Title

Process Structure-Aware Machine Learning Modeling for State Estimation and Model Predictive Control of Nonlinear Processes

Author

Alhajeri, Mohammed S.

Publication Date

2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Process Structure-Aware Machine Learning Modeling for State Estimation and Model Predictive Control of Nonlinear Processes

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Chemical Engineering

by

Mohammed Alhajeri

2022

ABSTRACT OF THE DISSERTATION

Process Structure-Aware Machine Learning Modeling for State Estimation and Model Predictive Control of Nonlinear Processes

by

Mohammed Alhajeri

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2022

Professor Panagiotis D. Christofides, Chair

Big data is a cornerstone component of the fourth industrial revolution, which calls on engineers and researchers to fully utilize data in order to make smart decisions and enhance the efficiency of industrial processes as well as control systems. In practice, industrial process control systems typically rely on a data-driven model (often linear) with parameters that are determined by industrial/simulation data. However, in some scenarios, such as in profit-critical or quality-critical control loops, first-principles concepts that are based on the underlying physico-chemical phenomena may also need to be employed in the modeling phase to improve data-based process models. Hence, process systems engineers still face significant challenges when it comes to modeling large-scale, complicated nonlinear processes. Modeling will continue to be crucial since process models are essential components of cutting-edge model-based control systems, such as model predictive control (MPC).

Machine learning models have a lot of potential based on their success in numerous applications. Specifically, recurrent neural network (RNN) models, designed to account for every input-output interconnection, have gained popularity in providing approximation of various highly nonlinear chemical processes to a desired accuracy. Although the training error of neural networks

that are dense and fully-connected may often be made sufficiently small, their accuracy can be further improved by incorporating prior knowledge in the structure development of such machine learning models. Physics-based recurrent neural networks modeling has yielded more reliable machine learning models than traditional, fully black-box, machine learning modeling methods. Furthermore, the development of systematic and rigorous approaches to integrate such machine learning techniques into nonlinear model-based process control systems is only getting started. In particular, physics-based machine learning modeling techniques can be employed to derive more accurate and well-conditioned dynamic process models to be utilized in advanced control systems such as model predictive control. Along with Lyapunov-based stability constraints, this scheme has the potential to significantly improve process operational performance and dynamics. Hence, investigating the effectiveness of this control scheme under the various long-standing challenges in the field of process systems engineering such as incomplete state measurements, and noise and uncertainty is essential. Also, a theoretical framework for constructing and assessing the generalizability of this type of machine learning models to be utilized in model predictive control systems is lacking.

In light of the aforementioned considerations, this dissertation addresses the incorporation of prior process knowledge into machine learning models for model predictive control of nonlinear chemical processes. The motivation, background and outline of this dissertation are first presented. Then, the use of machine learning modeling techniques to construct two different data-driven state observers to compensate for incomplete process measurements is presented. The closed-loop stability under Lyapunov-based model predictive controllers is then addressed. Next, the development of process-structure-based machine learning models to approximate large, nonlinear chemical processes is presented, with the improvements yielded by this approach demonstrated via open-loop and closed-loop simulations. Subsequently, the reliability of process-structure-based machine learning models is investigated in the presence of different types of industrial noise. Two novel approaches are proposed to enhance the accuracy of machine learning models in the presence of noise. Lastly, a theoretical framework that connects the accuracy of an RNN model to

its structure is presented, where an upper bound on a physics-based RNN model's generalization error is established. Nonlinear chemical process examples are numerically simulated or modeled in Aspen Plus Dynamics to illustrate the effectiveness and performance of the proposed control methods throughout the dissertation.

The dissertation of Mohammed Alhajeri is approved.

Carlos G. Morales Guio

Dante A. Simonetti

Tsu-Chin Tsao

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2022

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

First and foremost, I would want to praise and thank God, the All-Powerful, who has given the writer innumerable blessings, knowledge, and opportunities as a result of which I have finally been able to complete this thesis.

I would like to sincerely thank my advisor Professor Panagiotis D. Christofides for his unending support throughout my doctoral studies. I recall vividly the moment I first contacted Professor Christofides in the Winter of 2018 and how welcoming he was. Throughout my doctoral studies, he was always available to assist me and point me in the right direction. I consider myself extremely fortunate to have completed my Ph.D. under his supervision during my time here at UCLA. Professor Christofides is an outstanding researcher, mentor, educator, and source of inspiration. His guidance on control theory, process systems engineering, and most notably life decisions has had a significant impact on both my personal life and professional life as well. In the same vein, I would also like to thank Professor Dante A. Simonetti, Professor Carlos Morales-Guio and Professor Tsu-Chin Tsao, for serving on my doctoral committee.

In addition, I would like to thank all of my colleagues with whom I have worked over the years in the Christofides research group, including Dr. Zhihao Zhang, Dr. Yangyao Ding, Dr. Yichi Zhang, Mr. Matthew Tom, and Mr. Berkay Citmaci. I would particularly like to thank Professor Zhe Wu, Professor Fahad Albalawi, Dr. David Rincon, Mr. Junwei Luo, Dr. Yi Ming Ren, Mr. Fahim Abdullah, Ms. Aisha Alnajdi, Dr. Scarlett Chen, and Mr. Atharva Suryavanshi with whom I have collaborated extensively and spent long hours working on papers together. It was a nice and enriching voyage!

I was also fortunate to meet and get to know a number of people outside of my research group during my time at UCLA which made the experience much more enjoyable. I would like to thank my friend Dr. Anas Alaqeeli for our extensive philosophical and technical conversations about everything from stock markets to where to get the finest coffee in Los Angeles. This extends to Aziz Alawadi, Dr. Mohammaed Alabdullah, Alwaleed Aldhefieri, Dr. Ali Alsheri, Ali Dashti, Fahad Alburiki, M. Galadari with whom I had the pleasure of being good friends, in addition to

going through various adventures, playing sports, and hiking with them.

Last but not least, I am deeply indebted to my role model and father, my beloved mother, my siblings, my spouse, and my two little angels for their unwavering support and never-ending encouragement throughout my years of study as well as during the process of researching and writing this thesis. It would not have been possible to finish this task without them. In the same vein, thanks to all my friends back in Kuwait, specially Mohammed Alshammari, their kind words always pushed me forward. Also, I want to express my gratitude to my late grandparents, they will always be in my heart. They both helped shape the person I am today and gave me the courage and desire to pursue my higher education studies, and to always become a better version of myself.

Financial support from Kuwait University is gratefully acknowledged, and my work could not have been done without this support. In addition, I appreciate the role of the scholarship committee in Chemical Engineering Department and the Cultural Relations Department at Kuwait University with special thanks to Reeham Aladwhani and Sameerah Almansour. Also, I appreciate the services of the Kuwait Cultural office in Los Angeles, specifically, Dr. Mohammed Alreshidi, Dr. Hassan Alkanderi, Tatiana Mollahzadeh, and Anna Kim.

Chapter 2 contains versions of: Alhajeri, M., Z. Wu, D. Rincon, F. Albalawi and P. D. Christofides, "Machine Learning-Based State Estimation and Predictive Control of Nonlinear Processes, " *Chem. Eng. Res. & Des.*, 167, 268-280, 2021.

Chapter 3 contains versions of: Alhajeri, M., J. Luo, Z. Wu, F. Albalawi and P. D. Christofides, "Process Structure-Based Recurrent Neural Network Modeling for Predictive Control: A Comparative Study, " *Chem. Eng. Res. & Des.*, 179, 77-89, 2022.

Chapter 4 contains versions of: Alhajeri, M., F. Abdullah, Z. Wu and P. D. Christofides, "Physics-informed Machine Learning Modeling for Predictive Control Using Noisy Data, " *Chem. Eng. Res. & Des.*, 186, 34-49, 2022.

Chapter 5 contains versions of: Alhajeri, M., 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.*, submitted.

Curriculum Vitae

Education

Drexel University **Sep. 2016 - Sep. 2018**

M.S., Chemical Engineering **Philadelphia, PA**

Supervisor: Prof. Masoud Soroush

Thesis: "Nonlinear Model Predictive Control of Processes with Incomplete State Measurements"

Kuwait University **Sep. 2010 - Dec. 2015**

B.S., Chemical Engineering **Khalidya, Kuwait**

Experince

Ministry of Electricity and Water **Jan. 2016 - Aug. 2016**

Al-Zour Desalination Plant **Al-Zour, Kuwait**

Process engineer

Publications

1. **Alhajeri, M. S.**, A. Alnajdi, Z. Wu and P. D. Christofides, "Statistical Machine Learning in Model Predictive Control: An Overview of Recent Results," *Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control*, 6 pages, San Antonio, Texas, 2023.

2. **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.*, submitted.

4. Abdullah, F., **M. S. Alhajeri**, and P. D. Christofides, "Modeling and control of nonlinear processes using sparse identification: Using dropout to handle noisy data," *Industrial & Engineering Chemistry Research*, in press.

5. **Alhajeri, M. S.**, F. Abdullah, Z. Wu and P. D. Christofides, "Physics-informed Machine Learning Modeling for Predictive Control Using Noisy Data," *Chem. Eng. Res. & Des.*, 186, 34-49, 2022.

6. Ren Y., **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.

7. **Alhajeri, M. S.**, J. Luo, Z. Wu, F. Albalawi and P. D. Christofides, "Process Structure-Based Recurrent Neural Network Modeling for Predictive Control: A Comparative Study," *Chem. Eng. Res. & Des.*, 179, 77-89, 2022.

8. **Alhajeri, M. S.**, Z. Wu, D. Rincon, F. Albalawi and P. D. Christofides, "Machine Learning-Based State Estimation and Predictive Control of Nonlinear Processes, " *Chem. Eng. Res. & Des.*, 167, 268-280, 2021.

9. **Alhajeri, M. S.**, Wu, Z., Rincon, D., Albalawi, F. and Christofides, P.D., "Estimation-Based Predictive Control of Nonlinear Processes Using Recurrent Neural Networks," *Proceedings of 16th IFAC International Symposium on Advanced Control of Chemical Processes*, 6 pages, Venice, Italy, 2021.

10. **Alhajeri, M. S.** and M. Soroush, "Tuning Guidelines for Model Predictive Control," *Industrial & Engineering Chemistry Research*, 59, 4177–4191, 2020.

Chapter 1

Introduction

1.1 Motivation

Mathematical models that describe the relationship between the manipulated inputs and the process outputs are essential for constructing model-based control systems for industrial applications. Currently, the development of a process model is based on either first-principles or process data under various assumptions depending on the process of interest. However, certain limitations on model performance exist. For example, linearized models of nonlinear processes are only valid around a vicinity of the operating point that is used in the linearization. Furthermore, due to the dynamical nature alongside with the inherent nonlinear behavior and high complexity of most of chemical processes, it is usually not an easy task to find an accurate first-principles model.

Modern industries have seen an incredible increase in data availability over the past ten years; it is anticipated that, each year, machines and devices produce more than 1000 Exabytes of data (e.g., [124]). Industrial process control systems typically use a (linear) data-driven model with factors that are defined by industrial/simulation data [25, 108], though in some circumstances, such as in profit-critical control loops, first-principles models (with data-determined model parameters) that describe the underlying physico-chemical phenomena may also be used. Nevertheless, process systems engineering experts still face significant challenges when it comes to modeling large-scale,

complicated nonlinear processes. Model quality is influenced by a variety of variables, such as model parameter estimates, model uncertainty, the number of assumptions used during model construction, the model's dimensionality, its structure, and the amount of computational power required to solve the model in real-time (e.g., [37, 38]). Machine learning has received more focus in the identification of models in recent years. Recurrent neural networks (RNNs) are a popular machine learning technique for modeling general classes of nonlinear dynamical systems [23, 85]. Tensorflow and Keras are two examples of open-source software libraries for machine learning applications that have been developed over the past years concurrently with the development of machine learning algorithms and computing resources/platforms. These advancements have facilitated the usages of machine learning techniques beyond the computer science field to traditional engineering fields (e.g., [10, 80, 100, 112, 117]). In particular, since RNNs are able to approximate steady-state input-output interrelations and different systems' nonlinear dynamic behavior, feed-forward neural network (FNN) models and RNN models and their variants (e.g., [23, 40, 53, 85]), have shown promise for use in model-based control systems. In order to further enhance the performance of machine learning models, real-time data sets gathered from numerous sensors can be used for online adaptation and training to minimize modeling error and account for model uncertainties, and parallel computing units can be utilized to expedite calculations for real-time tasks like process control. Designing model-based control systems that utilize machine learning modeling methods to leverage massive data sets is, therefore, a major breakthrough in process system engineering that will have a vast influence on the industry, especially on the following generation of industrial control systems.

The fundamental benefit of data-driven modeling techniques is that no prior knowledge of the process is required. However, historical data including key measured process variables is required to develop data-driven models. The effectiveness of data-driven modeling methods is based on the quality and amount of process data. Moreover, owing to their capability to analyze data in vast quantities from industrial processes, machine learning techniques have recently attracted significant attention in traditional engineering fields. Machine learning is a broad family of

techniques including neural networks and their variants, which have been used successfully in regression and classification problems such as process modeling, process monitoring, and fault detection. RNNs and long short-term memory (LSTM) networks are two of the many types of neural networks that have gained popularity for modeling nonlinear dynamic systems from sequential data (i.e., time-series data) and have been used in advanced control strategies such as model predictive control (MPC) to predict the evolution of process states when first-principles process models are unavailable (e.g., [21, 29, 118]).

Standard RNN models, also known as fully-connected RNN models, are a popular candidate for analyzing time-series data within a black-box modeling framework. Such a modeling methodology, however, may not always be optimal, particularly for large chemical processes due to the complex interactions among process variables. Hence, to improve RNN performance, several studies (e.g., [48, 92, 129]) have looked into gray-box modeling, also referred to as hybrid modeling, which involves integration of prior physical knowledge and expertise into the modeling of neural networks. Another method is to reflect physical relations among the given process inputs and outputs into the modeling of neural networks (i.e., known as partially-connected modeling), which was discussed in several works such as in [62, 114], where they were shown to yield better closed-loop performance.

1.2 Background

With the continuous improvement of data availability and accessibility, machine-learning-based model predictive control (MPC) methods have received increasing attention as the next generation of control systems. Conceptually, an MPC contains three major components: a predictive model, an objective function and constraints, and a process optimizer [15]. By using the neural network as the predictive model, the MPC can capture the process dynamics and accordingly make smart decisions: approaching the target states efficiently, automatically, and economically. Furthermore, recent works have demonstrated that ML-based

MPC can be utilized to deal with various challenging tasks to improve manufacturing processes such as suppressing measurement noise and searching optimum economic benefits, which classical control techniques are incapable of accomplishing [30, 113]. Moreover, data-driven modeling [102] has historically drawn substantial attention in the context of MPC due to the fact that it is typically challenging to develop a first-principles model that captures complicated, nonlinear behavior of a large-scale process. According to studies [97, 99], modeling with neural networks has been effective in approximating nonlinear dynamical systems. Neural networks may better capture "difficult nonlinearities" due to a broader class of learnable nonlinear functions than polynomial approximation [72], the latter of which is often simple to solve. RNN-based modeling has been the subject of much study, which also helps to construct model-based control schemes that employ data-driven models to forecast process dynamics [73, 109].

The investigation of utilizing artificial intelligence (AI) techniques in chemical engineering has been carried out intensively. AI technology has provided classic and powerful modeling tools such as fuzzy logic in the 1960s [125], expert systems in the 1980s [57,59], and machine learning in the 1990s [101]. Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications [12, 27, 89, 109]. For example in [12], an artificial neural network (ANN) model was developed for a bio-diesel production process, where the ANN model provided an approximation of the percentage of fatty acid methyl ester yield within $\pm 8\%$ deviation from the experimental data. Additionally, among various ML modeling techniques, RNNs have been broadly employed for modelling a general class of dynamical systems for control and state estimation purposes [73]. In [89], a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the RNN model prediction could outperform a first-principles model for large-scale, complex, nonlinear process, due to its high degree of freedom to solve the complex non-linear regression problem with the process dataset.

Due to their high data needs, failure to yield physically consistent outputs, and lack of generalizability to out-of-sample conditions, even the most cutting-edge black box ML models

(for example dense fully-connected RNN models) have had only sporadic success when applied in scientific domains [49]. The research community is starting 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 an ML-only nor a scientific knowledge-only approach can be considered sufficient for complex scientific and engineering applications [3, 11, 79]. This paradigm uses domain-specific knowledge, but in supporting roles, such as feature engineering or post-processing, in a fundamentally different way than dominant approaches in the ML community. On the other hand, although the concept of combining scientific principles and ML models has only recently gained popularity [49], there has already been a substantial amount of research done on the subject. This research direction is being conducted in various disciplines other than chemical engineering including earth systems [81], climatology [51, 70], material exploration [16, 84], quantum chemistry [18, 86], biological sciences [123], 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 that humanity is currently facing with regard to the environment [33], health sciences [105], and food and nutrition security [47].

Similarly, in process systems engineering and chemical engineering disciplines, the present paradigm of numerical approaches to get approximate solutions is based solely on physics: numerical differentiation and integration algorithms are used to solve for systems of associated differential equations that reflect established physical principles throughout space and time [14, 45, 83]. 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, for turbulent flows, one may utilize Reynolds-averaged Navier-Stokes equations [19, 96]. But creating a simplified model that accurately captures a phenomenon is quite difficult. More crucially, only a portion of the dynamics of many complicated real-world processes is recognized. The equations may not accurately reflect the actual system states. On the other hand, numerous recent studies, from turbulence and reaction modeling to state prediction, have demonstrated that ML models can

produce realistic predictions and greatly speed up the simulation of complex dynamics compared to numerical solvers [50, 63]. However, ML models are dense and purely data-driven by nature, which has many limitations. Without strict boundaries, ML models are likely to provide predictions that defy the fundamental principles governing physical systems. Furthermore, ML models frequently experience difficulties with generalization, i.e., models trained on a single dataset cannot adequately adapt to unseen scenarios. Hence, approximating complicated dynamical systems in scientific areas cannot be considered to be a problem that is sufficiently solved by either ML models alone or simply physics-based methods. As a result, there is a significant necessity to integrate ML models with conventional physics-based methodologies, through which we can maximize the benefits of both techniques.

1.3 Dissertation Objectives and Structure

In the context of machine-learning-based model predictive control systems, this dissertation proposes control theoretic approaches to process operation and modelling. Chemical process examples are employed to demonstrate the applications of the suggested control strategies. The following summarizes this dissertation's goals in further detail:

1. To present a framework of integrating machine-learning-based state estimator implemented in the contexts of nonlinear chemical processes under machine-learning-based MPC, while ensuring closed-loop stability via Lyabunov stability constraints.

2. To develop a partially-connected RNN model based on the physical input-output relationship of highly nonlinear and complex chemical processes in the context of RNN-based MPC.

3. Overcoming the over-fitting issue originating due to the presence of noise in training data of partially-connected RNN models.

4. Developing a theoretical generalization error upper bound for a class of partially-connected RNN models.

The dissertation comprises of 6 chapters and is organized as follows.

As full state measurements may be unavailable in chemical plants, in chapter 2, we propose two machine learning-based state estimation approaches. The first approach integrates a RNN model within the extended Luenberger observer framework to develop data-based state estimators. The second approach utilizes a hybrid model that integrates feed-forward neural networks with first-principles models to capture process dynamics in the state estimator. Then, an output feedback model predictive controller is designed based on the state estimates provided by the machine-learning-based estimators to stabilize the closed-loop system at the steady-state. A chemical process example is utilized to illustrate the effectiveness of the proposed machine-learning-based state estimation and control approaches.

RNN models have demonstrated their ability in providing a remarkably accurate modeling approximation to describe the dynamic evolution of complex, nonlinear chemical processes in several applications. Although conventional fully-connected RNN models have been successfully utilized in MPC to regulate chemical processes with desired approximation accuracy, the development of RNN models in terms of model structure can be further improved by incorporating physical knowledge to achieve better accuracy and computational efficiency. Hence, chapter 3 investigates the performance of MPC based on two different RNN structures. Specifically, a fully-connected RNN model, and a partially-connected RNN model developed using a prior physical knowledge are considered. This study uses an example of a large-scale complex chemical process simulated by Aspen Plus Dynamics to demonstrate improvements in the RNN model and an RNN-based MPC, when prior knowledge of the process is taken into account.

In chapter 3, the proposed RNN modeling method was developed with clean training data. However, in practice, and generally in chemical processes, noise-free data is not always available. Furthermore, due to the occurrence of over-fitting at the learning phase, the modeling of chemical processes via ANNs by using corrupted data (i.e., noisy data) is an ongoing challenge. Therefore, Chapter 4 investigates the effect of both Gaussian and non-Gaussian noise on the performance of process-structure based RNN models, which take the form of partially-connected RNN

models, that are used to approximate a class of multi-input-multi-outputs nonlinear systems. Furthermore, two different techniques, specifically Monte Carlo dropout and co-teaching, are utilized in the development of partially-connected RNN models. These two techniques are employed to reduce the over-fitting in ANNs when noisy data is used in the training process and, hence, to improve the open-loop accuracy as well as the closed-loop performance under a Lyapunov-based model predictive controller (LMPC). Aspen Plus Dynamics, a well-known high-fidelity process simulator, is used to simulate a large-scale chemical process application in order to demonstrate the anticipated improvements in both open-loop approximation and closed-loop controller performance in the presence of Gaussian and non-Gaussian noise in the data set using physics-informed RNNs.

From chapters 3 and 4, alongside with different works in literature, it has been demonstrated that physics-informed RNN models (where the network structure is informed by the physical interactions among process variables) are preferable to dense RNN models. Motivated by this, chapter 5 focuses on developing a theoretical upper bound of the generalization error of partially-connected RNN models and its relationship to the corresponding error of fully-connected RNN models for the same training and testing data sets. The RNN models are subsequently used in the model predictive control of nonlinear processes. Through the use of a chemical process example, the advantages of the use of partially connected RNN models in MPC are then illustrated via open-loop and closed-loop simulations.

Chapter 6 summarizes the main results of the dissertation.

Chapter 2

Machine Learning-Based State Estimation and Predictive Control of Nonlinear Processes

2.1 Introduction

Closed-loop performance of chemical processes under model-based controllers (e.g., model predictive control (MPC)) depends on the model representation of the process, and the availability of real-time state measurements. In general, MPC uses a first-principles model or a data-driven process model to predict state evolution in the optimization problem, and adjusts its control actions with state feedback from the sensor measurements. However, measurements of key process states such as species concentration in a chemical reactor could be time-consuming and sometimes involves manual manipulation of samples during offline protocols [66, 126]. Additionally, the cost of equipment for getting the targeted measurement in real time also hinders its real-time application in chemical plants [76]. One way to address this issue is to combine measurable process state variables (e.g., pressure, level, and temperature measurements) with state estimation techniques to predict unmeasured states in real-time operation.

State estimation has been extensively studied in the literature, and includes methods for both deterministic and stochastic cases (e.g., [4, 28, 76, 78]). In stochastic state estimation, many methodologies have been proposed including recursive and optimization-based approaches, which can also address the constrained and unconstrained estimation problems. Extended Kalman filter (EKF) is one of the most popular recursive methods for unconstrained nonlinear systems. Moving horizon estimator is an optimization-based methodology that can account for constraints in its formulation. Additionally, other methodologies such as unscented Kalman filter, particle filter, constrained version of the EKF, and combination of the above methods, have been proposed to improve the performance of EKF (e.g., [4, 60, 76]). In deterministic state estimation, Luenberger-based observers are common estimation methods for the practitioners [9, 28]. Additionally, extended Luenberger observer, sliding mode observer, adaptive state observer, high-gain observer, geometric observer, backstepping observer have found diverse applications in many fields (e.g., [9]). Similarities and differences among the above methods and their advantages and disadvantages are further discussed in [9, 78]. In order to achieve a desired performance using these methodologies, a mathematical model for the targeted system is generally needed to describe process dynamics in a certain operating region. However, the development of such a process model for some complex reacting systems using first-principles knowledge could be challenging. For example for a catalytic carbon monoxide oxidation over Pt-alumina, a common Langmuir-Hinshelwood rate law is only valid in a small region of operation [77].

Machine learning has recently attracted an increasing level of attention in process modeling. Among many different machine learning methods, recurrent neural networks (RNN) and Long-Short-Term-Memory (LSTM) networks have been utilized to model dynamic systems due to their temporal dynamic behavior. Additionally, hybrid modeling that relies on both first-principles knowledge and process operational data can also be used to model nonlinear chemical processes and is one of the most interesting and challenging problems in the data science era [100]. The idea of hybrid modeling is to use the best features of first-principles model (i.e., parametric models) and of data-driven models (i.e., non-parametric models) to better capture the process dynamics. There

are many examples of hybrid modeling and their applications to chemical engineering problems in the literature (e.g., [13,58,71,77,103,128]). For example, in [13], a hybrid model was developed for a hydraulic fracturing process, where the process first-principles model was integrated with a deep neural network. Due to advance in computational science like in quantum computing, it is expected the applications of infeasible challenging problem from the past by means of hybrid modelling and consequently the need of more complex algorithms [128]. Recently, hybrid modelling has been also pointed out as one of the most interesting challenging problem in this data science era [100]. For a better performance of these methodologies, the mathematical model that represents the targeted system should be able to cover all the possible phenomena over the entire operating conditions. However, this could be a difficult task in some complex reacting systems by only using a first-principle model that it is tractable in online conditions. However, this can increased considerable the time needed for implementing the proposed techniques, specially when using first-principles models. To work around this issue due to the recent advances, machine learning modeling can be used to unify the modeling approach for state estimation and control theory in an efficient way. In [77], a neural network model was developed to represent the reaction kinetics and was coupled with the first-principles model to obtain a hybrid model that was successfully applied within the EKF. It is reported that hybrid models can not only augment the region of operation, but also provide a more general modeling framework that can build models faster and need no process insights [107]. Furthermore, hybrid artificial neural network has been also considered in control strategies [103, 122]. For example, MPC was implemented in a batch polymerization process using an hybrid artificial neural network (ANN) [98]. In this work, experimental studies showed the benefits of using hybrid ANN in the control loop over other methodologies [98]. Similarly, hybrid stacked recurrent neural network(RNN) model showed a better performance during gel effect prediction and also during the control studies in a batch polymerization process when comparing with a single RNN model [95]. Hybrid modelling using neural networks and first principle models in a batch polymerization system showed better set-point tracking over other two control design [106].

Machine learning models can be utilized in model-based controllers to predict future states. Recently, in [114, 118], machine-learning-based MPC schemes have been proposed to optimize process performance and ensure system stability with feedback measurements of process state variables assumed to be available. However, the assumption of availability of full state measurements for feedback control may not hold for the chemical processes with state variables difficult to measure in real time. In this work, we propose two machine learning approaches: a) recurrent neural networks, and b) hybrid models using feed-forward neural networks and first-principles models, to model nonlinear processes. Then, we integrate the RNN model and the hybrid model within the extended Luenberger observer framework and develop Lyapunov-based MPC using state estimates from machine-learning-based state estimators. Specifically, section 2.2 introduces the preliminaries, including the class of systems, and the formulation of extended Luenberger observer. Section 2.3 presents the formulation of RNN models and of the RNN-based Luenberger observe along with the formulation of hybrid models and of the hybrid-model-based state estimator. An output feedback model predictive controller that uses state estimates from the aforementioned machine-learning-based state estimators formulation is also introduced in section 2.3, and then it is applied to a chemical reactor example to illustrate the effectiveness of the proposed estimation approaches.

2.2 Preliminaries

2.2.1 Notations

The Euclidean norm of a vector is represented by $|\cdot|$. The standard Lie derivative is represented as $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$. The notation $\setminus$ stands for set subtraction, i.e., $A \setminus B = \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. The function $f(\cdot)$ is said to be of class C^1 if it is continuously differentiable.

2.2.2 Class of Systems

We consider the following class of continuous-time nonlinear systems in state-space form:

$$\dot{x} = F(x,u) := f(x) + g(x)u \tag{2.1a}$$

$$y = h(x) \tag{2.1b}$$

where the state vector is $x = [x_1, ..., x_n]^T \in \mathbf{R}^n$, the output vector is $y = [y_1, ..., y_q]^T \in \mathbf{R}^q$, and the input vector is $u = [u_1, ..., u_m]^T \in \mathbf{R}^m$. $F(x,u)$ is a nonlinear function with respect to x and u. The constraints on control inputs is given by u $\in U := \{{u_i}^{min} \leq u_i \leq {u_i}^{max}\}$. The function $f(\cdot)$, $g(\cdot)$ and $h(\cdot)$ are matrices of dimension $n \times 1$, $n \times m$, and $q \times 1$, respectively.

2.2.3 Extended Luenberger Observer

Extended Luenberger observer (ELO) has been proposed for nonlinear processes as a natural extension of Luenberger observer based on a linear approximation of the process [28, 127]. The practical goal of the state observer is to provide an estimation of the unmeasured internal states of a given system by utilizing measured states from the process along with the implemented inputs. The extended Luenberger observer is presented in the following form for the nonlinear system of Eq. 2.1.

$$\dot{\hat{x}} = F(\hat{x},u) + K(y - h(\hat{x})) \tag{2.2}$$

where $\hat{x}$ represents the estimated state vector, and the observer gain is denoted by K. The observer gain is also associated with desired properties from the state estimator and will be discussed in detail later. It is observed from Eq. 2.2 that the first term is the process model, and the last term $K(y - h(\hat{x}))$ is known as the output prediction error, which is also considered as a correction term.

The goal of the ELO is to minimize the estimation error (i.e., $e = x - \hat{x}$) in which the dynamic

of the error is determined by the following equation [28, 67]:

$$\dot{e} = F(\hat{x}+e,u) - F(\hat{x},u) \ - K(h(\hat{x}+e) - h(\hat{x})) \tag{2.3}$$

As shown in Eq. 2.3, the problem now is to determine under which conditions e can decay to zero. Therefore, it is important to design K to achieve this goal. In order to design K, Eq. 2.3 can be simplified to the following equation by linearizing the process model at a fixed point:

$$\dot{e} = (A - KL)e \tag{2.4}$$

where $A = [\partial F(x,u)/\partial x]_{x=\hat{x}}$ and $L = [\partial h(x,u)/\partial x]_{x=\hat{x}}$ are the linearized terms of the nonlinear system evaluated at a fixed-point (typically the operating steady-state). Finally, K is selected such that the eigenvalues of the matrix $A - KL$ have strictly negative real parts.

2.2.4 Stabilization via Control Lyapunov Function

We assume that there exists an observer and a feedback control law $u = \Phi(\hat{x}) \in U$ using the estimated states $\hat{x}$ from the observer to form an output feedback controller that can render the origin of the nonlinear system of Eq. 2.1 exponentially stable. This stabilizability assumption implies that there exists a C^1 Control Lyapunov function $V(x)$ such that the following inequalities hold for all $x, \hat{x}$ in an open neighborhood D around the origin:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{2.5a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(\hat{x})) \leq -c_3|x|^2, \tag{2.5b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{2.5c}$$

where c_1, c_2, c_3 and c_4 are positive constants. $F(x,u)$ is the nonlinear system of Eq. 2.1. A candidate controller for $\Phi(\hat{x})$ is provided by the universal Sontag control law [61]. Then,

following [118], we characterize the closed-loop stability region $\Omega_{\rho'}$ as a level set of Lyapunov function in the region D where the time-derivative of V is rendered negative under the controller $\Phi(\hat{x}) \in U$, i.e., $\Omega_{\rho'} := \{x \in D \mid V(x) \leq \rho'\}$, where $\rho' > 0$. Additionally, based on the Lipschitz property of $F(x,u)$ and the boundedness of u, there exist positive constants M, L_x, L_x' such that the following inequalities hold for all $x, x' \in D$ and $u \in U$:

$$|F(x,u)| \leq M \tag{2.6a}$$

$$|F(x,u) - F(x',u)| \leq L_x|x - x'| \tag{2.6b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x,u) - \frac{\partial V(x')}{\partial x} F(x',u) \right| \leq L_x'|x - x'| \tag{2.6c}$$

Remark 2.1. *The assumption of the existence of an output feedback controller satisfying Eq. 2.5 requires that the observer states are bounded in $\Omega_{\rho'}$ and the estimate error, $e = x - \hat{x}$, converges to zero within finite time. In this work, the ELO of Eq. 2.2 and the Sontag control law [90] are used as the observer and state feedback controller, respectively.*

2.3 Machine Learning Based State Estimation

In this Chapter, the theory and design of two ML-based state observers is discussed and constructed in the framework of Luenberger observer. The necessary stability assumptions were made to prove the stability of the closed-loop system under output feedback ML-based MPC.

2.3.1 RNN-based State Estimator

2.3.1.1 Recurrent Neural Network (RNN)

As a process model is needed in the extended Luenberger observer of Eq. 2.2. The following RNN model is developed to approximate the nonlinear system of Eq. 2.1 using process operational

data when a first-principles model is not available:

$$\dot{\bar{x}} = F_{rnn}(\bar{x}, u) := A\bar{x} + \Theta^T y \tag{2.7}$$

where $\bar{x} = [\bar{x}_1, ..., \bar{x}_n]$ is the RNN state vector, and $u = [u_1, ..., u_m]$ is the manipulated input vector. $y = [y_1, ..., y_n, y_{n+1}, ..., y_{m+n}] = [\sigma(\bar{x}_1), ..., \sigma(\bar{x}_n), u_1, ..., u_m] \in \mathbf{R}^{n+m}$ is a vector of both $\bar{x}$ and u, where $\sigma(\cdot)$ is the nonlinear activation function. A is a diagonal coefficient matrix and $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(m+n)\times n}$ is a matrix with neural network weights to be optimized. The structures of unfolded and folded RNNs are shown in Fig. 3.1. The coefficient matrices $A = diag[-a_1, ..., -a_n]$, $a_i > 0,\ i = 1, ..., n$ and Θ consist of neural network weights that will be optimized during training. In practical implementation, the neural network weights are optimized to minimize the error between predicted outputs and the actual outputs in the training dataset.

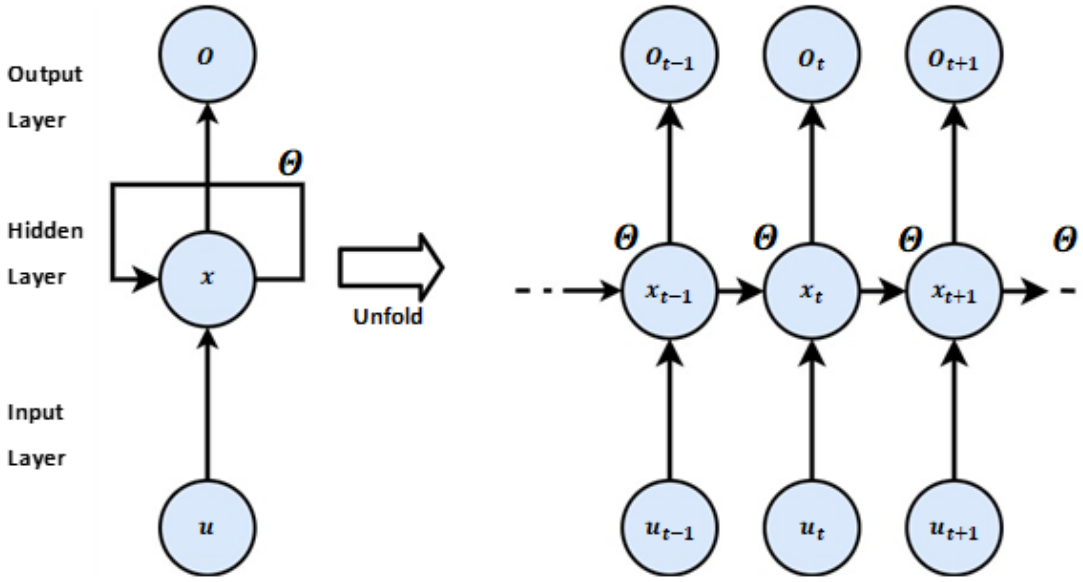

Figure 2.1: Structure of recurrent neural network.

