length. Interested readers are referred to [21] for the simulation studies of open-loop RNN generalization performance and system stability analysis that account for all the above factors for the nominal*

system without any disturbances.

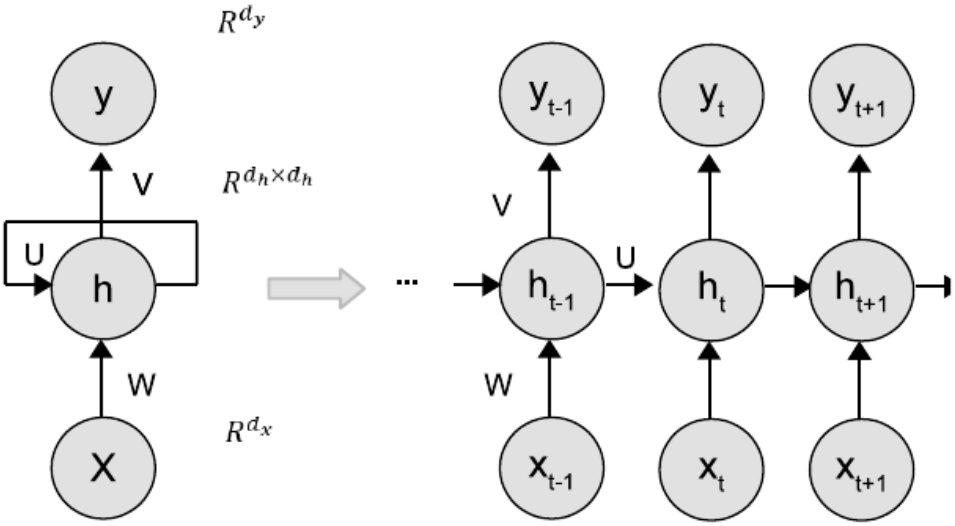

Figure 3.1: Recurrent neural network structure.

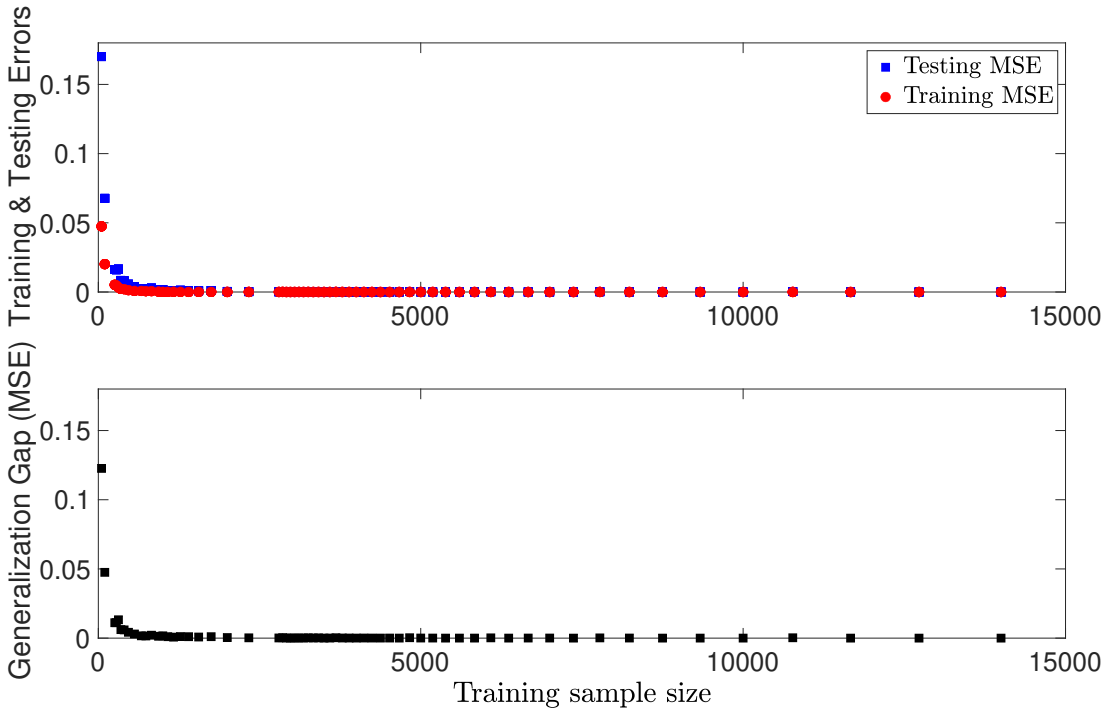

Figure 3.2: Generalization performance for the RNN models utilizing various sample sizes.

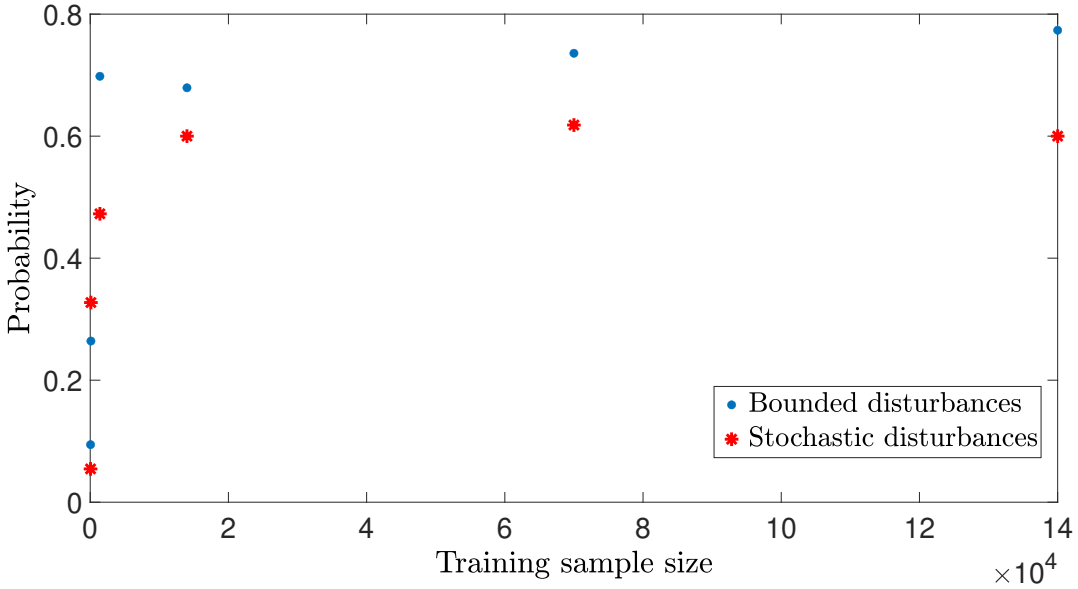

Figure 3.3: Probability of closed-loop stability under bounded disturbances (blue circles) and stochastic, unbounded disturbances (red asterisks), respectively, using the RNN-MPC trained with various sample sizes.

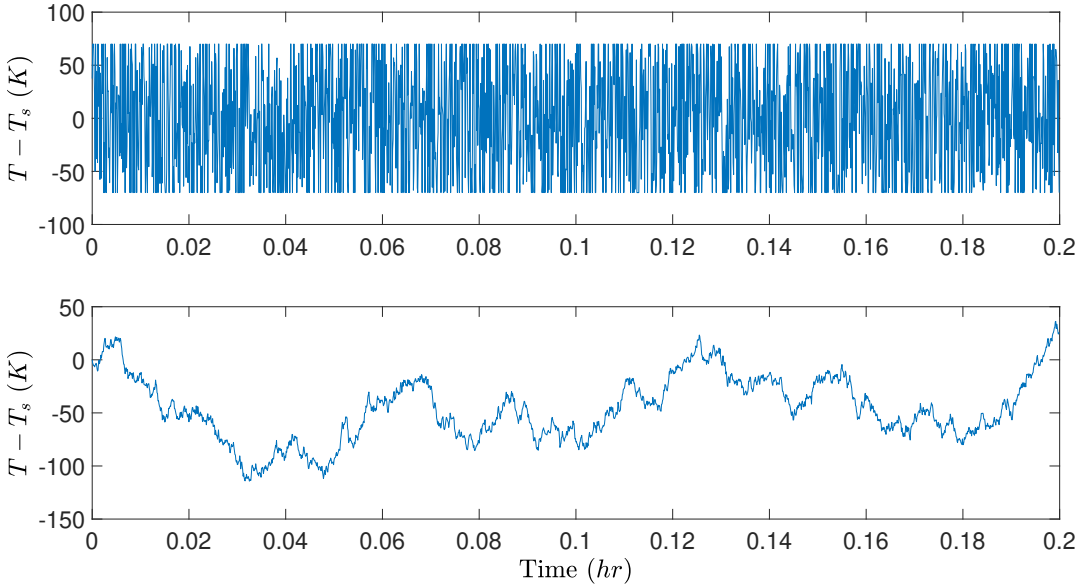

Figure 3.4: Bounded, Gaussian disturbance (top figure), and unbounded, Wiener process disturbance (bottom figure) on temperature T.

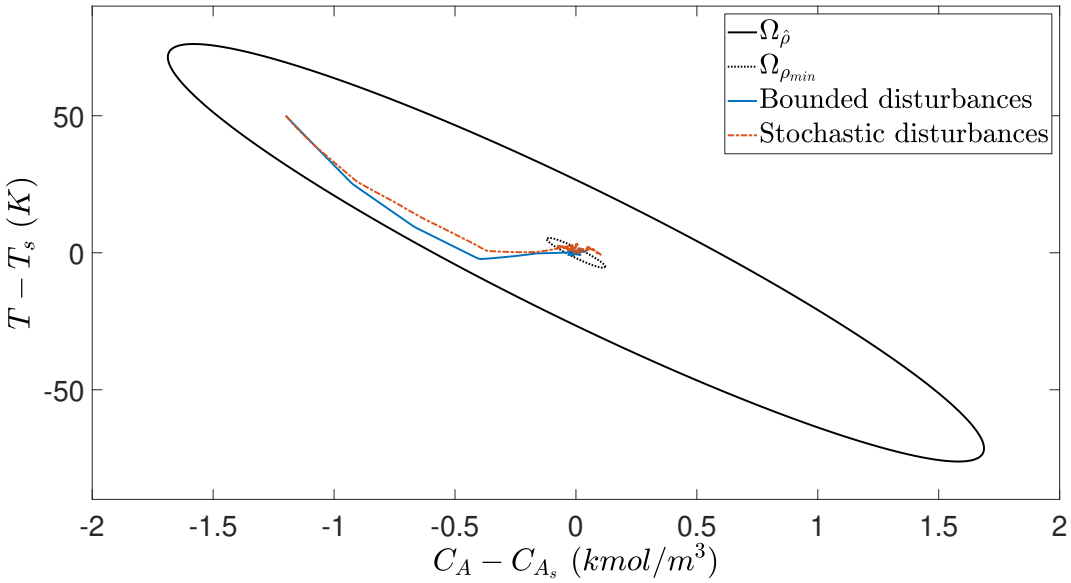

Figure 3.5: Closed-loop state trajectories under MPC with bounded disturbances (blue, solid line) and stochastic, unbounded disturbances (red, dashed line) for the same initial condition $(-1.2,\ 50)$.

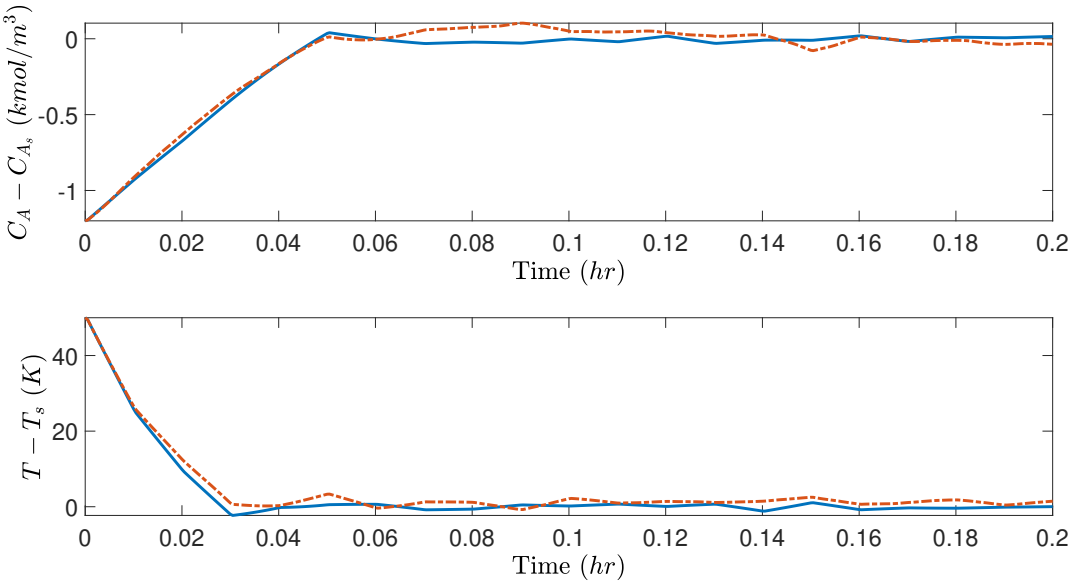

Figure 3.6: Closed-loop state profiles for the CSTR with bounded disturbances (blue, solid line) and stochastic, unbounded disturbances (red, dashed line) under RNN-MPC for the same initial condition $(-1.2,\ 50)$.

Chapter 4

On Generalization Error of Neural Network Models and its Application to Predictive Control of Nonlinear Processes

4.1 Introduction

An ongoing research issue in process systems engineering is the modeling of large-scale, complicated nonlinear processes. Traditional methods for modeling nonlinear systems include first-principles modeling, which is based on a fundamental comprehension of the core physico-chemical phenomena, and data-driven modeling, which identifies parameters from simulation or industry data (e.g., [87, 88]). Although the classic first-principles modeling approach has been widely utilized in the control, monitoring, and optimization of different

chemical processes, applying first-principles computational methods to represent complicated nonlinear systems can be time-consuming and inaccurate. Given their ability to successfully handle large data sets from processes and to model a diverse range of nonlinear functions, machine learning techniques are being used more and more to approximate complicated nonlinear systems (e.g., [89–92]). Among various machine learning modeling tools, when modeling nonlinear dynamic systems utilizing time-series data, recurrent neural networks (RNN) are frequently employed [93, 94]. Although machine learning techniques have been used in chemical process control since the nighties [56], they have recently gained popularity again due to a variety of factors, including more affordable computation (thanks to mature and effective libraries and hardware), the accessibility of large data sets, and sophisticated learning algorithms. The upcoming generation of industrial control systems will certainly be impacted by the developments of model predictive controllers (MPC) that make use of machine learning models with well-characterized fidelity.

The traditional choice to analyze time-series data in a black-box fashion is a fully-connected RNN model, which densely relates all the available inputs to all the outputs. However, this method may not always be the finest, particularly for intricate chemical processes. For instance, in an integrated chemical plant, the downstream units do not have an impact on the upstream ones. Therefore, in order to further increase a RNN model's accuracy, numerous studies (e.g., [95–97]) have studied gray-box modeling, also referred to in the literature as hybrid modeling, which involves incorporating prior knowledge into the design of neural network models of various chemical processes. A strategy for combining data-driven modeling with first-principles knowledge was recently developed by [98], and it explicitly

permits the inclusion of data on known gains among particular inputs and outcomes. With prior knowledge of relations, this suggested strategy can be used in large-scale processes. Also, other approaches to improve the RNN model's prediction accuracy were proposed, for example a weight-constrained RNN modeling was investigated in [36] with chemical process example and yield improvements in both open-loop and closed-loop simulations under ML based MPC.

Another modeling strategy to follow is the recently proposed partially-connected RNN which, as the name indicates, partially connects layers based on pre-existing knowledge in terms of physical relations among the underlying system inputs and outputs [36, 99, 100]. Specifically, on a large and complex chemical process modeled in Aspen Plus Dynamics simulator, [100] investigated this approach by carrying out open-loop and closed-loop simulations using a fully-connected RNN model against a partially-connected RNN model. A partially-connected RNN model was shown to outperform the fully-connected RNN model when incorporated into a MPC, with smoother state trajectories and less computational burden. Furthermore, in [59], they considered the case of industrial noise (i.e., non-Gaussian noise), where the Monte Carlo dropout and co-teaching strategies were used to train partially-connected RNN models to overcome the over-fitting issue. Subsequently, open-loop and closed-loop simulations were performed on an Aspen Plus Dynamics process model to illustrate the superiority of partially-connected RNN based MPC over fully-connected RNN models trained with dropout/co-teaching and standard partially-connected RNN models with regular training. Additionally, one can consider the Long Short-Term Memory (LSTM), a variant of the RNN that was introduced three decades ago, to model nonlinear systems. LSTMs have a

unique structure, which enables them to enhance the model's performance when dealing with data that requires long time dependencies. Such data may occur when modeling nonlinear time-delay systems or even nonlinear systems with disturbances and noise. For instance, in [59], a nonlinear system was modeled as an LSTM network using noisy data, and closed loop stability was achieved. Moreover, LSTMs have been shown to overcome the vanishing gradient phenomenon that usually occurs when using RNNs (see [50] for further details). Hence, LSTMs are widely used in many recent chemical engineering applications, and have proven to be an efficient and powerful machine learning tool.

The adaptation of machine-learning-based MPC to actual chemical processes is still met with several challenges, despite the effectiveness of machine learning approaches in approximating nonlinear process dynamics within the context of MPC. Furthermore, characterizing the generalization capability of machine learning models learned using finite training samples on new data is a significant challenge. The work of [21] has filled in this gap by computing an explicit expression for the theoretical upper bound of fully-connected RNN models' generalization error. However, the fundamental question regarding the generalization accuracy of partially-connected RNN models in MPC has not been addressed—specifically, how the structure of an RNN model affects its generalization accuracy.

Due to the aforementioned considerations, in this work, we develop, from machine learning theory, a conceptual framework to quantify generalization error bounds for partially-connected RNN models. Also, we integrate these models into model predictive control systems to be implemented in nonlinear chemical processes. This manuscript is divided into 5 sections. Section 4.2 presents the class of nonlinear systems considered and assump-

tions regarding system stability. In Section 4.3, the representation and the construction of RNNs both fully-connected and partially-connected, and LSTMs is presented. Section 4.4 starts with key definitions and lemmas, and then develop probabilistic generalization error upper bounds for partially-connected RNN models and LSTM networks. The integrating of a partially-connected RNN model and LSTM model into a MPC while accounting for Lyapunov stability considerations is proposed and discussed in Section 4.5. Lastly, the improvements associated with incorporating prior physical knowledge into RNN modeling is illustrated in Section 4.6 via both open-loop and closed-loop simulations using a two reactors in series chemical process under Lyapunov-based MPC.

4.2 Preliminaries

4.2.1 Notation

Given a vector $b \in \mathbb{R}^n$, its Euclidean norm is denoted by the operator $\|b\|$, and the weighted Euclidean norm of a vector is denoted by the operator $\|b\|_Q$ where Q is a positive definite matrix. Moreover, the infinity norm of b is given by $\|b\|_\infty$. Generally, for $b \in \mathbb{R}^n$ and $\gamma \geq 1$, $\|b\|_\gamma = \left(\sum_{i=1}^n |b_i|^\gamma\right)^{\frac{1}{\gamma}}$. Given a matrix $W \in \mathbb{R}^{m \times n}$, its Frobenius and spectral norms are denoted by $\|W\|_F$ and $\|W\|_\infty$, respectively. Given real numbers γ and κ, the γ-norm ,κ-norms of the columns of W is denoted by $\|W\|_{\gamma,\kappa} = \left(\sum_i^m \left(\sum_j^n |W_{j,i}|^\gamma\right)^{\frac{\kappa}{\gamma}}\right)^{\frac{1}{\kappa}}$. R_+ denotes non-negative real numbers. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{x} f(x)$. Set subtraction is denoted by '$-$', i.e., $A - B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously

differentiable. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be L-Lipschitz, $L \geqslant 0$, if $|f(a) - f(b)| \leq L|a - b|$ for all $a, b \in \mathbb{R}^n$. $\mathbb{P}(A)$ denotes the probability that the event A will occur. $\mathbb{E}[X]$ denotes the expected value of a random variable X. We note that the infinity norm of a vector and the spectral norm of a matrix both have the notation $\|.\|_\infty$, by mathematical convention. Therefore, in order to differentiate between them, it is important to identify the argument of the norm.

4.2.2 Class of Systems

We consider the class of multi-input multi-output (MIMO) nonlinear continuous-time systems represented by the following state-space form:

$$\dot{x} = \mathbf{F}(x, u) := F(x) + G(x)u \tag{4.1}$$

where the state vector of the system is $x = [x_1, ..., x_{n_x}]^T \in \mathbb{R}^{n_x}$, $y = [y_1, ..., y_{n_y}]^T \in \mathbb{R}^{n_y}$ is the output vector, and the manipulated input vector is $u = [u_1, ..., u_{n_u}]^T \in \mathbb{R}^{n_u}$. $\mathbf{F}(x, u)$ represents a nonlinear vector function of x and u. The constraints on control inputs are given by $u \in U := \{{u_i}^{\min} \leq u_i \leq {u_i}^{\max}\}$. $F(\cdot)$ and $G(\cdot)$ denote nonlinear vector and matrix functions of $n_x \times 1$ and $n_x \times n_u$ dimensions, respectively, and both are assumed to be sufficiently smooth.

4.2.3 Stabilizability assumption

We assume that there exists a control law $u = \Phi(x) \in U$ based on state feedback that can make the origin of the system of Eq. (4.1) exponentially stable. This stabilizability assumption implies the existence of a $\mathcal{C}^1$ control Lyapunov function denoted as $V(x)$, such that the following inequalities hold for all x in an open neighborhood D around the origin:

$$c_1\|x\|^2 \leq V(x) \leq c_2\|x\|^2 \tag{4.2a}$$

$$\frac{\partial V(x)}{\partial x}\mathbf{F}(x, \Phi(x)) \leq -c_3\|x\|^2 \tag{4.2b}$$

$$\left\|\frac{\partial V(x)}{\partial x}\right\| \leq c_4\|x\| \tag{4.2c}$$

where c_i, $i = 1, 2, 3, 4$, are positive constants. A candidate controller $\Phi(x)$ may be constructed via Sontag's control law ([101]). Then, following [15], we characterize the closed-loop stability region Ω_ρ to be a level set of the Lyapunov function in the region D in which the time-derivative $\dot{V}(x)$ is negative under the controller $u = \Phi(x) \in U$ such that $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, where $\rho > 0$. Furthermore, based on the Lipschitz property of $\mathbf{F}(x, u)$ and the boundedness of u, there exists positive constants M, $L_x, L_x^{'}$ such that the following inequalities hold for all $x, x' \in D$ and $u \in U$:

$$\|\mathbf{F}(x, u)\| \leq M \tag{4.3a}$$

$$\left\|\mathbf{F}(x, u) - \mathbf{F}(x', u)\right\| \leq L_x\|x - x'\| \tag{4.3b}$$

$$\left\|\frac{\partial V(x)}{\partial x}\mathbf{F}(x, u) - \frac{\partial V(x')}{\partial x}\mathbf{F}(x', u)\right\| \leq L_x^{'}\|x - x'\| \tag{4.3c}$$

4.3 Recurrent neural networks (RNNs)

The prospect of utilizing artificial intelligence (AI) techniques in chemical engineering has been investigated at length in the recent literature. AI technology has led to the rise of classical and powerful modeling tools such as fuzzy logic in the 1960s [102], expert systems in the 1980s [103, 104], and machine learning (ML) in the 1990s [56]. Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications [91, 105–107]. For example, in [105], an artificial neural network (ANN) model is developed for a bio-diesel production process. The ANN model provided an approximation of the percentage of fatty acid methyl ester yield within $\pm 8\%$ deviation from the experimental data. Similarly, recurrent neural networks (RNN) have been broadly employed for modeling a general class of dynamical systems for control and state estimation purposes [108]. In [106], a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the predictive ability of the RNN model could outperform the first-principles model for the large-scale, complex, nonlinear process studied. This was attributed to the RNN possessing a high degree of freedom to solve the complex non-linear regression problem with the process data set.

RNN models are a powerful tool for modeling dynamic systems. Consider an RNN model that resembles the nonlinear dynamics of the system of Eq. (4.1) using m sequences of T-time-length data points $(x_{i,t}, y_{i,t})$, with $x_{i,t} \in \mathbb{R}^{d_x}$ serving as the RNN input and $y_{i,t} \in \mathbb{R}^{d_y}$ serving as the RNN output with $i = 1, ..., m$ and $t = 1, ..., T$. It is important to emphasize

that the nonlinear system's inputs, states, and outputs in Eq. (4.1) are not always represented by the RNN inputs and outputs. Hence, all the vectors for RNN models are represented in boldface to distinguish them from those of the nonlinear system of Eq. (4.1).

Moreover, to make the discussion simpler, the RNN model of Eqs. (4.4) and (4.5) is created to forecast states over a single sampling period with overall time steps $T = \Delta/h_c$ (i.e., within one sampling period Δ, the RNN model aims to predict states evolution for each integration time step h_c). Thus, the present manipulated inputs and state measurements that will be employed over $t = 1 \rightarrow T$ make up the RNN input $x_{i,t}$, while the predicted states over $t = 1 \rightarrow T$ make up the RNN output $y_{i,t}$. Owing to the sample-and-hold execution of manipulated inputs, $x_{i,t}$ does not change over $t = 1 \rightarrow T$. The data set is created of m data sequences that were individually selected from an underlying distribution over $\mathbb{R}^{d_x \times T} \times \mathbb{R}^{d_y \times T}$. To simplify the theoretical development, we consider a single-hidden-layer RNN model with the following form to approximate the nonlinear dynamics of Eq. (4.1):

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t) \tag{4.4}$$

$$\mathbf{y}_t = \sigma_y(V\mathbf{h}_t) \tag{4.5}$$

where $\mathbf{h}_t$ denotes the hidden state, and W, U, and V are the weight matrices connecting different layers. The possibly nonlinear activation functions used are denoted by σ_h and σ_y. Specifically, σ_h is often chosen to be a nonlinear activation function that may take different forms (e.g, $tanh$ or $ReLU$), while σ_y typically uses a linear element-wise activation function for regression problems. Without loss of generality, we have the following assumptions for

the development of RNN models:

Assumption 4.1. *The input data is bounded, i.e.,* $\|x_{i,t}\| \leq B_x$ *for all* $i = 1, ..., m$ *and* $t = 1, ..., T$.

Assumption 4.2. *The Frobenius norms of the weight matrices are bounded, i.e.,* $\|W\|_F \leq B_{W,F}, \|Q\|_F \leq B_{V,F}, \|U\|_F \leq B_{U,F}$.

Assumption 4.3. *The training, validation, and testing data sets are drawn from the same distribution.*

Assumption 4.4. σ_h *is a 1-Lipschitz continuous activation function, where, a function* $f : \mathbb{R}^n \to \mathbb{R}^m$ *is said to be L-Lipschitz,* $L \geqslant 0$*, if* $|f(a) - f(b)| \leq L|a - b|$ *for all* $a, b \in \mathbb{R}^n$*. Additionally,* σ_h *is a positive-homogeneous function in the sense that* $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ *holds for all* $\alpha \geq 0$ *and* $z \in \mathbb{R}$.

Furthermore, consider a hypothesis class $\mathcal{H}$ of RNN models $h(\cdot)$ that map a d_x-dimensional input $\mathbf{x} \in \mathbb{R}^{d_x}$ to a d_y-dimensional output $\mathbf{y} \in \mathbb{R}^{d_y}$. The predicted output of the RNN model and the loss function are denoted by $\mathbf{y}_t = h(\mathbf{x}_t)$ and $L(\mathbf{y}_t, \tilde{\mathbf{y}}_t)$, respectively, where $L(\mathbf{y}, \tilde{\mathbf{y}})$ calculates the squared difference between the predicted output $\mathbf{y}$ and the true output $\tilde{\mathbf{y}}$.

4.3.1 Physics-informed RNNs

From a modeling point of view, even cutting-edge black-box ML models (e.g., dense fully-connected RNN models) have had only limited success when applied in scientific domains [109] due to such models' large data size needs, failure to yield physically consistent

outputs, and lack of generalizability to unseen samples. Researchers have begun to investigate the continuum between mechanistic and ML models, in which both scientific knowledge and data are integrated in a synergistic way. This is because neither a pure ML algorithm nor solely scientific theory may be sufficient for complex scientific and engineering applications (e.g., [110–112]). A physics-based machine learning paradigm uses domain-specific knowledge, but in supporting roles such as feature engineering or post-processing, in a way fundamentally different than dominant approaches in the ML field. On the other hand, the concept of combining scientific principles and ML in developing models has only recently gained popularity [109], when there has already been a substantial amount of research done on the subject. This research direction is being conducted in various disciplines including earth systems [113], climatology [114, 115], material exploration [116, 117], quantum chemistry [118, 119], biological sciences [120], and hydrology [121]. Early findings in isolated and straightforward scenarios have been encouraging, and expectations are growing that this paradigm will speed up scientific advancement and aid in resolving some of the global challenges with regards to the environment [122], healthcare [123], and food and nutrition security [124].

Similarly, in process systems engineering, the traditional paradigm of developing numerical approaches to approximate solutions is based solely on physics—numerical differentiation and integration algorithms are used to solve systems of differential equations that reflect established physical principles through space and time [125–127]. A different approach is to look for simplified models that can roughly characterize the dynamics of the underlying systems, such as the Euler equations for gas dynamics and the Reynolds-averaged Navier-

Stokes equations for turbulent flows [128, 129]. However, creating a simplified model that accurately captures a complex phenomenon is quite difficult. More importantly, only a portion of the dynamics of many complicated real-world processes may be captured by a simple model. The equations may not accurately reflect the original system's states. On the other hand, numerous recent studies, from turbulence to reaction modeling, have demonstrated that ML-based models can produce realistic predictions and greatly speed up the simulation of complex dynamics compared to numerical solvers [130–132]. However, ML-based models are dense and purely data-driven by nature, which has many limitations. Without strict boundaries, ML-based models are likely to provide predictions that defy the fundamental principles governing physical systems. Furthermore, machine learning models frequently experience difficulties with generalization, i.e., models trained on a single data set cannot adequately adapt to unseen scenarios. Hence, approximating complicated dynamical systems in scientific applications cannot be considered to be a problem that can be easily solved by either machine-learning-based models or physics-based theory alone. There is, therefore, a significant benefit in integrating machine learning models with conventional physics-based methodologies, through which we can maximize the benefits of both techniques.

