UNIVERSITY OF CALIFORNIA

Los Angeles

Machine Learning Modeling for Process Control and Electrochemical Reactor Operation

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

by

Junwei Luo

2023

ABSTRACT OF THE DISSERTATION

Machine Learning Modeling for Process Control

and Electrochemical Reactor Operation

by

Junwei Luo

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2023

Professor Panagiotis D. Christofides, Chair

Electrochemical reaction processes attract increasing attention as a promising chemical process alternative to achieve green and sustainable chemical manufacturing due to its property that can produce chemical products without directly burning fossil fuels. Among various electrochemical reaction processes, CO_2 reduction provides a possibility to capture the CO_2 gas in the atmosphere and effectively relieve the global warming crisis. However, due to the complex, stochastic, and nonlinear nature of the electrochemical reactions, modeling an electrochemical process using first-principles is very challenging. To investigate the physical-chemical phenomena of the electrochemistry of reduction of CO_2, the Laboratory of Electrochemical System Engineering at

UCLA has developed a gastight rotating cylinder electrode (RCE) cell to decouple mass transport phenomena from the intrinsic kinetics of the electrochemical reactions. Specifically, this RCE cell allows manipulation of two variables, the applied potential and rotation speed of the cylindrical electrode, which determines the mass transport profile and reaction kinetics of the electrochemical reactions, respectively. This design further enables the development of an advanced process control system for the reactor to control the desired output states by adjusting the two key input variables.

However, the absence of the first-principle model for the electrochemical reactor is a significant challenge to develop the process control system. To this end, considering the complexity of the process, neural network (NN) models are developed and used in this thesis work to capture the input-output relationship of the reactor and provide a data-based alternative to the unavailable, first-principle models. On the other hand, in a data-driven approach, the performance of the NN model is determined by the quality of the training data. Therefore, the inevitable noise, caused by experimental uncertainty, can corrupt the collected data and negatively impact the NN modeling task. Motivated by this concern, various methods, such as the Monte Carlo dropout and co-teaching algorithm, are adopted in this thesis to improve the robustness of the NN models against noisy data.

Eventually, this dissertation demonstrates an advanced model predictive (MPC) scheme based on experimental data-driven NN models that capture the input-output relationship of the electrochemical reactor. Furthermore, the Koopman operator method is adopted to perform on-line linearization to the NN model to improve the computational efficiency of the MPC and enable its real-time application to the experimental electrochemical reactor. Finally, simulations, open-, and closed-loop experiments are conducted to demonstrate the overall implementation and successful

performance of the proposed NN models and MPC schemes.

The dissertation of Junwei Luo is approved.

Carlos G. Morales-Guio

Dante A. Simonetti

Xiaochun Li

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2023

Contents

List of Figures

List of Tables

Acknowledgements

I would like to deliver my deepest gratitude to my advisor, Professor Panagiotis D. Christofides, for his support, encouragement, and guidance throughout my academic journey over the past four years. Professor Christofides exemplifies excellence as a researcher, mentor, instructor, and role model. I consider myself fortunate to be his student, as this Ph.D. experience has provided me with a solid foundation for my future career. I would also like to extend my sincere gratitude to my doctoral committee: Professor Morales-Guio, Professor Simonetti, and Professor Li for their time, comments, and valuable advice.

In addition, I would like to thank all of my colleagues in the Christofides research group, including Dr. David Rincon, Dr. Zhihao Zhang, Dr. Yangyao Ding, Dr. Scarlett Chen, Dr. Yichi Zhang, Professor Mohammed Alhajeri, Dr. Sungil Yun, Aisha Alnajdi, Vito Canuso, Matthew Tom, Feiyang Ou, Atharva Suryavanshi, and Yash Kadakia. I would like to particularly thank Professor Zhe Wu, Dr. Yi Ming Ren, Fahim Abdullah, Berkay Citmaci, Derek Richard, and Joon Baek Jang who worked closely with me to tackle significant challenges in my Ph.D. journal.

Last but not least, I would like to deliver my thanks to my family for their endless love and support. Especially to my soul mate, Kaiwen Yang, for her unwavering support, love, and encouragement throughout this journey.

Curriculum Vitae

Education

Virginia Polytechnic Institute and State University Sep 2015 - May 2019
B.Eng., Chemical Engineering **Blacksburg, Virginia**

Publications

1. D. Richard, J. Jang, B. Çıtmacı, J. Luo, V. Canuso, P. Korambath, O. Morales-Leslie, J. Davis, H. Malkani, P.D. Christofides, C.G. Morales-Guio, "Smart manufacturing inspired approach to research, development, and scale-up of electrified chemical manufacturing systems", Iscience, 26 (6), 2023
2. B. Çıtmacı, J. Luo, J. Jang, C.G. Morales-Guio, P.D. Christofides, "Machine learning-based ethylene and carbon monoxide estimation, real-time optimization, and multivariable feedback control of an experimental electrochemical reactor", *Chem. Eng. Res. & Des.*, 191, 658-681, 2023
3. B. Çıtmacı, J. Luo, J. Jang, C.G. Morales-Guio, P.D. Christofides, "Machine learning-based ethylene and carbon monoxide estimation, real-time optimization, and multivariable feedback control of an experimental electrochemical reactor", *Computer Aided Chemical Engineering* 52, 1519-1524, 2023
4. B. Çıtmacı, J. Luo, J. Jang, P. Korambath, C.G. Morales-Guio, J. Davis, P.D. Christofides, "Digitalization of an experimental electrochemical reactor via the smart manufacturing innovation platform", *Digital Chemical Engineering*, 5, 100050, 2022
5. B. Çıtmacı, J. Luo, J. Jang, V. Canuso, D. Richard, Y. Ren, C.G. Morales-Guio, P.D. Christofides, "Machine learning-based ethylene concentration estimation, real-time optimization and feedback control of an experimental electrochemical reactor", *Chem. Eng. Res. & Des.*, 185, 87-107, 2022
6. M. Alhajeri, J. Luo, Z. Wu, F. Albalawi, P.D. Christofides, "Process structure-based recurrent neural network modeling for predictive control: A comparative study", *Chem. Eng. Res. & Des.*, 179, 77-89, 2022
7. J. Luo, V. Canuso, J. Jang, Z. Wu, C.G. Morales-Guio, P.D. Christofides, "Machine learning-based operational modeling of an electrochemical reactor: Handling data variability and improving empirical models", *Comp. & Chem. Eng.*, 161, 107757, 2022
8. S. Yun, M. Tom, J. Luo, G. Orkoulas, P.D. Christofides, "Microscopic and Data-Driven Modeling and Operation of Thermal Atomic Layer Etching of Aluminum Oxide Thin Films", *Comp. Eng. Res. & Des.*, 177, 96-107
9. Z. Wu, J. Luo, D. Rincon, P.D. Christofides, "Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator", *Comp. & Chem. Eng.*, 168, 275-287, 2021

Chapter 1

Introduction

1.1 Motivation

The climate change and the global warming crises are becoming one of the most significant challenges in human history, threatening everyone. The Earth's climate is undergoing unprecedented changes due to human activities, primarily the excessive release of greenhouse gases into the atmosphere. These gases, mainly carbon dioxide (CO_2), act as a blanket trapping heat from the sun and causing a rise in global temperatures. This phenomenon, known as global warming, has far-reaching and devastating consequences. Recently, we have witnessed clearer signs of the significant consequences of the global warming crisis, including the melting of polar ice caps, extreme heat waves, and more frequent and intense hurricanes and floating, posing threats to ecosystems, agriculture, and human lives.

Unlike the classical chemical manufacturing industry, electrochemical reactions offer a promising approach to produce chemical products directly using electricity as the energy source instead of burning fossil fuels. Simultaneously, advancements are being made in generating electricity from

clean, green, and renewable resources [1, 8, 185]. Considering the aforementioned scenarios, the chemical manufacturing process can reduce its reliance on fossil fuels, which account for 90% of greenhouse gas emissions from chemical plants in Europe, according to a report in 2013 [18, 35]. Moreover, beyond the reduction of CO_2 emissions, an electrochemical reactor holds the potential to capture atmospheric CO_2 and convert it back into valuable chemical products using electricity generated from renewable resources.

However, in recent years, the operation of electrochemical reactors is still limited to the laboratory scale. One of the main obstacles is the incomplete understanding of the underlying reaction mechanisms. Electrochemical reactions often involve complex mechanisms; for example, considering the electrochemical reaction catalyzed with copper to convert CO_2, multiple research works have proposed unique explanations for the reaction mechanism [114]. Consequently, the development of a comprehensive first principles model that accurately describes these processes is still underway.

On the other hand, with the exponential growth of computational power and data science technology, data-driven approaches, especially machine learning (ML) and deep learning methods, have become increasingly accessible and vital in the field of modeling. In [64], the neural network (NN) model is interpreted as a universal approximation to any nonlinear process. Furthermore, [7, 169] have proposed methods to systematically analyze the generalization errors of NN models, enhancing their accuracy and reliability. Therefore, ML and NN methods are considered reliable and efficient options to capture critical nonlinear relationships in chemical processes, enabling the development of process models and the investigation of the underlying chemical-physical phenomena driving chemical reactions. These approaches can be used to explore electrochemical systems

and facilitate the scaling-up of electrochemical reactors. Specifically, considering the power of ML and NN modeling, process control systems can be developed for the electrochemical reactor without waiting for a fully advanced first principles-based model.

The initial challenge of using ML/NN modeling lies in effectively handling noisy data, considering the stochastic nature of electrochemical reactions and the inevitable experimental errors inherent in data collection. It is widely recognized that the quality of data significantly impacts the performance of data-driven models. If the data is excessively noisy or contaminated by critical experimental errors, the model may fail to accurately capture the desired relationships. Thus, the development of robust NN models capable of handling noisy data is essential for the successful application of data-driven methods in real-world scenarios. By addressing this challenge, data-driven approaches can be a feasible tool towards the practical implementation and control of electrochemical reactors.

Moreover, the achievement in [71] provided solid support towards designing and evaluating a NN model-based process control scheme for an operational electrochemical reactor. Specifically, valuable experimental data collected from over-three years of experiments provided a rich dataset to develop an NN model. Furthermore, the availability of a fully functioning rotating cylinder electrode (RCE) cell, capable of converting CO_2 into diverse carbon-based products, enabled the physical evaluation of the designed process control system. By implementing the NN-based process control system in an operational reactor, many technical details, such as data flow, hardware connection, delayed response, and computational time limit, which are often overlooked in simulation-based studies, were meticulously addressed. In summary, motivated by the recent developments in computational science technology and reactor design, this work aims to conduct an

interdisciplinary study to develop a process control system for the electrochemical reactor. Ultimately, this work is expected to make a valuable contribution towards combating the climate crisis by effectively closing the anthropogenic carbon cycle through the fusion of artificial intelligence (AI) and electrochemistry.

1.2 Background

Recent research breakthroughs have revealed the transformative potential of electrochemical CO_2 reduction in producing valuable chemical products, such as ethylene, ethanol, acetic acid, and n-propanol [34, 159, 160]. By harnessing this innovative method and replacing classical chemical processes that rely on fossil fuels, a significant reduction in carbon emissions can be achieved. Traditionally, these processes contribute significantly to greenhouse gas emissions due to the burning of fossil fuels for energy. With electrochemical CO_2 reduction, the utilization of electricity as the primary energy source offers a potential solution that will decarbonize the chemical industry [172]. As of today, copper-based catalysts are the only options for catalysts to electrochemically reduce CO_2 to C_{2+} products, because the CO_2 gas particles need to be reduced to CO in order to trigger the rest of the transformations that generate C_{2+} products [127, 188]. Furthermore, [75, 139] performed techno-economic evaluations for the electrochemical conversion of CO_2, which pointed out that having the price of electricity below $0.03\ \$/kWh$ is pivotal for ensuring the economic viability of electrochemical CO_2 conversion. Encouragingly, the scale of electricity generated from renewable resources has grown significantly, with global photovoltaics for solar power increasing by about 100 GW/year in 2020 [53] and cumulative wind power capacity reaching 589,872 MW

worldwide in 2018 [163]. These developments are continuously driving down the price of electricity generated from renewable sources. Considering the challenge and potential of this technology, [134] proposed a four-phase approach to scale up electrochemical CO_2 processes to an industrial level by leveraging smart manufacturing concepts. However, [172] emphasized the issue of low selectivity in the current electrochemical CO_2 process and underscored the need for a deeper understanding of the reaction mechanism to address this challenge effectively. Likewise, [134] discussed the drawbacks arising from the complex and intricate nature of the complete reaction, and they proposed the use of data-driven methods to tackle this problem.

The investigation of data-driven modeling utilizing AI techniques in chemical engineering has been carried out continuously. Practical data-driven modeling tools have been proposed in the 20th century, such as fuzzy logic in the 1960s ([183]), expert systems in the 1980s ([90, 96]), and ML in the 1990s ([156]). Among them, classic ML techniques demonstrated outstanding regression ability and have been widely applied to process modeling tasks. For example, [171] utilized the support vector regression (SVR) method to model heating, ventilation, and air-conditioning (HVAC) systems. Two SVR models were developed to fit the two outputs of the HVAC system (e.g., room temperature and room relative humidity) using two inputs (e.g., fan speed and chilled water valve opening), and the SVR model accurately captured the complex nonlinear input-output relationship for the HVAC system. In [189], the Gaussian process (GP) regression method was used to model a nonlinear vehicle system. The GP model was used to develop an MPC to perform safety control. [16] proposed an adaptive model identification method using the Sparse Identification of Nonlinear Dynamics (SINDy) for system identification, and ML-based feature selection and fine-tuning. The adaptive model identification method was demonstrated to require less data

to develop than SINDy and was applied to identify the dynamics of a Continuous Stirred Tank Reactor (CSTR) simulation. Generally, ML models can be used to effectively capture nonlinear relations, especially when the scale of the data set is small. However, in the case that sufficient data is available, deep learning methods, such as artificial neural network (ANN) structures, usually have superior performance in capturing nonlinear and complex systems.

With the significant growth in sensors, cloud computing, and database technologies, obtaining enough data for ANN modeling becomes a less stringent requirement. As a result, ANNs have been widely used in various research directions for engineering and process control. [11] adopted a feed-forward neural network (FNN) model to predict the percentage fatty acid methyl ester (%FAME) yield in a castor oil transesterification process using the four inputs (i.e., methanol to oil molar ratio, catalyst amount, temperature, and time). The developed FNN model was subsequently used to provide insights for estimating parameters for an explicit kinetic model. Due to the high prediction accuracy given by the NN model, researchers also used it as a "soft sensor", which refers to using measurable information to estimate the value of unmeasurable states in real-time. For example, [36] developed a soft sensor using the ANN model to predict the properties of the polymer product and used it to provide information for a feedback control system. Although the simple FNN model has demonstrated its ability to capture complex nonlinear process systems and its potential to be applied in process control systems, the recurrent neural network (RNN) model is widely admitted to be more suitable for processing time-series data. Thus, the RNN is more appropriate for process modeling.

For example, in 2018, [164] presented a study using an RNN model to develop an MPC for pharmaceutical manufacturing. Specifically, a two-layer long short-term memory (LSTM) net-

work model was developed to capture a non-linear system, which consisted of a single CSTR reactor carrying out a chain of reversible reactions in pharmaceutical manufacturing. The LSTM was shown to provide accurate predictions for the nonlinear process and was used to develop an effective RNN-MPC model evaluated by closed-loop simulations. In 2019, [167] elaborated the application of LSTM models to Lyapunov-based MPC, and provided a rigorous theoretical analysis of its closed-loop stability. More recently, in 2023, [73] proposed using a multiple timescale recurrent neural network (MTRNN) to account for the complex multi-timescale properties in a nonlinear chemical system and developed an MPC based on the MTRNN model. These researches showed that the integration of RNN modeling and MPC plays an important role in the advanced process control area and is still an active field of research. In addition to the RNN model, [26] proposed the neural ordinary differential equation (NODE) in 2018, which is considered a potentially better method to capture dynamic physical systems as a continuous approximation method, and it was demonstrated to be more robust to irregular sampled time-series data than RNNs [138]. The NODE model has been applied to various engineering research. [25] utilized NODE to develop a hybrid model which worked as the process model in MPC. Additionally, NODE was also utilized to identify physical systems [20, 92, 117].

Although many achievements have been obtained for ML-based MPC research, there is a critical drawback to its application in practical systems, which is the computational complexity. Specifically, the MPC scheme requires solving an optimization problem based on the process model to compute the optimal control actions. However, a non-linear ML model yields a nonlinear, non-convex optimization problem, which is complex and requires a clear and effective solution that has not been developed as of now. One promising method to account for this problem is

to develop a systematic way to linearize the ML model, such that the associated MPC can be significantly simplified (into a quadratic programming problem in most cases). In [179], the Taylor expansion method was utilized to linearize an RNN with nonlinear autoregressive exogenous architecture (NARX RNN), which was developed to model an air-conditioning and mechanical ventilation (ACMV) system. The proposed method was tested with closed-loop experiments, which demonstrated good performance from the linearized RNN-MPC. Furthermore, [143] provided a comprehensive review of the most recent developments and applications of MPC in the engineering domain, including additional examples for the application of MPC based on a linearized data-driven model.

1.3 Dissertation Objectives and Structure

This dissertation presents an ML-based methodology to implement advanced process control for electrochemical reactors, which involves data mining, NN modeling, process simulation, MPC, and experiments. The objectives of this dissertation can be summarized as follows:

1. To investigate novel NN designs that can be used to account for noisy datasets caused by experimental uncertainty and missing measurements. Additionally, their performance was evaluated through reliable simulations.

2. To develop a NN model capturing the input-output relationship of an operating electrochemical reactor and use it to provide insights to facilitate the development of a first principles-based model of the reactor.

3. To present an effective MPC framework based on the linearization of the NN model, which enabled a systematic approach to establish a practical data-driven MPC system to control operational electrochemical reactors.

4. To demonstrate the performance of the linearized NN-based MPC system with closed-loop experiments on an operational electrochemical reactor.

The remaining parts of this dissertation is organized as follows:

Chapter 2 presented a dropout method and a co-teaching learning algorithm that developed LSTM models to capture the ground truth (i.e., underlying process dynamics) phenomena of a nonlinear system from noisy data. A large-scale process simulator, Aspen Plus Dynamics, was used to generate process data based on an industrial chemical reactor example. The generated dataset was corrupted by sensor noise, which was determined using industrial data, to evaluate the performance and robustness of the proposed modeling approaches. The dropout method, a widely used method to reduce the overfitting of LSTM models, was adapted to account for noisy data. Another approach termed the co-teaching method required training LSTM models with additional noise-free data, which was generated from the evolution of a first principles model that employed several standard modeling assumptions. Through open and closed-loop simulations, the improvement of the model prediction accuracy was demonstrated.

Chapter 3 represented an approach towards developing an MPC based on the NODE, which is a recently proposed family of deep learning models that can perform a continuous approximation of a linear/nonlinear dynamic system by integrating the NN model with classical ordinary differential equation solvers. In the area of using NN for process modeling tasks, RNNs have become a popular

option and demonstrated their capability and superiority in process time-series data. RNNs have been utilized to design model predictive control (MPC) systems in various works. However, as a discrete approximation model, RNN requires strictly uniform step sequence data to operate, which makes it less robust to an irregular sampling scenario, such as missing data points during the operation due to sensor failure or other types of random errors. This chapter demonstrated an alternative approach for developing an MPC based on this novel continuous NN model. A chemical process example was utilized to demonstrate the performance of the NODE-based MPC. Furthermore, the performance of the NODE-based MPC under Gaussian and non-Gaussian noise was investigated, and the subsampling method was found to be effective against non-Gaussian noise.

Chapter 4 demonstrated the application of an FNN model to capture the input-output relationship of an experimental electrochemical reactor from experimental data that are obtained from easy-to-implement sensors. This FNN model was computationally efficient and can be used in real-time to determine energy-optimal reactor operating conditions. To account for the uncertainty of the experimental data, inspired by the maximum likelihood estimation (MLE) method, a statistical weighted-FNN was constructed to prevent over-fitting to uncertain data. Furthermore, an optimization framework was proposed to facilitate the development of an empirical, first principles (EFP) model using the insights from the weighted-FNN model. The weighted-FNN model developed in this chapter was used to construct a real-time optimizer (RTO) for the electrochemical reactor.

Chapter 5 developed an MPC scheme that utilized an on-line linearization of an RNN as the process model to implement real-time multi-input-multi-output (MIMO) control in the electro-

chemical reactor. Specifically, an LSTM model was developed from the experimental data of the electrochemical reactor to capture a dynamic time-series response of the electrochemical reactor. Instead of directly using the LSTM model as the process model for an MPC, the Koopman operator method was utilized to perform on-line linearization of the LSTM model, such that the optimization problem in the MPC was simplified into a quadratic programming problem. The performance of the LSTM model, Koopman-based optimization, and MPC using the linearization of the LSTM model were evaluated with various simulations as well as open and closed-loop experiments.

Chapter 2

Machine Learning-based Predictive Control Using Noisy Data: Evaluating Performance and Robustness via a Large-Scale Process Simulator

2.1 Introduction

ML has attracted an increasing level of attention in classical engineering fields in recent years due to its ability of analyzing big data from industrial processes. ML techniques such as neural networks and their variants have been successfully applied in process modeling, process monitoring, and fault detection, which fall into the categories of regression and classification problems. Among many types of neural networks, recurrent neural networks (RNN), and long short-term memory (LSTM) networks have become popular for modeling nonlinear dynamic systems from

time-series data, and have been incorporated in model predictive control (MPC) to predict evolution of process states when first-principles process models are unavailable. While many research works have studied neural network modeling of chemical processes using noise-free data, learning using noisy data is a practically challenging task due to the high capacity of neural network to fit noisy data (i.e., overfitting). Considering that the sensor measurements in chemical plants are commonly affected by noise in real-time operation, ML modeling of chemical processes using industrial noisy data remains an important research topic.

One way to handle noisy measurements in linear dynamic systems is Kalman filtering, e.g., [119]. Additionally, many other methods such as moving horizon estimation and unscented Kalman filter have been proposed to deal with data noise [119]. In the state estimation methodology, to establish a correct estimation, a model representation is generally needed and the covariance matrices need to be tuned as well [97]. Recently, the effect of learning with raw vibration signals from a laboratory-scale water flow system was studied using ML methods (i.e., LSTM and a feed-forward deep neural network) and a linear statistical learning approach (i.e., projection to latent structure, PLS) [144]. From their findings, it was found a poor performance from both ML methods and PLS when using raw vibration data and that further treatment of the data is needed for better model development. When exposing ML models to Gaussian noise, [180] has shown that ML models can efficiently learn the true process dynamics due to the dominant role of the internal states during the prediction step. However, as real-world data noise often follows a non-Gaussian distribution, ML models may undergo performance decay if no proper treatment is taken to handle data noise in realistic scenarios.

There are many works in the literature that study ML methods with different types of noise.

For example, it has been pointed out in [94] that assuming independent identically distributed (i.i.d.) noise is not realistic when modeling some chemical processes with Gaussian process (GP) regression. For that reason, the i.i.d. condition is relaxed to heteroscedastic noise for a ML structure in [94]. In [65], a Wiener-type recurrent neural network is tested with two types of noise, a white noise and a sinusoidal-type noise in order to evaluate robustness. In [86], the effect of Gaussian noise in RNN modeling of chaotic systems using short time-series data has been studied. Additionally, data preprocessing and smoothing techniques can be used to improve data quality. For example, noisy and redundant data are removed for an ensemble-based ML approach in a foaming control application in [6]. In order to perform data smoothing pretreatment and tackle missing data points, a third-order polynomial is implemented to the experimental data and later used with artificial neural networks to develop a deep reinforcement learning scheme that controls a bioreactor [102]. In [76], a linear filter is used to denoise the measurements in data-driven soft sensors when implemented with ML methods.

In terms of the applications of ML modeling approaches, the integration of ML models in model predictive controllers has attracted increasing attention in recent years. ML modeling and control of nonlinear processes under noise-free conditions have been explored in [58, 167], where the control performance depends greatly on the model accuracy. To handle noisy data subject to industrial data noise following a non-Gaussian distribution in ML modeling of nonlinear processes, Monte Carlo dropout and co-teaching methods have been utilized in [170] to develop LSTM models to capture the ground truth from noisy data. Specifically, the dropout method uses noisy data only and reduces the overfitting by randomly dropping out model weights at both training and testing phases. Co-teaching method uses noise-free data generated from simulations of first-principles

process model in addition to noisy data to improve model performance by accounting for noise-free pattern. However, as first-principles models are imperfect representations of reality due to simplified governing equations and boundary conditions, model mismatch may arise when noise-free data is generated based on first-principles solutions.

To evaluate the performance of the proposed machine leaning modeling approaches in the presence of mismatch between the industrial chemical process and its first-principles model, in this work, we consider a chemical reactor example simulated in the process simulator, Aspen Plus Dynamics, that allows to evaluate model mismatch and controller robustness to industrially-generated data noise. Chemical process simulators are widely used in industrial process design to provide more accurate steady-state and dynamic solutions than first-principles models due to their ready-to-use libraries involving thermodynamics properties, unit operations, and many other features [44]. In general, chemical process simulators can be classified into equation-oriented (e.g., EMSO software) and sequential modular approaches (e.g., Aspen Plus) [105]. In addition to process design, the integration of process control systems in processes simulators has been explored in the literature [146].

In this work, we generate noisy data using Aspen dynamic simulations, in which the process state measurements are corrupted by industrial sensor noise. Subsequently, the reactor first-principles model is developed in Matlab to generate noise-free data. The dropout LSTM model is trained using noisy data only, and the co-teaching LSTM model is trained using both noisy and noise-free data. The LSTM models are then incorporated in a Lyapunov-based model predictive controller that optimizes process performance while maintaining closed-loop system stability. Finally, we compare the dropout and co-teaching LSTM models with the LSTM model trained using

the standard learning algorithm and demonstrate their superiority in both open-loop and closed-loop operations. The rest of this chapter is organized as follows: in Section 2.2, the class of nonlinear systems, long short term memory networks, and machine-learning-based model predictive control scheme are presented. In Section 2.3, an overview of the dropout and co-teaching methods are provided. In Section 2.4, an industrial chemical reactor example is used to demonstrate the efficacy of the proposed ML modeling and control approaches.

2.2 Preliminaries

2.2.1 Notation

The Euclidean norm of a vector is denoted by the operator $|\cdot|$ and the weighted Euclidean norm of a vector is denoted by the operator $|\cdot|_Q$ where Q is a positive definite matrix. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$.

2.2.2 Class of Systems

We consider the class of continuous-time nonlinear systems described by the following system of first-order nonlinear ordinary differential equations:

$$\begin{aligned} \dot{x} &= F(x,u) := f(x) + g(x)u,\ x(t_0) = x_0 \\ y &= x + w \end{aligned} \tag{2.1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^m$ is the manipulated input vector, $y \in \mathbf{R}^n$ is the vector of state measurements that are sampled continuously, and $w \in \mathbf{R}^n$ is the noise vector. The input vector is constrained by $u \in U := \{u_i^{\min} \leq u_i \leq u_i^{\max}, i = 1, ..., m\} \subset \mathbf{R}^m$. $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times m$, respectively with $f(0)$ assumed to be zero such that the origin is a steady-state of the nominal (i.e., $w(t) \equiv 0$) system of Eq. 2.1 (i.e., $(x_s^*, u_s^*) = (0, 0)$, where x_s^* and u_s^* represent the steady-state state and input vectors, respectively). Throughout the manuscript, we assume that the full state measurements are continuously available at all times, and the initial time t_0 is taken to be zero ($t_0 = 0$).

2.2.3 Long Short Term Memory (LSTM) Model

Long short-term memory (LSTM) networks are a type of recurrent neural network (RNN) capable of modeling long-term dependencies in sequence prediction problems due to the design of three gates, i.e., the input gate, the forget gate, and the output gate, in the network structure. A schematic of LSTM network structure is shown in Fig. 2.1 [27]. In this work, the LSTM model is developed to predict the states of Eq. 2.1 given the control actions and the past noisy state measurements. Specifically, given the input sequence $m(k)$, $k = 1, ..., T$, where T is the number of measured states of the sampled-data system of Eq. 2.1, the following equations are used to

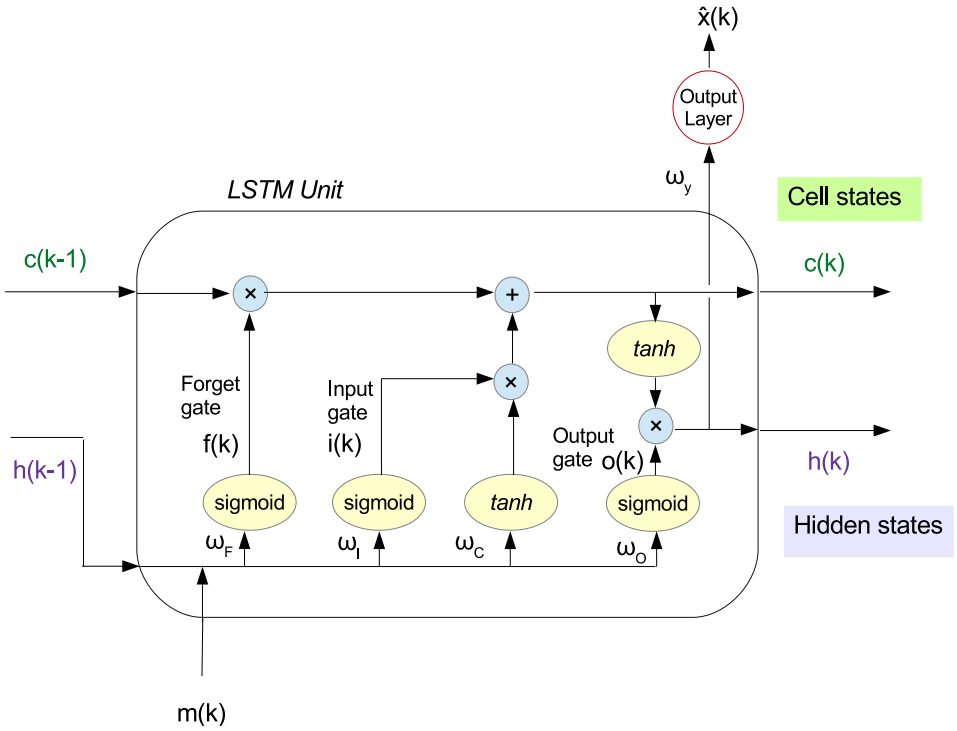


Figure 2.1: Schematic of LSTM units.

calculate the predicted output sequence $\hat{x}(k)$:

$$i(k) = \sigma(\omega_i^m m(k) + \omega_i^h h(k-1) + b_i) \tag{2.2a}$$

$$f(k) = \sigma(\omega_f^m m(k) + \omega_f^h h(k-1) + b_f) \tag{2.2b}$$

$$c(k) = i(k) tanh(\omega_c^m m(k) + \omega_c^h h(k-1) + b_c) + f(k)c(k-1) \tag{2.2c}$$

$$o(k) = \sigma(\omega_o^m m(k) + \omega_o^h h(k-1) + b_o) \tag{2.2d}$$

$$h(k) = o(k) tanh(c(k)) \tag{2.2e}$$

$$\hat{x}(k) = \omega_y h(k) + b_y \tag{2.2f}$$

where $m(k)$, $c(k)$, $h(k)$, $i(k)$, $f(k)$, and $o(k)$ are the input sequence, the cell state, the internal state, the outputs from the input gate, the forget gate, and the output gate, respectively. $\hat{x} \in \mathbf{R}^{n \times T}$ represent the LSTM network output sequences. The weight matrices for the LSTM input vector m, and the hidden state vector in the input gate are represented by ω_i^m and ω_i^h, respectively. Similarly, the weight matrices for the input vector m and hidden state vector h in calculating the cell state c, the forget gate f, and the output gate o are represented by ω_c^m, ω_c^h, ω_f^m, ω_f^h, ω_o^m, ω_o^h, respectively, with b_i, b_f, b_o, b_c representing the bias terms. Finally, the LSTM predicted state is calculated using Eq. 2.2f where ω_y and b_y denote the weight matrix and bias vector for the output, respectively. Since the LSTM model uses control actions and past state measurements to predict future states, the input sequence $m \in \mathbf{R}^{(n+m) \times T}$ contains the manipulated inputs $u \in \mathbf{R}^m$ and the past measured states $x \in \mathbf{R}^n$ within a certain period of time (i.e., T). The LSTM model uses the sigmoid activation function $\sigma(\cdot)$ and the hyperbolic tangent function $tanh(\cdot)$ as the nonlinear activation functions. Additionally, as LSTM networks are a type of recurrent neural networks, we can also present the LSTM model in the form of a continuous-time nonlinear system as follows:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T z \tag{2.3}$$

where $\hat{x} \in \mathbf{R}^n$ is the LSTM state vector, $u \in \mathbf{R}^m$ is the manipulated input vector, and $z = [z_1 \cdots z_{n+m+1}]^T = [\sigma(\hat{x}_1) \cdots \sigma(\hat{x}_n) \;\; u_1 \cdots u_m \;\; 1]^T \in \mathbf{R}^{n+m+1}$ is a vector of both the network states $\hat{x}$ and the inputs u. $\sigma(\cdot)$ represents nonlinear activation functions in each LSTM unit, and "1" represents the bias term. The diagonal matrix $A \in \mathbf{R}^{n \times n}$ and the matrix $\Theta \in \mathbf{R}^{(n+m+1) \times n}$ consist of the LSTM weights that will be optimized.

Training data can be generated from extensive open-loop simulations of the nonlinear system of Eq. 2.1 under various initial conditions and control actions. The system inputs 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, and the explicit Euler method is utilized with a sufficiently small integration time step $h_c < \Delta$ to integrate the continuous-time nonlinear system of Eq. 2.1 in simulations. Then, the LSTM model can be trained following the learning process as discussed in [167].

Remark 1. *The LSTM model of Eq. 2.3 is built for the network with one hidden layer (i.e., the network is formed with three layers: input layer, one hidden layer, and output layer). However, the LSTM development is not restricted to one hidden layer, and can be extended to multiple hidden layers for better approximation performance. Additionally, it should be noted that the LSTM internal states do not necessarily represent the actual process states in the first-principles model of Eq. 2.1. In this study, as the LSTM model is developed to predict future states given the past state measurements and manipulated inputs, the LSTM outputs correspond to the process states in the nonlinear system of Eq. 2.1.*

Remark 2. *Since the sensor measurements are now corrupted by industrial noise, the evolution of state variables can no longer rely on the current state measurement. Therefore, to reduce the dependence on the current state measurement, the ML models in this work are developed accounting for past (noisy) state measurements over a number of sampling periods to provide better predictions in a noisy environment.*

Remark 3. *As discussed in [170], the nonlinear activation functions such as $\tanh$ and $sigmoid$ functions are often used between the LSTM hidden layers to introduce nonlinearities into the net-*

work. Specifically, $sigmoid$ function is typically used for LSTM input/output/forget gates, and $tanh$ function is often used for the internal state and cell state. The linear unit is only used between the LSTM output layer and the last hidden layer. Since the second derivative of $tanh$ function decays slowly to zero, the vanishing gradient problem in the training of RNN and LSTM models could be tackled by using $tanh$ function. Although the saturating nonlinear activation function (i.e., $tanh$) can be limiting, it just means that the gradient near the boundaries will be small, hence taking smaller steps towards convergence; convergence is though guaranteed but at a slower pace. However, a non-saturating activation function like $ReLu$ might cause the LSTM model to diverge in the training.