After designing the RNN structure in terms of the number of layers and neurons and other hyper-parameters, the RNN is trained following the standard learning process as discussed in [118, 119]. Specifically, training, validation and testing datasets are generated from open-loop

simulations of Eq. 2.1 under different initial conditions and control actions for a finite period of time. The continuous-time system of Eq. 2.1 is integrated using explicit Euler method with a sufficiently small integration time step h_c, and the control actions u are applied in a sample-and-hold fashion, i.e., $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ and Δ is the sampling period. Since RNN models are able to capture process dynamic behavior from time-series data, the RNN model in this work is developed using all the integration time step data (i.e., data at each h_c step) within each sampling period to predict the state evolution for one sampling period. Additionally, as discussed in [118], the RNN models needs to satisfy a sufficiently small modeling error, i.e., $|\nu| = |F(x,u) - F_{rnn}(\bar{x},u)| \leq \gamma|x| \leq \nu_m$, between the RNN model and the nonlinear model of Eq. 2.1 during the training process such that it can well represent process dynamics in the operating region considered. Note that the modeling error ν is defined as the error between $\dot{x}$ from the first-principles model and the $\dot{x}$ predicted by the RNN model under the same input and states. The modeling error ν is not a constant for different states and input. However, since we limit the operation in the stability region Ω_ρ (i.e., both states and inputs are bounded), and the RNN model is trained using the datasets generated in Ω_ρ to achieve a very high accuracy, the modeling error can be made sufficiently small for all the states within Ω_ρ.

Remark 2.2. *The RNN models are chosen in this work due to the ability of modeling a general class of nonlinear dynamical systems through the feedback loop in the hidden layer that introduces the past information to the current network (similar to the evolution of dynamical systems where past states values influence current values). The proposed approach can be extended to other machine learning approaches such as long short-term memory networks which are also widely used in modeling nonlinear dynamical systems [21, 31].In addition to neural network modeling approaches, sparsity promoting algorithms, (extended) dynamic mode decompositions, and Koopman system identification have also been used to approximate nonlinear systems in recent works [55, 69].*

Remark 2.3. *The RNN model in this work is developed using noise-free data from extensive open-loop simulations of Eq. 2.1 to capture process dynamics in the operating region $\Omega_{\rho'}$. In*

addition to computer simulations, datasets can also be generated using industrial measurements and experimental data. In the case that real industrial measurements are corrupted by noise from sensors variability and common plant variance, co-teaching training algorithm and dropout technique can be utilized in machine learning modeling approaches to improve the approximation performance by reducing the impact of noise. The interested reader is referred to [113] for a detailed development of co-teaching and dropout methods.

Remark 2.4. *Overfitting is one of the most common issues in the development of neural network models [44, 113]. To reduce overfitting in this study, we start with a simple neural network with one layer and a few neurons, and keep increasing the number of layers and of neurons until no further improvement is noticed. We also use a large amount of data for validation such that the model performs well with respect to training and validation datasets.*

2.3.1.2 RNN-based State Estimator

The RNN model is then used in the extended Luenberger observer of Eq. 2.2 as follows:

$$\dot{\hat{x}} = F_{rnn}(\hat{x}, u) + K(y - h(\hat{x})) \tag{2.8}$$

Specifically, the state estimation based on the RNN model of Eq. 2.7 is obtained from the following steps. 1) Given an initial state estimate $\hat{x}(t_k)$ at time $t = t_k$ along with the manipulated input vector $u(t_k)$, the RNN model predicts the state at the next integration time step at $t = t_k + h_c$, then the state estimate at $t = t_k + h_c$ is obtained following Eq. 2.8 by adding the second term $h_c \times K(y - h(\hat{x}))$. 2) After the state estimate at $t = t_k + h_c$ is obtained, the above process is repeated with the same input u (because u remains constant within one sampling period). 3) Finally, the state estimate at the next sampling period $t = t_{k+1} := t_k + \Delta$ is obtained through $\frac{\Delta}{h_c}$ iterations of the above process.

Similarly, we assume that the RNN-based observer together with the state feedback control law $u = \Phi(\hat{x}) \in U$ form an output feedback controller that can render the origin of the RNN system of Eq. 2.7 exponentially stable. This implies that there exists a C^1 Lyapunov function $V(x)$ such that

the following inequalities hold for all $x, \hat{x}$ in an open neighborhood D around the origin:

$$\hat{c}_1|x|^2 \leq V(x) \leq \hat{c}_2|x|^2, \tag{2.9a}$$

$$\frac{\partial V(x)}{\partial x} F_{rnn}(x, \Phi(\hat{x})) \leq -\hat{c}_3|x|^2, \tag{2.9b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq \hat{c}_4|x| \tag{2.9c}$$

where $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$ and $\hat{c}_4$ are positive constants. Note that the control law in this section is designed based on the RNN model of Eq. 2.7, while the control law in Section 2.2.4 is designed based on the first-principles model of Eq. 2.1. Subsequently, a new closed-loop stability region Ω_ρ can be characterized within D, where Eq. 2.9 is satisfied under $u = \Phi(\hat{x}) \in U$. Since the RNN model is trained with a sufficiently small modeling error, the state estimation through RNN-based state estimator of Eq. 2.8 is sufficiently close to the estimated value provided by Eq. 2.2 when the process model of Eq. 2.1 is known. The following proposition demonstrates that the controller $u = \Phi(\hat{x}) \in U$ designed based on the estimated state from RNN-based estimator is able to stabilize the system of Eq. 2.1 if the modeling error $|\nu| = |F(x,u) - F_{rnn}(\bar{x},u)|$ is sufficiently small.

Proposition 2.1. *Consider the nonlinear system of Eq. 2.1 with an initial state $x_0 \in \Omega_\rho$ and a stabilizing control law $u = \Phi(\hat{x}) \in U$ based on the estimated states from Eq. 2.8, if the modeling error can be bounded, i.e., $|\nu| = |F(x,u) - F_{rnn}(x,u)| \leq \gamma|x|$, for all $x \in \Omega_\rho$ and $u \in U$, where γ is a positive real number satisfying $\gamma < \hat{c}_3/\hat{c}_4$, then the origin of the closed-loop system of Eq. 2.1 is rendered exponentially stable under $u = \Phi(\hat{x}) \in U$ for all $x, \hat{x} \in \Omega_\rho$.*

Proof: Based on Eq. 2.9, the time-derivative of V for the nonlinear system of Eq. 2.1 is derived as follows:

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

Therefore, $\dot{V} \leq -(\hat{c}_3 - \hat{c}_4\gamma)|x|^2 \leq 0$ holds if γ satisfies $\gamma < \hat{c}_3/\hat{c}_4$. This implies that for all $x_0 \in \Omega_\rho$, the origin of the nonlinear system of Eq. 2.1 is rendered exponentially stable under the controller $u = \Phi(\hat{x}) \in U$ with state estimates from RNN-based estimator.

2.3.2 Hybrid-Model-Based State Estimator

In this section, we introduce a state estimator designed based on a hybrid model that integrates feed-forward neural network (FNN) with first-principles model. In this case, the FNN model is only used to approximate the nonlinear terms in Eq. 2.1, while the first-principles model of Eq. 2.1 can be derived from physical laws such as mass and energy balances.

2.3.2.1 Feed-forward Neural Network (FNN)

We develop a feed-forward neural network $F_{NN}(x)$ with input vector $x = [x_1, ..., x_n]$ and output vector $y = F_{NN}(x) = [y_1, ..., y_m]$ to approximate the nonlinear terms in Eq. 2.1. Figure 2.2 shows the structure of a feed-forward neural network with three layers, i.e., input layer, hidden layer, and output layer. The hidden neurons h_j, $j = 1, ..., p$, and the outputs y_k are obtained using the following equations:

$$h_j = \sigma_1\left(\sum_{i=1}^{n} w_{ji}^{[1]}\, x_i + w_{j0}^{[1]}\right), \quad j = 1, 2, ..., p \tag{2.11}$$

$$y_k = \sigma_2\left(\sum_{i=1}^{p} w_{ki}^{[2]}\, h_i + w_{k0}^{[2]}\right), \quad k = 1, 2, ..., m \tag{2.12}$$

The weights in the first two layers are denoted by $w_{ji}^{[1]}$, and $w_{ki}^{[2]}$ respectively, with $w_{j0}^{[1]}$ and $w_{k0}^{[2]}$ representing biases. σ_1 and σ_2 are nonlinear activation functions such as hyperbolic tangent function, $tanh(x) = \frac{2}{(1+\, e^{-2x})} - 1$, and logistic sigmoid function $S(x) = \frac{1}{(1+\, e^{-x})}$. The activation function σ_1 is utilized with a linear combination of input variables x_i in the calculation of hidden neurons h_j, while the output variable y_k is calculated through σ_2 with a linear combination of hidden neurons.

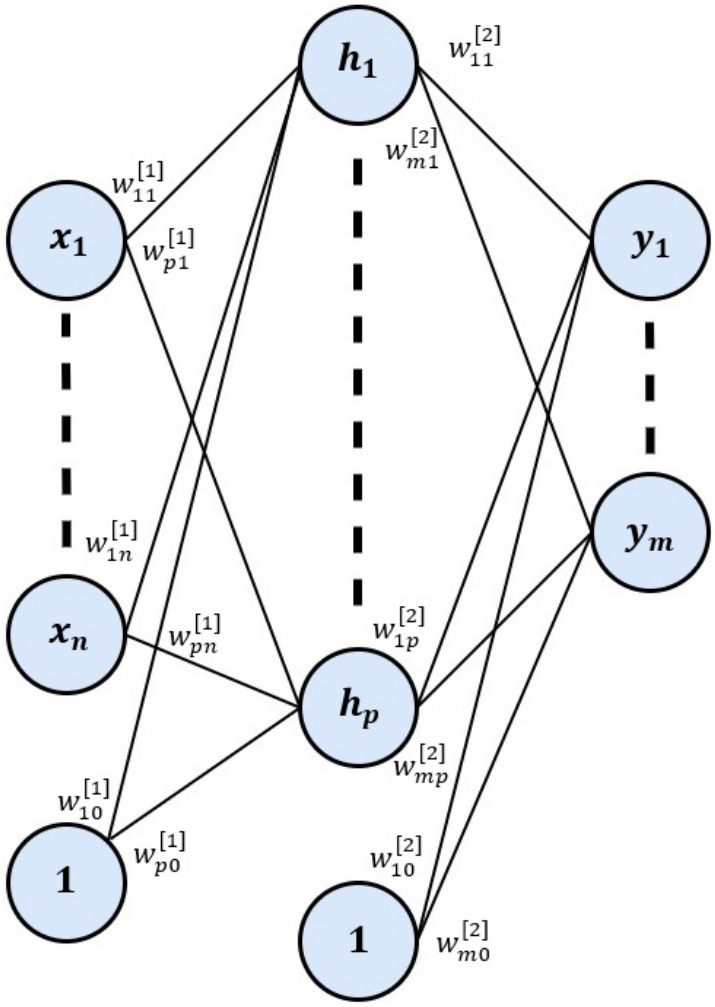

Figure 2.2: Three-layer feed-forward neural network structure with biases represented by neuron '1'.

The development of FNN models also requires datasets for training and validating. Hence, a set of input vectors $\{x^n\}$ with the corresponding output vectors $\{\hat{y}^n\}$ are used in constructing datasets. The data can be collected in a variety of ways, including but not limited to experimental data and extensive computer simulations. The FNN model is trained to minimize the following loss function:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} |y(x^n, w) - \hat{y}^n|^2 \tag{2.13}$$

where N is the number of data points in training. The loss function is the sum of squared error between the predicted output by FNN model and the actual output in datasets. The weight vectors

w are obtained by minimizing Eq.2.13 via the gradient descent optimization method $w^{\iota+1} = w^{\iota} - \eta \nabla E(w^{\iota})$, where the iteration is denoted by ι, and $\eta > 0$ is the learning rate.

2.3.2.2 Hybrid-model-based Estimator

The hybrid model is developed by integrating the FNN model with the first-principles model of Eq. 2.1. The FNN model is used to represent the nonlinear terms, while the first-principles model is developed from the physical knowledge [114]. The hybrid model for the system of Eq.2.1 is in the following form:

$$\dot{x} = F_h(x, F_{NN}(x), u) \tag{2.14}$$

where F_h denotes the hybrid model and F_{NN} is the FNN model used to capture the static nonlinear relationship between the inputs and outputs of the nonlinear terms. The hybrid-model-based state estimator utilizes the hybrid model of Eq.2.14 as the process model and includes the estimation correction term as follows:

$$\dot{\hat{x}} = F_h(\hat{x}, F_{NN}(\hat{x}), u) + K(y - h(\hat{x})) \tag{2.15}$$

F_h represents the hybrid model that is developed to capture the dynamics of the whole nonlinear system of Eq. 2.1. It should be noted that the hybrid model is used when a full first-principles model is unavailable. In that case, some of the nonlinearities like reaction rates are unknown and can be approximated by feedforward neural networks from data. Similar to the RNN-based estimator of Eq. 2.8, the hybrid-model-based state observer takes the state estimate $\hat{x}(t_{k-1})$ and the manipulated variable $u(t_{k-1})$ along with the last measurement $y(t_k)$ as inputs, and integrates Eq. 2.15 for one sampling period to estimate the states at the next sampling time. The explicit Euler method is used to integrate Eq. 2.15 with a small integration step h_c. Both estimators are initialized with initial conditions within Ω_ρ and, if they calculate an estimate outside of Ω_ρ, then this estimate is discarded and is replaced with an estimate within Ω_ρ which ensures via the conditions of Eq. 2.5 that closed-loop stability is maintained.

2.4 Output Feedback Model Predictive Control

In this section, an output feedback model predictive control (MPC) is designed based on state estimates provided by the RNN-based estimator to stabilize the nonlinear system of Eq. 2.1 at the steady-state. Specifically, the Lyapunov-based MPC is used in this work and the formulation is presented as the following optimization problem:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) dt \tag{2.16a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{rnn}(\tilde{x}(t), u(t)) \tag{2.16b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{2.16c}$$

$$\tilde{x}(t_k) = \hat{x}(t_k) \tag{2.16d}$$

$$\dot{V}(\hat{x}(t_k), u) \leq \dot{V}(\hat{x}(t_k), \Phi(\hat{x}(t_k))), \quad \text{if } \hat{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}} \tag{2.16e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } \hat{x}(t_k) \in \Omega_{\rho_{nn}} \tag{2.16f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. $\dot{V}(x,u)$ represents the time-derivative of V, i.e., $\frac{\partial V(x)}{\partial x}(F_{rnn}(x,u))$. The LMPC calculates the optimal input sequence $u^*(t)$ over the prediction horizon $t \in [t_k, t_{k+N})$, and sends the first control action $u^*(t_k)$ to the system to be applied for the next sampling period. Then the LMPC receives new measurements and is resolved with new state estimates at the next sampling time.

In the optimization problem of Eq. 2.16, Eq. 2.16a is the objective function of LMPC that minimizes the time-integral of $L(\tilde{x}(t), u(t))$ over the prediction horizon subject to the following constraints. The constraint of Eq. 2.16b is the RNN model of Eq. 2.7 for predicting state evolution given control actions and an initial state. Eq. 2.16c is the input constraint. Eq. 2.16d defines the initial condition $\tilde{x}(t_k)$ of Eq. 2.16b, which is the state estimates provided by the RNN-based state estimator of Eq. 2.8 at $t = t_k$. Specifically, given the state estimates at the previous time step, and

the control actions, the estimation for the current state at $t = t_k$ is obtained following the steps as discussed in Section 2.3.1.2. Then, the state estimates $\hat{x}(t_k)$ is used as the initial state for the prediction model of Eq. 2.16b, and also in the constraints of Eq. 2.16e. If $\hat{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}}$, the constraint of Eq. 2.16e is activated, under which the state is forced to move towards the origin since $\Phi(\hat{x})$ is a stabilizing feedback control law. If the estimated state $\hat{x}(t_k)$ enters a small neighborhood around the origin, $\Omega_{\rho_{nn}}$, then the constraint of Eq. 2.16f requires the states to remain inside $\Omega_{\rho_{nn}}$ for the entire prediction horizon.

The following theorem is established to demonstrate guaranteed closed-loop stability for the nonlinear system of Eq. 2.1 under the LMPC of Eq. 2.16 using state estimates from RNN-based estimator.

Theorem 2.1. *Consider the closed-loop system of Eq. 2.1 with an initial state $x_0 \in \Omega_\rho$ under the LMPC of Eq. 2.16. Let $\Delta > 0$, $\varepsilon_w > 0$ and $\rho > \rho_{min} > \rho_{nn} > \rho_s$ satisfy*

$$-\frac{\hat{c}_3 - \hat{c}_4\gamma}{\hat{c}_2}\rho_s + L_x' M\Delta \leq -\varepsilon_w \tag{2.17a}$$

$$\rho_{nn} = \max\{V(\bar{x}(t_k+\Delta)) \mid \hat{x}(t_k) \in \Omega_{\rho_s},\ u \in U\} \tag{2.17b}$$

$$\rho_{min} = \max\{V(x(t_k)) \mid \bar{x}(t_k) \in \Omega_{\rho_{nn}}\}. \tag{2.17c}$$

Then, for any initial state $x_0 \in \Omega_\rho$, it is guaranteed that the state is bounded in the stability region for all times, i.e., $x(t) \in \Omega_\rho$, $\forall t \geq 0$, and $x(t)$ ultimately converges to $\Omega_{\rho_{min}}$ for the closed-loop system of Eq. 2.1 under the LMPC of Eq. 2.16.

Proof: We first consider the estimated state $\hat{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}}$ at $t = t_k$. In this case, the LMPC uses the constraint of Eq. 2.16e to render the time-derivative of V under u less than that under the stabilizing controller $u = \Phi(\hat{x})$. We show that under the constraint of Eq. 2.16e, the state is able to move towards the origin over the next sampling period. Specifically, we derive the time-derivative

of $V(x)$ under $u = \Phi(\hat{x})$ for the nonlinear system of Eq. 2.1 as follows:

$$\begin{aligned}\dot{V}(x(t)) =& \frac{\partial V(x(t))}{\partial x} F(x(t), \Phi(\hat{x}(t_k))) \\ =& \frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(\hat{x}(t_k))) \\ &+ \frac{\partial V(x(t))}{\partial x} F(x(t), \Phi(\hat{x}(t_k))) \\ &- \frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(\hat{x}(t_k)))\end{aligned} \tag{2.18}$$

Since Eq. 2.10 shows that the controller $u = \Phi(\hat{x})$ can render the time-derivative of $V(x(t_k))$ negative, i.e., $\frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(x(t_k))) \leq -\hat{c}_3 |x(t_k)|^2 + \hat{c}_4 \gamma |x(t_k)|^2 < 0, \forall x(t_k) \neq 0$, we can further derive the following inequality for $t \in [t_k, t_{k+1})$:

$$\begin{aligned}\dot{V}(x(t)) \leq& -\frac{\hat{c}_3 - \hat{c}_4 \gamma}{\hat{c}_2} \rho_s + \frac{\partial V(x(t))}{\partial x} F(x(t), \Phi(\hat{x}(t_k))) \\ &- \frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(\hat{x}(t_k))) \\ \leq& -\frac{\hat{c}_3 - \hat{c}_4 \gamma}{\hat{c}_2} \rho_s + L_x' |x(t) - x(t_k)| \\ \leq& -\frac{\hat{c}_3 - \hat{c}_4 \gamma}{\hat{c}_2} \rho_s + L_x' M \Delta\end{aligned} \tag{2.19}$$

Therefore, if Eq. 2.17a is satisfied, the time-derivative of $V(x)$ under $u = \Phi(\hat{x})$ for the nonlinear system of Eq. 2.1 is rendered negative for the next sampling time, which implies the state of the system of Eq. 2.1 will move towards the origin under the constraint of Eq. 2.16e. It should be noted that the state estimate $\hat{x}$ is assumed to be bounded in Ω_ρ. If the estimate is outside of Ω_ρ, we discard it and replace with an estimate inside the Ω_ρ. This new estimate could be a state inside Ω_ρ that is closest to the original estimate. Since the state estimate is bounded in Ω_ρ for all times, and closed-loop stability is ensured under the output feedback controller assumed in Section 2.3.1.2 (i.e., the state feedback control law $u = \Phi(\hat{x})$ designed based on the estimated states from RNN observer), the true state can be driven into a small neighborhood around the origin within finite sampling periods. If the estimated state enters $\Omega_{\rho_{nn}}$, the LMPC activates the constraint of Eq. 2.16f to maintain the predicted states of the RNN model within $\Omega_{\rho_{nn}}$ over the

prediction horizon. However, since there exists a model mismatch between the nonlinear system of Eq. 2.1 and the prediction model of Eq. 2.16b (i.e., the RNN model), we need to show that the actual state of the nonlinear system of Eq. 2.1 is bounded in a small neighborhood around the origin under LMPC. To that end, we characterize the set $\Omega_{\rho_{min}}$ of Eq. 2.17c to account for the sufficiently small modeling error between the RNN model and the nonlinear system of Eq. 2.1. Eq. 2.17c shows that if the RNN predicted state $\bar{x}(t_k)$ is inside $\Omega_{\rho_{nn}}$, then the actual state of the nonlinear system of Eq. 2.1 is bounded in $\Omega_{\rho_{min}}$. This completes the proof of closed-loop stability for the system of Eq. 2.1 under LMPC.

Remark 2.5. *The formulation of the LMPC using hybrid-model-based estimator is very similar to that of Eq. 2.16 using RNN-based estimator, and therefore, is omitted here. The only difference in the LMPC formulation would be the prediction model of Eq. 2.16b, for which the hybrid model of Eq. 2.14 will be used to replace the RNN model. Additionally, the hybrid model will also be used to provide state estimates at each sampling time in LMPC. The closed-loop stability analysis for hybrid-model-based estimator is also similar to Theorem 2.1 based on the fact that a well-conditioned feed-forward neural network is obtained to represent the nonlinear terms with a sufficiently high accuracy.*

2.5 Application to a Chemical Reactor Example

In this section, a nonlinear chemical process is used to illustrate the application of the proposed RNN-based and hybrid-model-based estimators in the LMPC controller. A non-isothermal, a well mixed continuous stirred tank reactor (CSTR) is considered, with the following reversible first-order exothermic reaction [129]:

$$A \leftrightarrow B$$

The nonlinear dynamical model that describes the process dynamics is given by the following mass and energy balance equations:

$$\frac{dC_A}{dt} = \frac{1}{\tau}(C_{A0} - C_A) - r_A + r_B \tag{2.20a}$$

$$\frac{dC_B}{dt} = \frac{-1}{\tau}C_B + r_A - r_B \tag{2.20b}$$

$$\frac{dT}{dt} = \frac{1}{\tau}(T_0 - T) + \frac{-\Delta H}{\rho C_p}(r_A - r_B) + \frac{Q}{\rho C_p V} \tag{2.20c}$$

$$r_A = k_A e^{\frac{-E_A}{RT}} C_A \tag{2.20d}$$

$$r_B = k_B e^{\frac{-E_B}{RT}} C_B \tag{2.20e}$$

The concentration of A and B in the CSTR are given by C_A and C_B respectively, and T represents the reactor temperature. The feed concentration is denoted by C_{A0} and the feed temperature is denoted by T_0. As for the reaction kinetics, k_A and E_A represent the pre-exponential constant and the activation energy for the forward reaction, while k_B and E_B are for the reverse reaction. The reactor residence time is denoted by τ. V represents the reactor volume, ΔH is the reaction enthalpy, and the heat capacity of the mixture liquid is denoted by C_p. The CSTR is equipped with a heating/cooling jacket to provide/remove required heat at rate Q to/from the reactor. [129] has provided the optimal steady-states for the process described in Eq. 2.20. The optimal steady-state values and process parameter values are listed in Table 2.1.

Table 2.1: Parameter and steady-state values for the CSTR.

$T_o = 400\ K$	$T_s = 426.743\ K$
$k_A = 5000\ /s$	$E_A = 1 \times 10^4\ cal/mol$
$k_B = 10^6\ /s$	$E_B = 1.5 \times 10^4\ cal/mol$
$R = 1.987\ cal/(mol\ K)$	$\Delta H = -5000\ cal/mol$
$\rho = 1\ kg/L$	$C_p = 1000\ cal/(kg\ K)$
$C_{A0} = 1\ mol/L$	$V = 100\ L$
$C_{A_s} = 0.4977\ mol/L$	$\tau = 60\ s$
$C_{B_s} = 0.5023\ mol/L$	$Q_s = 40386\ cal/s$

2.5.1 Simulation Settings

The control objective is to drive C_A, C_B, and T to the steady-state by manipulating the heat input rate Q. The manipulated variable is considered in the deviation form as $u = Q - Q_s$. The control action u is bounded with upper bound $u^{UB} = 40,000\ cal/s$ and a lower bound $u^{LB} = -40,000\ cal/s$. The process states are all represented in the deviation form. The optimal steady-state is at $x^T = [x_1\ x_2\ x_3] = [C_A - C_{A_s}\ C_B - C_{B_s}\ T - T_s]$ such that the origin is the equilibrium point of this system. Since in practice not all process states are measurable [54], unmeasurable states needs to be estimated based on measurable states. In this case study, we assume that the only measured state is $x_3 = T - T_s$. Therefore, $x_1 = C_A - C_{A_s}$ and $x_2 = C_B - C_{B_s}$ can be estimated using the proposed machine-learning-based state estimators. Based on the measurement y of the state variable x_3, the machine-learning-based observer first utilizes the RNN model (or hybrid model) to predict x_1 and x_2, and then add the estimation error part ($K(y - \hat{x}_3)$) to obtain the state estimates at the current time step. Subsequently, the estimated states $\hat{x}^T = [\hat{x}_1\ \hat{x}_2\ \hat{x}_3]$ are sent to the MPC for solving the optimal control action for the next sampling period. Additionally, the nonlinear system of Eq. 2.20 is observable for the given output (i.e., temperature T) in the sense there exists an estimator-based output feedback controller that exponentially stabilizes the closed-loop system.

The nonlinear optimization problem of LMPC is solved using the IPOPT software package ([104]), and its python version, PyIpopt, with the sampling period $\Delta = 10s$. The objective function of LMPC is of the form: $L(x,u) = x^T Q x + u^T R u$, where $Q = diag[5 \times 10^4\ \ 5 \times 10^4\ \ 1]$, and $R = [10^{-7}]$ are penalty matrices that should be tuned properly to achieve a better MPC performance ([6]). The observer gains used in this work are $K = [0.0005, 0.0005, 0.5]$. The Lyapunov functions is given by $V(x) = x^T\ P\ x$, with the following positive definite P matrix:

$$P = \begin{bmatrix} 625 & 0 & 0 \\ 0 & 625 & 0 \\ 100 & 100 & 10^5 \end{bmatrix}$$

2.5.2 Neural Networks Model Training

The data generation, neural network training and validation process for the RNN model are carried out as follows. To generate the dataset for RNN model, the system of Eq. 2.20 was numerically integrated for one sampling period under different initial conditions. The explicit Euler method with an integration time step of $h_c = 0.5\ s$ is utilized. Specifically, a data set of size 1.6×10^6 was built using MATLAB. The data base was then divided into an input matrix with u, x_1, x_2, x_3 at $t = t_k$ and an output matrix with x_1, x_2, and x_3 as outputs at $t = t_{k+1}$, from which 70% of the data was utilized for model training, and 30% was for validation. Note that the full state measurements are available in the training stage as the data can be obtained offline, while in real-time operation of CSTR, only the temperature can be measured every sampling time. The RNN model was developed using Keras library with two hidden layers of 50 unit in each layer and *tanh* activation function, and an output layer with 3 neurons and linear activation function. 274 epochs were used for the training process.

Similarly, the data was generated for feed-forward neural networks using MATLAB simulations of different values of x_1, x_2, and x_3 in the reaction kinetics model of Eq. 2.20. A dataset consisting of 1.25×10^5 data points was generated with x_1, x_2, and x_3 as inputs, and reaction rate as the output. Keras library was used for the FNN model training with three inputs and one output (i.e., the output is $r_A - r_B$). Three layers were used with 6 neurons, 12 neurons, and 1 neuron in each layer, respectively. *Relu* activation function was utilized in the first two layers and sigmoid activation function was used for the last layer.

Remark 2.6. *The RNN is developed to approximate the process model using all the states including temperature and species concentrations. The RNN outputs are the process states in the first-principles model. To build the state estimator, the RNN model is used to replace the process model $F(\hat{x}, u)$ in the extended Luenberger observer of Eq. 2.2. The training dataset is generated from simulations of the first-principles model, and the RNN model is trained with a sufficiently high accuracy, which guarantees that the RNN model predictions are sufficiently close to the estimates of the first-principles model. As the RNN model is developed to approximate the process model,*

the datasets include all the process states.

Remark 2.7. *Hybrid modeling approaches require a careful selection of process parameter/variable in first-principle models which will be estimated through data based approaches. Local/global sensitivity analysis is one of the most common methods for selecting such parameters during model identification for hybrid representations ([41, 65, 103]). In this example, the nonlinear terms in a nonlinear process (i.e., the reaction rates in CSTR first-principles equations) are chosen to be represented by neural networks to better capture the nonlinearities in a wide operating region.*

2.5.3 Closed-loop Simulation Results

Closed-loop simulation study is carried out to demonstrate the performance of the two proposed estimation approaches in the CSTR of Eq. 2.20. The closed-loop simulations results using the RNN-based estimator with four different sets of initial conditions, IC_1, IC_2, IC_3, and IC_4 are shown in Figs. 2.3-2.6, and the closed-loop simulation results using hybrid-model-based estimator with the same four initial conditions are shown in Figs. 2.7-2.10.

It can be seen from Figs. 2.3-2.6 that starting from different initial conditions and different initial estimates, the closed-loop states are stabilized at the steady-state under LMPC using RNN-based state estimator. Specifically, in Fig. 2.3 and Fig. 2.4, we consider two initial estimates that are very close to the true state values, where the true states are obtained from the first-principles model of Eq. 2.20. It is demonstrated that the state estimates provided by the RNN-based estimator converge to the true state value quickly, and after that, the closed-loop states are driven the the steady-state smoothly. In Fig. 2.5 and Fig. 2.6, we consider two initial estimates that are not close to the true state values at the beginning. It is demonstrated that the state estimates still converge to the true states but takes longer time than those in Fig. 2.3 and Fig. 2.4. In all cases, closed-loop stability is achieved for the system under LMPC.

Subsequently, the mean squared errors (MSE) between true state profiles and estimate state

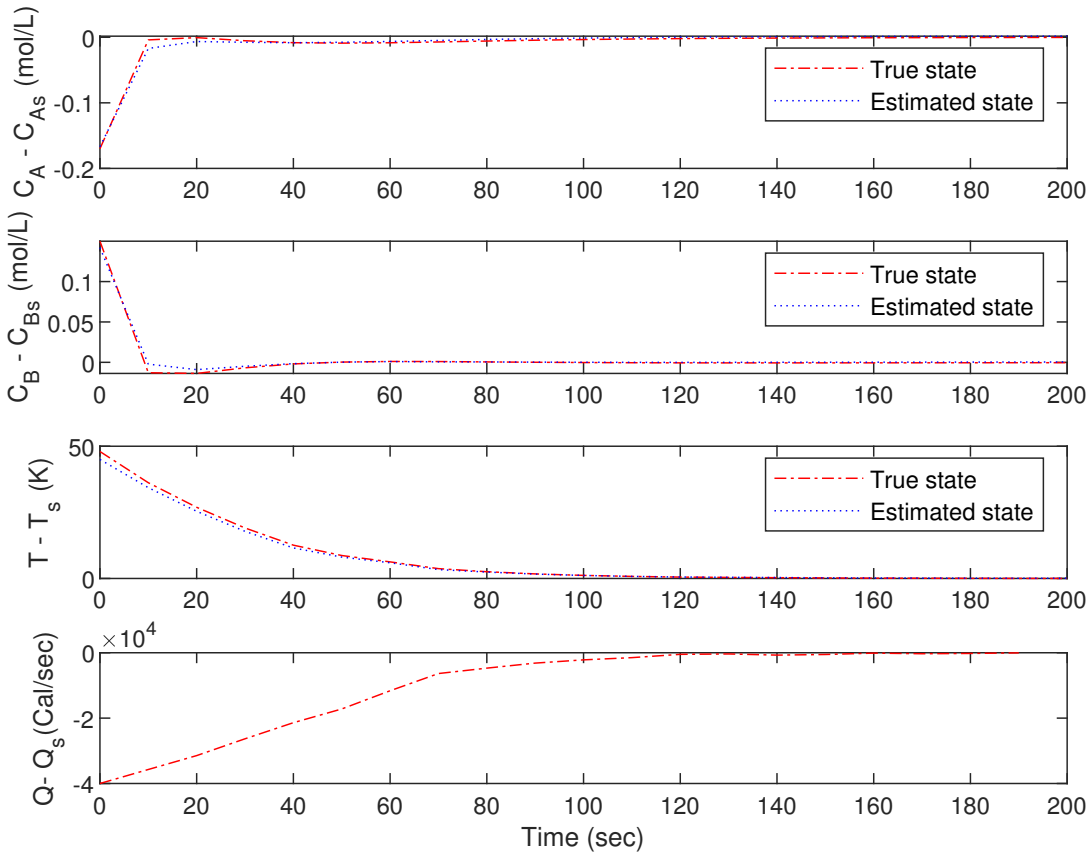

Figure 2.3: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using RNN-based estimator with the initial condition IC_1 (top three plots). The bottom plot displays the manipulated input profile.

profiles are used to evaluate the performance of the estimator. Table 2.2 summarized the MSE of state estimation using the RNN-based state estimator in the four closed-loop simulations. It is shown that all the closed-loop simulations achieve sufficiently small MSEs, and the simulations with IC_1 and IC_2 achieve better results due to better initial estimates. This is consistent with the closed-loop simulation results as shown in Figs. 2.3-2.6. Therefore, from this simulation study of CSTR example, it is demonstrated that the RNN-based estimator can estimate true state values with a sufficiently high accuracy.

The closed-loop simulation results using hybrid-model-based estimator with the same four initial conditions are shown in Figs. 2.7-2.10, and the MSE results are also summarized in

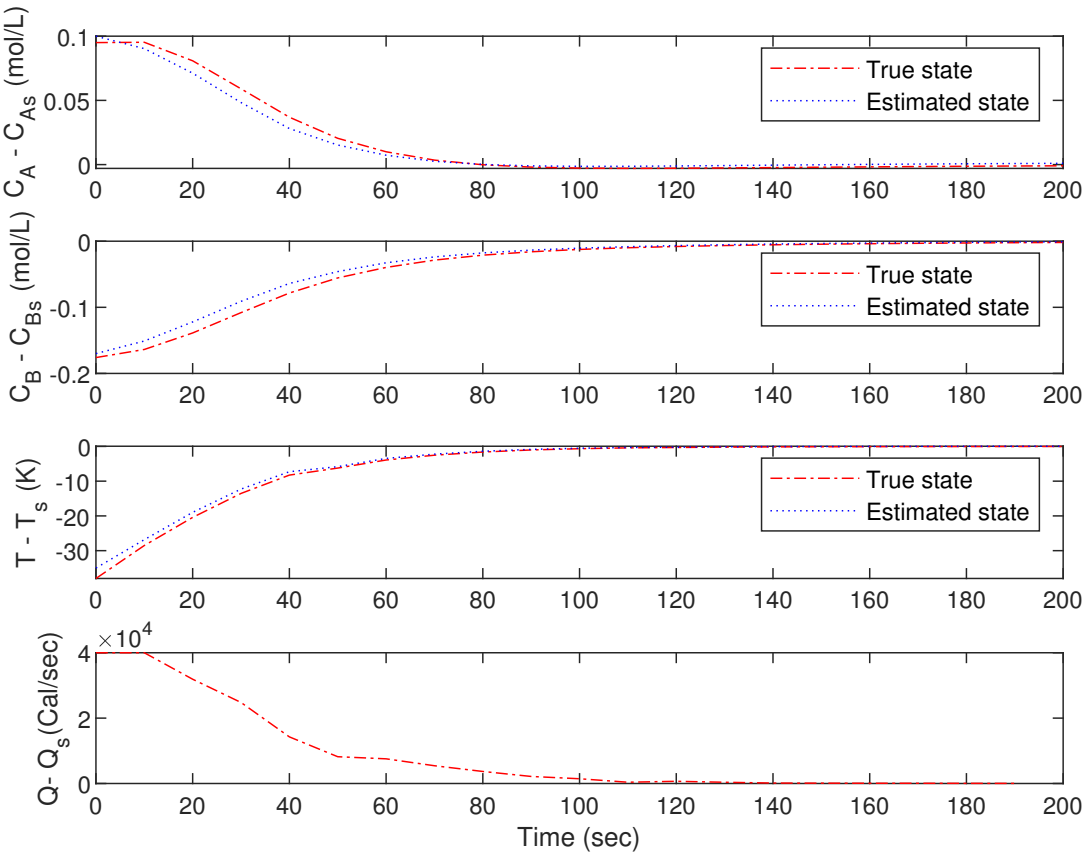

Figure 2.4: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using RNN-based estimator with the initial condition IC_2 (top three plots). The bottom plot displays the manipulated input profile.