The investigation of more structured system modeling is driven by a variety of factors. In contrast to a system with structured local connectivity, fully-connected systems necessitate long-range connections, and have slower communication times between neurons. Real-world problems may have local correlations as well. It would be considerably easier and take up less memory to construct networks with organized neighborhoods than a fully-connected network [133]. Typically, when creating a dynamic model for a general nonlinear process, a neural

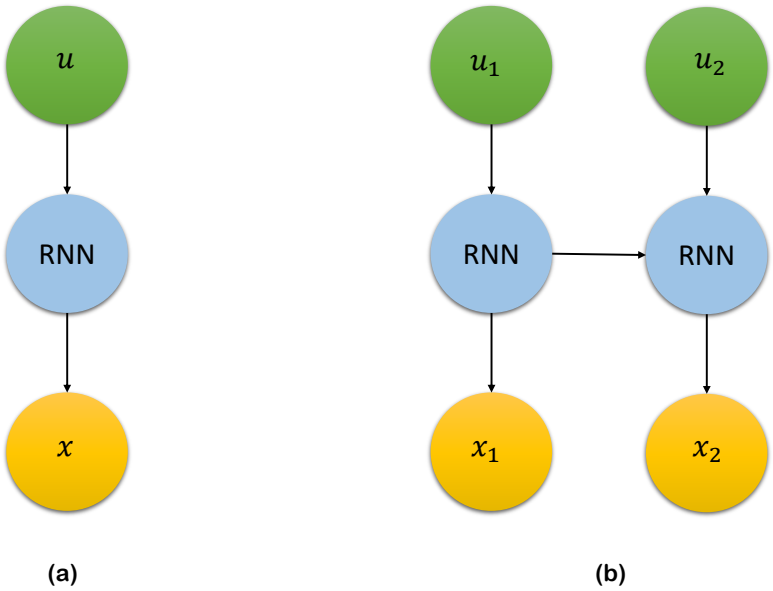

Figure 4.1: Structure of (a) standard fully-connected and (b) partially-connected RNN.

network model that uses all available process inputs to predict the desired output is preferred. Creating a fully-connected, black-box dynamic model for these processes is relatively simple with open-source machine learning tools, and such a model would attempt to capture any connections that might exist between each input and each output of the underlying process. As depicted in Fig. 4.1, at least three layers (i.e., an input layer, hidden layers, and an output layer) make up the general structure of a fully-connected RNN. For such reasons, fully-connected RNN models are frequently the best option for modeling processes where no prior knowledge is available.

Although standard RNNs do not consider any domain-specific knowledge in the model development phase and generally use fully-connected layers to capture input-output rela-

tionship using the given training dataset, it has been demonstrated in [36] that a priori process structural knowledge can be utilized to improve an RNN's performance by using a partially-connected architecture. Figure 4.1 shows the difference between fully-connected and partially-connected RNNs, from which it can be observed that the connection between some neurons is removed in a partially-connected structure to resemble the underlying input-output relationship based on a priori process structural knowledge. Partially-connected RNNs can be used to model a multiple-unit process in which upstream units affect downstream units, but not in the opposite direction. For example, consider the nonlinear system of Eq. (4.1) for which the input vector u_1 affects only the state x_1, but both u_1 and u_2 affect the state x_2, where $x = [x_1 \in \mathbb{R}^{n_{x_1}},\ x_2 \in \mathbb{R}^{n_{x_2}}]$ and $u = [u_1 \in \mathbb{R}^{n_{u_1}},\ u_2 \in \mathbb{R}^{n_{u_2}}] \in \mathbb{R}^{n_u}$ with $n_{u_1} + n_{u_2} = n_u$ and $n_{x_1} + n_{x_2} = n_x$. [36] demonstrates that, by using a partially-connected architecture, the number of weight parameters can be significantly reduced to achieve the desired model accuracy compared to a fully-connected RNN model. Additionally, in [100], an Aspen simulation study of two CSTRs in series was carried out to demonstrate that the MPC using partially-connected RNN models achieved better closed-loop performances with a reduced computational time. To better understand the benefits of partially-connected RNNs in terms of higher modeling accuracy, a theoretical analysis of generalization error needs to be carried out.

4.3.2 Long short-term memory RNN

In this subsection, we present the long short-term memory (LSTM) network. LSTM is a variant of RNNs that has been widely used to make predictions based on time series

data. Although standard RNNs have proven to be efficient in many engineering applications, RNNs struggle to deal with long time dependencies. Based on the structure of RNNs, as their hidden states are only propagated forward in time, they cannot receive future input data to predict the current state. As a result, the vanishing gradient phenomena is often encountered when training standard RNNs. As we proceed backwards through the layers of the network, the vanishing gradient problem occurs due to having exponentially decaying gradients in the loss function, which makes it harder to train the network and more challenging to retain information over longer time periods. Given these limitations, LSTM networks, with a well-known and unique structure, were introduced in 1997 [58]. Furthermore, in LSTMs, the concept of gates were introduced to keep track of how much useful history should be passed between the LSTM units. More specifically, the input gate, forget gate, and output gate control how much memory is to be stored in the cell state and retained throughout the network [134].

The LSTM is composed of several gates, and is formulated by the following equations:

$$f_t = \sigma_l(W_f \mathbf{x}_t + U_f \bar{\mathbf{h}}_{t-1}) \tag{4.6a}$$

$$r_t = \sigma_l(W_r \mathbf{x}_t + U_r \bar{\mathbf{h}}_{t-1}) \tag{4.6b}$$

$$o_t = \sigma(W_o \mathbf{x}_t + U_o \bar{\mathbf{h}}_{t-1}) \tag{4.6c}$$

$$\tilde{c}_t = \tanh(W_c \mathbf{x}_t + U_c \bar{\mathbf{h}}_{t-1}) \tag{4.6d}$$

$$c_t = f_t \odot c_{t-1} + r_t \odot \tilde{c}_t \tag{4.6e}$$

$$\bar{\mathbf{h}}_t = o_t \odot \tanh(c_t) \tag{4.6f}$$

where $W_f, W_r, W_o, W_c \in \mathbb{R}^{d_h \times d_x}$ are the weights associated with the inputs, and $U_f, U_r, U_o, U_c \in \mathbb{R}^{d_h \times d_h}$ are the weights associated with the hidden states. $r_t, f_t, o_t \in \mathbb{R}^{d_h}$ represent the input, forget, and output gates, respectively. $c_t \in \mathbb{R}^{d_h}$ is the cell state, and $\tilde{c}_t$ is the cell integrated with the input gate. σ_l is the nonlinear activation function *sigmoid*, and *tanh* is the hyperbolic tangent function. The output at time t is $\bar{\mathbf{y}}_t = \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_t)$. We note that the memory cell state c_t is one of the most important parts in LSTMs as it is the part that stores and carries the essential information for long-term and short-term dependencies. This information is then passed to the subsequent LSTM units, and is recursively updated through the other remaining gates (i.e., r_t, f_t, o_t). Specifically, the memory cell state, presented in Eq. (4.6e), consists of two terms: the first term expresses the quantity of old information to be discarded from the previous c_t, while the second term expresses the essential new information that is introduced to the memory cell state c_t [68]. Figure 4.2 shows the schematic of an LSTM cell with all its gates. In addition to the previous assumption, we also consider the following for

LSTM models:

Assumption 4.5. *The norms of weight matrices are bounded as follows:*

$$\|W_f\|_F \leq B_{W_f}\ , \|W_r\|_F \leq B_{W_r},\ \|W_o\|_F \leq B_{W_o},\ \|W_c\|_F \leq B_{W_c},\ \|U_g\|_F \leq B_{U_g},$$

$$\|U_r\|_F \leq B_{U_r}, \|U_o\|_F \leq B_{U_o}, \|U_{\bar{h}}\|_F \leq B_{U_{\bar{h}}}, \|Z\|_{1,\infty} \leq B_Z$$

Assumption 4.6. *The nonlinear activation functions $\sigma, \sigma_{\bar{y}}$ are 1-Lipschitz continuous, and $\sigma(0) = \sigma_{\bar{y}}(0) = 0$.*

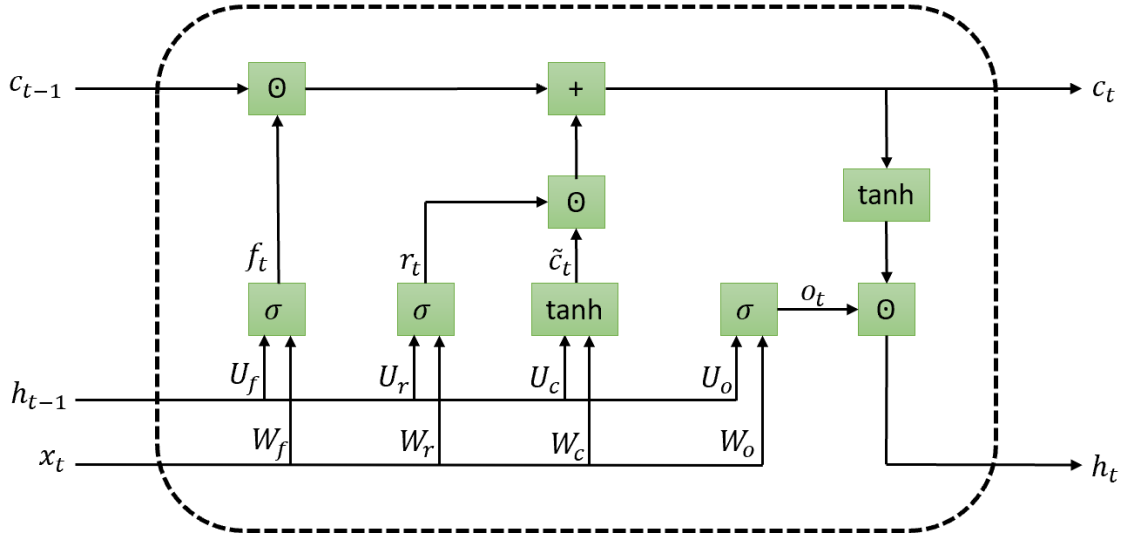

Figure 4.2: Schematic of an LSTM cell structure.

4.4 Generalization error

4.4.1 General considerations

Generalizability or generalization error is a metric that measures a machine learning model's ability to adapt to new, previously unseen data that is drawn from the same distribution as the one used to train the model. Several investigations were conducted for the interpretation and improvement of generalization of different machine learning-based models (e.g., [135, 136]). Furthermore, a theoretical analysis of the generalization error is of significant importance as it provides a fundamental understanding on how well the model performs on unseen data that will be collected in real-world systems. This section will provide derivations of the generalization error for RNNs using statistical learning theory. Before we present the results on generalization error bounds, we first introduce the necessary definitions of generalization error.

Definition 4.1. *A centered random variable $x \in \mathbb{R}$ is said to be sub-Gaussian with variance proxy σ^2 if $\mathbb{E}[x] = 0$ and the moment generating function satisfies:*

$$\mathbb{E}[\exp(aX)] \leq \exp\left(\frac{a^2\sigma^2}{2}\right), \forall a \in \mathbb{R} \tag{4.7}$$

Definition 4.2. *Given a data distribution, D, and a function, h, that predicts y (output) based on x (input), the generalization error is given by*

$$\mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y)\ \rho(x, y)\ dxdy. \tag{4.8}$$

where $\rho(x, y)$ denotes the joint probability distribution for x and y, and Y and X represent the vector space for all possible outputs and inputs, respectively.

Definition 4.3. *$L(\cdot, \cdot)$ is the loss function (e.g., mean squared error (MSE) for regression problems). Since the distribution may be unknown, the following empirical error is often used as an approximation measure for the generalization error:*

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m} \sum_{i=1}^{m} L(h(x_i), y_i) \tag{4.9}$$

where $S = (s_1, ..., s_m)$, $s_i = (x_i, y_i)$ includes m data samples drawn from the data distribution D.

Definition 4.4. *Given a set of data samples $S = \{s_1, ..., s_m\}$, and a hypothesis class $\mathcal{F}$ of real-valued functions, the definition of the empirical Rademacher complexity of $\mathcal{F}$ is*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i f(s_i) \right] \tag{4.10}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_m)^T$, and ϵ_i are Rademacher random variables that are independent and identically distributed (i.i.d.) and satisfy $\mathbb{P}(\epsilon_i = -1) = \mathbb{P}(\epsilon_i = 1) = 0.5$.

The following lemma gives the generalization error bound for a general class of RNN models.

Lemma 4.1. *Consider a hypothesis class $\mathcal{H}$ of vector-valued functions $h \in \mathbb{R}^{d_y}$ and a set of data samples $S = \{s_1, ..., s_m\}$. Let $L(.)$ be a L_r -Lipschitz function mapping $h \in \mathbb{R}^{d_y}$ to R.*

Then,

$$\mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \epsilon_i L(h(\mathbf{x}_i), \mathbf{y}_i) \right] \leq \sqrt{2} L_r \mathbb{E}_\epsilon \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \epsilon_{ik} h(\mathbf{x}_i) \right] \tag{4.11}$$

where $h_k(.)$ is the k^{th} component in the vector-valued function $h(.)$, and ϵ_{ik} is an $m \times d_y$ matrix of independent Rademacher variables. In the following text, we will omit the subscript ϵ of the expectation operator for simplicity.

Since the right-hand side of the previous inequality is generally difficult to compute, we can reduce it to scalar classes and derive the following bound:

$$\mathbb{E} \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \epsilon_{ik} h(\mathbf{x}_i) \right] \leq \sum_{k=1}^{d_y} \mathbb{E} \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \epsilon_i h(\mathbf{x}_i) \right] \tag{4.12}$$

where $\mathcal{H}_k, k = 1, ..., d_y$ are classes of scalar-valued functions that correspond to the components of vector-valued functions in $\mathcal{H}$. The previous inequality will later be used in the derivation of the generalization error bound for LSTM models.

Lemma 4.2 (c.f. Theorem 3.3 in [82])**.** *Let $\mathcal{H}$ be the hypothesis class of ML models that map $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_t\} \in \mathbb{R}^{d_x \times t}$ (i.e., the first t time step inputs) to $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ (i.e., the t^{th} output) and $\mathcal{G}_t$ be the loss function set with $\mathcal{H}$,*

$$\mathcal{G}_t = \{ g_t : (\boldsymbol{x}, \tilde{\boldsymbol{y}}) \to L(h(\boldsymbol{x}), \tilde{\boldsymbol{y}}), h \in \mathcal{H} \} \tag{4.13}$$

where $\tilde{\boldsymbol{y}}$ and $\boldsymbol{x}$ are the true output vector and the input vector of the ML model, respectively. Then, given a data set consisting of m i.i.d. data samples, the inequality below holds in

probability for all $g_t \in \mathcal{G}_t$ *over the data samples* $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$*:*

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m} \sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{4.14}$$

Eq. (4.14) demonstrates that the upper bound for the generalization error depends on the training error (first term), the Rademacher complexity of $\mathcal{G}_t$ (second term), and a function of the samples size m and the confidence δ. Therefore, to derive a generalization error bound for RNN models, an upper bound for the Rademacher complexity of RNN hypotheses needs to be developed.

Lemma 4.3. *Given a hypothesis class* $\mathcal{H}_k$ *of real-valued functions corresponding to the* k^{th} *component of the vector-valued function class* $\mathcal{H}$ *and a set of* m *i.i.d. data samples* $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, ..., m$*, the following inequality holds for the scaled empirical Rademacher complexity:*

$$\begin{aligned} m\mathcal{R}_s(\mathcal{H}_k) &= \mathbb{E}\left[\sup_{h \in \mathcal{H}_k} \sum_{i=1}^{m} \epsilon_i h(\mathbf{x}_i)\right]. \\ &= \frac{1}{\lambda} \log\ \exp\left(\lambda \mathbb{E}\left[\sup_{h \in \mathcal{H}_k} \sum_{i=1}^{m} \epsilon_i h(\mathbf{x}_i)\right]\right) \\ &\leq \frac{1}{\lambda} \log\ \mathbb{E}\left[\sup_{h \in \mathcal{H}_k} \exp\left(\lambda \sum_{i=1}^{m} \epsilon_i h(\mathbf{x}_i)\right)\right] \end{aligned} \tag{4.15}$$

where $\lambda > 0$ *is an arbitrary parameter.*

Lemma 4.4. *Let* $\mathcal{H}_{k,t}$, $k = 1, ..., d_y$ *be the class of real-valued functions that corresponds to the* k^{th} *component of the RNN output at* t^{th} *time step, with weight matrices and activation functions satisfying Assumptions 1–4. Given a set of* m *i.i.d. data samples* $S = (x_{i,t}, y_{i,t})^T -$

$t = 1$, $i = 1, ..., m$, *the following equation holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \leq \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}} \tag{4.16}$$

where $M = B_{V,F}B_{W,F}\frac{B_{U,F}^t - 1}{B_{U,F}^{-1}}$, *and* B_X *is the upper bound for RNN inputs.*

Lemma 4.5 (c.f. Theorem 1 in [21])**.** *Given a dataset* $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^T$ *with i.i.d. data samples,* $i = 1, ..., m$, *and the* L_r*-Lipschitz loss function class* $\mathcal{G}_t$ *associated with the RNN function class* $\mathcal{H}_t$ *that predicts outputs at the* t^{th} *time step, with probability at least* $1-\delta$ *over* S, *the following inequality holds for the RNN models:*

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{4.17}$$

The above equation represents the theoretical generalization error's upper bound for RNN models. This theory will be utilized to find a relationship between a RNN model and its structure in Section 4.4.2.

4.4.2 Physics-based RNNs generalization bound

In a partially-connected structure, the connections between inputs and outputs should be carefully designed to reflect a *priori* physical knowledge. In particular, as illustrated in Fig. 4.3, x_2 does not affect y_1, so the weights corresponding to the linkages between x_2 and y_1 (dashed lines in Fig. 4.3) are assigned a value of zero (i.e., $w_{i,j} = v_{l,j} = 0$). This structure superiority in accuracy and model identification to dense fully-connected RNNs has been demonstrated through several works. Hence, we develop the following theory to interpreter

this observation.

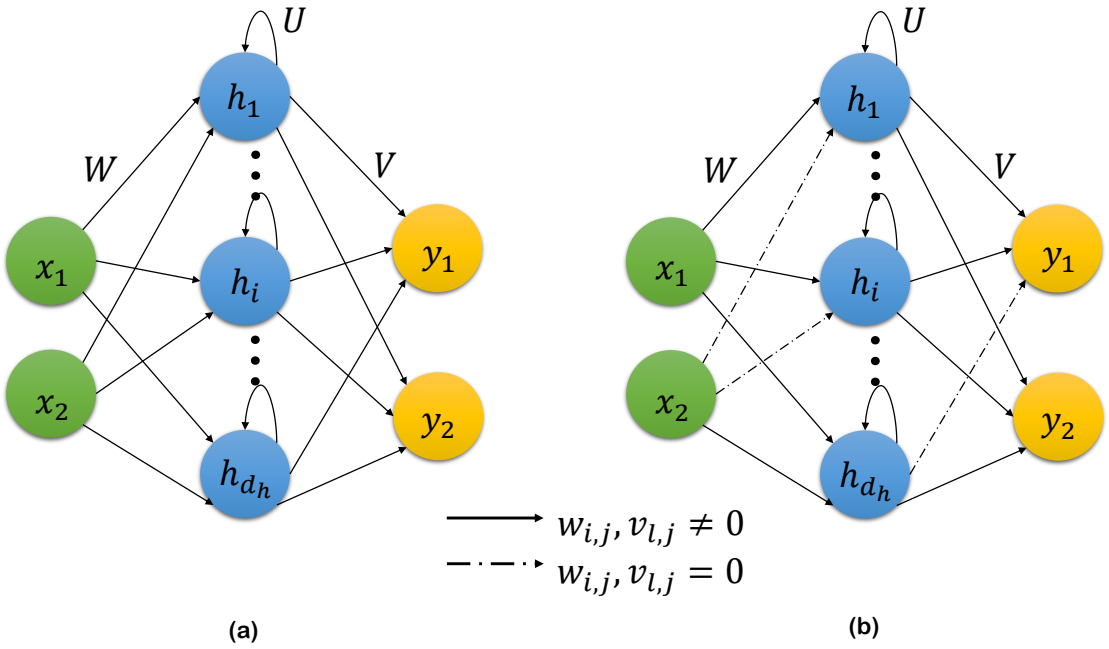

Figure 4.3: Weights and connections in (a) standard fully-connected and (b) partially-connected RNN structures, where zeroed weights for links between units are represented by dashed lines.

Theorem 4.1. *Consider the following inequalities:* $\mathbb{E}_{\nu}[g_t(\mathbf{x}, \mathbf{y})] \leq \nu$ *and* $\mathbb{E}_{\hat{\nu}}[g_t(\mathbf{x}, \mathbf{y})] \leq \hat{\nu}$, *where* ν *and* $\hat{\nu}$ *represent the generalization error bound for a fully-connected RNN model and a partially-connected RNN model, respectively. Given that both models are constructed with the same hyperparameters and trained over the same i.i.d data set with* m *samples, the following inequality holds:*

$$\hat{\nu} < \nu \tag{4.18}$$

Proof. If we let ν denote the right-hand side of Eq. 4.17, ν can be represented as the sum of the three terms in the right-hand side of Eq. 4.17 i.e., $\nu = \nu_I + \nu_{II} + \nu_{III}$, where the

subscripts I, II, and III are the term indices, and the same applies for $\hat{\nu}$ i.e., $\mathbf{E}_{\hat{\nu}}[g_t(\mathbf{x}, \mathbf{y})] \leq \hat{\nu} = \hat{\nu}_I + \hat{\nu}_{II} + \hat{\nu}_{III}$. Then, with respect to the first terms, $\hat{\nu}_I$ and ν_I, they depend on the sizes of both the training data set and the hypothesis class $\mathcal{H}$. Due to the dense structure of FCRNN models, the size of the hypothesis class $\mathcal{H}$ will be larger, which leads to a higher probability of convergence to the optimal hypothesis h^*. On the contrary, by incorporating physical knowledge into the RNN modeling by assigning some weight entries to be zero, the size of the hypothesis class $\mathcal{H}$ is reduced, yet the model can be closer to the optimal hypothesis h^* for the data distribution D. Thus, both models will have close values for the first term (i.e., $\nu_I \approx \hat{\nu}_I$). Additionally, since we are developing the two models using the same data set with m samples, the second terms for both the PCRNN model and the FCRNN model are approximately equal (i.e., $\nu_{II} \approx \hat{\nu}_{II}$). Therefore, we are left to investigate the third term, which is given by:

$$\nu_{III} = \mathcal{O}\left(L_r d_y \frac{M B_X (1 + \sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{4.19a}$$

$$\hat{\nu}_{III} = \mathcal{O}\left(L_r d_y \frac{\hat{M} B_X (1 + \sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{4.19b}$$

where $M = B_{V,F} B_{W,F}$, and $\hat{M} = B_{\hat{V},F} B_{\hat{W},F}$.

Note that M and $\hat{M}$ are products of the RNN weight matrix bounds in Eq. (4.17), where the PCRNN model weight matrices are denoted by the symbols $\hat{V}$ and $\hat{W}$, and their Frobenius norm bounds are $B_{\hat{V},F}$ and $B_{\hat{W},F}$, respectively. After training both FCRNN and PCRNN models with the same random initialization and optimization algorithm, the weight

matrices in the PCRNN model will have some zero entries, while the other entries (i.e., the nonzero ones) would be numerically close for both models. Since the Frobenius norm of matrix A is expressed as the square root of the matrix trace of $AA^{(H)}$, where $A^{(H)}$ is the conjugate transpose, more zero entries in the weight matrices will yield lower bounds on their Frobenius norms i.e.,

$$B_{\hat{W},F} < B_{W,F} \tag{4.20a}$$

$$B_{\hat{V},F} < B_{V,F} \tag{4.20b}$$

which yields

$$\hat{\nu}_{III} < \nu_{III} \tag{4.20c}$$

Hence, this proves that the partially-connected RNN modeling approach provides a lower generalization error bound than the dense fully-connected RNN architecture. □

Remark 4.1. *By incorporating process structural knowledge into the development of partially-connected RNN models, the complexity of RNN hypothesis class is reduced compared to fully-connected RNNs, which leads to a tighter bound on the Rademacher complexity. Additionally, by revealing the correct direction for RNNs to find the optimal weight parameters, the training error (the first term in Eq.* (4.17)*) is more likely to be minimized using the same hyperparameters (i.e., the number of layers and neurons) and the same training set of m i.i.d. data samples.*

4.4.3 LSTM Generalization Error

LSTM networks structure and complexity are different from standard fully-connected RNN models as well as partially-connected RNNs. Hence, a generalizability theoretical framework of LSTMs is discussed in this subsection. We recall the fundamental definitions and lemmas from the Section 4.4.1. In addition we present the following theoretical bases (i.e., remarks and lemmas) needed for the development of a LSTM network's generalization accuracy bound.

Remark 4.2. *From the definitions of norms, if given a vector $b \in \mathbb{R}^n$, the following inequalities hold:*

$$\|b\|_\infty = \max\{|b_i|\} \leq \sqrt{\sum_i^n |b_i|^2} = \|b\| \tag{4.21}$$

$$\|b\|_\infty = \max\{|b_i|\} \leq \sum_i^n |b_i| = \|b\|_1 \tag{4.22}$$

Lemma 4.6. *Given a hypothesis class $\mathcal{H}$ of vector-valued functions that map the LSTM inputs $\mathbf{x} \in \mathbb{R}^{d_x}$ to the hidden states $\bar{\mathbf{h}} \in \mathbb{R}^{d_h}$, and any convex and monotonically increasing function $p : R \to [0, \infty)$, the following inequality holds for the LSTM model of Eq. 4.6 with a 1-Lipschitz activation function $\sigma_y(0) = 0$, applied element-wise:*

$$\mathbb{E}\left[\sup_{\bar{h}\in\mathcal{H}, \|Z\|_{1,\infty}\leq B_Z} p\left(\lambda\|\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\|_\infty\right)\right] \leq 2\cdot\mathbb{E}\left[\sup_{\bar{h}\in\mathcal{H}} p\left(B_Z\|\sum_{i=1}^m \epsilon_i \bar{\mathbf{h}}_i\|_\infty\right)\right] \tag{4.23}$$

where $\|Z\|_{1,\infty}$ is the maximal 1-norm of its rows.

Proof. Based on the previous works of [21, 46], we proceed with this proof as follows:

$$\begin{aligned}
&\mathbb{E} \sup_{\|Z\|_{1,\infty} \leq B_Z} p\left(\| \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i) \|_\infty \right) \\
&= \mathbb{E} \sup_{\|z_k\|_1 \leq B_Z} \max_j \; p\left(\left| \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}} \left(z_j^T \bar{\mathbf{h}}_i \right) \right| \right) \\
&= \mathbb{E} \sup_{\|z\|_1 \leq B_Z} p\left(\left| \sum_{i=1}^{m} \epsilon_i \sigma_y \left(z^T \bar{\mathbf{h}}_i \right) \right| \right)
\end{aligned} \tag{4.24}$$

where z_k is the k-th row of the matrix Z. Since p is a convex monotonically increasing function, $p(|x|) \leq p(x) + p(-x)$, and hence, Eq. (4.24) can be bounded by:

$$\begin{aligned}
&\mathbb{E} \sup_{\|Z\|_{1,\infty} \leq B_Z} p\left(\| \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i) \|_\infty \right) \\
&\leq \mathbb{E} \sup_{\|z\|_1 \leq B_Z} p\left(\sum_{i=1}^{m} \epsilon_i \sigma_y \left(z^T \bar{\mathbf{h}}_i \right) \right) + \mathbb{E} \sup_{\|z\|_1 \leq B_Z} p\left(-\sum_{i=1}^{m} \epsilon_i \sigma_y \left(z^T \bar{\mathbf{h}}_i \right) \right)
\end{aligned} \tag{4.25}$$

Note that ϵ_i follows a symmetric distribution, i.e. $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = 0.5$. Hence, Eq. (4.25) becomes:

$$\begin{aligned}
&\mathbb{E} \sup_{\|Z\|_{1,\infty} \leq B_Z} p\left(\| \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i) \|_\infty \right) \\
&= 2\mathbb{E} \sup_{\|z\|_1 = B_Z} p\left(\sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}} \left(z^T \bar{\mathbf{h}}_i \right) \right) \\
&\leq 2\mathbb{E} \sup_{\|z\|_1 = B_Z} p\left(\sum_{i=1}^{m} \left(\epsilon_i z^T \bar{\mathbf{h}}_i \right) \right) \\
&\leq 2\mathbb{E} \sup_{\|z\|_1 = B_Z} p\left(\|z\|_\infty \| \sum_{i=1}^{m} (\epsilon_i \bar{\mathbf{h}}_i) \|_\infty \right)
\end{aligned} \tag{4.26}$$

Using Eq. (4.22) in Remark 4.2, the inequality in Eq. (4.26) is bounded as follows:

$$
\begin{aligned}
&\mathbb{E} \sup_{\|Z\|_{1,\infty} \leq B_Z} p\left(\| \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i) \|_\infty \right) \\
&\leq 2\mathbb{E} \sup_{\|z\|_1 = B_Z} p\left(\|z\|_1 \| \sum_{i=1}^{m} (\epsilon_i \bar{\mathbf{h}}_i) \|_\infty \right) \\
&\leq 2\mathbb{E} \sup\ p\left(B_Z\ \| \sum_{i=1}^{m} (\epsilon_i \bar{\mathbf{h}}_i) \|_\infty \right)
\end{aligned}
\tag{4.27}
$$

This completes the proof of Lemma 4.6, where it "peels off" the matrix Z between the LSTM hidden layer and the output layer. □

Remark 4.3. *It is well established that the activation functions sigmoid and* tanh *are essential for the development of different types of neural networks, including LSTMs. This is due to the LSTM network's special structure and the importance of the gating functionality performed by sigmoid functions. Therefore, the purpose of utilizing the ℓ_∞-norm, which is based on the peeling strategy as in Lemma 4.6, is to eliminate the requirement of the positive homogeneity property in activation functions. Hence, this lemma suits neural networks requiring non-positive and non-homogeneous activation functions.*

Remark 4.4. *In LSTMs, signals are categorized into two main types. The first type are propagating signals, which are $\bar{\mathbf{h}}_t$ and c_t. These signals connect the LSTM cells in sequential time steps with each other. In other words, the LSTM unit at a certain time step t, receives as an input the hidden state $\bar{\mathbf{h}}_{t-1}$ and the cell state c_{t-1}, both from the previous time step $t-1$, in addition to $\mathbf{x}_t$, which is the input vector at time t. The second type of signals are gating signals, represented by o_t, f_t, and r_t, which are responsible for the flow of information inside the LSTM unit. Their outputs range from 0 to 1, which means they determine how*

much of the information is passed. In the extreme cases, if the output is 0, this means no information is passed, and if the output is 1, this means all the information is passed. Taking this into consideration, these gating signals can be upper-bounded by 1, for simplicity in the generalization error bound proof.