Remark 4. *In addition to extensive open-loop simulations with different initial conditions and control actions, we can also generate ML datasets by simulating the system of Eq. 2.1 under a pseudo random input sequence to obtain a single continuous trajectory. Additionally, if the system is operated around an unstable steady-state, the holding period Δ for pseudo random input signals should be carefully chosen to make sure that the state is bounded in the stability region at all times. In Section 2.4, an Aspen Plus reactor example is used to demonstrate the data generation process using pseudo random input signals.*

Remark 5. *It should be mentioned that in this work, the LSTM model is trained using noisy data (i.e., the state measurements are corrupted by industrial data noise), which makes it challenging to obtain a well-conditioned LSTM model that can capture the ground truth (i.e., the underlying process dynamics of Eq. 2.1) using standard learning algorithms. Therefore, to handle the noisy training data, in Section 2.3, we propose two methods to improve model performance. The dropout*

method utilizes noisy data only and improves model performance by reducing overfitting to noisy data. The co-teaching method improves model prediction accuracy by taking advantage of noise-free data generated from computer simulations.

2.2.4 Model Predictive Control Using LSTM models

The LSTM model is incorporated in Lyapunov-based model predictive controller (LMPC) to provide state predictions in solving the MPC optimization problem. The formulation of LSTM-based MPC is presented as follows:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) dt \tag{2.4a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{2.4b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{2.4c}$$

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

$$\text{if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}} \tag{2.4d}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{2.4e}$$

where $\tilde{x}$, N and $S(\Delta)$ are the predicted state trajectory, the number of sampling periods in the prediction horizon, and the set of piecewise constant functions with period Δ. $\dot{V}(x, u)$ in Eq. 2.4d denotes the time-derivative of V, i.e., $\frac{\partial V(x)}{\partial x}(F_{nn}(x, u))$. The LMPC is implemented in a receding horizon manner, where the first control action $u^*(t_k)$ in the optimal input sequence $u^*(t)$, $\forall \ t \in [t_k, \ t_{k+N})$ is applied to the system for the next sampling period. Specifically, the LMPC minimizes the time-integral of the cost function $L(\tilde{x}(t), u(t))$ that achieves its minimum value at the steady-

state $(x_s^*, u_s^*) = (0, 0)$ accounting for the constraints of Eqs. 2.4b-2.4e. The control objective of LMPC is to maintain the closed-loop state in the stability region Ω_ρ for all times, and ultimately bound the predicted state in the target region $\Omega_{\rho_{nn}}$, which is a small level set of V around the origin. $\Phi_{nn}(x)$ in Eq. 2.4d is a pre-determined control law that renders the origin of the LSTM system of Eq. 2.3 exponentially stable. When a well-conditioned LSTM model with a sufficiently high model accuracy can be obtained using noise-free training data, the LMPC of Eq. 2.4 guarantees closed-loop stability of the nonlinear system of Eq. 2.1. Theoretical results on closed-loop stability can be found in [167].

Remark 6. *Due to the sample-and-hold implementation of control actions, the closed-loop system of Eq. 2.1 cannot be stabilized exactly at the steady-state. In the formulation of MPC of Eq. 2.4, we require the predicted states to be ultimately bounded in a small level set of Lyapunov function (i.e., $\Omega_{\rho_{nn}}$ in Eq. 2.4e) such that the true state will also be bounded in a small neighborhood around the origin. If for any initial state within the stability region Ω_ρ, the closed-loop state remains bounded in Ω_ρ for all times and ultimately converges to a small neighborhood around the origin where it will be maintained thereafter, then we say the system is practically stable under the LSTM-based MPC.*

2.3 Dropout and Co-teaching Methods

It was demonstrated in [170] that LSTM models are able to capture the underlying process dynamics using noisy data that is corrupted by Gaussian noise; however, a degraded model performance was noticed when training LSTM models with non-Gaussian noise. To handle indus-

trial noise that follows a non-Gaussian distribution, Monte Carlo dropout method and co-teaching method were utilized in [170] to learn the ground truth from noisy data. In this section, we present an overview of these two approaches and will evaluate their performance and robustness to industrial data noise using a chemical reactor example in Section 2.4.

2.3.1 Dropout Method

Monte Carlo dropout (MC dropout) method treats neural network weights as random variables and drops out randomly selected weights at both training and testing stages [46, 47]. As the neural network models using dropout method are essentially probabilistic models, an estimate of uncertainty for the model predictions can be obtained by performing Monte Carlo simulations.

Consider the LSTM model in the general form of Eq. 2.3. Let $\mathbf{W} = \{W_i\}_{i=1}^{L}$ represent the set of weight matrices of all LSTM layers that include both the weights and the bias terms, where $W_i : \mathbf{R}^{K_i \times K_{i-1}}$ is the weight matrix for the ith LSTM layer, and L is the number of layers. The goal of MC dropout method is to obtain the posterior distribution of LSTM weights W, i.e., $p(\mathbf{W})$, based on the training data $(\mathbf{M}, \mathbf{X})$, where $\mathbf{M}$ and $\mathbf{X}$ denote input and output data matrices. Specifically, as discussed in [46], the weight matrix W_i is defined as follows:

$$W_i = B_i \cdot \text{diag}(z_i) \tag{2.5}$$

where $z_i, i = 1, ..., L$ are used to represent the weights being dropped out with a certain probability and are a set of binary variables satisfying Bernoulli distribution. B_i are the variational variables to be optimized. After the LSTM model is trained using the MC dropout method, the predictive

distribution of LSTM output can be approximated by performing Monte Carlo dropout at test time. The predictive distribution of LSTM model is obtained via multiple realizations as follows:

$$p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{X},\ \mathbf{M}) \approx \frac{1}{N_t} \sum_{k=1}^{N_t} p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{W}_k) \tag{2.6}$$

where $\mathbf{m}^*$ and $\mathbf{x}^*$ are the LSTM input, and the corresponding output in the testing dataset, and $\mathbf{X}$, $\mathbf{M}$ are the LSTM inputs and outputs in the training dataset. N_t is the number of Monte Carlo realizations at the testing phase. Since the LSTM model obtained using the MC dropout method is a probabilistic model, the LSTM output is no longer deterministic given the same input, and needs to be approximated using Eq. 2.6. Therefore, a probabilistic distribution is obtained from Eq. 2.6 as an estimate of uncertainty for the model predictions. Additionally, the ground-truth process dynamics can then be approximated using the sample mean of all predictions when the training data is corrupted by noise.

Remark 7. *Note that the true process of Eq. 2.1 evolves deterministically, but our understanding of the process dynamics through ML models using dropout method is represented in a probabilistic manner because the training data is noisy. In fact, if we consider the additive sensor noise as an uncertain variable in the process model of Eq. 2.1, then the Monte Carlo dropout method provides an efficient way to model uncertain process dynamics (i.e., we can consider the LSTM model using the MC dropout method as an uncertain process model with the dropped LSTM weights being uncertain variables) and to predict underlying (nominal) process dynamics through a number of stochastic forward passes.*

Remark 8. *Since forward prediction is one of the most time-consuming parts in solving MPC re-*

gardless of using ML models or nonlinear first-principles models [176], parallel computing can be utilized to accelerate the computation of multiple forward passes in the Monte Carlo dropout method. Specifically, in parallel computing, each parallel computer node runs one forward prediction only and the host computer node averages the results and sends it to MPC. In [170], it was demonstrated that the computation time of running multiple forward predictions in parallel is significantly reduced compared to the calculation of the same number of forward predictions in serial mode. Additionally, it was demonstrated in [168] that through the implementation of parallel computing for forward predictions, the computation efficiency of the MPC optimization problem was also improved.

2.3.2 Co-teaching Method

The co-teaching method was originally proposed to improve model accuracy in image classification problems, for which the dataset is corrupted by noise [54]. Specifically, data noise in classification problems could cause mislabeled data (for example, an object "A" is mislabeled as object "B"), while in regression problems, data noise could cause a deviation from its ground truth value. In either case, it is challenging for ML model to achieve a desired model accuracy with a noisy dataset following the standard learning algorithm. Therefore, the co-teaching method provides an alternative way to train ML models under noisy labels by taking advantage of noise-free data and training two models at the same time [54, 178]. The intuition of co-teaching stems from the observations that neural networks will use a simple pattern to fit training data at the early stage of training process [54]. As a result, when assessing the loss function value under a simple pattern that approximates the relationship between neural network inputs and outputs, the noisy data

generally correspond to a larger loss function value, while noise-free data correspond to a smaller value.

Algorithm 1: Co-teaching Algorithm

for $i = 0$ **to** I_{max} **do**
  Select a mini-batch D_m from D
  Obtain the small-loss data sequences from model A:
  $D_A = \{x \in D_m \mid loss(A, x) \leq loss_T\}$
  Obtain the small-loss data sequences from model B:
  $D_B = \{x \in D_m \mid loss(B, x) \leq loss_T\}$
  Update the weight matrix of model A: $\mathbf{W_A} = \mathbf{W_A} - \eta \nabla loss(A, D_B)$
  Update the weight matrix of model B: $\mathbf{W_B} = \mathbf{W_B} - \eta \nabla loss(B, D_A)$
end

Fig. 2.2 shows two types of co-teaching structures (i.e., symmetric and asymmetric frameworks) that train two networks: A and B, simultaneously. The symmetric co-teaching training method is implemented following Algorithm 1, which is stated as follows: 1) at each training epoch, a mini-batch D_m is selected from the original mixed dataset D. Then, each model checks its data sequences (i.e., each pair of data labeled as input and output), and generates a small dataset (i.e., D_A and D_B) with all the data that has a low loss function value (e.g., $loss(A, x) \leq loss_T$), where $loss_T$ is the threshold for identifying small-loss data sequences; 2) this new small dataset is then sent to the peer network, and the neural network weights $\mathbf{W_A}$, $\mathbf{W_B}$ are updated with a learning rate η; 3) finally, the training is resumed, and the above process is repeated until the end of the training epochs I_{max}. The asymmetric co-teaching method is implemented in a similar way to train two models simultaneously. However, under asymmetric co-teaching framework, noise-free data is used by model A only, and noisy data is used by model B only. At each training epoch, model A injects a subset of noise-free data sequences into model B. Note that the information flows in

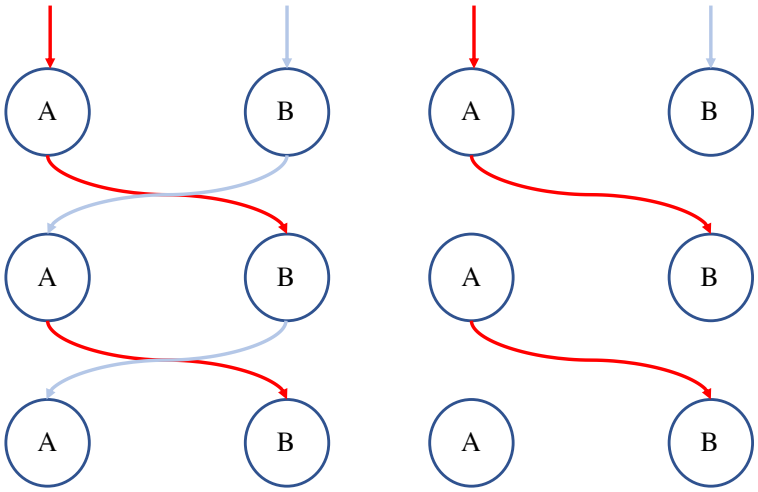


Figure 2.2: The symmetric (left) and asymmetric (right) co-teaching frameworks that train the two networks (A and B) simultaneously.

one direction in the asymmetric co-teaching framework, i.e., from model A to model B only.

Additionally, when using co-teaching method to solve the regression problem of LSTM modeling, the neural network structure needs to be carefully chosen. For example, the number of units in each network plays a role in learning the underlying process dynamics from a mixed dataset of both noisy and noise-free data. If a deep neural network with a large number of layers and neurons is used, the neural network may well fit the noisy data at an early stage (i.e., over-fitting) before effectively learning the ground truth from noise-free data. Additionally, the mixed dataset should be constructed with an appropriate ratio of noise-free data to noisy data. If noise-free data is insufficient, the neural networks may not be able to learn the ground truth, and may overfit the noisy data as training evolves.

In the following section, we use a chemical process example simulated in Aspen Plus Dynamics to illustrate the application of dropout and co-teaching LSTM modeling approaches and evaluate their performance and robustness. Specifically, we will discuss the following steps in this

case study: 1) data collection using Aspen simulation and first-principles model simulation, 2) LSTM training process, and 3) development of LSTM-based MPC that drives reactor temperature to its desired set-point. Through open-loop and closed-loop simulations, we demonstrate that the proposed LSTM model using dropout and co-teaching methods outperform the standard LSTM model in terms of more accurate predictions and better closed-loop performance.

Remark 9. *As both the dropout and co-teaching methods are developed for a general class of nonlinear systems, they are generally effective in improving the model accuracy that can be achieved under the standard ML training process. However, it was observed in [170] that the improvements of model accuracy under dropout/co-teaching methods vary depending on the noise levels. Therefore, how much improvement the dropout and co-teaching LSTM modeling approaches can achieve should be evaluated based on both the open-loop tests (i.e., prediction using testing dataset) and the closed-loop simulation under MPC.*

Remark 10. *Preprocessing of data, optimization of network structure in terms of the number of layers and neurons, and selection of loss functions, learning rate, and optimizers are common techniques for optimizing neural network performance. Beyond these standard tuning methods, the co-teaching method proposed in this work provides a guideline to improve training performance when both noisy and noise-free data are available. It will be shown in Section 2.4 that using the same mixed dataset with both noise-free and noisy data, the LSTM model using co-teaching method achieves better model accuracy than the LSTM following the standard training process.*

2.4 Application to an Aspen Plus Reactor Example

We consider an irreversible, second-order, exothermic reaction using Ethylene (A) and Benzene (B) to produce Ethyl benzene (EB) in a well-mixed, non-isothermal continuous stirred tank reactors (CSTR) [77]. The CSTR reactor is fed with two Hexane solutions in the feeding flow F_1 and F_2. The two flows have the same inlet temperature T_0, but different volumetric flowrate $F_{vj,in}$, $j = 1, 2$, where $j = 1, 2$ denotes the feeding flow F_1 and F_2. The reactants A and B are contained in each feeding flow separately with inlet molar concentration C_{A0} and C_{B0}. The reactions taking place in the CSTR are:

$$C_2H_4 + C_6H_6 \rightarrow C_8H_{10} \ \ (ethylbenzene) \tag{2.7a}$$

$$C_2H_4 + C_8H_{10} \rightarrow C_{10}H_{14} \ \ (di - ethylbenzene) \tag{2.7b}$$

$$C_6H_6 + C_{10}H_{14} \rightarrow 2C_8H_{10} \tag{2.7c}$$

In this study, the reactor model is developed in Aspen Plus and Aspen Plus Dynamics V11. The model is constructed and the steady-state simulations are first performed in Aspen Plus. Then, a dynamic simulation of the reactor process is carried out in Aspen Plus Dynamics to analyze its real-time performance. In Aspen Plus, a main flow sheet is designed with three valves and one CSTR as shown in Fig. 2.3. The valves play a role as a connector of fluid flow and parts by defining the pressure drop in the specific location, which is critical for generating a proper dynamic model. Without reasonable pressure drop in the process provided, Aspen Plus Dynamics can not identify the source making the fluid flow through the system and may result in failure of dynamic simulation. In this model, the pressure drop at V_1 and V_2 are both 5 bar, and the pressure drop at

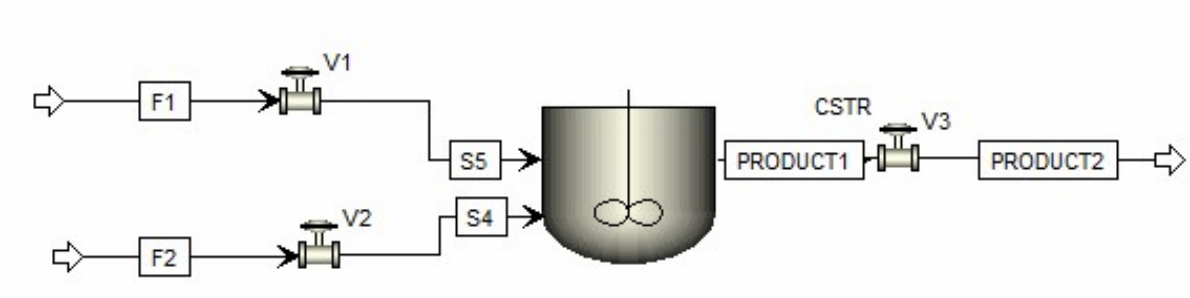


Figure 2.3: Aspen flow sheet of chemical reactor example (steady-state set-up).

V_3 is 2 bar.

Hexane is chosen as the solvent in the feeding flow F_1 and F_2 to ensure that the flow is in the liquid phase under the inlet temperature. Therefore, with a constant inlet volumetric flow rate, the amount of feeding reactants can be manipulated by adjusting the feeding concentration. Process parameter values used in the Aspen model are listed in Table 2.1, where C_A, C_B, ρ_L, V, and T are the concentration of ethane, the concentration of benzene, mass density, volume and temperature of the reacting liquid in the CSTR, respectively. C_p is the mass heat capacity of the liquid mixture and is assumed to be constant. C_{As} and C_{Bs} are the steady-state concentration of reactants A and B, and C_{A0}, C_{B0} are the inlet concentration of A and B.

A liquid-only CSTR equipped with a heating/cooling jacket that supplies/removes heat at a rate Q, is considered to carry out three reactions. The initial temperature and pressure of the CSTR are set to be $400\ K$ and $15\ bar$, respectively, which can be automatically adjusted by the steady-state simulation in Aspen. After incorporating the reactions of Eq. 2.7 in the CSTR, steady-state simulation is performed for analysis of plant behavior.

Before exporting the steady-state model to Aspen Plus Dynamics, the reactor geometry and

Table 2.1: Parameter values of Aspen model

$T_0 = 350.0\ K$	$T_s = 322.2\ K$
$F_{v1,in} = 50.0\ m^3/hr$	$F_{v2,in} = 23.6\ m^3/hr$
$C_{As} = 1.5454\ kmol/m^3$	$C_{Bs} = 4.2714\ kmol/m^3$
$C_{A0} = 4\ kmol/m^3$	$C_{B0} = 5\ kmol/m^3$
$Q_s = -695.1\ kJ/s$	$C_p = 2.41\ kJ/kg\ K$
$V = 60\ m^3/s$	$\rho_L = 683.7\ kg/m^3$
Heat Transfer Option	$Dynamic$
Medium Temperature	$298\ K$
Temperature Approach	$77.33\ K$
Heat Capacity of Coolant	$4200\ J/kg\ K$
Medium Holdup	$1000\ kg$

thermodynamic parameters are defined in the dynamic mode of Aspen. The vessel type, head type, and length of the CSTR in this study are vertical, flat, and 10 meters, respectively. The thermodynamic parameter values are listed in Table 2.1. To ensure that the dynamic mode is set up properly, we run the steady-state simulation again and complete the pressure check with the built-in Aspen Plus Pressure Checker with no error. Subsequently, the steady-state model is exported to Aspen Plus Dynamics with pressure driven chosen as the driven type of dynamic simulation.

2.4.1 Dynamic Model in Aspen Plus Dynamics

The flow sheet of the Apsen dynamic model is shown in Fig. 2.4. Specifically, a direct acting level controller, where direct means the output signal increases as the input signal increases, is added to the dynamic model to regulate the liquid level at 50 percent capacity in this study. Note that this controller can be designed following the default setting before exporting the steady-state

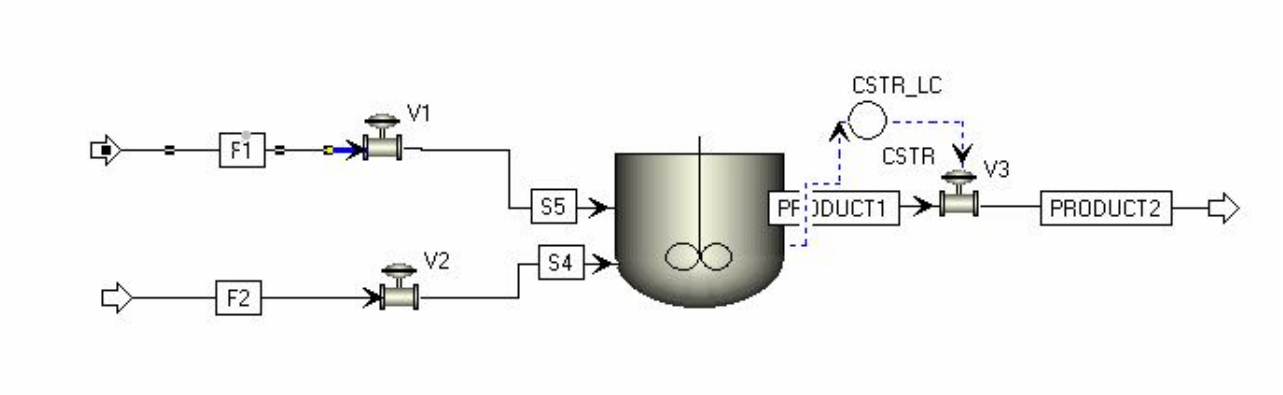


Figure 2.4: Aspen flow sheet of chemical reactor example (dynamic model set-up).

After configuration of the controller, we run a steady-state simulation in Aspen Plus Dynamics to obtain the steady-state for the dynamic model. The steady-state value of Q is $-695097.0\ W$. Then, the heating type of CSTR is changed to constant duty to allow the outside controllers to manipulate Q during the dynamic simulation. Similarly, the volumetric flow rates F_1 and F_2 are fixed, and a steady-state simulation is performed to ensure that the dynamic model reaches the steady-state before collecting data.

Since the Aspen dynamic model can be considered to be a high-fidelity process model for the CSTR, we use Aspen dynamic simulation to generate datasets for neural network training. Industrial noise is introduced on state measurements to represent common sensor variability in chemical plants. Fig. 2.5 shows the normalized data noise obtained from Aspen public domain data. Fig. 2.6 shows the probability distribution of the normalized industrial noise in Fig. 2.5, from which we verify that the noise follows a non-Gaussian distribution.

Open-loop simulation is carried out in Aspen Plus Dynamics using the pseudorandom input

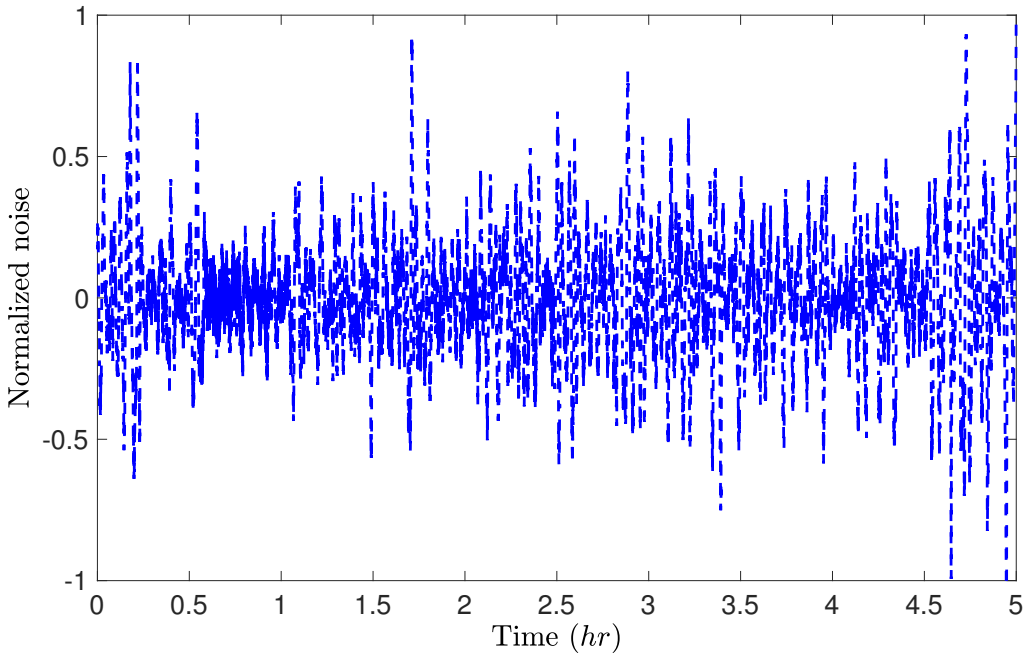


Figure 2.5: Normalized industrial noise from Aspen public domain data.

signals generated in Matlab. A local Message Passing Interface (MPI) is constructed to link Aspen with Matlab so that the Aspen dynamic model can automatically read the input signals from Matlab and apply them in the dynamic simulations. In the open-loop simulation, the manipulated input variable Q varies within the range of $[-1.0 \times 10^6\ W,\ -4.0 \times 10^5\ W]$, and is implemented in a sample-and-hold fashion with the value updated every five minutes of the simulation time. We run the open-loop simulation for $15,000$ minutes (simulation time) under pseudorandom input signals of Q with the industrial noise of Fig. 2.5 added on the temperature measurements. Specifically, since the industrial noise in Fig. 2.5 is the normalized result from Aspen public domain data, it is amplified by five times and then added on the temperature measurements. All the state measurements (e.g., C_A, C_B, and T) and input value Q are continuously recorded to build the dataset for neural network training.

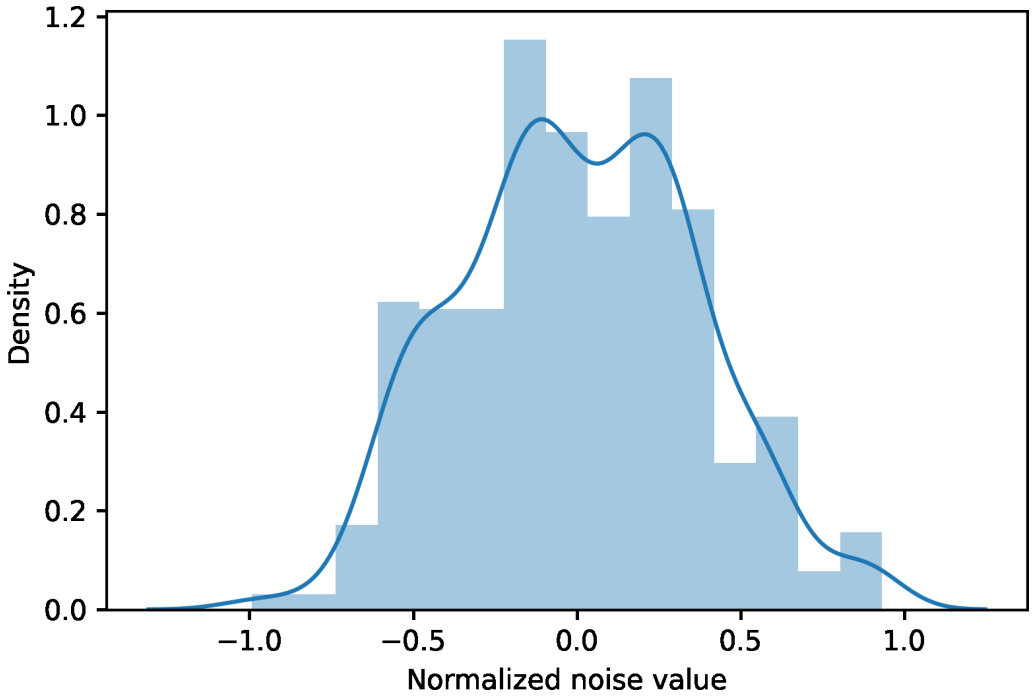


Figure 2.6: Probability density plot of normalized industrial noise in Fig. 2.5.

2.4.2 First-Principles Model

While Aspen Plus provides an efficient way that allows engineers to simulate, troubleshoot and optimize process performance and profitability with its high-fidelity process model, Aspen Plus models are typically not used in controller design due to their high computational cost. To reduce the computational time of solving the process model, first-principles models employing simplifying assumptions can be adopted in the design of model-based controllers. Additionally, extensive computer simulations using first-principles model is one of the most efficient data generation methods in ML.

In this study, we take advantage of the first-principles model of CSTR to generate noise-free

datasets for LSTM training using the co-teaching method. Although the first-principles model may not fully capture the Aspen model dynamics under the same operating conditions, we will demonstrate that the co-teaching method using noisy data from the Aspen model, and noise-free data from the first-principles model solutions is still able to improve prediction accuracy of the LSTM model. While in practice noisy data is provided by chemical plants, and noise-free data is unavailable, the implementation of co-teaching method in this case study implies that the co-teaching LSTM modeling approach can improve prediction accuracy by using noise-free data generated from first-principles models, which broadens its application in many process modeling problems in industry. By applying mass and energy balances, the dynamic model of CSTR is described by the following nonlinear ODEs:

$$\frac{dC_A}{dt} = \frac{F_{v1,in}}{V}(C_{A0} - C_A) - r_1 - r_2 \tag{2.8a}$$

$$\frac{dC_B}{dt} = \frac{F_{v2,in}}{V}(C_{B0} - C_B) - r_1 - r_3 \tag{2.8b}$$

$$\frac{dT}{dt} = \frac{F_{v1,in} + F_{v2,in}}{V}(T_0 - T) + \frac{-\Delta H_1}{\rho_L C_p} r_1 + \frac{-\Delta H_2}{\rho_L C_p} r_2 + \frac{-\Delta H_3}{\rho_L C_p} r_3 + \frac{Q}{\rho_L C_p V} \tag{2.8c}$$

$$r_1 = k_1 e^{-\frac{E_1}{RT}} C_A C_B \tag{2.8d}$$

$$r_2 = k_2 e^{-\frac{E_2}{RT}} C_A C_{EB} \tag{2.8e}$$

$$r_3 = k_3 e^{-\frac{E_3}{RT}} C_B C_{DEB} \tag{2.8f}$$

where r_j, $j = 1, 2, 3$ denote the rate of each reaction in Eq. 2.7 based on the rate law equation, and C_{EB}, C_{DEB} represent the concentration of C_8H_{10}, and of $C_{10}H_{14}$, respectively. The kinetic parameters for reactions are given in Table 2.2, where R, k_j, ΔH_j, and E_j, $j = 1, 2, 3$ represent ideal gas constant, pre-exponential constant, enthalpy of reaction, and activation energy of each

reaction, respectively.

Table 2.2: Parameter values of the first-principles model of CSTR

$k_1 = 1.528 \times 10^6\ m^3/kmol\ s$	$\Delta H_1 = -1.04 \times 10^5\ kJ/kmol$
$k_2 = 2.778 \times 10^5\ m^3/kmol\ s$	$\Delta H_2 = -1.02 \times 10^5\ kJ/kmol$
$k_3 = 0.4167\ m^3/kmol\ s$	$\Delta H_3 = -5.50 \times 10^2\ kJ/kmol$
$E_1 = 71160\ kJ/kmol$	$R = 8.314\ kJ/kmol\ K$
$E_2 = 83680\ kJ/kmol$	$E_3 = 62760\ kJ/kmol$
$V = 60\ m^3/s$	$\rho_L = 683.7\ kg/m^3$
$C_p = 2.41\ kJ/kg\ K$	

The manipulated input is the heat input rate Q represented in deviation variable form, i.e., $u^T = [Q - Q_s]$. Similarly, the process states are represented by $x^T = [C_A - C_{As}\ \ C_B - C_{Bs}\ \ T - T_s]$ where C_{As}, C_{Bs}, T_s are the steady-state values of C_A, C_B and T. By representing all the variables in deviation forms, the equilibrium of Eq. 2.8 is at the origin of state-space. The same pseudorandom signals of Q applied in Aspen simulations are applied to the open-loop simulation of the first-principles model of Eq. 2.8, where the explicit Euler method is used to integrate the nonlinear ODEs with a sufficiently small integration time step $h_c = 0.05\ min$. The input signals are applied in a sample-and-hold fashion with the sampling period $\Delta = 5\ min$. In open-loop simulations, process variables are measured every integration time step.

Fig. 2.7 compares the open-loop state profiles from Aspen simulation and first-principles model solutions under the same input sequences. Although the state profiles are close to each other, small deviations in the evolution of states can be noticed between the two models, which implies the existence of a mismatch between the Aspen model and the first-principles model of Eq. 2.8. The mismatch is caused by the difference between the balance-based first-principles equations and

the equations adopted by Aspen Plus during the simulation. Variation of parameters, such as liquid density, heat capacity, and the energy exchange coefficient, contribute to the model mismatch. With noisy data from Aspen simulation and noise-free data from first-principles solutions, the simulation study in this section provides an insight on the applicability of the co-teaching method in handling real industrial noisy data, for which the corresponding noise-free data is generally unavailable, but can be obtained from computer simulations using first-principles or empirical models.

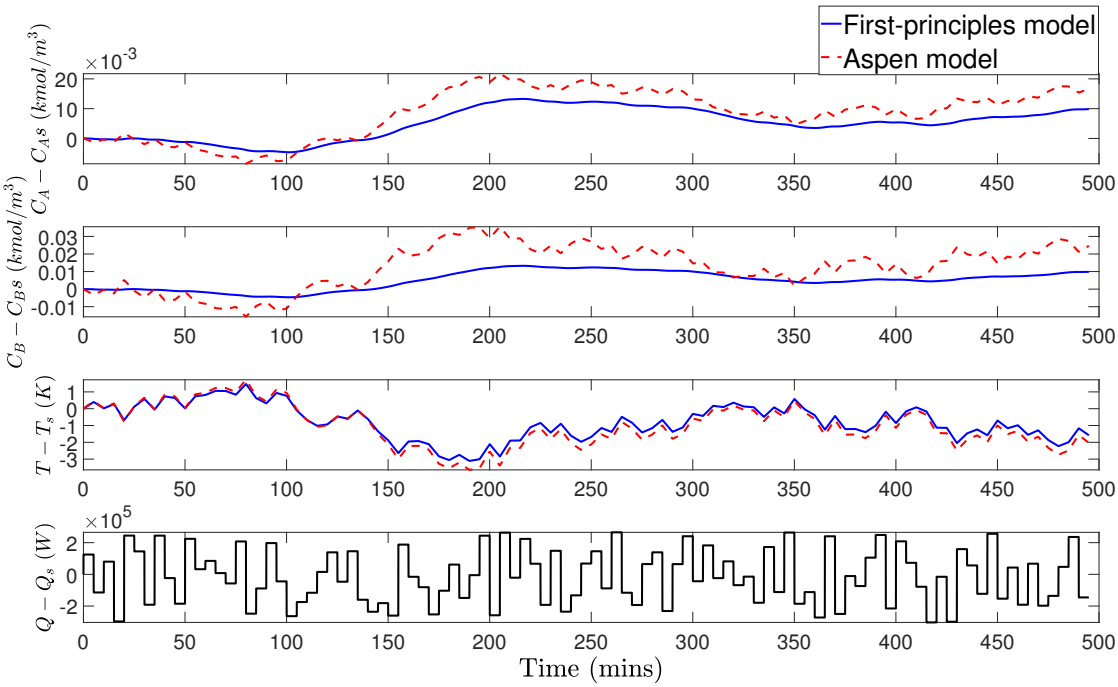


Figure 2.7: State profiles ($C_A - C_{As}$, $C_B - C_{Bs}$, $T - T_s$) from open-loop simulations of Aspen model and of first-principle model, respectively, under the same input sequences of Q.