Table 2.2. The closed-loop stability analysis and the MSE results are similar to those using RNN-based estimator, and are omitted here.

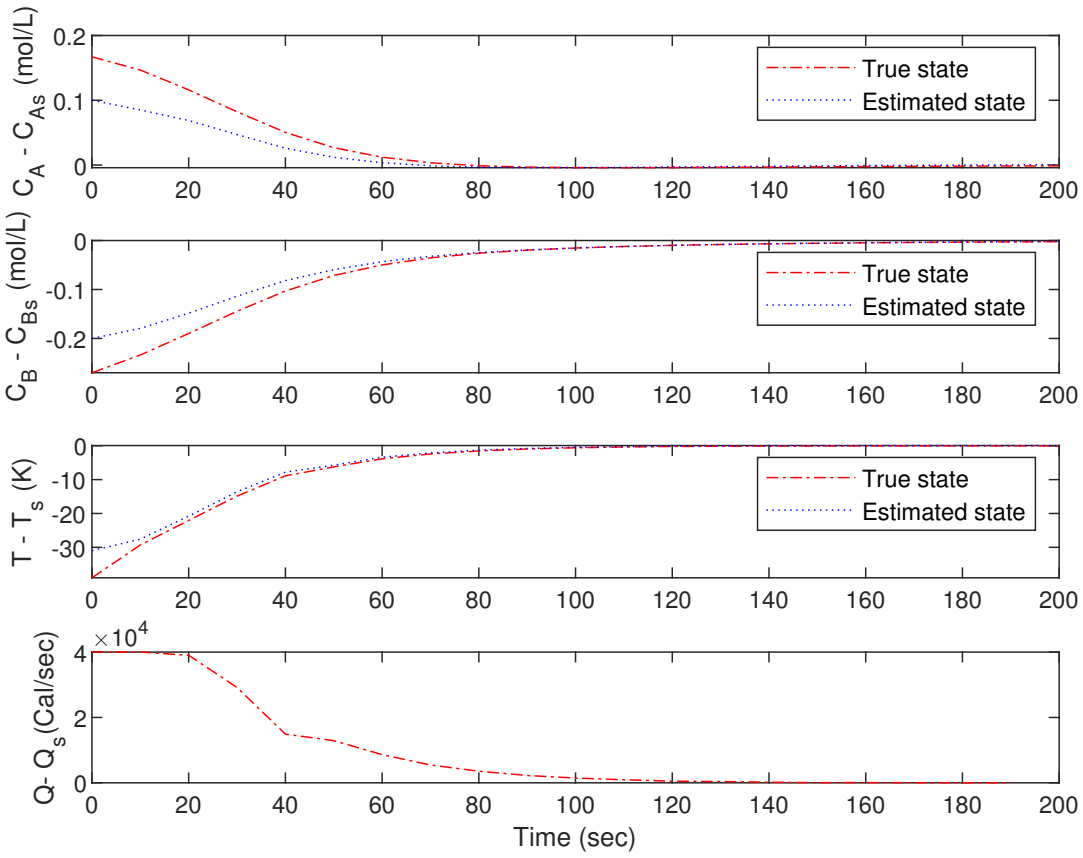

Figure 2.5: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using RNN-based estimator with the initial condition IC_3 (top three plots). The bottom plot displays the manipulated input profile.

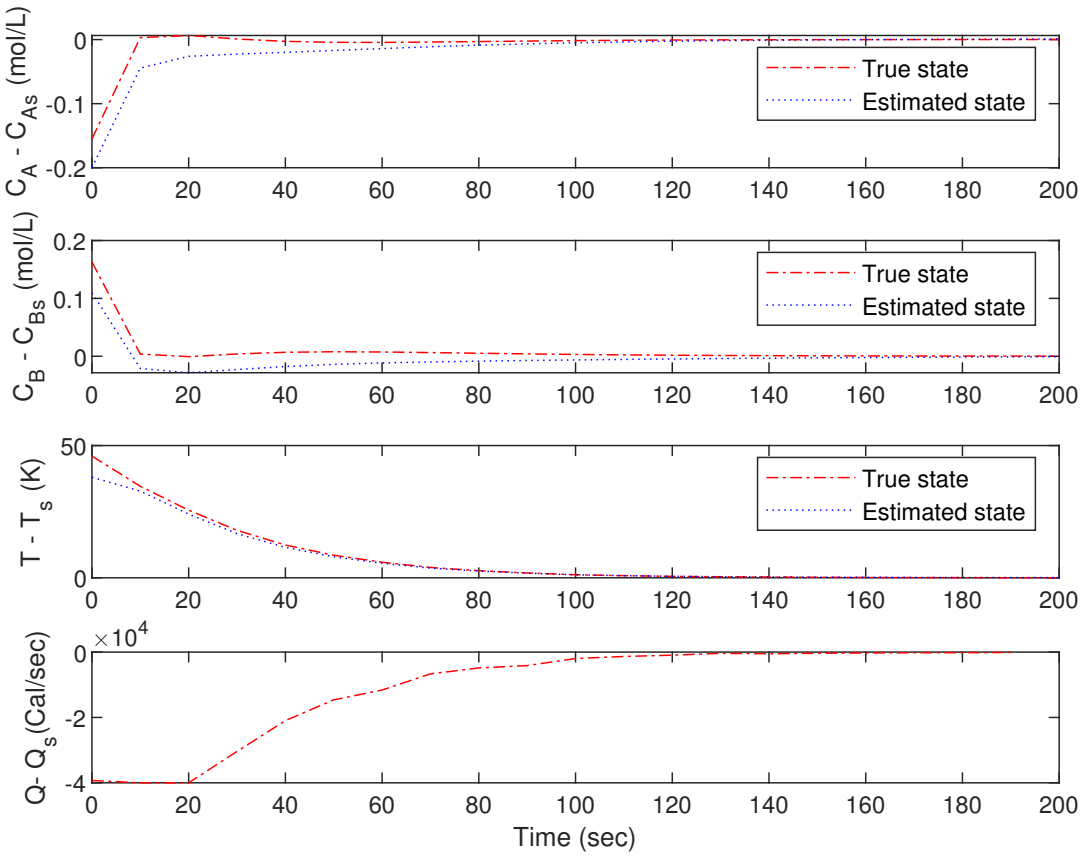

Figure 2.6: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using RNN-based estimator with the initial condition IC_4 (top three plots). The bottom plot displays the manipulated input profile.

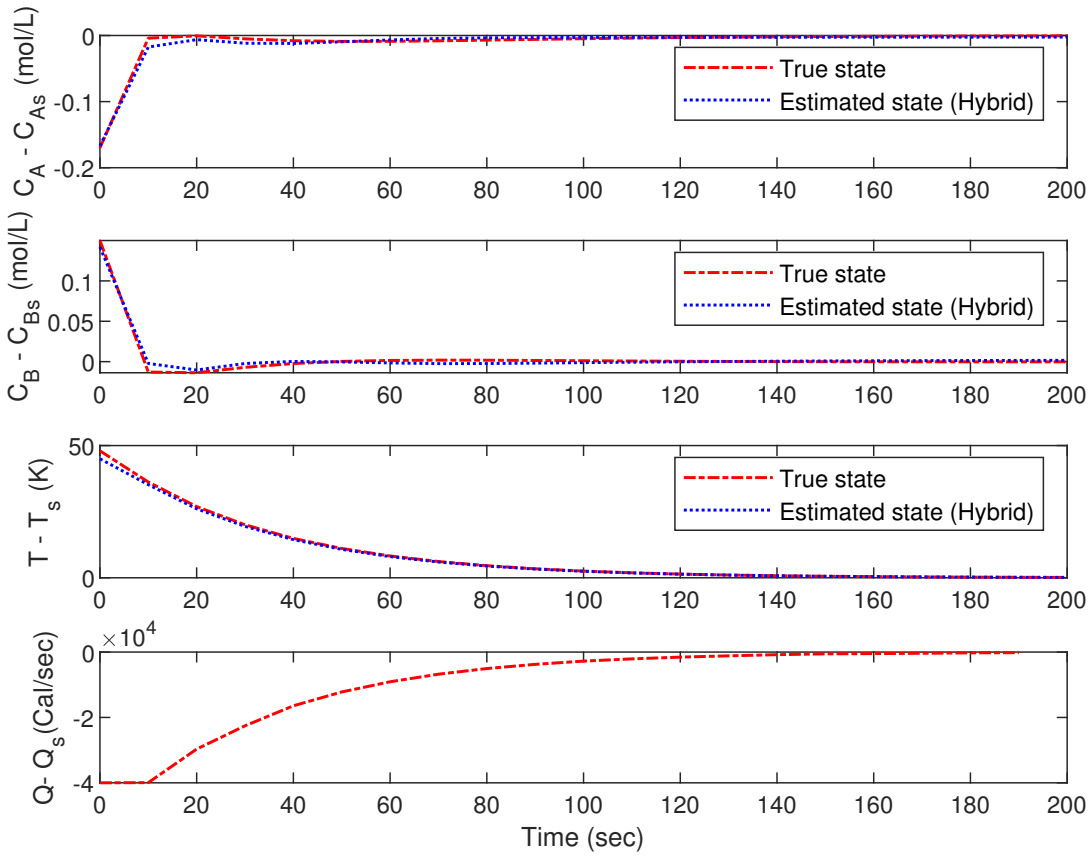

Figure 2.7: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using hybrid-model-based estimator with the initial condition IC_1 (top three plots). The bottom plot displays the manipulated input profile.

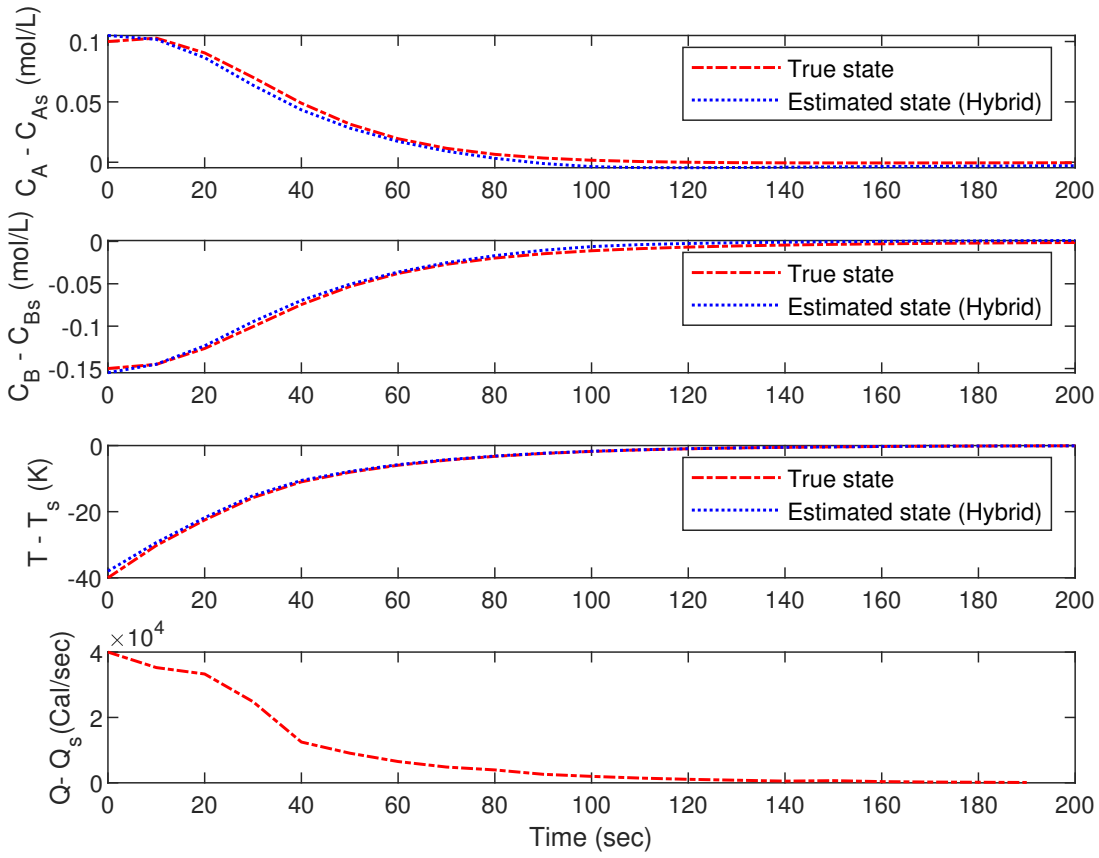

Figure 2.8: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using hybrid-model-based estimator with the initial condition IC_2 (top three plots). The bottom plot displays the manipulated input profile.

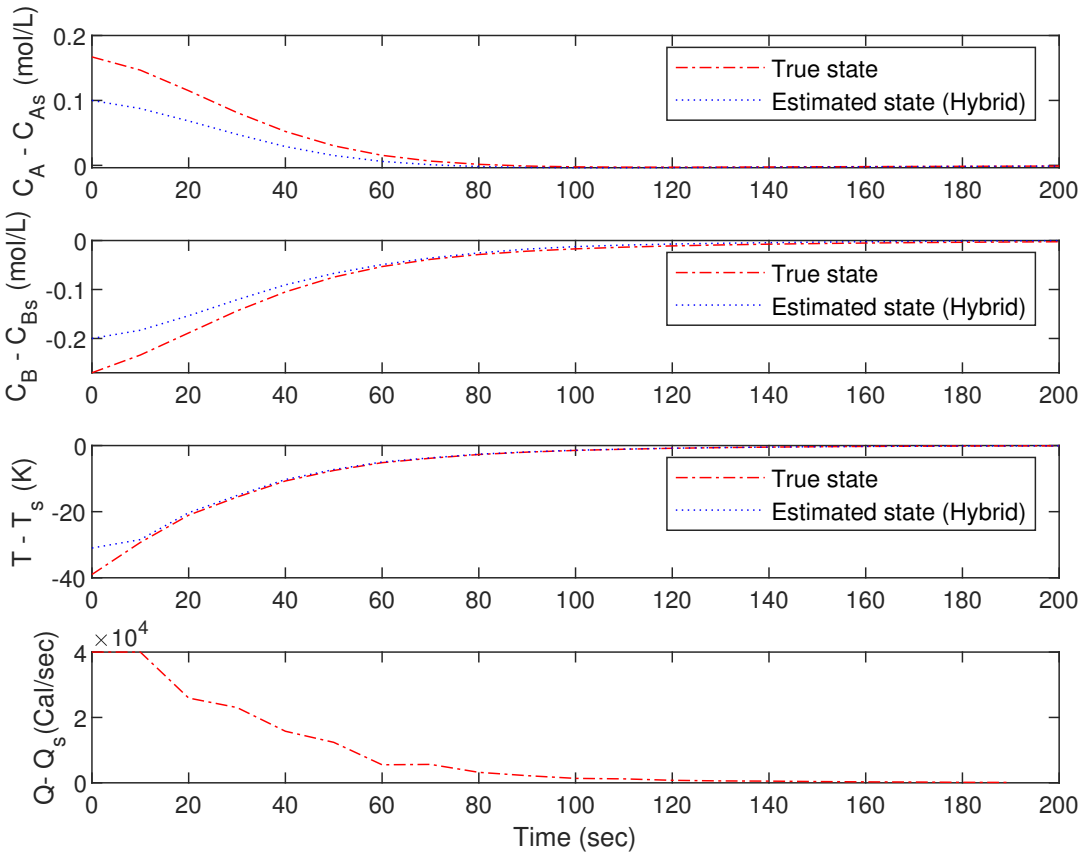

Figure 2.9: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using hybrid-model-based estimator with the initial condition IC_3 (top three plots). The bottom plot displays the manipulated input profile.

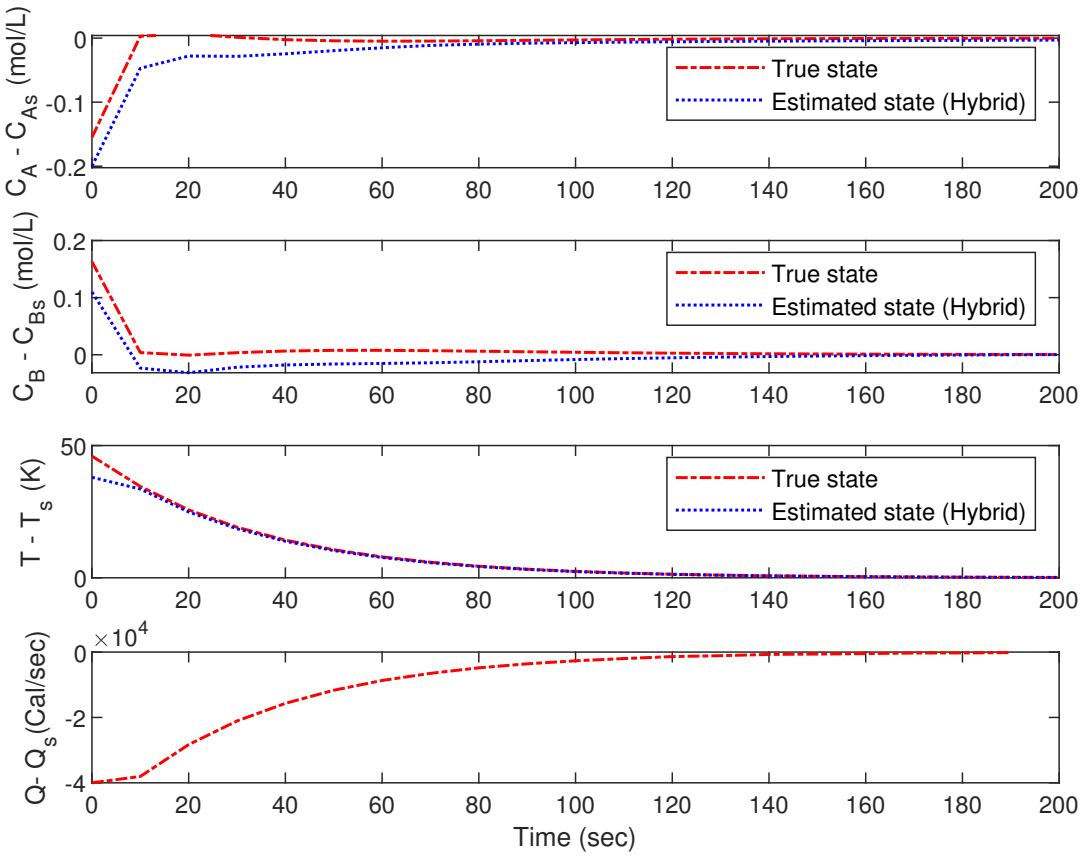

Figure 2.10: True state (red line) and estimated state (blue line) trajectories for the closed-loop CSTR under LMPC using hybrid-model-based estimator with the initial condition IC_4 (top three plots). The bottom plot displays the manipulated input profile.

Table 2.2: Estimation mean squared error of the closed-loop CSTR under LMPC using RNN-based and hybrid-model-based state estimators

Model	Simulation No.	$MSE\ of\ x_1$	$MSE\ of\ x_2$
RNN model	1	1.3699×10^{-5}	9.9753×10^{-6}
	2	1.9458×10^{-5}	5.606×10^{-5}
	3	6.0499×10^{-4}	5.3197×10^{-4}
	4	3.24×10^{-4}	7.41×10^{-4}
Hybrid model	1	1.6198×10^{-5}	1.409×10^{-5}
	2	1.5112×10^{-5}	1.266×10^{-5}
	3	5.8035×10^{-4}	4.5938×10^{-4}
	4	3.6534×10^{-4}	3.9027×10^{-4}

Chapter 3

Process structure-based recurrent neural network modeling for predictive control: A comparative study

3.1 Introduction

Building model-based control systems for industrial applications requires mathematical models that explain the connection between the manipulated inputs and the process outputs. Depending on the process of interest, the construction of a process model is currently founded either on first-principles theory or process data under various assumptions. However, there are some restrictions on model performance. For instance, linearized representations of nonlinear processes are only valid in a restricted region of the linearization's operational point. Finding an appropriate first-principles model is typically not an easy endeavor due to the dynamical nature, inbuilt nonlinear behavior, and high complexity of the majority of chemical processes.

To address this problem, the investigation of utilizing artificial intelligence (AI) techniques in chemical engineering has been carried out continuously. The AI technology has provided classic and powerful modeling tools such as fuzzy logic in the 1960s [125], expert systems in the

1980s [57, 59], and machine learning (ML) in the 1990s [101]. Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications [12, 27, 89, 109]. For example in [12], 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 ±8% deviation from the experimental data. Additionally, among various ML modeling techniques, recurrent neural networks (RNN) have been broadly employed for modelling a general class of dynamical systems for control and state estimation purposes [73]. In [89], a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the RNN model prediction can outperform a first-principles model for large-sacle, complex, nonlinear process, due to its high degree of freedom to solve the complex non-linear regression problem with the process dataset.

With the continuous improvement of data availability and accessibility, machine learning (ML) based model predictive control (MPC) methods receive increasing attention as the next generation of control systems. Conceptually, an MPC contains three major components: a predictive model, an objective function and constraints, and a process optimizer [15]. By using the neural network as the predictive model, the MPC can capture the process dynamics and accordingly make smart decisions: approaching the target states efficiently, automatically, and economically. Furthermore, recent works have demonstrated that ML-based MPC can be utilized to deal with various challenging tasks to improve manufacturing processes such as suppressing measurement noise and searching optimum economic benefits , for which classical control techniques are incapable to accomplish [30, 113].

Fully-connected RNN model (i.e., densely relate all the inputs to all the outputs) is the typical candidate to analyze time-series data in a black-box manner. However, such an approach is not always optimal, especially for complex chemical processes. For instance, in an integrated chemical plant the upstream units affect the downstream units but not the other way around. Therefore, to further improve the RNN model accuracy, various works [26, 48, 92]) have investigated the

gray-box modeling, also known as hybrid modeling, by introducing a *priori* physical knowledge into the development of neural network models of chemical processes. For example, [75] proposed a method which encode that the dynamics between a specific inputs and specific outputs are zero. Recently, the work of [62], formulated a partially-connected RNN model, where the outputs were connected to the impacting inputs only. The resulting RNN model was demonstrated to outperform a fully-connected model. Additionally, [114] used a partially-connected RNN in the framework of ML-based MPC. It was demonstrated that the partially-connected RNN model was able to improve the MPC performance compared to a fully-connected RNN model.

Taking the aforementioned considerations into account, the present work evaluates the performance of a partially-connected RNN-based MPC using a large-scale process simulator of a nonlinear chemical process. First, we construct a simulation model for a chemical plant used to produce Ethylbenzene with two continuous stirred tank reactors (CSTR) in series via the Aspen Plus and Aspen Plus Dynamics simulators. Subsequently, we train a fully-connected and a partially-connected RNN model, respectively, to capture the dynamics of the process using the same datasets obtained from extensive open-loop simulations. Finally, we compare each model's open-loop and closed-loop performance and demonstrate the advantages of using a partially-connected neural network in MPC.

The rest of this manuscript is organized as follow: In Section 3.2, the class of process systems, mathematics notations, and assumption of stabilizing control law are discussed. In Sections 3.3 and 3.4, the concepts methods used to construct fully-connected and partially-connected RNN models respectively are presented. A Lyapunov-based model predictive controller (LMPC) integrated with a RNN model is developed and discussed in Section 3.5. In the last section, the open-loop and closed-loop performances of MPCs using different RNN model structures are evaluated using the chemical process application.

3.2 Preliminaries

3.2.1 Notations

Through this manuscript the notation $|\kappa|$ represents the Euclidean norm of a vector κ. The notation $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ denotes the standard Lie derivative. For set subtraction " $-$ " is used, i.e., $A - B = \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is of class C^1 if it is continuously differentiable.

3.2.2 Class of Systems

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

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

where the state vector of the system is $x = [x_1, ..., x_n]^T \in \mathbf{R}^n$, $y = [y_1, ..., y_q]^T \in \mathbf{R}^q$ is the output vector, and the manipulated input vector is $u = [u_1, ..., u_m]^T \in \mathbf{R}^m$. $F(x,u)$ represents a nonlinear vector function of x and u which is assumed to be sufficiently smooth functions of its arguments. The constraints on control inputs are given by $u \in U := \{{u_i}^{min} \leq u_i \leq {u_i}^{max}\}$. The functions $f(\cdot)$, and $g(\cdot)$ are nonlinear vector and matrix functions of $n \times 1$ and $n \times m$ dimensions, respectively.

3.2.3 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. 3.1 exponentially stable. This stabilizability assumption implies the existence of a 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:

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

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

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

where c_i , $i = 1,2,3,4$, are positive constants. A candidate controller $\Phi(x)$ may be constructed via Sontag's control law formula [61]. Then, following [118], 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 $F(x,u)$ 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 $u \in U$:

$$|F(x,u)| \leq M \tag{3.3a}$$

$$|F(x,u) - F(x',u)| \leq L_x |x - x'| \tag{3.3b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x,u) - \frac{\partial V(x')}{\partial x} F(x',u) \right| \leq L_x^{'} |x - x'| \tag{3.3c}$$

3.3 Recurrent Neural Networks (RNN) Models

As mentioned in the introduction, RNN models are suitable for modeling time-series data. The recursive action in the hidden layer neurons allows the RNN to hold the memory of the previous states, such that it can adequately approximate a time-series dataset. In this work, the RNN model used to approximate the nonlinear system of Eq. 3.1 using the process operational data can be represented as:

$$\dot{\bar{x}} = F_{rnn}(\bar{x}, u) := A\bar{x} + \Theta^T \, y \tag{3.4}$$

where $\bar{x} = [\bar{x}_1, ..., \bar{x}_n]$ is the state vector of the RNN, and the manipulated input vector is $u = [u_1, ..., u_m]$. As a vector of both $\bar{x}$ and u, the vector y is defined as $[y_1, ..., y_n, y_{n+1}, ..., y_{m+n}]$=$[\sigma(\bar{x}_1), ...,$

$\sigma(\bar{x}_n), u_1, ..., u_m] \in \mathbf{R}^{n+m}$. The notation $\sigma(\cdot)$ represents the nonlinear activation function (e.g., a hyperbolic tangent function) used in the hidden layers. $A = diag[-a_1, ..., -a_n]$ is a diagonal negative coefficient matrix where $a_i > 0$ such that each state $\bar{x}$ is stable in the sense of bounded-input bounded-state stability. The notation $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(n+m)\times n}$ is a matrix containing associated weights to be optimized during the neural network training process. Therefore, the vector $\theta_i = b_i[w_{i1}, ..., w_{i(n+m)}]$ is an element of Θ where b_i is a constant, and w_{ij} stands for the weight on the connection between the jth input to the ith neuron where $j = 1, ..., (n+m)$ and $i = 1, ..., n$.

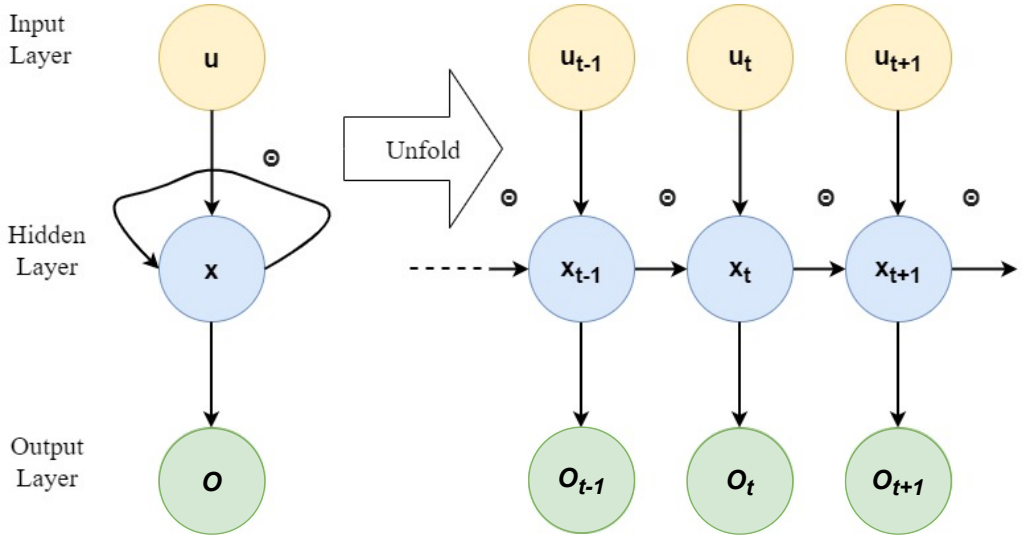

Figure 3.1: A schematic of a recurrent neural network.

Subsequently, the RNN is trained following a standard learning procedure as discussed in [8]. The datasets for training, validation and testing are generated from extensive open-loop simulations of the process model under sufficient variation of initial conditions and control actions. In particular, the continuous-time system of Eq. 3.1 is numerically integrated using the explicit Euler's method with an appropriately small integration time step h_c, and the control actions u are implemented in a sample-and-hold fashion, i.e., $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$

and Δ denotes sampling period. Since the RNN is known for its ability to capture nonlinear process dynamic behavior from time-series data [22,74], the RNN model can be trained using all or some of the integration step data points (i.e., data at each integration time step h_c) within each sampling time to be able to capture the state evolution. Furthermore, the RNN model needs to satisfy a sufficiently small modeling error ν (i.e., $|\nu| = |F(x,u) - F_{rnn}(\bar{x},u)| \leq \gamma|x| \leq \nu_{min}$, where $\gamma, \nu_{min} > 0$) during the model training process, and thus it can well represent process dynamics within the considered operating region.

Remark 3.1. *The modeling error ν is not a constant under different inputs and states. However, by limiting the operation to be within the stability region Ω_ρ, the inputs and states are both bounded. Therefore by training the RNN model using the datasets generated within Ω_ρ, the modeling error can be upper bounded by a sufficiently small positive value ν_{min} for all the states within the stability region.*

3.4 Partially-connected RNN Model

A neural network model which takes all accessible process inputs to provide prediction of the outputs of interest is favored in developing a dynamic model for a nonlinear process. Developing a dynamic model for such processes can be easily implemented using open-source machine learning packages, and this model would be capable to account for all possible relationships between every input and every output of the underling process. In Fig. 5.3, the illustration on the left side represents the general structure of a fully-connected RNN with an input layer, hidden layers, and an output layer. Hence, fully-connected RNN models are usually the prime candidate for processes with no priori knowledge.

The expression process structure knowledge refers to a physical understanding of the underlying process that exists in advance to the development of first-principles model process model. It includes but not limited to the model's intended purpose, soft or hard physical constraints imposed on the process by design considerations, and process structure. In this work,

we focus on process structure knowledge in terms of relationships between the process input and output variables. Particularly, in chemical process industry, the physical relations among the process variables can be straightforward. For instance, the upstream processes affect the downstream stage processes, while such an effect dose not exist in the opposite direction. This connection between upstream and downstream stages is often reflected in a mathematical model obtained via first-principles. Therefore, incorporating process knowledge that describes physical relations among the underlying process inputs and outputs into the neural network structural modeling will improve its performance as discussed in [94]. This modeling methodology is called partially-connected neural network [114].

In this study, a partially-connected RNN model, as illustrated in Fig. 5.3 on the right side, is developed for the nonlinear system of Eq. 3.1. We consider that $\mathbf{u} = [u^1 \in \mathbf{R}^{m_1}, u^2 \in \mathbf{R}^{m_2}]$ and $\mathbf{x} = [x^1 \in \mathbf{R}^{n_1}, x^2 \in \mathbf{R}^{n_2}]$ where $m = m_1 + m_2$ and $n = n_1 + n_2$, and that only the input vector u^1 affects the state vector x^1, while x^2 is affected by the two input vectors u^1 and u^2. By adjusting the RNN structure to explicitly exclude the connection between x^1 and u^2, physical knowledge is integrated into the RNN modeling of the nominal system. As a result, an improved approximation is achieved. For example, as discussed in [114], the partially-connected RNN models can remarkably reduce the required number of hidden neurons, and weight parameters to achieve the desired performance compared to fully-connected RNN model. Moreover, partially-connected RNNs may be able to capture the process dynamics using a smaller training dataset, since the use of process knowledge may simplify the optimization process of an RNN model by revealing the correct search direction.

The development of partially-connected RNN models follows the development framework of fully-connected RNN models, but with more specifications. A dataset for training/validation can be constructed either from experimental and industrial sources, or by extensive open-loop simulation as discussed in the second paragraph of the previous section. The rule of thumb of splitting the collected dataset to 70% training and 30% validating can be employed, or more sophisticated methods such as cross validation may be used. The input vectors u^1, u^2 and the output vectors x^1, x^2 should be specified before RNN models training. This step is also known as data pre-processing.

In this work, we used the open-source library 'Keras' in Python to construct and train the RNN models. Specifically, our models are constructed with an input layer, an output layer, and two hidden layers activated by the nonlinear functions: hyperbolic tangent and sigmoid functions. To generate a partially-connected RNN model, rather than taking the input vector $\mathbf{u}$ as in the fully-connected model, the input vectors u^1 and u^2 are fed separately using different input layers with respect to the process structure as illusteated in Fig 5.3. The first hidden layer takes the input vector u^1 and predicts the output vector x^1. Subsequently u^2 and x^1 are merged and sent to the second hidden layer to predict x^2. Eventually, the constructed model can provide prediction for both output vectors x^1 and x^2. The Pseudocodes 1 and 2 summarize the construction procedure of the partially-connected RNN models, and may be useful to colleagues who may pursue the approach.

Remark 3.2. *Both the fully-connected and the partially-connected RNN models structures are developed for the nominal system described in Eq. 3.1 assuming no disturbances. In the presence of time-varying disturbances, the prediction of RNN models may under-perform due to model mismatch. To resolve this issue, the RNN models can be updated online using recent process measurements to capture the process-model mismatch caused by the disturbances.*

Remark 3.3. *Note that partially-connected RNN models only reflect process structure on its internal network connection without explicit expression. Therefore this model is still a "black-box" model which is different from the hybrid models. For discussion on hybrid modeling of chemical processes, the interested readers may refer to [8, 39, 129].*

Remark 3.4. *The classes and features of layers in Pseudocodes are named in the Keras manner. The naming could be different for other machine learning application programming interface (API), such as Tensorflow and Pytorch. The number of input and output features, and the number of data points in the data sequence should be specified in the data prepossessing step as mentioned in this section and the previous section, respectively.*

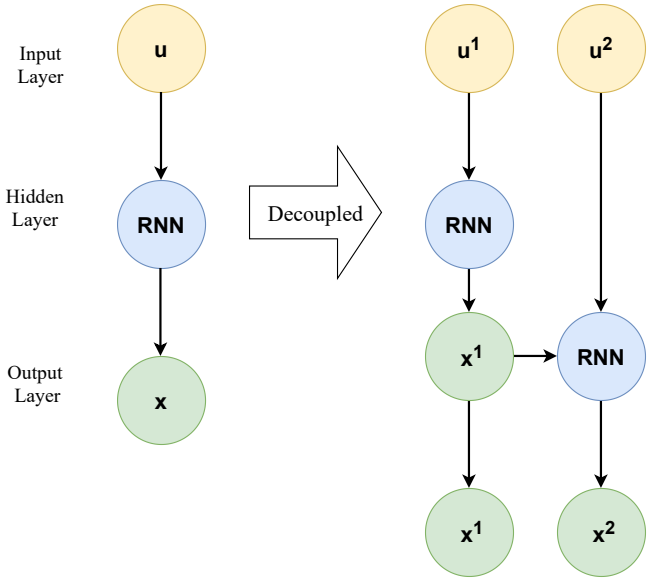

Figure 3.2: Fully-connected and Partially-connected RNN structure, where $\mathbf{u} = [u^1, u^2]$ and $\mathbf{x} = [x^1, x^2]$.