Lemma 4.7. *Let $\mathcal{H}_{k,t}, k = 1, ..., d_y$ be the class of real-valued functions that corresponds to the k^{th} component of the LSTM output at the t^{th} time step, with weight matrices and activation functions satisfying Assumptions 4.1–4.6. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, \ldots, m$, the following equation holds for the Rademacher complexity:*

$$m\mathcal{R}_s(\mathcal{H}_{k,t}) \leq \left(\frac{\sqrt{2}+\sqrt{t}}{\sqrt{2}}\right) \bar{M}\sqrt{m} \tag{4.28}$$

where $\bar{M} = B_V B_{W_c} B_x \frac{1-\bar{\beta}^t}{1-\bar{\beta}}$ and $\bar{\beta} = 1 + B_{U_c}$.

Proof. By investigating previous works of [21, 47], we proceed with this proof as following: Recalling that z_k is the k^{th} row of the weight matrix Z and by using Eq. (4.16) in lemma 4.3, we obtain the following bound for the Rademacher complexity $m\mathcal{R}_s(\mathcal{H}_{k,t})$:

$$\begin{aligned} m\mathcal{R}_s(\mathcal{H}_{k,t}) &= \mathbb{E}\left[\sup \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(z_k \bar{\mathbf{h}}_{i,t})\right] \\ &\leq \frac{1}{\lambda} \log \mathbb{E}\left[\sup \exp\left(\lambda \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(z_k \bar{\mathbf{h}}_{i,t})\right)\right] \end{aligned} \tag{4.29}$$

Now, by applying the peeling strategy of Eq. (4.23), we can further bound the previous inequality as follows:

$$
\begin{aligned}
m\mathcal{R}_s(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\left\|\sum_{i=1}^{m}\epsilon_i\bar{\mathbf{h}}_{i,t}\right\|_\infty\right)\right] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\bar{\mathbf{h}}_{i,t}\|_\infty\right)\right] \\
&= \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|_\infty\|\bar{\mathbf{h}}_{i,t}\|_\infty\right)\right]
\end{aligned}
\tag{4.30}
$$

Using Eq. (4.6f) from the LSTM equations and applying Eq. (4.21) from Remark 4.2, we expand the propagation signal $\bar{\mathbf{h}}_{i,t}$ and get the following bound:

$$
\begin{aligned}
m\mathcal{R}_s(\mathcal{H}_{k,t}) &= \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\|o_{i,t}\|_\infty\|\tanh(c_{i,t})\|\right)\right] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\|o_{i,t}\|_\infty\|c_{i,t}\|\right)\right]
\end{aligned}
\tag{4.31}
$$

Following Remark 4.4, we can bound the gating signal $o_{i,t}$ as follows: $\|o_{i,t}\|_\infty \leq 1$. Also, we apply Eq. (4.21) from Remark 4.2 and further expand the propagation signal $c_{i,t}$ using Eq. (4.6e) based on the LSTM structure, thereby obtaining the following inequality:

$$
m\mathcal{R}_s(\mathcal{H}_{k,t}) \leq \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\left(\|f_{i,t}\|_\infty\|c_{i,t-1}\| + \|r_{i,t}\|_\infty\|\tilde{c}_{i,t}\|\right)\right)\right] \tag{4.32}
$$

We further expand $\tilde{c}_{i,t}$ using Eq. (4.6d) and Remark 4.2. Subsequently, we expand $\bar{\mathbf{h}}_{i,t-1}$ using Eq. (4.6f). Moreover, using the upper bound of the input data and the upper bounds of the weight matrices mentioned in assumption 4.1 and assumption 4.5, respectively, the

following bound is obtained:

$$
\begin{aligned}
m\mathcal{R}_s(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\
&\quad + \|r_{i,t}\|_\infty\|\tanh(W_c x_{i,t} + U_c\bar{\mathbf{h}}_{i,t-1})\|)\Bigg)\Bigg] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\
&\quad + \|r_{i,t}\|_\infty\|W_c x_{i,t} + U_c\bar{\mathbf{h}}_{i,t-1}\|)\Bigg)\Bigg] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\
&\quad + \|r_{i,t}\|_\infty(B_{W_c}B_x + B_{U_c}\|\bar{\mathbf{h}}_{i,t-1}\|))\Bigg)\Bigg] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\
&\quad + \|r_{i,t}\|_\infty(B_{W_c}B_x + B_{U_c}\|o_{i,t-1}\|_\infty\|c_{i,t-1}\|))\Bigg)\Bigg] \\
&= \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|((\|f_{i,t}\|_\infty \\
&\quad + \|r_{i,t}\|_\infty\|o_{i,t-1}\|_\infty B_{U_c})\|c_{i,t-1}\| + B_{W_c}B_x\|r_{i,t}\|_\infty)\Bigg)\Bigg]
\end{aligned}
\tag{4.33}
$$

From the LSTM formulation, Eq. (4.6), and the fact that the LSTM's gating signals can be bounded by 1 (i.e., $\|f_{i,t}\| \leq 1$, $\|r_{i,t}\| \leq 1$, and $\|o_{i,t-1}\| \leq 1$), Eq. (4.33) can be further bounded as follows:

$$
m\mathcal{R}_s(\mathcal{H}_{k,t}) \leq \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|((1+B_{U_c})\|c_{i,t-1}\| + B_{W_c}B_x)\right)\right] \tag{4.34}
$$

By expanding the term $\|c_{i,t-1}\|$ recursively, the above inequality reaches:

$$\begin{aligned} m\mathcal{R}_s(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|B_{W_c}B_x\left(1+\beta+\beta^2+...\right)\right)\right] \\ &= \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|B_{W_c}B_x\sum_{p=0}^{t-1}\beta^p\right)\right] \end{aligned} \tag{4.35}$$

We apply the formula for the sum of a geometric series, $\sum_{p=0}^{t-1}\bar{\beta}^p = \frac{1-\bar{\beta}^t}{1-\bar{\beta}}$ to obtain the following:

$$m\mathcal{R}_s(\mathcal{H}_{k,t}) = \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|_2 B_{W_c}B_x\frac{1-\bar{\beta}^t}{1-\bar{\beta}}\right)\right] \tag{4.36}$$

where $\bar{\beta} = 1 + B_{U_c}$

Let $q = \bar{M}\sum_{i=1}^{m}\|\epsilon_i\|$, where $\bar{M}$ is the product of some weight matrices and contains a fraction involving $\bar{\beta}$, specifically $\bar{M} = B_V B_{W_c} B_x \frac{1-\bar{\beta}^t}{1-\bar{\beta}}$. Notice that the Rademacher complexity variables ϵ_i are the elements giving rise to randomness in the bound. Then, the Rademacher complexity bound in inequality 4.36 becomes:

$$\begin{aligned} m\mathcal{R}_s(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda q)] \\ &= \frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda(q-\mathbb{E}[q]))] + \mathbb{E}[q] \end{aligned} \tag{4.37}$$

Using Jensen's inequality, $\mathbb{E}[q]$ is bounded as follows:

$$\begin{aligned} \mathbb{E}[q] &= \mathbb{E}\left[\bar{M}\sum_{i=1}^{m}\|\epsilon_i\|\right] \\ &\leq \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1}^{m}\|\epsilon_i\|^2\right]} \\ &= \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1,\bar{i}=1}^{m}\epsilon_i^T\epsilon_{\bar{i}}\right]} = \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1,\bar{i}=1}^{m}(1)\right]} = \bar{M}\sqrt{m} \end{aligned} \tag{4.38}$$

We can show q is sub-Gaussian with the following variance factor v, since q satisfies a bounded-difference condition with respect to its random variables ϵ_i , i.e., $q(\epsilon_1, ..., \epsilon_i, ..., \epsilon_m)$ $- q(\epsilon_1, ..., -\epsilon_i, ..., \epsilon_m) \leq 2\bar{M}\|\mathbf{x}_{i,t}\|$:

$$v = \frac{1}{4}\sum_{i=1}^{m}(2\bar{M}\|\mathbf{x}_{i,t}\|)^2 = \bar{M}^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\| \tag{4.39}$$

According to the property of sub-Gaussian random variables in Definition 4.1, the following inequality holds for q:

$$\frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda(q - \mathbb{E}[q]))] \leq \frac{\lambda\bar{M}^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|}{2} \tag{4.40}$$

Assuming $\lambda = \frac{\sqrt{2t}}{\bar{M}\sqrt{\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|^2}}$, the Rademacher complexity can be bounded as the following:

$$\begin{aligned} m\mathcal{R}_s(\mathcal{H}_{k,t}) &\leq \frac{\lambda\bar{M}^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|}{2} + \bar{M}\sqrt{m} \\ &= \frac{\bar{M}\sqrt{mt}}{\sqrt{2}} + \bar{M}\sqrt{m} \\ &= \frac{\sqrt{2}+\sqrt{t}}{\sqrt{2}}\bar{M}\sqrt{m} \end{aligned} \tag{4.41}$$

□

Theorem 4.2. *Let $\mathcal{G}_t$ be the family of loss function associated with the hypothesis class $\mathcal{H}_t$ of vector-valued functions that map the LSTM inputs to the LSTM output at t^{th} time step, with weight matrices and activation functions satisfying Assumptions 4.1–4.6. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, ..., m$, with probability at least $1 - \delta$ over S,*

we have:

$$\mathbb{E}[g_t(\mathbf{x}, \boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + \mathcal{O}\left(L_r d_y \frac{\left(\sqrt{2}+\sqrt{t}\right)\bar{M}}{\sqrt{m}}\right) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \tag{4.42}$$

where $\bar{M} = B_Z B_{W_c} B_x \frac{1-\bar{\beta}^t}{1-\bar{\beta}}$ *and* $\bar{\beta} = 1 + B_{U_c}$.

Proof.

Using the inequalities obtained in lemma 4.1, we can derive the following upper bound for the loss function $L(\bar{\mathbf{h}}(\mathbf{x}_i), \mathbf{y}_i)$, with $\bar{\mathbf{h}}(\mathbf{x}_i)$ being vector-valued functions, as follows:

$$\begin{aligned}
\mathcal{R}_S(\mathcal{G}_t) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \epsilon_i L(\bar{\mathbf{h}}(\mathbf{x}_i), \mathbf{y}_i)\right] &\leq \sqrt{2} L_r \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{d_y} \epsilon_{ik} \bar{\mathbf{h}}_k(\mathbf{x}_i)\right] \\
&\leq \sqrt{2} L_r d_y \frac{\left(\sqrt{2}+\sqrt{t}\right)\bar{M}\sqrt{m}}{\sqrt{2m}} \\
&= L_r d_y \frac{\left(\sqrt{2}+\sqrt{t}\right)\bar{M}}{\sqrt{m}}
\end{aligned} \tag{4.43}$$

Plugging the Rademacher complexity bound that we just obtained, i.e., $\mathcal{R}_S(\mathcal{G}_t) \leq L_r d_y \frac{\left(\sqrt{2}+\sqrt{t}\right)\bar{M}}{\sqrt{m}}$, into inequality 4.14 mentioned in lemma 4.2, we get the following inequality of Theorem 4.2:

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + \mathcal{O}\left(L_r d_y \frac{\left(\sqrt{2}+\sqrt{t}\right)\bar{M}}{\sqrt{m}}\right) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \tag{4.44}$$

□

Remark 4.5. *By observing the three different terms in the generalization error bound of Eq.* (4.42)*, it appears that several ways to reduce the generalization error may be considered. The first approach is to design a LSTM network such that the value of the empirical loss*

$\frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$ *over the training data samples is minimized. Additionally, noticing that the number of training samples* m *occurs in the denominator of both the second and third terms of Eq.* (4.42)*, one can consider increasing the number of training samples* m *and, as a result, the value of the generalization error should decrease. Also, it is important to note that the complexity hypothesis class may affect the value of the generalization error, in the sense that increasing the complexity hypothesis class results in an increase in the values of the weight matrices bounds* M *and, subsequently, an increase in the second term* $\mathcal{O}$ *in Eq.* (4.42)*. Hence, when designing the LSTM network, we consider starting with a simple design and, based on the training and testing performance, we can increase the complexity such that it improves the model performance for a given application without causing overfitting to the data.*

4.5 RNN/LSTM based model predictive control

In this section, we integrate an RNN/LSTM model into a Lyapunov-based model predictive controller (LMPC) formulation. In particular, the partially-connected modeling of RNN/LSTM is executed as discussed in [100] and then employed as a predictive model to provide state estimation to solve the optimization problem of the LMPC, which is expressed

in the following form:

$$\mathcal{J} = \min_{u \in \mathsf{S}(\Delta)} \int_{t_k}^{t_k+\mathsf{P}} L(\tilde{x}(t), u(t))dt \tag{4.45a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{4.45b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_k + \mathsf{P}) \tag{4.45c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{4.45d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k)),$$

$$\text{if } x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}} \tag{4.45e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_k + \mathsf{P}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{4.45f}$$

where $\mathsf{S}(\Delta)$ denotes a set of piecewise constant functions with period Δ, $\tilde{x}$ is the state trajectory predicted by the RNN/LSTM model, and P is the prediction horizon expressed as a multiple of the sampling period (i.e., $\mathsf{P} = N \times \Delta$, $N > 0$). The time-derivative of the Lyapunov function V in Eq. (4.45e) is given as $\dot{V}(x, u)$, i.e., $\frac{\partial V(x)}{\partial x} F_{nn}(x, u)$. During the prediction horizon $t \in [t_k, \, t_k + \mathsf{P})$, the LMPC computes the optimum input sequence $u^*(t)$ and delivers the first control signal $u^*(t_k)$ to the system to be implemented for the following sampling period. After that, at the following sampling interval, the LMPC receives new data and is resolved with updated state estimations. Furthermore, the MPC optimization problem's goal is to minimize the integral of $L(\tilde{x}(t), u(t))$, given in Eq. (4.45a), which represents the cost function over the prediction horizon while satisfying the constraints of Eqs. (4.45b) to (4.45f). The RNN/LSTM model from Eq. (4.45b) is used to forecast the evolution of the closed-loop state trajectory $\tilde{x}(t_k)$ under the MPC, and its initial conditions are updated

according to Eq. (4.45d), where $x(t_k)$ is the last state measurement. The input constraints are expressed in Eq. (4.45c), and they are imposed across the prediction horizon.

To ensure the stability of the closed-loop system, when $x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is the target region, the condition of Eq. (4.45e) is triggered. As a result of this constraint, the Lyapunov function of the closed-loop state declines, and the state approaches the steady-state within a finite period of time. Eventually, when the state $x(t_k)$ arrives to $\Omega_{\rho_{nn}}$, the predicted closed-loop state will be kept within this region for the duration of the prediction horizon via the constraint of Eq. (4.45f). Following section 2.3, the controller $\Phi_{nn}(x)$ was developed with the intent of ensuring that the origin of the RNN/LSTM system is exponentially stable.

A well-conditioned RNN or LSTM model with sufficient model accuracy can be readily produced when utilizing noise-free data for training. Therefore, the closed-loop state is guaranteed to be bound inside the predefined stability region Ω_ρ during the simulation time and will finally converge to a small region around the origin via application of the RNN/LSTM-based LMPC of Eq. (4.45) for the regulation of the nonlinear system of Eq. (4.1). This is true provided that the modeling error, which is given by $|\upsilon| = |F(x,u) - F_{nn}(x,u)|$, is sufficiently small [15, 137].

4.6 Application

A chemical process example is used for demonstrating the anticipated improvements associated with physics-informed modeling of RNNs. Particularly, two non-isothermal continuous-stirred tank reactors (CSTR) in sequence with ideal mixing are taken into consideration, as

shown in Fig. 4.4, with each reactor containing an irreversible second-order exothermic reaction, where a raw material A is transformed to a product B (i.e., A $\rightarrow$ B). The feed flow rate to each reactor F_{i_o} contains only chemical A with initial concentration and temperature C_{A,i_o} and T_{i_o}, respectively, where $i = 1, 2$ is the reactor index. Each reactor has a heating jacket that delivers or removes heat at a rate of Q_i. The dynamical model describing the two CSTRs is derived from material and energy balance equations in the form of the following ODEs:

$$\frac{dC_{A,1}}{dt} = \frac{F_{1_o}}{V_1}(C_{A1_o} - C_{A,1}) - k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 \tag{4.46a}$$

$$\frac{dT_1}{dt} = \frac{F_{1_o}}{V_1}(T_{1_o} - T_1) + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 + \frac{Q_1}{\rho C_p V_1} \tag{4.46b}$$

$$\frac{dC_{A,2}}{dt} = \frac{F_1}{V_2} C_{A,1} + \frac{F_{2_o}}{V_2} C_{A,2_o} - \frac{F_1 + F_{2_o}}{V_2} C_{A,2} - k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 \tag{4.46c}$$

$$\frac{dT_2}{dt} = \frac{F_{2_o}}{V_2} T_{2_o} + \frac{F_1}{V_2} T_1 - \frac{F_1 + F_{2_o}}{V_2} T_2 + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 + \frac{Q_2}{\rho C_p V_2} \tag{4.46d}$$

The notations $C_{A,i}$, T_i, and Q_i represent the reactant A concentration, reactor temperature, and the heat supply rate, respectively. V_i is the volume of the reacting liquid, which has a density of ρ and a heat capacity of C_p that are constants for both reactors. ΔH, k_o, R, and E denote the reaction's enthalpy, pre-exponential constant, ideal gas constant, and activation energy, in the same order, and these parameters are unchanged for both reactors. Process parameter values are listed in Table 4.1.

The manipulated inputs for this process are the heat supply rate to both reactors (i.e., Q_1 and Q_2), which are represented in deviation form from their steady-state values as $u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$. The upper and lower physical bounds on the inputs are

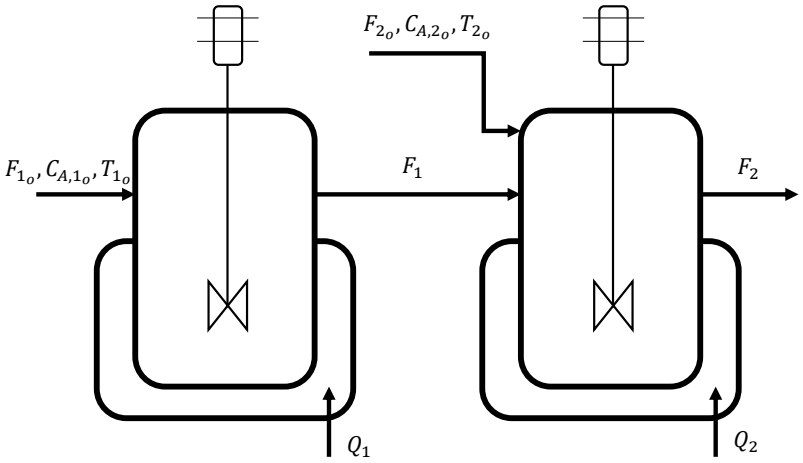

Figure 4.4: Two continuous-stirred tank reactors in series.

Table 4.1: Parameter and steady-state values for the CSTR

$C_{A,1_s} = 1.95\ kmol/m^3$	$T_{1s} = 402\ K$
$C_{A,1_o} = 4\ kmol/m^3$	$T_{2s} = 402\ K$
$C_{A,2_s} = 1.95\ kmol/m^3$	$Q_{1s} = 0.0\ kJ/h$
$C_{A,2_o} = 4\ kmol/m^3$	$Q_{2s} = 0.0\ kJ/h$
$T_{1_o} = 300\ K$	$T_{2_o} = 300\ K$
$F_{1_o} = 5\ m^3/h$	$F_{2_o} = 5\ m^3/h$
$V_1 = 1\ m^3$	$V_2 = 1\ m^3$
$k_o = 8.46 \times 10^6\ m^3/kmolh$	$E_A = 5 \times 10^4\ kJ/kmol$
$R = 8.314\ kJ/(kmol\ K)$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$\rho = 1000\ kg/m^3$	$C_p = 0.231\ kJ/(kg\ K)$

$[U^{\max}, U^{\min}] = [5, -5] \times 10^5\ kJ/h$, respectively. The states are also represented in deviation fashion from their steady-state values as $[x_1, x_2, x_3, x_4] = [C_{A,1} - C_{A,1_s}, T_1 - T_{1s}, C_{A,2} - C_{A,2_s}, T_2 - T_{2s}]$, such that the origin is the equilibrium point of the state-space representation of the underlying system.

4.6.1 Data generation and RNN models construction

Large data sets are necessary for the development of machine-learning-based models, and, generally speaking, the larger the data set size, the more accurate the model can be [54], provided that the data is independent and identically distributed. Large data sets are also accessible from a variety of sources, including industries, pilot plants and laboratories, and computer-based simulations. Industrial data is typically not accessible to the general public, and collecting data from pilot plants and laboratory studies is both expensive and time-consuming. Hence, we use extensive open-loop simulations in our work to create our data set.

For the development of the RNN model, the procedures for data generation, neural network training, and validation are described. The explicit Euler method with an integration time step of $h_c = 5 \times 10^{-4}\ h$ is used to numerically simulate the dynamic model of Eq. (4.46) for one sampling period Δ under various initial conditions (a total of 3000 different combinations of initial conditions). Particularly, MATLAB is used to create a data set of size m_{data}. The data set is then split into two matrices: an output matrix with x_1, x_2, x_3, and x_4 as outputs at $t = t_k + \Delta$ and an input matrix with u_1, u_2, x_1, x_2, x_3, and x_4 at $t = t_k$.

Subsequently, by using the generated data and the Keras library, two RNN models are constructed, where each of the models has two hidden layers with 30 neurons in each, and hyperbolic tangent (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) as the activation function for all layers except for the input and output layers. The activation function in the output layer is set to be linear. The links between the layers is untouched in the fully-connected RNN modeling, while, on

the other hand, the inputs are fed to different layers in the partially connected RNN modeling in a manner that reflects the physical structure of the underlying process. Specifically, the partially-connected RNN model is developed following the algorithm discussed in [100].

Using input information from the previous sampling interval, we forecast the evolution of the states for the subsequent 0.01 hr (the equivalent of one sampling time Δ). We use the Adam optimizer, a combination of RMSprop and gradient descent with momentum optimization techniques, as opposed to the conventional gradient descent optimization process. Additionally, we perform five-fold cross-validation on the RNN models in order to produce more reliable models, and select the models with the lowest validation MSE. In addition, we use five, different (i.e., no repetition) testing data sets, as shown in Fig. 4.5, to test the developed models. The generalization error for each testing data set is illustrated in Fig. 4.6, where the partially-connected RNN model yielded higher generalization accuracy (i.e., less error). These results aligned with Theorem 1.

4.6.2 Open-loop simulation

Before incorporating the generated models into closed-loop tests, open-loop simulations are essential to check that the predictive models can estimate the future trajectory adequately. Hence, we carried out open-loop simulations, illustrated in Fig. 4.7, where the time-varying inputs are randomly chosen. Furthermore, from the figure, it can be noticed that the state trajectories predicted by the partially-connected RNN model are all closer to the true state trajectories (denoted by FP) than the states predicted by the fully-connected RNN model.

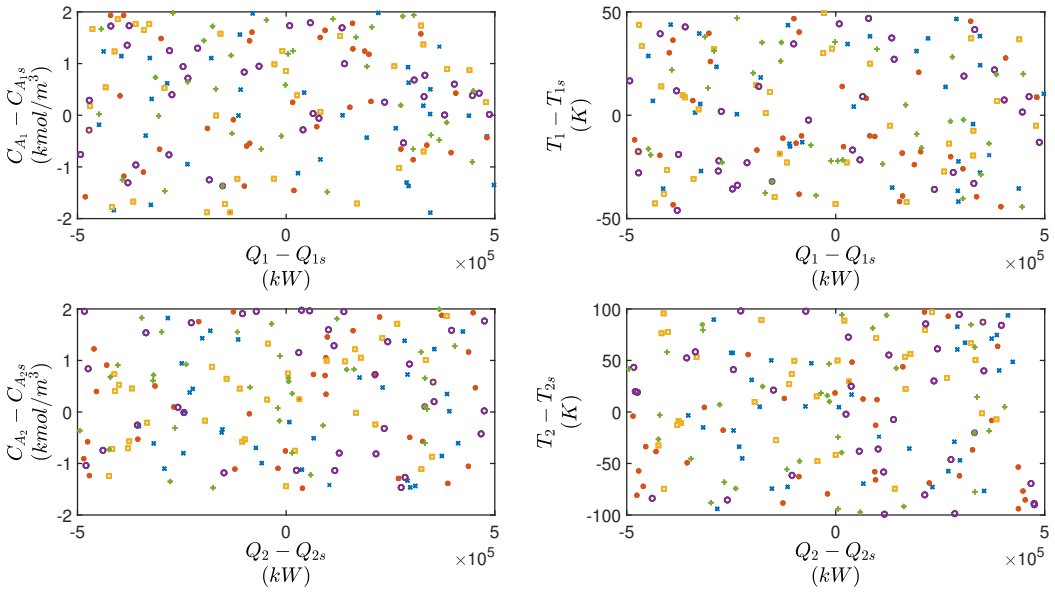

Figure 4.5: Five different testing data sets, where each marker indicates a single set.

Table 4.2 presents the open loop simulation MSEs between the predicted states from each RNN model architecture and the corresponding first-principles model outputs as the ground-truth process output value. From Table 4.2, the ratios of fully-connected RNN MSE to the partially-connected RNN MSE for x_1, x_2, x_3, and x_4 are 6.05, 2.3991, 4.3018, and 10.3013, receptively. All the ratios are greater than one, which implies that the partially-connected RNN architecture yields a higher accuracy for state estimation. Furthermore, the open-loop responses initiated close to the steady state and predicted by the partially-connected and the fully-connected RNN models under a step change only in u_2 are depicted in Fig. 4.8. The figure demonstrates that the partially-connected RNN model captures the process dynamics better, as the trajectory of the first reactor's temperature was not altered by this change. All these results indicate that both RNN models provide reasonable

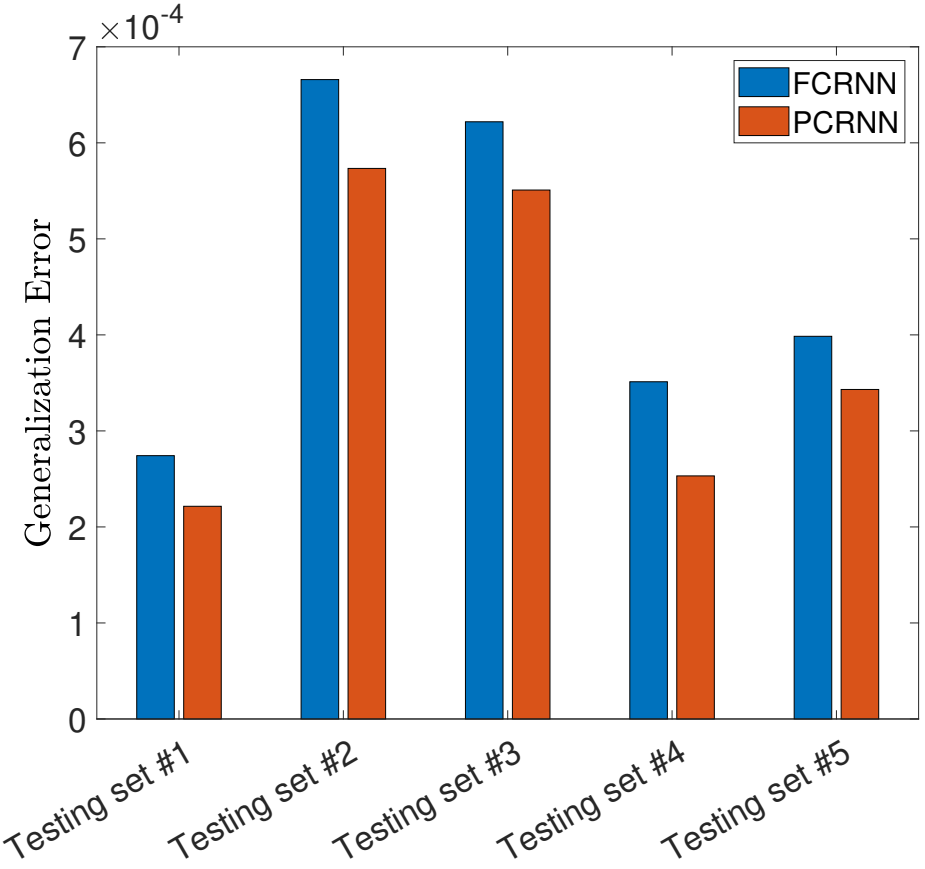

Figure 4.6: Generalization error for five different testing data sets, where PCRNN and FCRNN stand for partially-connected RNNs (orange bars) and fully-connected RNNs (blue bars), respectively.

prediction, yet, the partially-connected RNN model approximates the underlying process model more accurately.