Remark 11. *It should be clarified that in reality, the noisy data is from chemical plants due to common plant variance and sensors variability, and the noise-free data is generated from computational simulations based on first-principles process models. Therefore, when an industrial noisy dataset is available, computer simulations of the first-principles process model will only be used*

to generate noise-free datasets for the implementation of co-teaching method.

Remark 12. *To further reduce the model mismatch, an accurate source of parameters can be utilized for the first-principles model, and a real-time communication between the Aspen database and the first-principles model can be established.*

2.4.3 Dropout and Co-teaching LSTM Models

To reduce the impact of measurement noise in predicting future states, the LSTM models in this work rely on the state measurements over a past period of time to make predictions. The LSTM model is developed with C_A, C_B, T, and Q as inputs to predict the temperature T in the future time. Specifically, when using LSTM models in MPC to predict future states, the LSTM input vector at the current time step $t = t_k$ consists of the state measurements of $C_A(t)$, $C_B(t)$ and $T(t)$ over past five sampling periods, i.e., $\forall t \in [t_{k-5}, t_k]$ and the heat input rate $Q(t)$, $\forall t \in [t_{k-4},\ t_{k+1}]$ implemented in a sample-and-hold fashion. Note that the heat input rate within the last sampling period, i.e., $\forall t \in [t_k,\ t_{k+1}]$ is unknown at the current time step t_k as it is the variable that will be optimized by MPC to meet the control objective. The LSTM output vector at the current time step $t = t_k$ is the predicted temperature $T(t)$ over $t \in [t_{k-4}, t_{k+1}]$. Since the temperature measurements before the current time step are known, only the prediction of $T(t)$ in the last sampling period, i.e., $\forall t \in [t_k, t_{k+1}]$ will be used in MPC to solve the optimization problem.

After running Aspen dynamic simulations and open-loop simulations of the first-principles model of Eq. 2.8, we obtain a dataset with LSTM inputs and outputs and reshape it to the following tensor dimensions: [2467,500,4] for inputs and [2467,500,1] for outputs, where the first element represents the total number of data sequences, the second element represents the length of each

data sequence (i.e., 500 data points correspond to five sampling periods $25\ min$, in which the data point is collected every integration time step $h_c = 0.05\ min$), and the last element represents the dimension of inputs and outputs, respectively (i.e., the LSTM has four inputs: C_A, C_B, T, and Q, and one output: T). Among 2467 data sequences, 494 sets of data are used for validation, and 100 sets of data are used for testing. In the case of the co-teaching method, noiseless datasets are obtained from first-principles solutions under the same operating conditions as performed in Aspen simulations. The high-level application programming interface, Keras, is used to develop the standard, dropout, and co-teaching LSTM networks under the optimization algorithm *Adam*.

2.4.4 Open-loop Simulation Results

We first carry out an open-loop simulation study to demonstrate the improvement of LSTM model accuracy using dropout and co-teaching methods. Table 2.3 shows the mean squared errors (MSE) of reactor temperature predicted by LSTM models with respect to the ground truth (i.e., actual temperature value of the nominal system) from testing dataset. The results for standard LSTMs under different datasets are shown in item 1, and those for co-teaching and dropout LSTMs are shown in items 2 and 3, respectively.

Note that all the LSTM models in Table 2.3 are developed with the same structure in terms of the same number of neurons, layers, epochs, and the type of activation functions and of optimization algorithms. We first consider three types of datasets for standard LSTM models, and the results are reported in item 1 of Table 2.3. Specifically, in item 1a, the LSTM model is trained following the standard training process with noisy data only (i.e., noisy data from Aspen simulations in Section 2.4.1); in item 1b, the LSTM model is trained using a mixed dataset consisting of

Table 2.3: Open-loop prediction results under industrial noise

Methods	**MSE** T
1a) LSTM : only using noisy data	1.8217
1b) LSTM : mixed data (noise-free data from fp)	3.0357
1c) LSTM : mixed data (noise-free data from Aspen)	1.5386
2a) Co-teaching LSTM (noise-free data from fp)	0.8596
2b) Co-teaching LSTM (noise-free data from Aspen)	0.7140
3) Dropout LSTM : only using noisy data	0.8761

both noisy data from Aspen simulations and noise-free data from simulations of the first-principles model in Section 2.4.2 ("fp" in Table 2.3 represents the first-principles model); in item 1c, we consider a scenario where noise-free data is also available from Aspen simulations, thereby the LSTM model is trained using both noisy and noise-free data from Aspen simulations. However, it should be noted that the last scenario is considered only for comparison purposes since the noise-free data from chemical plants (here the Aspen model can be considered as a real chemical process) are generally unavailable. It can be seen from Table 2.3 that introducing noise-free data into brute force learning of LSTMs (i.e., standard LSTM models) may or may not improve their prediction accuracy. Specifically, when noise-free data from the same process (i.e., from Aspen model) is provided with noisy data, standard LSTM achieves a lower MSE in item 1c than the standard LSTM using noisy data only in item 1a; however, the standard LSTM using a mixed dataset with noise-free data from the first-principles model has a larger MSE due to the mismatch between the Aspen model (i.e., source of noisy data) and the first-principles model (i.e., source of noise-free data). This mismatch, if not handled appropriately, may misguide LSTM training and leads to worse prediction performance.

Subsequently, we train LSTM models using the co-teaching method with the same two types of mixed datasets (i.e., the noise-free data from Aspen model, and from first-principles model, respectively). The co-teaching LSTM training is initially equipped with a noisy dataset, and as the training evolves, noise-free data sequences are introduced into the learning process as discussed in the co-teaching algorithm. As shown in Table 2.3, the two co-teaching LSTM models have lower MSEs than the corresponding standard LSTM models using the same type of mixed dataset. It is also noticed that the co-teaching LSTM using noise-free data from Aspen simulations has the lowest MSE results among all the LSTM models in this study. Additionally, we train the dropout LSTM model using the noisy data only. It is shown in Table 2.3 that the dropout LSTM model (i.e., item 3) achieves lower MSE than the standard LSTM using the same noisy dataset (i.e., item 1a), which demonstrates the benefits of dropping out weights during training and testing to avoid overfitting to noisy data.

2.4.5 Closed-loop Simulation Results

Finally, we incorporate the LSTM models in the LMPC of Eq. 2.4, and carry out a closed-loop simulation study to demonstrate the improved closed-loop performance under dropout and co-teaching LSTM models. The control objective of LMPC is to stabilize the reactor temperature at its steady-state T_s by manipulating the heat input rate ΔQ. The LMPC objective function of Eq. 2.4a is designed with the following form that has its minimum value at the steady-state:

$$L(x,u) = |x_3|^2_{Q_1} + |u|^2_{Q_2} \tag{2.9}$$

where Q_1 and Q_2 are the coefficient matrices that represent the contributions of temperature and of control actions (both are in deviation forms) in the MPC objective function. In this example, we choose $Q_1 = 1$ and $Q_2 = 5 \times 10^{-9}$ to balance the contributions of process state and manipulated input to the objective function. The nonlinear optimization problem of LMPC is solved using the python module of the IPOPT software package [157], named PyIpopt with the sampling period $\Delta = 5\, min$.

Fig. 2.8 and Fig. 2.9 show the closed-loop state profiles (with noisy measurements) under LMPC using standard LSTM, dropout LSTM, and co-teaching LSTM models for two different initial conditions. Specifically, Fig. 2.8 shows that starting from an initial temperature $T = 340\ K$ higher than the steady-state value, all the LSTM models drive the temperature to its steady-state within 100 minutes. However, in Fig. 2.9, with an initial temperature $T = 300\ K$ lower than the steady-state value, the standard LSTM model takes much longer time than the dropout and co-teaching LSTM models to stabilize the temperature at the steady-state.

To further analyze their closed-loop performances in terms of state convergence speed and energy consumption, we use the MPC objective function as an indicator of closed-loop performance as it accounts for both state and input information. It can be seen from Eq. 2.9 that a lower objective function value implies a faster convergence to the steady-state and less consumption of Q during operation. Therefore, we integrate the objective function value over the closed-loop simulation period t_s, i.e., $L_s = \int_{t=0}^{t=t_s} L(x,u)dt$, for each LSTM model. For the initial condition $T = 340\ K$, L_s is calculated to be 44963.07 for standard LSTM, 39488.3 for dropout LSTM, and 37843.28 for co-teaching LSTM; for the initial condition $T = 300\ K$, L_s is calculated to be 120697.7 for standard LSTM, 40636.26 for dropout LSTM, and 41083.07 for co-teaching LSTM. In both cases,

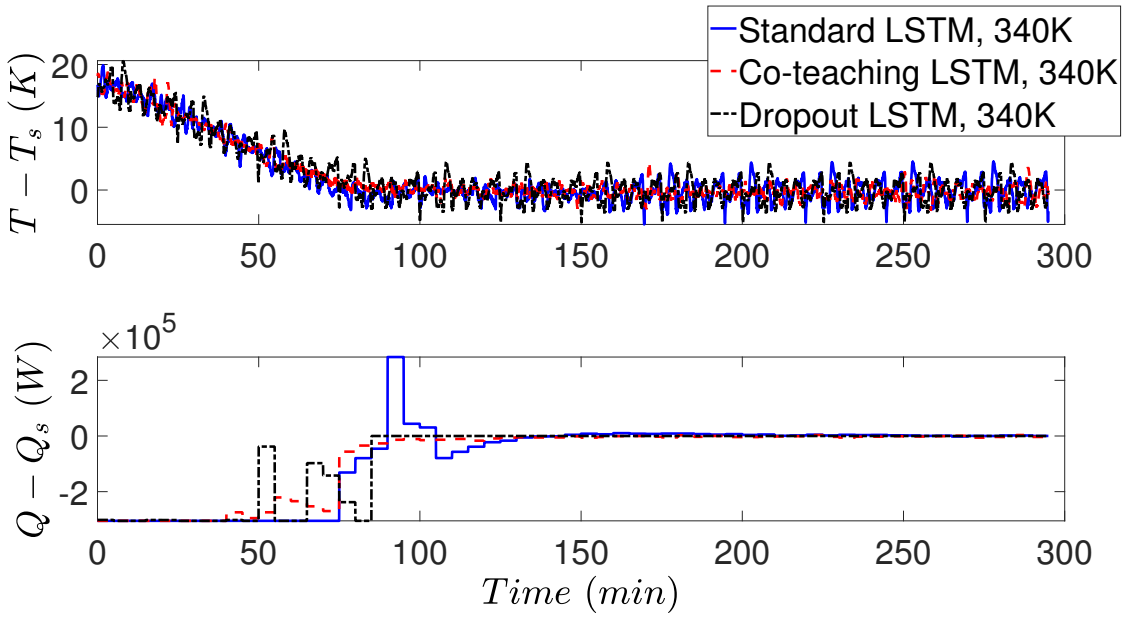


Figure 2.8: Closed-loop state profile ($x_3 = T - T_s$) and manipulated input profile ($u = Q - Q_s$) for the initial condition $T = 340\ K$ under the MPC using standard LSTM, co-teaching LSTM, and dropout LSTM, respectively.

co-teaching and dropout LSTM models have lower L_s values than standard LSTM model, which indicates an improvement in closed-loop performance. Additionally, we test more initial conditions of temperature within $[300, 340]\ K$ under LMPC. It is demonstrated that for $T_{initial} > T_s$, all the three LSTM models can stabilize the temperature at the steady-state within a short time, while for $T_{initial} < T_s$, co-teaching and dropout LSTM models significantly improve the dynamic responses than standard LSTM (like the one in Fig. 2.9). For all the tested initial conditions, dropout and co-teaching LSTM models achieve better closed-loop performances with lower values of L_s.

Remark 13. *It should be noted that closed-loop stability under LMPC is not guaranteed since a sufficiently high model accuracy as required for ML models in [167] may be unachievable for the proposed LSTM models that receive noisy measurements to predict true states. However, from*

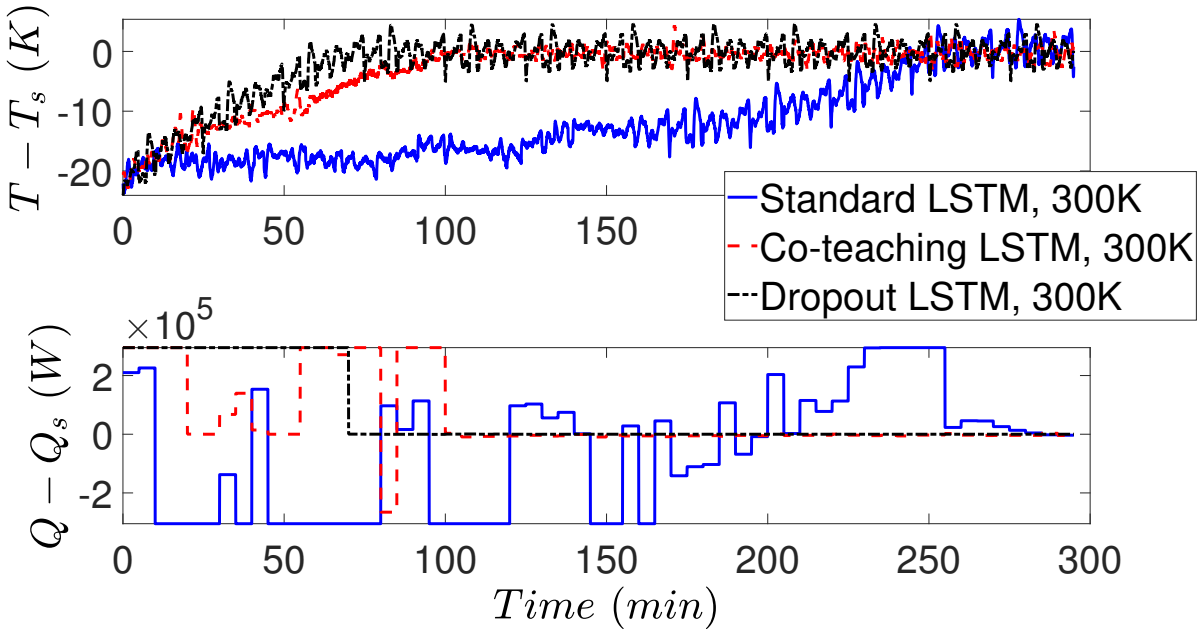


Figure 2.9: Closed-loop state profile ($x_3 = T - T_s$) and manipulated input profile ($u = Q - Q_s$) for the initial condition $T = 300\ K$ under the MPC using standard LSTM, co-teaching LSTM, and dropout LSTM, respectively.

extensive closed-loop simulation runs, it is demonstrated that with a small noise level, the state trajectories for all types of LSTM models successfully converge to the steady-state from different initial conditions. As the noise level increases, the closed-loop performance gets worse for all LSTM models in the sense that most of the state trajectories can still converge to the steady-state but showing oscillation over time. In this case, the proposed dropout and co-teaching LSTM models can be used to improve closed-loop performance when training with noisy data.

Chapter 3

Model Predictive Control of Nonlinear Processes Using Neural Ordinary Differential Equation Models

3.1 Introduction

Model predictive control is an advanced optimization-based control strategy that has been widely used in various industries and applications, ranging from chemical process control [15, 136] to autonomous vehicles [21, 72, 128]. The primary objective of a model predictive controller (MPC) is to regulate the behavior of a dynamical system by predicting its future behavior with a mathematical model and optimizing control inputs accordingly [109]. Specifically, the MPC repeatedly solves an optimization problem at each sampling instant, which takes into account a prediction of the future behavior of the system over a finite time horizon, to generate a control sequence that minimizes a chosen performance criterion over the entire prediction horizon. There-

fore, the prediction accuracy of the process model used by the MPC is essential to the MPC's performance, and developing an accurate predictive model continues to be a mathematical and scientific challenge.

First-principles models, developed based on a comprehensive understanding of the process mechanism, are the most robust and accurate predictive models that an MPC can use. However, developing such models is usually a time-consuming task that requires physicochemical information, which may not be available in most cases. To address this challenge, data-driven modeling methods have been used in chemical engineering to facilitate and generalize the modeling process while reducing costs. An example of a data-driven application in classical process control is the Fuzzy logic technique developed in the 1960s [182], which has been applied in various chemical engineering research efforts [95, 116, 177]. However, Fuzzy Logic Control (FLC) is usually implemented as a model-free approach [140], which is conceptually different from model-based approaches such as MPC. One advanced recent technique that is conceptually similar to the FLC is reinforcement learning [112]. With the advent of the 21st century, ML methods, which are a subset of data-driven techniques, have been widely used and demonstrated great success in both classification and regression problems. The classical ML methods such as Gaussian process (GP) regression [59, 186] and support vector regression [29, 171] became popular choices to model physical systems using data for process control purposes. With the development of computer software and hardware, more advanced and computationally intensive methods such as neural network (NN) modeling have also become accessible in conventional engineering disciplines. Since then, various NN techniques, including but not limited to feedforward neural network (FNN) [82, 107, 153, 181], recurrent neural network (RNN), and convolution neural network (CNN) [56] approaches have led to significant

innovations in process control research. In particular, RNN models trained with time-series data have been demonstrated to be an effective method to develop predictive models for nonlinear systems to be used as the process model in MPC for chemical process control [164, 170, 173]. In addition to NN approaches, other nonlinear methods that have been widely used in the process systems engineering literature to model dynamical systems for process control include nonlinear autoregressive methods [17] and sparse identification for nonlinear dynamics (SINDy) [2, 3, 4].

Recently, a novel class of neural network models, the neural ordinary differential equation (NODE), was proposed and has attracted significant attention due to their ability to learn continuous-time dynamical systems. As stated in [26], due to its continuous nature, the NODE model can incorporate time-series data having an arbitrary time span between each data point, which gives it an important advantage compared to the RNN, which is a discrete approximation of the time sequence data that requires a consistent time step size in the training set and provides prediction having the same time step sizes. On the other hand, the performance of the SINDy method is highly dependent on the candidate terms in the predefined bank of basis functions. Similar to other NN methods, the NODE has a high degree of freedom in its model structure and weight matrices, which makes it more generalizable than SINDy. Therefore, this new modeling method has been widely tested in various areas to identify and control physical processes. For example, [88] confirmed the feasibility of using NODE in linear and nonlinear structural identification, demonstrating their results on several numerical and experimental systems. [20] proposed a framework using the NODE model to extract parameters that describe the derivatives of physical states from the data. Furthermore, in [25], a knowledge-based neural ordinary differential equation (KNODE) model that can capture a dynamic process was developed and implemented by

designing a KNODE-based MPC for a quadrotor system. These advances illustrate the potential of NODE-based modeling for physical systems and motivate its use in the development of MPC for chemical processes.

Lastly, the application of all data-driven methods in a practical industrial system faces several critical challenges, one being that data collected from the system is usually corrupted by noise. Therefore, various researchers investigated methodologies to account for noisy data in the NN context. One way to handle noisy data is by smoothing the data with a noise filter in the data preprocessing step [154]. But this method usually involves averaging the measurements within a certain time window, which introduces an undesired delay to the control system. In addition to smoothing the data, algorithms can be used to reduce the effect of noise during the development of the NN model. For example, [170] demonstrated the Monte Carlo dropout and co-teaching method as effective ways to handle noisy data in LSTM models. However, [100] stated the dropout method cannot be used with the original NODE structure. Hence, they proposed a modification on the NODE model, namely the Stochastic Differential Equation (SDE), to allow the use of effective regularization methods (e.g., dropout) while preserving most of the design of the NODE model. Finally, [51] proposed an NODE-based framework to extract the ground-truth trajectory from noisy measurements.

Motivated by the theoretical advantage of using the NODE model in modeling continuous-time systems and the recent results supporting its implementation in physical systems, this work aims to develop a Lyapunov-based MPC (LMPC) based on a NODE model. The NODE model is designed to capture complex nonlinear relationships in a chemical process, so that this NODE-based LMPC can potentially be implemented in an industrial chemical process. The rest of this

Chapter is organized into the following sections: in Section 3.2, the mathematical notation and background of this work will be introduced; the NODE structure, training algorithm, and technical details will be discussed in Section 3.3 and the LMPC design and stability analysis will be addressed in the following section. In Section 3.5, a simulation case study of a chemical process is utilized to show the implementation of NODE modeling and the development of NODE-based LMPC. The performance of the NODE model and the NODE-based LMPC will be evaluated via open- and closed-loop simulations, respectively. Finally, Section 3.5 further demonstrates the application of NODE-based LMPC while considering the effect of Gaussian and non-Gaussian types of sensor noise. This study demonstrates that the NODE modeling approach can be used as an additional option to capture the derivative information of the state variables, which can be used in the contractive stability constraint of the LMPC. Specifically, different from the LMPC approaches that utilize recurrent neural network models to approximate the derivative of the state, NODE allows capturing the state derivative while being trained to fit the state trajectory. Furthermore, the continuous property of the NODE model enables using the subsampling method to account for noisy data and deal with measurement noise more effectively compared to recurrent neural network models.

3.2 Preliminaries

3.2.1 Notation

The time-derivative of x is represented by $\dot{x}$, that is, $\dot{x} := \frac{\mathrm{d}x}{\mathrm{d}t}$. $\hat{y}$ denotes the prediction of y using a mathematical model, and $\hat{V}(x) = V(\hat{x})$. $\mathbf{v}^\top$ represents the transpose of $\mathbf{v}$. "\" stands

for subtracting one set from another, such that $A \backslash B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and achieves a value of zero only when evaluated at zero.

3.2.2 Class of systems

The general class of continuous-time systems considered in this research can be expressed by the following equations:

$$\dot{x} = F(x, u) \tag{3.1a}$$

$$y = x + v \tag{3.1b}$$

$$x(t_0) = x_0 \tag{3.1c}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state vector and the manipulated input vector, respectively. An arbitrary nonlinear function, $F : \mathbb{R}^{n+m} \to \mathbb{R}^n$, mapping the state and input vectors to the time-derivative of the system, is assumed to be continuous and sufficiently smooth. $v \in \mathbb{R}^n$ represents the sensor noise affecting the state measurement y. t_0 and x_0 are used to denote the initial time and the corresponding initial state, respectively. Without loss of generality, the values of t_0 and x_0 are taken to be zero. By assuming $F(0, 0) = 0$ and that the system is in deviation form, i.e., $x_d = x - x_s; u_d = u - u_s$ where subscripts d and s denote deviation variables and the steady-state values of the state and input vectors, respectively, the steady-state of the nominal system with $v(t) = 0$ is located at the origin of the state space.

3.2.3 Defining Lyapunov-based Stability Region

To ensure that the nominal system of Eq. 3.1 can be used to construct a feasible process control problem, we first define an open region D in the state space around a selected set-point, which is a steady state of the system, such that the nominal system is closed-loop stable in the sense that any instantaneous state belonging to D can be brought to the set-point under a certain controller. Specifically, the stability criteria can be mathematically defined as the existence of a Lyapunov function $V(x)$ and a stabilizing controller $\Phi(x)$ such that, $\forall x$ in the region D, the following inequalities hold:

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

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

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

where $V(x)$ is a Lyapunov function, and c_1, c_2, c_3, c_4 are positive constants. In other words, Eq. 3.2 implies the existence of a controller that can ensure exponential stability of the state x around the set-point. One candidate controller can be the universal Sontag controller [99]. Therefore, one can first find a region (D) where the time-derivative of the Lyapunov function ($\dot{V}$) is negative under the controller $\Phi(x)$. Subsequently, we pick a subset of D, namely Ω_ρ, to be our stability region, such that $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$ where $\rho > 0$ and $\Omega_\rho \subset D$. The set of values of x that gives $V(x)$ equal to a positive constant ρ is the boundary of our stability region. Finally, the Lipschitz property of $F(x, u)$ combined with the bound on u implies the existence of positive constants

M, L_x, L'_x such that the following inequalities hold for all $x, x' \in D, u \in U$:

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

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

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

3.2.4 Neural Network Approximation of Time-series Data

Most popular neural network structures, such as the RNN family (e.g., vanilla RNN, Gated Recurrent Units (GRU), Long Short-Term Memory (LSTM) Units), compute their output using various logistic units (or neurons) to perform nonlinear transformations on the received input and propagate the result as the input of the next neuron. The result of the nonlinear transformations flowing from one neuron to another is named the hidden state. In the case of using an RNN model to capture time-series data, one of the common applications of neural network modeling in process control, the hidden state is passed chronologically until the desired final time step. Therefore, the RNN prediction for a time sequence state can be summarized as the following equation [26]:

$$h(k+1) = h(k) + f(h(k), x_{rnn}(k)) \tag{3.4}$$

where $h(k)$, $x_{rnn}(k)$, and $f(\cdot)$ stand for the hidden state of the RNN model for the k^{th} recurrent unit, the input for the k^{th} recurrent unit, which is usually the process measurements and control actions at time step k, and the nonlinear transformation performed on all received information in that unit. One may find that Eq. 3.4 is similar to the numerical integration step of an explicit

integration scheme. Therefore, continuously adding recurrent units to an RNN model is equivalent to using a smaller integration time step, where the nonlinear transformation within the RNN unit, $f(h(k), x_{rnn}(k))$, evaluates the right-hand side of the ordinary differential equation (ODE). The Neural Ordinary Differential Equation (NODE) is designed based on this observation in [26]. The structure of NODE is designed such that by fitting the output of the model to the data, the hidden state of the NODE will fit to the time-derivative of the data. The technical details of NODE will be discussed in Section 3.3, following the proposed workflow in [26].

3.2.5 Subsampling method

Subsampling is an effective statistical method that is used to reduce the size of the original data set by creating a subset of the original data [57]. Subsampling is widely used to derive inferences on a larger data set by using a representative subsample at a lower computational cost. For example, subsampling was used to downsample large biomedical data sets while preserving the distribution in [101], and computational power requirements for regression tasks were reduced through subsampling in [33]. Two of the most important tools used in developing ML models, train-test split and cross-validation, also belong to the subsampling family. In this work, we use subsampling to account for noisy data based on the assumption that some of the data points are more affected by noise than others. Therefore, by randomly selecting data points to create a subset that will be used to develop a prediction model, the impact of measurement noise can be mitigated, leading to the development of a better model. Subsampling has also been used in the data-driven modeling literature to account for data noise and has shown promising results [5, 66], inspiring the use of subsampling to handle noisy data in our work.

3.3 Neural Ordinary Differential Equations (NODE)

3.3.1 NODE Architecture

The NODE model in our work is developed to predict the state of a dynamical system using the following equation:

$$\hat{x}(t_k + t_p) = \text{ODESolver}\left(x(t_k), t_k, t_k + t_p, \mathfrak{f}(\hat{x}, u)\right) = F_{nn}(x_0, u, t_k, t_k + t_p, \mathfrak{f}) \tag{3.5}$$

where $\hat{x} \in \mathbb{R}^n$ is the NODE state vector and $x(t_k)$ represents the state of the nominal system of Eq. 3.1 at time step t_k. t_p stands for the time span from the initial state measurement at t_k. $\mathfrak{f}$ is the NN model used to capture the nonlinear dynamic relationship of the system, i.e., the right-hand side of the ODE representing the time-derivative,

$$\dot{\hat{x}} = \mathfrak{f}(\hat{x}, u) \tag{3.6}$$

Since we are working on a regression task with numerical data, an FNN model is used as the function $\mathfrak{f}$. For convenience, we refer to the NN model $\mathfrak{f}$ used in the NODE as the "core model" in the remainder of this manuscript, while F_{nn} is used to denote the overall NODE model, whose output is the state prediction itself following integration. The fundamental mathematical idea behind the NODE model is to find the optimum weight matrix θ^* of the FNN (core) model, such that the output of the FNN model can be used by the ODE solver to numerically calculate the predicted state from an arbitrary initial state measurement $x(t_k)$. Based on the universal approximation theorem [63, 64], the core model of a perfectly trained NODE model can capture the right-hand side of the

ODE of the desired system. Training of the NODE model refers to the process of optimizing the FNN weights based on the data.

The architecture of the NODE model is demonstrated in Fig. 3.1. Specifically, when using a trained NODE model to make a prediction, the instantaneous state measurement needs to be provided and will be used as the initial value of the ODE solver. The sequence of control inputs also needs to be provided to the NODE model, which will be used as the input of the FNN model to predict the "time-derivative of the state". The ODE solver will recursively call the FNN model to find the time-derivative and update the predicted state from the initial state value until the final time step of the prediction is reached. Finally, the prediction computed by the NODE model can be returned as a single value or as a vector of the state predictions at the desired time steps. If the prediction is returned as a time sequence, the intermediate time steps in the returning sequence need to be pre-defined and used as an argument of the prediction function. However, more intermediate states in the output will result in higher computational costs. Lastly, the output of the core model can be considered the hidden state of the NODE model and has a very similar value to the state derivative. However, although we defined and used the output of the core model in the NODE model as the time-derivative of the states, it actually does not have any physical meaning. Therefore, various types of regressive models can be used as the core model, but in the case of regressing a physical system, the FNN model is a popular candidate. In contrast, for an image classification task, for example, using a convolutional neural network (CNN) as the core model is the most common approach.

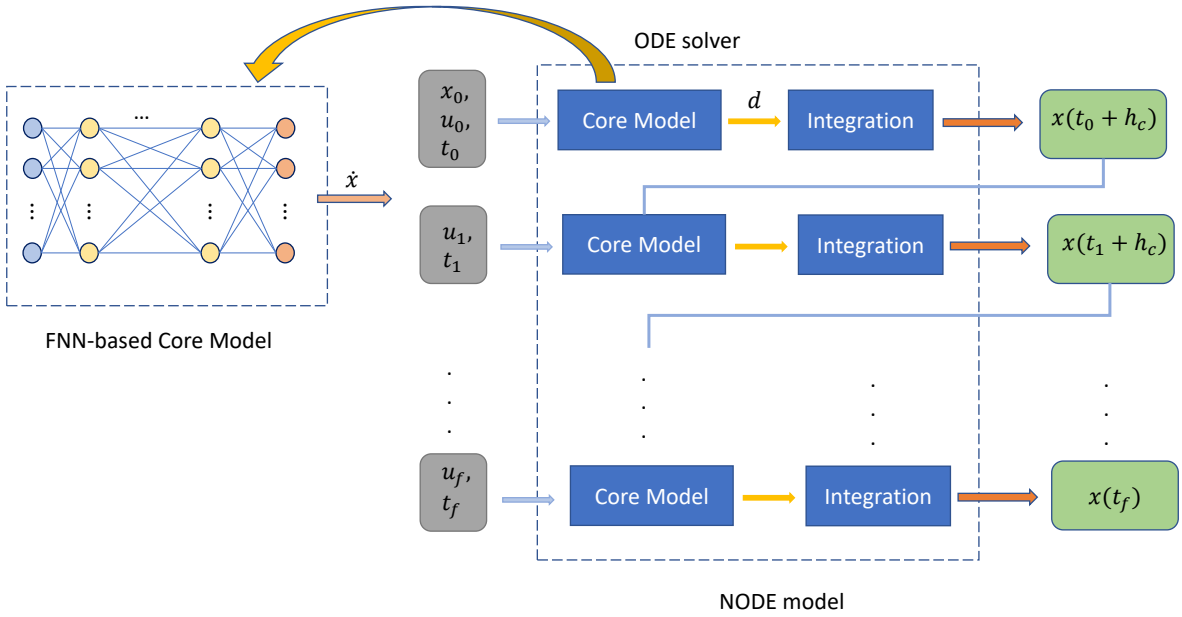


Figure 3.1: The architecture of the neural ordinary differential equation (NODE) model. The NODE model contains a nonlinear core function that maps the input to its hidden state, such that the time-series state prediction can be found by integrating the hidden state using an ODE solver. The core function used in this work is a feedforward neural network model whose structure is shown on the left. The blue circles in the FNN represent the input information represented in gray in the figure on the right-hand side.

3.3.2 Back-propagation

In the last subsection, we introduced the architecture of the NODE model and the unique characteristic of this model, which is the integration with an ODE solver within the neural network model. Since the ODE solver is involved in the training of the NODE model and impacts the backpropagation algorithm, the type of ODE solver (e.g., explicit Euler, Runge-Kutta, etc.) becomes one of the hyperparameters to be defined before training. This design differentiates the NODE from the common approach of training an FNN model with state time-derivatives and then computing the state prediction by using an explicit integration method with the output of the FNN

model. Specifically, for the second approach, time-derivatives of the state need to be included in the training data set and used as the reference data for the model training. Thus, the state time-derivatives have to be either measured (if measurable, e.g., velocity) or numerically approximated from the raw data of state measurements to develop the training data set. Additionally, in the second approach, the ODE solver is detached from the FNN model, so it is not involved in the model training.

The NODE model used in this study is an example of a supervised learning technique, which optimizes the parameters of a model by minimizing the difference between the model output and a reference data set. Training a neural network model includes two major steps: forward and backward propagation [28, 39, 132]. The forward propagation of the neural network model is a straightforward process that propagates the input through all the layers of the neural network model to compute the output [184]. Subsequently, the model output will be compared with the reference data set to calculate the loss of the model with respect to the user-defined loss function. The mean squared error (MSE) is a popular loss function for regression problems and is used in this study. Once calculated, the loss will be propagated backward, from the output layer to the input layer, to compute the derivative of the loss with respect to each weight parameter of the model. Finally, the weight parameters are optimized by the gradient descent method described by Eq. 3.7a below:

$$\mathbf{W}^{k+1} = \mathbf{W}^{k} - \alpha \frac{\partial Loss}{\partial \mathbf{W}} \tag{3.7a}$$

$$Loss = \frac{1}{2} \sum_{i=1}^{N_d} (\hat{Y}_i - Y_i)^{\top} (\hat{Y}_i - Y_i) \tag{3.7b}$$

where α is known as the learning rate, N_d denotes the number of data points in the training set,

and $\mathbf{W} = \{w_i \mid i = 1, 2, \ldots, N\}$ is the weight matrix containing all N weight parameters in the neural network model. $\frac{\partial Loss}{\partial \mathbf{W}}$ is the loss gradient with respect to each weight parameter, $Y_i = [y_{1,i}, y_{2,i}, \ldots, y_{n,i}]$ is the i^{th} state measurement, and $\hat{Y}$ is the model prediction of Y. In an FNN model, neurons are densely connected to each other, multiplying their respective weight values during the propagation phases. Therefore, the loss gradient with respect to the weight parameters can be easily computed by applying the chain rule iteratively from the output layer to the input layer. However, in NODE, the output given by the output layer of the FNN model is not directly used in computing the model loss. Considering the MSE of Eq. 3.7b and the NODE definition of Eq. 3.5, the NODE prediction is computed by the ODE solver, which means the chain rule can not be applied directly to propagate the gradient of loss.