Pseudocode 1: Partially-connected RNN Construction

input layer-1:

{

layer class: Input

units: (number of input features in vector u^1*)*

input shape: (number of data points in the input data sequence, number of input features in vector u^1*)*

connected to: hidden layer-1

}

hidden layer-1:

{

layer class: Long Short-term Memory

units: (Number of inputs) $\times$ *(Number of outputs)*

return sequences: true

activation function: tanh

recurrent activation function: sigmoid

recurrent initializer: orthogonal

use bias: true

connected to: output layer-1

}

output layer-1:

{

layer class: Dense

units: number of outputs

activation function: linear

output shape: (number of data points in the output data sequence , number of outputs in vector x^1*)*

connected to: merge layer

}

Pseudocode 1: Partially-connected RNN Construction (continued)

input layer-2

{

layer class: Input

units: (number of input features in vector u^2)

input shape: (number of data points in the input data sequence , number of input features in vector u^2)

connected to: merge layer

}

merge layer:

{

layer class: Concatenate

connected to: hidden layer-2

}

hidden layer-2

{

layer class: Long Short-term Memory

units: (Number of inputs) $\times$ (Number of outputs)

return sequence: true

activation function: tanh

recurrent activation function: sigmoid

recurrent initializer: orthogonal

use bias: true

connected to: output layer-2

}

output layer-2

{

layer class: Dense

units: number of output

activation function: linear

output shape: (number of data points in the output data sequence , number of outputs in vector x^2)

}

Pseudocode 2: Partially-connected RNN Training

model compile
{
optimizer: adam (candidate optimizer: RMSprop, SGD, etc.)
loss function: mean squared error
}

early stop
{
monitor: validation loss
early stop condition: 1×10^{-8}
}

model fit
{
training (x_t, y_t) *:*
x_t*: python list (input training set for input layer 1, input training set for input layer 2)*
y_t*: python list (output training set for output layer 1, output training set for output layer 2)*
batch size: 32 (defaults value)
epochs: 50 (user choice, other numbers can be used)
validation (x_v, y_v) *:*
x_v*: python list (input validation set for input layer 1, input validation set for input layer 2)*
y_v*: python list (output validation set for output layer 1, output validation set for output layer 2)*

callbacks: early stop
}

3.5 RNN-Based Model Predictive Control

In this section, we incorporate an RNN model in a Lyapunov-based model predictive control (LMPC) strategy. Specifically, the RNN model (with partially-connected or fully-connected structure) provides state predictions to solve the MPC optimization problem, which is formulated as follows:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+P}} L(\tilde{x}(t), u(t)) dt \tag{3.5a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{rnn}(\tilde{x}(t), u(t)) \tag{3.5b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.5c}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+P}) \tag{3.5d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}} \tag{3.5e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+P}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{3.5f}$$

where predicted state trajectory is $\tilde{x}$. $S(\Delta)$ denotes the set of constant piecewise functions with period Δ, and the prediction horizon is P. The function $\dot{V}(x,u)$ in Eq. 3.5e is the time-derivative of Lyapunov function V (i.e., $\frac{\partial V(x)}{\partial x}(F_{rnn}(x,u))$). The LMPC computes the optimal inputs series $u^*(t)$ over the specified prediction horizon $t \in [t_k, t_{k+P})$. The first optimal inputs $u^*(t)$ of the each prediction horizon is sent to the system to be implemented for the next sampling period. Then, the new state measurements are fed back to the LMPC and the control optimization problem is resolved again with the new state measurements at the next sampling period. Moreover, the objective of the MPC optimization problem is to minimize the time integral cost function $L(\tilde{x}, u)$ as represented in Eq. 3.5a over the prediction horizon while satisfying the constraints of Eq. 3.5b-3.5f. The first constraint of Eq. 3.5b is the RNN model from Eq. 3.4 that is utilized to predict the evolution of the closed-loop state. In Eq. 3.5b, $x(t_k)$ is used to update the initial condition of the prediction $\tilde{x}(t_k)$. As for the inputs, the constraints are represented by Eq. 3.5d, which are applied throughout the entire prediction horizon.

To maintain the closed loop stability, the contractive constraint of Eq. 3.5e is activated when $x(t_k) \in \Omega_\rho - \Omega_{\rho_{rnn}}$. This constraint forces the Lyapunov function of the closed-loop states to decrease, and as a result, the actual state will approach the steady-state in finite time. Furthermore, if the last state $x(t_k)$ enters the desired region $\Omega_{\rho_{rnn}}$, then the predicted closed-loop state will be maintained within this region for the entire prediction horizon. The work of [118] demonstrated that when using RNN-based LMPC as in Eq. 3.5 to control a nonliner system as given in Eq. 3.1, the closed-loop state is guaranteed to be bounded within the stability region Ω_ρ for all times, and ultimately it will converge to a very small neighborhood around the origin under the assumption that the modeling error ν is sufficiently small.

Remark 3.5. *In the case where $x(t_k)$ are not fully available online, a state observer may be used to estimate the unmeasured states form the measured ones. The previous work [8] developed two different machine learning based state estimators in the framework of ML based LMPC for nonlinear processes. It was demonstrated that both the ML-based and the hybrid-model based estimators achieved accurate state estimation, and that all state trajectories initiating from various initial conditions converged to the steady-state under the LMPC.*

3.6 Application to a Chemical Process Modeled in Aspen Plus

In this section, we use a large-scale chemical process to evaluate the proposed partially-connected RNN-based LMPC. Firstly, we develop dynamic simulations of two models for the chemical process using the Aspen Plus Dynamics V11 and first-principles modeling principles, respectively. Subsequently, a process time-series dataset is collected to train and test the RNN models via extensive open-loop simulation. Finally, open-loop simulations and closed-loop simulations under RNN-model based MPC are carried out and discussed.

3.6.1 Dynamic Model in Aspen Plus Dynamics

The Ethylbenzene (EB) production process using Ethylene (E) and Benzene (B) as raw materials is considered. The main reaction for this process is a second-order, exothermic, and irreversible reaction, and it is taking place along with two other side reactions as described in Eq. 3.6 below in two non-isothermal, well mixed continuous stirred tank reactors (CSTR). The chemical reactions are as follows:

$$C_2H_4 + C_6H_6 \rightarrow C_8H_{10} \tag{3.6a}$$

$$C_2H_4 + C_8H_{10} \rightarrow C_{10}H_{14} \tag{3.6b}$$

$$C_6H_6 + C_{10}H_{14} \rightarrow 2\,C_8H_{10} \tag{3.6c}$$

In this work, the two CSTRs are placed in series, and the process model is developed using Aspen Plus and Aspen Plus Dynamics V11, known as a high-fidelity software for complex chemical processes. Initially, the process model is constructed in Aspen Plus where a steady-state simulation is performed and checked based on material and energy balances. Subsequently, we carry out dynamic simulation of this process in Aspen Plus Dynamics to analyse and control its dynamic performance. We construct the steady-state and the dynamical models through the following procedure:

(1) Inlet Streams Specification: The raw materials are fed to each reactor as Hexane solutions by the flow rates F_1 and F_2. Hexane solution is used to ensure that the inlet flows remain in liquid phase under the feeding temperature. The concentration of Ethylene, Benzene, Ethylbenzene, and Di-Ethylbenzene are denoted by C_E, C_B, C_{EB}, and C_{DEB}, respectively. T_i, ρ_i, and V_i are the temperature, mass density, and the liquid volume of CSTR$_i$, $i = 1,2$. The mass heat capacity of the liquid mixture is denoted by C_p, and it is assumed to be constant. The values of process parameters used are listed in Table 4.1, where the subscript "o" indicates the initial state, and "s" represents the steady-state.

(2) Pressure Drop Selection: To establish a dynamic model for Aspen Plus Dynamics, valves are essential as connectors of parts and fluid flow by manipulating pressure drop throughout the process. With a proper pressure drop, the simulation runs smoothly and the model is able to specify the flow direction, and if inadequate pressure drop is selected, it will return a simulation error. In our model, the pressure drop in valves v_1, v_2, v_3, and v_4 are chosen to be 5, 5, 2, and 14 bars, respectively.

(3) Reactor Setting: Each CSTR_i is associated with a heating/cooling jacket which supplies/removes heat at rate Q_i, $i = 1, 2$. The initial pressure of both CSTRs are set to be 15 bar, and the initial temperature of the first and second CSTR are 400K and 450K, respectively, to keep the reactants and products in liquid phase during the process. Those values will be automatically adjusted by performing build-in steady-state simulations. After setting up the reactions in the two CSTRs, steady-state simulation is executed for the purpose of analysing the plant behavior.

(4) Thermodynamic Parameters & Reactor Geometric: Before exporting the steady-state model from Aspen Plus to Aspen Plus Dynamics, the thermodynamic parameters and the reactor geometry need to be specified. For our model, the vessels type is vertical, the head type is flat, and the length of each CSTR is ten meters. The thermodynamics parameters are listed in Table 4.1.

(5) Pressure Checking: To make sure that the dynamic model is set properly, we run the steady-state simulation again and perform pressure checking via the built-in Aspen Plus pressure checker without encountering errors. Subsequently, the steady-state model is exported to Aspen Plus Dynamics.

(6) Dynamic Model Initialization: A direct-acting level controller is added to each reactor to maintain the reactors at half capacity. The level controller can be designed and added following the default setting in Aspen Plus before exporting the steady-state model, or it can be developed manually in Aspen Plus Dynamics. Following the level controllers

configuration, we apply a steady-state simulation to obtain the steady-state values of the dynamical model, which gives $Q_{1s} = -911.455kW$ and $Q_{2s} = -6835.270kW$.

(7) Data Type Configuration: In order to allow the outside control of the manipulated variables (i.e., Q_1 and Q_2) during the dynamic simulation, the heating type of the two reactors are changed to constant duty. Also, the volumetric flow rates F_1 and F_2 are set as fixed constants, and a steady-state simulation is performed again to ensure that the dynamic model remains at steady-state after the data type configuration. Hence, building the process dynamical model is completed, and the model's flow sheet is illustrated in Fig. 4.5.

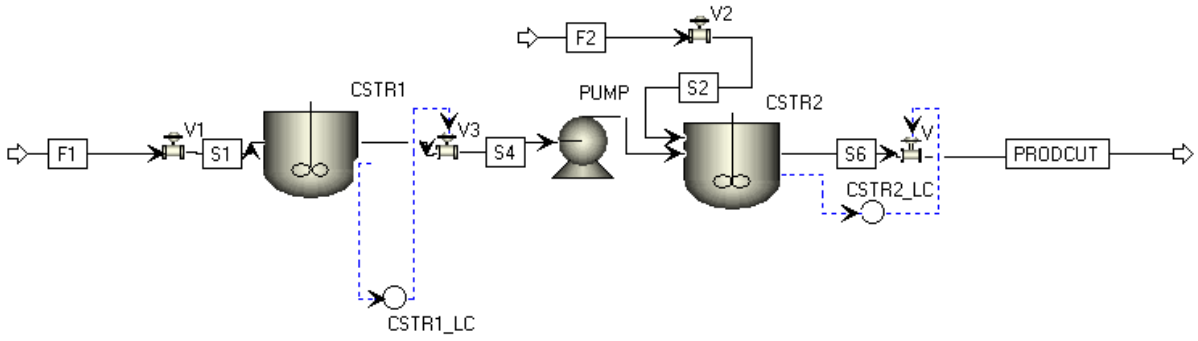

Figure 3.3: Aspen Plus model flow sheet of two reactors in series.

Open-loop simulation is performed using the constructed dynamical model with pseudorandom input signals generated by a MATLAB script. To link Aspen Plus Dynamics with MATLAB, a local message passing interface (MPI) is created, such that the dynamical model is able to automatically read the input signals from MATLAB and then implement them in the dynamic simulations. In particular, the MATLAB script generates the manipulated variables in deviation form against their steady-state values ($u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$). The two manipulated variables randomly vary within the range of $[-1 \times 10^4 kW, 1 \times 10^3 kW]$, and $[-1.5 \times 10^4 kW, 5 \times 10^3 kW]$ respectively, and are implemented to the dynamic simulation in a sample-and-hold manner that the values are updated every five minutes (simulated time). All input values and output states (e.g., C_E, C_B, and T) are recorded in time-series to establish the training/validating dataset.

Table 3.1: Parameter and steady-state values of the Aspen Plus model.

$T_{1_o} = 350\ K$	$T_{1_s} = 310.523\ K$
$T_{2_o} = 350\ K$	$T_{2_s} = 430.542\ K$
$F_1 = 43.2\ m^3/hr$	$F_2 = 91.079\ m^3/hr$
$C_{E_1} = 4.2455\ kmol/m^3$	$C_{E_2} = 0.3254\ kmol/m^3$
$C_{B_1} = 5.3532\ kmol/m^3$	$C_{B_2} = 1.3841\ kmol/m^3$
$C_{EB_1} = 0.1854\ kmol/m^3$	$C_{EB_2} = 3.8744\ kmol/m^3$
$C_{DEB_1} = 9.1426 \times 10^{-7}\ kmol/m^3$	$C_{DEB_2} = 0.0058\ kmol/m^3$
Heat transfer option	*Dynamics*
Medium temperature	298 K
Temperature approach	77.33 K
Heat capacity of coolant	4200 J/kgK
Medium holdup	1000 kg
$C_p = 2.411\ kJ/kgK$	$\rho_1 = 639.1530\ kg/m^3$
$V_1 = V_2 = 60\ m^3$	$\rho_2 = 607.5040\ kg/m^3$

3.6.2 First-principles Model Development

Aspen Plus Dynamics is a highly efficient software that allows chemical engineers to simulate, and to optimize chemical process performance and profitability. However, due to their long computational time, typically the Aspen Plus models are not the optimal option to generate data for deep-learning models which require comparatively large amount of data. To overcome this issue, first-principles models with simplifying assumptions are well-established candidates to generate data for machine learning [24, 93]. By applying the concepts of mass and energy balances, the first-principles models for the CSTRs are developed. Specifically, the dynamic model of the first CSTR is represented by the following ODEs:

$$\frac{dC_{E_1}}{dt} = \frac{F_1 C_{Eo_1} - F_{out_1} C_{E_1}}{V_1} - r_1 - r_2 \tag{3.7a}$$

$$\frac{dC_{B_1}}{dt} = \frac{F_1 C_{Bo_1} - F_{out_1} C_{B_1}}{V_1} - r_1 - r_3 \tag{3.7b}$$

$$\frac{dC_{EB_1}}{dt} = \frac{-F_{out_1} C_{EB_1}}{V_1} + r_1 - r_2 + 2r_3 \tag{3.7c}$$

$$\frac{dC_{DEB_1}}{dt} = \frac{-F_{out_1} C_{DEB_1}}{V_1} + r_2 - r_3 \tag{3.7d}$$

$$\frac{dT_1}{dt} = \frac{(T_{0_1} F_1 - T_1 F_{out_1})}{V_1} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_1 C_p} r_j + \frac{Q_1}{\rho_1 C_p V_1} \tag{3.7e}$$

The dynamic model of the second CSTR is comprised of the following ODES:

$$\frac{dC_{E_2}}{dt} = \frac{F_2 C_{Eo_2} + F_{out_1} C_{E_1} - F_{out_2} C_{E_1}}{V_2} - r_1 - r_2 \tag{3.8a}$$

$$\frac{dC_{B_2}}{dt} = \frac{F_2 C_{Bo_2} + F_{out_1} C_{B_1} - F_{out_2} C_{B_1}}{V_2} - r_1 - r_3 \tag{3.8b}$$

$$\frac{dC_{EB_2}}{dt} = \frac{F_{out_1} C_{EB_1} - F_{out_2} C_{EB_2}}{V_2} + r_1 - r_2 + 2r_3 \tag{3.8c}$$

$$\frac{dC_{DEB_2}}{dt} = \frac{F_{out_1} C_{DEB_1} - F_{out_2} C_{DEB_2}}{V_2} + r_2 - r_3 \tag{3.8d}$$

$$\frac{dT_2}{dt} = \frac{(T_{0_2} F_2 + T_1 F_{out_1} - T_2 F_{out_2})}{V_2} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_2 C_p} r_j + \frac{Q_2}{\rho_2 C_p V_2} \tag{3.8e}$$

where the reaction rates are calculated by the following expressions:

$$r_1 = k_1 e^{\frac{-E_1}{RT_i}} C_{E_i} C_{B_i} \tag{3.9a}$$

$$r_2 = k_2 e^{\frac{-E_2}{RT_i}} C_{EB_i} C_{E_i} \tag{3.9b}$$

$$r_3 = k_3 e^{\frac{-E_3}{RT_i}} C_{DEB_i} C_{B_i} \qquad , \; i = 1,2 \tag{3.9c}$$

The parameters used in the first-principles models are listed in Table 4.1. Figs. 3.4 - 3.5 show open-loop simulations of the first-principles model and of the Aspen Plus model under the same time-varying inputs and initial conditions. This simulation illustrate the good agreement between the two models within the operating region.

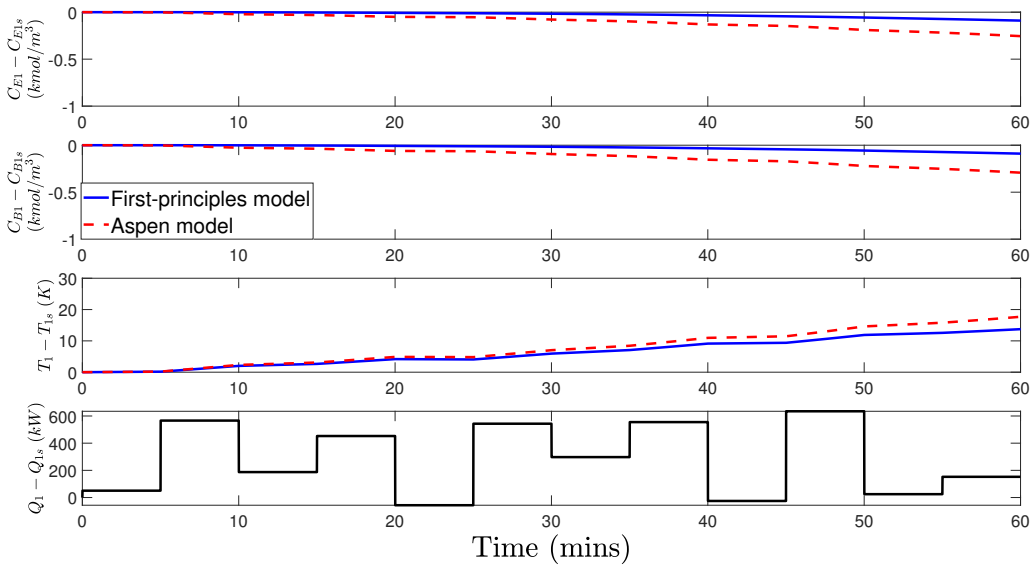

Figure 3.4: Open-loop state and manipulated input profiles for $CSTR_1$.

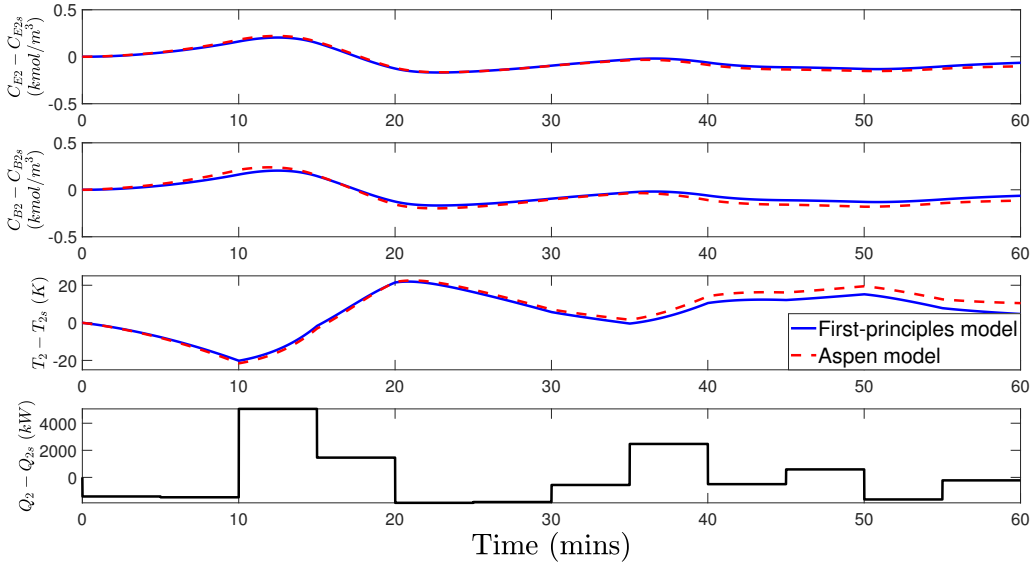

Figure 3.5: Open-loop state and manipulated input profiles for $CSTR_2$.

3.6.3 Data Generation and RNN Models Development

In this work, we create a dataset that contains open-loop simulation data from both the Aspen Plus and the first-principles models to develop the two RNN models. Using Keras library, the two RNN models are constructed following the diagram shown in Fig. 3.6. The fully-connected and partially-connected RNN models are designed as follows: they have two long short term memory

(LSTM) layers with fifty neurons in each, and they are activated by hyperbolic tangent functions (i.e., $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$). For the training part, the fully-connected and partially-connected RNN models are both developed based on the same dataset and via using Keras library with the same neural network parameters as follows: two hidden layers with fifty neurons in every layer, and hyperbolic tangent as the activation function. The output layer is activated by a linear activation function, which gives estimation for ten states, and there are twelve inputs for both neural network models. The input and output variables are listed in Table. 4.1. We use the input data that covers the five minute sampling period to predict the states evolution for the next five minutes. Rather than using the conventional gradient decent optimization algorithm, we use Adam optimizer which is a combination of two algorithms, the gradient descent with momentum and the RMSprop. Moreover, to produce more robust models, we apply five-fold cross validation, and select the model with the least mean squared error (MSE) for each RNN structure.

We train the two models starting with 50 *epochs*, and the partially-connected RNN model reaches the early stopping criteria (i.e., $validation\ loss = 1 \times 10^{-8}$) at the 20^{th} *epoch*. Thus, we use 20 *epochs* to train the fully-connected RNN model for comparison consistency. The training and validating summary for the two models is illustrated in the Fig. 3.7, and the Fig. 3.8. We can see that the two developed models yield high accuracy in both training and validating processes, where the accuracy is demonstrated in terms of MSE between the model prediction and the reference value.

For the open-loop simulation, we designate a message passing interface (MPI) for files exchanging to adopt random control actions from a Python script to the Aspen Plus simulation. The control actions are simultaneously applied to the RNN models. Therefore, this simulation fulfills two objectives: a) checking the connection between Python and Aspen Plus, and b) testing for the open-loop prediction of the two RNN models.

Table 3.3 summarizes the simulation results, and presents the MSE between the predicted states from each RNN model and the corresponding Aspen Plus Dynamics model outputs as reference. Furthermore, the open-loop responses predicted by the partially-connected and the

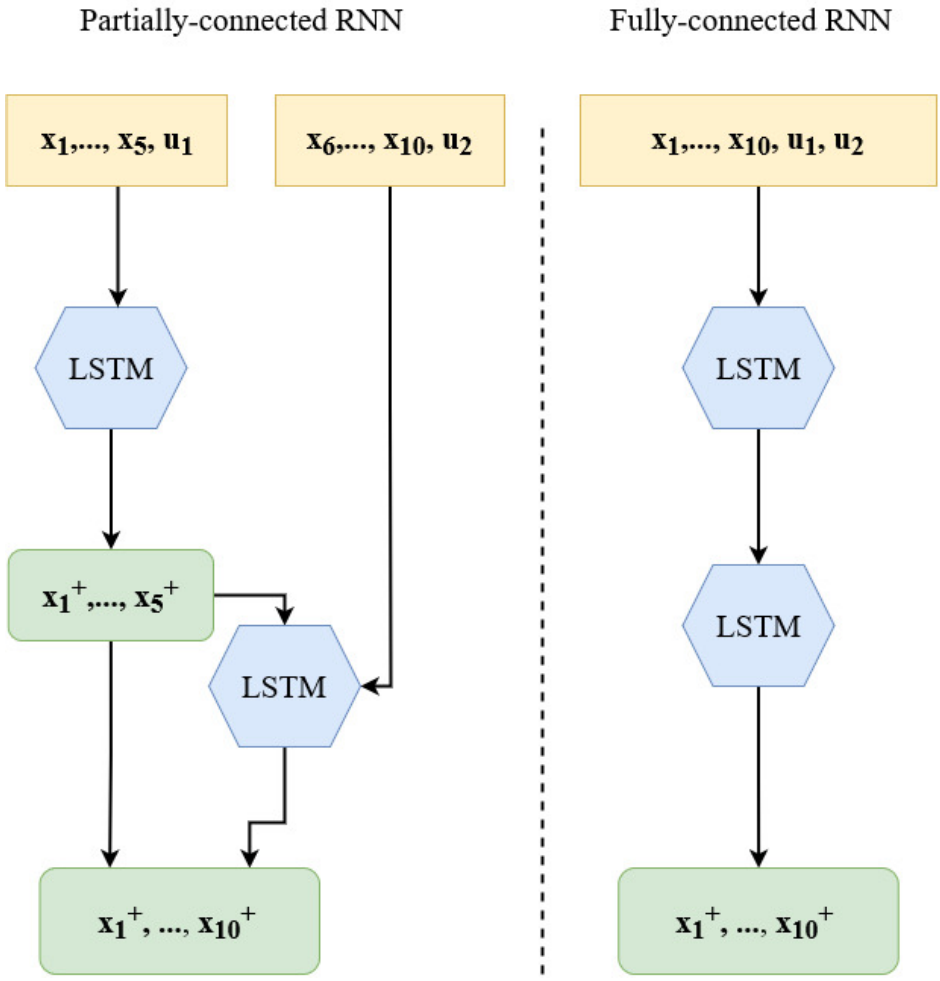

Figure 3.6: RNN modeling structures for Ethylbenzene production in two CSTRs in series, where x_i and x_i^+ are defined at $t = t_k$ and $t = t_k + \Delta$, respectively.

fully-connected RNN models under a step change in u_2 are displayed in Fig.3.9. The Figure demonstrates that the partially-connected RNN model has an improved model identification of the process dynamics. These results indicate that both RNN models provide accurate prediction, yet, the partially-connected RNN model approximates the Aspen process model more accurately.

3.6.4 Closed-loop Simulation: First-principles Process Model

Subsequently, we develop LMPCs based on the fully-connected RNN model and the partially-connected RNN model respectively, to perform the closed-loop simulation with the

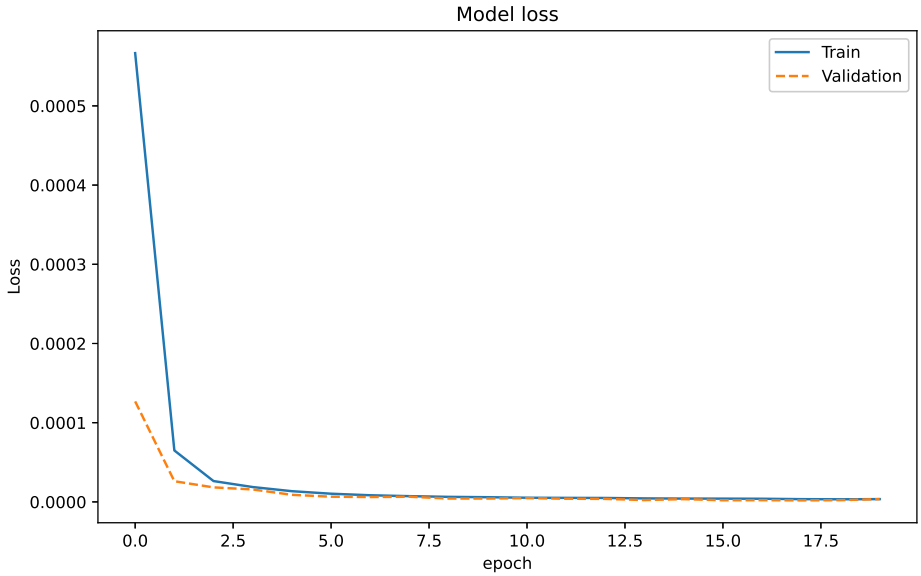

Figure 3.7: Partially-connected RNN model training and validation loss functions.

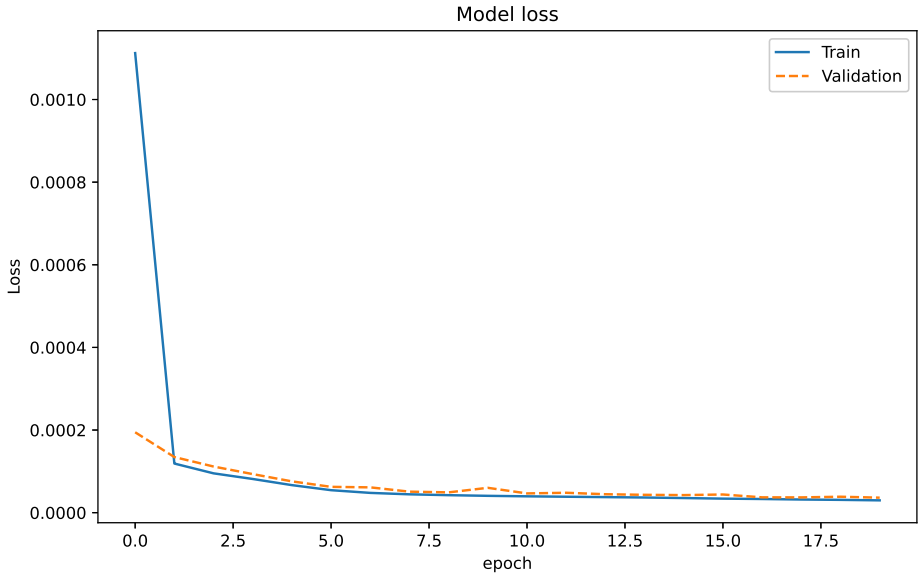

Figure 3.8: Fully-connected RNN model training and validation loss functions.

confidence that both RNN models provide high accuracy approximation for the process outputs. In order to develop the LMPCs, we use the Python version of the interior point optimizer (IPOPT) package to solve the nonlinear optimization problem of the LMPC for each sampling time Δ. This optimizer is an open source package which can be used for solving large-scale nonlinear optimization problems. It employs an interior point line search filter technique which intends to

Table 3.2: Input and output states of the RNN models.

Notation	State (in deviation form)
x_1	Concentration of Ethane for $CSTR_1$
x_2	Concentration of Benzene for $CSTR_1$
x_3	Concentration of Ethylbenzene for $CSTR_1$
x_4	Concentration of Butylbenzene for $CSTR_1$
x_5	Temperature of the reactor for $CSTR_1$
x_6	Concentration of Ethane for $CSTR_2$
x_7	Concentration of Benzene for $CSTR_2$
x_8	Concentration of Ethylbenzene for $CSTR_2$
x_9	Concentration of Butylbenzene for $CSTR_2$
x_{10}	Temperature of the reactor for $CSTR_2$
u_1	Heating/cooling duty of the reactor for $CSTR_1$
u_2	Heating/cooling duty of the reactor for $CSTR_2$
RNN	
Inputs	**Outputs**
$x_1(t_k)$	$x_1(t_k+\Delta)$
$x_2(t_k)$	$x_2(t_k+\Delta)$
$x_3(t_k)$	$x_3(t_k+\Delta)$
$x_4(t_k)$	$x_4(t_k+\Delta)$
$x_5(t_k)$	$x_5(t_k+\Delta)$
$x_6(t_k)$	$x_6(t_k+\Delta)$
$x_7(t_k)$	$x_7(t_k+\Delta)$
$x_8(t_k)$	$x_8(t_k+\Delta)$
$x_9(t_k)$	$x_9(t_k+\Delta)$
$x_{10}(t_k)$	$x_{10}(t_k+\Delta)$
$u_1(t_k)$	
$u_2(t_k)$	

find a local solution of nonlinear programming problems. 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 setpoint error and control actions, respectively. The two matrices are critically impacting the performance of the LMPC and require proper tuning, hence, the tuning guidelines discussed in [6] is followed. Lastly, we choose $V(x) = x^T P x$ as the Lyapunov function, where P is a positive definite matrix obtained by applying grid search.

Under the LMPC, we first perform the closed-loop simulation using the first-principles process model of Eq. 3.7 - 3.8 integrated by explicit Euler method with times step $h_c = 0.05min$, and the

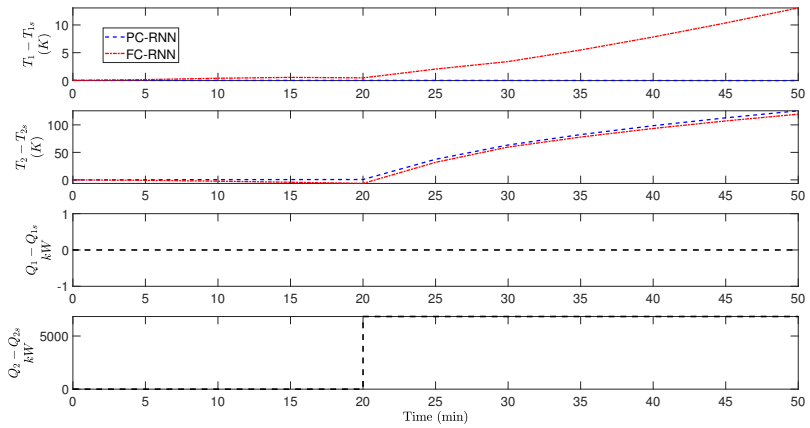

Figure 3.9: Open-loop simulation under step change in $(Q_2 - Q_{2s})$ of the two RNN models: partially-connected RNN (denoted by PC-RNN in dashed line), and the fully-connected RNN (denoted by FC-RNN in dash-dotted line).

Table 3.3: MSE comparison of the open-loop prediction results between the RNN models and the Aspen Plus simulation model.

	Partially-connected RNN	**Fully-connected RNN**
T_1	2.52×10^{-3}	6.93×10^{-1}
C_{E_1}	6.626×10^{-7}	5.51×10^{-5}
T_2	1.837×10^{-1}	1.723
C_{E_2}	1.996×10^{-2}	1.988×10^{-2}

results are shown in Fig. 3.10 - 3.11. From those figures, both LMPCs, each based on its predictive RNN model, are able to stabilize the process by driving the states close to the steady-state values. However, state trajectories resulting from the partially-connected RNN based LMPC are smoother and do not exhibit oscillation around the steady-state. This simulation is used to find the MPC parameters, such as parameters in the cost function, that would deliver a desired closed-loop performance. Furthermore, it is important to check the closed-loop state evolution before applying the proposed LMPC to the high-fidelity Aspen Plus Dynamics model.

Remark 3.6. *The MPI implements information exchange by defining a digital platform that allows access from the python-based MPC and the Aspen dynamic software. Specifically, a python script is developed to automatically upload the MPC output to the shared platform during the simulation.*

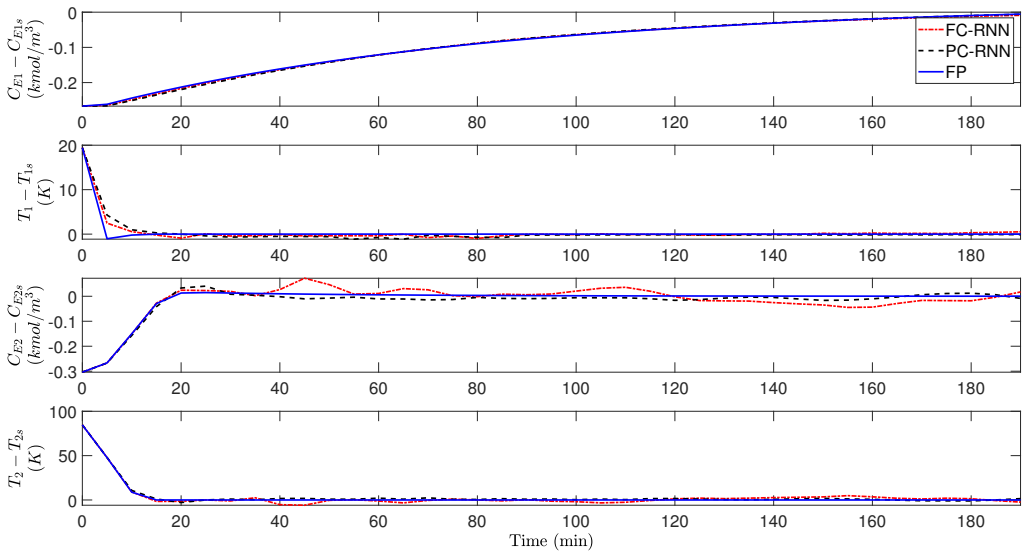

Figure 3.10: State profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

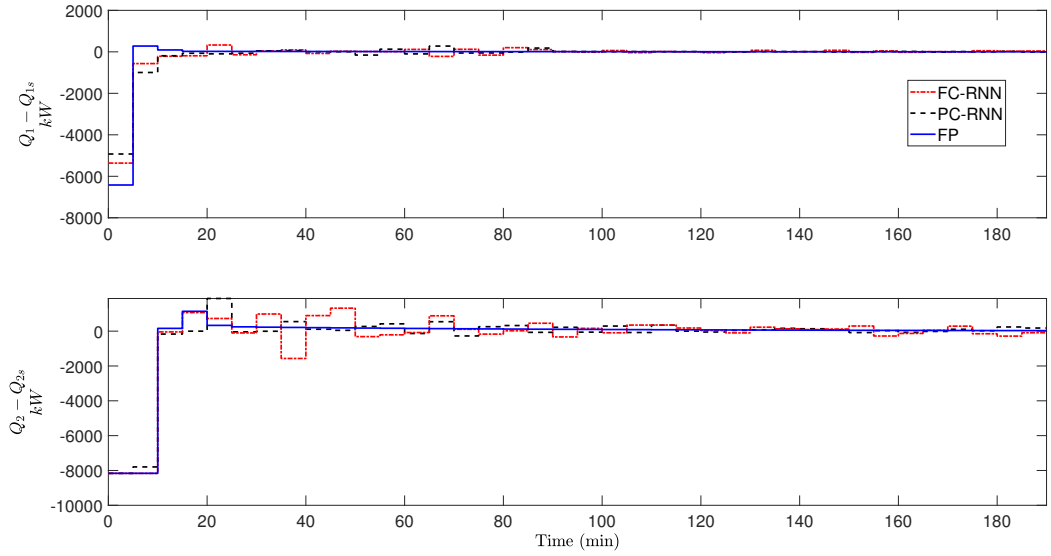

Figure 3.11: Input profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

Simultaneously, the Aspen dynamic simulation can search and find the correct input data to update its control states via the build-in script function. Subsequently, the Aspen dynamic simulation can send the real-time measurement to the MPC using the same scripts and platform to carry on the next iteration. Commercial platforms, such as Google Drive and Dropbox, can be efficient candidates to adopt for data exchanging if it is not necessary to construct a specified database for users' projects.