Table 4.2: Open-loop prediction results (MSE)

State	Modeling architecture	
	FCRNN	PCRNN
x_1	0.0065	0.0011
x_2	125.4551	52.2929
x_3	0.0134	0.0031
x_4	156.3076	15.1736

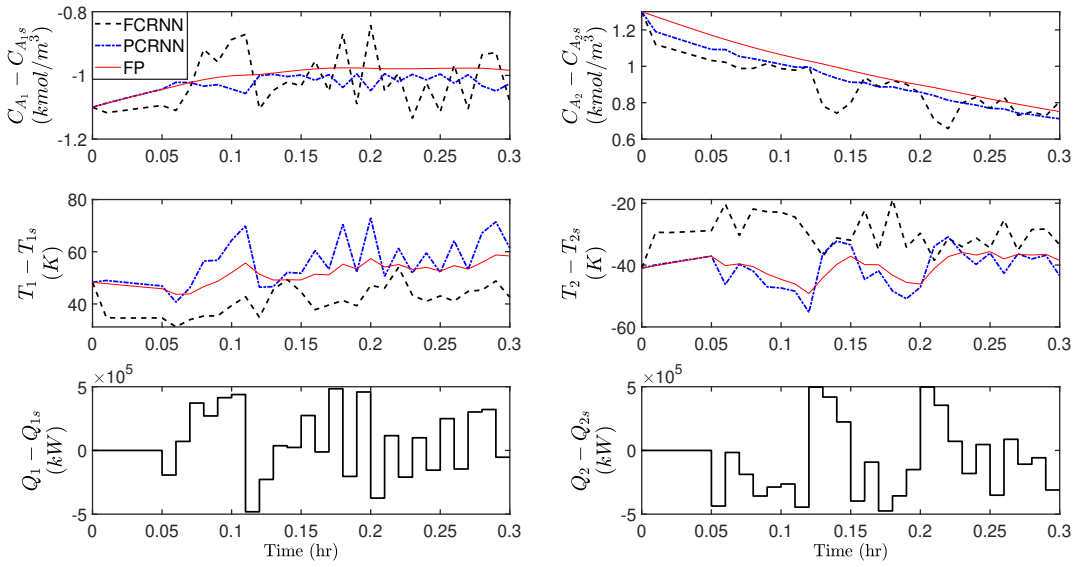

Figure 4.7: Time-varying profiles of the states and inputs for the second open-loop simulation under random time-varying inputs using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN model (black line).

4.6.3 Closed-loop simulation

Next, in order to run closed-loop simulations with the certainty that both RNN models give high accuracy approximation for the process outputs, we design LMPCs based on the fully-connected RNN model and the partially-connected RNN model, respectively. For each sampling period, the nonlinear minimization problem of the LMPC is solved using the Python front-end of the interior point optimizer (IPOPT) software [138]. For the purpose of solving complex nonlinear optimization problems, this optimizer is an open source program. It uses an interior point line search filter technique to try to locate a local solution to a nonlinear mathematical program. The LMPC objective function is defined as $L(x,u) = x^T \mathbf{Q}\, x + u^T \mathbf{R}\, u$, where $\mathbf{Q}$ and $\mathbf{R}$ are diagonal penalty matrices for the set-point error and control actions,

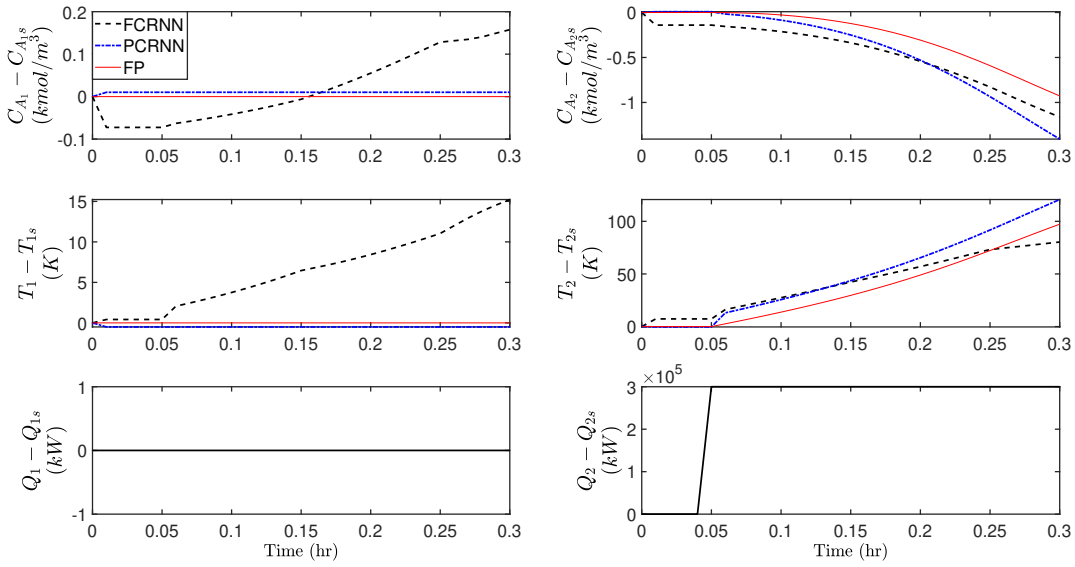

Figure 4.8: Time-varying profiles of the states and inputs for the open-loop simulation under a step change in u_2 using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN model (black line).

respectively. The two matrices critically impact the performance of the LMPC and require proper tuning. MPC tuning guidelines discussed in [139] are followed. Lastly, we choose $V(x) = x^T P x$ as the Lyapunov function, where P is a positive definite matrix obtained by applying a grid search.

Under the LMPCs, we perform closed-loop simulations initiating from two different initial conditions, and the results are shown in Figs. 4.9 and 4.10. From the figures, both LMPCs, each based on its predictive RNN model, were able to drive the states to the steady-state values and to stabilize the system within a small neighborhood around the origin. However, the partially-connected RNN-based LMPC yielded better performance in terms of state trajectories being smoother and not exhibiting fluctuation around the steady

state. Moreover, the MSE of each state corresponding to the two RNN architectures are calculated for the two closed-loop simulations and presented in Table 4.3. As it can be noted from the table, the partially-connected RNN model yielded a more reliable controller with smaller MSE values by an order of magnitude compared to the fully-connected RNN model. Due to the fully-connected RNN's deterioration in the prediction accuracy caused by the assumption that every input influences every potential output, these results are expected.

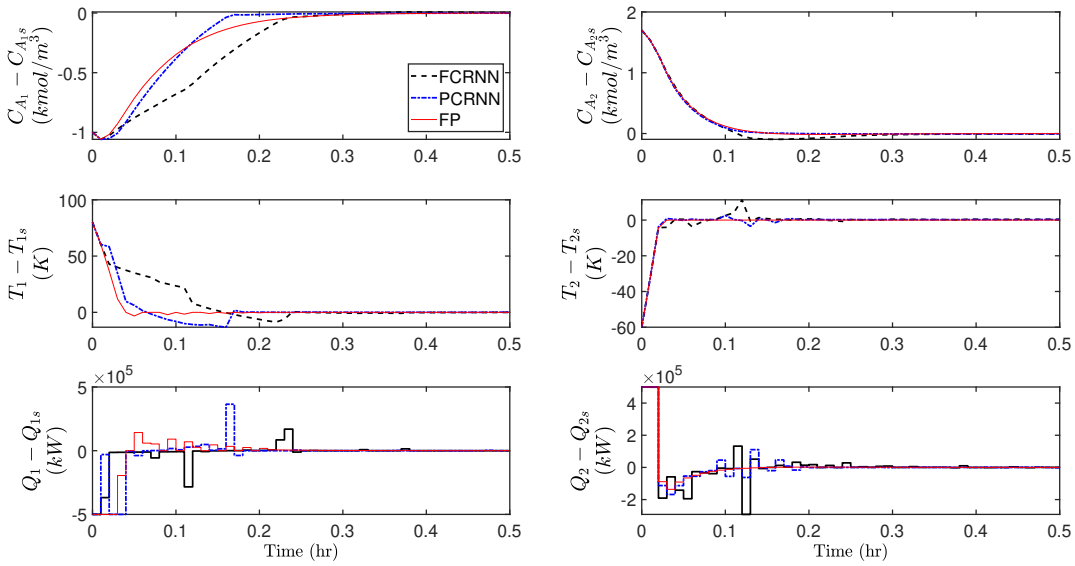

Figure 4.9: State and input profiles of the first closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

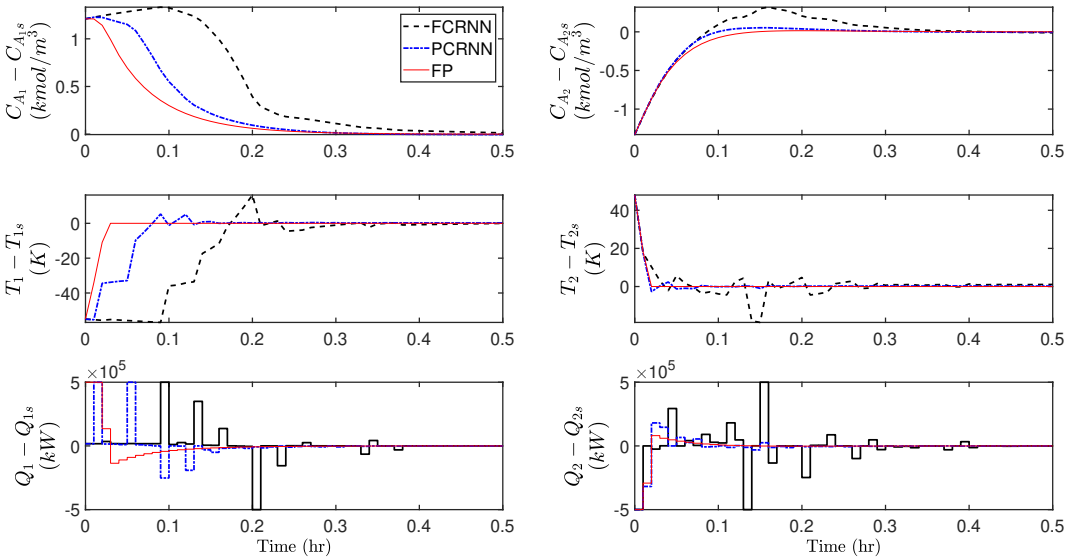

Figure 4.10: State and input profiles of the second closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

Table 4.3: Closed-loop prediction results (MSE)

State	1st Closed-loop Simulation		2nd Closed-loop Simulation	
	FCRNN	PCRNN	FCRNN	PCRNN
x_1	0.020705316	0.002051591	0.2411215725	0.025175541
x_2	170.280344	39.13188574	596.3087136	88.72292971
x_3	0.001812249	0.000111788	0.015817003	0.001061295
x_4	3.788903947	0.511854559	20.3473683	0.400205264

Chapter 5

Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems

5.1 Introduction

Various applications in the field of chemical engineering involve systems that exhibit different time scales. Instances of such systems include biochemical processes, catalytic reactors and distillation columns. Furthermore, other scientific sectors such as power electronics, communication networks, and biological systems also feature systems with dynamics evolving in disparate time scales. Specifically, in the context of chemical engineering, researchers have utilized the method of singular perturbation to decouple two-time-scale systems into reduced-order subsystems, each associated with a distinct time-scale [e.g., 140]. This approach simplifies the analysis of the ordinary differential equations governing the system,

allowing for the design of a suitable well-conditioned control law that stabilizes the system. In the context of control system design, model predictive control (MPC) is widely recognized as one of the leading and convenient approaches employed in stabilizing different types of nonlinear systems. MPC is essentially an optimization problem formulated with a well-defined objective function aimed to enhance the performance of the system, while meeting constraints associated with the system's physical structure and closed-loop stability. The fact that MPC accounts for multiple inputs, outputs, and constraints within its framework is what makes it a suitable choice for designing control systems to stabilize chemical processes. However, in singularly perturbed systems, the presence of distinct time scales can lead to challenges when employing MPC without accounting for the state evolution in the different time scales. Neglecting this aspect can result in degradation of the closed-loop performance, long time consumption or, in more critical situations, instability and stiffness in the control system due to issues with the controller's effectiveness and the presence of ill-conditioning [20].

However, if the time-scale multiplicity is accounted for, MPC has proven to be an efficient control strategy when it comes to situations involving time-scale separation. For instance, the research conducted by [141] presented the design of a composite control system which included two MPC designs, one for the slow subsystem and one for the fast subsystem. Using stability analysis and the theory of singular perturbations, this work investigated and guaranteed the closed-loop stability for the full, nonlinear two-time-scale system under Lyapunov-based tracking MPC. To additionally consider process economics, a similar strategy was then proposed in the work of [142], which focused on designing a composite

controller involving two MPC problems. Specifically, for stability of the fast subsystem (and full two-time-scale system as a result), a Lyapunov-based tracking MPC was used for the fast states based on the error between the fast states and the slow manifold using a quadratic cost function, while a Lyapunov-based economic MPC was used for the slow subsystem to optimize economic considerations and achieve any other desirable closed-loop stability properties using a non-quadratic, economic cost function. While the work of [141, 142] focused on first-principles-based controllers, in the work of [143], a two-time-scale system was separated into the slow and fast subsystems using singular perturbation strategy and then modeled using only data from the system via the method of sparse identification for nonlinear dynamics. Subsequently, an MPC was designed based on the reduced-order slow subsystem modeled using sparse identification, and closed-loop stability guarantees were derived. Furthermore, numerical simulations were used to demonstrate the reduced computational cost of the reduced-order sparse-identified model.

Due to the vast advancements in the chemical industry and the multitude of complex reactions occurring in real-life chemical process applications, engineers encounter challenges when constructing first-principles models for these chemical processes. As a result, in recent years, machine learning (ML) models have been utilized to approximate the dynamics of chemical processes. These models are highly beneficial when designed carefully as they offer a reliable and efficient alternative to classical first-principles models and can then be incorporated into MPC frameworks. Many works have been conducted in this field; for instance, the work of [49] offered essential insights and fundamental theory regarding the integration of machine learning techniques into MPC schemes. The work focused on designing a

recurrent neural network (RNN) model to approximate the nominal nonlinear system and integrated the model within an MPC framework to stabilize the system. In [144], polynomial nonlinear-autoregressive-with-exogenous-inputs (NARX) models are used to build nonlinear MPC, with the relevant polynomial terms to retain selected via sparse regression. The proposed MPC was applied to a multi-input-multi-output chemical reactor, and an algebraic modeling language was used to reduce computational times.

Another research direction termed "approximate MPC" involves the replacement of the MPC optimization problem by a closed-form expression that approximates the MPC control actions without having to solve an optimization problem. This is meant to address the computational challenges inherent in classical MPC at the expense of a lower accuracy and generalizability. In [145], an approximate MPC was designed for heating, ventilation, and air-conditioning (HVAC) systems due to the stringent computational resources available for building control systems. Specifically, the MPC optimization problem was imitated by a recurrent neural network with a structure of nonlinear autoregressive network with exogenous inputs trained with data from 30 days of operation in closed-loop under the MPC. While both the MPC and approximate MPC greatly reduced the cooling consumption of the HVAC system when tested, the approximate MPC solved for the control actions over 100 times faster throughout the testing period, with only a 12% degradation in performance compared to the MPC. An alternate research direction to reduce the computational burden of MPC is the design of "explicit MPC", in which the optimal control actions are computed off-line using multiparametric programming methods, allowing the online component of the MPC to have to only search a table of linear gains and compute a single function evaluation.

A summary of studies investigating explicit MPC is provided in [146]. Machine learning has also been investigated in the context of explicit MPC. In [147], for example, an explicit control law was learned for discrete-time linear time-invariant systems using machine learning. Specifically, the key challenge addressed was that of high dimensionality, which has been attempted to be solved in the existing literature by identifying suboptimal polytope partitions of the state space and designing control laws based on such regions. [147] focused on tackling high-dimensionality in a similar manner but by using reinforcement learning and additionally investigated constraint satisfaction and feasibility guarantees for the explicit MPC. Specifically, by using deep neural networks with rectified linear units as the activation function and a modified policy gradient algorithm based on prior knowledge, the objectives were met and demonstrated via three numerical examples of varying levels of dimensionality and complexity. A potential limitation of the approximate and explicit MPC approaches is that they are practically applicable to only smaller-scale systems or even though they are applicable to systems with very fast sampling times [146]. However, for general nonlinear systems and nonlinear MPC, due to the highly nonconvex optimization problem, methods such as the use of suboptimal polytopes [147] may not be readily extended to the nonlinear case.

While ML techniques have shown great success when incorporating them into MPC schemes, practical application of machine learning-based MPC schemes to real-life processes poses significant challenges. The fact that the ML model is developed using a finite number of data samples makes it challenging to assess the accuracy of the model when considering generalized scenarios. Therefore, researchers have developed methods to quantify the error

of the process model used in MPC and study their effects on closed-loop performance and stability. For example, [148] proposed the use of Bayesian neural networks (BNN) to quantify the plant-model mismatch and subsequently create adaptive scenarios in real-time for scenario-based MPC using the BNN. Based on simulation results from a cold atmospheric plasma system, the proposed approach outperformed scenario-based MPC without adaptive scenarios and adaptive-scenario-based MPC with Gaussian process regression. In more theoretical fronts, researchers have also studied the concept of generalization error bounds, which are crucial to study in order to ensure the construction of reliable models that are capable of performing accurate predictions on unseen or new data. Specifically, a low generalization error bound guarantees the effectiveness and high performance of the ML model and, consequently, the MPC utilizing the designed ML model. To address this, [21] derived a generalization error bound for a fully-connected RNN (FCRNN) model, discussed the main factors that affect the bound, and integrated the FCRNN model into an MPC scheme to stabilize a nonlinear process. Moreover, the work by [149] focused on deriving a generalization error bound for a feed-forward neural network (FNN) that was used to construct a control Lyapunov-barrier function. The work by [150] studied the generalization error bound for both a partially-connected RNN (PCRNN) and an LSTM-RNN and subsequently carried out a comparison study between the generalization performance of an FCRNN versus a PCRNN. Additionally, all the aforementioned works incorporated the designed machine learning models into Lyapunov-based MPC schemes to demonstrate the ability of the LMPCs to stabilize a chemical process. To the best of our knowledge, the application of generalization error bounds to neural networks modeling two-time-scale systems has not been investigated.

In light of the above considerations, in this chapter, we use the theory of statistical machine learning to construct the generalization error bounds for neural networks modeling two-time-scale systems. Moreover, the computational time for a neural network-based MPC is known to be an important practical consideration. However, the computational time for an RNN-based MPC with and without decomposition into lower-order subsystems has not been studied. Therefore, we introduce two LMPC frameworks, one designed based on an RNN model that predicts the slow state vector and the other designed based on an RNN model that models the full two-time-scale system, and compare the two LMPC schemes in terms of closed-loop stability and computational time. The rest of the chapter is structured as follows: Section 5.2 introduces essential preliminaries as well as the general class of two-time-scale systems considered in this work. Section 5.3 discusses the generalization error bounds of neural networks modeling two-time-scale systems. In Section 5.4, we present a machine learning-based Lyapunov-based MPC using an RNN that predicts the slow state vector, followed by closed-loop stability analysis and results. In Section 5.5, a chemical process example is used to demonstrate the effectiveness of the designed controller.

5.2 Preliminaries

5.2.1 Notation

The Euclidean norm of a vector is denoted by $|\cdot|$. The transpose of the vector x is given by x^T. Given a matrix B, its Frobenius norm is denoted as $\|B\|_F$. Moreover, the infinity norm of 1-norms of the columns of B is denoted as $\|B\|_{1,\infty} = \max_j \sum_i |B_{i,j}|$. The expression

$L_F V(x)$ denotes the standard Lie derivative $L_F V(x) := \frac{\partial V(x)}{x} f(x)$. Set subtraction is denoted by '\', i.e., $A \backslash B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be L-Lipschitz continuous, if there exists $L \geq 0$ such that, $|f(a) - f(b)| \leq L|a - b|$ for all $a, b \in \mathbb{R}^n$. A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ belongs to class $\mathcal{KL}$ if, for each constant value of t, the function $\beta(\cdot, t)$ is of class $\mathcal{K}$, and for each constant value of s, the function $\beta(s, \cdot)$ is decreasing and approaches zero as $s \to \infty$. The probability that the event A will occur is denoted as $\mathbb{P}(A)$. Additionally, the expected value of a random variable X is denoted as $\mathbb{E}[X]$.

5.2.2 Class of systems

The following family of ordinary differential equations describes the general class of two-time-scale continuous-time nonlinear systems with k states considered in this work:

$$\dot{x} = f_1(x, z, u, \epsilon) \tag{5.1a}$$

$$\epsilon \dot{z} = f_2(x, z, \epsilon) \tag{5.1b}$$

where $x \in \mathbb{R}^n$ is the slow state vector and $z \in \mathbb{R}^p$ is the fast state vector, with $n + p = k$. $u \in \mathbb{R}^{q_1}$ is the bounded manipulated input vector. The input vector u is constrained by $u \in U := \{|u_i| \leq u_{1_{\max,i}}, i = 1, ..., q_1\}$. We assume the vector functions $f_1(x, z, u, \epsilon)$ and $f_2(x, z, \epsilon)$ to be sufficiently smooth vector functions in $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively. Moreover,

we assume that the closed-loop stability region of the nonlinear system of Eq. (5.1) is defined by the region $\Omega_{\rho_{_F}}$, where $\Omega_{\rho_{_F}} := \{x \in D | V(x,z) \leq \rho_{_F}\}$ where $\rho_{_F} > 0$ and D is an open neighborhood around the origin. The speed ratio of the slow to the fast dynamics of the system is represented by the small positive parameter ϵ. In Eq. (5.1b), we observe that the speed ratio ϵ pre-multiplies the derivative of the fast state vector z, which enables us to utilize a well-known strategy called "reduced-order modeling" via singular perturbations. In this approach, we will be able to decompose the system of Eq. (5.1) into two different reduced-order subsystems. We follow the strategy thoroughly illustrated in [20].

The following equations are obtained through setting $\epsilon = 0$ in Eq. (5.1):

$$\dot{\bar{x}} = f_1(\bar{x}, \bar{z}, \bar{u}, 0) \tag{5.2a}$$

$$0 = f_2(\bar{x}, \bar{z}, 0) \tag{5.2b}$$

where $\bar{x}$ and $\bar{z}$ are the slow state vector and the fast state vector associated with the system of Eq. (5.1) under the case where $\epsilon = 0$, respectively. Additionally, the following assumption is considered to be fundamental in the theory of singular perturbations.

Assumption 5.1. *Eq.* (5.2b) *has an isolated solution, which is determined by,*

$$\bar{z} = \bar{f}_2(\bar{x}) \tag{5.3}$$

where $\bar{z}$ is a quasi-steady state for the fast state z, and the function $\bar{f}_2 : \mathbb{R}^n \to \mathbb{R}^p$ is continuously differentiable.

We substitute Eq. (5.3) into Eq. (5.2a) to obtain the reduced-order slow subsystem,

$$\dot{\bar{x}} = f_1(\bar{x}, \bar{f}_2(\bar{x}), \bar{u}, 0) \tag{5.4}$$

Furthermore, to obtain the dynamics of the reduced-order fast subsystem, we introduce a fast timescale $\tau = t/\epsilon$ and a new coordinate $\bar{\bar{z}} = z - \bar{z}$ to re-write Eq. (5.1b) with respect to the newly introduced variables $\bar{\bar{z}}$ and τ. Then, the fast subsystem can be expressed as the derivative of $\bar{\bar{z}}$ with respect to τ while setting $\epsilon = 0$,

$$\frac{\mathrm{d}\bar{\bar{z}}}{\mathrm{d}\tau} = f_2(\bar{x}, \bar{\bar{z}} + \bar{f}_2(\bar{x}), 0) \tag{5.5}$$

Starting at t_0, the initial conditions of the state vectors x and z are given by the vectors x_0 and z_0, respectively. Based on Eq. (5.1) to Eq. (5.5), and through utilizing the basic theoretical concepts of two-time-scale systems in [20], the following equation will hold for all $t \in [t_p, T]$, where $t_p > t_0$

$$z(t) = \bar{z}(t) + \mathcal{O}(\epsilon). \tag{5.6}$$

where $\bar{z}(t)$ reflects the slow transient of z, and $\mathcal{O}(\epsilon)$ is an error of order epsilon. A variable, $z(t)$, is $\mathcal{O}(\epsilon)$ where ϵ is a positive constant, if there exists a positive constant $\bar{k}$ (independent of ϵ) such that $|z(t)| \leq \bar{k}\epsilon$. Furthermore, if we were to consider the time interval where $t \in [t_0, T]$, Eq. (5.6) would be slightly modified such that the approximation of the state z is given by the following:

$$z(t) = \bar{z}(t) + \bar{\bar{z}}(t) + \mathcal{O}(\epsilon), \tag{5.7}$$

where $\bar{z}(t)$ and $\bar{\bar{z}}(t)$ are the slow and fast transients of z, respectively. Additionally, the approximation of state x by $\bar{x}$ is given by the following equation, for all $t \in [t_0, T]$

$$x(t) = \bar{x}(t) + \mathcal{O}(\epsilon). \tag{5.8}$$

The fast subsystem of Eq. (5.5) needs to satisfy certain stability properties in order for the above closeness of solutions estimates to hold true, which are described by the following assumption.

Assumption 5.2. *The equilibrium $\bar{\bar{z}}(\tau) = 0$ of Eq. (5.5) demonstrates uniform asymptotic stability in x_0 and t_0. In addition, $z_0 - \bar{z}(t_0)$ resides within its domain of attraction. As a result, for all $\tau \geq 0$, $\bar{\bar{z}}(\tau)$ exists.*

If Assumption 5.2 holds true, then

$$\lim_{\tau \to \infty} \bar{\bar{z}}(\tau) = 0, \tag{5.9}$$

This implies that, at some time $t_p > t_0$, z will approach close to its quasi-steady state $\bar{z}$. The local stability of the equilibrium of the fast subsystem holds if the following verifiable condition holds.

Assumption 5.3. *All the eigenvalues of $\frac{\partial f_2}{\partial z}$, computed for $\epsilon = 0$, along $\bar{x}(t), \bar{z}(t)$, exhibit real parts less than a constant negative value, i.e.*

$$\Re\lambda \left\{ \frac{\partial f_2}{\partial z} \right\} \leq -c < 0. \tag{5.10}$$

Utilizing the aforementioned assumptions, we establish the well-known "Tikhonov's theorem" in the following theorem.

Theorem 5.1. *(c.f. Theorem 3.1 in [20]). Consider Assumptions 5.2 and 5.3 hold true. Then, for all* $t \in [t_0, T]$*, Eqs.* (5.7) *and* (5.8) *are valid. In addition, there exists a specific time instant* $t_p \geq t_0$ *such that Eq.* (5.6) *is valid for all* $t \in [t_p, T]$*.*

Several sources such as [151] are useful to analyze slightly varied formulations of the Theorem 5.1 and to review its proof. At this point, we have investigated several essential concepts of two-time-scale systems. We will assume that the error between x and $\bar{x}$ is not greater than $\mathcal{O}(\epsilon)$. This assumption allows us to simplify the analysis by approximating $\bar{x}$ as x. Similarly, the same principle applies to z and $\bar{z}$, with our focus on capturing the slow transient of the fast state dynamics z, that is $\bar{z}$. Therefore, based on this assumption, we will proceed to express the reduced order slow subsystem of Eq. (5.4) as the following throughout the manuscript,

$$\dot{x} = F(x, u) := f(x) + g(x)u, \qquad x(t_0) = x_0, \tag{5.11}$$

where $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector functions of dimensions $n \times 1$ and $n \times q_1$, respectively. Without loss of generality, we assume the initial time $t_0 = 0$ and that $f(0) = 0$; hence, the steady-state of the nonlinear system of Eq. (5.11) is the origin. Finally, we will assume that the fast subsystem is globally asymptotically stable which is needed in order to estabilish stability of the closed-loop system under model predictive control using machine learning models in section 4.

Assumption 5.4. *The origin of the closed-loop fast subsystem described by Eq.* (5.5) *exhibits global asymptotic stability, uniformly in* x*. This implies that there exists a function* $\beta_{\bar{\bar{z}}}$ *of class* $\mathcal{KL}$ *such that for any* $\bar{\bar{z}}(0) \in \mathbb{R}^p$*,*

$$|\bar{\bar{z}}(t)| \leq \beta_{\bar{\bar{z}}}\left(|\bar{\bar{z}}(0)|, \frac{t}{\epsilon}\right) \qquad \forall t \geq 0 \tag{5.12}$$

5.2.3 Stabilizability assumption via control Lyapunov function

Regarding the dynamics of the slow subsystem described in Eq. (5.11), we assume that there exists a locally Lipschitz feedback controller, $\Phi(x) \in U$, that renders the origin of the slow subsystem of Eq. (5.11) asymptotically stable, i.e., there exists a $\mathcal{C}^1$ Lyapunov function $V(x)$ that is continuously differentiable and meets the following set of inequalities:

$$a_1(|x|) \leq V(x) \leq a_2(|x|), \tag{5.13a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -a_3(|x|), \tag{5.13b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq a_4(|x|) \tag{5.13c}$$

where a_1, a_2, a_3 and a_4 are class $\mathcal{K}$ functions for all $x \in \mathbb{R}^n \subset D$. Additionally, given that the system $F(x, u)$ has a Lipschitz property and the input u is constrained by the set U, then there exists positive constants M, L_x, and $L_x^{'}$ such that the subsequent inequalities hold for

all $x, x' \in D$ and $u \in U$:

$$|F(x,u)| \leq M \tag{5.14a}$$

$$|F(x,u) - F(x',u)| \leq L_x|x - x'| \tag{5.14b}$$

$$\left|\frac{\partial V(x)}{\partial x}F(x,u) - \frac{\partial V(x')}{\partial x}F(x',u)\right| \leq L_x^{'}|x - x'| \tag{5.14c}$$

Moreover, the region $\Omega_\rho := \{x \in D | V(x) \leq \rho\}, \rho > 0$, is defined as the stability region for the slow subsystem of Eq. (5.11).

Now we will discuss the incorporation of machine learning models, particularly neural networks, in process systems engineering. The first step in developing any type of neural network model is generating a comprehensive data set that effectively captures the input-output relation. This data set is used to train the network and enables it to learn patterns present in the data and make accurate predictions. In order to generate a data set that captures the dynamics of the nonlinear system of Eq. (5.1) within the region Ω_ρ, we follow the data generation technique described in [49]. The first step is to carry out various open-loop simulations of the nonlinear system of Eq. (5.1), covering a wide-range of initial conditions (i.e., $x_0 \in \Omega_\rho$ and $z_0 \in \Omega_\rho$) and valid input signals under sample-and-hold implementation (i.e., the input is applied to the system as piecewise constant functions, $u = u(t_k) \in U, \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$, Δ is the sampling period). To integrate the nonlinear system of Eq. (5.1), the well-established forward Euler approach is used with a sufficiently small integration time step $h_c \ll \Delta$. As a result, an extensive data set containing the dynamics of the system is generated, which is then utilized to train a neural network employing the

Keras library.