3.3.2.1 Adjoint Sensitivity method

[26] proposed to use the adjoint sensitivity method to propagate the gradient of loss through the ODE solver in the NODE model. The adjoint sensitivity method, proposed in [120], is a popular method used in scientific research to efficiently compute the gradient of a model loss with respect to model parameters (or inputs). In [42], a detailed process to develop an adjoint model is introduced and an example of its application in meteorology demonstrated. [26] provided the mathematical derivation and proof of applying the adjoint method in the development of NODE. In short, an adjoint state will be created to represent any derivative information that is useful to train the neural network model. By doing so, the time-derivative of the adjoint state can be formulated based on the chain rule. Finally, the loss gradients can be computed by integrating the time-derivative of the adjoint states backward in time. A detailed implementation of this training

algorithm is demonstrated in the following steps:

1. **Define adjoint states:** The first step in training the NODE model is to define the adjoint states. One can start this step by first identifying what gradient information is needed to train their NN model. The first adjoint state is defined as the loss derivative with respect to the NODE output, which can be represented as $\mathbf{a} = \frac{\partial Loss}{\partial \hat{Y}}$. Since the loss function is usually selected to be an explicit quantity (e.g., MSE, MAE), the gradient $\frac{\partial Loss}{\partial \hat{Y}}$ can be computed analytically using the model prediction $\hat{Y}$. Subsequently, considering the case of using FNN as the core model, the second adjoint state is defined to represent the loss derivative with respect to the FNN weight parameters, that is, $\mathbf{a}_W = \frac{\partial Loss}{\partial W}$. In this study, the ODE function of the system is assumed to not contain any explicit term in time. Therefore, time is not an input of the core model, but it can be included following a similar approach. Lastly, all adjoint states are augmented into a column vector to perform the next step of the calculation.

2. **Set up the time-derivative of the adjoint states:** With an augmented adjoint state vector, $\mathbf{a}_{\text{aug}} = [\mathbf{a}\ \mathbf{a}_W]^\top$, following the derivation in [26], the time-derivative of the adjoint state can be expressed in the following form:

$$\frac{\partial \left(\frac{\partial Loss}{\partial \hat{Y}}(t)\right)}{\partial t} = \frac{\partial \mathbf{a}(t)}{\partial t} = -\mathbf{a}(t)\frac{\partial \mathfrak{f}}{\partial \hat{Y}} \tag{3.8a}$$

$$\frac{\partial \left(\frac{\partial Loss}{\partial W}(t)\right)}{\partial t} = \frac{\partial \mathbf{a}_W(t)}{\partial t} = -\mathbf{a}(t)\frac{\partial \mathfrak{f}}{\partial W} \tag{3.8b}$$

Since W is the weight vector that contains all the N weight parameters in the core model, Eq. 3.8b is a vector of N equations, which can be represented as $\frac{\partial \mathbf{a}_{W_i}(t)}{\partial t} = -\mathbf{a}(t)\frac{\partial \mathfrak{f}}{\partial W_i}, i =$

$1, \ldots, N$ respectively.

3. **Integrate backward in time:** Based on the definition of the adjoint states and their time-derivatives in the previous steps, we can compute the numerical values of the adjoint states by integrating Eq. 3.8 backward in time. Specifically, we denote the initial time step by t_0 and the final time step by t_f. Based on Eq. 3.7b, the adjoint state $\mathbf{a}(t_f)$ will simply be the sum of the model predictions at the final time step for each trajectory in the training set, i.e., $\sum \hat{Y}(t_f)$. With the adjoint state $\mathbf{a}(t_f)$ known, if we assume $\mathbf{a}_{\mathbf{w}}(t_f)$ to be zero, the adjoint states at t_0 can be found by the following expression, where t' is a notational substitute for t:

$$\mathbf{a}(t_0) = \mathbf{a}(t_f) - \int_{t_f}^{t_0} \mathbf{a}(t') \frac{\partial \mathfrak{f}}{\partial \hat{Y}} \, \mathrm{d}t' \tag{3.9a}$$

$$\mathbf{a}_W(t_0) = \mathbf{a}_W(t_f) - \int_{t_f}^{t_0} \mathbf{a}(t') \frac{\partial \mathfrak{f}}{\partial W} \, \mathrm{d}t' \tag{3.9b}$$

which can be solved with an ODE solver by approximating the partial derivative terms in Eq. 3.9 using the automatic differentiation method. At the end, the gradient of the loss with respect to weights at the initial time step, $\mathbf{a}_{\mathbf{w}}(t_0)$, is used to update the model based on Eq. 3.7a.

3.3.2.2 Automatic Differentiation

Automatic differentiation (AD) is an efficient and cheap method to approximate the gradient between two variables and is widely used in the development of neural network models. The AD method is well-developed and supported in modern ML Application Programming Interfaces (APIs). For example, AD is provided as the `autograd` function in PyTorch and the

`GradientTape` function in TensorFlow. Both ML APIs use a computational graph to implement the AD method. A simplified demonstration of a computational graph is a map that reflects all the connections in a neural network model and has a database for the derivative of various common math operations. For example, if a result (c) in a computational graph is computed by multiplying two inputs ($c = a \times b$), then the derivative of the output with respect to either of the inputs is simply the other input ($\frac{\partial c}{\partial a} = b$), and the derivative of the output with respect to all of its inputs will be stored in the computational graph. When performing the forward propagation of the neural network model, all necessary gradients will be computed and stored. Therefore, during backpropagation, the loss of the gradient with respect to any parameter can be systematically computed based on the chain rule. In this work, since we used PyTorch to develop our model, the derivative terms in Eq. 3.9 are approximated using the `autograd` function.

3.4 Lyapunov-based Model Predictive Control using NODE models

This section formulates the design of an LMPC designed using the NODE model of Eq. 3.5 to predict the future state trajectory, and then presents a closed-loop stability analysis of the nonlinear system of Eq. 3.1 under the proposed NODE-based LMPC. Due to the sample-and-hold implementation of the controller, the closed-loop states can only be driven to a small neighborhood around the origin. We clarify, for the subsequent propositions and proofs, that the core model of the NODE model represents the time-derivative of the state, i.e., $\dot{\hat{x}}$, which is also the right-hand side of the ODE model to be captured.

3.4.1 Lyapunov-based control using NODE models

We assume the existence of a stabilizing feedback controller $u = \Phi_{nn}(x) \in U$ that renders the origin of the core model of Eq. 3.6 exponentially stable in an open neighborhood around the origin $\hat{\phi}_u \in \mathbb{R}^n$ in the sense that there exist a continuously differentiable control Lyapunov function $\hat{V}(x)$ and positive constants $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$, $\hat{c}_4$ such that the following inequalities hold for all $x \in \hat{D}$:

$$\hat{c}_1|x|^2 \leq \hat{V}(x) \leq \hat{c}_2|x|^2 \tag{3.10a}$$

$$\dot{\hat{V}}(x) = \frac{\partial \hat{V}(x)}{\partial x} \mathfrak{f}(x, \Phi_{nn}(x)) \leq -\hat{c}_3|x|^2 \tag{3.10b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{c}_4|x| \tag{3.10c}$$

We begin by characterizing the region $\hat{\phi}_u$ where the constraints of Eq. 3.10 are met under the controller $u = \Phi_{nn}(x)$, followed by defining the closed-loop stability region of the NODE model of Eq. 3.6 to be a level set of the Lyapunov function inside $\hat{\phi}_u$: $\Omega_{\hat{\rho}} := \{x \in \hat{\phi}_u \mid \hat{V}(x) \leq \hat{\rho}\}$ where $\hat{\rho} > 0$. The assumptions of Eq. 3.2 and Eq. 3.10 are the stabilizability requirements of the nonlinear system of Eq. 3.1 and the NODE model of Eq. 3.6, respectively.

As the data set for constructing the NODE model is generated via open-loop simulations with $x \in \Omega_\rho$ and $u \in U$, we have $\Omega_{\hat{\rho}} \subseteq \Omega_\rho$. Moreover, there exist positive constants M_{nn} and L_{nn} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|\mathfrak{f}(x, u)| \leq M_{nn} \tag{3.11a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \mathfrak{f}(x, u) - \frac{\partial \hat{V}(x')}{\partial x} \mathfrak{f}(x', u) \right| \leq L_{nn}|x - x'| \tag{3.11b}$$

The following proposition is developed to demonstrate that the feedback controller $u = \Phi_{nn}(x)$ can stabilize the nominal system of Eq. 3.1, despite the model mismatch between Eq. 3.1 and the NODE model of Eq. 3.6, if the modeling error is sufficiently small.

Proposition 1. (c.f. proposition 2 in [167]) *Under the assumption that the origin of the closed-loop NODE system of Eq. 3.6 is rendered exponentially stable under the controller* $u = \Phi_{nn}(x) \in U \ \forall\, x \in \Omega_{\hat{\rho}}$*, if there exists a positive real number* $\gamma < \hat{c}_3/\hat{c}_4$ *that constrains the modeling error* $|\nu| = |F(x,u) - \mathfrak{f}(x,u)| \leq \gamma|x|, \ \forall\, u \in U$ *and* $\forall\, x \in \Omega_{\hat{\rho}}$*, then the origin of the nominal closed-loop system of Eq. 3.1 under* $u = \Phi_{nn}(x) \in U$ *is also exponentially stable* $\forall\, x \in \Omega_{\hat{\rho}}$*.*

Proof. To prove that the origin of the nominal system of Eq. 3.1 may be rendered exponentially stable $\forall\, x \in \Omega_{\hat{\rho}}$ under the controller designed using the NODE model of Eq. 3.5, we prove that $\dot{\hat{V}}$ for the nominal system of Eq. 3.1 is still rendered negative for all $x \in \Omega_{\hat{\rho}}$ under the controller $u = \Phi_{nn}(x)$. The time-derivative of $\hat{V}$ is computed based on Eq. 3.10b and Eq. 3.10c, as follows:

$$\begin{aligned}
\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{nn}(x)) \\
&= \frac{\partial \hat{V}(x)}{\partial x} \big(\mathfrak{f}(x, \Phi_{nn}(x)) + F(x, \Phi_{nn}(x)) - \mathfrak{f}(x, \Phi_{nn}(x)) \big) \\
&\leq -\hat{c}_3 |x|^2 + \hat{c}_4 |x| \big| F(x, \Phi_{nn}(x)) - \mathfrak{f}(x, \Phi_{nn}(x)) \big| \\
&\leq -\hat{c}_3 |x|^2 + \hat{c}_4 \gamma |x|^2
\end{aligned} \tag{3.12}$$

By choosing the modeling error γ to satisfy $\gamma < \hat{c}_3/\hat{c}_4$, it can be ensured that $\dot{\hat{V}} \leq -\tilde{c}_3 |x|^2 \leq 0$ where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4 \gamma > 0$. Consequently, the closed-loop state of the nominal system of Eq. 3.1 converges to the origin $\forall\, x_0 \in \Omega_{\hat{\rho}}$ under $u = \Phi_{nn}(x) \in U$. □

Remark 14. *In this work, the terminology of "modeling error" is used to describe the difference*

between Eq. 3.1 and Eq. 3.6 because the core model of the trained NODE model is expected to capture the right-hand-side of the ODE function of Eq. 3.1. However, the NODE model is trained using the measured states, i.e., the solution of Eq. 3.1b, such that the output of the core model of Eq. 3.6 is the hidden state of the NODE model. Hence, the training loss of the NODE model refers to the error in the state, while the modeling error refers to the error in the time-derivative.

3.4.2 Sample-and-hold implementation of Lyapunov-based MPC

Since the Lyapunov-based MPC designed using the NODE model of Eq. 3.5 is implemented in a sample-and-hold fashion, in the next two propositions, the sample-and-hold properties of the Lyapunov-based controller $u = \Phi_{nn}(x)$ are derived. In particular, the following proposition derives an upper bound for the error between the states calculated by the nominal system of Eq. 3.1 and the states predicted by the NODE model of Eq. 3.5.

Proposition 2. (c.f. proposition 3 in [167]) *For the nonlinear system $\dot{x} = F(x, u)$ of Eq. 3.1 and the NODE core model $\dot{\hat{x}} = \mathfrak{f}(\hat{x}, u)$ of Eq. 3.6 with the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, there exist a function $f_w(\cdot)$ of class $\mathcal{K}$ and a positive constant κ such that the following inequalities hold $\forall\, x, \hat{x} \in \Omega_{\hat{\rho}}$:*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{\nu_m}{L_x}(\mathrm{e}^{L_x t} - 1) \tag{3.13a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{3.13b}$$

Proof. The error vector between the solutions of the nonlinear system of Eq. 3.1 and the NODE model of Eq. 3.5 is defined as $e(t) = x(t) - \hat{x}(t)$, whose time-derivative can be calculated as

follows:

$$\begin{aligned} |\dot{e}(t)| &= |F(x,u) - \mathfrak{f}(\hat{x},u)| \\ &\leq |F(x,u) - F(\hat{x},u)| + |F(\hat{x},u) - \mathfrak{f}(\hat{x},u)| \end{aligned} \tag{3.14}$$

In Eq. 3.14, the first term can be bounded using Eq. 3.3b as follows:

$$|F(x,u) - F(\hat{x},u)| \leq L_x|x(t) - \hat{x}(t)|, \qquad \forall\, x, \hat{x} \in \Omega_{\hat{\rho}}, \tag{3.15}$$

while the second term $|F(\hat{x},u) - \mathfrak{f}(\hat{x},u)|$ represents the modeling error, which is bounded by $|\nu| \leq \nu_m \; \forall \hat{x} \in \Omega_{\hat{\rho}}$. As a result, based on Eq. 3.15 and the bounded modeling error, $\dot{e}(t)$ is bounded as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L_x|x(t) - \hat{x}(t)| + \nu_m \\ &\leq L_x|e(t)| + \nu_m \end{aligned} \tag{3.16}$$

The norm of the error vector can be bounded as follows for all $x(t), \hat{x}(t) \in \Omega_{\hat{\rho}}$ using the zero initial condition (i.e., $e(0) = 0$):

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{\nu_m}{L_x}(\mathrm{e}^{L_x t} - 1) \tag{3.17}$$

Finally, to derive Eq. 3.13b $\forall x, \hat{x} \in \Omega_{\hat{\rho}}$, we derive the Taylor series expansion of $\hat{V}(x)$ around $\hat{x}$,

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{3.18}$$

where κ is a positive real number. Using Eq. 3.10a and Eq. 3.10c, it follows that

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2, \tag{3.19}$$

which completes the proof of 2. □

The final proposition below is developed to prove that the closed-loop state of the actual nonlinear system of Eq. 3.1 remains bounded in $\Omega_{\hat{\rho}}$ for all times, and can be ultimately bounded in a small neighborhood around the origin, denoted by $\Omega_{\rho_{\min}}$, under the sample-and-hold implementation of the Lyapunov-based controller $u = \Phi_{nn}(x) \in U$.

Proposition 3. *Consider the nonlinear system of Eq. 3.1 under the controller* $u = \Phi_{nn}(\hat{x}) \in U$ *that meets the conditions of Eq. 3.10 and is designed to stabilize the NODE system of Eq. 3.6. The controller is implemented in a sample-and-hold fashion, such that* $u(t) = \Phi_{nn}(\hat{x}(t_k))$, $\forall\, t \in [t_k, t_{k+1})$, *where* $t_{k+1} := t_k + \Delta$. *Furthermore, let* $\epsilon_s,\ \epsilon_w > 0$, $\Delta > 0$ *and* $\hat{\rho} > \rho_{\min} > \rho_{sp} > \rho_s$ *satisfy*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}M_{nn}\Delta \leq -\epsilon_s \tag{3.20a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x M\Delta \leq -\epsilon_w \tag{3.20b}$$

and

$$\rho_{sp} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{3.21a}$$

$$\rho_{\min} \geq \rho_{sp} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{3.21b}$$

Based on the above assumptions, for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, *the following inequality holds:*

$$\hat{V}(x(t)) \leq \hat{V}\left(x(t_k)\right), \forall t \in [t_k, t_{k+1}) \tag{3.22}$$

and the state $x(t)$ of the nonlinear system of Eq. 3.1 is bounded in the level set $\Omega_{\hat{\rho}}$ for all times and ultimately trapped in the smaller level set $\Omega_{\rho_{\min}}$.

Proof. Part 1: Assuming $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, it will be first shown that the value of the Lyapunov function $\hat{V}(\hat{x})$ is decreasing under the controller $u(t) = \Phi_{nn}(x(t_k)) \in U \; \forall \, t \in [t_k, t_{k+1})$, where $x(t)$ and $\hat{x}(t)$ denote the solutions of the nonlinear system of Eq. 3.1 and the NODE system of Eq. 3.5, respectively. The time-derivative of the Lyapunov function along the trajectory $\hat{x}(t)$ of the NODE model of Eq. 3.5 can be written $\forall t \in [t_k, t_{k+1})$ as

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\ &= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\ &\quad - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k)))\end{aligned} \tag{3.23}$$

where the first term can be bounded as follows using Eq. 3.10a and Eq. 3.10b:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) \leq &- \frac{\hat{c}_3}{\hat{c}_2} \rho_s + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\ &- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} \mathfrak{f}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k)))\end{aligned} \tag{3.24}$$

In Eq. 3.24, bounding the difference using the Lipschitz condition of Eq. 3.11 with the fact that $\hat{x} \in \Omega_{\hat{\rho}}$, $u \in U$, $\dot{\hat{V}}(\hat{x}(t))$ can be upper-bounded $\forall \, t \in [t_k, t_{k+1})$:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) &\leq - \frac{\hat{c}_3}{\hat{c}_2} \rho_s + L_{nn} |\hat{x}(t) - \hat{x}(t_k)| \\ &\leq - \frac{\hat{c}_3}{\hat{c}_2} \rho_s + L_{nn} M_{nn} \Delta\end{aligned} \tag{3.25}$$

Therefore, the satisfaction of the condition of Eq. 3.20a ensures that the following inequality holds

$\forall\, \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s},\ \forall\, t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(\hat{x}(t)) \leq -\epsilon_s \tag{3.26}$$

By integrating the aforementioned differential equation over the time interval $t \in [t_k, t_{k+1})$ with respect to time, it can be deduced that $V(\hat{x}(t_{k+1})) \leq V(\hat{x}(t_k)) - \epsilon_s \Delta$. So far, we have demonstrated that, for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the closed-loop state of the NODE system of Eq. 3.5 remains confined within the closed-loop stability region $\Omega_{\hat{\rho}}$ at all times and progresses towards the origin under the controller $u = \Phi_{nn}(\hat{x}) \in U$ when implemented in a sample-and-hold approach.

It should be noted that Eq. 3.26 may not hold when $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$, meaning that the state may exit Ω_{ρ_s} within a single sampling period. Therefore, we establish another region $\Omega_{\rho_{sp}}$ based on Eq. 3.21a to ensure that the closed-loop state $\hat{x}(t)$ of the NODE model does not depart from $\Omega_{\rho_{sp}}$ over a single sampling period, i.e., during $t \in [t_k, t_{k+1})$, $\forall u \in U$, $\forall \hat{x}(t_k) \in \Omega_{\rho_s}$. If the state $\hat{x}(t_{k+1})$ exits Ω_{ρ_s}, Eq. 3.26 is satisfied again at $t = t_{k+1}$, thereby reactivating the controller $u = \Phi_{nn}(x(t_{k+1}))$ and directing the state towards Ω_{ρ_s} during the following sampling period. Consequently, it is demonstrated that the state approaches $\Omega_{\rho_{sp}}$ for the closed-loop NODE system of Eq. 3.5 for all $\hat{x}_0 \in \Omega_{\hat{\rho}}$. In Part 2, we demonstrate that the closed-loop state of the actual nonlinear process of Eq. 3.1 can also be confined within $\Omega_{\hat{\rho}}$ for all times and eventually contained within a small neighborhood around the origin with the sample-and-hold implementation of the controller $u = \Phi_{nn}(x) \in U$.

Part 2: We repeat the analysis carried out for the NODE system of Eq. 3.5. First, we suppose $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and derive the following expression for the time-derivative of $\hat{V}(x)$ for

the nonlinear system of Eq. 3.1:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\ &= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\ &\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)))\end{aligned} \tag{3.27}$$

From Eq. 3.12, it can be inferred that $\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) \leq -\tilde{c}_3 |x(t_k)|^2 \; \forall \, x \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. By utilizing Eq. 3.10a and the definition of Lipschitz continuity in Eq. 3.11, the following inequality can be established $\forall \, t \in [t_k, t_{k+1})$ and $\forall \, x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x |x(t) - x(t_k)| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x M \Delta\end{aligned} \tag{3.28}$$

Thus, if the condition of Eq. 3.20b is fulfilled, the inequality below is valid $\forall \, x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}, \forall \, t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{3.29}$$

The above differential equation may be integrated in time between any two points within the following sampling period, i.e., $[t_k, t_{k+1})$ to obtain the following inequalities for the Lyapunov function $\hat{V}$ for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$\hat{V}(x(t_{k+1})) \leq V(x(t_k)) - \epsilon_w \Delta \tag{3.30}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)), \quad \forall \, t \in [t_k, t_{k+1}) \tag{3.31}$$

As a result, the state of the closed-loop system of Eq. 3.1 stays within $\Omega_{\hat{\rho}}$ for all time. In addition, the controller $u = \Phi_{nn}(x)$ can continue to steer the state of the nonlinear system of Eq. 3.1 toward the origin within each sampling period. Furthermore, if the initial state $x(t_k)$ satisfies $x(t_k) \in \Omega_{\rho_s}$, then it was already demonstrated in Part 1 that the state of the NODE model of Eq. 3.5 is confined within $\Omega_{\rho_{sp}}$ for one sampling period. Given the bounded modeling error between the actual nonlinear system of Eq. 3.1 and the NODE model of Eq. 3.5 described by Eq. 3.13a, there is a compact set $\Omega_{\rho_{\min}} \supset \Omega_{\rho_{sp}}$ satisfying Eq. 3.21b such that the state of the nonlinear system of Eq. 3.1 remains within $\Omega_{\rho_{\min}}$ for one sampling period if the state of the NODE model of Eq. 3.5 is constricted within $\Omega_{\rho_{sp}}$. If the state $x(t)$ enters $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_s}$, we have already demonstrated that Eq. 3.31 holds, and thus, under $u = \Phi_{nn}(x)$, the state will be driven back towards the origin during the next sampling period, ultimately constricting the closed-loop system to remain within $\Omega_{\rho_{\min}}$. Thus, the proof of Proposition 3 is completed, having shown that for any $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, the closed-loop state trajectories of the nonlinear system described by Eq. 3.1 remain within $\Omega_{\hat{\rho}}$ and ultimately within $\Omega_{\rho_{\min}}$ if the assumptions of Proposition 3 are satisfied. □

3.4.3 Lyapunov-based MPC formulation

Model predictive control is an advanced process control technique that computes the control action by solving an optimization problem based on a given predictive model and feedback measurement. A Lyapunov-based MPC is a class of MPC with additional constraints based on the value of the Lyapunov function and its time-derivative at the current state. The optimization

problem of LMPC can be written as follows:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N_h}} L(\hat{x}(t), u(t))\, \mathrm{d}t \tag{3.32a}$$

$$\text{s.t.} \quad \hat{x}(t) = F_{nn}(x_0, u, t_k, t_{k+N_h}) \tag{3.32b}$$

$$u(t) \in U,\ \forall\, t \in [t_k, t_{k+N_h}) \tag{3.32c}$$

$$\hat{x}(t_k) = x(t_k) \tag{3.32d}$$

$$\dot{\hat{V}}(\hat{x}, u) \leq -k\hat{V}(\hat{x}),\ \text{if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{sp}} \tag{3.32e}$$

$$\hat{V}(\hat{x}) \leq \rho_{sp},\ \forall\, t \in [t_k, t_{k+N_h}),\ \text{if } x(t_k) \in \Omega_{\rho_{sp}} \tag{3.32f}$$

where $L(\cdot)$ is the cost function based on the state and control input values, such that the objective of the optimization problem is to minimize the integral of the cost function in the time span between t_k and t_{k+N_h}, where N_h is the prediction horizon, which is an integer multiple of Δ. $S(\Delta)$ represents the set of piecewise constant functions with a period Δ, which the input u is restricted to. F_{nn} represents the NODE model, described by Eq. 3.5, that gives as its output the state prediction $\hat{x}(t)$ over the prediction horizon by integrating the core model of Eq. 3.6 starting from the initial condition of Eq. 3.32d, which is the state measurement at t_k. U denotes the allowable range of the control action u, and $\Omega_{\rho_{sp}} := \{x \in \phi(x) \mid \hat{V}(x) \leq \rho_{sp},\ \rho_{sp} < \rho\}$ is a subset of Ω_ρ that defines the process state region considered to be "close enough" to the set-point when deploying the LMPC for set-point tracking. The goal of the LMPC is to calculate the optimal sequence of control actions $u = u^*(t),\ t \in [t_k,\ t_{k+N_h})$ and apply the first move of the sequence, $u^*(t_k)$, over the next sampling period. Once the process evolves for one sampling period, the LMPC is resolved again at the next sampling period.

The time-derivative of the Lyapunov function depends on the design of the function $\hat{V}(x)$. Considering the design to be $\hat{V}(x) = x^\top P x$ where P is a user-defined positive definite matrix, $\dot{\hat{V}}(x)$ can be expressed as $2x^\top P\dot{x}$. The time-derivative of the process states, $\dot{x}$, can be easily found if an explicit ODE of the system is known. In the case of an NODE model, the time-derivative of the process states can be approximated by the hidden state, which is the output of the core model. Therefore, numerical approximation of the output derivative ($\dot{x}$) by methods such as finite-differences is not necessary.

Remark 15. *As we demonstrate in the application section, quadratic Lyapunov functions can be used to effectively design the Lyapunov-based stability constraints for the LMPC. It is important to note that the operating region is not limited by the choice of a quadratic Lyapunov function. Non-quadratic Lyapunov functions could be considered for this task as well but they are harder to construct. The design of Lyapunov functions is an important research area in nonlinear systems and control; various methods have been proposed to design a Lyapunov function, for example [49, 52].*

3.4.4 Closed-loop stability analysis

The LMPC formulation presented in Eq. 3.32 is used to derive the following theorem, which guarantees recursive feasibility of the LMPC optimization problem and also closed-loop stability of the actual nonlinear system of Eq. 3.1 under the sample-and-hold implementation of the resulting optimal control actions.

Theorem 1. *Assuming the controller $\Phi_{nn}(x)$ satisfies Eq. 3.10, the closed-loop system of Eq. 3.1*

under the LMPC of Eq. 3.32 is considered. Let $\Delta > 0, \epsilon_s > 0$*, and* $\hat{\rho} > \rho_{\min} > \rho_{sp} > \rho_s$ *be such that they satisfy Eq. 3.21a and 3.21b. If the conditions of Propositions 2 and 3 are met, a feasible solution for the optimization problem of Eq. 3.32 always exists for any initial state* $x_0 \in \Omega_{\hat{\rho}}$*. Moreover, it is guaranteed that the LMPC of Eq. 3.32 maintains* $x(t) \in \Omega_{\hat{\rho}}$ *for all* $t \geq 0$*, and that* $x(t)$ *of the closed-loop system of Eq. 3.1 eventually converges to* $\Omega_{\rho_{\min}}$ *.*

Proof. We begin by establishing the recursive feasibility of the optimization problem in Eq. 3.32 for all states $x \in \Omega_{\hat{\rho}}$. In particular, if at time t_k, $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{sp}}$, then the control action $u(t) = \Phi_{nn}(x(t_k)) \in U$, $t = [t_k, t_{k+1})$ computed using the state measurement $x(t_k)$ satisfies the input constraint of Eq. 3.32c and the Lyapunov-based constraint of Eq. 3.32e. Furthermore, if $x(t_k) \in \Omega_{\rho_{sp}}$, the control actions obtained from $\Phi_{nn}(x(t_{k+i}))$, $i = 0, 1, ..., N_h - 1$ comply with the input constraint of Eq. 3.32c and the Lyapunov-based constraint of Eq. 3.32f since Proposition 3 shows that the states predicted by the NODE model of Eq. 3.32b remain inside $\Omega_{\rho_{sp}}$ under the controller $\Phi_{nn}(x)$. Hence, if $x(t) \in \Omega_{\hat{\rho}}$ for all times, the LMPC optimization problem of Eq. 3.32 is recursively feasible for all initial states $x_0 \in \Omega_{\hat{\rho}}$.

We will show that $\forall\, x_0 \in \Omega_{\hat{\rho}}$, the state of the closed-loop system of Eq. 3.1 under the LMPC scheme of Eq. 3.32 remains bounded in $\Omega_{\hat{\rho}}$ $\forall\, t$ and ultimately converges to a small neighborhood around the origin $\Omega_{\rho_{\min}}$ as described by Eq. 3.21b.

Assume $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{sp}}$ at time t_k. In this case, the constraint of Eq. 3.32e is activated, and the control action u is computed to decrease the value of $\hat{V}(\hat{x})$ based on the predicted states obtained from the NODE model of Eq. 3.32b over the next sampling period. Furthermore, Eq. 3.31 shows that, if the constraint of Eq. 3.32e is met, then $\dot{\hat{V}}(x) \leq -\epsilon_w$ holds for $t \in [t_k, t_{k+1})$ when

applying the control action $u^*(t_k)$ to the nonlinear system of Eq. 3.1. Consequently, the value of the Lyapunov function calculated using the state of the actual nonlinear system of Eq. 3.1, $\hat{V}(x)$, decreases within the following sampling period, implying that the closed-loop state can be driven into $\Omega_{\rho_{sp}}$ within a finite number of sampling periods.

Once the state enters $\Omega_{\rho_{sp}}$, the constraint of Eq. 3.32f is activated to ensure that the predicted states of the NODE model of Eq. 3.32b remain in $\Omega_{\rho_{sp}}$ throughout the prediction horizon. While there may exist a mismatch between the NODE model of Eq. 3.32b and the nonlinear system of Eq. 3.1, based on the results of Proposition 3, we can guarantee that the state $x(t)$ of the nonlinear system of Eq. 3.1 remains bounded in $\Omega_{\rho_{\min}}$ $\forall\, t \in [t_k, t_{k+1})$ as characterized by Eq. 3.21b if the state predicted by the NODE model of Eq. 3.32b stays in $\Omega_{\rho_{sp}}$.

Thus, at the next sampling period $t = t_{k+1}$, if the state $x(t_{k+1})$ is still within $\Omega_{\rho_{sp}}$, then the constraint of Eq. 3.32f ensures that the predicted state $\hat{x}$ of the NODE model of Eq. 3.32b remains in $\Omega_{\rho_{sp}}$, thereby keeping the state x of the actual nonlinear system of Eq. 3.1 inside $\Omega_{\rho_{\min}}$. However, if the state exits $\Omega_{\rho_{sp}}$ such that $x(t_{k+1}) \in \Omega_{\rho_{\min}} \backslash \Omega_{\rho_{sp}}$, we can retrace the proof for $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{sp}}$ to show that the state will once again be driven toward the origin since the constraint of Eq. 3.32e will be triggered. This concludes the proof of the states of the actual nonlinear system of Eq. 3.1 being bounded under the sample-and-hold implementation of the controller $u = \Phi_{nn}(x)$ and being ultimately driven to a small neighborhood around the origin $\Omega_{\rho_{\min}}$ $\forall\, x_0 \in \Omega_{\hat{\rho}}$. □

Table 3.1: Parameters of the CSTR example.

$T_0 = 300$ K	$k_0 = 8.46 \times 10^6$ m^3/kmol hr
$C_p = 0.231$ kJ/kg K	$\rho_L = 1000$ kg/m^3
$F = 5$ m^3/hr	$E = 5 \times 10^4$ kJ/kmol
$R = 8.314$ kJ/kmol K	

3.5 Application of NODE-based Model Predictive Control in Chemical Process

In this section, a chemical process involving a continuous stirred tank reactor (CSTR), which facilitates a reaction converting reactant A to product B, is utilized to demonstrate the development and application of NODE-based LMPC in a chemical process setting. Based on mass and energy balances, the following ODEs are used to describe the CSTR system:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \mathrm{e}^{\frac{-E}{RT}} C_A^2 \tag{3.33a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 \mathrm{e}^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{3.33b}$$

where C_{A0} is the concentration of reactant A in the feed flow, while T and C_A are the temperature and the concentration of A, respectively, in the CSTR. ρ_L and C_p represent the density and heat capacity of the liquid mixture in the CSTR, respectively, and are assumed to be constant. ΔH denotes the enthalpy of the reaction. The constant parameters are listed in Table 3.1. The unstable steady state of the system at $(C_A = 1.95 \text{ kmol/m}^3, T = 402 \text{ K})$ is selected as the set-point of this example, and the control objective of the LMPC is to maintain the state of the system around this set-point. The manipulated control variables in this example are the inlet concentration of

reactant A and the heat duty of the coolant jacket of the CSTR, with the steady state control actions set to $C_{A0s} = 4\ \mathrm{kmol/m^3}, Q_s = 0\ \mathrm{kJ/hr}$. The bounds of the manipulated control variables are $C_{A0} \in C_{A0s} \pm 3.5\ \mathrm{kmol/m^3}$ and $Q \in Q_s \pm 5 \times 10^5\ \mathrm{kJ/hr}$.

3.5.1 Noise-free Example

3.5.1.1 Data Collection and Preprocessing

Training data used to develop the NODE model is obtained by performing open-loop simulations. Specifically, the first step of the data collection is to define the open region D, which requires the design of a Lyapunov function $\hat{V}(x)$. In our work, the control Lyapunov function is designed to be $\hat{V}(x) = x^\top P x$, where P is the positive definite matrix $P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix}$. Subsequently, a level set where $\rho = 375$ is selected to be the closed-loop stability region Ω_p, which is shown as the ellipse in Fig. 3.2. Since the desired region of operation is Ω_p, this is also chosen to be the sampling region for data collection.

After determining the sampling region Ω_ρ, numerous open-loop state trajectories are obtained by integrating Eq. 3.33 from randomly drawn initial states under random, constant control inputs using the explicit Euler method over a time span of 0.045 hr. A very small step size (i.e., $h_c \ll 0.005\ \mathrm{hr}$) is used in the explicit Euler method to generate the state trajectories, but since the process is assumed to have a sampling period of 0.005 hr, the generated state trajectories are then evenly down-sampled to have a time interval of 0.005 hr between data points. As a result, the training data set is made up of various state trajectories generated from a wide range of initial conditions and control inputs. Each trajectory in the training data set contains 10 samples or data points.

Moreover, the data is generated in deviation form, $x = (C_A - C_{As}, T - T_s)$ and $u = (C_{A0} - C_{A0s}, Q - Q_s)$, such that the steady state is at the origin (i.e., $x_s = (0, 0),\ u_s = (0, 0)$). Following this method, a training data set containing 1000 trajectories was collected and found to be enough to train the NODE model in this example.

Similar to other neural network models, data needs to be scaled prior to being used to train an NODE model. The scaling step is a part of data preprocessing. The MaxAbs scaler, which scales the data set by dividing each variable by the maximum absolute value of each variable in the data set (without any subtraction), respectively, is adopted in this work to maintain the steady-state of the scaled data set at the origin. As a result, the training data is scaled to a range between -1 and 1. Besides scaling the data, the training data set is split into two parts, such that 80% of the trajectories are used to train the model and the rest are used to validate the model performance. Specifically, the model weights are updated based on the loss with respect to 80% of all trajectories and the remaining 20%, usually named the validation set, is not included in the calculation of the loss for the weight update but are used to compute the validation loss, which is an important metric to ensure that the model does not overfit the training data. Lastly, there are additional 100 trajectories that are reserved for testing the trained model's performance, which is usually called the test set, and these are generated in the same manner as the training data.