3.6.5 Closed-loop Simulation: Aspen Plus Dynamic Model

Finally, we carry out closed-loop simulations of the Aspen Plus Dynamic model under the proposed LMPCs, and the results are shown in Fig. 3.12 - 3.13. Both LMPCs stabilize the process at the steady-state exhibiting similar dynamic performance. The responses under the fully-connected RNN-based LMPC exhibits noticeable oscillation, while the ones under the LMPC that utilizes partially-connected RNN model are smoother. This is expected since the fully-connected RNN assumes that every input affects every possible output, which interferes with the prediction accuracy. We note that an ensemble of RNN models may be developed from the training date set and used to make average predictions of the future state evolution ([118,119]), but this approach was not pursued in the present work as the MPCs implemented using the developed RNN models produced desired closed-loop responses.

Another critical performance metric is the computational time needed to calculate the control action, which impacts the feasibility of the controller in real-time operation. To evaluate this we run closed-loop simulation, and record the computational time to solve for the optimum controller actions in each sampling period for the two different RNN-based LMPCs. The computational time ratio of the fully-connected RNN-LMPC to the partially-connected RNN-LMPC is presented in Fig. 3.14. As illustrated in Fig. 3.14, the fully-connected RNN-LMPC requires longer computational time to find the optimal solution in 32 out of the 39 calculations. On average, the computational times for the fully-connected RNN-LMPC and the partially-connected RNN-LMPC are 2.1161 and 1.65305 minutes, respectively. Furthermore, the computation times mean μ,

Table 3.4: Statistical analysis of the computational times (in minutes) needed to calculate the control actions using the two RNN structuers in LMPC.

	Partially-connected RNN	**Fully-connected RNN**
μ	1.65305	2.1161
σ	0.6185	0.87922
range $(min - max)$	$(0.38 - 3.77)$	$(1.21 - 4.8)$

standard deviation σ, and range using each RNN model are listed in Table 3.4. From the table, the three parameters indicate that the computational times for the fully-connected RNN-LMPC are larger than the partially-connected RNN-LMPC. In particular, using the partially-connected RNN structure in the LMPC, the overall computational time has been reduced by approximately 22%.

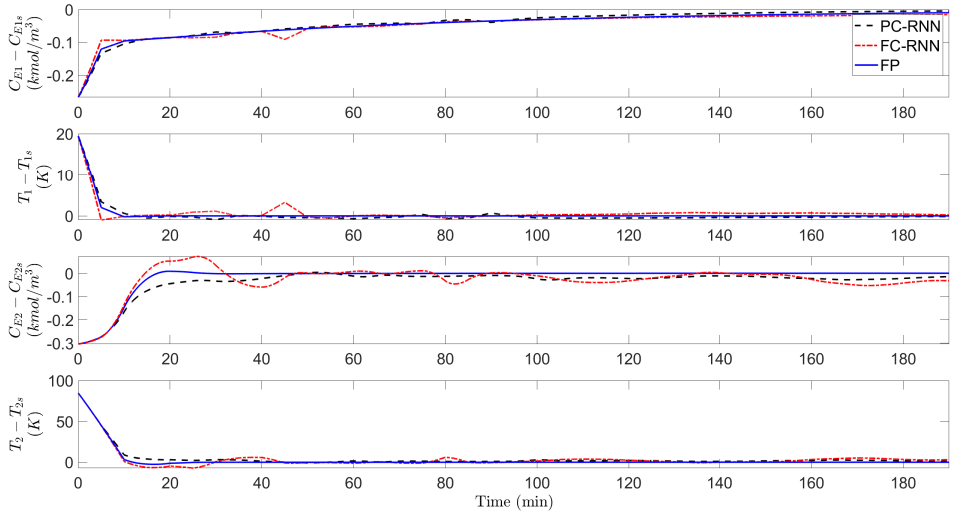

Figure 3.12: State profiles of the closed-loop simulation of the Aspen Plus Dynamics model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

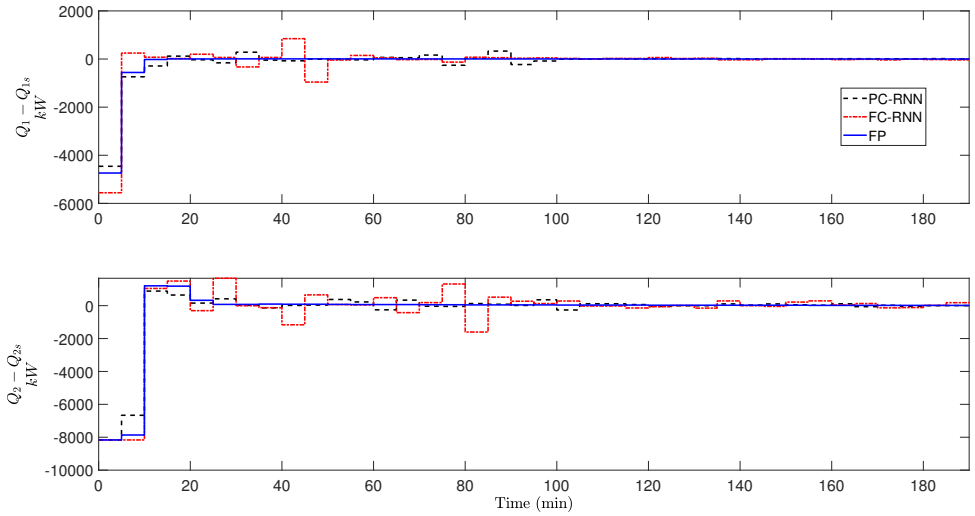

Figure 3.13: Input profiles of the closed-loop simulation of the Aspen Plus Dynamics model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

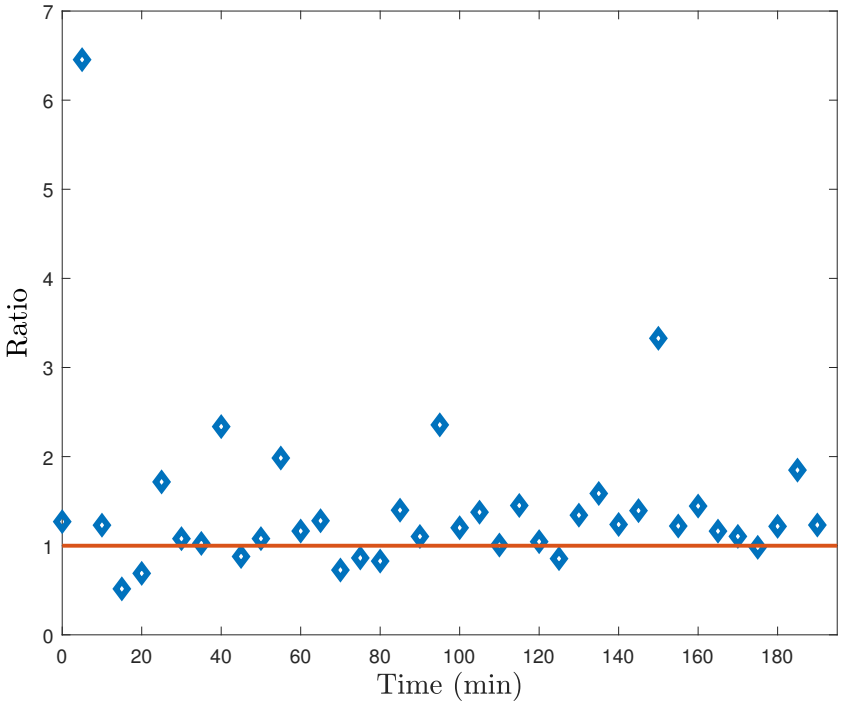

Figure 3.14: Ratio of computational time needed to calculate the control actions by LMPC using fully-connected RNN and partially-connected RNN at each samplint time Δ.

Chapter 4

Physics-informed Machine Learning Modeling for Predictive Control Using Noisy Data

4.1 Introduction

To mitigate the effect of noise in data sets during modeling, several methods have been proposed in the literature. For linear dynamical systems, the famous Kalman filtering is one approach for dealing with noisy measurements. Furthermore, other approaches have been proposed such as unscented Kalman filter and moving horizon estimation(e.g., [76]). An accurate explicit model representation is generally required in the state estimation technique in order to achieve a correct estimation, and also tuning of the co-variance matrices is required [60]. The result of learning using raw vibration signals generated from water flow system in a laboratory-scale was recently investigated by [87] using three different approaches; linear statistical learning approach, LSTM neural networks, and feed-forward neural network (FNN). According to their analyses, both the machine learning methods and the linear statistical model under-performed when utilizing raw vibration signals, and additional data treatment was required to enhance the developed models

performance.

Since machine learning approaches were originally developed in the field of computer science, they generally either presume the access to high-fidelity data, or refer to mislabeling in classification tasks and numerical fallacies in regression tasks as "noisy data" [42]. As a result, the accuracy and effectiveness of these methods are typically evaluated using noise-free data sets. However, finding and collecting noise-free data in the domain of science and engineering, especially chemical engineering, remains a long-standing challenge. Thus, numerous works in the literature investigate machine learning methods with various types of noise. For instance, in the work of [46], the robustness of an RNN model of Wiener type is evaluated by two types of noise: white noise and sinusoidal noise. Furthermore, a study by [52] investigated the impact of Gaussian noise on an RNN modeling of chaotic systems represented by short time-series. Moreover, data quality can also be improved by using data preparation and smoothing techniques. For foaming control implementation in bio-processes using ensemble-based machine learning method, noisy and repetitive data are filtered in the work of [2]. In the same vein, [64] conducted a data smoothing (i.e., pre-treatment) and dealt with incomplete data points by applying a third-order polynomial to the experimental data and then combined it with an ANN to create a deep reinforcement learning strategy to control a bio-reactor.

Many machine learning modeling algorithms can handle Gaussian noise. However, non-Gaussian noise can cause a degraded modeling performance between the input and the (noise-free) ground-truth output, and this is because of its over-fitting of the corrupted training data set's noisy behavior. In nonlinear processes modeling by machine learning algorithms exposed to industrial data noise with a non-Gaussian distribution, the work of [116] utilized Monte Carlo dropout and co-teaching. The Monte Carlo dropout strategy in the neural network training process is an excellent way to minimize over-fitting to non-Gaussian noisy input without requiring any a prior process knowledge. As for the co-teaching technique, it's essentially using noise-free data generated from a first-principles model to mitigate the impact of noise in the training phase of machine learning models. [1] used a training data set where 20% of the data set was noise-free to

improve the overall modeling performance using co-teaching method.

Standard RNN models, also known as fully-connected RNN models, are a popular option for analyzing time-series data within a black-box modeling framework. Such a modeling methodology, however, may not always be preferable, particularly for large chemical processes due to the complex interactions among process variables. Hence, to improve RNN performance, several studies(e.g., [48, 92, 129]) have looked into gray-box modeling, also referred to as hybrid modeling, which involves integration of a prior physical knowledge and expertise into the modeling of neural networks. Another method is to reflect physical relation among the given process inputs and outputs into the modeling of neural networks (i.e., known as partially-connected modeling) which was proposed in [114]. In this direction, [5] examined the partially-connected method by comparatively investigating open-loop and closed-loop simulation utilizing fully-connected RNN model against partially-connected RNN model on a large-scale complex chemical process modeled in Aspen Plus Dynamics. It was demonstrated that a partially-connected RNN model outperformed a fully-connected RNN model in terms of smoother state trajectories and lower computational burden under the MPC controller. Yet, to our knowledge, the performance of partially-connected RNN has not been studied in the presence of industrial noise.

Taking into consideration the preceding factors, the current study takes noise into account and attempts to evaluate the performance of a partially-connected RNN-based MPC through application on a large-scale nonlinear chemical process. Initially, we use the process simulators Aspen Plus and Aspen Plus Dynamics, to create a simulation model of a chemical plant that produces Ethylbenzene via two continuous stirred tank reactors (CSTR) in series. Then, we carry out extensive open-loop simulations using Aspen dynamical model to construct a base data set, which will be corrupted with two types of noise (i.e., Gaussian and non-Gaussian) to obtain two separate data sets based on the noise type. Subsequently, we train a standard partially-connected RNN model as described in [5] using the noisy data. Then, two other models are developed by employing the co-teaching and Monte Carlo dropout techniques, respectively. Eventually, we evaluate both open-loop and closed-loop performance of each model, and show the benefits

of using the two proposed approaches to overcome noisy data presence in a partially-connected RNN-based MPC framework.

The remainder of this manuscript is structured as follows: In Section 4.2, the class of nonlinear chemical process systems, mathematical notation, and stabilizing feedback control law assumptions are discussed. Next, the conceptualization and the development of partially-connected RNN models, and LSTM models are introduced in Section 4.3. Subsequently, Sections 4.4 and 4.5 introduce the concepts of co-teaching and Monte Carlo dropout techniques, respectively. The incorporation of a partially-connected RNN model into a model predictive controller with Lyapunov stability assumption is proposed and discussed in Section 4.5. In Section 4.7, the open-loop as well as the closed-loop performances of MPCs using different RNN models that underwent different training procedures are assessed utilizing chemical process application modeled in Aspen Plus Dynamics simulator.

4.2 Preliminaries

4.2.1 Notations

Throughout this work, the Euclidean norm of a vector κ is represented by the notation $|\kappa|_2$. The standard Lie derivative is notated by $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$. Set subtraction operator used in this work is " $-$ ", as in $A - B = \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is regarded as of class C^1 if it is continuously differentiable in its domain.

4.2.2 Class of systems

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

$$\dot{x} = F(x,u) := f(x) + g(x)u, \quad x(t_o) = x_o \tag{4.1a}$$

$$y = x + \nu \tag{4.1b}$$

where $x = [x_1, ..., x_n]^T \in \mathbf{R}^n$ denotes the state vector, and the vector of manipulated variables (i.e., inputs) is denoted by $u = [u_1, ..., u_r]^T \in \mathbf{R}^r$. The term $y = [y_1, ..., y_n]^T \in \mathbf{R}^n$ represents the vector of state measurements that are continuously sampled, and the measurements noise vector is denoted by $\nu \in \mathbf{R}^n$. The input vector u is bounded by a lower bound $u^{\min}$ and an upper bond $u^{\max}$ and both are $\in \mathbf{R}^r$. $f(\cdot)$ is a vector function $\in \mathbf{R}^{n\times 1}$ and $g(\cdot)$ is a matrix function $\in \mathbf{R}^{n\times r}$, and both are assumed to be adequately smooth. We assume that the entire state vector is in deviation form from the steady state of the considered system, such that when $\nu(t) = 0$ and the function $F(0,0)$ is equal to zero then the origin is a steady-state of the nominal system in Eq. 4.1, i.e., $(x_s, u_s) = (0,0)$, where the subscript "s" indicates the steady-state.

4.2.3 Stabilizability Assumption

For closed-loop stability considerations, a stabilizing feedback controller $u = \Phi(x) \in U$, where $U := \{u^{\min} \leq u \leq u^{\max}\}$ is assumed to exist. This controller is assumed to be able to enforce the steady state (i.e., the origin) of the system of Eq. 4.1 to be exponentially stable in a neighborhood around the origin. Such an assumption implies that a control Lyapunov function of class C^1 exists and is represented by $V(x)$, such that for all x in an open neighborhood D around the origin, the following inequalities hold:

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

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

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

where c_i are positive real numbers $\forall\ i \in \{1,...,4\}$. There are several methods to construct the controller $\Phi(x)$; for instance, a possible method is the universal Sontag's control law ([61]). Other methods can also be applied, such as finding a well-tuned proportional control law (i.e., P-controller). Once the controller is chosen, subsequently, following [118] the closed-loop stability region Ω_ρ is defined to be a level set of $V(x)$ within the region D that is characterized under the controller $u = \Phi(x) \in U$, i.e., $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\ , \rho > 0\}$.

4.3 Recurrent neural networks model (RNN)

RNN models, as noted in the introduction, are an effective tool for modeling time-series data. The hidden layer neurons' recursive action enables the RNN to retain the memory of earlier states, allowing it to adequately mimic a time-series dataset behaviour. RNN models are utilized to estimate the nonlinear system of Eq. 4.1 using process operational data in this study is represented as follows:

$$\dot{\bar{x}} = F_{rnn}(\bar{x}, u) := A\bar{x} + \Theta^T\ \gamma \tag{4.3}$$

where the RNN state vector is $\bar{x} = [\bar{x}_1, ..., \bar{x}_n]$, and $u = [u_1, ..., u_r]$ is the vector of the manipulated inputs. The vector γ is based on $\bar{x}$ and u, and is defined as $[\gamma_1, ..., \gamma_n, \gamma_{n+1}, ..., \gamma_{n+r}]$=$[\alpha(\bar{x}_1), ...,$ $\alpha(\bar{x}_n), u_1, ..., u_r] \in \mathbf{R}^{n+r}$. The notation $\alpha(\cdot)$ denotes the nonlinear activation function utilized for the activation of the hidden layers. Such activation functions include the sigmoid function and the hyperbolic tangent function. The diagonal matrix $A = diag[-a_1, ..., -a_n]$ consists of negative coefficients with each $a_i > 0$ with the intention that the states are kept stable in the sense of bounded-input-bounded-state stability. Regarding the matrix $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(n+r)\times n}$, it consists of a vector $\theta_i = b_i[w_{i1}, ..., w_{i(n+r)}]$, where the elements of each θ_i are b_i, which are constants, and w_{ij}, which is the weight on the interrelation that links the jth input to the ith neuron, where $j \in \{1, ..., (n+r)\}$, and $i \in \{1, ..., n\}$.

4.3.1 Partially-connected RNN

For constructing a dynamic model for a nonlinear process, a neural network model that takes all available process inputs and predicts the desired outputs is usually preferred. Using open-source machine learning software, a dynamic RNN model for such processes may be simply developed, and this model will be able to account for all conceivable correlations between each input and output of the underlying process. The general construction of such a fully-connected RNN with an input layer, hidden layers, and an output layer is depicted on the left side of Fig. 5.3. As a result, fully-connected RNN models are typically the best initial choice for modeling processes where no prior knowledge is available.

The term "process prior knowledge" is defined as a physical understanding of the process considered that exists before the derivation of the first-principles process model. It involves, but is not restricted to, the model's intended goal, each and every hard and soft physical constraint imposed on the process as a result of design considerations, as well as the process structure. We focus on process structure knowledge in the form of connections between process input and output variables in this paper. Physical connections between process variables can sometimes be straightforward, especially in the chemical process industry. Upstream processes, for example, have an effect on downstream processes, whereas the opposite effect may not exist. This relationship among upstream and downstream stages is frequently reflected explicitly in a first-principles mathematical model. As a result, as discussed in [94], integrating process knowledge that defines physical relationships among the considered system's inputs and outputs into neural network modeling enhance its performance. This modeling approach is referred to as partially-connected neural network modeling.

We consider the system with $x = [x_a, x_b]$ and $u = [u_a, u_b]$, such that $x_a \in \mathbf{R}^{n_1}$, $x_b \in \mathbf{R}^{n_2}$, $u_a \in \mathbf{R}^{r_1}$, and $u_b \in \mathbf{R}^{r_2}$ where $r_1 + r_2 = r$ and $n_1 + n_2 = n$. It is assumed that the input vector u_a exclusively influences the state vector x_a, whereas x_b is influenced by both u_a and u_b. Physical knowledge is incorporated into the RNN modeling of the nominal system by modifying the RNN structure to explicitly disconnect the links between u_b and x_a. In due course, an enhanced

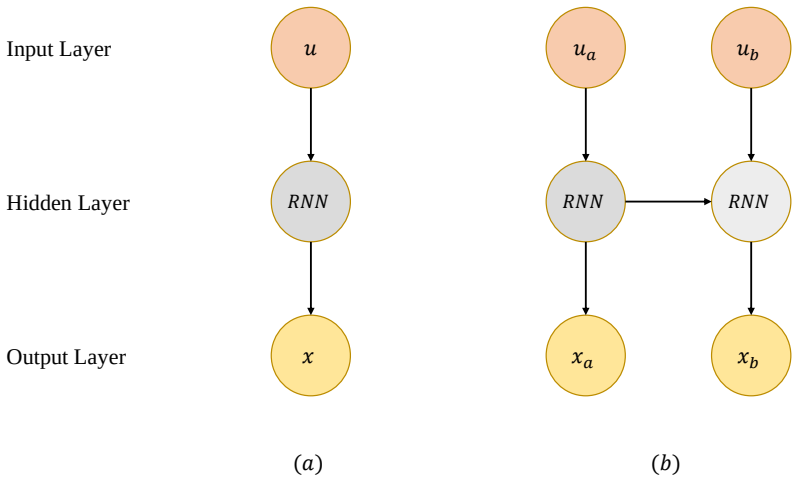

Figure 4.1: Schematic of (a) Standard and (b) Physics-informed RNN structures, with $u = [u_a, u_b]$ and $x = [x_a, x_b]$.

model approximation is achieved as a result of implementing partially-connected RNN model as discussed and demonstrated in [5] and [114].

The development framework for partially-connected RNN models is similar to that of fully-connected RNN models, but with additional specifications. [5] discussed the construction and training processes of partially-connected RNN models and provided the pseudocode of these modeling methods. Basically, a training/validation dataset can be collected from experimental and industrial sources, or through extensive open-loop simulations. The general principle of splitting the collected data set into 70% training and 30% validating can be followed, or more advanced techniques such as cross-validation may be employed. Prior to training RNN models, the input vectors u_a and u_b, as well as the output vectors x_a and x_b, should be defined, which is also referred to as data preprocessing.

To build and train the RNN models in this study, we used the Keras library, which is an open source Python library. Our models are consist of four layers: an input layer, two hidden layers, and an output layer, where the hidden neurons are activated by nonlinear functions: hyperbolic tangent

and sigmoid functions. Instead of feeding the full input vector u through one input layer, in partially connected RNN models the input vectors u_a and u_b are fed independently through different input layers according to the process structure, as illustrated in Fig. 5.3. The first hidden layer predicts the output vector x_a based on the input vector u_a. Subsequently, u_b and x_a are concatenated and then sent to the subsequent hidden layer to estimate x_b. Ultimately, the developed model will be able to estimate both the output vectors x_a and x_b given u_a and u_b.

4.3.2 Long short term memory (LSTM)

Long-short-term memory networks, or LSTMs in short, are a class of RNNs, and henceforth will be referred to as RNN-LSTM. Due to the design of three gates in the network structure, namely the input gate, the forget gate, and the output gate, RNN-LSTM networks are capable of capturing long-term dependencies in problems involving sequential prediction. Figure 4.2 shows a diagram of an LSTM network. In this study, RNN-LSTM based models are developed in the framework of partially-connected modeling. When given control actions and previous noisy state measurements, the generated RNN-LSTM models are used to predict the states of Eq. 4.1. In particular, considering the input sequence $m(k)$, where $k = 1,...,T$ and T denotes the number of measured states of the nominal system in Eq. 4.1, the approximate output sequence $\bar{x}(k)$ is computed using the following equation:

$$i(k) = \sigma(\omega_i^m m(k) + \omega_i^h h(k-1) + b_i) \tag{4.4a}$$

$$f(k) = \sigma(\omega_f^m m(k) + \omega_f^h h(k-1) + b_f) \tag{4.4b}$$

$$c(k) = f(k)c(k-1) + i(k)\tanh(\omega_c^m m(k) + \omega_c^h h(k-1) + b_c) \tag{4.4c}$$

$$o(k) = \sigma(\omega_o^m m(k) + \omega_o^h h(k-1) + b_o) \tag{4.4d}$$

$$h(k) = o(k)\tanh(c(k)) \tag{4.4e}$$

$$\bar{x}(k) = \omega_y h(k) + b_y \tag{4.4f}$$

where $m(k)$ denotes the input sequence, while $c(k)$ and $h(k)$ are the cell state and the internal

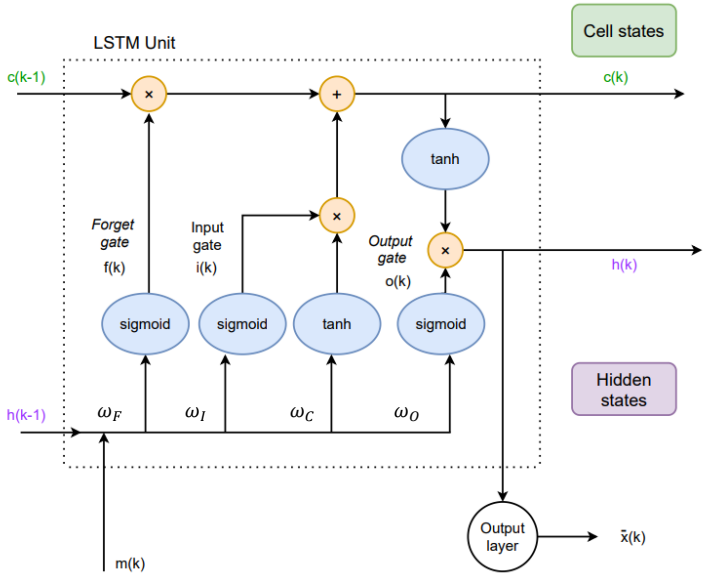

Figure 4.2: RNN-LSTM network schematic.

state, respectively. Regarding the gates, $f(k)$ is the forget gate, $o(k)$ represents the output gate, and the output of the input gate is denoted by $i(k)$. The output sequences $\bar{x} \in \mathbf{R}^{n \times T}$ is the RNN-LSTM network output.

The weight matrices here are given by ω_p^l, where $p \in \{i, c, f, o\}$ represents the gate or the state, while l represents the associated vector (i.e., either m or h). Similarly, the bias term is denoted by b_p. Eventually, Eq. 4.4f is used to compute the predicted state of the RNN-LSTM, which the terms b_y and ω_y represent the bias vector and the output weight matrix, respectively. Because the RNN-LSTM model predicts future states using control actions and previous state measurements, the input sequence $m \in \mathbf{R}^{(n+r) \times T}$ consists of manipulated input $u \in \mathbf{R}^r$ along with previous measured states $x \in \mathbf{R}^n$ during a given time period (i.e., T). The nonlinear activation functions used in the LSTM model are $\sigma(\cdot)$ and $\tanh(\cdot)$, which are sigmoid and hyperbolic tangent functions, respectively.

4.4 Co-Teaching technique

Noisy data in classification problems could result in mislabelling (e.g., an image with a label "A" is mislabeled as "B"), and for regression problems noise in the data might result in a deviation from its ground truth value (i.e., the true value). In both cases, it is quite challenging for a machine learning model to fulfill the desired model accuracy with a noisy data set following the standard learning algorithms.

The co-teaching technique was originally proposed to improve the accuracy of ML-based models in image classification problems when the training data set is corrupted with noise ([42]). [1] utilized co-teaching in the context of regression to handle noisy data with an observed enhancement in model accuracy. In the same vein, to reduce the effect of noisy data on RNN-LSTM modeling, [116] employed the co-teaching method, and achieved better closed-loop performance under MPC when applied to a reactor example in comparison with the standard RNN-LSTM model. To our knowledge, little attention has been paid to the implementation of the co-teaching method in solving regression problems. Hence, there is growing research into extending the co-teaching technique to regression problems using RNN-LSTM networks.

The idea behind the co-teaching technique comes from the fact that earlier in the training process, neural networks would employ a simple pattern to fit training data [42]. As a result, when evaluating the loss function value under a simple pattern that approximates the relationship between inputs and outputs of a neural network, noisy data typically have a large loss function value, and noise-free data typically have a small value. Therefore, merging the two data sets provides a possible way to train machine learning models in the presence of noisy data by benefiting from noise-free data. Such clean data can be generated from simulation of first-principles models. Therefore, mixing the two data sets in some ratio (i.e., noisy data set and noise-free data set) can improve the robustness of the RNN-LSTM model training process to over-fitting to noisy data. To apply this technique, after merging the two data sets (i.e., the noisy and the clean data) into one data set, then the new data set is to be shuffled first and then split into training and validating sets as a prerequisite for machine learning based model

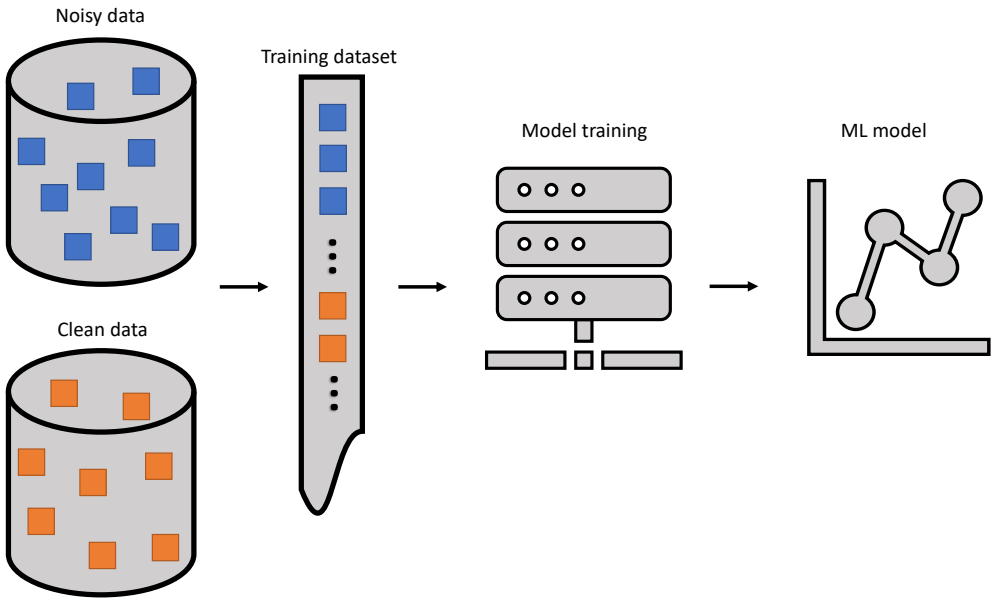

Figure 4.3: Developing machine learning (ML) model via co-teaching method.

development. Subsequently, both partially-connected and fully-connected RNN models can be developed following the procedure proposed in [114] and [5].

4.5 Dropout technique

Another candidate strategy to reduce over-fitting caused by noisy data, specifically non-Gaussian type [116], without having any prior process knowledge is to utilize a dropout technique in the neural network development process. In particular, as depicted in Fig. 4.4, a dropout strategy arbitrarily drops the associations between units in neighboring layers during training and provides an efficient method for combining various neural network structures to improve prediction accuracy.

Consider the RNN-LSTM model in its general form as in Eq. 4.3. For all RNN-LSTM layers, let $W = \{W_1, ...W_L\}$ denote the set of weight matrices that incorporate both weights and bias terms, where W_1 denotes the weight matrix associated with the first layer in the RNN-LSTM structure and L denotes the number of layers. The primary objective of the Monte Carlo (MC) dropout technique

is to acquire the posterior distribution (i.e., an integration of the prior distribution as well as the likelihood function, which reveals what information the observed data contain) of the RNN-LSTM weights W, denoted by $p(W)$, from the training data (M,X) in which M and X are the input and output data matrices, respectively. The weight matrix W_i is described as follows, following the method in [35]:

$$W_i = B_i \cdot Z_i \tag{4.5a}$$

where $Z_i = diag(z_i)$ and $i = 1,...,L$. Each z_i is a set of binary variables that follows the well-known Bernoulli distribution and symbolizes the weights which are dropped out according to a particular user-defined probability, and B_i denotes the variational variables that shall be optimized. The RNN-LSTM predicted output distribution can be estimated by conducting Monte Carlo dropout at testing time after the RNN-LSTM model has been trained using the Monte Carlo dropout technique. Several realizations of the RNN-LSTM model are used to obtain the predicted distribution by taking the mean of the distribution as follows:

$$p(x^*|m^*,M,X) \approx \frac{1}{N_t}\sum_{i=1}^{N_t} p(x^*|m^*,W_i) \tag{4.6a}$$

where the RNN-LSTM input and output are m^* and x^* in the test set, respectively. The total number of Monte Carlo realizations at the testing stage is N_t. Given the same input, the RNN-LSTM output is no longer deterministic, because the RNN-LSTM model obtained utilizing the MC dropout technique is a probabilistic model, which is approximated by its mean value in this work, as shown in Eq. 4.6. As a result of using Eq. 4.6, a probabilistic distribution is obtained as an approximation of the model predictions' uncertainty. Furthermore, the ground-truth process dynamics can be roughly evaluated via the sample mean of all the model estimations.

The dropout rate (i.e., the likelihood of a neuron to be dropped out) plays a critical role in the model's performance, and is determined during model training to accomplish desired training/validation results. In particular, a small dropout value (e.g., 0.2–0.5) is suggested to begin

with. Then, the user may increase the dropout rate value if over-fitting still occurs, or to decrease the value if the network fails to capture (i.e., learn) the dynamics of the underlying process.

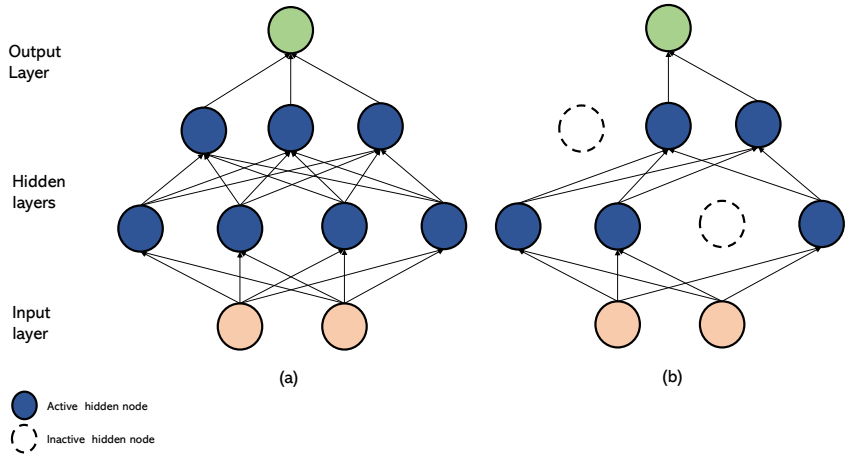

Figure 4.4: Fully-connected neural network layers (a) without dropout and (b) with dropout.