5.2.4 Recurrent neural networks

Consider designing a recurrent neural network (RNN) model that predicts only the slow state vector x of Eq. (5.1), or in other words, to approximate the dynamics of the slow subsystem of Eq. (5.11). The network is trained using m data points $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$ of T-time-length, where $i = 1, ..., m$ and $t = 1, ..., T$. $\mathbf{x}_{i,t} \in \mathbb{R}^{d_x}$ and $\mathbf{y}_{i,t} \in \mathbb{R}^{d_y}$ are the RNN input and output, respectively. Additionally, d_x and d_y are the dimensions of the RNN input and output, respectively. It is worth mentioning that the notations used for the RNN inputs and outputs are represented in boldface in order to establish a clear distinction between the RNN notations and the notations used to describe the nonlinear two-time-scale system of Eq. (5.1). The RNN input $\mathbf{x}_{i,t}$ comprises the current slow state measurements along with the manipulated inputs applied through the time steps $t = 1, ..., T$, where the manipulated inputs are generated randomly within the set U. The RNN output $\mathbf{y}_{i,t}$ comprises the predicted future slow state through the time steps $t = 1, ..., T$. As a result, the RNN model is designed to make future predictions of the slow states over one sampling period with $T = \frac{\Delta}{h_c}$ time steps. The data set utilized to develop the RNN model is constructed by generating m independent data sequences obtained from an underlying distribution over $\mathbb{R}^{d_x \times T} \times \mathbb{R}^{d_y \times T}$.

With the aim of simplification, we introduce a single-hidden-layer RNN model, given that $\mathbf{h}_i \in \mathbb{R}^{d_h}$ are the hidden states and can be evaluated as follows:

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) \tag{5.15}$$

where the element-wise nonlinear activation function is denoted by σ_h. $U \in \mathbb{R}^{d_h \times d_h}$ is the weight matrix associated with the hidden states, while $W \in \mathbb{R}^{d_h \times d_x}$ is the weight matrix associated with the input vector. Furthermore, the following equation evaluates the output layer $y_{i,t}$ of the RNN model:

$$\mathbf{y}_{i,t} = \sigma_y(\mathcal{V}\mathbf{h}_{i,t}) \tag{5.16}$$

where element-wise activation function of the output layer is denoted by σ_y, and $\mathcal{V} \in \mathbb{R}^{d_y \times d_h}$ is the weight matrix associated with the output layer. In this particular application, the input of the RNN model will be the current slow states measurement, $x \in \mathbb{R}^n$, as well as the manipulated input $u \in \mathbb{R}^{q_1}$ for the next sampling period. The output of the RNN model will be the predicted future slow states x for at least one sampling period ahead. The basic RNN structure is represented in Fig. 5.1.

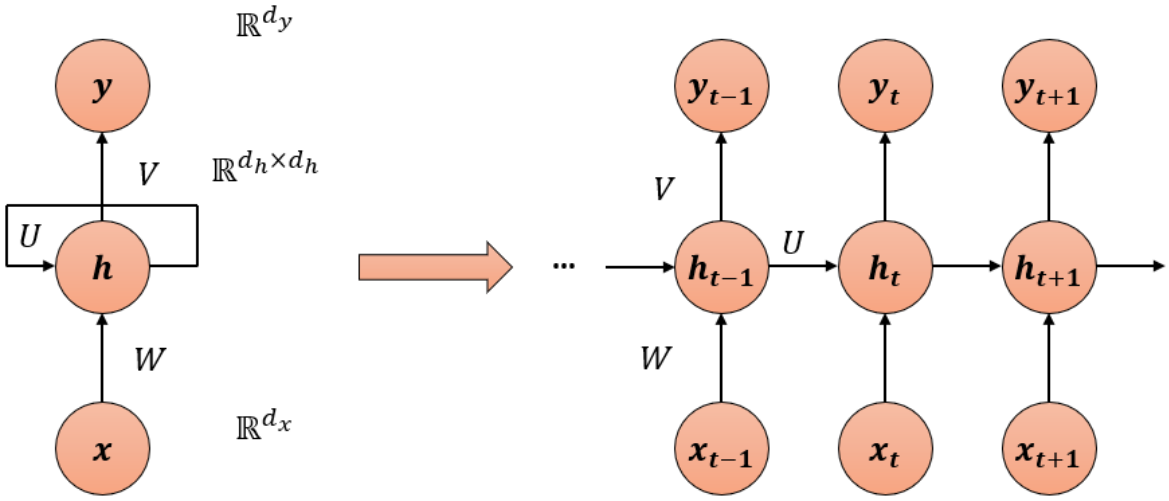

Figure 5.1: Recurrent nueral network structure.

Let $\check{\mathbf{y}}$ be the predicted value and $\mathbf{y}$ be the actual value. We can then define the loss

function $L(\mathbf{y}, \breve{\mathbf{y}})$ that computes the squared difference between the actual and the predicted value. More precisely, we will consider the L_2 (i.e., mean squared error) loss function. Without loss of generality, we establish the following standard assumptions that are required for the study of generalization error bounds and are reviewed for completeness:

Assumption 5.5. *The inputs of the RNN are bounded (i.e.,* $\|\mathbf{x}_{i,t}\| \leq B_X$ *for all* $i = 1, ..., m$ *and* $t = 1, ..., T$*).*

Assumption 5.6. *The Frobenius norms of the weight matrices are bounded (i.e.,* $\|\mathcal{V}\|_F \leq B_{\mathcal{V},F}, \|W\|_F \leq B_{W,F}, \|U\|_F \leq B_{U,F}$*).*

Assumption 5.7. σ_h *is a positive homogenous and 1-Lipschitz continuous nonlinear activation function (i.e.,* $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ *holds* $\forall \alpha \geq 0$ *and* $z \in \mathbb{R}$*).*

Assumption 5.8. *The datasets used for training, testing, and validation are drawn from the same distribution.*

Remark 5.1. *Although the RNN model is developed to approximate the dynamics of the slow subsystem described by Eq.* (5.11)*, we emphasize the fact that the data used to train this RNN model is constructed using the nonlinear two-time-scale system of Eq.* (5.1)*. The reason is that, in most practical scenarios, engineers are limited to working with data obtained from the original nonlinear two-time-scale system of Eq.* (5.1)*. Additionally, it is possible to design an RNN model that predicts the full two-time-scale dynamics of Eq.* (5.1) *using full state measurement.*

Remark 5.2. *Assumption 5.5 takes into account the bounded nature of RNN inputs. This aligns with the observation that the states* x *and the inputs* u *are restricted within certain*

limits, where $x \in \Omega_\rho$ and $u \in U$. Assumption 5.6 requires that the weight matrices of the RNN are bounded. This requirement can be met while training the RNN, since the search for the optimal RNN parameters is limited to a finite class of neural network hypotheses. Assumption 5.8 indicates that the RNN model constructed using data derived from industrial operations will be utilized on the exact same process, under the condition that the data distribution remains unchanged. It is important to point out that in the context of assessing the generalization performance for machine learning models, Assumption 5.8 is regarded as an essential one. Several well-known activation functions can be used in constructing the RNN model that satisfy Assumption 5.7; for instance, the rectified linear unit (ReLu) activation function is one such example.

5.2.5 Feedforward neural networks

Using the RNN model described in Section 5.2.4, we are able to predict the slow state vector x. However, it is important to devise a methodology to predict the values of the fast states. In line with the framework proposed in [143], we consider the design of a feedforward neural network (FNN) that predicts the fast states using the previously predicted slow states from the RNN model. More precisely, the FNN input will be the slow states x, and its output will be the fast states z. The FNN model is trained using m independent data points from an underlying distribution over $\mathbb{R}^{d_x^F} \times \mathbb{R}^{d_y^F}$, where d_x^F and d_y^F are the dimensions of the FNN input and output, respectively. In this particular application, given that the FNN input is the slow state, $x \in \mathbb{R}^n$, and its output is the fast state $z \in \mathbb{R}^p$, we have $d_x^F = n$ and $d_y^F = p$.

The general form of an FNN model can be formulated as follows:

$$\mathbf{y}^F = \sigma_d(Q_d\sigma_{d-1}(Q_{d-1}\sigma_{d-2}(...\sigma_1(Q_1\mathbf{x}^F)))) \tag{5.17}$$

where d is the total number of FNN layers, $\mathbf{y}^F \in \mathbb{R}^{d_y^F}$ is the predicted output of the FNN, and $\mathbf{x}^F \in \mathbb{R}^{d_x^F}$ is the FNN input. For each FNN layer l, where $l = 1, \ldots, d$, the weight parameter matrix is denoted as Q_l, and the activation function is represented as σ_l. The depth of the network is represented by the number of layers d. However, the width of the network is the maximal number of neurons in a hidden layer and is represented by $h_{\max}$. Given that h_l denotes the number of neurons in the l^{th} layer, the width of the network can be expressed as $h_{\max} = \max_{l=1,\ldots,d}\{h_l\}$. The general FNN structure is shown in Fig. 5.2.

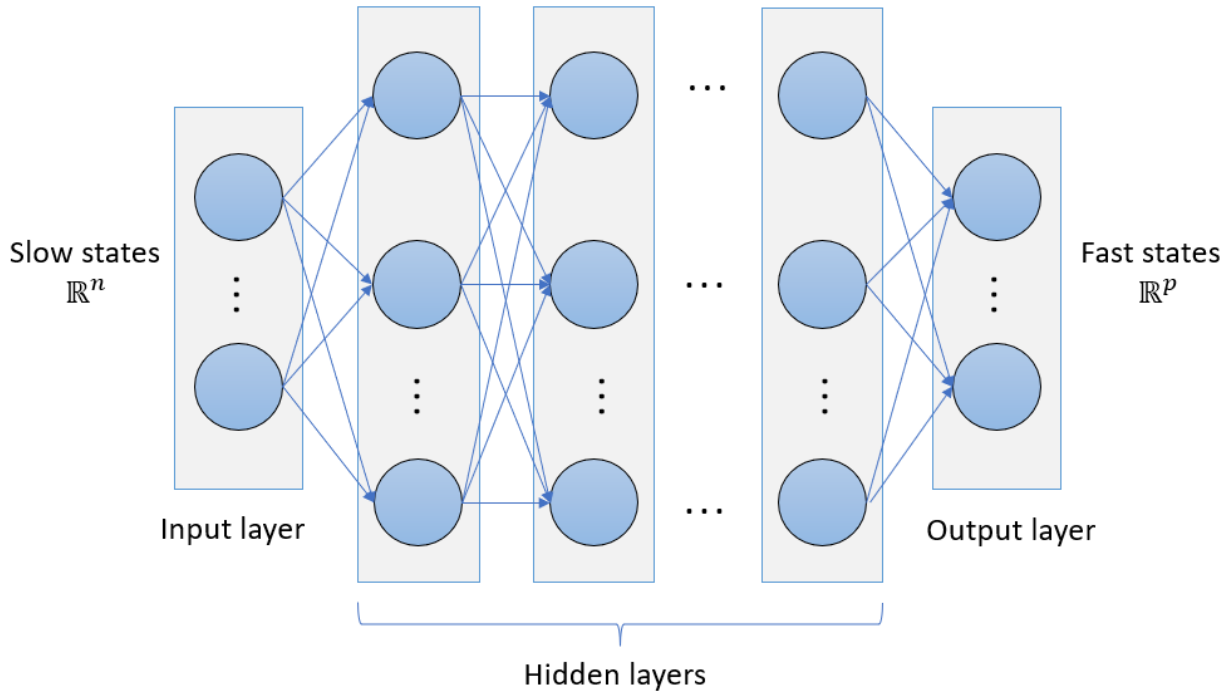

Figure 5.2: Feedforward neural network structure.

If $\breve{\mathbf{y}}^F$ is the predicted value and $\mathbf{y}^F$ is the actual value, we can define the loss function $L(\mathbf{y}^F, \breve{\mathbf{y}}^F)$ that computes the squared difference between the actual and the predicted value. More precisely, we will consider L_2 (i.e., mean squared error) loss function. In a similar manner to the discussion on RNNs, we establish the following assumptions for the FNN model developed:

Assumption 5.9. *The inputs of the FNN are bounded, (i.e.,* $\|\mathbf{x}_i^F\| \leq B_X$ *for all* $i = 1, ..., m$*).*

Assumption 5.10. *The norms of the weight matrices are bounded, in the sense that the maximal 1-norm of the rows of the weight matrices in the hidden layers and output are bounded, (i.e.,* $\|Q\|_{1,\infty} \leq B_Q$*).*

Assumption 5.11. *The activation function* σ_l *is 1-Lipschitz continuous and satisfies* $\sigma_l(0) = 0$*, where* $l = 1, ..., d$ *(e.g.,* $\tanh(\cdot)$*).*

Assumption 5.9 is an implication of the assumption that the RNN output (slow states) is bounded. Assumptions 5.9–5.11 follow the same reasoning as Assumptions 5.5–5.7 as explained in Remark 5.2. Building upon the previous assumptions, it should be noted that Assumption 5.8 remains applicable for the FNN model.

5.3 Generalization error bounds of neural networks modeling two-time-scale systems

The primary goal of constructing any neural network model is to make accurate predictions. Therefore, analyzing the generalization error bounds for various types of neural

networks modeling different types of nonlinear systems enables us to improve the construction and design of these network models. Typically, the assessment of neural networks is conducted using a finite set of training samples. Hence, it is essential to investigate the generalization error bound for the neural network model, which allows us to evaluate the performance of the model in accurately predicting outcomes for new, unseen data. In other words, considering new data from the same distribution, the generalization error bound quantifies the ability of a neural network model to make accurate predictions for unseen data, which the neural network has not encountered before.

In this work, we consider an RNN that predicts the dynamics of the slow states. Then, an FNN is utilized to predict the fast states using the predicted slow states as the input for the FNN. The generalization error bounds of the neural networks modeling two-time-scale systems are investigated. Particularly, in this section, we will discuss the generalization error bounds of both the RNN (which predicts the slow states dynamics) and the FNN (which predicts the fast states). The generalization error bound is derived utilizing basic concepts of statistical machine learning theory. Hence, in the upcoming subsection, we outline fundamental concepts and definitions used to derive generalization error bounds. Additionally, in the following subsection, it is worth mentioning that the same principles that apply to $\mathbf{x}$,$\mathbf{y}$ and $\breve{\mathbf{y}}$ also extend to $\mathbf{x}^F$,$\mathbf{y}^F$ and $\breve{\mathbf{y}}^F$, respectively. In the same vein, principles that apply to d_x and d_y, are also true for d_x^F and d_y^F, respectively. Let $\mathcal{H}$ be the hypothesis class of RNN functions $h(\cdot)$ that map the d_x-dimensional input $\mathbf{x} \in \mathbb{R}^{d_x}$ to the d_y-dimensional output $\breve{\mathbf{y}} \in \mathbb{R}^{d_y}$. Similarly, let $\mathcal{H}^F$ be the hypothesis class of FNN functions $h^F(\cdot)$ that map the d_x-dimensional input $\mathbf{x}^F \in \mathbb{R}^{d_x}$ to the d_y-dimensional output $\breve{\mathbf{y}}^F \in \mathbb{R}^{d_y}$.

For the following subsection, we also note that the principles that apply to $\mathcal{H}$ and $h(\cdot)$ are also valid for $\mathcal{H}^F$ and $h^F(\cdot)$, respectively. It should be noted that the dimensions d_x and d_y differ based on the network type, whether it is an RNN or an FNN.

5.3.1 Generalization error bound preliminaries

Definition 5.1. *Consider a function, h, which makes predictions of the output $\mathbf{y}$ corresponding to the input $\mathbf{x}$, along with an underlying distribution D. The generalization error or the expected loss is formulated as follows,*

$$\mathbb{E}[L(h(\mathbf{x}), \mathbf{y})] = \int_{X \times Y} L(h(\mathbf{x}), \mathbf{y})\, \rho(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{y}. \tag{5.18}$$

where $\rho(\mathbf{x}, \mathbf{y})$ represents the joint probability distribution for $\mathbf{x}$ and $\mathbf{y}$, while X and Y represent the vector space for all possible inputs and outputs, respectively, and L denotes the loss function.

We introduce the following definition of the empirical error as an approximation of the expected loss due to the fact that the joint probability distribution ρ is unknown in many scenarios.

Definition 5.2. *Considering a data set consisting of m data samples $S = (s_1, ..., s_m)$, with each $s_i = (\mathbf{x}_i, \mathbf{y}_i)$, the empirical error or risk can be expressed as,*

$$\hat{\mathbb{E}}_S[L(h(\mathbf{x}), \mathbf{y})] = \frac{1}{m} \sum_{i=1}^{m} L(h(\mathbf{x}_i), \mathbf{y}_i) \tag{5.19}$$

We note that the m data samples are gathered from the same data distribution. In this

work, the loss function $L(\mathbf{y}, \check{\mathbf{y}})$ is chosen as the mean squared error (i.e., L_2 loss function). Furthermore, the generalization error bounds are derived using "Rademacher complexity", a widely recognized concept in the field of machine learning theory that is used to determine the complexity and richness of a class of functions. The Rademacher complexity is defined as follows:

Definition 5.3. *Given a data set $S = \{s_1, ..., s_m\}$ with m samples, and a hypothesis class $\mathcal{F}$ of real-valued functions, then the empirical Rademacher complexity of $\mathcal{F}$ can be defined as follows,*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i f(s_i) \right] \tag{5.20}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_m)^T$ consists of the Rademacher random variables ϵ_i, these variables are independent and identically distributed (i.i.d.), satisfying the condition that $\mathbb{P}(\epsilon_i = -1) = \mathbb{P}(\epsilon_i = 1) = 0.5$.

Given that $\mathcal{G}_t$ is the class of loss functions associated with the function class $\mathcal{H}$ and is defined as follows:

$$\mathcal{G}_t = \{g_t : (\mathbf{x}, \mathbf{y}) \rightarrow L(h(\mathbf{x}), \mathbf{y}), h \in \mathcal{H}\}, \tag{5.21}$$

where $\mathbf{x}$ and $h(\mathbf{x})$ are the model's input and output, respectively, while $\mathbf{y}$ denotes the actual value of the output, the upper bound of the generalization error can be computed using the Rademacher complexity $\mathcal{R}_S(\mathcal{G}_t)$ as in the following Lemma.

Lemma 5.1. *(c.f. Theorem 3.3 in [82]) Given a data set $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, that consist of m i.i.d. data samples, then with probability $1 - \delta$ the following inequality holds for*

all $g_t \in \mathcal{G}_t$ over the data samples S:

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m} \sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{5.22}$$

For a comprehensive proof of Lemma 5.1, interested readers can refer to [82] and [21]. By observing Eq. (5.22), clearly, the upper bound of the generalization error relies on three terms. The first term is the empirical loss ($\frac{1}{m} \sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$). The second term is the Rademacher complexity ($2\mathcal{R}_S(\mathcal{G}_t)$), while the third term depends on the data set size m along with the confidence level δ. At this stage, the goal is to be able to effectively quantify the generalization error bound using predefined and measurable values. This can be done through imposing a further upper bound on the Rademacher complexity term. Having established the background and basic concepts of the generalization error bound, our next discussion will focus on examining the generalization error bounds for neural networks modeling two-time scale systems.

5.3.2 RNN generalization error bound

In this subsection, we will be investigating the generalization error bound for the RNN model that predicts the slow dynamics of the two-time-scale system defined in Eq. (5.1). We start by introducing the following Lemma, which upper bounds the Rademacher complexity in terms of the RNN parameters.

Lemma 5.2. *Let $\mathcal{H}_{k,t}$, $k = 1, ..., d_y$ represent the class of real-valued functions that associated to the k^{th} component of the RNN output at t^{th} time step, with activation functions and*

weight matrices satisfying Assumptions 5.5—5.8. Given a data set $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, ..., m$, *consisting of* m *i.i.d. data samples, the following inequality holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \leq \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}} \tag{5.23}$$

where $M = B_{\mathcal{V},F} B_{W,F} \frac{B_{U,F}^{t}-1}{B_{U,F}^{-1}}$, *and* B_X *is the upper bound for RNN inputs.*

For a comprehensive and detailed proof of Lemma 5.2, interested readers may refer to [21]. Applying the results derived and proven in [21], the following Lemma specifically addresses the generalization error bound for the RNN model that predicts the slow dynamics of the two-time-scale system defined in Eq. (5.1).

Lemma 5.3. *Consider the general class of two-time-scale continuous-time nonlinear systems described by Eq.* (5.1) *under the assumption that the slow and fast states of Eq.* (5.1) *are stable, and the perturbation parameter* ϵ *is sufficiently small. An RNN satisfying Assumptions 5.5—5.8 is constructed to predict the slow states of the system at the* t^{th} *time step. Given the* L_r*–Lipschitz loss function that belongs to the family of loss functions* $\mathcal{G}_t$ *associated with the RNN function of class* $\mathcal{H}_t$ *and a data set* $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, *with* m *i.i.d. data samples, the following inequality holds true with probability at least* $1-\delta$ *over* S, *for all* $t \in [t_0, T]$*:*

$$\mathbb{E}[g_t^{RNN}(\breve{x}, x)] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\breve{x}_i, x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{M B_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{5.24}$$

where $M = B_{\mathcal{V},F} B_{W,F} \frac{B_{U,F}^{t}-1}{B_{U,F}^{-1}}$, *and* B_X *is the upper bound for RNN inputs.*

Eq. (5.24) indicates that the generalization error bound of the RNN that predicts the slow state vector x depends on a number of factors, such as the number of training samples m, the time length t of the RNN inputs, the upper bound on the input vector B_X, the complexity hypothesis class in terms of weight matrices M, as well as the empirical loss $(\frac{1}{m}\sum_{i=1}^{m} g_t(\breve{x}_i, x_i))$. We emphasize that the actual slow state is denoted as x and the predicted slow state by the RNN model is $\breve{x}$. Additionally, based on Eq. (5.8), the generalization error bound of Eq. (5.24) holds for all $t \in [t_0, T]$, under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small.

Now we will discuss the implementation of the RNN generalization error bound for a specific loss function. A locally Lipschitz continuous loss function is used to optimize the RNN weights and biases. To be more precise, we employ the MSE loss function of the form,

$$L = \frac{1}{m}\sum_{i=1}^{m}(\breve{\mathbf{y}}_i, \mathbf{y}_i)^2, \tag{5.25}$$

where $\breve{\mathbf{y}}$ and $\mathbf{y}$ are the predicted and actual values, respectively. Since the loss function L is locally Lipschitz continuous, it satisfies the following inequality,

$$|L(\mathbf{y}, \breve{\mathbf{y}}_\mathbf{2}) - L(\mathbf{y}, \breve{\mathbf{y}}_\mathbf{1})| \leq L_r|\breve{\mathbf{y}}_\mathbf{2} - \breve{\mathbf{y}}_\mathbf{1}| \tag{5.26}$$

where L_r is the local Lipschitz constant for the loss function L. Since the RNN model predicts the slow states of Eq. (5.1), the loss function is computed over the actual and predicted slow states, x and $\breve{x}$, respectively. Moreover, the expected loss of L is upper bounded by the

following inequality with probability at least $1-\delta$:

$$\mathbb{E}[L(\breve{x},x)] \leq \frac{1}{m}\sum_{i=1}^{m} L(\breve{x}_i,x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{5.27}$$

Given that the MSE loss function L computes the error between the RNN output x and the predicted RNN output $\breve{x}$, then the upper bound on $|\breve{x}-x|$ can be written as follows:

$$|\breve{x}-x| \leq \sqrt{\frac{1}{m}\sum_{i=1}^{m} L(\breve{x}_i,x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right)} \tag{5.28}$$

5.3.3 FNN generalization error bound

In this subsection, we will be investigating the generalization error bound for the FNN model that predicts the fast states of the two-time-scale system defined in Eq. (5.1). We start by introducing the following Lemma, which upper bounds the Rademacher complexity in terms of the FNN parameters.

Lemma 5.4. *(c.f. Theorem 2 in [46]) Given a class of scalar-valued functions $\mathcal{H}_k^F$, and a neural network with depth d where, $\|Q_1\|_{1,\infty} \leq B_Q$ for all $l = 1,...,d$ and Assumptions 5.8–5.11 are satisfied, the following inequality holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_k^F) \leq \frac{2B_X(B_Q)^d\sqrt{d+1+\log(d_x)}}{\sqrt{m}} \tag{5.29}$$

For a comprehensive and detailed proof of Lemma 5.4, interested readers may refer to [46]. Applying the results derived and proven in [149], the following Lemma specifically addresses the generalization error bound for the FNN model that predicts the fast states

using the slow states of the two-time-scale system defined in Eq. (5.1).

Lemma 5.5. *Consider the general class of two-time-scale continuous-time nonlinear systems described in Eq.* (5.1) *under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small. An FNN satisfying Assumptions 5.8–5.11 is constructed to predict the fast states using the slow states. Given L_r–Lipschitz loss functions associated with the vector-valued FNN hypothesis class $\mathcal{H}^F$ and a data set S consisting of m i.i.d. data samples, the following inequality holds true with probability at least $1-\delta$ over S, for all $t \in [t_p, T]$, where t_p is as defined in Theorem 5.1:*

$$\mathbb{E}[g_t^{FNN}(\breve{z}, z)] \leq \frac{1}{m}\sum_{i=1}^{m} g_L(\breve{z}_i, z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X (B_Q)^d \sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right) \quad (5.30)$$

Eq. (5.30) indicates that the generalization error bound of the FNN modeling the fast subsystem depends on several factors, such as the number of training samples m, the upper bound on the input vector B_X, the complexity hypothesis class in terms of upper bound of the weight matrices B_Q, number of layers d, as well as the empirical loss $(\frac{1}{m}\sum_{i=1}^{m} L(\breve{z}_i, z_i))$. We emphasize that the actual fast state is denoted as z and the predicted fast state by the FNN model is $\breve{z}$. Therefore, the generalization error bound of Eq. (5.30) holds for all $t \in [t_p, T]$, under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small.

Now we will discuss the implementation of the FNN generalization error bound specifically on the MSE loss function as a locally Lipschitz continuous loss function such as the MSE is used to optimize the FNN weights and biases. The MSE loss function satisfies the

inequality of Eq. (5.26). Since the FNN model predicts the fast states of Eq. (5.1), the loss function is computed over the actual and predicted fast states, z and $\breve{z}$, respectively. Moreover, the expected loss of L is upper bounded by the following inequality with probability at least $1-\delta$:

$$\mathbb{E}[L(\breve{z},z)] \leq \frac{1}{m}\sum_{i=1}^{m} L(\breve{z}_i,z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X (B_Q)^d \sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right) \tag{5.31}$$

Given that the MSE loss function L computes the error between the RNN output z and the predicted RNN output $\breve{z}$, then the upper bound on $|\breve{z}-z|$ can be written as follows:

$$|\breve{z}-z| \leq \sqrt{\frac{1}{m}\sum_{i=1}^{m} L(\breve{z}_i,z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X (B_Q)^d \sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right)} \tag{5.32}$$

5.4 Machine learning-based LMPC using an RNN that approximates the slow subsystem

Having investigated generalization error bounds for neural networks modeling two-time-scale systems, the rest of the manuscript focuses on the design and performance of LMPC using RNN models with and without timescale decomposition. Specifically, the objective is to design an RNN to predict only the slow state vector x, following the approach outlined in Section 5.2.4, and to then incorporate the designed RNN model within an LMPC scheme to show that it is sufficient to stabilize the full two-time-scale system of Eq. (5.1). In this section, we will present a stability analysis for the proposed LMPC framework. For simplicity, in this section, we will present the RNN model that predicts the slow state vector x, as a

continuous-time nonlinear system of the following form:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T \hat{z} \tag{5.33}$$

where $\hat{x} \in \mathbb{R}^n$ is the RNN state vector and $u \in \mathbb{R}^{q_1}$ is the manipulated input vector. The weight matrices are denoted as A and Θ. The diagonal coefficient matrix A has negative diagonal entries. Θ is defined as $\Theta = [\theta_1, ..., \theta_n] \in \mathbb{R}^{(q_1+n)\times n}$ with entries $\theta_i = b_i[w_{i1}, ..., w_{i(q_1+n)}], i = 1, ..., n$, given that w_{ij} represents the weight associated with the connection between the i^{th} neuron and the j^{th} output, with i ranging from 1 to n and j ranging from 1 to $(q_1 + n)$. The vector $\hat{z} = [\hat{z}_1, ..., \hat{z}_n, \hat{z}_{n+1}, ..., \hat{z}_{n+q_1}] = [\sigma(\hat{x}_1)...\sigma(\hat{x}_n), u...u_{q_1}] \in \mathbb{R}^{n+q_1}$ consists of both network states $\hat{x}$ and inputs u. Additionally, a nonlinear activation function $\sigma(\cdot)$ is applied to the network states $\hat{x}$ when constructing the vector $\hat{z}$. It is worth mentioning that the weight matrices and activation functions meet the requirements of Assumptions 5.5–5.8. For simplicity, we assume a single-hidden-layer RNN model represented by Eq. (5.33), which does not explicitly include bias terms. Nevertheless, it should be emphasized that the results obtained in this section are not limited to single-hidden-layer RNN models, but can be generalized to include deep RNN models that consist of multiple hidden layers.

5.4.1 Lyapunov-based control using an RNN model

Taking into account the RNN model designed to predict the values of the slow states, we assume that there exists a stabilizing control law $\Phi_{nn}(x) \in U$ (e.g., a P-controller, a Lyapunov-based controller) that can render the origin of the RNN model of Eq. (5.33)

asymptotically stable in an open neighborhood around the origin denoted as $\hat{D}$. Hence, there exists a continuously differentiable Lyapunov function $\hat{V} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ that satisfies the following inequalities:

$$\hat{a}_1(|x|) \leq \hat{V}(x) \leq \hat{a}_2(|x|) \tag{5.34a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{a}_3(|x|) \tag{5.34b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{a}_4(|x|) \tag{5.34c}$$

where $\hat{a}_i$ are class $\mathcal{K}$ functions for all $x \in \mathbb{R}^n \subset \hat{D}$. The Lyapunov function $\hat{V}$ can be designed using a quadratic formulation, which satisfies the required criteria, or, alternately, using learning-based approaches as conducted using a linear parameter-varying (LPV) framework in [152]. In the simulation example presented in our work below, a quadratic Lyapunov function is used, and extensive open-loop and closed-loop simulations were conducted to verify the asymptotic stability assumption is met. The Lyapunov level set that ensures stability of the RNN model of is defined as $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. This implies that the closed-loop stability region of the RNN model of Eq. (5.33) is denoted as $\Omega_{\hat{\rho}}$. Moreover, there exist positive constants M_{nn} and L_{nn} such that the following inequalities are satisfied for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|F_{nn}(x, u)| \leq M_{nn} \tag{5.35a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, u) - \frac{\partial \hat{V}(x')}{\partial x} F_{nn}(x', u) \right| \leq L_{nn}|x - x'| \tag{5.35b}$$

The feedback controller $u = \Phi_{nn}(x) \in U$ can stabilize the dynamics of the slow subsystem of Eq. (5.11) under a sufficiently small modeling error between the nominal slow subsystem and the RNN model. This result is shown in the following proposition.