The data preprocessing step, including scaling and splitting the data, is handled using Scikit-learn, a popular Python-based ML package. Specifically, conventionally, in most ML packages, the inputs of the neural network model and the reference/output data are first augmented into two different tensors. After that, the Scikit-learn scaler function is applied to fit and transform the input and output tensors, respectively. However, this scaling strategy will cause a mismatch between

the model input and first time step of the output trajectory. Specifically, the prediction of our NODE model is designed to include the initial value of the state, which is provided by the input tensor during the model training. Therefore, the initial value of each trajectory in the output tensor corresponds to t_0 and should be identical to the states values in the input tensor. If the tensors are scaled separately, scaling factors (e.g., mean, maximum absolute value, etc.) will be different for the input and the output tensors. This is because, using time-series data as an example, data in the output tensor is evolved in time from the data in the input tensor, which is very likely to yield different statistics for the two tensors. If the different means are subtracted from the input and output tensors, the initial value of each trajectory corresponding to t_0 will be different for each tensor. To avoid this mismatch, the same scaling factor should be applied to the variables representing the same physical quantities. Lastly, to ensure no information leakage occurs during the model building process, the maximum absolute value used to scale the data should be based on only the 80% of the training data set.

Remark 16. *In practice, the knowledge of the operating region can be obtained from the process design and operational purpose as well as past experience of operating the process. For example, if the goal is developing a process control system for an existing chemical process, the operating region of the process should be determined from the previous operation experience (if the objective is to update an existing control system) or from process design and simulation using reliable software used in industry such as Aspen Hysys or Aveva PRO/II (if the objective is to develop and implement a control system to a new process). The training data in those cases will be collected from the past operation or process simulation, respectively.*

3.5.1.2 NODE training

In this subsection, the details of the NODE model development using PyTorch are provided. Specifically, the core model of the NODE is chosen to be an FNN model to map neural network inputs to the hidden state of the NODE model. The core model has two hidden layers and each hidden layer contains 64 neurons activated by the hyperbolic tangent (tanh) function. [41] suggested that the NODE model should be differentiable everywhere, but activation functions like ReLU, which are not, have been used in NODE-based research [41, 78]. Nevertheless, to avoid any potential failure caused by the lack of differentiability of the activation function, tanh is used as the activation function in our core model. The Adaptive Moment Estimation (ADAM) optimizer is used to update the weight matrices in the core model. Mini-batches are used for the gradient descent method with a batch size of 32 trajectories. Lastly, an explicit Euler solver is used as the ODE solver in the NODE model.

Hyperparameter tuning is a critical step in neural network development and is done via a coarse, exploratory search followed by manual fine-tuning in this work. Specifically, we observed that using two hidden layers in the core model can significantly improve the learning ability of the NODE model but having more than two hidden layers did not provide any significant improvement despite the uptick in computational resource usage. For the number of neurons to be used in the hidden layers, we applied a two-dimensional grid search using 8, 16, 32, 64, and 128 neurons for each hidden layer and assessed the model performance at each combination. It was found that using more than 64 neurons did not significantly improve the model performance. Therefore, we used 64 neurons in our hidden layer. The NODE model is trained with 300 epochs, and the testing

loss of the trained model is 3.1×10^{-4}. Furthermore, the mean absolute errors were 0.015 $\mathrm{kmol}/\mathrm{m}^3$ and 0.8 K for the prediction of C_A and T, respectively.

Remark 17. *The model obtained using the aforementioned hyperparameter tuning strategy may not be the most accurate possible model. However, due to the low dimension of our system and the complexity of the neural network structure, further fine-tuning of the hyperparameter is not expected to significantly improve the model performance. We note that finding the best possible model or the most effective hyperparameter tuning strategy are not the major objectives of this study.*

Remark 18. *Detailing the coding implementation of the NODE is not the objective of this study. However, since the NODE is a recently proposed model, such that its training is not fully supported in the commonly used ML APIs such as PyTorch and TensorFlow, we outline some key steps of our implementation. An official NODE python package,* `torchdiffeq`*, is provided in [26]. In our study, the intermediate steps of the data sequence are important when evaluating the loss of the model. Therefore, the integral of the adjoint state* $\mathbf{a}(t)$ *is updated at each intermediate observation during the backpropagation. We developed our code based on an open-source GitHub project [152]. Modifications needed to be added to the ODE solver and the backpropagation function to correctly handle the control actions in the neural network input.*

3.5.1.3 NODE-based LMPC performance

After training the NODE model, we conduct open-loop simulations using the NODE model and benchmark it against the first-principles (FP) equation to evaluate its prediction accuracy. Specifically, open-loop simulations are conducted under fixed control inputs over two sampling

periods for both the NODE model and the FP model and the state trajectories compared. The simulations start from randomly selected initial conditions inside the stability region Ω_ρ. Subsequently, a random, constant control input is applied to the process for two sampling periods. Fig. 3.2 depicts the open-loop trajectories for both the NODE model and the FP model of Eq. 3.33 under identical input signals as described. The close agreement between the state trajectories predicted by both models throughout the two sampling periods supports the fact that the NODE model prediction has been trained to a high level of accuracy.

After ensuring the accuracy of the NODE model, closed-loop simulations under the LMPC of Eq. 3.32 using the NODE process model are conducted. The objective function of the LMPC is defined to be $L(x,u) = x^\top \bar{Q} x + u^\top R u$, where $\bar{Q}$ and R are the parameter matrices of the state and input penalty terms in the objective function, respectively. Fig. 3.3 illustrates the state and input profiles of the CSTR in closed-loop under the NODE-based LMPC, where the process state is brought from a randomly drawn initial state $x_0 = (C_A = 1.6\ \mathrm{kmol/m^3}, T = -64.5\ \mathrm{K})$ to the steady-state set-point and maintained within $\Omega_{\rho_{sp}}$. Furthermore, the closed-loop performance of the NODE-based LMPC and the FP-based LMPC are found to be very similar, indicating that the NODE-based LMPC can perform as well as the FP-based LMPC due to the high accuracy of the NODE model. Finally, Fig. 3.4 shows closed-loop state-space trajectories from several random initial conditions in Ω_ρ under the NODE-based LMPC, all of which are found to be successfully stabilized around the set-point.

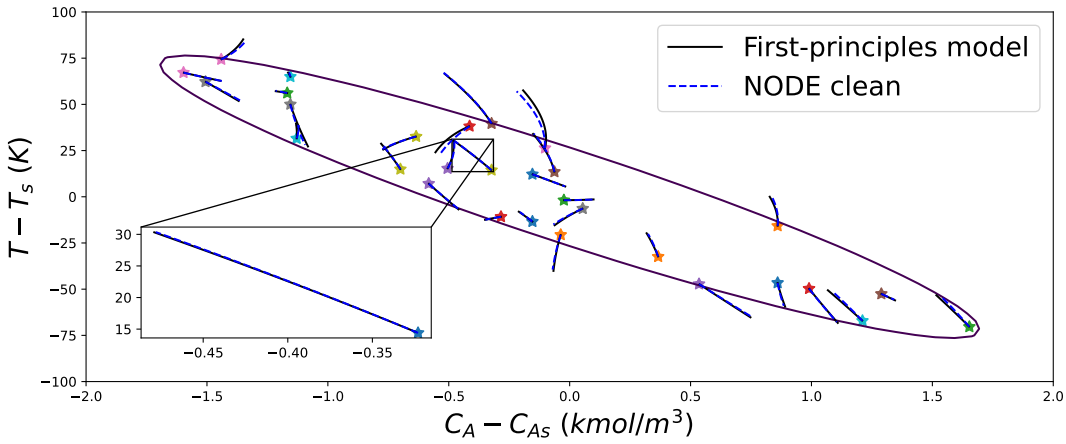


Figure 3.2: Open-loop state-space trajectories of the CSTR given by the first-principle model (black line) and the trained NODE model (blue dash). The purple ellipse denotes the boundary of the pre-defined stability region, Ω_ρ, of the CSTR system. Star icons are the initial state pairs randomly drawn within the stability region.

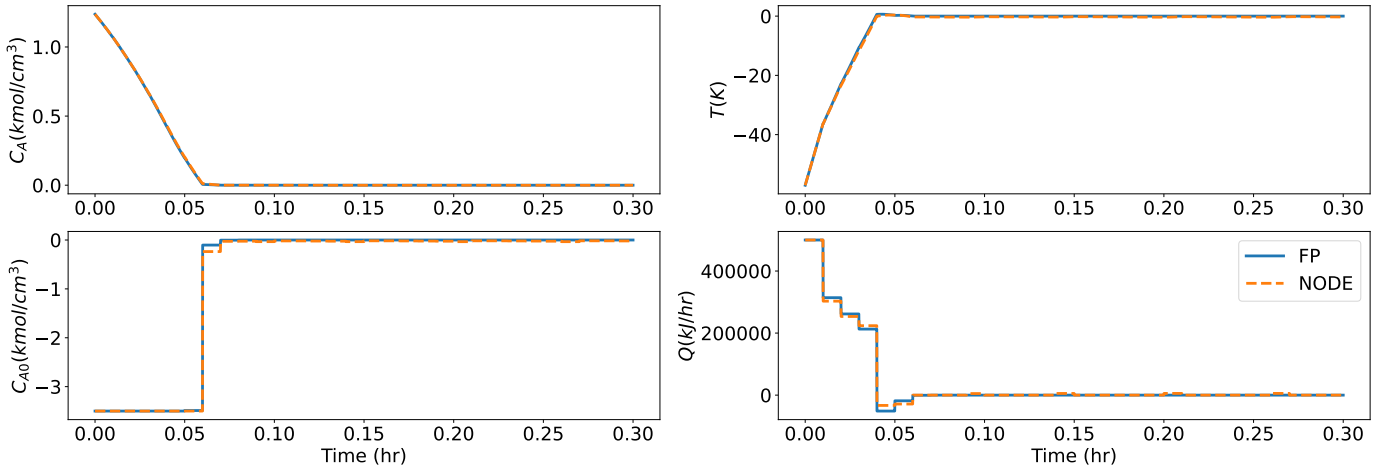


Figure 3.3: State and input profiles for the CSTR under the LMPC using the first-principles process model (blue line) and the NODE process model (orange line).

Remark 19. *Although changing the type of ODE solver in the NODE is not recommended, the parameters used in the ODE solver can be changed based on the model performance without any negative impact. For example, the integration time step in the explicit Euler solver used in this study was tuned when developing the LMPC to balance prediction accuracy with computational cost. Besides this, in the PyTorch framework, additional mathematical operations, such as a lambda*

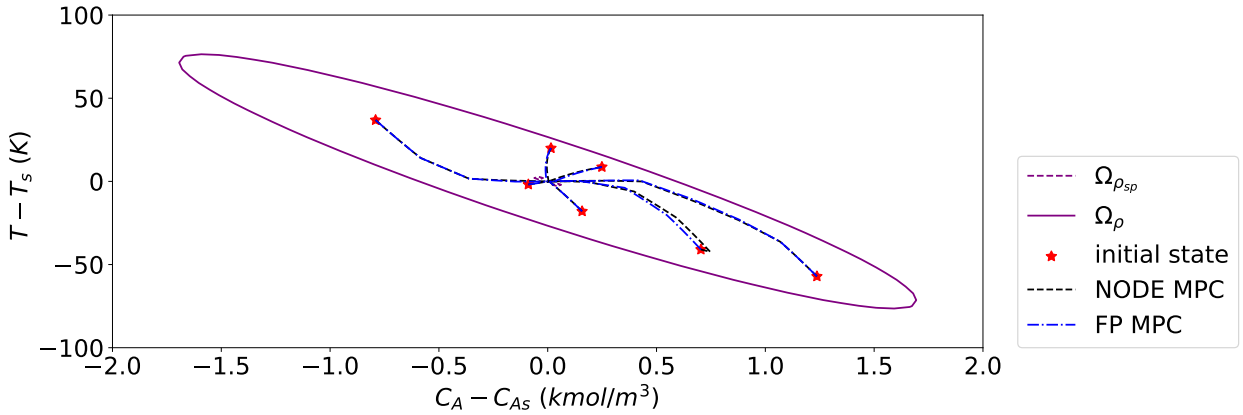


Figure 3.4: NODE-based LMPC performance in closed-loop simulation. The closed-loop simulation used the NODE-based LMPC to stabilize the CSTR from various initial conditions represented by red stars. The closed-loop state trajectories under the LMPC are demonstrated in black dashed line and compared with the state trajectories under an LMPC based on FP equation. The NODE-based LMPC successfully bring the state to the desired region $\Omega_{\rho_{sp}}$ and having a very similar performance comparing to the LMPC designed using FP equation.

layer, can be added to the core model after training. Specifically, a lambda layer can be designed to ensure that the output of the core model is equal to zero when the neural network inputs are all at the steady-state. By adding such a lambda layer after training, the NODE model can be easily programmed to provide correct information for some critical well-known process conditions without harming the training process (i.e., including the lambda layer during the training process may affect the gradient descent step of the neural network model training). In this work, it is found that adding a lambda layer to ensure zero output at the steady state did not have a significant effect on the LMPC performance. Therefore, the simulations in this work are carried out without involving any lambda layer.

3.5.2 Noisy Data Example: Gaussian Noise

After showing the NODE-based LMPC is capable of controlling a noise-free system, we further investigate if a NODE-based LMPC can be used in the case of noisy data, which is more practical in an industrial setting. For the first scenario, we assumed that the process noise follows a Gaussian distribution. The process noise follows the distribution, $v \sim \mathcal{N}(0, \sigma^2)$, and is added to the clean data set reported in Section 3.5.1.1 to generate the noisy data set. Three noisy data sets corrupted by increasing strength of process noise are generated for this study and the details of the noise levels are listed in Table 3.2. Subsequently, each noisy data set is used to train an NODE model having the same structure as the one developed using the clean data set. Table 3.2 also includes the testing loss of the NODE model under each noise level.

The testing loss for the model developed with noisy data is calculated using the reference data also corrupted by the same strength of noise (e.g., if the NODE model is developed using the data set corrupted by the weak level of noise, the testing data used to calculate its loss is also corrupted by the weak level of noise), instead of comparing with the clean data. This is because a clean data set is not available in a practical system with measurement noise. Therefore, the models are compared in the sense of how well they can fit the available data. As a result, the NODE model trained with the weak noise has a slightly higher loss than the one trained with clean data, and the loss increases with increasing noise levels, which demonstrates the negative impact of measurement noise on the model training.

To account for noisy measurements, the subsampling method, following the workflow shown in Fig. 3.5, is adopted in this study. We introduce a tuning parameter for the subsampling method,

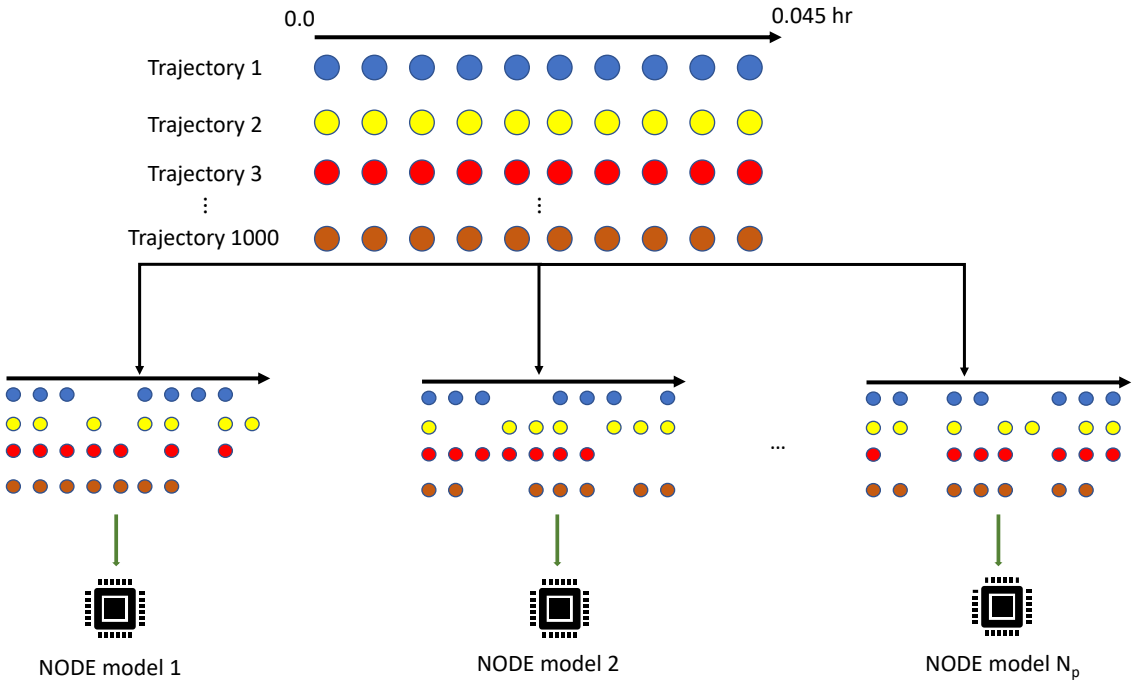
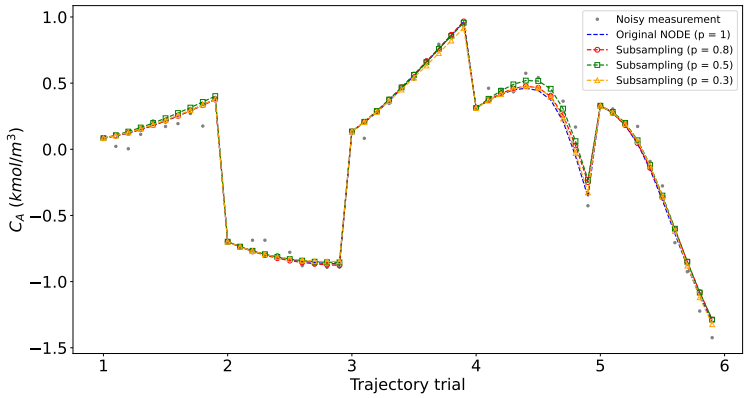
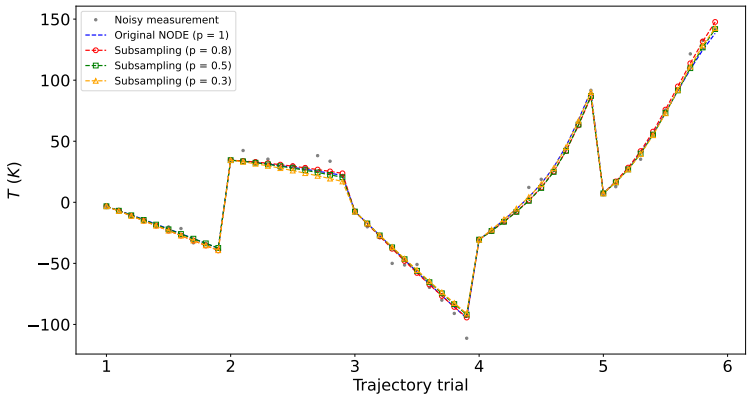


Figure 3.5: The workflow of using subsampling method with a subsampling factor $p = 0.3$ to develop neural ordinary differential equation (NODE) models. By using the subsampling method, different training data can be created in each attempt, which allows for training of a new model. N_p in this figure is used to denotes the number of NODE models to generate in the workflow.

named subsampling factor p, which denotes the percentage of data points randomly selected to be kept in the data set while dropping out the rest of the data. Specifically, the number of trajectories in the training set will remain the same, but some data points in each trajectory will be dropped out randomly to match the desired p value. Additionally, since the data points to be dropped are randomly selected, redoing the subsampling with the same p value will result in a different training data set each time, which leads to training a different NODE model each time. Three p values (i.e., 30%, 50%, 80%) are used to subsample the data, and 5 NODE models are trained for each value of p. Finally, only the lowest training loss among the 5 models is reported in Table 3.2, and the corresponding model is used to develop an LMPC for the next step. The training loss in Table 3.2



(a)



(b)

Figure 3.6: Open-loop simulation results for (a) concentration of reactant A in the CSTR (C_A) and (b) temperature of the CSTR (T) using the NODE model trained with Gaussian noisy data.

shows that using the subsampling method does not have a significant impact on the NODE model performance. Open-loop simulations are further conducted to evaluate the model performance. Fig. 3.6 demonstrates the performance of the NODE models for different values of p by showing the predicted state trajectories, which are overlapping with each other.

Finally, closed-loop simulations are conducted to evaluate the performance of the NODE-based LMPC using different subsampling factors. Specifically, during the closed-loop simulations,

Table 3.2: Training loss of NODE model using Gaussian noisy data.

	Weak Noise	**Medium Noise**	**Strong Noise**
	$\sigma_{C_A} = 0.05\ \text{kmol/m}^3$ $\sigma_T = 5\ \text{K}$	$\sigma_{C_A} = 0.15\ \text{kmol/m}^3$ $\sigma_T = 15\ \text{K}$	$\sigma_{C_A} = 0.25\ \text{kmol/m}^3$ $\sigma_T = 25\ \text{K}$
p	**Mean Squared Error (MSE)**		
1	0.0076	0.0290	0.0365
0.8	0.0074	0.0320	0.0370
0.5	0.0069	0.0270	0.0370
0.3	0.0068	0.0300	0.0500

a non-zero initial state is first randomly drawn from the stability region $\Omega_\rho \backslash \Omega_{\rho_{sp}}$, following which the respective LMPC is used to bring the process to the set point, which is the origin of the state space. Specifically, the closed-loop simulation is run for a duration of 0.3 hr with a sampling time of 0.01 hr. Therefore, there are 30 state feedback measurements used by the LMPC in the simulation, which are all corrupted by noise. To ensure a fair comparison between each LMPC, the noise added to the feedback measurements must be consistent. Thus, the sensor noise is only sampled once from a Gaussian distribution and then saved in order to be used in all the closed-loop simulations. Fig. 3.7 compares the performance of the NODE-based LMPCs developed with four p values in the presence of weak noise (as defined in Table 3.2). It is observed that all the LMPCs successfully stabilized the process from the various initial conditions. However, the clean state may not remain in the designed stability region $\Omega_{\rho_{sp}}$ under the effect of noise. Proposition 4 in [167] derived how the region of ultimate boundary, $\Omega_{\rho_{min}}$, increases as the disturbance bound and sampling period increase. Based on the closed-loop simulations in Fig. 3.7, by using $\Omega_{\rho_{sp}} = 2$, we found the region of ultimate boundary expanded to $\Omega_{\rho_{\min}} = 60$ under weak Gaussian noise.

Finally, the LMPC performance is quantified by calculating the integral of the LMPC cost

function (Eq. 3.32) over the simulation duration, i.e., $\int_{t=0}^{t=0.3hr} L(x(\tau), u(\tau))\, \mathrm{d}\tau$. The quantified loss shows that the subsampling model with $p = 0.8$ has better performance than the NODE model without subsampling in all five closed-loop simulations. In four out of five simulations, the NODE model with $p = 0.8$ gave an LMPC cost function value approximately 1% lower than the model without subsampling, and in the fifth closed-loop simulation, the NODE model with $p = 0.8$ reduced the LMPC cost function by 15% compared to the model with $p = 1$. The NODE models with $p = 0.5$ and $p = 0.3$ did not have a lower LMPC cost function in all five simulations compared to the model without subsampling, but the differences were within 3%. Therefore, the improvement gains from using subsampling is minor in the case of Gaussian noise and possibly even due to numerical/experimental differences between runs.

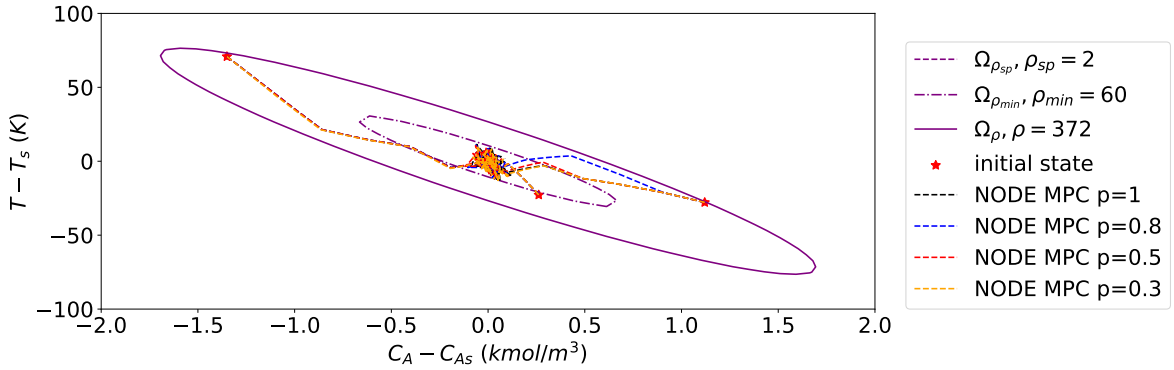


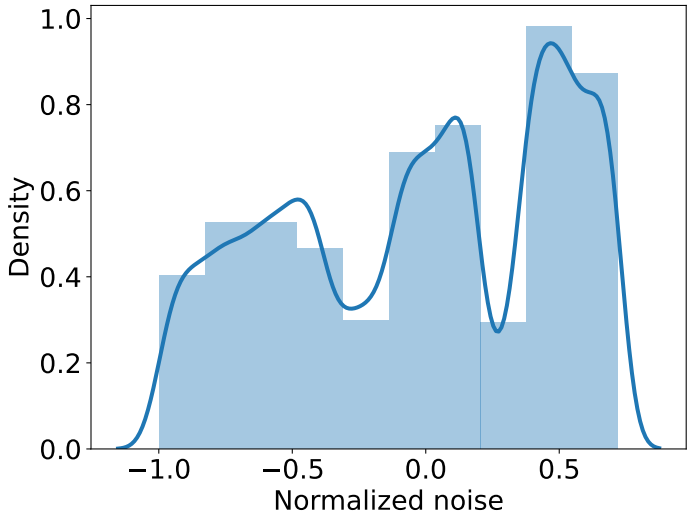
Figure 3.7: Closed-loop simulation results under weak Gaussian noise using LMPC based on NODE model developed with different subsampling factors. Red stars represent the initial condition for of each closed-loop simulation and the black, blue, red, and orange dash line are the state trajectories controlled by LMPC based on the NODE model developed with subsampling factor $p = 1, 0.8, 0.5$ and 0.3, respectively.

3.5.3 Noisy Data Example: Non-Gaussian Noise

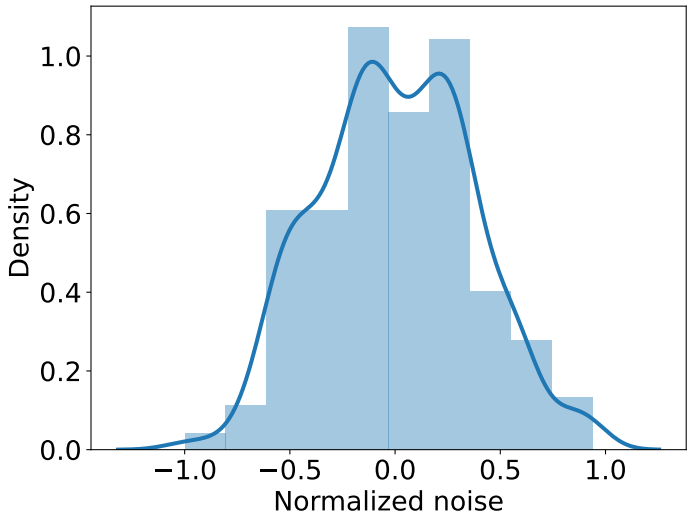
Although the assumption of Gaussian noise is very useful in many applications, non-Gaussian noise is another commonly observed noise distribution in the chemical sector. In this section, we investigate how the NODE model performs under non-Gaussian noise. First, non-Gaussian noise is extracted from an industrial data provided by Aspentech. Fig. 3.8 shows the non-Gaussian noise distribution for C_A and T, respectively. Next, to generate the non-Gaussian noisy reactor data set, the noise value sampled from the above distribution is added to the clean data set following a similar process as described in Section 3.5.2. Different levels of noise are added to the clean data for a comprehensive investigation. The level of noise, **O**, is defined by the maximum value of noise that can be added to the clean data set. Specifically, taking the weak non-Gaussian noise as an example, the maximum noise that can be added to the clean data set is $0.05\ \mathrm{kmol/m^3}$ and 5 K for C_A and T, respectively. To add non-Gaussian noise to a single data point in the clean data set, a noise value from -1.0 to 1.0 is first randomly sampled from the non-Gaussian distribution and then multiplied by the maximum noise value of each variable before being added to the respective clean data point.

Table 3.3: Training loss of NODE model using non-Gaussian noisy data.

	Weak Noise	**Strong Noise**
	$\mathbf{O}_{C_A} = 0.05\ \mathrm{kmol/m^3}, \mathbf{O}_T = 5\ \mathrm{K}$	$\mathbf{O}_{C_A} = 0.10\ \mathrm{kmol/m^3}, \mathbf{O}_T = 5\ \mathrm{K}$
p	**Mean Squared Error (MSE)**	
1	0.0025	0.0031
0.8	0.0024	0.0026
0.5	0.0018	0.0024
0.3	0.0019	0.0023



(a)



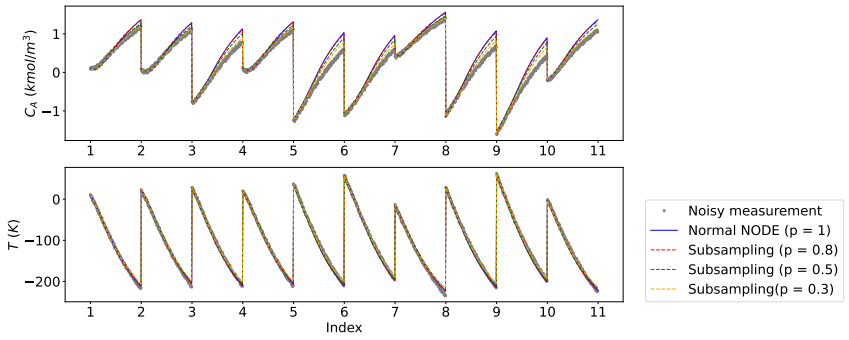
(b)

Figure 3.8: Non-Gaussian noise distribution for (a) concentration of reactant A in the CSTR (C_A) and (b) temperature (T) of the CSTR. The non-Gaussian noise is scaled to sit between -1.0 to 1.0 and is multiplied by the maximum noise parameter depending on the strength of the noise before adding it to the clean data.

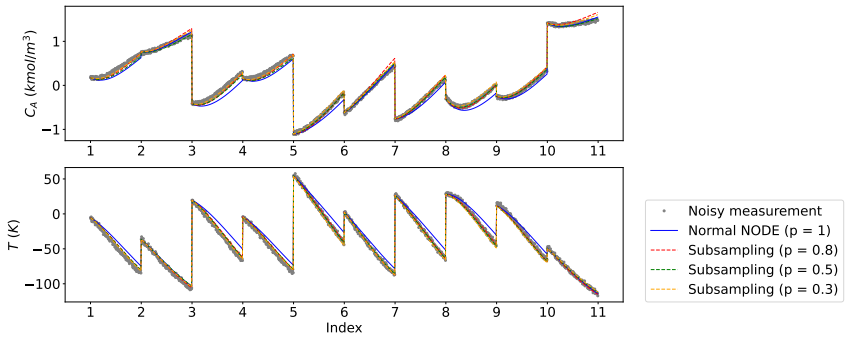
Subsampling with the same range of p values are used to reduce the effect of non-Gaussian noise. The training loss for each value of p and noise level is summarized in Table 3.3. The

maximum noise value of temperature is fixed at 5 K because having a measurement noise of 5 degrees is practically very significant. A sensor that gives a bigger measurement error can be considered to be a dysfunctional sensor and requires maintenance or replacement. On the other hand, the concentration sensor may have more perturbation in its measurement than $0.05\ \mathrm{kmol/m^3}$, so a noise level of $0.1\ \mathrm{kmol/m^3}$ is included in the study as the strong noise level. Based on the training loss values listed in Table 3.3, the subsampling method successfully improves the model performance when the training data set is corrupted with non-Gaussian noise. The best models to fit the noisy data are the models trained with subsampling factors of $p \leq 0.5$, which reduce the training loss by 28% and 26% for weak and strong non-Gaussian noise, respectively, but the loss for models with $p = 0.5$ and $p = 0.3$ are, in fact, very similar and both show improvement over models with larger p values. The open-loop simulations shown in Fig. 3.9 further demonstrates the model improvements by using the subsampling method.

Specifically, for weak non-Gaussian noise (Fig. 3.9a), the NODE models trained with subsampling method predict C_A better compared to the model without subsampling, but there is no significant difference in terms of the temperature prediction, although this may be due to the small room for improvement for the case of the temperature prediction. For stronger noise (Fig. 3.9b), using the subsampling method improves the predictive performance of the model for both states. Please note that we only change the strength of the C_A noise, but since the temperature and concentration of the CSTR are coupled, increasing the noise level for C_A will also affect the model prediction of the temperature. Closed-loop simulations similar to the Gaussian case are used to evaluate the LMPC performance under strong noise, and the results are shown in Fig. 3.10. All the NODE-based LMPCs successfully stabilized the process from the non-zero initial state by ul-



(a)



(b)

Figure 3.9: Open-loop simulation results using NODE model training with (a) weak and (b) strong non-Gaussian noisy data.

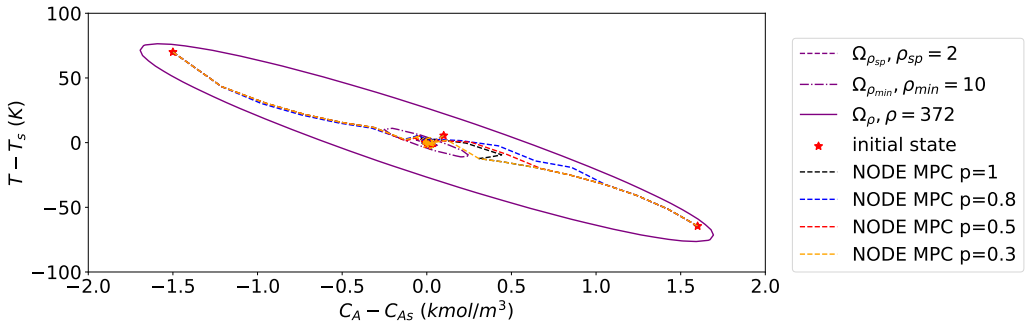


Figure 3.10: Closed-loop simulation results under strong Gaussian noise using LMPC based on NODE model developed with different subsampling factors. Red stars represent the initial condition for of each closed-loop simulation and the black, blue, red, and orange dash line are the state trajectories controlled by LMPC based on the NODE model developed with subsampling factor $p = 1, 0.8, 0.5$ and 0.3 respectively.

timately maintaining the states within $\Omega_{\rho_{\min}} = 10$. By quantifying the loss over the simulation duration, it is found that using a subsampling factor of $p = 0.3$ gives better LMPC performance

compared to the LMPC without subsampling. The largest reduction in the LMPC cost function via subsampling was found to be 34% in closed-loop simulations.

Chapter 4

Machine Learning-Based Operational Modeling of an Electrochemical Reactor: Handling Data Variability and Improving Empirical Models

4.1 Introduction

The electrochemical transformation of carbon dioxide (CO_2) into carbon-based fuels and chemicals has received growing interest in this century because of its potential to reduce CO_2 emissions and facilitate the production of energy from renewable sources [108]. The biggest challenge for research in this area is the difficulty in determining and quantifying the products that result from the reduction of CO_2. Specifically, the CO_2 reduction pathways constitute a complex web of reactions that result in the production of various alkanes, alkenes, and oxygenate species [114].