Remark 4.1. *Since the considered training data is corrupted with noise, our perception of the process dynamics via machine learning-based models utilizing the dropout strategy is expressed in a probabilistic manner. The Monte Carlo dropout method is an effective tool for modeling uncertain process dynamics and to estimate underlying nominal dynamics by a sequence of probabilistic forward passes. In particular, the RNN-LSTM model, developed via the Monte Carlo dropout technique, can be considered an uncertain process model, with the RNN-LSTM weights as uncertain variables.*

Remark 4.2. *The Monte Carlo dropout technique can be used to address over-fitting issues for different tasks with regards to chemical processes. For example, it can be employed to improve the machine learning-based model accuracy for state estimation, fault diagnosis, and other tasks in the presence of process noise, measurements noise, and other uncertainties associated with the process of interest. Furthermore, this technique may not only be used for regression but can also be applied to classification problems. The interested reader may refer to [35], [36], [91], [110], and [56] for details.*

4.6 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 modelling of RNN-LSTM is executed as discussed in [5] and then utilized as a predictive model to provide state estimation to solve the optimization problem of the LMPC, which is expressed in the following form:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_k+P} L(\tilde{x}(t), u(t))dt \tag{4.7a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{rnn}(\tilde{x}(t), u(t)) \tag{4.7b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_k + P) \tag{4.7c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{4.7d}$$

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

$$\text{if } x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}} \tag{4.7e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_k + P), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{4.7f}$$

where $S(\Delta)$ 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., $P = N \times \Delta$, $N > 0$). The time-derivative of the Lyapunov function V in Eq. 4.7e is given as $\dot{V}(x,u)$, i.e., $\frac{\partial V(x)}{\partial x}(F_{rnn}(x,u))$. During the prediction horizon $t \in [t_k, t_k + P)$, 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. 4.7a, which represents the cost function over the prediction horizon while satisfying the constraints of Eqs. 4.7b–4.7f. The RNN-LSTM model from Eq. 4.7b 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. 4.7d, where $x(t_k)$ is the last state measurement. The input constraints are expressed in Eq. 4.7c,

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. 4.7e is triggered. As a result of this constraint, the Lyapunov function of the closed-loop states 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. Following section 2.3, the controller $\Phi_{nn}(x)$ was developed with the intent of ensuring that the origin of the RNN-LSTM system of Eq. 4.3 exponentially stable.

When using noise-free data for training, a well-conditioned RNN-LSTM model can be generated with adequate model accuracy. Therefore, by using the RNN-LSTM based LMPC of Eq. 4.7 to control a nonlinear system as that of Eq. 4.1, the closed-loop state is assured to be bounded within the stability region Ω_ρ throughout the simulation time and eventually will converge to a small region around the origin under the condition that the modeling error, i.e., $|\nu| = |F(x,u) - F_{rnn}(\bar{x},u)|$, is sufficiently small [8, 118].

4.7 Application to a Chemical Process Using Aspen Plus Simulator

In this section, we evaluate the proposed partially-connected RNN-based LMPC using a large-scale chemical process in the presence of industrial noise. First, we use Aspen Plus Dynamics V11 to create a dynamic model of a chemical process. Following that, a time-series data set of the process states and input variables is generated in order to train and test the RNN-LSTM models using extensive open-loop simulation. Subsequently, open-loop and closed-loop simulations using an RNN-model-based MPC are performed and discussed. Henceforth, PCRNN-LSTM will stand for partially-connected RNN-LSTM.

4.7.1 Process description

The process of producing Ethylbenzene (EB) from Ethylene (E) and Benzene (B) as reactive raw materials is used to demonstrate the performance of the proposed control strategy. There are three reactions in this process, which take place in two nonisothermal and well-mixed continuous stirred tank reactors (CSTR). The reaction scheme is shown below:

$$C_2H_4 + C_6H_6 \rightarrow C_8H_{10} \quad \text{(primary)} \tag{4.8a}$$

$$C_2H_4 + C_8H_{10} \rightarrow C_{10}H_{14} \tag{4.8b}$$

$$C_6H_6 + C_{10}H_{14} \rightarrow 2\,C_8H_{10} \tag{4.8c}$$

where the desired reaction is the second-order, exothermic, irreversible reaction labeled as "primary".

Table 4.1: Parameter values, steady-state values, and model configuration of the Aspen Plus model.

$T_{1_o} = 350\ K$	$T_{1_s} = 310.523\ K$
$T_{2_o} = 350\ K$	$T_{2_s} = 430.542\ K$
$F_1 = 43.2\ m^3/hr$	$F_2 = 91.079\ m^3/hr$
$C_{E_1} = 4.2455\ kmol/m^3$	$C_{E_2} = 0.3254\ kmol/m^3$
$C_{B_1} = 5.3532\ kmol/m^3$	$C_{B_2} = 1.3841\ kmol/m^3$
$C_{EB_1} = 0.1854\ kmol/m^3$	$C_{EB_2} = 3.8744\ kmol/m^3$
$C_{DEB_1} = 9.1426 \times 10^{-7}\ kmol/m^3$	$C_{DEB_2} = 0.0058\ kmol/m^3$
Heat transfer option	*Dynamics*
Medium temperature	$298\ K$
Temperature approach	$77.33\ K$
Heat capacity of coolant	$4200\ J/kgK$
Medium holdup	$1000\ kg$
$C_p = 2.411\ kJ/kgK$	$\rho_1 = 639.1530\ kg/m^3$
$V_1 = V_2 = 60\ m^3$	$\rho_2 = 607.5040\ kg/m^3$

The first-principles model for the two CSTRs is derived using mass and energy balances. Particularly, the dynamic model of the two reactors is given by the following system of ordinary

differential equations (ODEs):

$$\frac{dC_{E_1}}{dt} = \frac{F_1 C_{Eo_1} - F_{out_1} C_{E_1}}{V_1} - r_{1,1} - r_{1,2} \tag{4.9a}$$

$$\frac{dC_{B_1}}{dt} = \frac{F_1 C_{Bo_1} - F_{out_1} C_{B_1}}{V_1} - r_{1,1} - r_{1,3} \tag{4.9b}$$

$$\frac{dC_{EB_1}}{dt} = \frac{-F_{out_1} C_{EB_1}}{V_1} + r_{1,1} - r_{1,2} + 2r_{1,3} \tag{4.9c}$$

$$\frac{dC_{DEB_1}}{dt} = \frac{-F_{out_1} C_{DEB_1}}{V_1} + r_{1,2} - r_{1,3} \tag{4.9d}$$

$$\frac{dT_1}{dt} = \frac{(T_{0_1} F_1 - T_1 F_{out_1})}{V_1} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_1 C_p} r_{1,j} + \frac{Q_1}{\rho_1 C_p V_1} \tag{4.9e}$$

$$\frac{dC_{E_2}}{dt} = \frac{F_2 C_{Eo_2} + F_{out_1} C_{E_1} - F_{out_2} C_{E_1}}{V_2} - r_{2,1} - r_{2,2} \tag{4.9f}$$

$$\frac{dC_{B_2}}{dt} = \frac{F_2 C_{Bo_2} + F_{out_1} C_{B_1} - F_{out_2} C_{B_1}}{V_2} - r_{2,1} - r_{2,3} \tag{4.9g}$$

$$\frac{dC_{EB_2}}{dt} = \frac{F_{out_1} C_{EB_1} - F_{out_2} C_{EB_2}}{V_2} + r_{2,1} - r_{2,2} + 2r_{2,3} \tag{4.9h}$$

$$\frac{dC_{DEB_2}}{dt} = \frac{F_{out_1} C_{DEB_1} - F_{out_2} C_{DEB_2}}{V_2} + r_{2,2} - r_{2,3} \tag{4.9i}$$

$$\frac{dT_2}{dt} = \frac{(T_{0_2} F_2 + T_1 F_{out_1} - T_2 F_{out_2})}{V_2} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_2 C_p} r_{2,j} + \frac{Q_2}{\rho_2 C_p V_2} \tag{4.9j}$$

where the reaction rates are calculated by the following expressions:

$$r_{i,1} = k_1 e^{\frac{-E_1}{RT_i}} C_{E_i} C_{B_i} \tag{4.10a}$$

$$r_{i,2} = k_2 e^{\frac{-E_2}{RT_i}} C_{EB_i} C_{E_i}, \qquad i = 1,2 \text{ (reactor index)} \tag{4.10b}$$

$$r_{i,3} = k_3 e^{\frac{-E_3}{RT_i}} C_{DEB_i} C_{B_i} \tag{4.10c}$$

In this work, the two CSTRs are connected in series, so that the output of the first reactor affects the output of the second one, but not vice versa. Furthermore, the model of this process is created with Aspen Plus and Aspen Plus Dynamics V11, which are high-fidelity simulators that are used for steady-state and/or dynamics of complex chemical processes modeling. The process model is initially built in Aspen Plus, where steady-state simulation is carried out and solved using material and energy balances. Following that, we use Aspen Plus Dynamics to run a dynamic simulation

of the underlying process to analyze and control its dynamical performance. The dynamic model is developed following the procedure described in [5], and the resulting flow sheet is shown in Fig. 4.5.

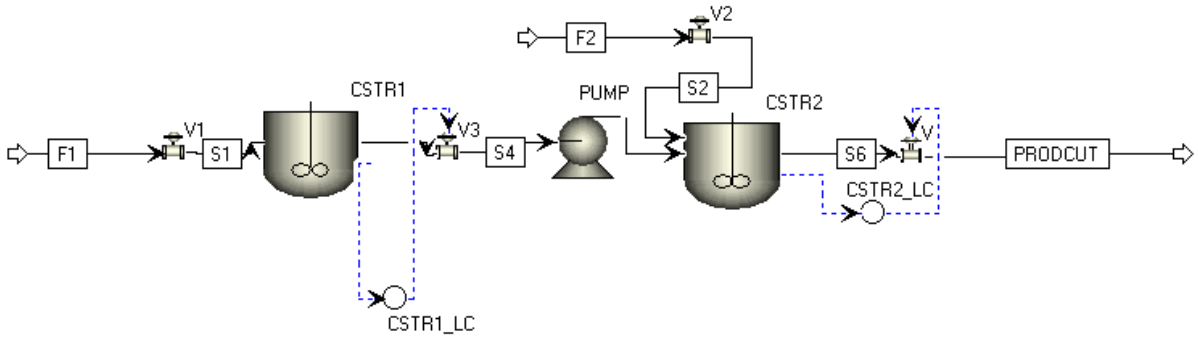

Figure 4.5: Aspen Plus model flow sheet of two chemical reactors in series.

The flow rates F_1 and F_2 are the raw material feed to the first and second reactors, respectively. The concentrations of the species considered in this process are given as: C_E, C_B, C_{EB}, and C_{DEB} and they represent Ethylene, Benzene, Ethylbenzene, and Di-Ethylbenzene, respectively. Process parameters such as reactor temperature, mass density, and liquid volume of each CSTR are denoted, in the same order, by T_i, ρ_i, V_i where i refers to the CSTR index $\in \{1,2\}$. The values of these parameters are listed in Table 4.1, which also includes the steady-state values and the liquid mixture's mass heat capacity, denoted as C_p which is considered constant in this work. The subscript "s" refers to steady-state value, and "o" represents the value at $t = t_o$. To control the reactors temperature, we added a cooling/heating jacket to each reactor that removes/provides heat to the reactor at a rate Q_i. The pressure is set to 15 bar initially for both reactors, while the temperatures of the first and second reactors at $t = t_o$ are set to $T_{1o} = 400K$ and $T_{2o} = 450K$, respectively. This choice of initial temperature, is made to keep both the reactants and products as liquids during the operation (i.e., simulation of the process). These values will be automatically adjusted by running Aspen Plus built-in steady-state simulations. After configuring the reactions in the two reactors, the steady-state simulation is carried on in order to analyze the behavior of the process. Moreover, the reactor geometry as well as thermodynamic parameters must be specified

before exporting the steady-state model to Aspen Plus Dynamics. In this study, the vessels in the Aspen model are all vertical, the heads are flat, and each CSTR is ten meters long. Table 4.1 contains a list of the thermodynamic parameters that were used in the Aspen model. Finally, after running a pressure check, the steady-state model is exported to Aspen Plus Dynamics to generate and initiate the dynamic model. Moreover, to prepare the model for both open-loop and closed-loop simulation the flow rates F_1 and F_2 are fixed, and the two reactors' heating modes are switched to constant duty to enable external control over the manipulated variables (i.e., Q_1 and Q_2). By following the procedure outlined above, the dynamical model of the considered process is completed.

4.7.2 Data generation and model training

Data are required for the development of data-driven models and, generally given that the data are independent and identically distributed, the larger the data set size, the more accurate the model can be [111]. However, excessive use of corrupted data (e.g., noisy, repeated, etc.) in the training data-set may lead to a less accurate RNN model, especially if the training is not carried out with potential pitfalls in mind. Therefore, this trade-off between data generation/collection and model accuracy should always be taken into account. In addition, large data sets are available from several sources, including industries, pilot plants, and computer-based simulations. However, in general, industrial data is not publicly accessible, and collecting data from pilot plants and laboratory experiments are expensive and time intensive. Therefore, in this work, we build our data set via extensive open-loop simulations utilizing an alternative approach

The constructed dynamical model in Aspen Plus Dynamics is used in open-loop simulations with the input signals randomly generated via MATLAB and then implemented in a sample-and-hold fashion, such that the input remains fixed throughout each sampling time Δ. A local message passing interface (MPI) is created to connect Aspen Plus Dynamics and MATLAB, so that the Aspen dynamical model can automatically read the input signals from MATLAB. The MATLAB code, in particular, generates the manipulated variables in deviation form in relation

to their steady-state values (i.e., $u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$). The two manipulated variables randomly vary within the lower bounds $[u_1^{\min}, u_2^{\min}] = [-1 \times 10^4 kW, -1.5 \times 10^4 kW]$ and the upper bounds $[u_1^{\max}, u_2^{\max}] = [1 \times 10^3 kW, 5 \times 10^3 kW]$. Both inputs are employed to the dynamic simulation in which the values are updated every five minutes (i.e., the sampling time). According to Aspen's manual, the default numerical integration method of the Aspen process dynamic model is the Implicit Euler scheme with an adaptive integration time step. In this study, we used a sampling time $\Delta = 5$ *min* i.e., the integration scheme records the process state values every 5 minutes in process operation time. Note that the numerical integration time step while adaptive is always several orders of magnitude smaller that the sampling time. Other integration techniques in Aspen that are available for users to choose from include the 4th-order Runge-Kutta method, the explicit Euler scheme, and Gear methods.

We employ Aspen dynamic simulations to generate data sets for neural network training, since the Aspen dynamic model can be regarded as a high-fidelity process model for various complex chemical processes such as a system of CSTRs. To simulate typical sensor variability in chemical plants, industrial noise is incorporated into the state measurements. The normalized data noise obtained from Aspen public domain is shown in Fig. 4.6. The probability distribution of the normalized industrial noise in Fig. 4.6 is shown in Fig. 4.7, from which we confirm that the industrial noise has a non-Gaussian distribution. With the random inputs generated from Matlab as discussed in the previous paragraph, and the normalized noise are amplified by six times and then added to the reactors temperature measurements. To create the training/validating data set, all input values and output states (such as T, C_A, and C_B) are recorded as sequential time series data. For the Gaussian noise case, we generated normalized white noise with zero mean and standard deviation of 0.1, which was amplified six times and added to the temperature measurements. The dataset generated via this procedure for the Gaussian and non-Gaussian cases are denoted as $S_{noisy(G)}(x,u)$ and $S_{noisy(NG)}(x,u)$, respectively.

For the co-teaching method, noise-free data is required for the ML-based model development. This noise-free data set is non-trivial to collect in practice (i.e., not readily available from the

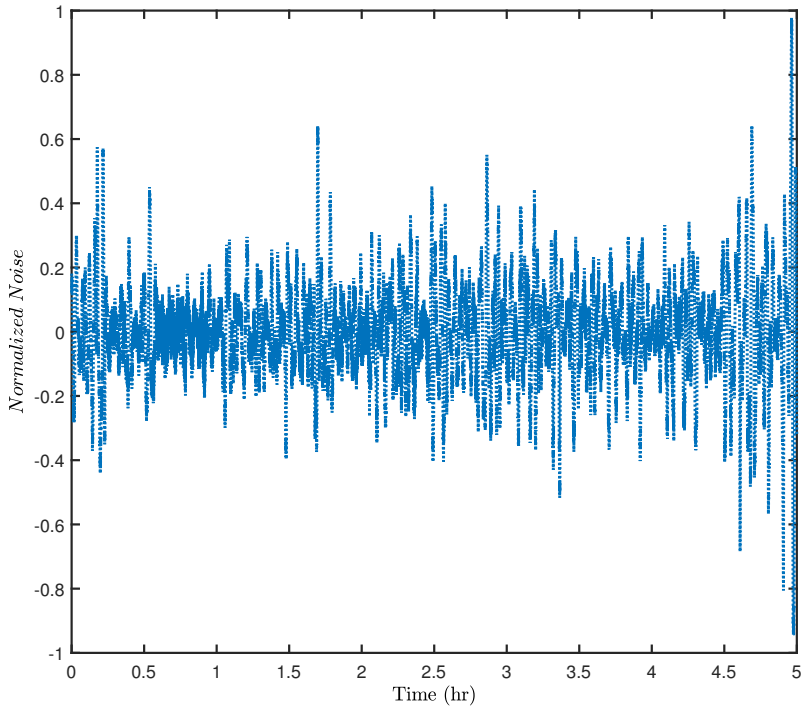

Figure 4.6: Normalized industrial noise from Aspen public domain data.

laboratory/plant). First-principles (FP) models with simplifying assumptions are developed as candidates for generating such clean data for training/validating processes to solve this challenge. Therefore, in this work, we numerically solved the FP model developed in the previous section to generate noise-free data $S_{FP}(x,u)$ to train the PCRNN-LSTM model using the co-teaching method. The noise-free open-loop trajectories of the Aspen dynamical model and the first-principles model denoted by FP under time-varying inputs are shown in Fig. 4.8. Although the first-principles model may not fully capture the dynamics of the Aspen model under the different operating conditions, we will show that the co-teaching method using noisy data from the Aspen model and noise-free data from solving the first-principles model can still improve the LSTM model's prediction accuracy.

The generated data sets are used in the development of three different RNN-LSTM models, as illustrated in Fig. 4.9. Both the standard RNN-LSTM model and the dropout RNN-LSTM model use either $S_{noisy(G)}$ or $S_{noisy(NG)}$ according to the noise type, yet we use the dropout rate while training the dropout RNN model. Basically, we perform a grid search for the dropout

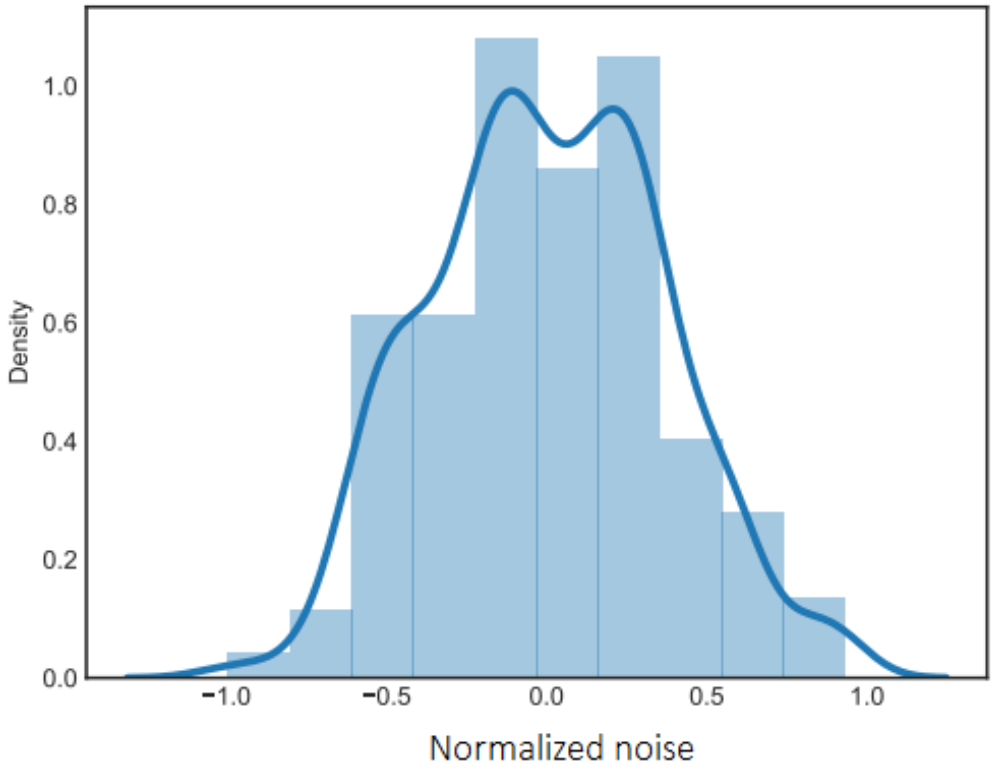

Figure 4.7: Probability density plot of normalized industrial noise introduced to the Aspen model.

rate in the range 0.2–0.5 that achieves the lowest training/validation loss (i.e., mean squared error (MSE)), and the results are summarized in Table 4.3. From the table, for the case of non-Gaussian noise and Gaussian noise, dropout rates of 0.2 and 0.4 are selected, respectively. As for the co-teaching model, it was trained typically as the standard model. Using the Keras library, the three RNN-LSTM models are constructed following the strategy discussed in section 4.3.1. The models are designed as follows: each neural network has two LSTM layers with thirty neurons in each, where we select the hyperbolic tangent functions (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) as the activation function. In addition, the output layer is to be activated by a linear activation function, and this layer will provide the estimated ten states, while the input layer will receive twelve inputs. The inputs to RNN-LSTM models are the states and the manipulated variables at t_k, where the model outputs are the states at $t = t_k + \Delta$, and all are defined in Table 4.2. Specifically, we predict the evolution of the states for the next five minutes (the equivalent of one sampling time) using input

data from the previous five-minute sampling period. Rather than the traditional gradient descent optimization algorithm, we employ the Adam optimizer, which is a hybrid of two algorithms: gradient descent with momentum and RMSprop. Furthermore, to generate more robust models, we perform a five-fold cross-validation on the RNN-LSTM models and choose the models with the lowest validation MSE.

Table 4.2: Input and output states of the RNN-LSTM models.

Notation	State (in deviation form)	
x_1	Concentration of Ethane	1^{st} CSTR
x_2	Concentration of Benzene	
x_3	Concentration of Ethylbenzene	
x_4	Concentration of Diethylbenzene	
x_5	Reactor's Temperature	
u_1	Heating/cooling duty	
x_6	Concentration of Ethane	2^{nd} CSTR
x_7	Concentration of Benzene	
x_8	Concentration of Ethylbenzene	
x_9	Concentration of Diethylbenzene	
x_{10}	Reactor's Temperature	
u_2	Heating/cooling duty	

After the development of the three RNN-LSTM models, we run two open-loop simulations considering the existence of Gaussian and non-Gaussian noise independently. Figures 4.10 and 4.11 illustrate the open-loop prediction of the three models, with the process output (i.e., the Aspen dynamical model output) denoted as the "True state", in response to time varying inputs generated randomly from MATLAB, and starting from the exact same initial conditions. Both figures demonstrate the improvement in the prediction accuracy of the RNN-LSTM models utilizing the two proposed methods; the dropout and the co-teaching methods. The open-loop prediction MSEs of the two scenarios are listed in Table 5.2, with noticeable improvements in the approximation of the process outputs, since both the dropout and the co-teaching RNN-LSTM provided relatively lower MSE values compared to the standard RNN-LSTM model.

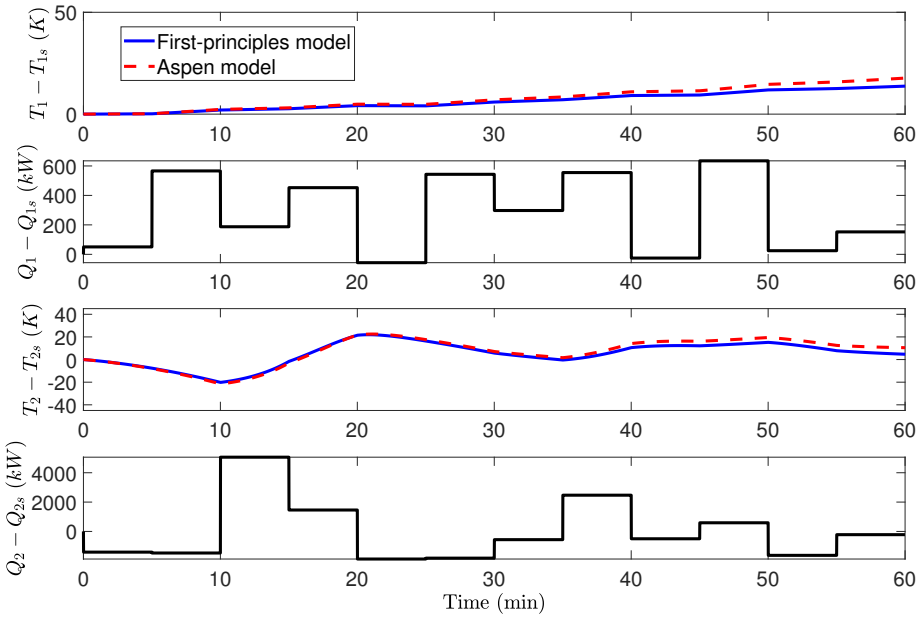

Figure 4.8: Open-loop state and manipulated inputs profiles for the process (noise-free).

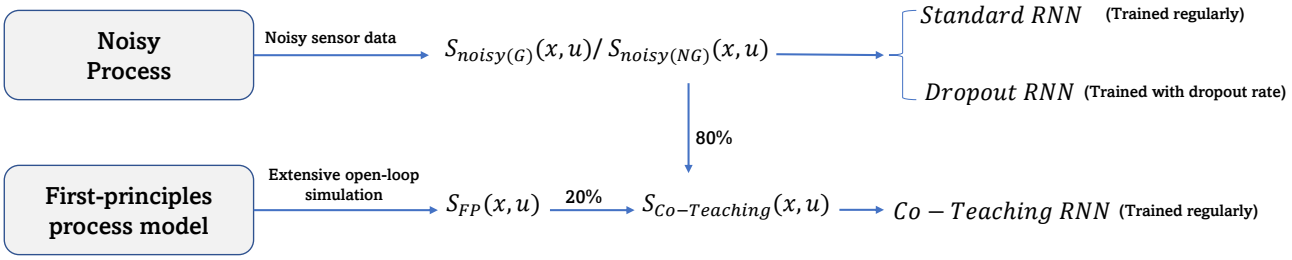

Figure 4.9: Partially-connected RNN model development methods.

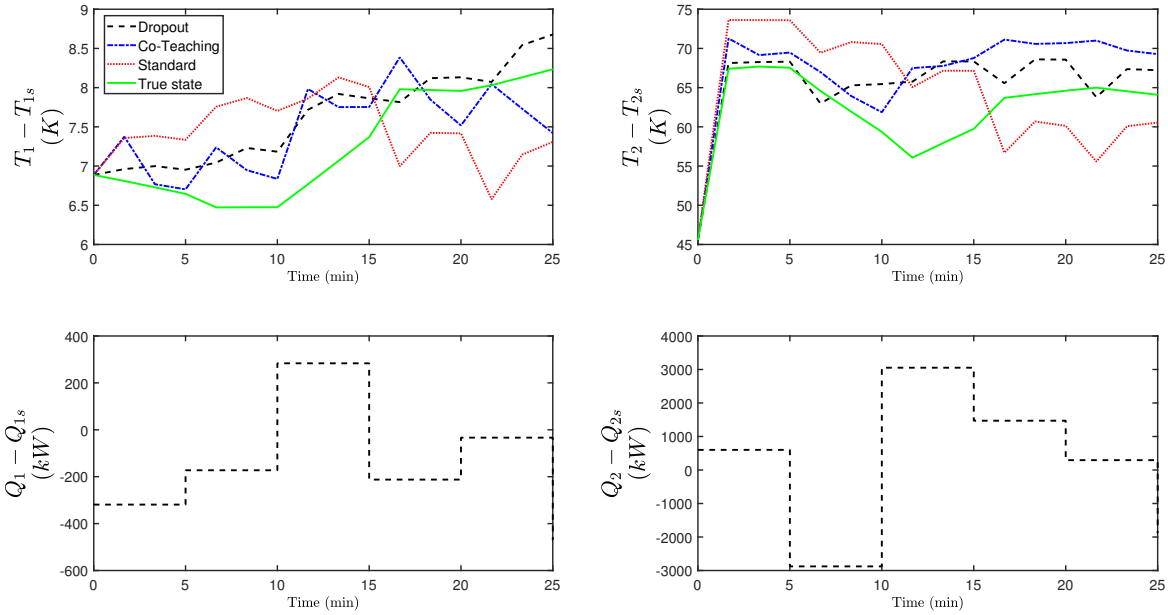

Figure 4.10: Open-loop state trajectory predicted by dropout LSTM, co-teaching LSTM, and standard LSTM, respectively, under the same time varying inputs in the presence of non-Gaussian noise.

Table 4.3: Open-loop prediction MSE results under Gaussian and non-Gaussian noise.

Dropout rate	Non-Gaussian	Gaussian
0.2	7.487×10^{-6}	2.02×10^{-6}
0.3	8.66×10^{-6}	2.87×10^{-6}
0.4	7.428×10^{-6}	5.08×10^{-6}
0.5	1.144×10^{-5}	6.75×10^{-6}

Table 4.4: Open-loop prediction results (MSE) by the three different models under non-Gaussian and Gaussian industrial noise.

Model	Non-Gaussian		Gaussian	
	x_5	x_{10}	x_5	x_{10}
Dropout	0.2445	24.197	0.27385	10.5538
Co-teaching	0.28458	35.56	0.265	19.255
Standard	0.9088	47.564	0.4485	21.968

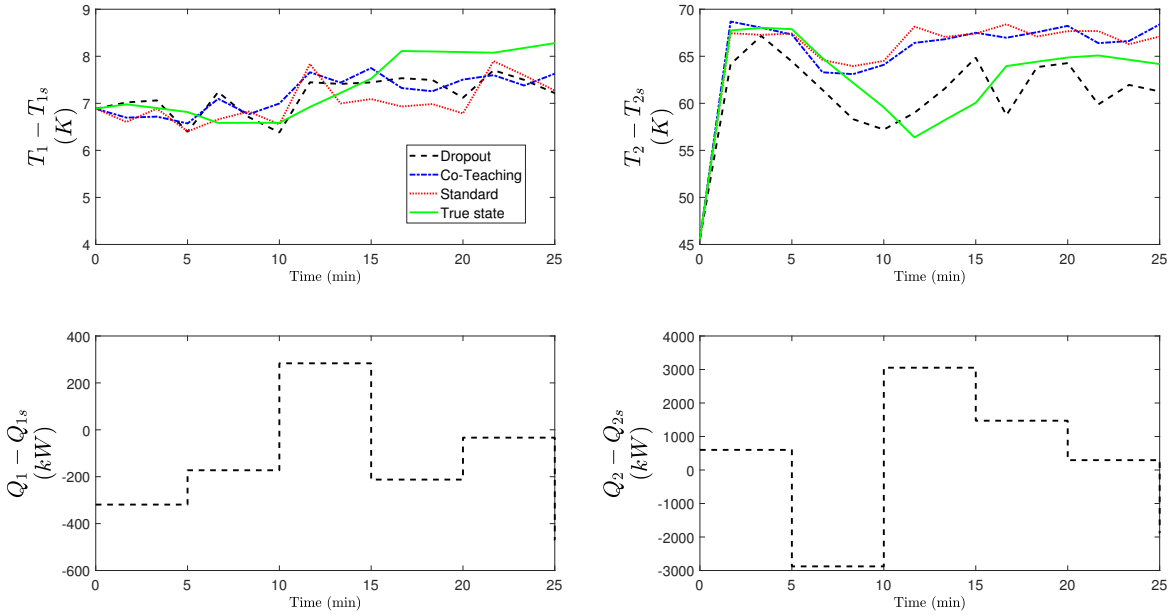

Figure 4.11: Open-loop state trajectory predicted by dropout LSTM, co-teaching LSTM, and standard LSTM, respectively, under the same time varying inputs in the presence of Gaussian noise.

4.7.3 Closed-loop simulation: Gaussian noise

Following the open-loop simulations, we performed closed-loop simulations in the case of existence of Gaussian noise in the measurement of the two reactor temperatures under an LMPC controller utilizing each of the three different PCRNN-LSTM models separately. The process dynamics corresponding to the PCRNN-LSTM model developed by the co-teaching method versus the standard PCRNN-LSTM model is presented in Fig. 4.12. Both LMPCs were able to derive the process to the desired steady-state and to stabilize the system around the origin even in the presence of Gaussian industrial noise. However, the improvement of the closed-loop performance when utilizing the co-teaching method is noticeable throughout the figure, as the temperatures trajectories are smoother and show less oscillation in comparison to the standard model.

Concurrently, a closed-loop simulation is performed using the dropout trained PCRNN-LSTM model. The results are plotted in Fig. 4.13. From the state trajectories and control actions in

Fig. 4.13, similarly to the co-teaching strategy, the dropout method has observably enhanced the performance of the LMPC.

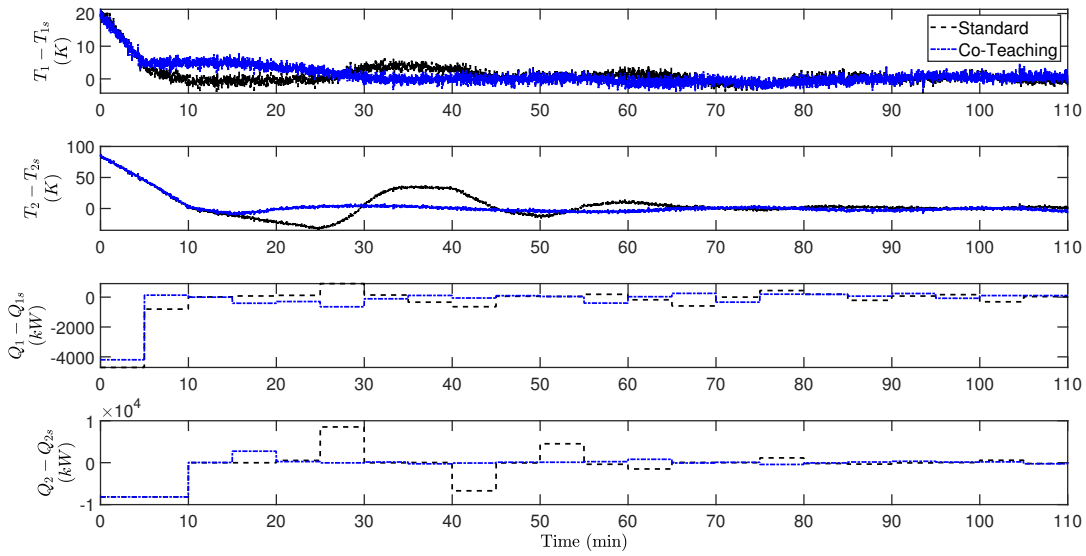

Figure 4.12: State and input profiles of the closed-loop simulation in the presence of Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

4.7.4 Closed-loop simulation: non-Gaussian noise

In this subsection, we discuss the results of closed-loop simulation in the existence of non-Gaussian type of noise. Figure 4.14 shows the closed-loop inputs and state trajectories under the LMPC when using the PCRNN-LSTM model trained according to the standard strategy and the co-teaching strategy. It is notable that the standard PCRNN-LSTM model shows significant variation when the closed-loop state reaches the steady-state. This stems from the fact that the states predicted using the standard LSTM model are not sufficiently close to the true state values; hence, the LMPC is deceived to provide a solution that drives the process states in the wrong direction. In the same figure, the co-teaching based PCRNN-LSTM demonstrated enhanced model accuracy, and that the LMPC using the co-teaching method was successfully able to derive the state

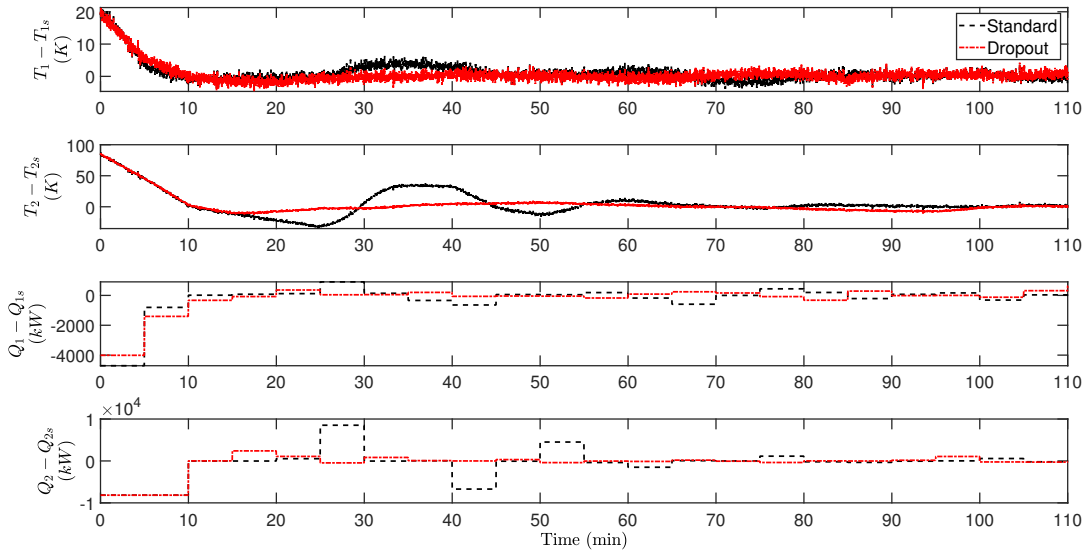

Figure 4.13: State and input profiles of the closed-loop simulation in the presence of Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dash-dotted line).

to the steady-state. As for the stability, the co-teaching method enabled the LMPC to maintain the state in a small neighborhood around the steady-state more smoothly in comparison with the PCRNN-LSTM model built following the standard training algorithm. Note that the MPC will not be able to stabilize the system exactly at the steady state due to the sample-and-hold implementation of control actions and the model mismatch between the LSTM model and the actual nonlinear process. Therefore, if the state trajectory of the closed-loop system starting from the stability region Ω_ρ remains bounded in Ω_ρ and converges to a small compact set around the origin where it will be maintained thereafter, then the system is considered practically stable under the sample-and-hold implementation of MPC.