Proposition 5.1. *Consider the RNN model of Eq.* (5.33) *that satisfies the stabilizability conditions of Eq.* (5.34) *and is rendered asymptotically stable around the origin under the control law* $u = \Phi_{nn}(x) \in U$ *for all* $x \in \Omega_{\hat{\rho}}$. *Then, the origin of the slow subsystem of Eq.* (5.11) *is asymptotically stable if there exists a positive real number* ν_m, *where* $\nu_m < \hat{a}_3(|x|)/\hat{a}_4(|x|)$, *such that the modeling error between the slow subsystem of Eq.* (5.11) *and the RNN model of Eq.* (5.33) *(i.e.,* $\nu = |F(x,u) - F_{nn}(x,u)|$*) is upper bounded by* ν_m *for all* $x \in \Omega_{\hat{\rho}}$.

Proof. We follow the strategy outlined in [143] to prove Proposition 5.1. The main objective is to show that, under the control law $u = \Phi_{nn}(x) \in U$, the slow subsystem of Eq. (5.11) is asymptotically stable around the origin for all $x \in \Omega_{\hat{\rho}}$ if $\Phi_{nn}(x)$ renders the RNN designed to approximate the slow subsystem also asymptotically stable with a bounded modeling error between the nominal slow subsystem and the RNN model. In other words, we show that $\dot{\hat{V}}(x) \leq 0$ for the slow subsystem of Eq. (5.11) under the controller $u = \Phi_{nn}(x) \in U$ based on the RNN model. We utilize the inequalities in Eqs. (5.34b) and (5.34c) and compute $\dot{\hat{V}}(x)$

as follows:

$$\begin{aligned}\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{nn}(x)) \\ &= \frac{\partial \hat{V}}{\partial x}(F_{nn}(x, \Phi_{nn}(x)) + F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x)) \\ &\leq -\hat{a}_3(|x|) + \hat{a}_4(|x|)(F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))) \\ &\leq -\hat{a}_3(|x|) + \nu_m \hat{a}_4(|x|)\end{aligned} \tag{5.36}$$

By assigning $\nu_m < \frac{\hat{a}_3(|x|)}{\hat{a}_4(|x|)}$, we get $\dot{\hat{V}}(x) \leq -\tilde{a}_3(|x|) \leq 0$ where $\tilde{a}_3(|x|) = -\hat{a}_3(|x|) + \nu_m \hat{a}_4(|x|) > 0$. Since $\hat{a}_3$ and $\hat{a}_4$ are known functions chosen in such a way as to ensure the above result holds. To show that, if the general class $\mathcal{K}$ functions are chosen as $\hat{a}_3 = a_3|x|$ and $\hat{a}_4 = a_4|x|$, where a_3 and a_4 are constants, then $\nu_m = \frac{a_3}{a_4}$. Hence, we guarantee closed-loop stability of the slow subsystem of Eq. (5.11) around the origin under the control law $\Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$.

□

The designed RNN model of Eq. (5.33) is incorporated into an LMPC scheme, where the control actions computed by the LMPC are implemented in a sample-and-hold fashion. Moreover, this sample-and-hold implementation of the LMPC control law $u = \Phi_{nn}(x) \in U$ establishes certain properties. These properties will be studied in the following two propositions. Specifically, the following proposition shows that the error between the state of the slow subsystem of Eq. (5.11) and the predicted state by the RNN model Eq. (5.33) is bounded.

Proposition 5.2. *Consider the RNN model of Eq.* (5.33) *and the slow subsystem of Eq.* (5.11)*, starting with the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$. There exists a class $\mathcal{K}$ function $f_w(\cdot)$*

and a positive constant κ such that the following inequalities are satisfied for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{\nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{5.37a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{5.37b}$$

Proof. Taking into consideration that $e(t) = x(t) - \hat{x}(t)$ represent the error vector between the state of the slow subsystem of Eq. (5.11) and the state of the RNN model of Eq. (5.33), hence, the bound for the time-derivative of $e(t)$ is given as follows:

$$\begin{aligned} |\dot{e}(t)| &= |F(x,u) - F_{nn}(\hat{x},u)| \\ &\leq |F(x,u) - F(\hat{x},u)| + |F(\hat{x},u) - F_{nn}(\hat{x},u)| \end{aligned} \tag{5.38}$$

Using Eq. (5.14b), for all $x, \hat{x} \in \Omega_{\hat{\rho}}$, the first term in Eq. (5.38) can be bounded as follows:

$$\begin{aligned} |F(x,u) - F(\hat{x},u)| &\leq L_x|x(t) - \hat{x}(t)| \\ &\leq L_x|x(t) - \hat{x}(t)| \end{aligned} \tag{5.39}$$

The term $|F(\hat{x},u) - F_{nn}(\hat{x},u)|$ in Eq. (5.39) denotes the modeling error, and it is upper bounded by ν_m for all $\hat{x} \in \Omega_{\hat{\rho}}$. Hence, the term $\dot{e}(t)$ can be further bounded utilizing the bound of the modeling error and the bound of Eq. (5.39), as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L_x|x(t) - \hat{x}(t)| + \nu_m \\ &\leq L_x|e(t)| + \nu_m \end{aligned} \tag{5.40}$$

Taking into account the zero initial condition (i.e., $e(0) = 0$), we integrate the inequality of

Eq. (5.40) and obtain the following upper bound for the error vector for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{\nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{5.41}$$

Eq. (5.37b) can be derived using the Taylor series expansion of $\hat{V}(x)$ around $\hat{x}$ as follows, for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{5.42}$$

where κ is a positive real number. Furthermore, Eqs. (5.34a) and (5.34b) are used to further upper bound $\hat{V}(x)$ as follows:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{5.43}$$

□

The following proposition demonstrates that the closed-loop state of the slow subsystem described in Eq. (5.11) is maintained within the stability region $\Omega_{\hat{\rho}}$ at all times. Additionally, utilizing the Lyapunov-based controller $u = \Phi_{nn}(x) \in U$ through sample-and-hold implementation, ensures that closed-loop state of the slow subsystem can be ultimately bounded within a small region around the origin $\Omega_{\rho_{\min}}$.

Proposition 5.3. *(c.f Proposition 3 in [143]) Consider the slow subsystem of Eq.* (5.11) *under the controller $\Phi_{nn}(\hat{x}) \in U$ that satisfies the conditions of Eq.* (5.34) *and is implemented in sample-and-hold fashion (i.e., $\Phi_{nn}(\hat{x}(t_k)), \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$) to stabilize the RNN model of Eq.* (5.33). *Then, there exist $\epsilon_w > 0$, $\Delta > 0$ and $\hat{\rho} > \rho_{\min} > \rho_{nn} > \rho_s$ that*

satisfy

$$-\hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn}M_{nn}\Delta \leq -\epsilon_s \tag{5.44a}$$

$$-\tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'}M_F\Delta \leq -\epsilon_w \tag{5.44b}$$

and

$$\rho_{nn} := \max\{\hat{V}\Big(\hat{x}(t+\Delta)\Big) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{5.45a}$$

$$\rho_{\min} \geq \rho_{nn} + -\hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))f_w(\Delta) + \kappa\Big(f_w(\Delta)\Big)^2 \tag{5.45b}$$

such that, for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$*, there exists a class* $\mathcal{KL}$ *function* β_x *and a class* $\mathcal{K}$ *function* $\bar{\gamma}$ *such that the following inequality is satisfied:*

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{5.46}$$

and the state $x(t)$ *of the nominal slow subsystem of Eq.* (5.11) *is bounded in* $\Omega_{\hat{\rho}}$ *for all times and ultimately bounded in* $\Omega_{\rho_{\min}}$*.*

Proof. Assuming $x(t_k) = \hat{x}(t_k)$, the first part of establishing this proof involves showing that $\hat{V}(\hat{x})$ is decreasing under the control law $u(t) = \Phi_{nn}(x(t_k)) \in U$ for $t \in [t_k, t_{k+1})$, where $x(t_k)$ and $\hat{x}(t_k)$ are the state of the slow subsystem of Eq. (5.11) and the state of the RNN model of Eq. (5.33), respectively. The time-derivative of $\hat{V}(\hat{x})$ for all $t \in [t_k, t_{k+1})$ is calculated as follows:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
&= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\Big) \\
&\quad + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))\Big) \\
&\quad - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\Big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\Big)
\end{aligned}
\tag{5.47}
$$

Utilizing the inequalities in Eqs. (5.34a) and (5.34b), we further bound $\dot{\hat{V}}(\hat{x}(t))$ as follows:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) \leq & - \hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
& - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k)))
\end{aligned}
\tag{5.48}
$$

Applying the Lipschitz inequalities in Eq. (5.35), we proceed with bounding $\dot{\hat{V}}(\hat{x}(t))$ as follows:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &\leq - \hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn}|\hat{x}(t) - \hat{x}(t_k)| \\
&\leq - \hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn} M_{nn} \Delta
\end{aligned}
\tag{5.49}
$$

Therefore, if Eq. (5.44a) is satisfied, the following inequality holds for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$
\dot{\hat{V}}(x(t)) \leq -\epsilon_s \tag{5.50}
$$

Upon integrating the aforementioned differential equation over the time interval $t \in [t_k, t_{k+1})$, it is derived that $\hat{V}(\hat{x}(t_{k+1})) \leq \hat{V}(\hat{x}(t_k)) - \epsilon_s \Delta$. Therefore, when Eq. (5.44a) is satisfied, this leads to the fact that $\dot{\hat{V}}(x(t))$ is negative for any $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. As a result, utilizing the control law $u = \Phi_{nn}(\hat{x})$, under sample-and-hold fashion, ensures that the closed-loop state

of the RNN model of Eq. (5.33) is bounded within the region $\Omega_{\hat{\rho}}$ and moves toward the origin. Nevertheless, it should be observed that Eq. (5.50) may not hold true in cases where $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$, which indicated that the state $\hat{x}(t_k)$ may leave the region Ω_{ρ_s} within one sampling period. Hence, we introduce the region $\Omega_{\rho_{nn}}$ in Eq. (5.45a) to guarantee that the closed-loop state $\hat{x}(t_k)$ of the RNN model will be bounded in the region $\Omega_{\rho_{nn}}$ within one sampling period, for all $t \in [t_k, t_{k+1})$, $u \in U$ and $\hat{x}(t_k) \in \Omega_{\rho_s}$. Consider the case where $\hat{x}(t_{k+1})$ leaves the region Ω_{ρ_s}, then the controller $u = \Phi_{nn}\left(x(t_{k+1})\right)$ reactivates to derive the states into the region Ω_{ρ_s} and Eq. (5.50) will be satisfied again at $t = t_{k+1}$. Up to this point, it can be concluded that the state of the RNN system of Eq. (5.33) is ultimately bounded in $\Omega_{\rho_{nn}}$ for all $x_0 \in \Omega_{\hat{\rho}}$.

The second part of this proof is to demonstrate that the controller $u = \Phi_{nn}(x) \in U$, implemented in sample-and-hold fashion, effectively bounds the states of the slow subsystem of Eq. (5.11) in $\Omega_{\hat{\rho}}$ and can ultimately derive the states to a small region around the origin. This requires showing that, $\hat{V}(x)$ for the slow subsystem of Eq. (5.11), is decreasing under the control law $u(t) = \Phi_{nn}(x(t_k))$ for $t \in [t_k, t_{k+1})$ and $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. The derivative of $\hat{V}(x(t))$ with respect to time is computed as follows:

$$\begin{aligned}
\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
&= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) \\
&\quad + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
&\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)))
\end{aligned} \tag{5.51}$$

Using the inequality in Eq. (5.36), which holds for all $x \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the first term in Eq. (5.51) can be further bounded as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) \leq & -\tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), \xi) \\ & - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0)\end{aligned} \tag{5.52}$$

where $\tilde{a}_3(\cdot)$ was previously defined in the proof of Proposition 5.1. By using the Lipschitz condition stated in Eq. (5.14), we derive the following upper bound for $\dot{\hat{V}}(x(t))$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) \leq & \tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'} |x(t) - x(t_k)| \\ \leq & \tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'} M_F \Delta\end{aligned} \tag{5.53}$$

Therefore, if Eq. (5.44b) holds true, the following inequality is satisfied for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and for all $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{5.54}$$

By integrating the above differential equation over the time interval $t \in [t_k, t_{k+1})$ between any two arbitrary points within the previous time interval, it can be shown that for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_S}$, the following results are obtained:

$$\hat{V}(x(t_{k+1})) \leq V(x(t_k)) - \epsilon_w \Delta \tag{5.55}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)), \forall t \in [t_k, t_{k+1}) \tag{5.56}$$

Based on Eq. (5.54), it can be observed that $\dot{\hat{V}}(x(t))$ is negative for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_S}$. As a result, the state of the slow subsystem of Eq. (5.11) remains bounded within the region

$\Omega_{\hat{\rho}}$ continuously and can be ultimately driven towards the origin in each sampling period through implementing the control law $u = \Phi_{nn}(x)$. In addition, if $x(t_k) \in \Omega_{\rho_s}$, the state of the RNN model of Eq. (5.33) is bounded within the region $\Omega_{\rho_{nn}}$ for one sampling period, this outcome has been demonstrated in the first part of the proof. Taking onto account the bounded modeling error between the state of the RNN, as described in Eq. (5.33), and the state of the slow subsystem of Eq. (5.11), a compact set $\Omega_{\rho_{\min}} \supset \Omega_{\rho_{nn}}$ exists and satisfies Eq. (5.45b). The set $\Omega_{\rho_{\min}}$ is introduced to ensure that the state of the slow subsystem of Eq. (5.11) is bounded within the region $\Omega_{\rho_{\min}}$ during one sampling period, provided that the state of the RNN represented by Eq. (5.33) is bounded within the region $\Omega_{\rho_{nn}}$. In the case where the state $x(t)$ enters the set $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_S}$, it has been shown that Eq. (5.56) is satisfied. Therefore, under the control law $u = \Phi_{nn}(x)$, the state $x(t)$ will be driven towards the origin in the next sampling period, and ultimately bounding the state of the slow subsystem within a small region around the origin $\Omega_{\rho_{\min}}$. Hence, considering the continuity property of the Lyapunov function $\hat{V}$, it follows that there exists a class $\mathcal{KL}$ function β_x and a class $\mathcal{K}$ function $\bar{\gamma}$ such that if $x_0 \in \Omega_{\hat{\rho}}$, then $x(t) \in \Omega_{\hat{\rho}}$, for all $t \leq t_0$:

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{5.57}$$

□

5.4.2 Machine learning-based LMPC formulation

In this subsection, we introduce the machine-learning based LMPC formulation that utilizes the designed RNN model of Eq. (5.33) to predict future states over a certain future horizon. Furthermore, the LMPC is essentially an optimization problem consisting of an objective function and a number of constraints. This optimization problem is solved repeatedly to compute the optimal input trajectory. The following shows the formulation of the Machine learning-based LMPC optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t), u(t)) dt \tag{5.58a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{5.58b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{5.58c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.58d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k))), \text{ if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.58e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.58f}$$

where the predicted slow state trajectory is denoted as $\tilde{x}$. The set of piece-wise constant functions with period Δ is denoted as $S(\Delta)$, and the number of sampling periods in the prediction horizon is denoted by N. The optimal control action $u^*(t)$ is calculated by repeatedly solving the machine learning-based LMPC optimization problem over the entire prediction horizon $t \in [t_k, t_{k+N})$. Then, the optimal control action $u^*(t_k)$ computed at the first sampling period is transmitted by the controller to be applied to the process. Sub-

sequently, the resulting real-time state, $x(t_k)$, is fed back to the machine learning-based LMPC optimization problem to compute the optimal input trajectory for the next sampling time. The optimization problem aims to minimize the time-integral of $L_{MPC}(\tilde{x}(t), u(t))$ over the prediction horizon, this is represented in the cost function of Eq. (5.58a). As for the constraints of the optimization problem, the first constraint demonstrated in Eq. (5.58b) is essentially the RNN model utilized to approximate the states of the slow subsystem dynamics. Eq. (5.58c) represents the input constraints that may be applied to the process over the entire prediction horizon. The initial condition needed to solve Eq. (5.58b), is basically the slow state measurement at $t = t_k$, this is specified in the constraint of Eq. (5.58d). In the case where, $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, the constraint Eq. (5.58e) ensures that the closed-loop state converges towards the origin. Furthermore, if the state $x(t_k)$ enters the region $\Omega_{\rho_{nn}}$, then it is necessary to guarantee that the predicted states by the RNN model remain bounded within the region $\Omega_{\rho_{nn}}$ throughout the entire prediction horizon. This is accomplished by employing Eq. (5.58f).

5.4.3 Closed-loop stability

Assuming certain conditions are met, the following theorem establishes the closed-loop stability of a singularly perturbed system described in Eq. (5.1) when the machine learning-based LMPC of Eq. (5.58) is implemented.

Theorem 5.2. *Consider the singularly perturbed system of Eq.* (5.1) *in closed loop with the optimal control action* u^* *calculated by the machine learning-based LMPC of Eq.* (5.58)

based on the controller $\Phi_{nn}(x)$ *that meets the conditions of Eq.* (5.34)*. Additionally, assuming that Propositions 5.1–5.3 and Assumptions 5.1 and 5.4 hold true, then there exist class* $\mathcal{KL}$ *functions* β_x,β_z*, a pair of positive real numbers* (δ, d) *and* $\epsilon^* > 0$ *such that, if* $\max\{|x(0), |z(0)|\} \leq \delta$ *and* $\epsilon \in (0, \epsilon^*]$*, then, for all* $t \geq 0$*,*

$$|x(t)| \leq \beta_x \left(|x(0)|, t\right) + \bar{\gamma}(\rho_{\min}) + d \tag{5.59}$$

$$|z(t)| \leq \beta_z \left(|z(0)|, \frac{t}{\epsilon}\right) + d \tag{5.60}$$

Proof. We follow the proof analysis as in [143]. Moreover, we consider that the By substituting the optimal control action u^* into Eq. (5.1), the closed-loop system can be represented as follows:

$$\dot{x} = f_1(x, z, u^*, \epsilon) \tag{5.61a}$$

$$\epsilon \dot{z} = f_2(x, z, \epsilon) \tag{5.61b}$$

We set $\epsilon = 0$ and obtain the following,

$$\dot{x} = f_1(x, z, u^*, 0) \tag{5.62a}$$

$$0 = f_2(x, z, 0) \tag{5.62b}$$

where the assumption that the error between z and $\bar{z}$ is not greater than $\mathcal{O}(\epsilon)$ enabled us to simplify the analysis by approximating $\bar{z}$ as z. By solving Eq. (5.62b) for z, we get a unique, isolated root $z = \bar{f}(x)$. We substitute the root into Eq. (5.62a) and we get the following for

the reduced order slow subsystem,

$$\dot{x} = f_1(x, \bar{f}_2(x), u^*, 0) \tag{5.63}$$

By observing the LMPC equation of Eq. (5.58), in the case where $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, Eq. (5.58e) is activated such that the Lyapunov function $\hat{V}$ under the calculated control law u is decreased. According to the finding in Proposition 5.3 and considering a sufficiently small modeling error, the state of the slow subsystem of Eq. (5.61a) will gradually converge to the origin and enter the region $\Omega_{\rho_{nn}}$ within a finite number of sampling time steps. Eq. (5.58f) is activated once the state $x(t_k)$ enters the region $\Omega_{\rho_{nn}}$, where it ensures that the predicted state is bounded within $\Omega_{\rho_{nn}}$ through the whole prediction horizon. Additionally, by ensuring that the modeling error and the sampling time are sufficiently small, Proposition 5.3 establishes that the actual state of Eq. (5.61a) can be maintained within a slightly expanded set $\Omega_{\rho_{min}}$, encompassing $\Omega_{\rho_{nn}}$. Hence, LMPC guarantees that, for any initial state $x_0 \in \Omega_{\hat{\rho}}$, the state $x(t)$ of the closed-loop slow subsystem described by Eq. (5.61a) is maintained bounded within the region $\Omega_{\hat{\rho}}$ at all times, and fulfills the bound of Eq. (5.57) that has been established in Proposition 5.3.

Additionally, by introducing a fast timescale $\tau = t/\epsilon$, a new coordinate $\bar{\bar{z}} = z - \bar{f}_2(x)$ and considering $\epsilon = 0$, the closed-loop fast subsystem is given by:

$$\frac{d\bar{\bar{z}}}{d\tau} = f_2(x, \bar{\bar{z}} + \bar{f}_2(x), 0) \tag{5.64}$$

According to Assumption 5.4, global asymptotic stability is achieved for the origin of the

closed-loop fast subsystem of Eq. (5.64), satisfying Eq. (5.12) for any $\bar{\bar{z}}(0) \in \mathbb{R}^p$. As a result, the closed-loop system of Eq. (5.61) meets all the assumptions required for Theorem 1 in [153] to hold true. Hence, there exist class $\mathcal{KL}$ functions β_x,β_z, a pair of positive real numbers (δ, d) and there exists $\epsilon^* > 0$ such that if $\max\{|x(0), |z(0)|\} \leq \delta$ and $\epsilon \in (0, \epsilon^*]$, such that the closed-loop states of the slow and fast systems are bounded by Eqs. (5.59) and (5.60). □

Remark 5.3. *It is possible to define the general class of two-time-scale systems with an additional input associated to the fast subsystem as follows:*

$$\dot{x} = f_1(x, z, u, \epsilon) \tag{5.65a}$$

$$\epsilon\dot{z} = f_2(x, z, u_2, \epsilon) \tag{5.65b}$$

where $u_2 \in \mathbb{R}^{q_2}$ is the bounded manipulated input vector associated with the fast subsystem. The input vector u_2 is constrained by $u_2 \in U_2 := \{|u_i| \leq u_{2_{\max,i}}, i = 1, ..., q_2\}$. However, if there exists a stabilizing control law $u_2 = h_{stable}(x, z)$ that stabilizes the dynamics of the fast subsystem, then the general class of two-time-scale systems defined by Eq. (5.65) *can be reduced to the class of systems presented in Eq.* (5.1)*, which is the primary focus of this work. Furthermore, there are relevant works that have addressed the stability analysis of classes of systems defined as Eq.* (5.65)*, they share similar concepts but incorporate certain modifications, interested readers my refer to [141] to further explore this subject.*

5.5 Application to a chemical process example

In this section, we will apply the proposed reduced-order machine learning strategy to a chemical process example. Specifically, we will build two ML models—a reduced-order ML model to approximate the slow subsystem, and another ML model to capture the dynamics of the full two-time-scale system. Subsequently, each ML model will be incorporated into an LMPC scheme, and the performance of the controllers will be compared in terms of closed-loop properties and computational efficiencies.

We consider a perfectly-mixed, non-isothermal CSTR in which an irreversible and endothermic reaction that transforms chemical A to product B ($A \rightarrow B$) occurs. Fig. 5.3 depicts the CSTR, where C_A represents the concentration of chemical A, and T_r represents the temperature of the reactor contents. C_{A0} denotes the inlet molar concentration of chemical A, which is fed into the reactor at flow rate F and temperature T_{A0}. Assuming that the vessel maintains a constant holdup, V_r denotes the volume of liquid present in the reactor. Due to the occurrence of an endothermic reaction within the reactor, a heating jacket with volume V_j is used to provide the energy as required. T_{j0} is the inlet temperature of the heat transfer fluid provided to the jacket with a flow rate F_j. The reactor contents and the heat transfer fluid maintain constant densities, denoted as ρ_m and ρ_j, respectively. In addition, they both have constant heat capacities, denoted as $c_{p,m}$ and $c_{p,j}$, respectively. The enthalpy of the reaction is ΔH_r, the heat transfer coefficient is U, and the heat transfer contact area between the reactor and the jacket is A_r. Given that k_0 is the pre-exponential constant, R is the ideal gas constant, and E is the activation energy of the reaction, the time-varying rate

constant of the reaction is denoted by k and is given by the following equation:

$$k = k_0 \exp\left(\frac{-E}{RT_r}\right) \tag{5.66}$$

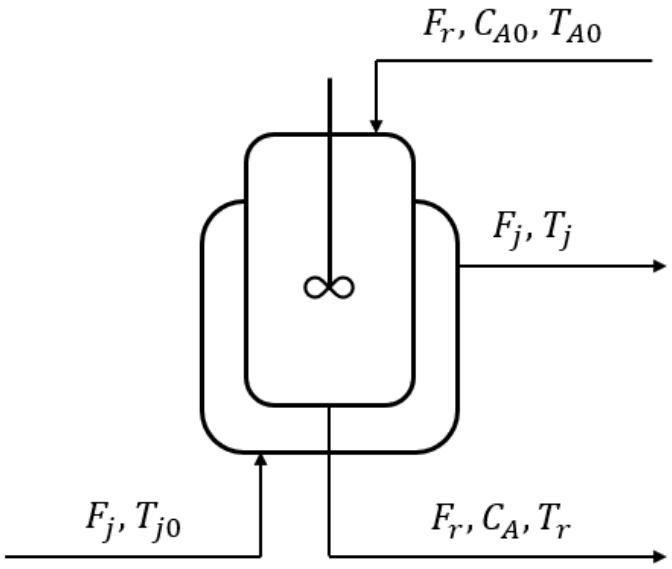

Figure 5.3: The continuous-stirred tank reactor with jacket.

The first-principles equations of the CSTR are described by the following dynamic material and energy balance equations:

$$V_r \frac{\mathrm{d}C_A}{\mathrm{d}t} = F_r(C_{A0} - C_A) - k_0 \exp\left(\frac{-E}{RT_r}\right) C_A V_r \tag{5.67a}$$

$$V_r \frac{\mathrm{d}T_r}{\mathrm{d}t} = F_r(T_{A0} - T_r) + \frac{-\Delta H_r}{\rho_m C_{p,m}} k_0 \exp\left(\frac{-E}{RT_r}\right) C_A V_r + \frac{UA_r}{\rho_m C_{p,m}}(T_j - T_r) \tag{5.67b}$$

$$V_j \frac{\mathrm{d}T_j}{\mathrm{d}t} = F_j(T_{j0} - T_j) - \frac{UA_r}{\rho_j C_{p,j}}(T_j - T_r) \tag{5.67c}$$

where the values of the process parameters are enlisted in Table 5.1. Additionally, we consider the constant $\epsilon = \frac{V_j}{V_r}$ to be the singular perturbation parameter. Hence, the sys-

tem of Eq. (5.67) can be written in the standard singularly perturbed form of Eq. (5.1), which implies that the concentration of chemical A (C_A) and the temperature of the reactor (T_r) are the slow states, whereas the temperature of the heating jacket (T_j) can be considered as the fast state. The input of the system of Eq. (5.67) is the feed concentration of chemical A (C_{A0}). The control objective is to drive the system to its stable steady state $(C_{As}, T_{rs}, T_{js}) = (2.54\ kmol\ m^{-3},\ 274.4\ K,\ 303.3\ K)$, corresponding to the steady-state input value of $C_{A0s} = 3.75\ kmol\ m^{-3}$, while the input is permitted to vary within the range $\Delta C_{A0} = [-3.5, 3.5]\ kmol\ m^{-3}$. In order to shift the steady state of the system of Eq. (5.67) to the origin, we express the states and the input in terms of deviation variables and rewrite the system of Eq. (5.67). Specifically, the deviation variables are defined as follows:

$$\Delta C_A = C_A - C_{As} \tag{5.68a}$$

$$\Delta T_r = T_r - T_{rs} \tag{5.68b}$$

$$\Delta T_j = T_j - T_{js} \tag{5.68c}$$

$$\Delta C_{A0} = C_{A0} - C_{A0s} \tag{5.68d}$$

Therefore, to write the CSTR system of Eq. (5.67) in the standard form of Eq. (5.1), we define $x^T = [\Delta C_A \quad \Delta T_r]$, $z = \Delta T_j$ and the input $u = \Delta C_{A0}$. The explicit Euler method is utilized to simulate the CSTR system, i.e., Eq. (5.67) is numerically integrated with a sufficiently small time step $h_c = 10^{-4}$ h. The open-source software package IPOPT [86] is employed to solve the ML-based LMPC optimization problem of Eq. (5.58), with a sampling time $\Delta = 0.03$ h.

Table 5.1: Notation and parameter values of the CSTR.