In addition, recent small-scale experiments on this process show varying levels of experimental uncertainty, due to the minimum measurable limit of the sensors and other inevitable experimental errors. This can introduce a level of uncertainty into the data which can increase the probability of over-fitting.

Although developing mathematical models, such as first-principle models, is a classic and reliable way to describe and predict a physical process, the uncertainty and complexity of most engineering systems make it challenging to implement. To overcome this problem, various data-driven models as well as artificial intelligence (AI) approaches have been proposed historically. Early on, in the 1960s, an epochal AI logic, the Fuzzy Logic, was proposed by Zadeh to approximate uncertain features [183]. From then on, techniques of ML for real-time process operation were investigated in 1990s [156], such as the expert system [90]. Additionally, the auto-regressive model provided statistical strategies to develop data-driven models based on recorded observation. For example, the auto-regressive-moving-average model (ARMA) proposed by Peter Whittle back in the early 1950s [161].

With the development of open-source deep-learning libraries and availability of large datasets from experimental electrochemical reactors (as well as other chemical reactor systems), ML modeling of electrochemical reactors and other reactor systems has become a growing field of interest within chemical engineering. Specifically, various versions of artificial neural network (ANN) models have demonstrated their ability to address regression and classification problems in the context of chemical process modeling [62, 103, 126, 147, 166, 167]. An ANN has many degrees of freedom which gives it the ability to capture the complex, nonlinear relationships between an electrochemical reactor system input and output variables. Additionally, it is a customary approach

to combine an ANN model with an empirical, first-principles model (that is a model that is based on chemical reaction engineering fundamentals yet its parameters are fitted to experimental data) to investigate complex reaction mechanisms and reactor macroscopic input-output behavior [48, 111].

Over the last few years, ANNs have been used to model chemical engineering manufacturing processes in several studies. For example, in [37], a feed-forward neural network (FNN) model was developed to correlate the input and output variables of a SiO_2 atomic layer deposition (ALD) process to calculate optimal half-cycle times to full coverage, which is an important industrial parameter. Additionally, in [70], a methodology was discussed wherein neural networks were used for parameter estimation from experimental data. These research investigations provide a strong support for using neural network models as a reliable approximation to analyze complex nonlinear relationships from simulation/experimental data for electrochemical reactors.

Other works have applied deep learning methods to improve operational aspects of industrial chemical processes. ANNs have been used as process models to replace traditional models to further optimize the control and operation of chemical and industrial processes. In [13], a deep reinforcement learning controller was used to control a hydraulic fracturing process to improve safety and optimization of system operation. In addition, an operational model was constructed for this process using a hybrid approach of a deep neural network and a first-principles model [12]. ANNs were further used to determine optimum operating conditions for chemical and industrial processes [68, 69, 80, 81, 106, 131], which contributed to maximizing the feasibility of novel processes from economic and safety perspectives.

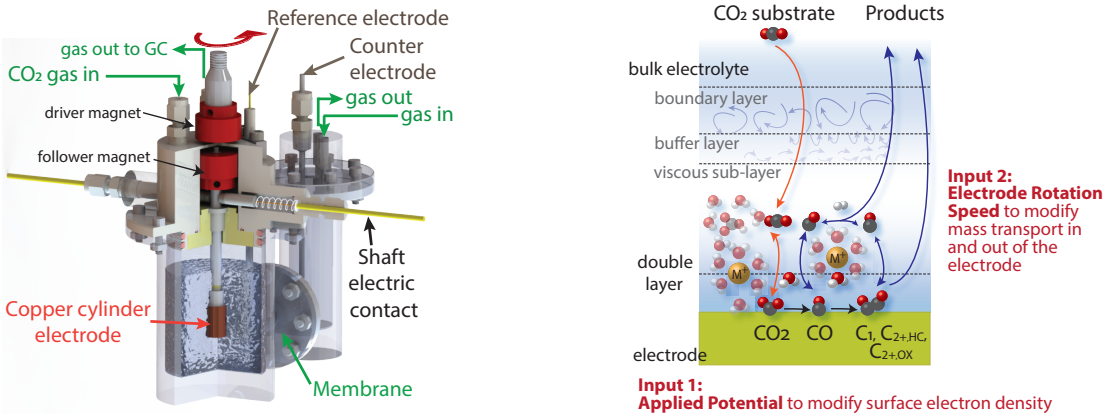
Motivated by the above considerations, this work develops an FNN model using steady-state, input-output experimental electrochemical reactor data by solving a nonlinear regression prob-

lem accounting for data variability. This FNN model is computationally-efficient and can be used in real-time to determine safe and energy-optimal electrochemical reactor operating conditions. Specifically, the maximum likelihood estimation (MLE) concept was adopted to develop an FNN model development algorithm to account for the uncertainty and variability of the experimental data by determining their confidence interval and weighing each point accordingly in the FNN model training process. Therefore, the FNN model is able to account for the data variability and provide the statistically most likely trajectory of the experiment output over a broad set of operating conditions. This probabilistic method decreases the chance that the model will overfit to specific training points with large variation. The key novelty of this work is the development of an operational model for a state-of-the-art electrochemical reactor using a statistical ML method. In addition, the insights obtained from the FNN model are used to propose specific modifications to a classical, empirical first-principles model (EFP model) of electrochemical phenomena to improve its prediction capability, which can contribute to the investigation of the unknown first-principle chemical reactor equations.

The rest of this chapter is organized as follows. In Section 4.2, the experimental reactor setup and the kinetics of the electrochemical reactions are described. In Section 4.3, the formulation and the construction method of the FNN model are discussed. In Section 4.4, the methodology of the maximum likelihood estimation is integrated with the FNN modeling method. In Section 4.5, the performance of the FNN models is evaluated, and the statistical FNN model predictions and insights are used to improve an EFP model for this reactor.

4.2 Preliminaries

This section introduces the background of the experimental electrochemical reactor employed in this work. Specifically, the experimental setup and basic operating reactor mode are presented in this section. Then, an overview of the input-output behavior for this process is used to further explain the data structure used in the neural network model. The experimental reactor and microscopic transport diagrams are shown in Fig. 4.1.



(a) (b)

Figure 4.1: A diagram showing (a) the electrochemical reactor and (b) the multiple and complex reaction and mass transfer processes involved in the transformation of CO_2 to CO and further reduced products on the poly-crystalline copper cylinder electrode.

4.2.1 Experimental Electrochemical Reactor

The reactor was designed to study the effect of mass transport on electrochemical CO_2 reduction while keeping the electrochemical cell hermetically gas-tight for the online detection of gas products. This is allowed by magnetic coupling where the driver magnet outside, connected to the modulated speed rotator (MSR), transmits torque to the follower magnet inside the reactor (Fig. 4.1). The reactor has two chambers separated by an ion-exchange membrane to prevent the

crossover of products. One chamber contains the working electrode, which is the cathode in this case. The other chamber contains the counter electrode (anode). The CO_2 gas is directly bubbled into both the chambers where the electrodes are submerged in 0.2 M potassium bicarbonate buffer electrolyte. The cathode is a rotating cylinder electrode (RCE) made of polycrystalline copper. Copper is the only known single transition metal that can reduce CO_2 into hydrocarbons and oxygenates with more than two carbons (C_{2+}) at an appreciable rate, and it plays a critical role as the catalyst in the overall reaction scheme [113]. As the RCE shaft continuously stirs the electrolyte solution, hydrodynamics formed around the electrode can be systematically controlled by setting a rotation speed from the MSR. Finally, gas and liquid products are analyzed by gas chromatograph (GC) and nuclear magnetic resonance (NMR) spectroscopy, respectively, to determine the product composition under well-controlled mass transport characteristics. Further details on the reactor design and experimental setup are newly reported [71].

The product compositions quantified using GC and NMR are then used to determine the production rate of each species and the reaction selectivity with respect to desired products. Polycrystalline copper produces various products as tabulated in Table 4.1 at a quantifiable level. Here, the competing hydrogen evolution reaction and the production of formate are excluded from the table and from the selectivity calculation, since they do not share the same reaction pathway as the products in Table 4.1. That is, although carbon monoxide and formate are 2-electron reduction products, carbon monoxide is the main reaction intermediate towards further reduced products while formate can not be further reduced. Among the products sharing carbon monoxide as common intermediate, the desired products are the C_{2+} oxygenate species (labeled in Table 4.1) as they are of high value and are commonly used as liquid fuels and reagents. Therefore, the selectivity

Table 4.1: Electrochemical reactions to reduce CO_2 to various products on copper.

Index	Reaction	Classification
1	$CO_2 + 6H_2O + 8e^- \longrightarrow CH_4 + 8OH^-$	C_1 hydrocarbon (HC)
2	$2CO_2 + 8H_2O + 12e^- \longrightarrow C_2H_4 + 12OH^-$	C_{2+} hydrocarbon (HC)
3	$CO_2 + 5H_2O + 6e^- \longrightarrow CH_3OH + 6OH^-$	C_1 oxygenate (OX)
4	$2CO_2 + 9H_2O + 12e^- \longrightarrow C_2H_5OH + 12OH^-$	C_{2+} oxygenate (OX)
5	$2CO_2 + 5H_2O + 8e^- \longrightarrow CH_3COO^- + 7OH^-$	C_{2+} oxygenate (OX)
6	$2CO_2 + 8H_2O + 10e^- \longrightarrow (CH_2OH)_2 + 10OH^-$	C_{2+} oxygenate (OX)
7	$2CO_2 + 6H_2O + 8e^- \longrightarrow HOCH_2CHO + 8OH^-$	C_{2+} oxygenate (OX)
8	$2CO_2 + 7H_2O + 10e^- \longrightarrow CH_3CHO + 10OH^-$	C_{2+} oxygenate (OX)
9	$3CO_2 + 13H_2O + 18e^- \longrightarrow C_3H_7OH + 18OH^-$	C_{2+} oxygenate (OX)
10	$3CO_2 + 11H_2O + 16e^- \longrightarrow C_3H_5OH + 16OH^-$	C_{2+} oxygenate (OX)
11	$3CO_2 + 11H_2O + 16e^- \longrightarrow CH_3COCH_3 + 16OH^-$	C_{2+} oxygenate (OX)
12	$3CO_2 + 11H_2O + 16e^- \longrightarrow C_2H_5CHO + 16OH^-$	C_{2+} oxygenate (OX)
13	$CO_2 + H_2O + 2e^- \longrightarrow CO + 2OH^-$	

for this experiment is defined as the ratio of the rate of C_{2+} oxygenate production to the rate of hydrocarbon production.

With respect to the reactor mode of operation, the CO_2 gas dissolves into the buffer solution, and is carried to the electrode surface by convective mass transport caused by the rotating electrode. Subsequently, the CO_2 molecules are adsorbed onto the electrode surface and reduced to oxygenate and hydrocarbon products through consecutive proton-coupled electron injection steps. Therefore, the surface reaction rate is determined by the electron density on the Cu surface and the adsorption rate of CO_2 molecules to the Cu surface. The electron density is dictated by the applied potential and the adsorption rate of CO_2 is the result of complex mass transport and electrode kinetics at the electrode/electrolyte interface (Fig. 4.1).

4.3 Development of Machine Learning Model

In this section, a neural network model is constructed to capture the steady-state behavior of the reactor at varying applied potentials and electrode rotation speeds using experimental electrochemical reactor input-output data. The neural network model formulation, training process and the data collection process are presented in the following subsections.

4.3.1 FNN Learning Algorithm

The general structure of an FNN model is shown in Fig. 4.2 and can be mathematically represented by the following equations:

$$\mathbf{Y} = F_{NN}(X) = \begin{cases} h_j^{[1]} & = \sigma^{[1]}(\sum_{i=1}^{p} \omega_{ji}^{[1]} x_i + b^{[1]}) \\ h_j^{[2]} & = \sigma^{[2]}(\sum_{i=1}^{p} \omega_{ji}^{[2]} h_i^{[1]} + b^{[2]}) \\ y_j & = \sigma^{[l]}(\sum_{i=1}^{p} \omega_{ji}^{[l]} h_i^{[l]} + b^{[l]}) \end{cases} \tag{4.1}$$

where $\mathbf{X} = [x_1, \ldots, x_n] \in \mathbf{R}^n$ and $\mathbf{Y} = [\hat{y}_1, \ldots, \hat{y}_m] \in \mathbf{R}^m$ are the input and output vectors of the FNN model, respectively. $\omega_{ji}^{[k]}$, $i = 1, \ldots, p$, $j = 1, \ldots, p$, and $k = 1, \ldots, l$, stand for the weights connecting the ith input from the prior layer to the jth neuron in the kth layer, where l is the number of layers. p represents the number of neurons used in each layer. Therefore, $i = 1, \ldots, n$ for the first hidden layer, because there are n units in the input layer. $b^{[k]}$ and $\sigma^{[k]}(\cdot)$ denote the bias and activation function used in the kth layer.

In this study, a centralized two-input-multi-output FNN model is constructed to capture the nonlinear relationship between the two input states in Table 4.2 (i.e., rotation speed and applied

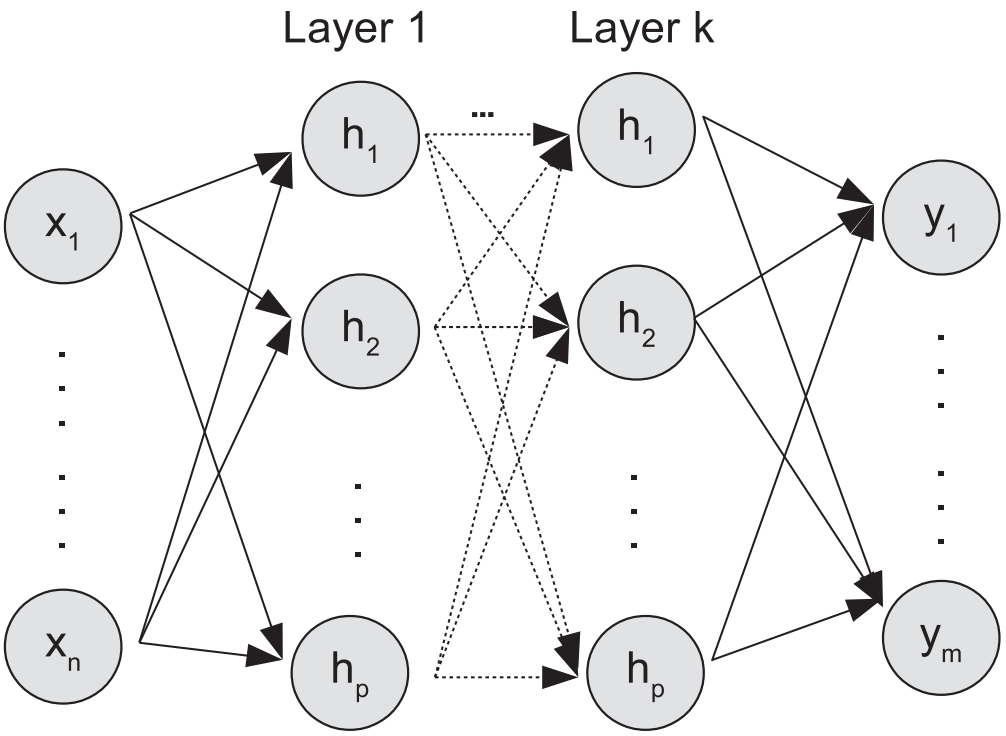


Figure 4.2: General structure of an FNN model, where subscript p is the index of neurons in the kth hidden layer.

potential) and the fourteen outputs listed in Table 4.3. Specifically, the input and output training data are scaled by the maximum value of each respective state such that all the normalized states fall in the range between 0 and 1. The input layer is densely connected to 64 neurons in the hidden layer using the Rectified Linear Unit (ReLu) activation function, as defined in Eq. 4.2. The hidden layer is densely connected to the output layer using the Softplus activation function, $S(x) = log(1+e^x)$. Both the ReLu and Softplus functions are used to restrict the output predictions to be strictly non-negative, and introduce nonlinearity to the model.

$$ReLu(z) = \begin{cases} z & \text{for} \quad z > 0 \\ 0 & \text{for} \quad z \leq 0 \end{cases} \tag{4.2}$$

Remark 20. *A single hidden layer is used for this model because it is the simplest structure to sufficiently capture the data trends. Additionally, we apply a grid search for the number of neurons in the FNN, with 64 neurons having the best prediction. Specifically, neural networks with less than 64 neurons underfit the data, while networks with more neurons would overfit the data. In this work, both the prediction accuracy (in terms of mean-squared-error) and the output trajectories are considered to design the hyperparameters of the FNN model. Classical hyperparameter tuning algorithms did not perform effectively to capture reasonable trajectories due to the difficulty of developing an explicit formula to evaluate the prediction trends. However, hyperparameter tuning algorithms, such as Bayesian optimization and random forest methods, are powerful tools to optimize the neural network structure [165]. We recommend that other users consider using those methods to develop their ML models.*

4.3.2 Data Generation and Dataset

As listed in Table 4.3, the oxygenates considered are Outputs 3-12, and the hydrocarbons are Outputs 1 and 2 (methane and ethylene). Therefore, the selectivity defined in the previous section is calculated as follows:

$$Selectivity := \frac{\sum_{i=3}^{12} y_i}{\sum_{i=1}^{2} y_i} \tag{4.3}$$

where y_i refers to the production rate of species i, as defined in Table 4.3. Data are collected for the range of potential and rotation speed within which the reactor will operate. Specifically, the potential is varied from -1.2 to -1.47 (V vs. the standard hydrogen electrode (SHE)), and the rotation speed is varied from 100 revolutions per minute (rpm) to 800 rpm. For the data collection process, the potentiostat is set to be a constant potential, and the electrode is rotated at a constant angular speed. The reactor is allowed to operate at steady-state for twenty minutes prior to the data collection. Then, the reactor operates continuously with product samples taken every twenty minutes to determine the concentration of the thirteen relevant products, followed by the calculation of selectivity from the results of each sample.

The sampling process is an 80-minute experiment which constitutes one data point for each input and output states. The sampling is repeated three to four times to ensure the data are consistent over time and to obtain the statistical information for the experimental results under the same operating condition. Specifically, this 80-minute experiment is repeated over 100 times to generate the data library that cover the specified range of operating conditions. Subsequently, the data are grouped into a single data vector based on the similarity of the operating conditions to compute

the mean and standard deviations. As a result, 21 data points with mean and standard deviation information are collected from 2 or 5 independent experiments, depending on the availability of experimental data.

4.3.3 Design of the Experiment

The range of potentials is limited by the overall resistance of the electrochemical cell (between the working and the counter electrode). This issue is resolved in the second generation of the cell reported in the work [71] by removing the channel that connects the two chambers to shorten the distance between the two electrodes and increase the surface area of the ion-exchange membranes. However, in this work, the first generation of the reactor is used which is not able to apply potentials more negative than -1.47 V vs SHE. The potential range is chosen to see appreciable rates of product generation considering the detection limits of the sensors (GC and NMR). The maximum rotation speed possible is 2000 rpm, as provided by the vendor of the RCE (Pine Research Instrumentation).

On the other hand, we have restricted the maximum electrode rotation speed to 800 rpm mainly due to the mechanical instability of the custom-machined parts of the electrochemical cell. Additionally, the chosen range of rotation speed is appropriate for studying mass transport effects from the perspective of mass transport characteristics around the RCE. As shown in the work [71], the film mass-transfer coefficient decreases as the electrode rotation speed increases with a 0.59 order dependency. The further increase in the rotation of the electrode beyond 800 rpm has a minimum effect on the mass transport properties of the cell. The lower-bound of the rotation speed range is 100 rpm, below which the relationship between the film mass-transfer coefficient and the

rotation speed starts to flatten out due to the convection created by the bubbling of CO_2 in the bulk of the electrochemical cell.

4.3.4 Standard FNN Training

The mean-squared-error (MSE) is used in the standard FNN training as the loss function that minimizes the difference between the experimental data value and the model predictive value. The MSE loss function is given below:

$$Loss = \frac{1}{d}\frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m}\left|y_{i,j} - \hat{y}_{i,j}\right|^2 \tag{4.4}$$

where d and m are the number of data points in the training dataset and the number of output states. Specifically, from the original 21 data points, 4 are reserved for testing and the remaining 17 are used for training. Then, the 17 points are randomly split into training and validation sets with 80% used for training and 20% used for validation. The testing procedure compares the mean-squared difference between the FNN prediction and the testing data, using the loss function of Eq. 4.4 to evaluate the model performance. During this process, the parameter vector $\mathbf{W}$, which contains all the weights and bias of the neural network, is optimized using Eq. 4.5 to minimize the loss function.

$$\mathbf{W} = \mathbf{W} - \eta\frac{V_{dw}}{\sqrt{S_{dw}} + \epsilon} \tag{4.5}$$

where η is the learning rate, ϵ is a small positive number to prevent the denominator being zero. V_{dw} and S_{dw} introduce the momentum and root-mean-square factors of the parameters gradient to facilitate the optimization process. In practice, ϵ, V_{dw}, and S_{dw} can be set up by the ML API (e.g.

Keras) automatically by specifying the optimizer. Tuning the value of ϵ will not have significant impact to the model performance. Additionally, user can tune the learning rate η to improve the model performance. Usually, it is a small positive real value in the range between 0.0 and 1.0.

Table 4.2: Input states of the FNN model.

Index	**Input State**	**Units**
1	Applied Potential	V vs. the standard hydrogen electrode (V vs. SHE)
2	Rotation Speed	rpm

Table 4.3: Output states of the FNN model.

Index	**Output State**	**Chemical Formula**
1	methane production rate	CH_4
2	ethylene production rate	C_2H_4
3	methanol production rate	CH_3OH
4	ethanol production rate	C_2H_5OH
5	acetate production rate	CH_3COO^-
6	ethylene glycol production rate	$(CH_2OH)_2$
7	glycolaldehyde production rate	$HOCH_2CHO$
8	acetaldehyde production rate	CH_3CHO
9	n-propanol production rate	C_3H_7OH
10	allyl alcohol production rate	C_3H_5OH
11	acetone production rate	CH_3COCH_3
12	propionaldehyde production rate	C_2H_5CHO
13	carbon monoxide production rate	CO
14	selectivity	

4.4 Maximum Likelihood Estimation in Machine Learning Reactor Modeling

Despite the standard FNN's capability of correlating the input and output variables of a complex nonlinear process, it treats all the data points equally, which might lead to overfitting when the data contains inconsistent levels of random error from the experimental data. To address this issue, the MLE method, originally developed by R.A. Fisher in the 1920s, is adopted to train the FNN model to optimize the parameter set that maximizes the likelihood function of a probabilistic model [110]. Specifically, the likelihood function, $\mathcal{L}(\cdot)$, is used to correlate an unknown parameter vector (θ) with a random variable set (z) based on its probability-density function, $f(z, \theta)$. The maximum likelihood method can search for an optimum parameter set θ^* by maximizing the "likelihood of the sample", $\prod_{i=1}^{n} f(z, \theta)$, and it has been proven that this method can provide a solution to this optimization problem [67]. The MLE method assumes that the data are from a single population with the same standard deviation. However, this section proposes a modification that assumes each set of input parameters corresponds to a different population. Thus, each data point with its collected standard deviation is treated as an independent random variable.

To apply this method in our study, we first consider the experimental dataset to be a pseudo-probabilistic sample following the Gaussian distribution with an associated standard deviation. Therefore, the FNN outputs $\hat{y}_{i,j}$ need to follow the same distribution as the reference data $y_{i,j}$, which means the joint likelihood of the neural network output is of Gaussian distribution, and can

be expressed as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{X};\mathbf{W},\sigma) &= \prod_{k=1}^{d\times m} f_{\mathbf{Y}}(y_k) \\ &= \prod_{k=1}^{d\times m}(2\pi\sigma_k^2)^{-0.5} \times \mathbf{exp}\left[-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{m}\left|\frac{y_{i,j}-\hat{y}_{i,j}(\mathbf{X},\mathbf{W})}{\sigma_{i,j}}\right|^2\right]\end{aligned} \tag{4.6}$$

where σ_i is the standard deviation for each data point. Subsequently, we find the optimum weight matrix $\mathbf{W}^*$ by maximizing the logarithm of the joint likelihood function:

$$\begin{aligned}\mathbf{W}^* :&= \arg\max_{\mathbf{W}} \mathbf{log}\mathcal{L}(\mathbf{X};\mathbf{W},\sigma) \\ &= \arg\max_{\mathbf{W}}\left(-\frac{1}{2}\sum_{k=1}^{d\times m}\mathbf{log}(2\pi\sigma_k^2) - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{m}\left|\frac{y_{i,j}-\hat{y}_{i,j}(\mathbf{X},\mathbf{W})}{\sigma_{i,j}}\right|^2\right) \\ &= \arg\max_{\mathbf{W}}\left(-\sum_{k=1}^{d\times m}\mathbf{log}(2\pi\sigma_k^2) - \sum_{i=1}^{d}\sum_{j=1}^{m}\left|\frac{y_{i,j}-\hat{y}_{i,j}(\mathbf{X},\mathbf{W})}{\sigma_{i,j}}\right|^2\right)\end{aligned} \tag{4.7}$$

Since the first term of Eq. 4.7 is independent of $\mathbf{W}$, the maximum likelihood estimation of this model can be further simplified into Eq. 4.8.

$$\mathbf{W}^* = \arg\min_{\mathbf{W}}\left(\sum_{i=1}^{d}\sum_{j=1}^{m}\left|\frac{y_{i,j}-\hat{y}_{i,j}(\mathbf{X},\mathbf{W})}{\sigma_{i,j}}\right|^2\right) \tag{4.8}$$

The maximum likelihood estimation weighted FNN model (weighted-FNN) is constructed using the same architecture and dataset as the standard FNN. However, the weighted-FNN model considers the standard deviation of each data point in its training process. Specifically, the sample standard deviation is calculated for each data point. Then, the coefficient of variance (v) of each data point is determined by the ratio of standard deviation and the respective output mean. This

normalizes the data variability to allow for unbiased comparison between quantities of different magnitudes. The loss function, shown as Eq. 4.9, integrates Eq. 4.4 and Eq. 4.8. Thus, the weight matrix of the weighted-FNN is optimized to maximize both the accuracy of the prediction and the likelihood function during the training process.

$$Loss = \frac{1}{d}\frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m}\frac{1}{v_{i,j}^2}\left|y_{i,j} - \hat{y}_{i,j}\right|^2 \tag{4.9}$$

Remark 21. *In this study, error bars are constructed to represent the region of one standard deviation of uncertainty with respect to the mean, which is approximately 70% confidence interval for Gaussian distributed variables. Any statistic model can be used to develop a weighted-FNN model if it can provide reasonable statistical information of the experimental observations.*

Remark 22. *As shown in [40, 87], the simplified log-likelihood function (Eq. 4.8) can be used directly as the loss function of a weighted-FNN model, since it contains the sum of squared error (SSE) in the loss function. We integrate it with Eq. 4.4 to demonstrate its similarity to the mean-squared error (MSE) loss function.*

Remark 23. *Bayesian optimization is another acknowledged method to develop statistical ML model. Similar to the MLE method, the Bayesian optimization also considers the likelihood function model, which can account for data variance. Instead of focusing on the likelihood function, the Bayesian method implements optimization based on the posterior distribution of the ML model, which is defined by the Bayes' rule [145]. Therefore, the prior distribution of the parameter vector ($p(\theta)$) and the marginal likelihood of the observed data ($p(D)$) can be adopted to develop the statistical model.*

4.5 Machine Learning Model Results and Analysis

In this section, we first compare the prediction performance of the standard FNN and weighted-FNN models. Subsequently, the weighted-FNN model's ability to capture the physical phenomenon behind the experiment is demonstrated through a comparison with a classical, EFP model. Additionally, we propose an algorithm to improve the empirical model performance using the neural network model results and insights. Parameters used to generate the EFP models are described in this section and listed in Table 4.4.

4.5.1 FNN vs. weighted-FNN

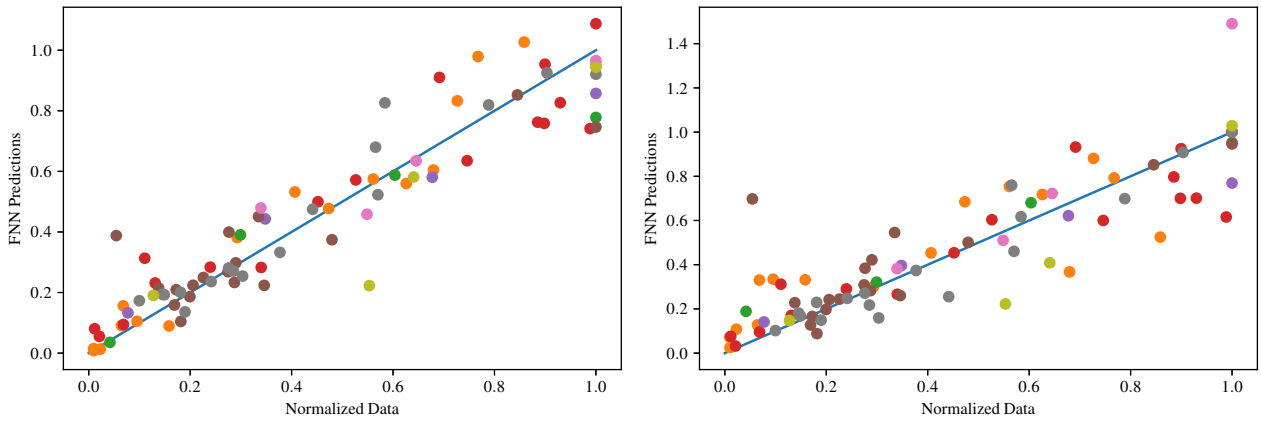
We first compare the performance of the weighted-FNN against the standard FNN. To account for the stochastic nature during the neural network training process, a Python script is used to train 100 FNN models in parallel with the structure discussed in Section 4.3 and with randomly partitioned training and validation sets. The best FNN and weighted-FNN are chosen to minimize the MSE for the training dataset. This training method ensures the selected models are trained consistently following the same criteria. Then, the selected FNN and weighted-FNN models are evaluated with respect to the testing dataset, using the MSE between the normalized FNN outputs and the normalized testing set. The MSEs for the standard FNN and weighted-FNN are 0.0751 and 0.0791, respectively, which demonstrates a slight better performance of the standard FNN. It is shown in Fig. 4.3 that both models give accurate predictions across the majority of the data points, but the overall MSE for the weighted-FNN prediction increases, since it ignores the data points with high variance. However, the MSE of the two methods are sufficiently small, which implies

Table 4.4: Process parameters for EFP models with units.

Quantity	**Value**	**Units**
EFP model (limiting conditions)		
k_0	2.32×10^{-12}	$cm \cdot s^{-1}$
α	0.5	
F	96485	$C \cdot mol^{-1}$
R	8.314	$J \cdot mol^{-1} \cdot K^{-1}$
T	298	K
$E^{0\prime}$	-0.52	V
C_{CO_2}	3.40×10^{-5}	$mol \cdot cm^{-3}$
D_{CO_2}	1.91×10^{-5}	$cm^2 \cdot s^{-1}$
d_{RCE}	1.2	cm
ν_{H_2O}	1.03×10^{-2}	$cm^2 \cdot s^{-1}$
EFP model		
$k_{0,5}$	2.02×10^{-28}	$mol \cdot cm^{-1} \cdot s^{-2}$
$k_{0,6}$	7.47×10^{-32}	$mol \cdot cm^{-1} \cdot s^{-2}$
$k_{0,7}$	2.61×10^{-13}	$mol^{-\frac{1}{2}} \cdot cm \cdot s^{-\frac{1}{2}}$
α_5	0.7	
α_6	0.85	
α_7	0.665	
EFP model (updated)		
$k_{0,5}$	7.2×10^{-22}	$mol \cdot cm^{-1} s^{-2}$
$k_{0,6}$	1.6×10^{-17}	$mol^{\frac{1}{4}} \cdot cm^{\frac{3}{2}} \cdot s^{\frac{5}{4}}$
$k_{0,7}$	9.5×10^{-23}	$mol \cdot cm^{-1} \cdot s^{-2}$
α_5	0.42	
α_6	0.53	
α_7	0.49	

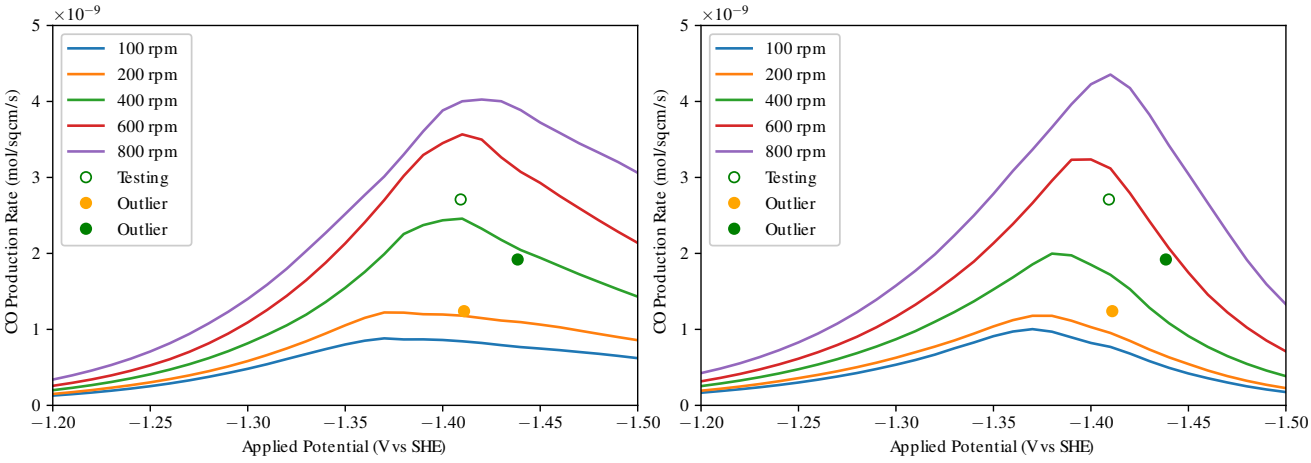
that both models capture the input-output relationship well.

To further compare the performance of the two models, the predictions for CO production rate are compared to some labeled outlier points due to a slight drift in operating conditions in Fig. 4.4. As shown in the figure, the weighted-FNN weighs the data points with critical experimental uncertainty less while the standard FNN overfits these points. This demonstrates the ability of the



(a) (b)

Figure 4.3: Comparison between the observed experimental outcome and the neural network predictions from (a) standard FNN and (b) weighted-FNN models.