Similarly, the dropout strategy in training the PCRNN-LSTM model, referred to as the "dropout model", is used as the predictive model for LMPC. The closed-loop simulation results of using the dropout model for the LMPC prediction is illustrated in Fig. 4.15. The figure demonstrates the superiority of the dropout model over the standard PCRNN-LSTM model in both metrics: (i) smoothness of the trajectories, and (ii) the ability to stabilize the underlying process within the

stability region Ω_ρ as discussed previously.

In the same vein, we carried out a closed-loop simulation in the presence of non-Gaussian noise using fully-connected RNN-LSTM models (FCRNN-LSTM) trained according to the three strategies considered in this work. Figs. 4.16 and 4.17 show the process state under the FCRNN-LSTM-based LMPC. Specifically, Fig. 4.16 depicts the states and inputs trajectories by models trained by co-teaching and standard strategies. Once more, the co-teaching improved the LMPC performance more observably in the $T_2 - T_{2s}$ trajectory. Similarly, in Fig. 4.17, the trained FCRNN-LSTM model based on the dropout strategy provided a more favorable closed-loop dynamics in relation to the standard FCRNN-LSTM. The three FCRNN-LSTM model based LMPCs were also able to derive the system to steady-state values, but took longer time and experienced more oscillatory behaviour than the PCRNN-LSTM models.

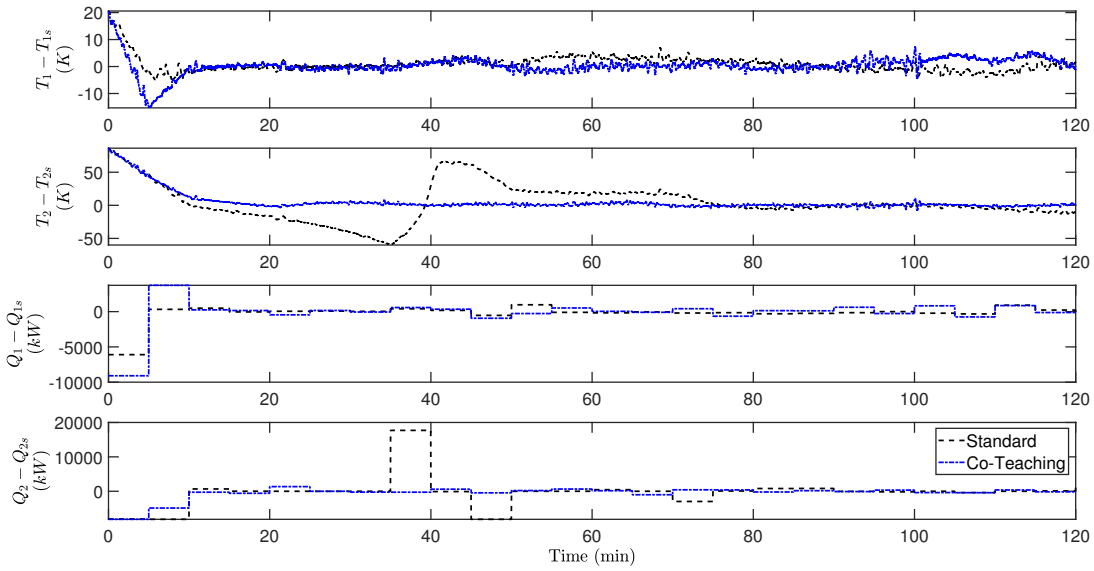

Figure 4.14: State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

In both fully-connected and partially-connected architectures, using co-teaching and dropout approaches lead to changes in the value of the integration of the cost function over the complete simulation duration against the standard method. In the partially-connected scenario, the cost

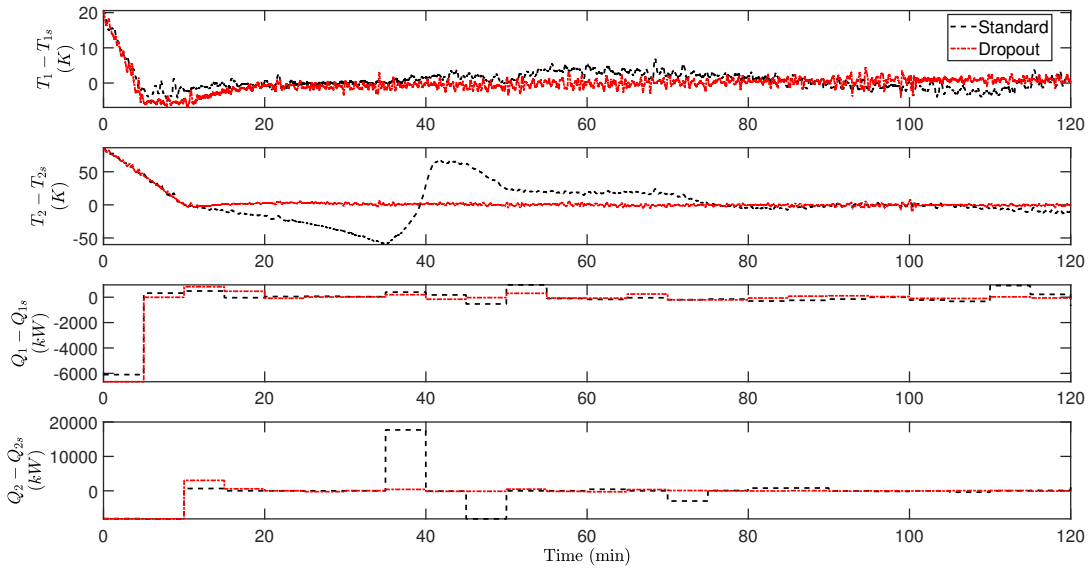

Figure 4.15: State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dash-dotted line).

function values for co-teaching and dropout approaches fell by 40.6% and 45.05%, respectively, when compared to the standard method, as shown in Table 4.5. Moreover, by using the partially-connected modeling over the fully-connected modeling the value of the integrated cost function associated with the controllers using the standard, co-teaching, and dropout based models are reduced by 37.5%, 12%, and 4%, respectively.

Table 4.5: The value of the time integral of the cost function using different models under two modeling architectures.

Model	Partially-connected	Fully-connected
Standard RNN-LSTM	1.01×10^9	1.6×10^9
Co-Teaching RNN-LSTM	6.01×10^8	6.8×10^8
Dropout RNN-LSTM	5.54×10^8	5.77×10^8

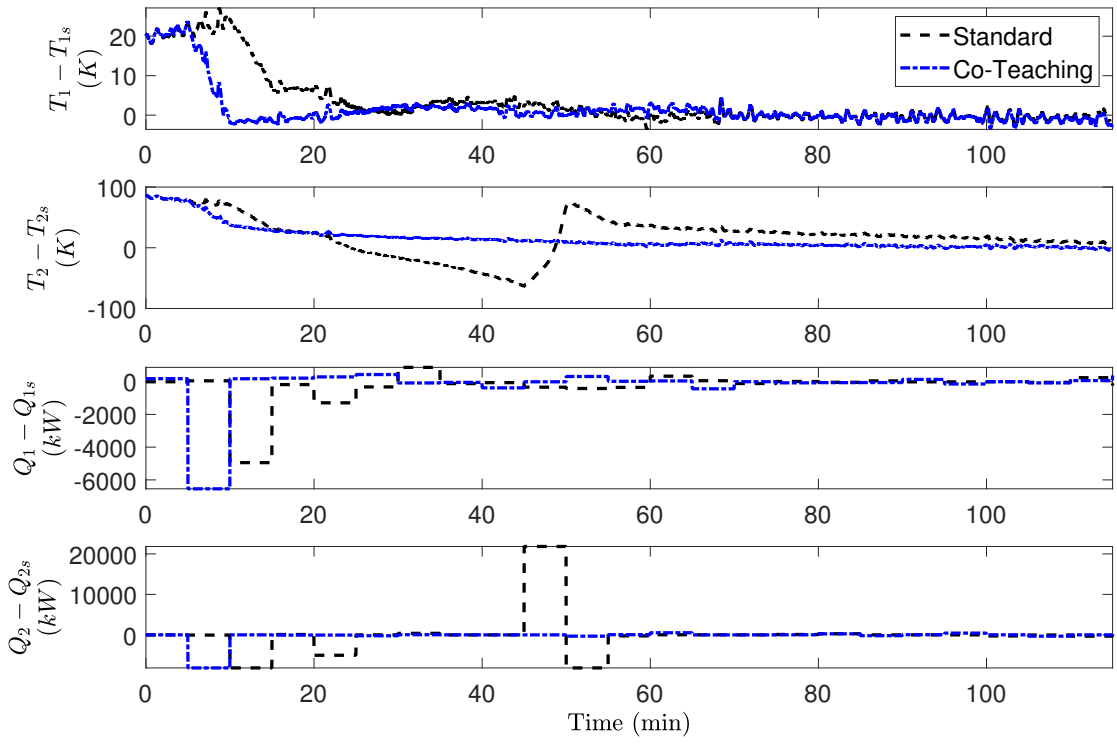

Figure 4.16: State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using FCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

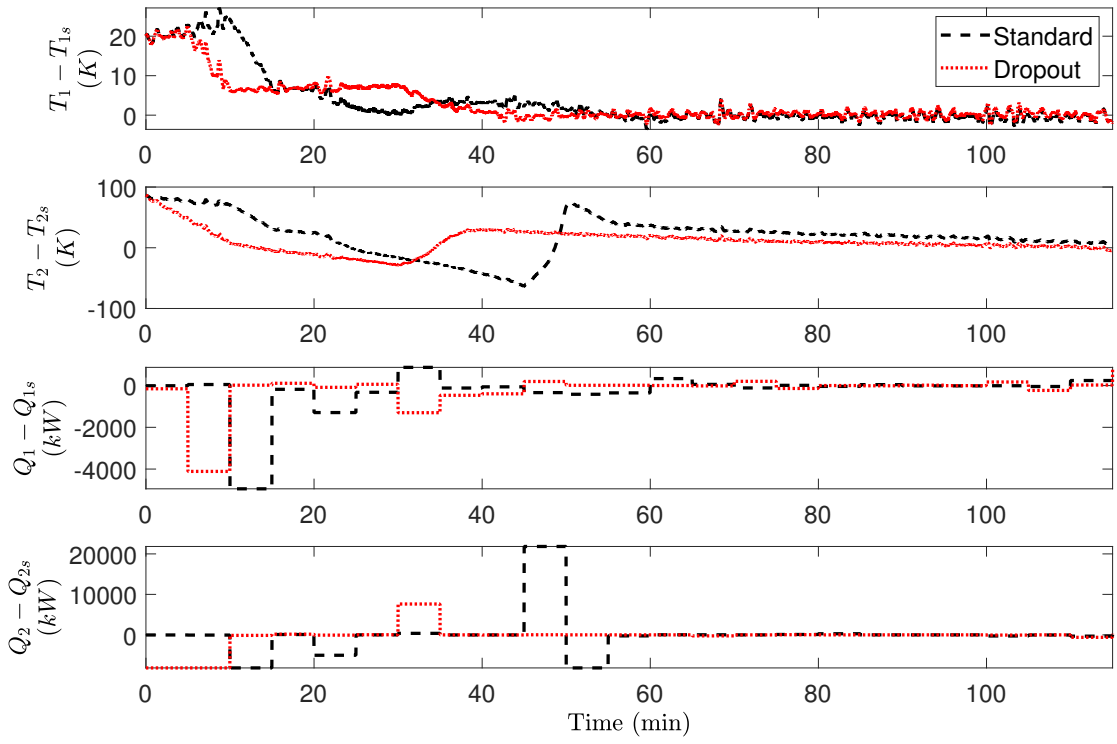

Figure 4.17: State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using FCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dotted line).

Chapter 5

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

5.1 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., [25, 108]). 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., [10,43,88,109]). Among various machine learning modeling tools, when modeling nonlinear

dynamic systems utilizing time-series data, recurrent neural networks (RNN) are frequently employed [34, 120]. Although machine learning techniques have been used in chemical process control since the nighties [101], 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 surly be impacted by the developments of MPC systems 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., [26, 48, 92]) have studied gray-box modeling, also referred to in literatuer 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 [75], 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 [114] 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 partially-connected RNN which as the name indicates it partially connects layers based on pre-existed knowledge in terms of physical relations among the underlying system inputs and outputs, and it was proposed in several works [5, 62, 114]. Specifically, On a large and complex chemical process modeled in Aspen Plus Dynamics simulator, [5] investigated this approach by evaluating 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 under MPC, with smoother state trajectories and less computing work. Furthermore, in [7] they considered the case of industrial noise (i.e., non-Gaussian noise), where they used the Monte Carlo dropout and the co-teaching strategies to train partially-connected RNN models to overcome the over-fitting issue. Subsequently, open-loop and closed-loop simulations were performed on Aspen dynamics process model to illustrate the superiority of partially-connected RNN based MPC over fully-connected RNN models trained in the same manner and a stander partially-connected RNN model. Additionally, one can consider an LSTM to model nonlinear systems, it is a machine learning technique that stands for Long Short-Term Memory Recurrent Neural Network. It is considered one of RNN variants, that was introduced in three decades ago. LSTMs have a unique structure in which it enables them to enhance the system'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 [7], a nonlinear system was modeled using an LSTM network through noisy data, and closed loop stability was achieved and analyzed. Moreover, LSTMs have been shown to overcome the vanishing gradient phenomena that usually occurs when using RNNs (interested readers are referred to [20]). 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 constrained by basic challenges, despite the effectiveness of machine learning approaches in simulating nonlinear chemical processes within the context of MPC. Furthermore, characterizing the generalization capability for machine learning models learned using finite training samples on new data is a significant challenge. The work of [115] has fill in the gap with constructing a theoretical upper bound for 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 5.2 presents the class of nonlinear systems considered and assumptions regarding system stability. In Section 5.3, the representation and the construction of RNNs both fully-connected and partially-connected is presented. Section 5.4 starts with key definitions and lemmas, and then develop probabilistic generalization error upper bounds for partially-connected RNN models. The integrating of a partially-connected RNN model model into a MPC while accounting for Lyapunov stability considerations is proposed and discussed in Section 5.5. Lastly, the improvements associated with incorporating prior physical knowledge into RNN modeling is illustrated in Section 5.6 via both open-loop and closed-loop simulations using a two reactors in series chemical process under Lyapunov-based MPC (LMPC).

5.2 Preliminaries

5.2.1 Notation

Given a vector $b \in R^n$, 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 $b \in R^n$, and $\gamma \geq 1$ $\|b\|_\gamma = (\sum_{i=1}^{n} |b_i|^\gamma)^{\frac{1}{\gamma}}$. Given a matrix $W \in R^{m \times n}$, its Frobenius and spectral norms are denoted by $\|W\|_F$, $\|W\|_\infty$, respectively. For simplicity, we will drop the subscript ∞ when dealing with the spectral norm and write it as $\|W\|$. Given real values γ, κ, the γ-norm ,κ-norms of the columns of W is denoted by, $\|W\|_{\gamma,\kappa} = \left(\sum_i^m \sum_j^n \left(|W_{j,i}|^\gamma\right)\right)^{\frac{1}{\kappa}}$. 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., $A \backslash B := \{x \in R^n | x \in A, x \notin B\}$. A function $f(.)$ is of class $\mathscr{C}^1$ if it is continuously differentiable. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathscr{K}$ if

it is strictly increasing and is zero only when evaluated at zero. A function $f : R^n \to R^m$ is said to be L-Lipschitz, $L \geqslant 0$, if $|f(a) - f(b)| \leq |a - b|$ for all $a, b \in R^n$. $\mathbb{P}(A)$ denotes the probability that the event A will occur. $\mathbb{E}[X]$ denotes the expected value of a random variable X.

5.2.2 Class of Systems

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

$$\dot{x} = \mathbf{F}(x,u) := F(x) + G(x)u \tag{5.1}$$

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

5.2.3 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. 5.1 exponentially stable. This stabilizability assumption implies the existence of a $\mathscr{C}^1$ control Lyapunov function denoted as $V(x)$, such that the following inequalities hold for all x in an open neighborhood D around the origin:

$$c_1\|x\|^2 \leq V(x) \leq c_2\|x\|^2 \tag{5.2a}$$

$$\frac{\partial V(x)}{\partial x}\mathbf{F}(x,\Phi(x)) \leq -c_3\|x\|^2 \tag{5.2b}$$

$$\|\frac{\partial V(x)}{\partial x}\| \leq c_4\|x\| \tag{5.2c}$$

where c_i , $i = 1,2,3,4$, are positive constants. A candidate controller $\Phi(x)$ may be constructed via Sontag's control law formula ([61]). Then, following [118], 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 $u \in U$:

$$\|\mathbf{F}(x,u)\| \leq M \tag{5.3a}$$

$$\|\mathbf{F}(x,u) - \mathbf{F}(x',u)\| \leq L_x \|x - x'\| \tag{5.3b}$$

$$\|\frac{\partial V(x)}{\partial x}\mathbf{F}(x,u) - \frac{\partial V(x')}{\partial x}\mathbf{F}(x',u)\| \leq L_x^{'} \|x - x'\| \tag{5.3c}$$

5.3 Recurrent neural networks (RNNs)

The investigation of utilizing artificial intelligence (AI) techniques in chemical engineering has been carried out continuously. The AI technology has provided classic and powerful modeling tools such as fuzzy logic in the 1960s [125], expert systems in the 1980s [57, 59], and machine learning (ML) in the 1990s ([101]). Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications [12, 27, 89, 109]. For example, in [12], 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 ±8% deviation from the experimental data. Similarly, recurrent neural networks (RNN) have been broadly employed for modelling a general class of dynamical systems for control and state estimation purposes [73]. In [89], a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the RNN model prediction can outperform a first-principles model for large-scale, complex, nonlinear process, due to its high degree of freedom to solve the

complex non-linear regression problem with the process dataset.

RNN models are a powerful tool for modeling dynamic systems Considering an RNN model that resembles the nonlinear dynamics of the system of Eq. 5.1 using m sequences of T-time-length data points $(x_{i,t}, y_{i,t})$, with $x_{i,t} \in \mathbf{R}^{d_x}$ serving as the RNN input and $y_{i,t} \in \mathbf{R}^{d_y}$ serving as the RNN output and that $i = 1, ..., m$ and $t = 1, ..., T$. It is important to emphasize that the nonlinear system inputs, states, and outputs in Eq. 5.1 are not always represented by the RNN inputs and outputs. Hence, all the vectors for RNN models are represented in boldface to identify them from the notations for the nonlinear system of Eq. 5.1.

Moreover, to make the discussion simpler, the RNN model of Eqs. 5.4-5.5 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 dataset is created of m data sequences that were individually selected from an underlying distribution over $\mathbf{R}^{d_x \times T} \times \mathbf{R}^{d_y \times T}$. To simplify the discussion, we consider a single-hidden-layer RNN model with the following form to approximate the nonlinear dynamics of Eq. 5.1.

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t) \tag{5.4}$$

$$\mathbf{y}_t = \sigma_y(V\mathbf{h}_t) \tag{5.5}$$

where $\mathbf{h}_t$ denotes the hidden state, and W, U, and V are the weight matrices connecting different layers. The 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 5.1. *Input data are bounded. i.e.,* $\|x_{i,t}\| \leq B_x$ *for all* $i = 1, ..., m$ *and* $t = 1, ..., T$

Assumption 5.2. *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 5.3. *Training, validation, and testing datasets are drawn from the same distribution.*

Assumption 5.4. 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* $z \in \mathbf{R}$

Furthermore, consider a hypothesis class $\mathscr{H}$ of RNN models $h(\cdot)$ 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 $L(\mathbf{y}_t, \tilde{\mathbf{y}}_t)$, respectively, where $L(\mathbf{y}, \tilde{\mathbf{y}})$ calculates the squared difference between the predicted output $\mathbf{y}$ and the true output $\tilde{\mathbf{y}}$.

5.3.1 Physics-informed RNNs

Even the most cutting-edge black-box ML models, like dense fully-connected RNN models, have had only limited success when used in scientific fields [49]. This is because such models require a lot of data, may produce physically inconsistent results, and may exhibit reduced accuracy when applied to samples that haven't been seen before. Therefore, researchers have started to look at the transition between mechanistic and ML models, where data and scientific knowledge are combined in a beneficial way. This is due to the possibility that neither a pure ML algorithm nor a purely scientific theory may be enough for complicated applications (e.g., [3, 11, 79]). In a fundamentally different way from popular approaches in the ML field, a physics-based machine learning modeling strategy makes use of pre-existing physical knowledge in supporting roles like feature engineering or post-processing. Despite extensive research on this topic, the concept of using ML in conjunction with scientific principles for modeling complex systems has just lately become popular [49]. This investigation is being conducted in several fields in the realm of science, where early findings in isolated and straightforward scenarios have been encouraging, and expectations are growing that this paradigm will speed up scientific advancement.

Similarly, in process system engineering and chemical engineering core research areas, the present paradigm of numerical approaches to get approximated solutions is based solely on physics: numerical differentiation and integration algorithms are used to solve for systems of differential equations that reflect established physical principles throughout space and time [14, 45, 83]. 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 for turbulent flows one may utilize Reynolds-averaged Navier-Stokes equations [19, 96]. But creating a simplified model that accurately captures a phenomenon is quite difficult. More crucially, only a portion of the dynamics of many complicated real-world processes is recognized. The equations could not accurately reflect the actual system states. On the other hand, numerous recent studies, from turbulence and reaction modeling to state prediction, have demonstrated that ML-based models can produce realistic predictions and greatly speed up the simulation of complex dynamics compared to numerical solvers [50, 63]. 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 dataset cannot adequately adapt to unseen scenarios. Hence, approximating complicated dynamical systems in scientific areas cannot be considered sufficiently fulfilled by either machine-learning-based models alone or simply physics-based methods. As there is a significant necessity to integrate machine learning models with conventional physics-based methodologies, through which we can maximize the benefits of both techniques.

The investigation of more structured systems 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 space to construct networks with organized neighborhoods than a fully-connected network [17]. Typically, when creating a dynamic model for a nonlinear process, a neural network model that

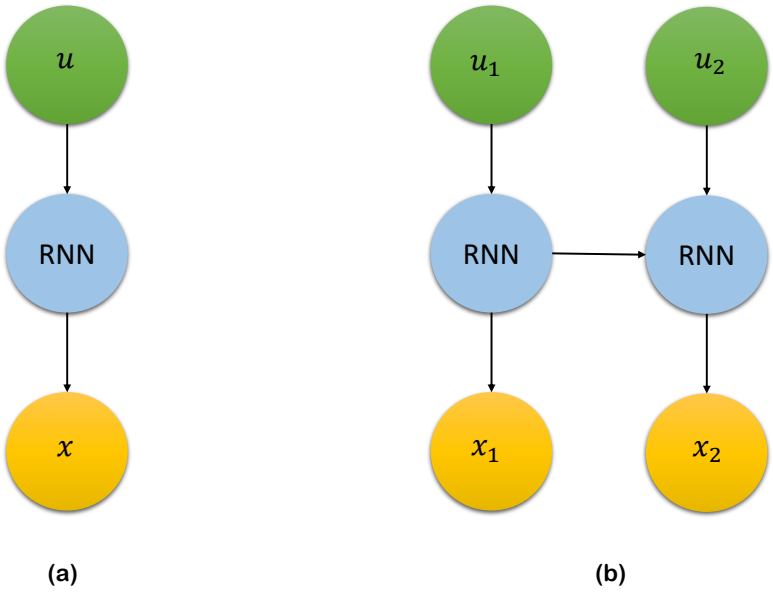

Figure 5.1: Structure of (a) standard fully-connected and (b) partially-connected RNN.

uses all available process inputs to predict the desired outputs evolution is preferred. Creating a dynamic model for these processes is simple to do with open-source machine learning tools, and the model would be able to take into account any connections that might exist between each input and each output of the underlying process. As depicted in Fig. 5.1, 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 processes that lack prior knowledge.

Although standard RNNs do not consider any domain-specific knowledge in model development and generally use fully-connected layers to capture input-output relationship using the given training dataset, it has been demonstrated in [114] that a priori process structural knowledge can be utilized to improve RNN performance by developing a partially-connected architecture. Fig. 5.1 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 from 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. 5.1 for which the input vector u_1 affects the state x_1 only, and both u_1 and u_2 affect the state x_2, where $x = [x_1 \in \mathbf{R}^{n_{x_1}},\ x_2 \in \mathbf{R}^{n_{x_2}}]$ and $u = [u_1 \in \mathbf{R}^{n_{u_1}},\ u_2 \in \mathbf{R}^{n_{u_2}}] \in \mathbf{R}^{n_u}$, where $n_{u_1} + n_{u_2} = n_u$ and $n_{x_1} + n_{x_2} = n_x$. [114] demonstrates that by using a partially-connected architecture, the number of weight parameters can be significantly reduced to achieve a desired model accuracy compared to a fully-connected RNN model. Additionally, in [5], 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 computation 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 developed.

5.4 Generalization error

5.4.1 General considerations

Generalizability or generalization accuracy is a metric that measures a machine learning model's ability to adapt properly to new, previously unseen data, which 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., [32, 82]). Furthermore, a theoretical analysis of generalization error is of significant importance as it provides a fundamental understanding on how good the model performs on unseen data that will be collected in real-world systems. This section will provide the 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 as follows.

Definition 5.1. *A centered random variable $x \in R$ is said to be sub-Gaussian with variance proxy*

σ^2, *if* $\mathbb{E}[x] = 0$, *and the moment generating function satisfies:*

$$\mathbb{E}[\exp(aX)] \leq \exp\left(\frac{a^2\sigma^2}{2}\right), \forall a \in R \tag{5.6}$$

Definition 5.2. *Given a data distribution D, and a function h that predicts y (output) based on x (input), the generalization error is given by*

$$\mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y)\ \rho(x, y)\ dxdy. \tag{5.7}$$

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 5.3. $L(\cdot, \cdot)$ *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:*

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i) \tag{5.8}$$

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

Definition 5.4. *Given a set of data samples* $S = \{s_1, ..., s_m\}$, *and a hypothesis class* $\mathscr{F}$ *of real-valued functions, the definition of empirical Rademacher complexity of* $\mathscr{F}$ *is*

$$\mathscr{R}_S(\mathscr{F}) = \mathbb{E}_\varepsilon \left[\sup_{f \in \mathscr{F}} \frac{1}{m}\sum_{i=1}^{m} \varepsilon_i f(s_i)\right] \tag{5.9}$$

where $\varepsilon = (\varepsilon_1, ..., \varepsilon_m)^T$, *and* ε_i *are Rademacher random variables that are independent and identically distributed (i.i.d.) and satisfy* $\mathbb{P}(\varepsilon_i = -1) = \mathbb{P}(\varepsilon_i = 1) = 0.5$.

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

Lemma 5.1. *Consider a hypothesis class* $\mathscr{H}$ *of vector-valued functions* $h \in R^{d_y}$ *,and a set of data*

samples $S = \{s_1, ..., s_m\}$. Let $L(.)$ be a L_r -Lipschitz function mapping $h \in R^{d_y}$ to R, then we have:

$$\mathbb{E}_\varepsilon \left[\sup_{f \in \mathscr{F}} \sum_{i=1}^{m} \varepsilon_i L(h(\mathbf{x}_i), \mathbf{y}_i) \right] \leq \sqrt{2} L_r \mathbb{E}_\varepsilon \left[\sup_{h \in \mathscr{H}} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \varepsilon_{ik} h(\mathbf{x}_i) \right] \tag{5.10}$$

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 expectation 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:

$$\mathbb{E} \left[\sup_{h \in \mathscr{H}} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \varepsilon_{ik} h(\mathbf{x}_i) \right] \leq \sum_{k=1}^{d_y} \mathbb{E} \left[\sup_{h \in \mathscr{H}} \sum_{i=1}^{m} \varepsilon_i h(\mathbf{x}_i) \right] \tag{5.11}$$

where $\mathscr{H}_k, k = 1, ..., d_y$, are classes of scalar-valued functions that correspond to the components of vector-valued functions in $\mathscr{H}$.

Lemma 5.2 (c.f. Theorem 3.3 in [68])**.** *Let $\mathscr{H}$ be the hypothesis class of ML models that map $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_t\} \in \mathbf{R}^{d_x \times t}$ (i.e., the first t-time-step inputs) to $\mathbf{y}_t \in \mathbf{R}^{d_y}$ (i.e., the t-th output), and $\mathscr{G}_t$ be the loss function set with $\mathscr{H}$.*

$$\mathscr{G}_t = \{ g_t : (\boldsymbol{x}, \tilde{\mathbf{y}}) \rightarrow L(h(\boldsymbol{x}), \tilde{\mathbf{y}}), h \in \mathscr{H} \} \tag{5.12}$$

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

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m} \sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathscr{R}_S(\mathscr{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{5.13}$$

Eq. 5.13 demonstrates that the upper bound for the generalization error depends on the training error (first term), the Rademacher complexity of $\mathscr{G}_t$ (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 5.3. *Given a hypothesis class $\mathscr{H}_k$ of real-valued functions corresponding to the $k-$th component of vector-valued function class $\mathscr{H}$, 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:*

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_k) &= \mathbb{E}\left[\sup_{h\in\mathscr{H}_k}\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right]. \\ &= \frac{1}{\lambda}\log\,\exp\left(\lambda\mathbb{E}\left[\sup_{h\in\mathscr{H}_k}\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right]\right) \\ &\leq \frac{1}{\lambda}\log\,\mathbb{E}\left[\sup_{h\in\mathscr{H}_k}\exp\left(\lambda\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right)\right] \end{aligned} \tag{5.14}$$

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

Lemma 5.4. *Let $\mathscr{H}_{k,t}$, $k = 1,...,d_y$, 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*

$$\mathscr{R}_S(\mathscr{H}_{k,t}) \leq \frac{M(\sqrt{2log(2)t}+1)B_X}{\sqrt{m}} \tag{5.15}$$

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 5.5 (c.f. Theorem 1 in [115])**.** *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 $\mathscr{G}_t$ associated with the RNN function class $\mathscr{H}_t$ 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:*

$$\mathbb{E}[g_t(\mathbf{x},\mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i,\mathbf{y}_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathscr{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{5.16}$$

The above equation represent the theoretical generalization accuracy upper bound of RNN

models. This theory will be utilized to build a relation between a RNN model and its structure in the subsection 5.4.2.

5.4.2 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. 5.2, x_2 does not affect y_1, so the weights corresponding to the linkages between x_2 and y_1 (dashed lines in Fig. 5.2) 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.

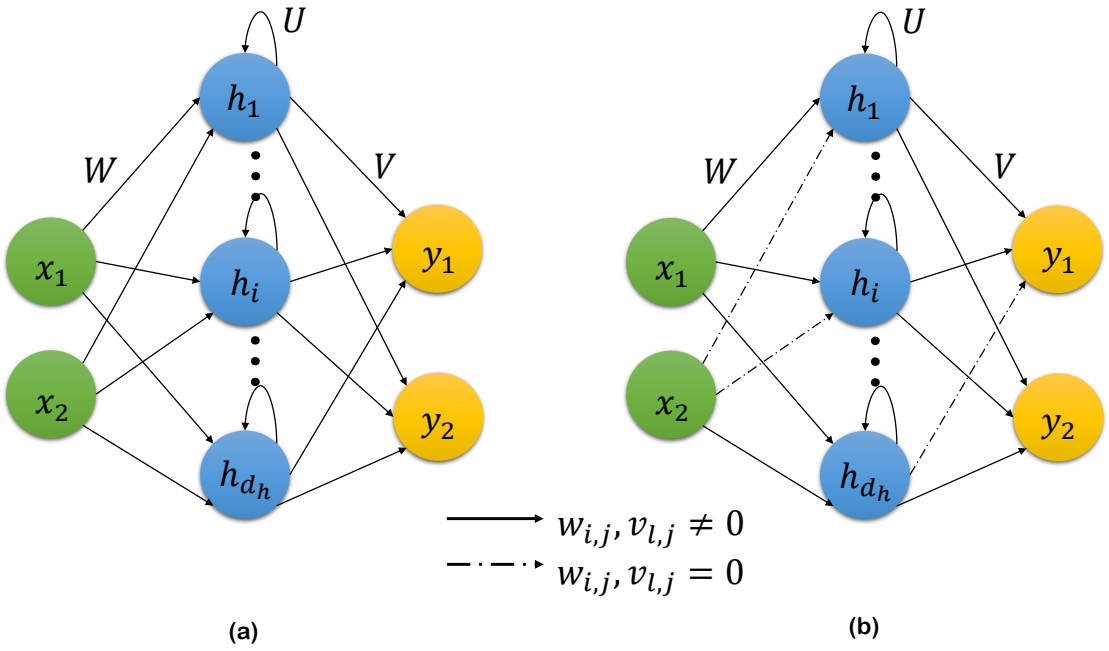

Figure 5.2: Weights and connections in (a) standard fully-connected and (b) partially-connected RNN structures, where zeroed weights for links between units are represented by dashed lines.

Theorem 5.1. *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 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:

$$\hat{\nu} < \nu \tag{5.17}$$

Proof. If we let ν denote the right-hand side of Eq. 5.16, ν can be represented as the sum of the three terms in the right-hand side of Eq. 5.16 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 $\mathscr{H}$. Due to the dense structure of FCRNN models, the size of the hypothesis class $\mathscr{H}$ 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 $\mathscr{H}$ 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 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:

$$\nu_{III} = \mathscr{O}\left(L_r d_y \frac{M B_X (1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{5.18a}$$

$$\hat{\nu}_{III} = \mathscr{O}\left(L_r d_y \frac{\hat{M} B_X (1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{5.18b}$$

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. 5.16, where we symbolize the PCRNN model weight matrices with hat as $\hat{V}$ 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.,

$$B_{\hat{W},F} < B_{W,F} \tag{5.19a}$$

$$B_{\hat{V},F} < B_{V,F} \tag{5.19b}$$

which yields

$$\hat{\nu}_{III} < \nu_{III} \tag{5.19c}$$

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

Remark 5.1. *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. 5.16) 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.*

5.5 RNN based model predictive control

In this section, we integrate an RNN model into a Lyapunov-based model predictive controller (LMPC) formulation. In particular, the partially-connected modelling of RNN is executed as discussed in [5] 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:

$$\mathscr{J} = \min_{u \in \mathsf{S}(\Delta)} \int_{t_k}^{t_k+\mathsf{P}} L(\tilde{x}(t), u(t))dt \tag{5.20a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{5.20b}$$

$$u(t) \in U, \ \forall\, t \in [t_k, t_k + \mathsf{P}) \tag{5.20c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.20d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}} \tag{5.20e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall\, t \in [t_k, t_k + \mathsf{P}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.20f}$$

where $\mathsf{S}(\Delta)$ denotes a set of piecewise constant functions with period Δ, $\tilde{x}$ is the state trajectory predicted by the RNN 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. 5.20e 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. 5.20a, which represents the cost function over the prediction horizon while satisfying the constraints of Eqs. 5.20b–5.20f. The RNN model from Eq. 5.20b 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. 5.20d, where $x(t_k)$ is the last state measurement. The input constraints are expressed in Eq. 5.20c, 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. 5.20e is triggered. As a result of this constraint, the Lyapunov function of the closed-loop states 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. Following section 2.3, the controller $\Phi_{nn}(x)$ was developed with the intent of ensuring that the origin of the RNN system is exponentially stable.