$V_r = 1.0\ m^3$	$E = 8.0 \times 10^3\ kcal\ kg^{-1}$
$V_j = 0.08\ m^3$	$U = 1000.0\ kcalh^{-1}m^{-2}K^{-1}$
$A_r = 6.0\ m^3$	$k_0 = 3.36 \times 10^6 h^{-1}$
$C_{A0s} = 3.75\ kmol\ m^{-3}$	$\rho_m = 900.0\ kg\ m^{-3}$
$C_{As} = 2.54\ kmol\ m^{-3}$	$\rho_j = 800.0\ kg\ m^{-3}$
$R = 1.987\ kcal\ kmol^{-1}K^{-1}$	$c_{p,m} = 0.231\ kcal\ kg^{-1}K^{-1}$
$F_r = 3.0\ m^3h^{-1}$	$c_{p,j} = 0.200\ kcal\ kg^{-1}K^{-1}$
$F_j = 20.0\ m^3h^{-1}$	$\Delta H_r = 5.4 \times 10^4\ kcal\ mol^{-1}$
$T_{rs} = 274.4\ K$	$T_{j0} = 357.5\ K$
$T_{js} = 303.3\ K$	$T_{A0} = 310.0\ K$

5.5.1 Data generation and RNN models development

The first step in building the RNN models is to generate a comprehensive data set that captures the dynamics of the system of Eq. (5.67). In order to implement the explicit Euler algorithm, we first need to initialize the process model using a set of initial conditions $(C_A(0), T_r(0), T_j(0))$, where $C_A(0) \in [0, 9]\ mol/m^3$, $T_r(0) \in [280, 370]$ K and $T_j(0) \in [300, 390]$ K. We conduct open-loop simulations by integrating the CSTR system defined by Eq. (5.67) using a sufficiently small time step h_c form 10^6 random initial conditions within the specified ranges and applying random input signals $C_{A0} \in [0.5, 7.5]\ mol/m^3$ over a period of one sampling period Δ, which yields a data set of 10^6 trajectories of length equal to Δ. The data set is then divided into three sets—training, validation and testing. The training of the models is performed using the open-source library Keras [5]. The first RNN model is designed such that it predicts the dynamics of the full two-time-scale system of Eq. (5.67) (i.e, it predicts the dynamics of the slow and fast states). In other words, it utilizes the current state measurements $x(t_k)$, $z(t_k)$, and the manipulated input for the subsequent sampling period $u(t) \in [t_k, t_{k+1})$ to predict the future slow and fast state mea-

surements of $x(t)$ and $z(t)$ $\forall t \in [t_k, t_{k+1}]$. For simplicity, we will denote the model that predicts the dynamics of the full two-time-scale system of Eq. (5.67) as RNN_F, where the subscript "F" denotes "full" and not "fast". This model is designed using a single layer of 20 long short-term memory recurrent neural network (LSTM-RNN) units, and the validation and testing errors achieved were 1.434×10^{-6} and 1.432×10^{-6}, respectively. In addition, the total number of learning parameters for training this model is 2063. We note that the network was designed in a step-wise manner. Initially, we started with a simple structure, consisting of a single layer network with one LSTM-RNN unit, but the validation error was high. Gradually, we increased the network's complexity by adding more LSTM-RNN units. This progression was continued until we achieved a satisfactory level of the validation loss. In this particular case, the optimal outcome was achieved using a single layer network with 20 LSTM-RNN units. Further increasing the number of LSTM-RNN units beyond 20 resulted in an increase in the validation loss. Therefore, RNN_F is constructed using 20 LSTM-RNN units. On the other hand, the second RNN, denoted as RNN_S, is designed such that it predicts the dynamics of only the slow states of Eq. (5.67). Specifically, this RNN utilizes the current slow state measurement $x(t_k)$ and the manipulated input for the subsequent sampling period $u(t) \in [t_k, t_{k+1})$ to predict the future slow state measurements of $x(t)$ $\forall t \in [t_k, t_{k+1}]$. This model is designed using a single layer of 5 LSTM-RNN units corresponding to a total of 192 learning parameters following the same step-wise manner strategy used to design the RNN_F, with a validation error of 2.67×10^{-4} and a testing error of 2.68×10^{-4}. The results of both models are summarized in Table 5.2. The low testing error for both models indicates that the modeling error, as defined in Proposition 5.1, is sufficiently small over a wide range

of open-loop trajectories. Due to having fewer learning parameters, the RNN_S model is significantly less complex compared to the RNN_F model, possibly leading to a lower computational cost and making it a more practical choice for control applications. Therefore, our aim is to investigate whether the LMPC scheme utilizing RNN_S can meet the desired control objective sufficiently more efficiently compared to an LMPC scheme using RNN_F, which will be demonstrated via closed-loop simulations of the CSTR of Eq. (5.67) under both LMPC designs. However, before proceeding, it is essential to define the Lyapunov and objective functions used in both LMPC schemes. For the RNN_F-based LMPC, since the controller design is based on the dynamics of the full two-time-scale system of Eq. (5.67), the Lyapunov function is defined as:

$$V_F(x,z) = \begin{bmatrix} x - x_s \\ z - z_s \end{bmatrix}^T P_F \begin{bmatrix} x - x_s \\ z - z_s \end{bmatrix} \tag{5.69}$$

where the matrix P_F is given by

$$P_F = \begin{bmatrix} 91.6 & 2.9 & 4.3 \\ 2.9 & 0.8 & 0.2 \\ 4.3 & 0.2 & 0.7 \end{bmatrix}$$

The objective function for the RNN_F-based LMPC is given by $L_F(x,z,u) = \left|x\right|^2_{Q_{1F}} + \left|z\right|^2_{Q_{2F}} + \left|u\right|^2_{Q_{3F}}$, where $Q_{1F} = \begin{bmatrix} 20 & 0 \\ 0 & 0.05 \end{bmatrix}$, $Q_{2F} = 6$ and $Q_{3F} = 0.1$. On the other hand, the RNN_S-based LMPC is designed based on the dynamics of the slow subsystem of the CSTR, specifically Eqs. (5.67a) and (5.67b) expressed in the standard form. Therefore, the Lyapunov

function of the slow-subsystem used in the RNN_S-based LMPC is given by:

$$V(x) = (x - x_s)^T P (x - x_s) \tag{5.70}$$

where the matrix P is defined as

$$P = \begin{bmatrix} 84.9 & 1.2 \\ 1.2 & 0.4 \end{bmatrix}$$

The objective function is given by $L(x,u) = \left|x\right|^2_{Q_1} + \left|u\right|^2_{Q_2}$ where, $Q_1 = \begin{bmatrix} 20 & 0 \\ 0 & 0.05 \end{bmatrix}$ and $Q_2 = 0.05$.

Remark 5.4. *In this particular application, we used long short-term memory recurrent neural network (LSTM-RNN) units to construct the models. LSTM-RNNs are a special type of RNNs known for their ability to overcome the common problem of vanishing/exploding gradients in RNNs, resulting in generally better performance. For more information about LSTM-RNNs, interested readers may refer to [58]. However, we can always utilize classical RNNs that are well-suited for this particular application by choosing appropriate structures and carefully tuning hyperparameters to obtain the desired objective as well as an acceptable level of performance.*

5.5.2 Simulation results

In this subsection, we will carry out closed-loop simulations of the CSTR system described by Eq. (5.67) under the two LMPC schemes, one with each RNN model. We first establish the ability of both LMPCs to achieve the control objective adequately compared

Table 5.2: Specifications of the constructed recurrent neural network models.

RNN model	RNN_F	RNN_S
Number of units used	20	5
Testing error	1.432×10^{-6}	2.68×10^{-4}
Validation error	1.434×10^{-6}	2.67×10^{-4}
Number of learning parameters	2063	192

to LMPCs that utilize the respective first-principles systems as their process models. Subsequently, we compare the RNN-based LMPCs to each other in terms of closed-loop performance and also the computational time required for the simulations.

To establish the satisfactory performance of both RNN-based LMPCs, we compare their closed-loop performance to first-principles-based LMPCs, with the latter being the baseline for the best possible performance that can be achieved under LMPC. Specifically, the LMPC of Eq. (5.58) is considered in two scenarios, with the process model of Eq. (5.58e) being different between each scenario:

- Scenario 1: The LMPC utilizing RNN_F as the process model is compared to an LMPC employing the first-principles model of Eq. (5.67), denoted as FP_F, as its process model.

- Scenario 2: The LMPC utilizing RNN_S as the process model is compared to an LMPC employing the first-principles slow-subsystem of Eqs. (5.67a) and (5.67b), denoted as FP_S, as its process model. In this case, for the first-principles-based LMPC, we note that the full CSTR system of Eq. (5.67) is integrated but only the slow states C_A and T_r from Eqs. (5.67a) and (5.67b) are used in calculating the LMPC cost function of

Eq. (5.58a) and the Lyapunov function V.

For both scenarios, we start the simulations from the initial condition $IC_{\text{main}} = (\Delta C_A(0), \Delta T_r(0), \Delta T_j(0)) = (1, 30, 40)$. The LMPC prediction horizon is set to $N = 3$, while the remaining controller parameters were described in Section 5.5.1. The state and input trajectories under the LMPCs of scenario 1 and 2 are shown in Fig. 5.4 and Fig. 5.5, respectively. To compare the performance of the different LMPCs quantitatively, the time-integral of the cost function of the LMPC, $\int_{t=0}^{t_f} L(x(\tau), u(\tau))\ \mathrm{d}\tau$, is calculated over the entire simulation duration, $t_f = 3$ h. For scenario 1, the cost function values are 3458 and 3485 for the FP_F-based LMPC and the RNN_F-based LMPC, respectively, while, for scenario 2, the values of the cost function are 176 and 179 for the FP_S-based LMPC and the RNN_S-based LMPC, respectively. For both scenarios, we notice that the value of the cost function when utilizing the corresponding RNN model closely aligns with that achieved when employing the respective first-principles model, demonstrating the reliable predictive performance of the designed RNN models and their ability to drive the system to the steady-state when incorporated into an LMPC.

Next, we demonstrate the computational efficiency of the RNN_S-based LMPC due to its lower complexity while still achieving the desired controller performance. For IC_{main}, the computational time for the RNN_F-based LMPC shown in Fig. 5.4 is 5578 *sec*, whereas, for the RNN_S-based LMPC shown in Fig. 5.5, it is significantly lower at 2170 *sec*, which is 61% lower. This substantial difference in computational time can make the RNN_S a more practical choice for real-time application of MPC, where computations must be completed

within a sampling period to be sent to the actuator. Hence, to rigorously investigate the impact of integrating each of RNN_F and RNN_S with an LMPC framework in terms of stability and computational demand, we conduct closed-loop simulations from several different initial conditions in addition to IC_{main}.

The initial conditions chosen for further investigation are outlined in Table 5.3. Ten different initial conditions (IC_i where $i = 1, ..., 10$) within the operating region are selected, along with IC_{main}. For each initial condition, we simulate the CSTR system described in Eq. (5.67) under the RNN_F-based LMPC as well as the RNN_S-based LMPC. The state and input profiles under each LMPC for 4 representative initial conditions are depicted in Figs. 5.6 to 5.9, which indicate that both LMPC schemes successfully drive the process of Eq. (5.67) to its steady state. The closed-loop behavior were similar for the remaining 6 initial conditions in terms of convergence to the steady state. Additionally, Table 5.3 provides the computational times required, in terms of CPU time, for the entire simulation duration of $t_f = 3$ h when applying either LMPC scheme for the 11 different initial conditions. For all the tested initial conditions, the computational time required to execute the RNN_S-based LMPC is less than that required for the RNN_F-based LMPC. Among all the initial conditions in Table 5.3, the largest relative difference in computational time is observed for IC_3, with a percentage difference of 92% between the two LMPC schemes. Hence, the incorporation of the RNN_S in the LMPC framework has been demonstrated to be more practical and computationally efficient compared to employing the RNN_F, without compromising on stability and closed-loop performance.

The computational time of an MPC optimization problem can be influenced by several

factors. As mentioned in [154], MPC computational times are generally affected by the horizon length N, the number of the constraints in the MPC optimization problem, as well as the number of states and control actions of the system. The computational time is also directly related to the complexity of the process model. Hence, given the lower computational times across all 11 initial conditions tested, incorporating RNN_S into an LMPC framework is a more practical approach compared to the use of RNN_F. In the study of [155], it is observed that the CPU time of the IPOPT optimization problem increases as the complexity of the problem increases. Additionally, several works have discussed the computational complexity evaluation of neural networks. For instance, the work of [156] offers a comprehensive and systematic analysis for quantifying and comparing the computational complexity of neural network layers. The work proposes and computes three computational metrics per layer for different types of neural networks and presents the exact mathematical expressions and derivations, providing a deep understanding of various networks' complexities. Upon analyzing the results derived by [156], it becomes apparent that a neural network's complexity is proportional to its hyperparameters, for example, factors like the number of neurons, the number of hidden units, the input time sequence size, the number of output neurons and other parameters associated to the structure, size and design of the network, all of which affect the complexity and performance of a network, as well as the time required to solve the ML-based MPC optimization problem. As RNN_S consisted of only 5 LSTM-RNN units, while RNN_F consisted of 20 LSTM-RNN units, the complexity of RNN_F is much higher than that of RNN_S, leading to a correspondingly complex MPC optimization problem. Hence, over the entire simulation duration of $t_f = 3$ h, the total CPU time required to solve all

100 MPC optimization problems over the 100 sampling periods is significantly less for the RNN_S-based LMPC than the RNN_F-based LMPC.

Remark 5.5. *Although the state and input trajectories of Fig. 5.6 have not completely settled, the process is stabilized under both the RNN_F-based LMPC and the RNN_S-based LMPC. However, the process requires to be simulated for a longer duration, exceeding* 3 *h, for the states and input to ultimately converge to their steady state values and remain close to the origin. The reason for limiting the simulation time is to ensure fair and clear comparisons between the computational times for both controllers from various initial conditions as listed in Table 5.3.*

Remark 5.6. *In this particular application, RNN_S is designed such that it utilizes the current slow state measurement and the manipulated input for the next sampling period. As a result, the output of RNN_S will be the predicted future slow states x for only one sampling period ahead (1Δ). Consequently, in this application, since the prediction horizon is chosen to be an integer multiple of the sampling time Δ (i.e., $N = 3$ implies that $t_{k+N} = t_k + 3\Delta$ in Eq.* (5.58a)*), based on the design of RNN_S, RNN_S is required to be invoked at least three times to be able to make predictions for the entire prediction horizon. Additionally, an alternative method involves designing an RNN model, denoted as RNN_{new}, with the same goal of predicting the slow states. However, RNN_{new} utilizes not only the current slow state measurement and the manipulated input for the next sampling period, but also the current fast state measurement to predict the future slow states one sampling period Δ ahead, i.e., $x(t_{k+1})$. To obtain predictions for the entire prediction horizon ($N = 3$), an FNN takes as*

its input the output of RNN_{new} and predicts the fast state z at the next sampling period, i.e., the FNN predicts $z(t_{k+1})$ from $x(t_{k+1})$, following the design described in Section 5.2.5. Subsequently, for the next sampling period, the outputs of both RNN_{new} and the FNN are given as inputs to RNN_{new}, enabling it to predict the slow states $x(t_{k+2})$ for the next sampling period in the prediction horizon.

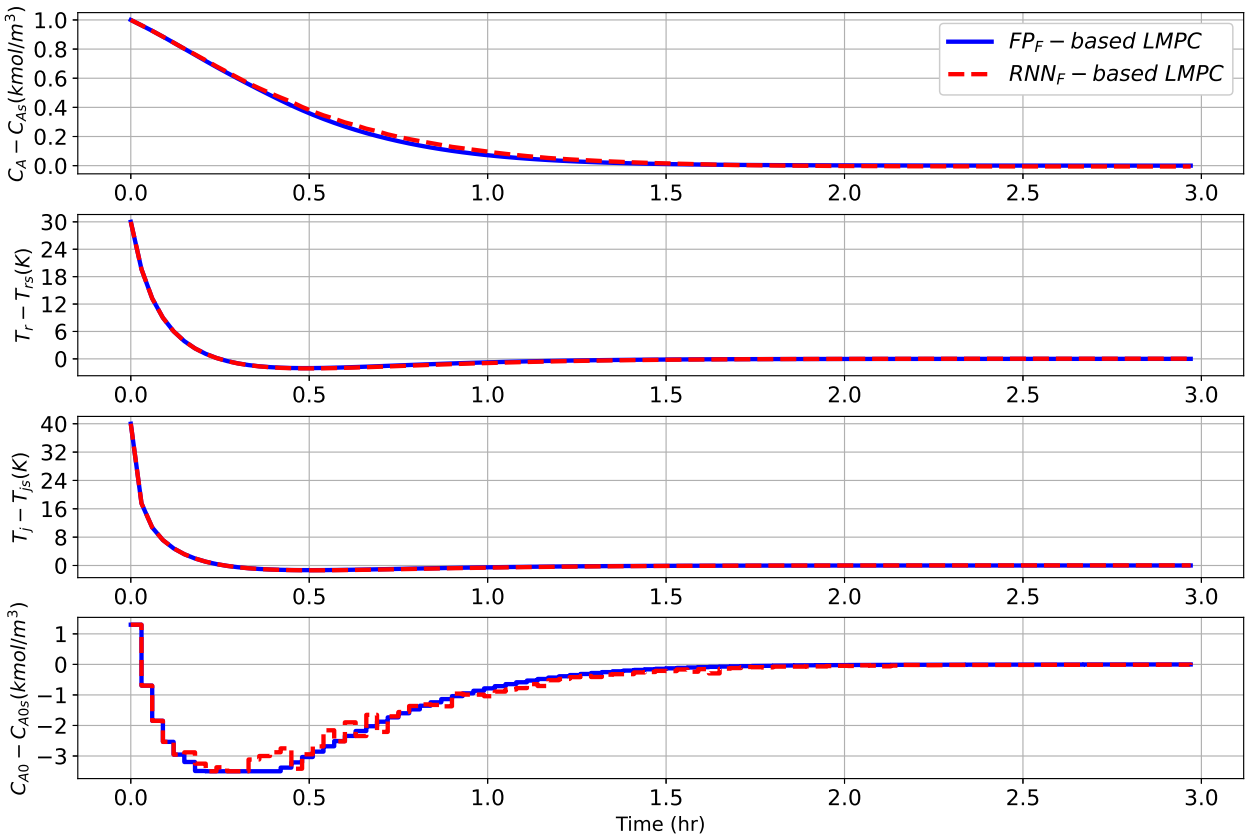

Figure 5.4: States and input trajectories of the CSTR under the Lyapunov based MPC using the first-principles model of the full process (FP_F-based LMPC, blue line), and the RNN_F (RNN_F-based LMPC, red dashed line).

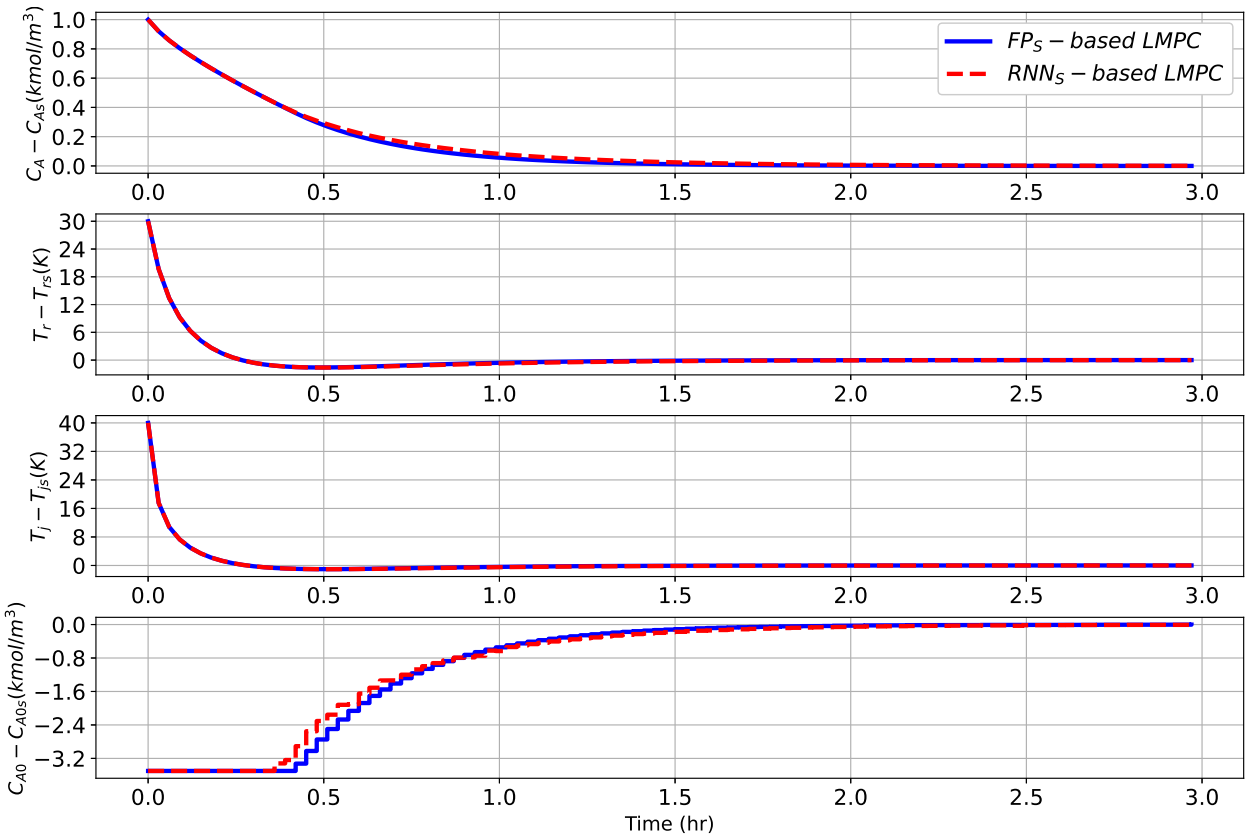

Figure 5.5: States and input trajectories of the CSTR under the Lyapunov based MPC using the first-principles model of the slow-subsystem (FP_S-based LMPC, blue line), and the RNN_S (RNN_S-based LMPC, red dashed line).

Table 5.3: Computational times for the RNN_F-based LMPC and the RNN_S-based LMPC over the simulation duration of $t_f = 3$ h starting from various initial conditions within the operating region.

Index	Initial condition	Computational time (sec)	
	$(\Delta C_A(0), \Delta T_r(0), \Delta T_j(0))$	RNN_F-based LMPC	RNN_S-based LMPC
IC_{main}	(1, 30, 40)	5578	2170
IC_1	(–1, 50, 40)	16,059	1807
IC_2	(–1, -10, –3)	4801	2667
IC_3	(–3, 30, 5)	31,884	2417
IC_4	(–2, –10, 100)	17,896	1921
IC_5	(1, 90, 10)	6078	1990
IC_6	(1, 20, 60)	5795	2404
IC_7	(3, –6, 20)	21,231	2411
IC_8	(1, 50, 50)	5161	2225
IC_9	(–2, 30, 60)	10,275	2194
IC_{10}	(–1, 10, 80)	18,146	2106

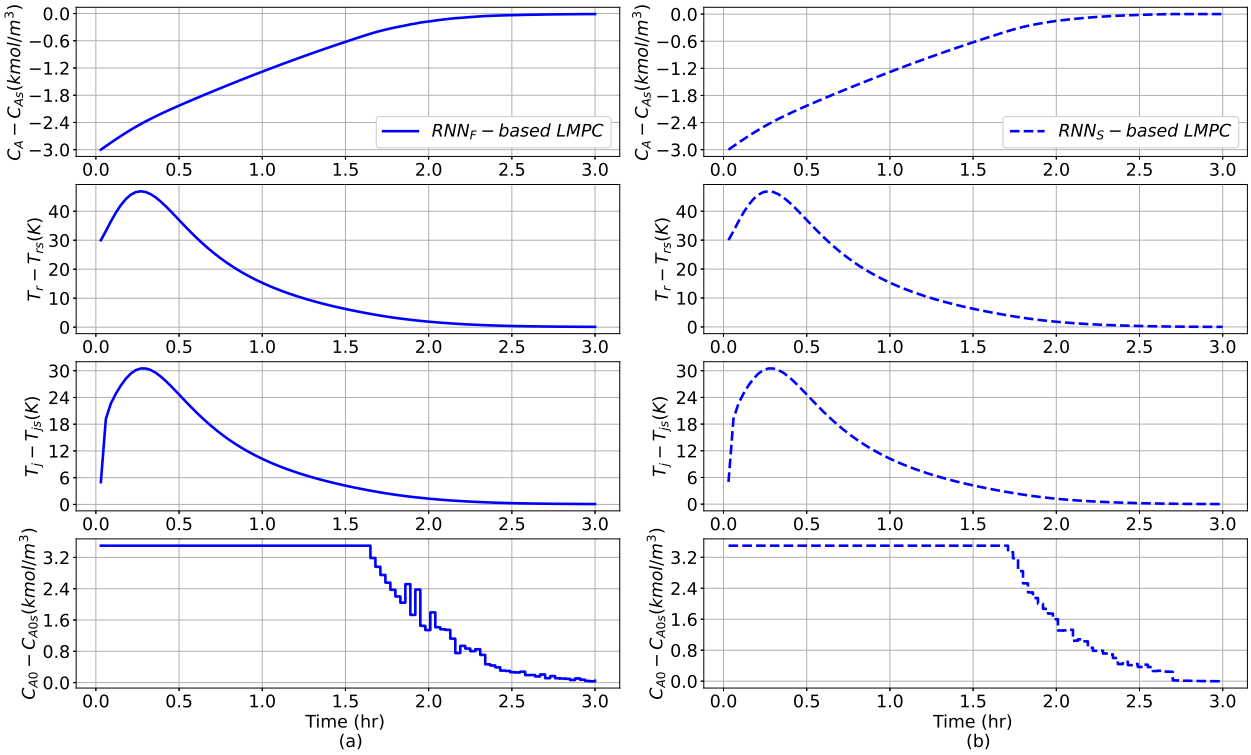

Figure 5.6: Considering the initial condition $IC_3 = (-3, 30, 5)$ (a) illustrates the time-varying profiles of the states and the input under RNN_F-based LMPC (solid line), while (b) shows the time-varying profiles of the states and the input under RNN_S-based LMPC (dashed line).

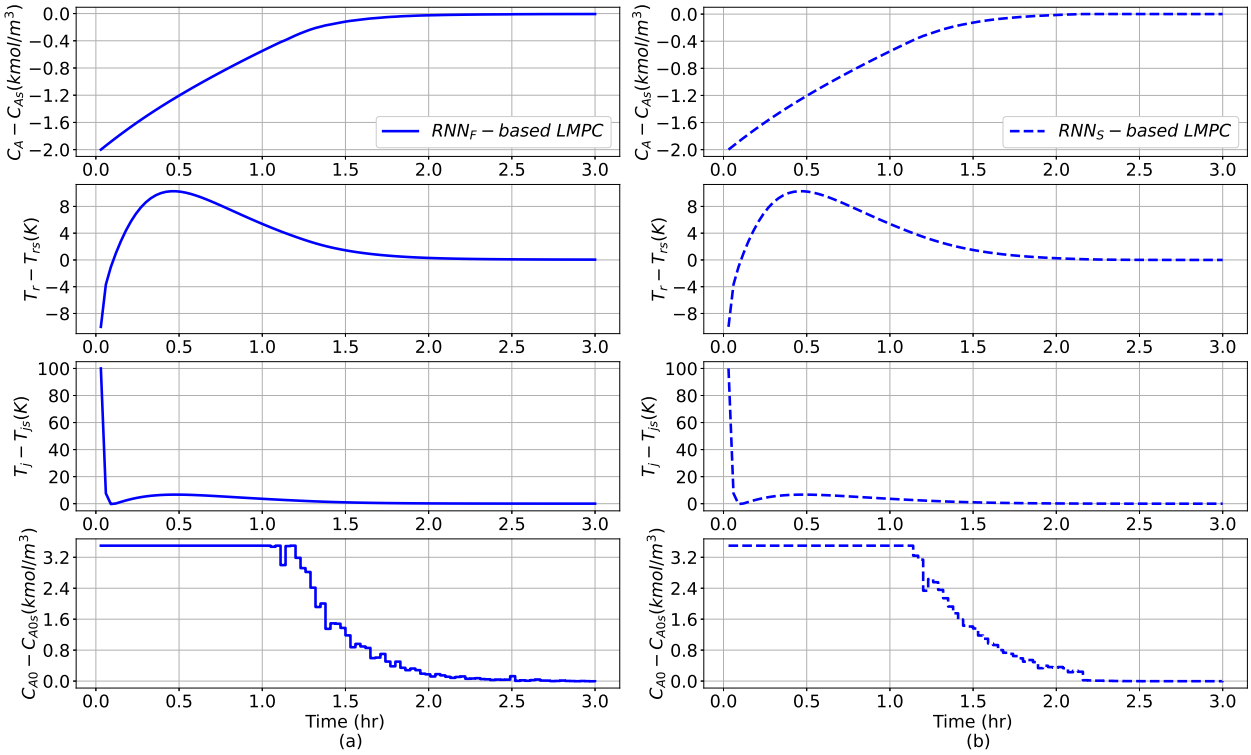

Figure 5.7: Considering the initial condition $IC_4 = (-2, -10, 100)$ (a) illustrates the time-varying profiles of the states and the input under RNN_F-based LMPC (solid line), while (b) shows the time-varying profiles of the states and the input under RNN_S-based LMPC (dashed line).

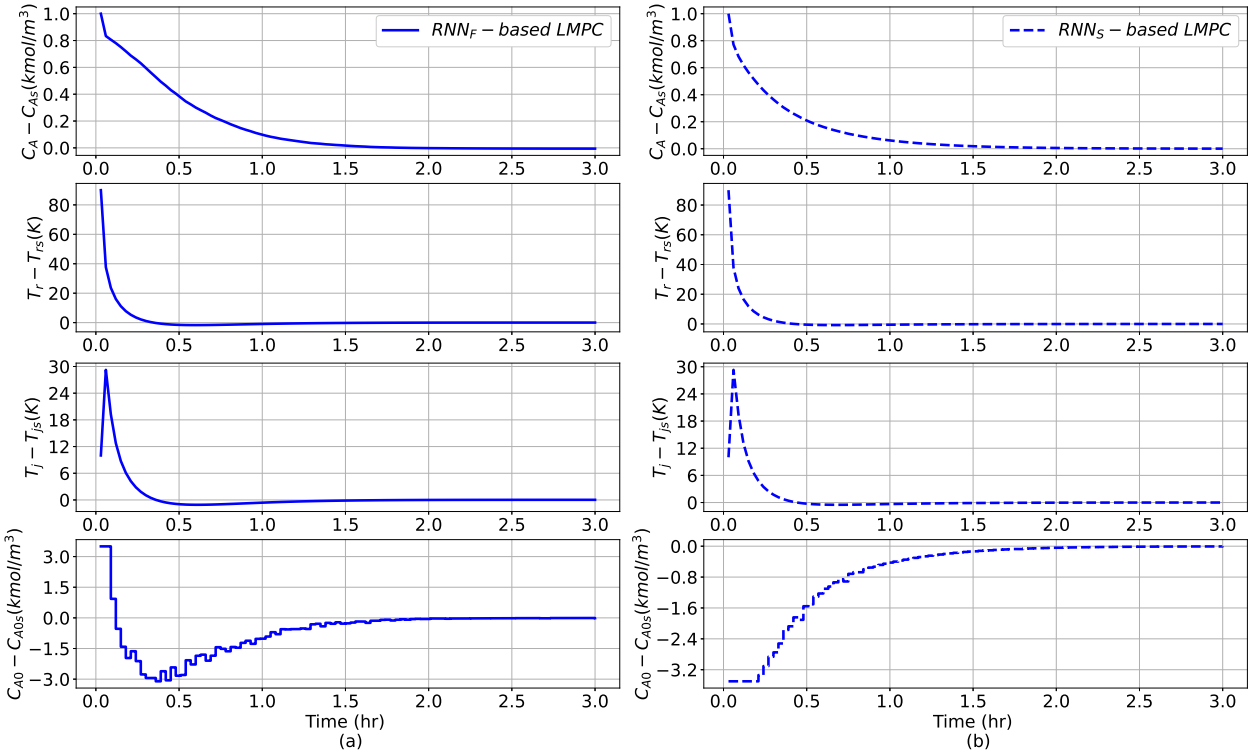

Figure 5.8: Considering the initial condition $IC_5 = (1, 90, 10)$ (a) illustrates the time-varying profiles of the states and the input under RNN_F-based LMPC (solid line), while (b) shows the time-varying profiles of the states and the input under RNN_S-based LMPC (dashed line).