(a) (b)

Figure 4.4: The CO production rate predictions for various applied potentials in the unit of V vs. the standard hydrogen electrode (V vs. SHE). The solid points are labeled uncertain data as having a drift in potential. The open point is from the testing set. (a) The CO prediction of standard FNN model overfitted the labeled uncertain data points. (b) The weighted-FNN model successfully learned the experimental uncertainty and provide prediction accordingly, but this feature introduce extra error to the testing results.

weighted-FNN model to improve its prediction by accounting for data variance. The goal of MLE method is to generate models with a higher statistical significance that are suitable to be implemented with an experimental dataset. The weighted-FNN model demonstrates that it can provide

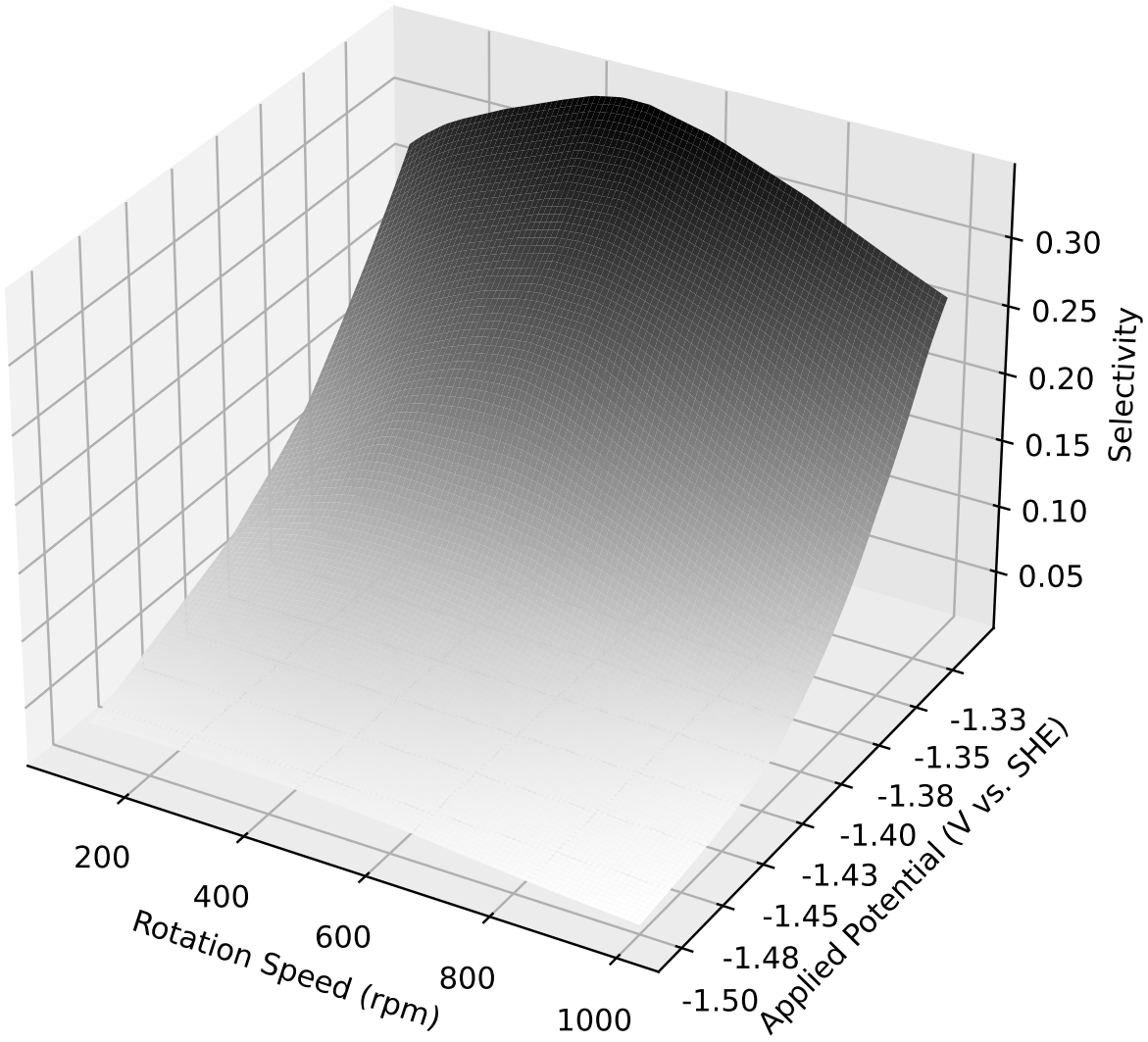


Figure 4.5: Selectivity of oxygenate species with respect to rotation speed and applied potential (in the unit of V vs. the standard hydrogen electrode (V vs. SHE)) as predicted by the ML model.

an accurate approximation of the experimental data while outperforming the standard FNN in its ability to mitigate the impact of experimental uncertainty. Therefore, to simplify the discussion, the weighted-FNN will be used in the remainder of this section to provide comprehensive selectivity predictions for the electrochemical reactor, which is shown in Fig. 4.5, and for comparison with other models (i.e., henceforth the FNN will only refer to the weighted-FNN, and the standard FNN will not be included).

Remark 24. *The outlier points are included in the dataset, since they are valid but have higher variability. The predictive models are developed based on the experimental observation even if some points are less likely to be reproduced. On the other hand, invalid data points from a failed experiment should not be included in the dataset.*

4.5.2 EFP model vs. weighted-FNN

First-principles models (FP models) are a fundamental approach to describing the operation of an electrochemical reactor according to the energy and mass balances as well as reaction kinetics. However, due to the complex mass transfer and reaction mechanisms of this process, it is challenging to obtain an accurate first-principles model. As a substitute, ML modeling provides an alternative approach to representing the physio-chemical phenomena in the reactor with a desired prediction accuracy. In this subsection, an EFP model of a rotating electrode reactor is developed following the derivation in [14] to determine the rate of CO production under limiting conditions. Specifically, this model assumes that only a single, first-order reaction is occurring with no side reactions, and the reaction occurs only on the electrode surface following Butler-Volmer kinetics, which means it cannot capture the comprehensive kinetics of this experiment. The resulting

equation is given as follows:

$$r_{CO} = \frac{k_f C^*_{CO_2}}{1 + k_f/m_0} \tag{4.10}$$

where $C^*_{CO_2}$ is the bulk concentration of CO_2, k_f is the kinetic rate constant, and m_0 is the convective mass transfer coefficient. The rate constant k_f changes based on the applied potential, and is calculated as follows:

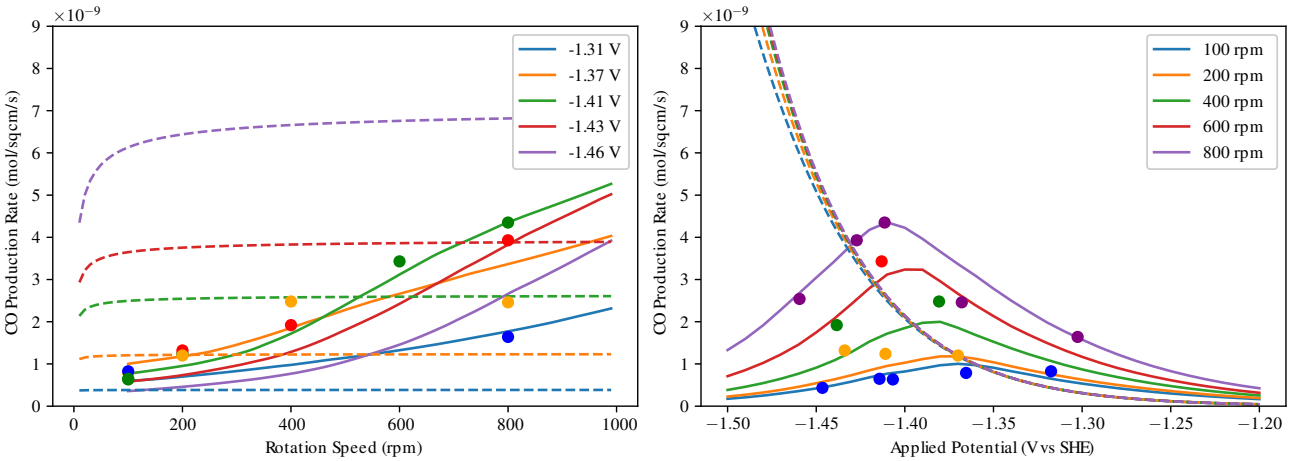
$$k_f = k^0 \exp\left[-\frac{\alpha F}{RT}\left(E - E^{0\prime}\right)\right] \tag{4.11}$$

where k^0 is the standard rate constant, α is the symmetry factor, F is Faraday's constant, R is the gas constant, T is the temperature, E is the applied potential, and $E^{0\prime}$ is the standard reduction potential. The mass transfer coefficient m_0 is determined based on the rotation speed of the electrode, but this correlation will change depending on the type of rotating electrode. For some simple rotating electrode geometries such as a flat disk, a mass transfer coefficient is determined analytically, assuming a linear velocity profile in the boundary layer. However, the electrode used in this experiment has a cylindrical geometry which is more complicated, so the mass transfer coefficient is determined experimentally from the Sherwood number correlation as follows [71]:

$$m_0 = 0.204 Re^{0.59}_{RCE} Sc^{0.33} \frac{D_{CO_2}}{d_{RCE}} \tag{4.12}$$

where Re_{RCE} is the Reynold's number, Sc is the Schmidt number, D is the diffusion coefficient, and d_{RCE} is the diameter of the RCE. The diffusion coefficient is assumed to be the same for the reactant and product species for simplicity. Since the Sherwood number is determined experimentally, this model will be referred to as an EFP model.

The comparison between this EFP model and the FNN prediction is shown in Fig. 4.6. As shown, the EFP model trajectory is similar to the FNN prediction under the operating conditions with less negative applied potentials and lower rotation speeds because the side reactions are limited at these conditions. After switching to more negative conditions, the EFP model's assumption becomes invalid. Therefore, these two models present different predictions after passing threshold conditions. This comparison demonstrates that the neural network can correctly capture the input-output relations from the experimental data.



(a) (b)

Figure 4.6: The CO production rates for the first EFP model (dashed) and the weighted-FNN model predictions (solid) compared with the training data points over the range of (a) rotation speed, and (b) applied potentials in the unit of V vs. the standard hydrogen electrode (V vs. SHE). This EFP model can capture the general trend of the reactor for low applied potential and rotation speed. However, for the more negative potential and higher rotation speed, the initial assumption of the EFP model becomes invalid.

4.5.3 EFP Model Improvement

EFP modeling is an efficient way to find out how a number of experimental variables affect the experimental data without requiring complete knowledge of the underlying physical phenomena

needed for large-scale FP models [19]. Although the neural network has demonstrated its ability to capture steady-state behavior of the electrochemical reactor, an empirical model with an explicit form is essential to improve the reactor phenomena understanding. However, parameter tuning and selection for an empirical model are challenging. Therefore, we propose an algorithm to use neural network model results to improve the EFP model structure.

Specifically, we first develop an EFP model consisting of several regression problems that predict the production rate of seven different classes of species produced in the reactor. This model is derived utilizing the same reaction kinetics and transport phenomena considerations mentioned in Section "EFP model vs. weighted-FNN". As shown in Eq. 4.13, the empirical regressions of interest are the production rate of C_1 products (FNN Outputs 1 and 3 listed in Table 4.3), C_{2+} hydrocarbons (FNN Output 2), and C_{2+} oxygenates (FNN Outputs 4, 5, 6, 7, 8, 9, 10, 11, and 12), denoted as r_{C1}, $r_{C2+,HC}$, and $r_{C2+,OX}$ respectively.

$$\begin{aligned} r_{C_1} &= k_{0,5} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{-1} \exp\left(-\frac{\alpha_5 z_5 F}{RT} E\right) \\ r_{C_{2+,HC}} &= k_{0,6} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{-1} \exp\left(-\frac{\alpha_6 z_6 F}{RT} E\right) \\ r_{C_{2+,OX}} &= k_{0,7} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{0.5} \exp\left(-\frac{\alpha_7 z_7 F}{RT} E\right) \end{aligned} \tag{4.13}$$

The notation C_{CO} is the concentration of carbon monoxide, and J_{HCO_3} is the flux of bicarbonate, both of which are calculated at the inner Helmholtz plane based on the bulk concentration, rotation speed, and applied potential. Additionally, Sh_{RCE} is the Sherwood number of the rotating-cylinder electrode, which relates directly to the rotation speed. The rate constants $k_{0,i}$ and symmetry factors α_i are obtained by linearizing the equations for r_{C1}, $r_{C2+,HC}$, and $r_{C2+,OX}$ with respect to the applied potential. Furthermore, since these rate expressions each describe multiple products from

different reaction steps, the number of electrons z_i is not a single value, and a modification is necessary for this case. Specifically the parameter α_i is considered as a fitting parameter for the exponential relationship between the potential and the rates. Thus, by fixing z_i to 1, α_i becomes an arbitrary positive value that can be optimized in the regression problem.

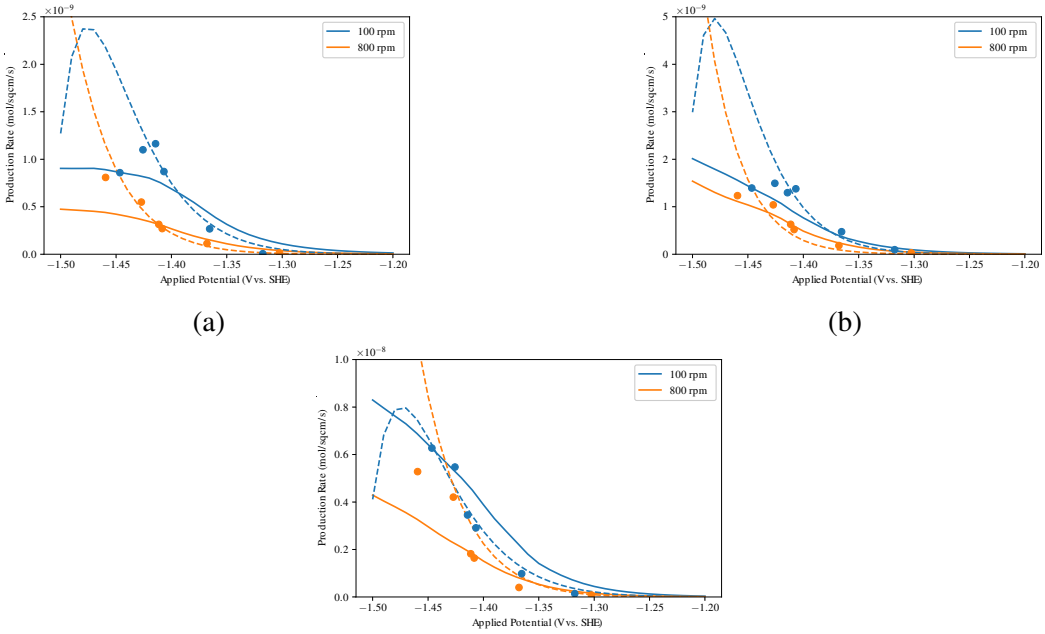


Figure 4.7: The production rates of (a) r_{C_1}, (b) $r_{C_{2+,HC}}$, and (c) $r_{C_{2+,OX}}$ from the EFP model (dashed) and the weighted-FNN model (solid) compared with the reference data points over the range of applied potentials in the unit of V vs. the standard hydrogen electrode (V vs. SHE).

Subsequently, the EFP model is compared with the FNN model using the testing set as described in Section "Data Generation and dataset". Table 4.5 shows that the FNN model outperforms the EFP model with significantly lower MSEs for all three rates, which implies that the accuracy of the EFP model can be improved by minimizing the deviation between FNN prediction and EFP model prediction. In other words, the FNN prediction can be considered as additional reference data to improve the EFP model performance for this reactor.

Additionally, by comparing the prediction trends from both models in Fig. 4.7, the EFP model overestimates the effect of applied potential for higher rotation speed. Thus, the empirical model can be improved by modifying the existing terms. The process is summarized by the following algorithm:

Algorithm 2: Empirical, First-Principles Model Improvement Procedure

X is the input data sequence, Θ contains the regression parameters in the empirical model to be optimized, C represents any additional system parameters that can be added to the empirical model, E and F_{ANN} are the general equations for the empirical and FNN models respectively, $\mathbf{D}$ calculates the distance, and I_{max} is the maximum number of iterations.

for $i = 0$ **to** I_{max} **do**

Obtain the distance between the empirical model and the FNN model:

$$\mathbf{D}(\Theta, X, C) = \sum \left[E(\Theta, X, C) - F_{ANN}(X)\right]^2$$

if $\mathbf{D}(\Theta, X, C_{new}) < \mathbf{D}(\Theta, X, C)$ **then**

$C = C_{new}$

else

$C = C$

end

Optimize the regression parameters to minimize the distance:

$$\Theta_{new} = \{\theta_{i,new} \in \Theta \mid D(\Theta_{new}, X, C) < D(\Theta, X, C)\}$$

end

Table 4.5: Testing data MSE results of the FNN model and the empirical, first-principles model.

Rate Index	FNN	EFP
$r_{C_{2+,OX}}$	0.0239	0.042
$r_{C_{2+,HC}}$	0.0098	0.042
r_{C_1}	0.003	0.021

Specifically, the regression parameters, θ_i, can be optimized to minimize the difference be-

Table 4.6: The non-scaled MSE for the updated and orignial EFP models.

Rate Index	**EFP (original)**	**EFP (updated)**
$r_{C_{2+},OX}$	3.68E-19	9.20E-20
$r_{C_{2+},HC}$	1.11E-18	2.32E-19
r_{C_1}	8.46E-18	5.76E-18

tween the two models using a user-defined optimization algorithm in each iteration of the procedure. Additionally, new system parameters, C_i, can be introduced to further develop the empirical model. For example, terms that describe the influence of gas pressure and flow rate on the electrochemical reaction can be included to further improve the current empirical model. Therefore, by following this procedure, the improvement of an empirical model can be represented as an optimization problem, which can be accomplished automatically by computers.

To demonstrate this procedure, we use the FNN to tune the parameters for the proposed EFP model. The difference between the EFP and FNN models is minimized on the range of -1.47V to -1.30V as this is the range for the data collected to train the FNN model. The MSE between the EFP model and FNN is calculated at intervals of 0.01V. For the model of Eq. 4.13, we apply a grid search to find the optimum values for the parameters k_i, α_i, and the exponent of the flux term J_{HCO_3}. Specifically, we search the α_i values from 0 to 1 with step sizes of 0.01, and the J_{HCO_3} exponents from -1 to 1 with step sizes of 0.25. However, small changes in the α_i and exponent of J_{HCO_3} can cause the k_i to change by several orders of magnitude which caused problems when attempting to use non-linear optimization packages. Therefore, the grid search for k_i must cover a large range of magnitudes from 0 to 1. To decrease the computational complexity, the grid search for k_i is split into two subsequent searches. Given that k_i can be expressed in the form of $A \times 10^{-B}$, we first search for the optimum order of magnitude B, and then search for the optimum number A.

Specifically, the first search follows a geometric sequence from 10^{-25} to 10^{-10} with a geometric ratio of 10 (i.e., 10^{-25}, 10^{-24}, 10^{-23}, ..., 10^{-10}). Then, given the optimum order of magnitude B, we search for the optimal number from 1 to 10 with step size of 0.1 (i.e., 0.1×10^{-B}, 0.2×10^{-B}, 0.3×10^{-B}, ..., 10.0×10^{-B}) for the order of magnitude B and $B + 1$.

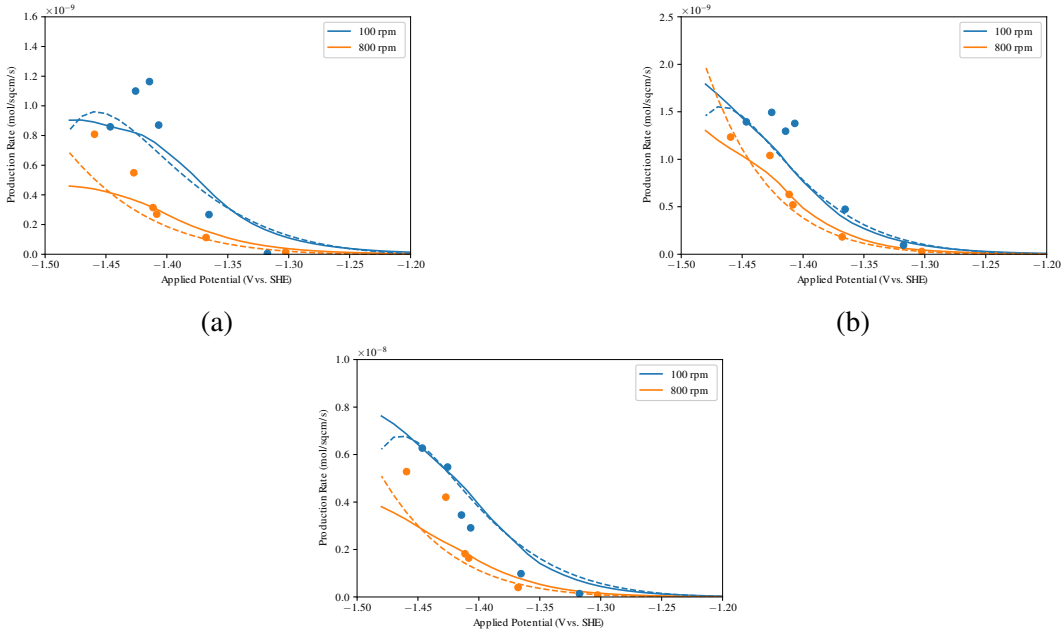


Figure 4.8: The production rates of (a) r_{C_1}, (b) $r_{C_{2+,HC}}$, and (c) $r_{C_{2+,OX}}$ from the updated EFP model (dashed) and the weighted-FNN model (solid) compared with the reference data points over the range of applied potentials in the unit of V vs. the standard hydrogen electrode (V vs. SHE).

Following this procedure, the optimum parameters are determined and listed in Table 4.4. Then, the new EFP model is tested against the same reference data points, and its MSE results are compared with the original EFP model listed in Table 4.6 . As a result, the MSE for r_{C1}, $r_{C2+,HC}$, and $r_{C2+,OX}$ against the reference dataset are decreases by 75%, 79%, and 32%, respectively. The new empirical, first-principles equations are updated in Eq. 4.14 to reflect the changes made from the procedure. Moreover, the new EFP model predictions are also shown in Fig. 4.8, which shows

the overestimating problem is solved with the new parameters.

$$
\begin{aligned}
r_{C_1} &= k_{0,5} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{-1} \exp\left(-\frac{\alpha_5 z_5 F}{RT} E\right) \\
r_{C_{2+,HC}} &= k_{0,6} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{-0.25} \exp\left(-\frac{\alpha_6 z_6 F}{RT} E\right) \\
r_{C_{2+,OX}} &= k_{0,7} C_{CO} Sh_{RCE}^{-0.5} J_{HCO_3}^{-1} \exp\left(-\frac{\alpha_7 z_7 F}{RT} E\right)
\end{aligned}
\tag{4.14}
$$

Remark 25. *Minimizing the difference between the two models will not result in exactly the same predictive model. During the optimization process, the empirical model structure derived from physical relations should remain unaltered. Furthermore, the additional terms C_i that have not been included in the previous empirical models should have physical meanings. In this way, the empirical model is modified toward a lower MSE for its prediction while respecting the physics of the experiment.*

Remark 26. *The experimental data is used to calculate the original EFP parameters. Specifically, the original EFP model is determined using traditional methods to extract kinetic parameters of α_i and k_i. Since the reaction rate is proportional to the exponential of the applied potential, the relationship is linearized by plotting the natural log of reaction rate against the applied potential. A linear regression is then used to find the slope and intercept of the observed data. The value of α_i is then extracted from the slope, and k_i is extracted from the intercept. Furthermore, by using the FNN model to propose an updated EFP model, meaningful process parameters can be extracted from the neural network regression, which provides additional explicit values to evaluate the neural network performance.*

Chapter 5

Machine Learning-Based Predictive Control Using On-line Model Linearization: Application to an Experimental Electrochemical Reactor

5.1 Introduction

In today's chemical manufacturing industry, fossil fuels serve as the primary energy source for the chemical industry, leading to significant energy consumption and greenhouse gas emissions [18]. Alternatively, there has been increasing interest in electrochemical reactions, such as converting carbon dioxide (CO_2) into carbon-based fuels and chemicals with electricity, as a means to mitigate CO_2 emissions. This approach holds the potential for leveraging electricity generated from renewable resources as an energy source for large-scale chemical manufacturing, furthermore con-

tributing to global-scale renewable energy storage and closing the anthropogenic carbon cycle [35]. Although electrochemical conversion of waste CO_2 is very promising, several challenges hinder the widespread adoption of electrochemical reactors on an industrial scale. Perhaps most importantly, the conversion of CO_2 through electrochemistry requires significant energy consumption [151]. Researchers have focused on improving energy efficiency in electrochemistry through the development of more efficient and selective catalysts through nanostructuring, doping of transition metals, utilization of single-atom catalysts, etc. [23, 79, 93, 114] as well as the design of devices to reduce the overall cell potential and address parasitic carbonation problems [130, 174, 187]. On the other hand, discussions on process scale-up have been limited so far [134]. We have identified another critical challenge in scaling up electrolyzers to be the absence of advanced process control schemes for electrochemical reactors due to the complex and nonlinear nature of electrochemical processes. Since the realization of an economically viable electrochemical process will require optimization in process integration and cascade reactor train [43, 118, 129], the development of a control scheme to regulate individual electrochemical reactor units is necessary.

To address this issue, [29] proposed a feedback control scheme using proportional-integral (PI) controllers utilizing a support vector regression-based (SVR) hybrid model as a state estimator. This approach enabled real-time state estimation for a PI controller and, subsequently, implementation of single-input-single-output (SISO) control in a gastight rotating cylinder electrode (RCE) cell. Building on this work, [31] introduced a recurrent neural network (RNN) model as an improved state estimator, surpassing the performance of the SVR model. This RNN model captured relationships between process variables and gas product concentration and allowed for the implementation of multi-input-multi-output (MIMO) control using PI control techniques for the same

RCE reactor. Alongside the classical control strategies, model predictive control (MPC) methods have emerged as vital components in industrial process control design [91, 125]. MPC offers the advantage of computing optimal control actions by anticipating future output states, making it a powerful tool for multivariable control while considering process constraints and nonlinearities, for example, [61].

Although the specific application of MPC in electrochemical reactors is limited, MPC has been widely used in various research areas, including chemical reactors, battery management, and self-driving cars. For instance, [133] provided a comprehensive discussion on implementing MPC for a crude oil distillation unit in the petroleum industry. Furthermore, [24] explored MPC design for a multivariable distillation column, demonstrating superior performance compared to PI-based control through MATLAB simulations using the Wood and Berry Model. MPC has also been applied to develop a battery management system, which is similar to the application of an electrochemical reactor in the sense that both tasks involve manipulating electrochemical reactions, even though the battery management system focuses on storing and releasing electricity instead of generating products using electrical potential. For example, [122] proposed a nonlinear MPC design based on the electrochemical models capturing the internal phenomena of the battery to solve the charge unbalancing problem in lithium-ion cells connected in series. These applications have demonstrated the ability of MPC to control systems with electrochemical reactions. Considering the advantages of MPC over classic control strategies (e.g., PI control) with respect to explicitly handling actuator and state constraints, multivariable interactions and nonlinearities, it is potentially practical and valuable to leverage the application of MPC to control electrochemical reactors.

Implementing MPC in electrochemical reactions poses two major challenges that need to be overcome: model accuracy and computational expense. The accuracy of the model prediction is crucial for the performance of MPC. Ideally, a first-principles-based model that accurately captures the underlying phenomena in the electrocatalytic system would be optimal. However, such models are often unavailable for practical cases. To this end, in our research, we focus on a data-driven approach to model the process system. Data-driven modeling offers a systematic approach that can be applied to any process system if sufficient data quantity and quality are ensured. One of the significant examples of data-driven modeling is ML, which is a class of techniques that can be generally applied to various systems without the need for formulating specific physical patterns discovered in experiments [38]. Classical ML methods, including SVR, linear regression, Gaussian process regression, and decision trees, have been widely utilized for modeling tasks [16, 59, 149, 171, 186]. Additionally, deep learning methods, employing neural network (NN) structures, have demonstrated superior performance in capturing nonlinear and complex systems compared to classical ML methods. As a result, NN modeling has drawn significant attention and has been applied in recent research works [115, 148]. Considering the nonlinearity and complexity of the electrochemical reactor and to facilitate the modeling process, a NN method is utilized to model the system, which has been demonstrated to be an effective technique for this specific reactor in [31].

While the use of nonlinear data-driven models in MPC has shown promising performance in various research studies, implementing MPC with a nonlinear model generally involves solving a non-convex optimization problem. This complexity often results in high computational costs and unstable gradient concerns. [175] demonstrated in their work that using RNN models in MPC can be highly accurate yet intractable to solve in real-time, motivating the exploration of

linearization approaches to improve the computational efficiency of MPC. Several techniques have been proposed for linearizing nonlinear models, such as Taylor series, piecewise linearization, etc. [89, 98, 141]. Specifically, the Koopman operator method is developed to be a data-driven approach that can be applied to any nonlinear model [10, 85, 124]. [175] showed the Koopman operator method is a type of linearization approach that can have better performance than the classical Taylor series method, particularly when linearizing over a larger domain. Furthermore, the application of MPC using a linearized model has been studied in-depth in the chemical engineering domain [74, 104]. These results have highlighted the potential of employing MPC with on-line linearized models in practical control applications. By leveraging efficient linearization techniques, NN-based MPC can potentially be applied to control the electrochemical reactor effectively.

Motivated by the above considerations, this study aims to develop an advanced process control scheme using MPC with suitable process models for an electrochemical CO_2 reduction reactor. Specifically, a neural network model is initially constructed using reactor data to capture the nonlinear complex input-output relation of the reactor, followed by on-line linearization of the NN model using the Koopman operator method to reduce the computational cost of MPC. The control design is applied experimentally to the electrochemical reactor. This chapter is organized into the following subsections: Section 5.2 introduces the background information of this study, including the mathematical notation used in this section, the overall design of the process, and the equipment setup. Section 5.3 elaborates on the technical details for the design and development of a NN model. Section 5.4 discusses the Koopman operator method and the procedure of using it to linearize the NN model in real-time. Finally, Section 5.5 reports the results of this study including simulation, open- and closed-loop experiments.

5.2 Preliminaries

5.2.1 Notation

For a matrix $\mathbf{M}$, the notation $\mathbf{M}^{-1}$ is used to represent the inverse of the matrix $\mathbf{M}$ and $\mathbf{M}^{\dagger}$ denotes the pseudoinverse of matrix $\mathbf{M}$. x, $\hat{x}$, and u are the controlled outputs of the process control system (i.e., the productivity of the reactor for the targeting species), the prediction of the process output given by the process model (i.e., the NN model), and the inputs (control actions) calculated by the process control system (i.e., applied potential, rotation speed, and current), respectively.

5.2.2 Process Overview

The overall objective of our process is to electrochemically reduce CO_2 into valuable chemical products and fuels. A copper electrode is used in this process because it is the only known single-element catalyst that can reduce CO_2 into C_{2+} hydrocarbons and alcohol products, which are energy-dense and valuable, with a considerable production rate [121]. However, the process of electrochemical CO_2 reduction on copper is intricate, which results in the production of 17 different chemicals through a series of complex reaction pathways [114]. Among multiple factors contributing to the complex reaction mechanisms, mass transport and reaction kinetics play critical roles. Specifically, the transport phenomena in the diffusion boundary layer are directly related to the residence time of the reactant CO_2 and intermediates near the catalyst surface as well as the adsorption on and desorption of the catalyst, which determine the selectivity of final products. On the other hand, the reaction kinetics on the catalyst surface is related to the number of electrons transferred to the surface, which can be manipulated by the applied potential. Therefore, we aim

to control the selectivity of electrochemical CO_2 reduction by controlling the aforementioned two input factors, potential and electrode rotation speed. Applying real-time control to any process requires on-line measurements of the process outputs. In this work, the productivity of four gas-phase products (i.e., hydrogen (H_2), carbon monoxide (CO), methane (CH_4), and ethylene (C_2H_4)) can be monitored in real-time using a gas chromatograph (GC). The overall reaction formulas producing these four products are summarized as follows:

$$2\,CO_2 + 8\,H_2O + 12\,e^- \longrightarrow C_2H_4 + 12\,OH^- \tag{5.1a}$$

$$CO_2 + H_2O + 2\,e^- \longrightarrow CO + 2\,OH^- \tag{5.1b}$$

$$CO_2 + 6\,H_2O + 8\,e^- \longrightarrow CH_4 + 8\,OH^- \tag{5.1c}$$

$$2\,H_2O + 2\,e^- \longrightarrow H_2 + 2\,OH^- \tag{5.1d}$$

Finally, the production rates of CO and C_2H_4 are chosen to be the control outputs to be regulated by the process control system. These two outputs are influenced differently by the input variables; specifically, the production rate of CO is highly correlated to the rotation speed, and the production rate of C_2H_4 is strongly influenced by the applied potential [31].

5.2.3 Electrochemical Reactor Setup

The gastight RCE cell was designed to examine how mass transport and reactions kinetics affect the electrochemical reduction of CO_2 while ensuring a gastight environment for the real-time detection of gas products [71]. As shown in Fig. 5.1, the experimental reactor consists of two reaction chambers separated by an anion-exchange membrane preventing the crossover of

products. The cathode is the working electrode in the cylindrical geometry carrying out the CO_2 reduction reaction while the Pt foil anode works as the counter electrode. Before each experiment, polycrystalline Cu RCE was mechanically and electrochemically polished following the procedure described in [71] followed by roughening of the surface via electrochemical redox cycling in the presence of chloride ions [135]. The preparation for this catalyst is the same as in our previous work [31]. Both the working and the counter electrodes are immersed in 0.2 M potassium bicarbonate electrolyte solutions. During the experiment, the CO_2 gas is directly bubbled into the electrolyte in both chambers with a fixed volumetric flow rate of 20 mL/min. Subsequently, the dissolved CO_2 molecules are transported to the reacting surface on the cathode to be reduced to various products. The potentiostat manipulates the potential applied to the working electrode against a reference Ag/AgCl electrode and records the electrical current passed between the working and the counter electrode. The control of the mass transport properties in the reactor is made possible by magnetically coupling the shaft where the RCE is mounted to another magnet connected to the modulated speed rotator (MSR) outside the reactor.

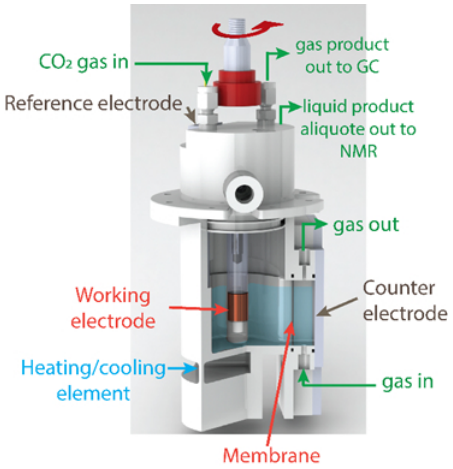


Figure 5.1: The experimental setup of the gastight rotating cylinder electrode (RCE) cell.