A well-conditioned RNN model with appropriate model accuracy can be 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 applying the RNN based LMPC of Eq. 5.20 for the regulation of the nonlinear system as that of Eq. 5.1. This is ture provided that the modeling error, which is given by $|\upsilon| = |F(x,u) - F_{nn}(x,u)|$, is sufficiently small [8, 118].

5.6 Application to a chemical process

A chemical process example is used for demonstrating the anticipated improvements associated with physics-informed modelling of RNNs. Particularly, two non-isothermal continuous stirred tank reactors (CSTR) in sequence with ideal mixing are taken into consideration, with each reactor experiencing an irreversible second-order exothermic reaction, where a raw material A transforms 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}, 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 he two CSTRs represented by the following system of ODEs is derived from material and energy balance equations:

$$\frac{dC_{A,1}}{dt} = \frac{F_{1_o}}{V_1}(C_{A1_o} - C_{A,1}) - k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 \tag{5.21a}$$

$$\frac{dC_{B,1}}{dt} = \frac{F_{1_o}}{V_1} C_{B,1} + k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 \tag{5.21b}$$

$$\frac{dT_1}{dt} = \frac{F_{1_o}}{V_1}(T_{1_o} - T_1) + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 + \frac{Q_1}{\rho C_p V_1} \tag{5.21c}$$

$$\frac{dC_{A,2}}{dt} = \frac{F_1}{V_2} C_{A,1} + \frac{F_{2_o}}{V_2} C_{A,2_o} + \frac{F_1 + F_{2_o}}{V_2} C_{A,2} - k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 \tag{5.21d}$$

$$\frac{dC_{B,2}}{dt} = \frac{F_1}{V_2} C_{B,1} - \frac{F_1 + F_{2_o}}{V_2} C_{B,2} + k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 \tag{5.21e}$$

$$\frac{dT_2}{dt} = \frac{F_{2_o}}{V_2} T_{2_o} + \frac{F_1}{V_2} T_1 - \frac{F_1 + F_{2_o}}{V_2} T_2 + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 + \frac{Q_2}{\rho C_p V_2} \tag{5.21f}$$

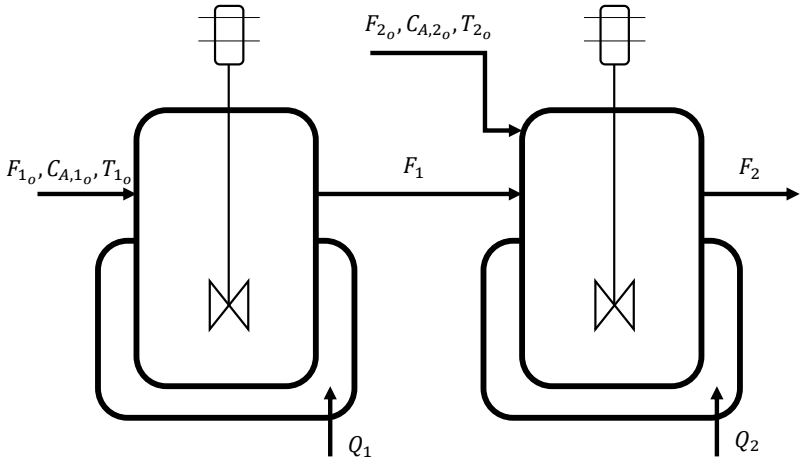

Figure 5.3: Two continuous-stirred tank reactors in series.

the notations $C_{A,i}$, , T_i and Q_i, represent the A reactant 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 5.1.

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

Table 5.1: Parameter and steady-state values for the CSTR

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

and $u_2 = Q_2 - Q_{2s}$. The upper and the lower physical bounds on the inputs are $[U^{max}, U^{min}] = [5, -5] \times 10^5\ kJ/h$, respectively. The state are also to be represented in deviation fashion from their steady-state value 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.

5.6.1 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 [111], providing that the data are 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 used extensive open-loop simulations in our work to create our data set.

For the development of the RNN model, the following procedures are followed for data generation, neural network training, and validation. 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. 5.21 for an

one sample time under various initial conditions (a total of 3000 different combinations of initial conditions). Particularly, MATLAB was used to create a data set of size m_{data}. The data set was 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 of data generation and by using the Keras library, two RNN models are constructed where each of the models has two hidden layers with 30 neurons at each, and hyperbolic tangent (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) as activation function, that beside to 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 modelling 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 [5].

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 performed 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) as shown in Fig. 5.4 testing data sets to test the developed models. The generalization error for each testing data set is illustrated in Fig. 5.5, where the partially-connected RNN model yielded higher generalization accuracy (i.e., less error). These results aligned with Theorem 1.

5.6.2 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 an open-loop simulation as illustrated in Fig. 5.6, where the time-varying inputs are randomly chosen. Furthermore, from the figure, it can be noticed that the state trajectories

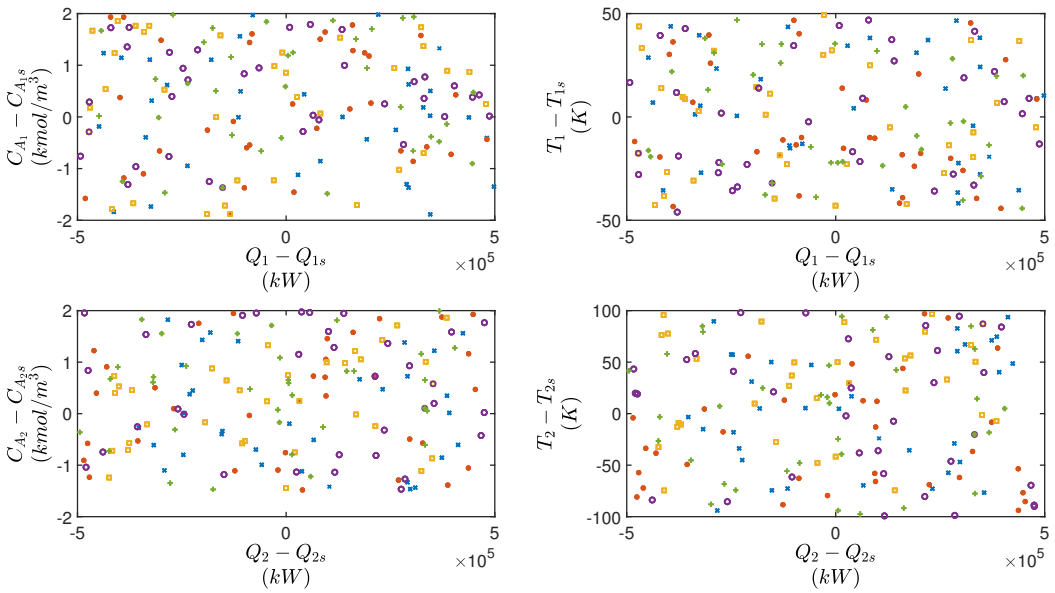

Figure 5.4: Five different testing data sets, where each marker indicates a single set.

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.

Table 5.2 presents the open loop simulation MSEs between the predicted states from each RNN model architecture and the corresponding fist-principle model outputs as the ground-truth process output value. Based on the table, 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 grater than one, which implies the higher accuracy associated with employing partially-connected RNN architecture 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. 5.7. The figure demonstrates that the partially-connected RNN model has an improved model identification of the process dynamics, as the trajectory of the first reactor temperature was not altered by this change. All these results indicate that both RNN models provide reasonable prediction, yet, the partially-connected RNN model approximates the underlying process model more accurately.

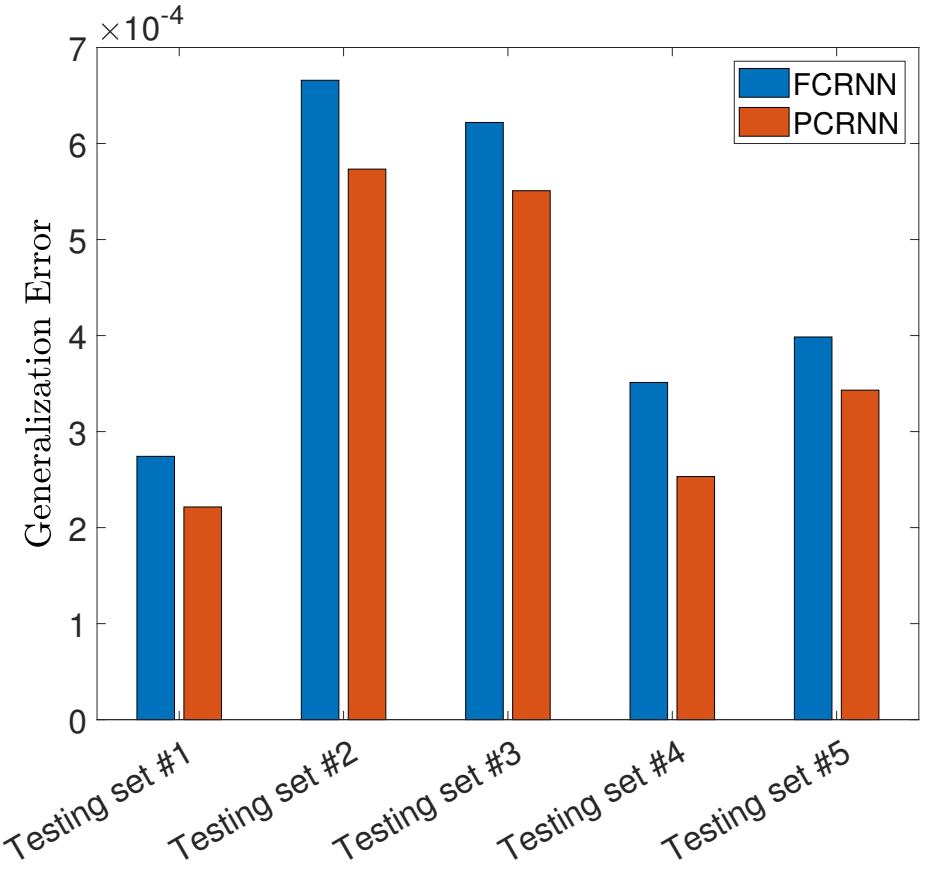

Figure 5.5: Generalization error for five different testing data sets, where PCRNN and FCRNN stands for partially-connected RNNs (orange bars) and fully-connected RNNs (blue bars), respectively.

Table 5.2: 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

5.6.3 Closed-loop simulation

Next, in order to run the closed-loop simulation 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

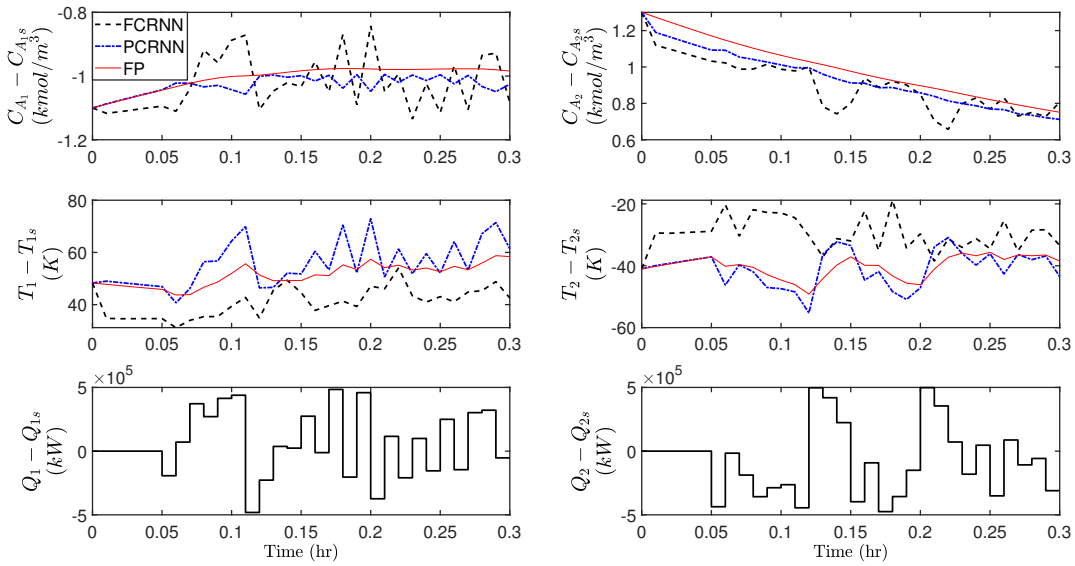

Figure 5.6: Time-varying profiles of the states and inputs for the second open-loop simulation under random time varying inputs using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN (black line).

sample period, the nonlinear minimization problem of the LMPC is solved using the Python version of the interior point optimizer (IPOPT) software. For the purpose of resolving complex nonlinear optimization issues, 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 programming problems. 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 setpoint error and control actions, respectively. The two matrices are critically impacting the performance of the LMPC and require proper tuning, hence, the MPC tuning guidelines discussed in [6] is followed. Lastly, we choose $V(x) = x^T P x$ as the Lyapunov function, where P is a positive definite matrix obtained by applying grid search.

Under the LMPC, we perform two closed-loop simulations and the results are shown in Fig. 5.8 and 5.9 initiating from two different initial conditions. 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 small neighborhood around the origin. However, partially-connected RNN based

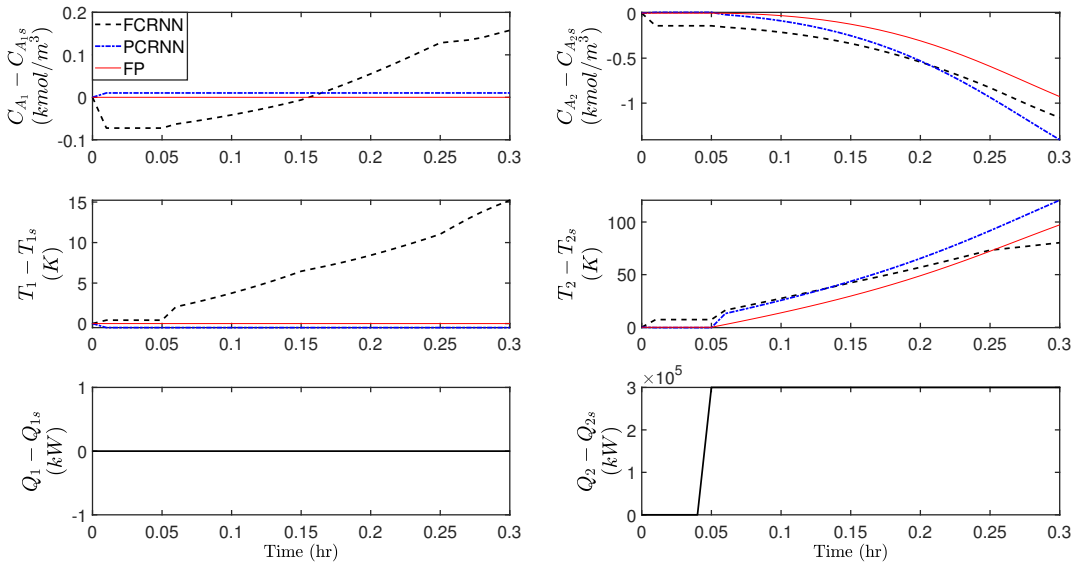

Figure 5.7: Time-varying profiles of the states and inputs for the open-loop simulation under a step change in u2 using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN (black line).

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 modeling techniques are calculated for the two closed-loop simulations and presented in Table 5.3. As it can be noted from the tables, the partially-connected RNN model yielded more reliable controller performance with smaller MSE values by an order of magnitude compared to fully-connected RNN model.Due to the fully-connected RNN's interference with the prediction accuracy caused by the assumption that every input influences every potential output, this results are expected.

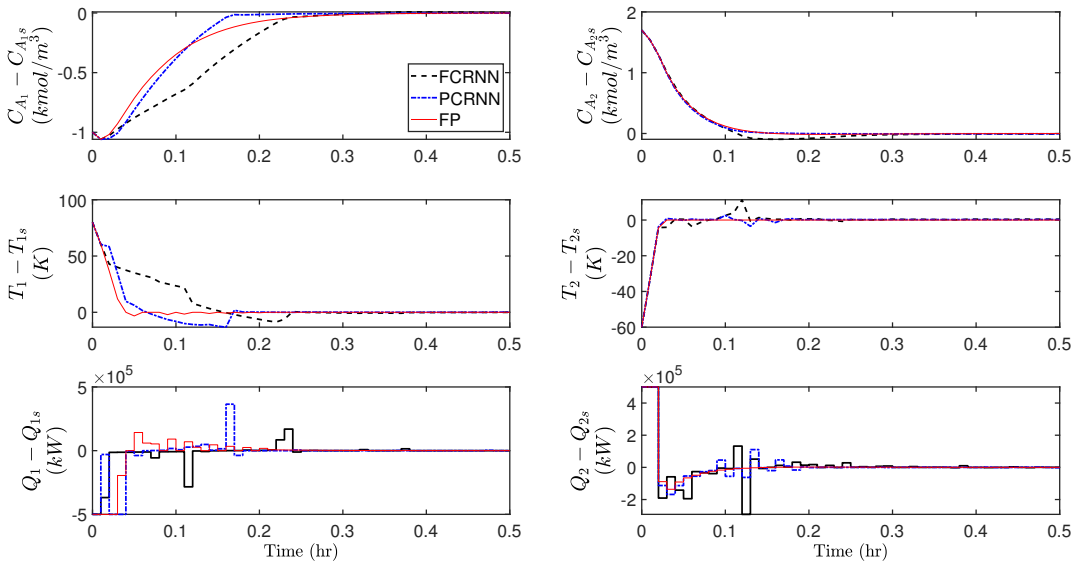

Figure 5.8: Sate and input profiles of the first closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

Table 5.3: 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

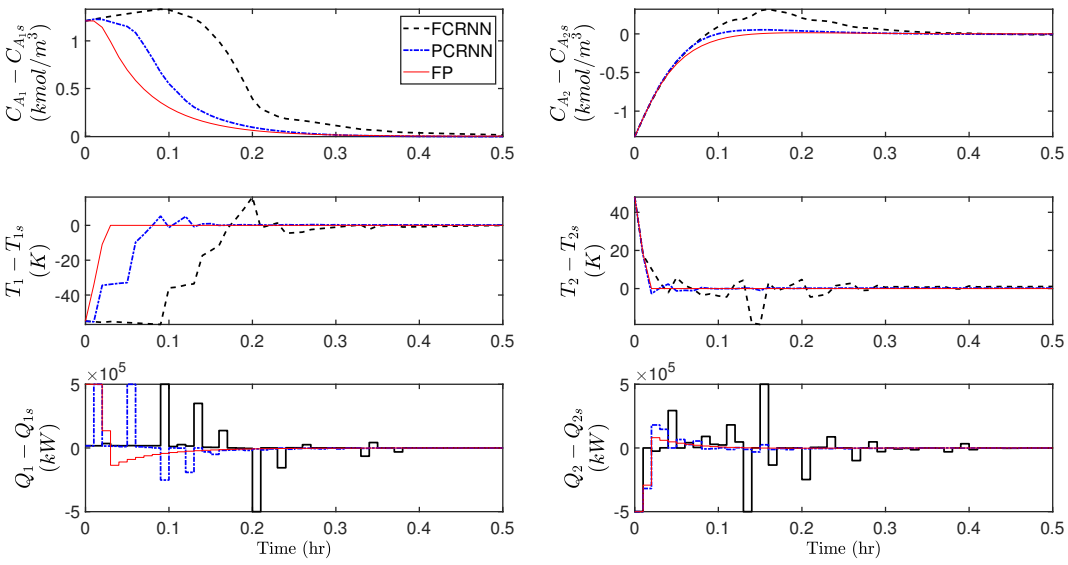

Figure 5.9: State and input profiles of the second closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

Chapter 6

Conclusions

This dissertation discusses *prior*-knowledge-based designs of machine-learning-based MPC systems to improve closed-loop performance and process operation of nonlinear chemical processes. Firstly, machine-learning-based state estimation schemes were proposed and incorporated in Lyapunov-based MPC controllers to overcome the challenge of incomplete state measurements. Next, physical process structure knowledge was incorporated into the development of data-driven-based models for large, highly nonlinear, and complex chemical processes. The performance enhancement associated with these models was investigated through prediction accuracy and model identification testing, and in closed-loop dynamics under Lyapunov-based MPC. Lastly, a theoretical framework that interprets the higher generalization accuracy of physics-based RNN models compared to typical dense RNN models was presented.

In Chapter 2, we proposed machine-learning-based state estimation approaches for nonlinear processes. The RNN model was first developed to represent process dynamics in the operating region, and then it was incorporated in an extended Luenberger observer. Then, the RNN-based estimator was used to provide state estimates for the optimization problem of the LMPC. Subsequently, a hybrid model was developed to represent process dynamics and used in the state estimator. From closed-loop simulations, it was demonstrated that both the RNN-based estimator and the hybrid-model-based estimator achieved the desired accuracy in state estimation, and all the

state trajectories initiating from different initial conditions converged to the steady-state under the LMPC using machine-learning-based estimators.

In Chapter 3, a partially-connected RNN model, which integrates a *priori* process-structure knowledge into the RNN modeling, was developed and utilized as a predictive model in an LMPC scheme, and evaluated in a dynamic simulation of a chemical process using Aspen Plus Dynamics. It was then compared with a fully-connected RNN-based LMPC. The open-loop simulations demonstrated the superiority of the partially-connected RNN by yielding smaller prediction errors. Furthermore, the closed-loop simulations demonstrated that the chemical process under the partially-connected RNN-based LMPC had smoother state trajectories, and the optimal control action calculations required a smaller computational time. Finally, the partially-connected RNN-based LMPC gave a steady control signal after the process reached steady-state, which eliminated the states variance when under a fully-connected RNN-based LMPC.

In Chapter 4, we used the Monte Carlo dropout and the co-teaching strategies to train PCRNN-LSTM models for predicting underlying process dynamics (ground truth) from noisy data. The Ethylbenzene production process conducted in two CSTRs in series was considered, and modeled via the Aspen Plus Dynamics simulator. The co-teaching and dropout strategies were applied to create PCRNN-LSTM models with two type of noise independently (i.e., Gaussian and non-Gaussian), where noisy and noise-free data were generated by extensive open-loop simulations of the Aspen dynamical model and the first-principles model, respectively. Subsequently, open-loop as well as closed-loop simulations were performed to illustrate the superiority of PCRNN-LSTM based models trained via co-teaching and dropout techniques over the standard PCRNN-LSTM modeling technique in terms of enhancing the accuracy of open-loop predictions as well as improving the closed-loop performance. In comparison to the standard approach, both co-teaching and dropout techniques obtained lower values of the time integral of the cost function in the two modeling methodologies (i.e., partially-connected and fully-connected), indicating faster convergence and lower energy consumption.

In Chapter 5, we used the Rademacher complexity approach for vector-valued functions

to create an upper generalization error bound for partially-connected RNN models. The theoretical base connecting a RNN model's accuracy to its architecture was proposed and proved. Open-loop simulations utilizing a complex two-reactors-in-series process example were performed to demonstrate the superior model accuracy achieved by 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, adopting the partially-connected RNN model was shown to yield smoother state trajectories with less loss function values (i.e., smaller mean squared error values), and the applied inputs produced less oscillatory behavior.

Bibliography

[1] 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.

[2] A. Agarwal, Y.A. Liu, and C. McDowell. 110th anniversary: ensemble-based machine learning for industrial fermenter classification and foaming control. *Industrial & Engineering Chemistry Research*, 58:16719–16729, 2019.

[3] 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.

[4] R. Alexander, G. Campani, S. Dinh, and F.V. Lima. Challenges and opportunities on nonlinear state estimation of chemical and biochemical processes. *Processes*, 8:1462, 2020.

[5] M. 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.

[6] M. Alhajeri and M. Soroush. Tuning guidelines for model-predictive control. *Industrial & Engineering Chemistry Research*, 59:4177–4191, 2020.

[7] 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.

[8] 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.

[9] J. M. Ali, N. H. Hoang, M. A. Hussain, and D. Dochain. Review and classification of recent observers applied in chemical process systems. *Computers & Chemical Engineering*, 76:27–41, 2015.

[10] 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.

[11] 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 scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[12] 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.

[13] M. S. F. Bangi and J. S. Kwon. Deep hybrid modeling of chemical process: Application to hydraulic fracturing. *Computers & Chemical Engineering*, 134:106696, 2020.

[14] J. C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20:247–260, 1996.

[15] E. F. Camacho and C. B. Alba. *Model Predictive Control*. Springer Science & Business Media, 2013.

[16] 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.

[17] A. Canning and E. Gardner. Partially connected models of neural networks. *Journal of Physics A: Mathematical and General*, 21:3275, 1988.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6:911–917, 1995.

[23] T. WS Chow and Y. Fang. A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. *IEEE Transactions on Industrial Electronics*, 45:151–161, 1998.

[24] H. Chun, E. Lee, K. Nam, J. Jang, W. Kyoung, S. Noh, and B. Han. First-principle-data-integrated machine-learning approach for high-throughput searching of ternary electrocatalyst toward oxygen reduction reaction. *Chem Catalysis*, 1:855–869, 2021.

[25] 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.

[26] S. F. De Azevedo, B. Dahm, and F. R. Oliveira. Hybrid modelling of biochemical processes: A comparison with the conventional approach. *Computers & Chemical Engineering*, 21:S751–S756, 1997.

[27] 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.

[28] D. Dochain. State and parameter estimation in chemical and biochemical processes: a tutorial. *Journal of Process Control*, 13:801–818, 2003.

[29] Picard D. Drgoňa, J., M. Kvasnica, and L. Helsen. Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216, 2018.

[30] M. Ellis, J. Liu, and P. Christofides. *Economic model predictive control*, volume 5. Springer, Switzerland, 2017.

[31] M. J. Ellis and V. Chinde. An encoder–decoder lstm-based empc framework applied to a building hvac system. *Chemical Engineering Research and Design*, 160:508–520, 2020.

[32] 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.

[33] 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.

[34] J. Fan and M. Han. Nonliear model predictive control of ball-plate system based on gaussian particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–6. Brisbane, Australia, 2012.

[35] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059, New York, USA, 2016.

[36] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1019–1027, Barcelona, Spain, 2016.

[37] H. Ge, Y. Liang, and M. Marchese. A modified particle swarm optimization-based dynamic recurrent neural network for identifying and controlling nonlinear systems. *Computers & structures*, 85:1611–1622, 2007.

[38] S. Sam Ge and C. Wang. Adaptive neural control of uncertain mimo nonlinear systems. *IEEE Transactions on Neural Networks*, 15:674–692, 2004.

[39] D. Ghosh, E. Hermonat, P. Mhaskar, S. Snowling, and R. Goel. Hybrid modeling approach integrating first-principles models with subspace identification. *Industrial & Engineering Chemistry Research*, 58:13533–13543, 2019.

[40] K. Gurney. *An Introduction to Neural Networks*. CRC press, 2018.

[41] M. P. R. Haaker and Peter J. T. Verheijen. Local and global sensitivity analysis for a reactor design with parameter uncertainty. *Chemical Engineering Research and Design*, 82:591–598, 2004.

[42] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in Neural Information Processing Systems*, 31:8527–8537, 2018.

[43] 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.

[44] H. Hassanpour, B. Corbett, and P. Mhaskar. Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chemical Engineering Research and Design*, 161:26–37, 2020.

[45] 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.

[46] Y. Hsu and J. Wang. A wiener-type recurrent neural network and its control strategy for nonlinear dynamic applications. *Journal of Process Control*, 19:942–953, 2009.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] J. Krishnaiah, C.S. Kumar, and M.A. Faruqi. Modelling and control of chaotic processes through their bifurcation diagrams generated with the help of recurrent neural network models: Part 1—simulation studies. *Journal of Process Control*, 16:53–66, 2006.

[53] S. N. Kumpati, P. Kannan, et al. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27, 1990.

[54] M. J. Kurtz and M. A. Henson. State and disturbance estimation for nonlinear systems affine in the unmeasured variables. *Computers & Chemical Engineering*, 22:1441–1459, 1998.

[55] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[56] Y. Kwon, D. Lee, Y. Choi, and S. Kang. Uncertainty-aware prediction of chemical reaction yields with graph neural networks. *Journal of Cheminformatics*, 14:1–10, 2022.

[57] C. Lee. Fuzzy logic in control systems: fuzzy logic controller. I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:404–418, 1990.

[58] D. Lee, A. Jayaraman, and J. S. Kwon. Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling. *PLOS Computational Biology*, 16:e1008472, 2020.

[59] S. Liao. Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, 28:93–103, 2005.

[60] F. V. Lima and J. B. Rawlings. Nonlinear stochastic modeling to improve state estimation in process monitoring and control. *AIChE Journal*, 57:996–1007, 2011.

[61] Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems & Control Letters*, 16:393–397, 1991.

[62] Y. Lu, M. Rajora, P. Zou, and S. Liang. Physics-embedded machine learning: case study with electrochemical micro-machining. *Machines*, 5:4, 2017.

[63] 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.

[64] Y. Ma, D. A. Noreña-Caro, A. J. Adams, T. B. Brentzel, J. A. Romagnoli, and M. G. Benton. Machine-learning-based simulation and fed-batch control of cyanobacterial-phycocyanin production in plectonema by artificial neural network and deep reinforcement learning. *Computers & Chemical Engineering*, 142:107016, 2020.

[65] J. Madar, J. Abonyi, and F. Szeifert. Feedback linearizing control using hybrid neural networks identified by sensitivity approach. *Engineering Applications of Artificial Intelligence*, 18:343–351, 2005.

[66] T. F. McKenna, S. Othman, G. Fevotte, A. M. Santos, and H. Hammouri. An integrated approach to polymer reaction engineering: a review of calorimetry and state estimation. *Polymer Reaction Engineering*, 8:1–38, 2000.

[67] A. Mesbah, A EM Huesman, H JM Kramer, and P MJ Van den Hof. A comparison of nonlinear observers for output feedback model-based control of seeded batch crystallization processes. *Journal of Process Control*, 21:652–666, 2011.

[68] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT press, 2018.

[69] A. Narasingam and J. S. Kwon. Koopman lyapunov-based model predictive control of nonlinear chemical process systems. *AIChE Journal*, 65:e16743, 2019.

[70] 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.

[71] R Oliveira. Combining first principles modelling and artificial neural networks: a general framework. *Computers & Chemical Engineering*, 28:755–766, 2004.

[72] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. *Automatica*, 46:647–656, 2010.

[73] 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.

[74] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

[75] 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.

[76] S. C. Patwardhan, S. Narasimhan, P. Jagadeesan, B. Gopaluni, and S. L. Shah. Nonlinear bayesian state estimation: A review of recent developments. *Control Engineering Practice*, 20:933–953, 2012.

[77] G. Porru, C. Aragonese, R. Baratti, and A. Servida. Monitoring of a CO oxidation reactor through a grey model-based EKF observer. *Chemical Engineering Science*, 55:331–338, 2000.

[78] A. Radke and Z. Gao. A survey of state and disturbance observers for practitioners. In *Proceedings of the American Control Conference*, pages 5183–5188, Minneapolis, Minnesota, 2006.

[79] 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.

[80] J. B. Rawlings and C. T. Maravelias. Bringing new technologies and approaches to the operation and control of chemical process systems. *AIChE Journal*, 65:e16615, 2019.

[81] 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.

[82] R. Roelofs. *Measuring Generalization and Overfitting in Machine learning*. Doctoral Dissertation, University of California, Berkeley, 2019.

[83] 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.

[84] 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.

[85] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[86] K. Schütt, P. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, 30:992–1002, 2017.

[87] D. Shah, J. Wang, and Q. P. He. Feature engineering in big data analytics for iot-enabled smart manufacturing–comparison between deep learning and statistical learning. *Computers & Chemical Engineering*, 141:106970, 2020.

[88] 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.

[89] A. K. Singh and H. P.and S. Mishra Singh. Validation of ANN-based model for binary distillation column. In *Proceeding of International Conference on Intelligent Communication, Control and Devices*, pages 235–242, Singapore, 2017.

[90] E. D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13:117–123, 1989.

[91] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

[92] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: A review. *Computers & Chemical Engineering*, 20:743–791, 1996.

[93] K. Takahashi and Y. Tanaka. Material synthesis and design from first principle calculations and machine learning. *Computational Materials Science*, 112:364–367, 2016.

[94] M. L. Thompson and M. A. Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40:1328–1340, 1994.

[95] Y. Tian, J. Zhang, and J. Morris. Modeling and optimal control of a batch polymerization reactor using a hybrid stacked recurrent neural network model. *Industrial & Engineering Chemistry Research*, 40:4525–4535, 2001.

[96] 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.

[97] A. P. Trischler and G. M. T. D'Eleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *Neural Networks*, 80:67–78, 2016.

[98] A. Y. Tsen, S. S. Jang, D. S. H. Wong, and B. Joseph. Predictive control of quality in batch polymerization using hybrid ann models. *AIChE Journal*, 42:455–465, 1996.

[99] A. Van Mulders, J. Schoukens, M. Volckaert, and M. Diehl. Two nonlinear optimization methods for black box identification compared. *Automatica*, 46:1675–1681, 2010.

[100] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65:466–478, 2019.

[101] R. Vepa. A review of techniques for machine learning of real-time control strategies. *Intelligent Systems Engineering*, 2:77–90, 1993.

[102] M. Viberg. Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, 31:1835–1851, 1995.

[103] M. Von Stosch, R. Oliveira, J. Peres, and S. F. de Azevedo. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Computers & Chemical Engineering*, 60:86–101, 2014.

[104] 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.

[105] 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*, 6:1–39, 2020.

[106] N. C. Wei, M. A. Hussain, and A. K. A. Wahab. Control of a batch polymerization system using hybrid neural network-first principle model. *The Canadian Journal of Chemical Engineering*, 85:936–945, 2007.

[107] J. A. Wilson and L. F. M. Zorzetto. A generalised approach to process state estimation using hybrid artificial neural network/mechanistic models. *Computers & Chemical Engineering*, 21:951–963, 1997.

[108] Z. T. Wilson and N. V. Sahinidis. The alamo approach to machine learning. *Computers & Chemical Engineering*, 106:785–795, 2017.

[109] W. Wong, E. Chee, J. Li, and X. Wang. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6:242, 2018.

[110] H. Wu and J. Zhao. Deep convolutional neural network model based chemical process fault diagnosis. *Computers & Chemical Engineering*, 115:185–197, 2018.

[111] 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.

[112] Z. Wu and P. D. Christofides. Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7:494, 2019.

[113] Z. Wu, J. Luo, D. Rincon, and P. 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.

[114] 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.

[115] Z. Wu, D. Rincon, Q. Gu, and P. D. Christofides. Statistical machine learning in model predictive control of nonlinear processes. *Mathematics*, 9:1912, 2021.

[116] Z. Wu, D. Rincon, J. Luo, and P. D. Christofides. Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE Journal*, 67:e17164, 2021.

[117] Z. Wu, A. Tran, Y. M. Ren, C. S. Barnes, S. Chen, and P. D. Christofides. Model predictive control of phthalic anhydride synthesis in a fixed-bed catalytic reactor via machine learning modeling. *Chemical Engineering Research and Design*, 145:173–183, 2019.

[118] 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.

[119] 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.

[120] 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.

[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] S. Yang, P. Navarathna, S. Ghosh, and B. W. Bequette. Hybrid modeling in the era of smart manufacturing. *Computers & Chemical Engineering*, 140:106874, 2020.

[123] 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.

[124] S. Yin and O. Kaynak. Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103:143–146, 2015.

[125] L. A. Zadeh. Probability measures of fuzzy events. *Journal of Mathematical Analysis and Applications*, 23:421–427, 1968.

[126] N. J. Zambare, M. Soroush, and M. C. Grady. Real-time multirate state estimation in a pilot-scale polymerization reactor. *AIChE Journal*, 48:1022–1033, 2002.

[127] M. Zeitz. The extended Luenberger observer for nonlinear systems. *Systems & Control Letters*, 9:149–156, 1987.

[128] S. Zendehboudi, N. Rezaei, and A. Lohi. Applications of hybrid models in chemical, petroleum, and energy systems: A systematic review. *Applied Energy*, 228:2539–2566, 2018.

[129] Z. Zhang, Z. Wu, D. Rincon, and P. D. Christofides. Real-time optimization and control of nonlinear processes using machine learning. *Mathematics*, 7:890, 2019.