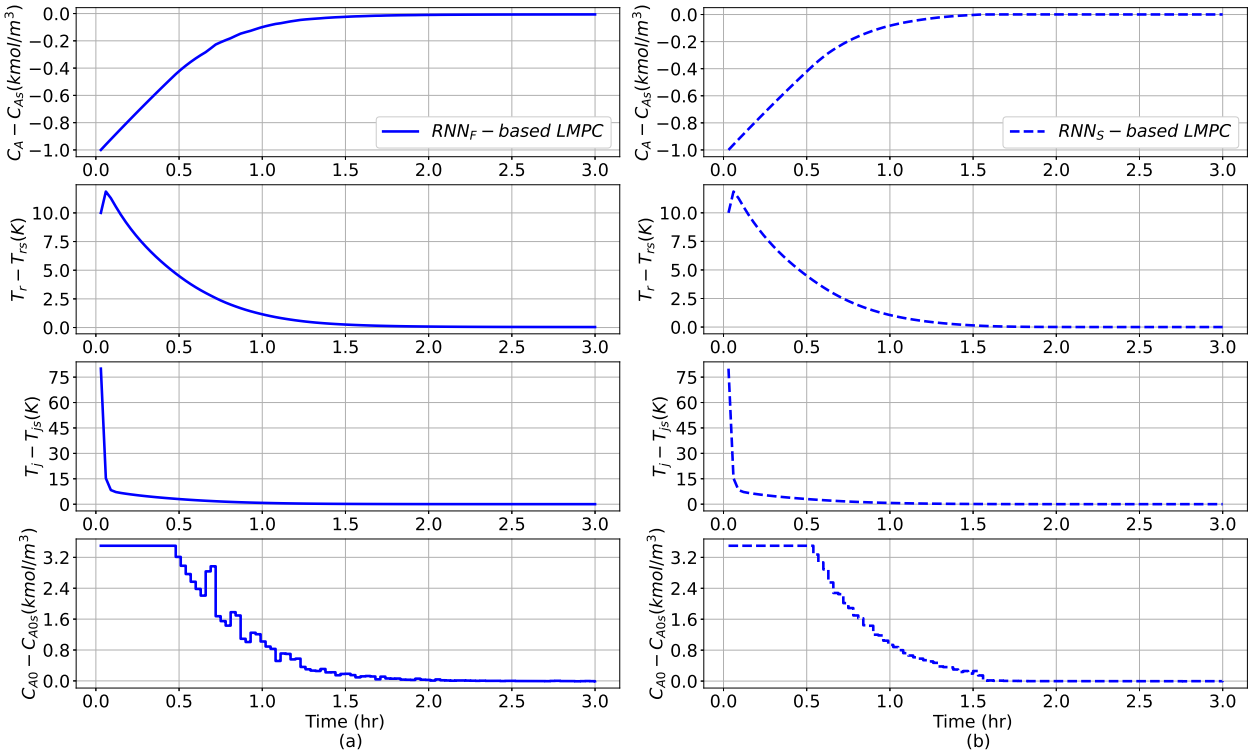

Figure 5.9: Considering the initial condition $IC_{10} = (-1, 10, 80)$ (a) illustrates the time-varying profiles of the states and the input under RNN_F-based LMPC (solid line), while (b) shows the time-varying profiles of the states and the input under RNN_S-based LMPC (dashed line).

Chapter 6

Conclusion

This dissertation discusses many advancements within the development of machine learning-based MPC frameworks to achieve closed-loop stability, enhance the process performance, and tackle challenges associated with complex nonlinear systems. First, the dissertation investigated machine learning-based model predictive control of time-delay systems, proposing a predictor methodology to compensate the effect of input delays, Next generalization error bounds were derived for different types of machine learning models using the theory of statistical machine learning. Moreover, a comprehensive study was conducted on neural network models to model two-time-scale systems, including an investigation of the generalization error bounds for these models.

In **Chapter 2**, we considered a nonlinear time-delay system expressed using nonlinear differential difference equations, and we approximated it with a perturbed nonlinear system with bounded perturbations. First, we introduced Long Short Term Memory Recurrent Neural Networks (LSTMs) to model the system dynamics in the absence of time delays and

used this LSTM model to construct an LSTM-based MPC. In the presence of small state delays, we established closed-loop stability under the LSTM-based MPC. Subsequently, the LSTM network was used to develop a closed-loop LSTM-based predictor, that compensated for the effect of input time-delays. Finally, we applied the proposed control schemes to a chemical reactor example with time-delays and demonstrated their ability to stabilize the closed-loop system under small and large state and input time-delays.

In **Chapter 3**, we developed machine-learning-based predictive control schemes for nonlinear systems subject to stochastic disturbances with unbounded variation and bounded disturbances, respectively. We first derived a generalization error bound for the RNN models developed for the nominal system using the Rademacher complexity method from statistical learning theory. Then, we established system stability results for the uncertain system with unknown disturbances in a bounded manner. With regards to the uncertain system with stochastic disturbances under RNN-MPC, we accounted for the distribution information of disturbances, and derived the probabilistic closed-loop stability properties. Through the simulation of a chemical reactor example, we demonstrated that the training data sample size affects the RNN generalization performance, and closed-loop stability for the MPC using RNN models.

In **Chapter 4**, we used the Rademacher complexity approach for vector-valued functions to create an upper bound for the generalization error of two classes of RNN models—partially-connected and fully-connected—and LSTM networks. For the partially-connected RNN, theoretical results connecting a RNN model's accuracy to its architecture were established and proved, as for the latter a generalizability bound for this specific structure of

RNNs. Open-loop simulations utilizing a complex, nonlinear chemical process was performed to demonstrate the superior model accuracy of the partially-connected RNN when compared to the fully-connected RNN across various testing data sets. Additionally, the developed partially-connected RNN model was then utilized in the design of a Lyapunov-based MPC. Through several closed-loop simulations, it was shown that adopting the partially-connected RNN model yielded smoother state trajectories with lower loss function values (i.e., smaller mean squared errors), and the applied inputs suffered less from oscillatory behavior.

In **Chapter 5**, we introduced a general class of nonlinear two-time-scale systems where the two distinct time scales can be decoupled into the slow subsystem and the fast subsystem dynamics using singular perturbation analysis. Machine learning was employed to approximate the dynamics of both subsystems. In particular, an RNN was used to predict the slow state vector or, in other words, to approximate the dynamics of the slow subsystem, while an FNN was used to approximate the dynamics of the fast subsystem. To draw inferences on the performance of the neural network models on unseen data, generalization error bounds for both neural networks when modeling two-time-scale systems were derived. Subsequently, the RNN modeling the slow states (RNN_S) was incorporated into an LMPC framework, and sufficient conditions to guarantee closed-loop stability were derived under the sample-and-hold implementation of the LMPC. Finally, a chemical process example was used to demonstrate the adequacy of the LMPC using an RNN to predict the slow states when compared to first-principles-based LMPC and also an LMPC using an RNN to model the full two-time-scale system. Through closed-loop simulations, while both RNN-based LMPCs were able to achieve the desired control objective of driving the system to the steady

state, due to the significantly lower complexity of the RNN approximating only the slow subsystem, the LMPC based on RNN_S required significantly less computational time in all simulations than the LMPC using the RNN modeling the full singularly perturbed system, which supported the use of RNN_S in real-time MPC applications.

Bibliography

[1] I. T. Cameron and K. Hangos. *Process modelling and model analysis*. Elsevier, 2001.

[2] R. Davies. Industry 4.0: Digitalisation for productivity and growth. Technical Report PE 568.337, European Parliamentary Research Service, 2015.

[3] S. Yin and O. Kaynak. Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103:143–146, 2015.

[4] F. Tahir and M. Khan. Big Data: the Fuel for Machine Learning and AI Advancement. Technical report, EasyChair, 2023.

[5] F. Chollet et al. Keras. https://keras.io, 2015.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[7] A. M. Schweidtmann, E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Technik*, 93(12):2029–2039, 2021.

[8] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[9] K. Zarzycki and M. Ławryńczuk. LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. *Sensors*, 21(16):5625, 2021.

[10] F. Bonassi, M. Farina, J. Xie, and R. Scattolini. On recurrent neural networks for learning-based control: recent results and ideas for future developments. *Journal of Process Control*, 114:92–104, 2022.

[11] Y. Zheng, C. Hu, X. Wang, and Z. Wu. Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. *Journal of Process Control*, 128:103005, 2023.

[12] T. Asrav and E. Aydin. Physics-informed recurrent neural networks and hyper-parameter optimization for dynamic process systems. *Computers & Chemical Engineering*, 173:108195, 2023.

[13] T. Xiao, Z. Wu, P. D. Christofides, A. Armaou, and D. Ni. Recurrent neural-network-based model predictive control of a plasma etch process. *Industrial & Engineering Chemistry Research*, 61(1):638–652, 2021.

[14] J. Meng, C. Li, J. Tao, Y. Li, Y. Tong, Y. Wang, L. Zhang, Y. Dong, and J. Du. Rnn-lstm-based model predictive control for a corn-to-sugar process. *Processes*, 11(4):1080, 2023.

[15] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. *AIChE Journal*, 65(11):e16734, 2019.

[16] K. Huang, K. Wei, F. Li, C. Yang, and W. Gui. LSTM-MPC: A deep learning based predictive control method for multimode process control. *IEEE Transactions on Industrial Electronics*, 2022.

[17] M. Jung, P. R. da Costa Mendes, M. Önnheim, and E. Gustavsson. Model Predictive Control when utilizing LSTM as dynamic models. *Engineering Applications of Artificial Intelligence*, 123:106226, 2023.

[18] N. L. Jian, H. Zabiri, and M. Ramasamy. Control of the multi-timescale process using multiple timescale recurrent neural network-based model predictive control. *Industrial & Engineering Chemistry Research*, 62(15):6176–6195, 2023.

[19] Q.-C. Zhong. *Robust control of time-delay systems*. Springer Science & Business Media, 2006.

[20] P. Kokotović, H. K. Khalil, and J. O'reilly. *Singular perturbation methods in control: analysis and design*. SIAM, 1999.

[21] Z. Wu, D. Rincon, Q. Gu, and P. D. Christofides. Statistical machine learning in model predictive control of nonlinear processes. *Mathematics*, 9:1912, 2021.

[22] K. Holkar and L. M. Waghmare. An overview of model predictive control. *International Journal of control and automation*, 3(4):47–63, 2010.

[23] M. Abu-Ayyad and R. Dubay. Real-time comparison of a number of predictive controllers. *ISA transactions*, 46(3):411–418, 2007.

[24] J. E. Normey-Rico and E. F. Camacho. *Model predictive control of dead-time processes*. Springer, 2007.

[25] O. J. Smith. Closer control of loops with dead time. *Chemical engineering progress*, 53:217–219, 1957.

[26] M. Schwenzer, M. Ay, T. Bergs, and D. Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, 2021.

[27] A. Visioli. *Practical PID control*. Springer Science & Business Media, 2006.

[28] C. R. Cutler and B. L. Ramaker. Dynamic matrix control—A computer control algorithm. In *joint automatic control conference*, page 72, San Francisco, CA, USA, 1980.

[29] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.

[30] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[31] M. Pirdashti, S. Curteanu, M. H. Kamangar, M. H. Hassim, and M. A. Khatami.

Artificial neural networks: applications in chemical engineering. *Reviews in Chemical Engineering*, 29(4):205–239, 2013.

[32] L. Hewing, K. Wabersich, M. Menner, and M. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[33] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[34] Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[35] S. Chen, Z. Wu, D. Rincon, and P. D. Christofides. Machine learning-based distributed model predictive control of nonlinear processes. *AIChE Journal*, 66:e17013, 2020.

[36] Z. Wu, D. Rincon, and P. D. Christofides. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control*, 89:74–84, 2020.

[37] J. Luo, B. Çıtmacı, J. B. Jang, F. Abdullah, C. G. Morales-Guio, and P. D. Christofides. Machine learning-based predictive control using on-line model linearization: Application to an experimental electrochemical reactor. *Chemical Engineering Research and Design*, 197:721–737, 2023.

[38] O. Santander, V. Kuppuraj, C. A. Harrison, and M. Baldea. Deep learning model

predictive control frameworks: Application to a fluid catalytic cracker–fractionator process. *Industrial & Engineering Chemistry Research*, 62(27):10587–10600, 2023.

[39] R. K. Al Seyab and Y. Cao. Differential recurrent neural network based predictive control. *Computers & Chemical Engineering*, 32(7):1533–1545, 2008.

[40] P. Kittisupakorn, P. Thitiyasook, M. A. Hussain, and W. Daosud. Neural network based model predictive control for a steel pickling process. *Journal of process control*, 19(4):579–590, 2009.

[41] M. Ellis and P. D. Christofides. Economic model predictive control of nonlinear time-delay systems: Closed-loop stability and delay compensation. *AIChE Journal*, 61(12):4152–4165, 2015.

[42] R. M. Esfanjani, M. Reble, U. Münz, S. K. Y. Nikravesh, and F. Allgöwer. Model predictive control of constrained nonlinear time-delay systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 1324–1329. IEEE, 2009.

[43] M. Reble, R. M. Esfanjani, S. K. Y. Nikravesh, and F. Allgöwer. Model predictive control of constrained non-linear time-delay systems. *IMA journal of mathematical control and information*, 28(2):183–201, 2011.

[44] M. Reble and F. Allgöwer. General design parameters of model predictive control for nonlinear time-delay systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 176–181. IEEE, 2010.

[45] M. Reble, F. D. Brunner, and F. Allgöwer. Model predictive control for nonlinear time-delay systems without terminal constraint. *IFAC Proceedings Volumes*, 44(1):9254–9259, 2011.

[46] N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. In *Conference on Learning Theory*, pages 297–299. PMLR, 2018.

[47] M. Chen, X. Li, and T. Zhao. On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*, 2019.

[48] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine Learning-Based Predictive Control of Nonlinear Processes. Part I: Theory. *AIChE Journal*, 65:e16729, 2019.

[49] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine Learning-Based Predictive Control of Nonlinear Processes. Part II: Computational Implementation. *AIChE Journal*, 65:e16734, 2019.

[50] S. Chen, Z. Wu, and P. D. Christofides. Decentralized machine-learning-based predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 162:45–60, 2020.

[51] X. Chen, M. Heidarinejad, J. Liu, and P. D. Christofides. Distributed economic MPC: Application to a nonlinear chemical process network. *Journal of Process Control*, 22:689–699, 2012.

[52] M. S. Alhajeri, Z. Wu, D. Rincon, F. Albalawi, and P. D. Christofides. Machine-

learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280, 2021.

[53] Z. Wu, J. Luo, D. Rincon, and P. D. Christofides. Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator. *Chemical Engineering Research and Design*, 168:275–287, 2021.

[54] Z. Wu, A. Alnajdi, Q. Gu, and P. D. Christofides. Statistical machine-learning–based predictive control of uncertain nonlinear processes. *AIChE Journal*, 68:e17642, 2022.

[55] J. Hoskins and D. Himmelblau. Process control via artificial neural networks and reinforcement learning. *Computers & Chemical Engineering*, 16:241–251, 1992.

[56] R. Vepa. A review of techniques for machine learning of real-time control strategies. *Intelligent Systems Engineering*, 2:77–90, 1993.

[57] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65:466–478, 2019.

[58] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[59] M. S. Alhajeri, F. Abdullah, Z. Wu, and P. D. Christofides. Physics-informed machine learning modeling for predictive control using noisy data. *Chemical Engineering Research and Design*, 186:34–49, 2022.

[60] J. K. Hale and S. M. V. Lunel. *Introduction to functional differential equations*, volume 99. Springer Science & Business Media, 2013.

[61] C. Antoniades and P. D. Christofides. Feedback control of nonlinear differential difference equation systems. *Chemical Engineering Science*, 54:5677–5709, 1999.

[62] C. Antoniades and P. D. Christofides. Robust control of nonlinear time-delay systems. *International Journal of Applied Mathematics and Computational Science*, 9:811–837, 1999.

[63] J. Liu, D. Munoz de la Pena, P. D. Christofides, and J. F. Davis. Lyapunov-based model predictive control of nonlinear systems subject to time-varying measurement delays. *International Journal of Adaptive Control and Signal Processing*, 23:788–807, 2009.

[64] X.-M. Zhang, Q.-L. Han, A. Seuret, F. Gouaisbaut, and Y. He. Overview of recent advances in stability of linear systems with time-varying delays. *IET Control Theory & Applications*, 13:1–16, 2019.

[65] O. J. Smith. Closed control of loop with dead time. *Chemical Engineering Progress*, 53:217–219, 1957.

[66] C. Kravaris and R. A. Wright. Deadtime compensation for nonlinear processes. *AIChE Journal*, 35:1535–1542, 1989.

[67] M. A. Henson and D. E. Seborg. Time delay compensation for nonlinear processes. *Industrial & Engineering Chemistry Research*, 33:1493–1500, 1994.

[68] Y. M. Ren, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, 165:107956, 2022.

[69] F. Abdullah, Z. Wu, and P. D. Christofides. Handling noisy data in sparse model identification using subsampling and co-teaching. *Computers & Chemical Engineering*, 157:107628, 2022.

[70] C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7:108–116, 1995.

[71] Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv preprint arXiv:1905.13210*, 2019.

[72] D. Zou and Q. Gu. An improved analysis of training over-parameterized deep neural networks. *arXiv preprint arXiv:1906.04688*, 2019.

[73] J. Hanson, M. Raginsky, and E. Sontag. Learning Recurrent Neural Net Models of Nonlinear Systems. *arXiv preprint arXiv:2011.09573*, 2020.

[74] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

[75] M. Mittal, M. Gallieri, A. Quaglino, S. Salehian, and J. Koutník. Neural Lyapunov model predictive control. *arXiv preprint arXiv:2002.10451*, 2020.

[76] D. Limon, J. Calliess, and J. Maciejowski. Learning-based nonlinear model predictive control. *IFAC-PapersOnLine*, 50:7769–7776, 2017.

[77] H. Deng and M. Krstić. Output-feedback stabilization of stochastic nonlinear systems driven by noise of unknown covariance. *Systems & Control Letters*, 39:173–182, 2000.

[78] H. Deng, M. Krstic, and R. J. Williams. Stabilization of stochastic nonlinear systems driven by noise of unknown covariance. *IEEE Transactions on automatic control*, 46:1237–1253, 2001.

[79] M. Mahmood and P. Mhaskar. Lyapunov-based model predictive control of stochastic nonlinear systems. *Automatica*, 48:2271–2276, 2012.

[80] T. Homer and P. Mhaskar. Output-feedback Lyapunov-based predictive control of stochastic nonlinear systems. *IEEE Transactions on Automatic Control*, 63:571–577, 2017.

[81] Z. Wu, J. Zhang, Z. Zhang, F. Albalawi, H. Durand, M. Mahmood, P. Mhaskar, and P. D. Christofides. Economic model predictive control of stochastic nonlinear systems. *AIChE Journal*, 64:3312–3322, 2018.

[82] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[83] A. Maurer. A vector-contraction inequality for rademacher complexities. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

[84] R. Khasminskii. *Stochastic stability of differential equations*, volume 66. Springer Science & Business Media, 2011.

[85] B. Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003.

[86] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[87] A. Cozad, N. V. Sahinidis, and D. C. Miller. A combined first-principles and data-driven approach to model building. *Computers & Chemical Engineering*, 73:116–127, 2015.

[88] Z. T. Wilson and N. V. Sahinidis. The ALAMO approach to machine learning. *Computers & Chemical Engineering*, 106:785–795, 2017.

[89] H. Han, X. Wu, and J. Qiao. Real-time model predictive control using a self-organizing neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 24:1425–1436, 2013.

[90] J. M. Ali, M. A. Hussain, M. O. Tade, and J. Zhang. Artificial Intelligence techniques applied as estimator in chemical process systems–A literature survey. *Expert Systems with Applications*, 42:5915–5931, 2015.

[91] W. C. Wong, E. Chee, J. Li, and X. Wang. Recurrent neural network-based model

predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6:242, 2018.

[92] H. Shahnazari, P. Mhaskar, J. M. House, and T. I. Salsbury. Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Computers & Chemical Engineering*, 126:189–203, 2019.

[93] J. Fan and M. Han. Nonliear model predictive control of ball-plate system based on gaussian particle swarm optimization. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–6. Birsbane, Australia, 2012.

[94] J. Xu, C. Li, X. He, and T. Huang. Recurrent neural network for solving model predictive control problem in application of four-tank benchmark. *Neurocomputing*, 190:172–178, 2016.

[95] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: A review. *Computers & Chemical Engineering*, 20:743–791, 1996.

[96] S. F. De Azevedo, B. Dahm, and F. Oliveira. Hybrid modelling of biochemical processes: A comparison with the conventional approach. *Computers & Chemical Engineering*, 21:S751–S756, 1997.

[97] O. Kahrs and W. Marquardt. The validity domain of hybrid models and its application in process optimization. *Chemical Engineering and Processing: Process Intensification*, 46:1054–1066, 2007.

[98] N. Patel, J. Nease, S. Aumi, C. Ewaschuk, J. Luo, and P. Mhaskar. Integrating Data-Driven Modeling with First-Principles Knowledge. *Industrial & Engineering Chemistry Research*, 59:5103–5113, 2020.

[99] Y. Lu, M. Rajora, P. Zou, and S. Liang. Physics-embedded machine learning: case study with electrochemical micro-machining. *Machines*, 5:4, 2017.

[100] M. S. Alhajeri, J. Luo, Z. Wu, F. Albalawi, and P. D. Christofides. Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design*, 179:77–89, 2022.

[101] Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems & Control Letters*, 16:393–397, 1991.

[102] L. A. Zadeh. Probability measures of fuzzy events. *Journal of Mathematical Analysis and Applications*, 23:421–427, 1968.

[103] S. Liao. Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, 28:93–103, 2005.

[104] C. Lee. Fuzzy logic in control systems: fuzzy logic controller. I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:404–418, 1990.

[105] A. Banerjee, D. Varshney, S. Kumar, P. Chaudhary, and V. K. Gupta. Biodiesel production from castor oil: ANN modeling and kinetic parameter estimation. *International Journal of Industrial Chemistry*, 8:253–262, 2017.

[106] A. Singh, H. P. Singh, and S. Mishra. Validation of ANN-based model for binary distillation column. In *Proceeding of International Conference on Intelligent Communication, Control and Devices*, pages 235–242. Springer, Singapore, 2017.

[107] A. C. S. R. Dias, W. B. da Silva, and J. C. S. Dutra. Propylene polymerization reactor control and estimation using a particle filter and neural network. *Macromolecular Reaction Engineering*, 11:1700010, 2017.

[108] Y. Pan and J. Wang. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 59:3089–3101, 2011.

[109] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29:2318–2331, 2017.

[110] M. Alber, A. Buganza Tepole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold, et al. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ Digital Medicine*, 2:1–11, 2019.

[111] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, et al. Workshop report on basic research needs for sci-

entific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[112] R. Rai and C. K. Sahu. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access*, 8:71050–71073, 2020.

[113] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, et al. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566:195–204, 2019.

[114] V. M. Krasnopolsky and M. S. Fox-Rabinovitz. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, 19:122–134, 2006.

[115] P. A. O'Gorman and J. G. Dwyer. Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, 10:2548–2563, 2018.

[116] R. Cang, H. Li, H. Yao, Y. Jiao, and Y. Ren. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *Computational Materials Science*, 150:212–221, 2018.

[117] G. R. Schleder, A. C. Padilha, C. M. Acosta, M. Costa, and A. Fazzio. From DFT to

machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials*, 2:032001, 2019.

[118] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, 30, 2017.

[119] I. Chakraborty, K. J. Bodurtha, N. J. Heeder, M. P. Godfrin, A. Tripathi, R. H. Hurt, A. Shukla, and A. Bose. Massive electrical conductivity enhancement of multilayer graphene/polystyrene composites using a nonconductive filler. *ACS Applied Materials & Interfaces*, 6:16472–16475, 2014.

[120] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS Computational Biology*, 16:e1007575, 2020.

[121] T. Xu and A. J. Valocchi. Data-driven methods to improve baseflow prediction of a regional groundwater model. *Computers & Geosciences*, 85:124–136, 2015.

[122] J. H. Faghmous and V. Kumar. A big data guide to understanding climate change: The case for theory-guided data science. *Big Data*, 2:155–163, 2014.

[123] L. Wang, J. Chen, and M. Marathe. Tdefsi: Theory-guided deep learning-based epidemic forecasting with synthetic information. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6:1–39, 2020.

[124] X. Jia, A. Khandelwal, D. J. Mulla, P. G. Pardey, and V. Kumar. Bringing automated, remote-sensed, machine learning methods to monitoring crop landscapes at scale. *Agricultural Economics*, 50:41–50, 2019.

[125] J. C. Butcher. A history of Runge-Kutta methods. *Applied Numerical Mathematics*, 20:247–260, 1996.

[126] P. Sagaut, M. Terracol, and S. Deck. *Multiscale and multiresolution approaches in turbulence-LES, DES and Hybrid RANS/LES Methods: Applications and Guidelines*. World Scientific, 2013.

[127] B. Houska, F. Logist, M. Diehl, and J. V. Impe. A tutorial on numerical methods for state and parameter estimation in nonlinear dynamic systems. *Identification for Automotive Systems*, pages 67–88, 2012.

[128] B. Chaouat. The state of the art of hybrid RANS/LES modeling for the simulation of turbulent flows. *Flow, turbulence and combustion*, 99:279–327, 2017.

[129] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, pages 3424–3433, Sydney, Australia, 2017.

[130] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

[131] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118:e2101784118, 2021.

[132] J. Luo, V. Canuso, J. B. Jang, Z. Wu, C. G. Morales-Guio, and P. D. Christofides. Machine learning-based operational modeling of an electrochemical reactor: Handling data variability and improving empirical models. *Industrial & Engineering Chemistry Research*, 61:8399–8410, 2022.

[133] A. Canning and E. Gardner. Partially connected models of neural networks. *Journal of Physics A: Mathematical and General*, 21:3275, 1988.

[134] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[135] R. Roelofs. *Measuring Generalization and Overfitting in Machine learning*. Doctoral Dissertation, University of California, Berkeley, 2019.

[136] F. Emmert-Streib and M. Dehmer. Evaluation of regression models: Model assessment, model selection and generalization error. *Machine Learning and Knowledge Extraction*, 1:521–551, 2019.

[137] M. S. Alhajeri, Z. Wu, D. Rincon, F. Albalawi, and P. D. Christofides. Machine-learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280, 2021.

[138] L. T. Biegler and V. M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33:575–582, 2009.

[139] M. S. Alhajeri and M. Soroush. Tuning guidelines for model-predictive control. *Industrial & Engineering Chemistry Research*, 59:4177–4191, 2020.

[140] P. D. Christofides and P. Daoutidis. Feedback control of two-time-scale nonlinear systems. *International Journal of Control*, 63:965–994, 1996.

[141] X. Chen, M. Heidarinejad, J. Liu, and P. D. Christofides. Composite fast-slow MPC design for nonlinear singularly perturbed systems. *AIChE Journal*, 58:1802–1811, 2012.

[142] M. Ellis, M. Heidarinejad, and P. D. Christofides. Economic model predictive control of nonlinear singularly perturbed systems. *Journal of Process Control*, 23:743–754, 2013.

[143] F. Abdullah, Z. Wu, and P. D. Christofides. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Computers & Chemical Engineering*, 153:107411, 2021.

[144] A. Nikolakopoulou and R. D. Braatz. Polynomial NARX-based nonlinear model predictive control of modular chemical systems. *Computers & Chemical Engineering*, page 108272, 2023.

[145] S. Yang, M. P. Wan, W. Chen, B. F. Ng, and S. Dubey. Experiment study of machine-

learning-based approximate model predictive control for energy-efficient building control. *Applied Energy*, 288:116648, 2021.

[146] A. Alessio and A. Bemporad. A survey on explicit model predictive control. *Nonlinear Model Predictive Control: Towards New Challenging Applications*, pages 345–369, 2009.

[147] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American control conference (ACC)*, pages 1520–1527. IEEE, 2018.

[148] Y. Bao, K. J. Chan, A. Mesbah, and J. M. Velni. Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust Bayesian neural networks. *International Journal of Robust and Nonlinear Control*, 33(5):3312–3333, 2023.

[149] S. Chen, Z. Wu, and P. D. Christofides. Statistical Machine-Learning-based Predictive Control Using Barrier Functions for Process Operational Safety. *Computers & Chemical Engineering*, 163:107860, 2022.

[150] M. S. Alhajeri, A. Alnajdi, F. Abdullah, and P. D. Christofides. On generalization error of neural network models and its application to predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 189:664–679, 2023.

[151] F. Hoppensteadt. Properties of solutions of ordinary differential equations with small parameters. *Communications on Pure and Applied Mathematics*, 24:807–840, 1971.

[152] Y. Bao, H. S. Abbas, and J. M. Velni. A learning- and scenario-based MPC design for nonlinear systems in LPV framework with safety and stability guarantees. *International Journal of Control*, 0(0):1–20, 2023.

[153] P. D. Christofides and A. R. Teel. Singular perturbations and input-to-state stability. *IEEE Transactions on Automatic Control*, 41:1645–1650, 1996.

[154] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[155] L. Pöri. Comparison of two interior point solvers in model predictive control optimization. Master's thesis, Aalto University, 2016.

[156] P. J. Freire, S. Srivallapanondh, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn. Computational complexity evaluation of neural network applications in signal processing. *arXiv preprint arXiv:2206.12191*, 2022.