5.2.4 Model Predictive Control

MPC is an advanced control strategy used in various industrial processes. It involves utilizing a dynamic mathematical model of the system to predict its future state or output behavior and optimize control actions by iteratively solving an optimization problem over a defined time horizon. Specifically, MPC determines the optimal control actions to minimize a specified cost function while satisfying system constraints. The design of the MPC in this work can be mathematically defined as the following optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N_h}} L(\hat{x}(t), u(t))\, \mathrm{d}t \tag{5.2a}$$

$$\text{s.t.} \quad \hat{x}(t) = F_{nn}(x(t), u(t)) \tag{5.2b}$$

$$L(\hat{x}(t), u(t)) = (\hat{x}(t) - x_r)^\top Q(\hat{x}(t) - x_r) + (u(t) - u_r)^\top R(u(t) - u_r) \tag{5.2c}$$

$$u(t) \in U,\ \forall\, t \in [t_k, t_{k+N_h}) \tag{5.2d}$$

$$\hat{x}(t_k) = x(t_k) \tag{5.2e}$$

$$|u(t_k) - u(t_{k-1})| \leq u_c \tag{5.2f}$$

where x and $u \in \mathbb{R}^m$ are the output states and control actions (calculated by the model predictive control system), respectively. The set U represents the control action space that defines the upper and lower bounds of the m control actions applied to the reactor. The absolute difference between the control actions to be applied in the next control period from the instance time $u(t_k)$ and control action applied in the current control period $u(t_{k-1})$ is bounded by the vector u_c containing absolute boundaries for m control actions (in this particular case, $m = 2$ as we have two manipulated inputs

and dimension of $u(t_k)$ is 2). Furthermore, x_r and u_r are the reference values for the output states and control actions. Q and R represent the weight parameters (both are positive definite matrices) of the penalty terms for the output states and control actions, respectively, in the quadratic cost function $L(x, u)$. Therefore, by minimizing the cost function L with an appropriate manipulated input trajectory, the reactor can be driven to the desired set-point given by x_r by applying the first calculated control action $u(t_k)$ at each sampling time, and then repeating this process in the next sampling time. Finally, F_{nn} is the NN model, N_h is the prediction horizon, and the set $S(\Delta)$ comprises of piecewise constant functions having a period of Δ.

In this work, the outputs of the process to be regulated by the model predictive control system, x, are the production rates of CO, C_2H_4, and H_2. Specifically, we are aiming to get the productivity of CO and C_2H_4 to a certain set-point while minimizing the productivity of the side product hydrogen from the competing hydrogen evolution reaction.

5.3 Neural Network Modeling

To account for the complexity of the electrochemical reaction mechanism and fill in the lack of a first-principles model, a neural network (NN) model is developed to capture the dynamic response of the output states under various input conditions. Subsequently, the trained NN model is utilized as the process model of the MPC to estimate the output states over a certain time horizon known as the prediction horizon N_h. This section describes the design and development of the NN model for this purpose.

5.3.1 Data Collection

The data set used to develop the NN model is similar to the one reported in [31], and three types of experiments (i.e., open-loop steady input, step changes, and closed-loop experiments) are performed to collect the data. Specifically, constant inputs (applied potential and catalyst rotation speed) are applied to generate some portion of the training set data, which provides information about the expectation of the steady state output values under certain input conditions in addition to the dynamic trends while reaching respective steady states. In the second type of experiment, step-change inputs of random amplitudes are applied, but the input actions remain in a predefined range throughout the experiment. Finally, the closed-loop experimental results from [31] are included in the data set. Although the controller type used in [31] is different from the one in this work, the underlying physico-chemical phenomena are the same. Thus, including those results can help the model to capture the dynamic behavior of the system more efficiently.

GC is used to monitor the outlet concentrations of the gas products in real-time during data collection. Specifically, the GC takes a gas sample injection and quantifies the production rates of the four gas-phase products every 1300 seconds during the experiment. Analyzing the injected gas sample takes 15 minutes, and the GC needs to cool down for 400 seconds before taking the next injection. Therefore, only four data points can be collected from a one-hour duration of the experiment. As a result, there are a total of about 200 GC measurements collected at the end of data generation experiments for the training, which is not enough to train a neural network model. To address this problem, a 3^{rd}-order polynomial regression based on three consecutive GC measurements is applied to determine a probable output data trajectory between every GC

measurement using the inputs measured every second. More details about this data enhancement process are reported in [31].

5.3.2 Long Short-term Memory Networks

Among many ML methods that can be used to capture nonlinear processes, the RNN family has been proven to be an effective modeling strategy for time-series forecasting tasks. Recently, RNN models have become popular in the research area of process modeling and control and have been applied in many academic and industrial works [55, 167]. The long short-term memory network (LSTM) is one of the well-developed NN models that belongs to the RNN family. It shares the major design of architecture with other types of RNN models that have information flowing in two directions to capture the time-dependent relationship within the training data [132]. Furthermore, the LSTM model has its special "gates design" to store the historical information and determine how to use it to predict the output [60].

The architecture of the LSTM model used in this work is shown in Fig. 5.2. The model is developed to predict the output state at the next consecutive sampling time using p historical state predictions and control actions. Therefore, there are only three outputs given by the model, which represent production rates of CO, C_2H_4, and H_2 (in ppm) at the $p + 1$ time step. Specifically, the LSTM layer maps the time-sequence input containing the historical state prediction and control actions to 180 hidden states. Subsequently, a dropout layer with a 30% dropout rate of the hidden states is inserted to prevent overfitting, and the remaining hidden states are densely connected to the output nodes.

Remark 27. *The number of hidden states and percentage rate of dropout are included in the*

hyperparameters of the LSTM model. Therefore, their value can be found following the general hyperparameter tuning process. Specifically, in this work, we performed a random search to locate those values. More precise methods to perform the hyperparameter tuning include cross-validation and grid search. More details about hyperparameter tuning can be found in [45].

Remark 28. *In the area of ML, preventing the model from overfitting the data is an important task. Overfitting refers to the situation where the NN model can perform well with the training data set but fails to maintain good performance for data outside the training set. Several factors contribute to this problem, and a critical one is when the NN model has too many weight parameters, which can result in allowing the model to memorize the training data instead of extracting underlying trends from it. One method to reduce overfitting is the regularization [50], with the dropout method used in this work being one example of a regularization method [150, 158].*

5.3.3 Model Training

Based on prior knowledge of the experimental reactor, the input sequence of the LSTM model is designed to contain one hour (3600 seconds) of historical information. However, if the data is formatted on a per-second basis, each sequence in the data set will contain 3600 elements, which results in unnecessarily high computational costs in both time and space consumption. Therefore, the time step of the data sequence is designed to be on a per-hundred seconds basis to reduce the length of the input sequence to 36 elements. The time space in the input sequence is preserved in the output sequence, and since the output sequence only contains 1 time step, the overall function of the LSTM model is to use the one-hour historical information up to the instant to predict the

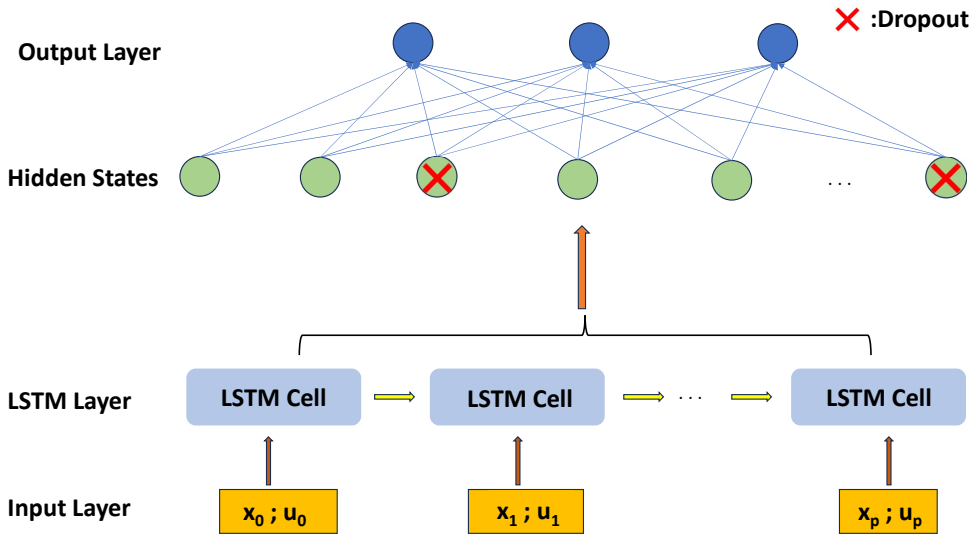


Figure 5.2: The architecture of the LSTM model used in this work that processes the input sequence with a LSTM layer and yields the prediction for the output states at the next time step (i.e., 100 secs later from the instantaneous point in time).

output states at 100 seconds later in the future (i.e., the output sequence has a shape of (1,3), where 3 is the number of output species).

This sliding window algorithm was employed to create a training dataset from a collection of 35 experiments. Specifically, a window of one hour was used as the input for the training data, and the output of the LSTM was determined to be the production rates of the target species at 100 seconds after the final time step of the input sequence. The window was systematically slid by a stride of 100 seconds, and the first 1000 (seconds of) measurements in each experiment were skipped to enhance the reliability of the training data. Therefore, the input of the LSTM model has a shape of (36, 6), where 6 denotes the input features (i.e., surface potential, rotation speed, current, and previous states of production rates for C_2H_4, CO, H_2) measured at the respective time

step. The sliding window algorithm is applied to 18 experimental data sets to generate the data sequence to develop the LSTM model. When developing an NN model for time-series forecasting problems, it is crucial to ensure that the validation data retains a certain level of independence from the training data to avoid potential information leakage. To address this concern, we randomly allocated results from 5 out of the total 18 experiments as the testing set, while the remaining experiments were assigned to the training set. This training set is further divided into two parts before the model training for ratio validation purpose, using the train-test ratio of 70:30. Finally, the Scikit-learn Minmax Scaler was utilized to normalize the data.

In this study, the LSTM model was trained using the TensorFlow API. The model was optimized with the NADAM optimizer. As the data did not provide dense coverage of the overall operating conditions, it was crucial to maintain generalization and prevent overfitting. Therefore, we applied L2 regularization to the LSTM layer with a factor of 0.07 and perform 30% recurrent dropout within the LSTM cells. The mean squared error was selected to be the cost function to evaluate the model performance. The LSTM model underwent training for 45 epochs with a batch size of 32. Additionally, a callback function was utilized to capture the best-performing weights based on minimization of the validation loss throughout the training process. As a result, the training and validation loss of the trained LSTM model were found to be 0.0028 and 0.00456, respectively.

Remark 29. *The length of the input sequence (i.e., 3600 seconds) was found based on the combination of experimental observations and hyperparameter tuning. Specifically, from the experiment, we found that the dead time of the process can vary up to 2000 secs, which meant, to capture the*

delay of the reactor, the length of the input sequence should be at least 2000 secs. Starting from there, we tuned the length of the input sequence and found that, with the length of 3600 secs, the LSTM model can capture well the dynamics of the process. Notably, increasing the length of the input sequence will result in higher computational cost, and since the length of the input sequence can be considered as a part of the hyperparameter tuning, the cross-validation method was used in this step.

5.3.4 Model Performance

The trained model demonstrates significantly low training and validation losses, indicating its successful training. To further assess the model's performance, a comparison is made between the model's predictions, based on input data recorded from a validation experiment, and the corresponding output state measurements. Figure 5.3 is an example of such a comparison, with solid curves representing the predictions made by the LSTM model and dashed curves representing the measured trends during the experiment. The close alignment between the curves depicted in Fig. 5.3 highlights the model's adequate prediction capabilities.

Once the model demonstrated its proficiency in predicting the dynamic behavior of the output state, we proceeded to evaluate its ability to accurately capture the reactor's steady state performance. The electrochemical reactor is an inherently stable process in the operating region of interest, meaning that regardless of the initial output state conditions, the application of the same constant control actions throughout a period of time should lead to the convergence of the outlet species concentrations to the same steady state every time. The prediction results of the LSTM model for an open-loop experiment are shown in Fig. 5.4. It can be observed that, regardless of

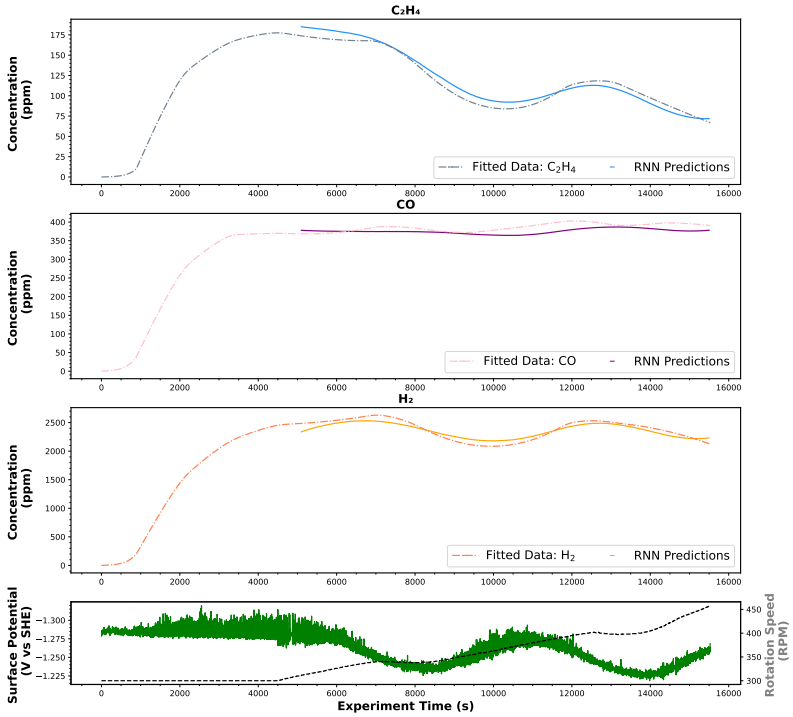


Figure 5.3: LSTM predictions of C_2H_4, CO, and H_2 concentrations compared to the reference data in the testing set. Inputs (surface potential and electrode rotation speed) used for the prediction are shown at the bottom.

the starting point of the trend, the predictions consistently converge to the same steady state for all three output states under a fixed control action. However, due to the stochastic nature of the electrochemical reaction and other experimental uncertainties, there exists a variance in the steady state. Therefore, the steady state given by the LSTM is ideally the average of the steady state values obtained if the experiment is repeated with the same fixed control inputs.

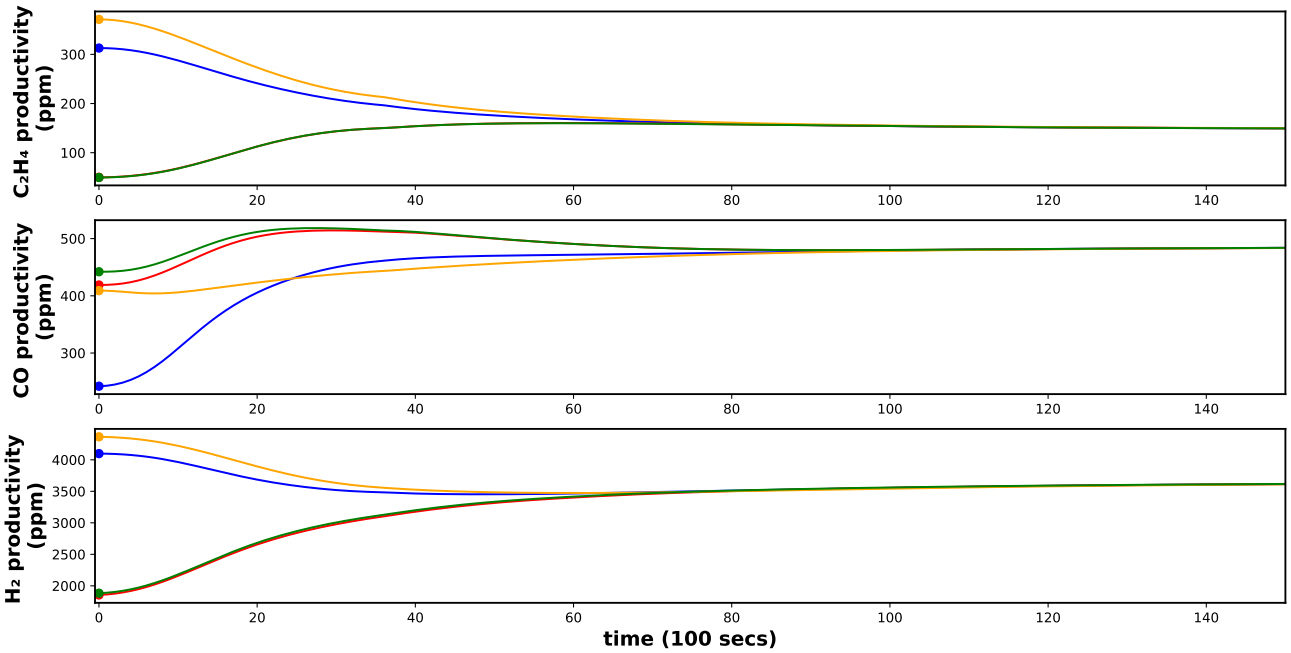


Figure 5.4: Open-loop simulation using the trained LSTM model with consistent fixed inputs from various initial states. The predicted trends for different initial states are represented in different colors.

5.4 Koopman Operator-based Linearization of RNN Model

The motivation behind the exploration of a method to linearize the neural network model for utilization in MPC arises from the fact that NN-based MPC involves solving a constrained optimization problem with a highly nonlinear NN model. Consequently, this optimization problem becomes a challenging nonlinear optimization task, which remains a topic of considerable mathematical exploration without a definitive approach for effective resolution. As a result, solving the nonlinear optimization problem within a reasonable time frame (certainly, within the process sampling time for real-time control purposes) might not be possible, which renders this type of nonlinear MPC application impractical for many industrial processes. On the other hand, the de-

velopment of an MPC framework with a linearized system is a well-established approach. By approximating the NN model with a linear system on-line and at each sampling time, an MPC can be formulated as a quadratic programming problem, which lends itself to efficient solution techniques. This implies that if we can effectively approximate the NN model with a linear system, the NN model-based MPC can be solved quickly and efficiently, and applied to real-world applications. This section presents the systematic process utilized in this project, drawing inspiration from the work of [175], to linearize the RNN-based process model and integrate it into an MPC.

5.4.1 Koopman Operator Theory

[175] presented a method to linearize an RNN model based on the principles of the Koopman operator theory. The Koopman operator theory, initially proposed by Bernard Koopman in the 19th century [83, 84], plays an important role in analyzing, modeling, and controlling nonlinear processes. The core concept of the Koopman operator theory involves mapping inputs of a nonlinear function into a higher-dimensional feature space, thereby obtaining a linear approximation of the nonlinear system [83]. In other words, the Koopman operator can linearize an arbitrary finite-dimensional nonlinear system at the cost of expanding its dimensionality up to infinity. Notably, this concept is also similar to the idea of feature engineering, in the ML terminology, which serves as a fundamental aspect in various ML models such as support vector machines (SVM) [32].

The Koopman operator can be defined mathematically with the following equations:

$$x_{k+1} = \mathbf{f}(x_k) \tag{5.3a}$$

$$\mathcal{K}g(x_k) \triangleq g(\mathbf{f}(x_k)) = g(x_{k+1}) \tag{5.3b}$$

where Eq. 5.3a is the discrete representation of a nonlinear dynamical system, and the function $\mathbf{f}$ captures the output state evolution of the system from an arbitrary time step k. Eq. 5.3b is the definition of the Koopman operator $\mathcal{K}$, where $g(\cdot)$ are a set of scalar functions named observables. From Eq. 5.3b, it can be easily proven that $\mathcal{K}$ is a linear operator, which allows finding the eigen decomposition of $\mathcal{K}$ and rewriting the evolution of the nonlinear system as a linear combination based on the eigen decomposition of $\mathcal{K}$ as follows:

$$\mathcal{K}\phi_j(x) = \lambda_j \phi_j(x), \quad j = 1, 2, \ldots, \infty \tag{5.4a}$$

$$\mathcal{K}g(x_k) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x_k)\mathbf{v}_j \tag{5.4b}$$

where λ_j, ϕ_j, $\mathbf{v}_j$ are known as the eigenvalues, the eigenfunctions, and the mode of the Koopman operator $\mathcal{K}$.

The discussion about Koopman operator theory so far has been centered around an autonomous system with time-varying inputs. However, to allow using this method in a dynamic control system requires extending the Koopman theory to be able to handle a non-autonomous system including time-varying control inputs. In [124], a generative Koopman with inputs and control (KIC) method was proposed to generalize the application of Koopman operator theory to non-autonomous systems. Specifically, the KIC method defined a new representation of the Koopman operator as follows:

$$x_{k+1} = \mathbf{f}(x_k, u_k) \tag{5.5a}$$

$$\mathcal{K}g(x_k, u_k) \triangleq g(\mathbf{f}(x_k, u_k), u_{k+1}) = g(x_{k+1}, u_{k+1}) \tag{5.5b}$$

where u_k is the input applied at the k^{th} time step, and Eq. 5.5a is the discrete representation of any nonlinear system accepting external inputs. There are other works proposing different formulations for Koopman operator with inputs [85], and the core ideas shared around those methods involve augmenting the states x and the inputs u into the same matrix and use it to form the observables instead of just the states, which all allow linearizing the nonlinear system using the method applied to an autonomous system.

Remark 30. *The method of constructing the observables g is an essential research area of Koopman operator theory, and there are significant efforts on this subject, such as using a nonlinear function to augment the state measurements [22, 124, 137, 162]. In this work, we define the observables to be the output states of the system, such that $g(x) = x$.*

Remark 31. *The Koopman operator can also be applied to a dynamic system with continuous representation. However, the focus of our mathematical analysis and investigation in this work is centered on the discrete representation, as the LSTM model can be considered as a discrete approximation of the underlying nonlinear dynamic system. Therefore, applying the Koopman operator to a discrete nonlinear dynamic system fits better to the application of the Koopman operator to the LSTM model.*

5.4.2 Dynamic Mode Decomposition

Although the Koopman operator method suggests linearizing a nonlinear system into an infinite-dimensional linear system, it is practical to work with a finite dimension that is high

enough to achieve the desired accuracy. Considering this, the Dynamic Mode Decomposition (DMD) method, first proposed in [142], is an effective method to provide a finite-dimensional approximation of the Koopman operator. Specifically, the DMD method is a data-driven method that requires obtaining measurements to start with. We define $O_k = g(x_k)$ to be the observation of a nonlinear system and $O_k^+ = O_{k+1}$ to be the observation one time-step after O_k. By performing experiments or simulations with the nonlinear system, time-sequence data can be collected for the observations and yield:

$$\mathbf{O} = [O_0, O_1, \ldots, O_{n_s}], \;\; \mathbf{O}^+ = [O_0^+, O_1^+, \ldots, O_{n_s}^+] \tag{5.6}$$

where n_s is the total number of samples. Notably, the notation O_1 is not necessarily the next time step of O_0. Subsequently, the DMD of the nonlinear system based on the measurements can be found as the eigen decomposition of the linear mapping matrix $\mathbf{A}$ that forms the equation,

$$\mathbf{O}^+ = \mathbf{A}\mathbf{O} \tag{5.7}$$

The analytical solution of Eq. 5.7 yields the matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{O}^+\mathbf{O}^\dagger$. Finally, the eigenvalues and eigenvectors of $\mathbf{A}$ are the approximation of the eigenvalues and the mode of the Koopman operator, respectively.

For a nonlinear system with inputs, the Dynamic Mode Decomposition with control (DMDc) method was proposed in [123], which includes the measurements of the control actions $\mathbf{O}_u = [u_0, u_1, \ldots, u_{n_s}]$ to compute the linear mapping matrix $\mathbf{G} = \mathbf{O}^+ \begin{bmatrix} \mathbf{O} \\ \mathbf{O}_u \end{bmatrix}^\dagger$ defined for the DMDc method. Similarly, the singular value decomposition of $\mathbf{G}$ can provide a finite approximation

of KIC. Eventually, an arbitrary non-autonomous nonlinear system defined as Eq. 5.5a can be linearized with the DMDc method into the following system:

$$\mathbf{G} = [\mathbf{A}\ \mathbf{B}] \tag{5.8a}$$

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k \tag{5.8b}$$

$$y_k = \mathbf{C}x_k + \mathbf{D}u_k \tag{5.8c}$$

Furthermore, the process of computing the matrix $\mathbf{G}$ involves solving a linear least-squares problem, which can be solved more effectively in practice with the regression method rather than finding the analytical solution [85]. Therefore, the extended dynamic mode decomposition (EDMD) method proposed in [162] introduced a regression procedure to approximate the Koopman operator.

5.4.3 Linearization of LSTM model and Performance Evaluation

The Koopman operator theory and EDMD method are utilized to linearize the LSTM model because they are data-driven and independent of the form of the nonlinear model [9, 175]. The first step of implementing these methods is to collect time sequence trajectories of the LSTM model. Following the procedure of [124, 175], at the k^{th} time step, we first define the vectors $y = [\hat{x}_{k+1}, \hat{x}_{k+2}, \ldots, \hat{x}_{k+N_t}]^\top$, $\hat{x} = [\hat{x}_k, \hat{x}_{k+1}, \ldots, \hat{x}_{k+N_t-1}]^\top$, and $u = [u_k, u_{k+1}, \ldots, u_{k+N_t-1}]^\top$, where N_t is the distance between the farthest time step contained in the linearization samples and the k^{th} time step. Notably, the historical information that is used by the LSTM model to make predictions up to the k^{th} time step is available at the time t_k. Thus, the prediction $\hat{x}_{k+1}$ can be

computed using the LSTM model. Furthermore, by adding the new prediction and the next control action u_{k+1} while removing the first element of the LSTM input, the vector y can be obtained by iteratively running the LSTM model.

In this work, we applied a constraint on how much the input actions can be changed from one sampling time to the next, which is mathematically defined by the following equations:

$$u_k = [v_k, r_k, c_k] \tag{5.9a}$$

$$c_k = C(v_k, r_k) \tag{5.9b}$$

$$u_d \triangleq [v_{k+1} - v_k, r_{k+1} - r_k] \tag{5.9c}$$

$$|u_d| \leq u_c = [v_b, r_b] \tag{5.9d}$$

where v_i, r_i, c_i are the surface potential, rotation speed, and the current value given by the reactor at the i^{th} sampling time. The surface potential and rotation speed are the control actions that can be manipulated during the experiment, and the current varies as a consequence of these control actions. v_b and r_b are positive numbers referring to the maximum absolute step changes allowed per time step for the potential and rotation speed and are equal to 0.01 V and 30 RPM, respectively.

The data to linearize the LSTM model is generated with respect to the constraints of Eq. 5.9. Specifically, we first determined the number (N_s) and the length (N_t) of the time-sequence data. Then, we randomly generate N_t control actions starting from the same initial control action u_0 that obey the constraints of Eq. 5.9 and run the LSTM model to generate one time sequence of "measurements" of y. This process is repeated N_s times to obtain N_s sequences. [155] pointed

out that, due to the reduction of the problem into a linear regression formulation, the data set used to perform the DMD-based method does not need to retain the sequential order of the data points (i.e., the rows of the data matrices can be shuffled such that, for example, the last row of the target vector y, $\hat{x}_{k+N_t}$ can instead be moved to be the first row, as long as the last rows of $\hat{x}$ and u, $\hat{x}_{k+N_t-1}$ and u_{k+N_t-1} are also moved to be their first rows, respectively). Therefore, the data matrices y, x, and u with a shape of $(N_s \times N_t)$ can be reshaped into three vectors containing $(N_s \times N_t)$ elements (also note $n_s = N_s \times N_t$), as long as the triplets of x_i, u_i, and y_i remains the same. With this data structure, the linear least-square regression problem to find $\mathbf{G}$ can be easily solved by using the Scikit-learn linear regression function without fitting the intercept. The pseudocode to implement the linearization of LSTM in our work is represented in algorithm 3.

Algorithm 3: Procedure of linearizing the LSTM model.

Initial: $v_b \geq 0, r_b \geq 0, NN$; // NN: the initial inputs of LSTM with shape (1, 36, 6)
Initial: x_{train} , y_{train} ; // Define empty arrays to store linearization samples
for $i = 1$ **to** N_s **do**
  $NN_i = NN$; // Initialize the LSTM input by making a deepcopy of NN
  for $t = 1$ **to** N_t **do**
    $v_i = NN_i[0, -2, 0] + rand(-v_b, v_b)$; // $rand(l, h)$: randomly pick number between l and h
    $r_i = NN_i[0, -2, 1] + rand(-r_b, r_b)$ $c_i = C(v_i, r_i)$; // $C(\cdot)$: eq. 5.9b
    $NN_i[0, -1, :3] = [v_i, r_i, c_i]$
    $\hat{x}_{i,t} = LSTM(NN_i)$; // $LSTM(\cdot)$: LSTM prediction
    $x_{\text{train}}.append(NN_i[0, -1, :])$
    $y_{\text{train}}.append(\hat{x}_{i,t})$
    $NN_i[0, :-1, :] = NN_i[0, 1:, :]$
    $NN_i[0, -1, 3:] = \hat{x}_{i,t}$
  end
end
$[A, B] = LG(x_{\text{train}}, y_{\text{train}})$; // $LG(\cdot)$: Sickit-learn linear regression

The prediction given by the linearized model was compared with the original LSTM predic-

tion. Specifically, the initial inputs to activate the LSTM model are randomly cropped from existing experimental results and provided to algorithm 3 to generate a linearized model. Subsequently, the linearized model was utilized to make predictions over a time span based on a sequence of control actions randomly picked within the step change constraint and compared to the prediction given by the original LSTM model using the same control sequence. The comparison is shown in Fig. 5.5, where the prediction given by the original LSTM model over a time span of 800 secs is represented in the blue solid curve, while the prediction given by the linearized model is denoted in the red dashed curve. The predictions given by the two models are close to each other, which supports that the linearized model can approximate the LSTM model adequately.

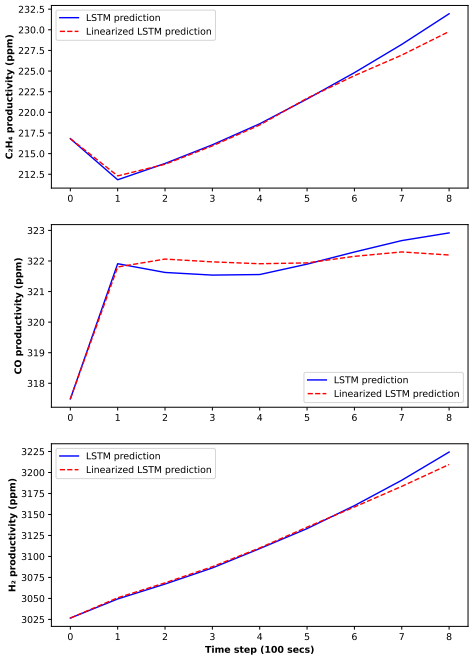


Figure 5.5: Comparison between the linearized model prediction (dashed curve) and the original LSTM model prediction (solid curve) over a sampling period.

Remark 32. *Notably, for this work, the current flowing at a fixed applied potential depends on the electrolyte solution resistance due to the Ohmic loss. The solution resistance between the working and the reference electrode is determined from the electrochemical impedance spectroscopy (EIS) and is around* $6.5 \pm 0.3\ \Omega$ *in the RCE cell setup when using 0.2 M potassium bicarbonate electrolyte. Although the value is practically constant, there are slight variances from experiment to experiment and during the experiment, while it is measured only before and after the experiments [31]. Therefore, the measured current value will not be the same with the same control action, and thus, provide additional information for our LSTM model to learn the electrochemical reactor system better. The current value is measured and recorded during the experiment, and those measurements are used to train the LSTM model. However, when collecting samples for the Koopman-based linearization of the LSTM model, the value of the current needs to be approximated with the correlation between* c_k *and the control actions denoted as transformation* C *in Eq. 5.9b. In simulations, this value was approximated using the average resistance obtained from various experiments. For the closed-loop experiments, the resistance value was measured right before starting the experiment and used to anticipate the current value in the prediction horizon.*

5.5 Closed-Loop Experiments

The details of implementing the linearized NN-based MPC for the electrochemical reactor are presented in this section. As a quick recap, referring to Eq. 5.2, the main objective of the MPC in this work is to drive the productivity of C_2H_4 and CO to their specific set-points while suppressing the productivity of H_2. The set-points for C_2H_4 and CO are selected to be 147 ppm

and 478 ppm, respectively, such that $x_r = [147, 478, 0]$. Furthermore, by replacing the LSTM model used in Eq. 5.2d with the linearized model, the overall optimization problem within the MPC becomes a quadratic programming problem, which is convex and can be solved efficiently. In the closed-loop experiment, the MPC is operated in a sample-and-hold manner, which means it will give the optimum control action over a certain control period (i.e., 100 seconds in this work), and the control action will be held fixed during the control period. The overall workflow of the MPC is demonstrated in Fig. 5.6. Specifically, the LSTM model worked as the state estimator throughout the experiment. When entering a new sampling time, algorithm 3 was used to compute the linearization of the LSTM model for the specific time-instant, which was then used to find the MPC control action by solving a QP problem. The Gurobi optimizer was used to solve the optimization problem in this work.

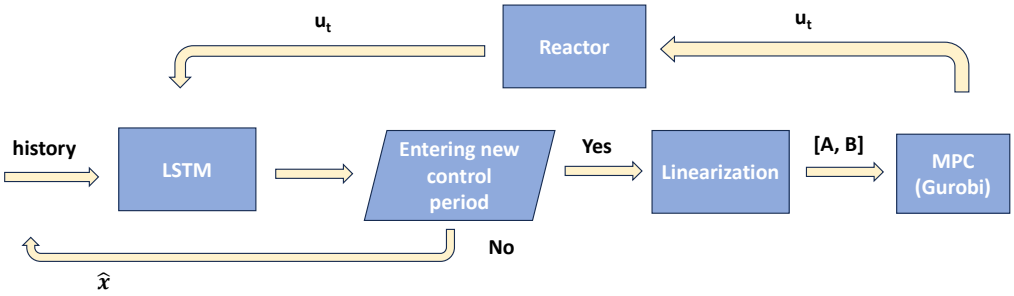


Figure 5.6: The overall workflow of the MPC in this work. The LSTM model is used as a state estimator when the MPC is not activated. Once entering a new sampling time, the MPC is activated and computes the control action for the reactor with the linearization of the LSTM model.

5.5.1 Implementation of MPC in the Experimental Setup

The LSTM model (without linearization) worked as the state estimator in the closed-loop experiments by predicting the instantaneous reactor productivity. When the processing of a new GC measurement is finished, the LSTM model has to reinitialize its prediction, which means that it uses the state measurements in the input of LSTM instead of the state prediction given by the LSTM in the previous iteration. This reinitialization is expected to prevent the accumulation of prediction errors. Specifically, the output states are estimated through 3^{rd}-order polynomials based on the 3 consecutive GC measurements up to the newest measurement and used as the input of the LSTM model to predict the output state at 100 seconds after the newest GC injection made. Note that the GC measurement has a delay of 15 minutes because it takes 15 minutes to separate and analyze the sample taken from the injection. Therefore, through the reinitialization, the LSTM predicted the output state 15 minutes ago again, and needs to run iteratively using the reinitialized prediction to correct all the predictions for the previous 15 minutes. Furthermore, since this 3^{rd}-order polynomial's approximation can only be activated once every 21 minutes, it can not be used as the process model or state estimator that requires to be able to give prediction every 100 seconds. But once the 3^{rd}-order polynomial approximation is activated, it can estimate the output states for the last 1 hr effectively and accurately.

The Laboratory Virtual Instrument Engineering Workbench (LabVIEW) software was utilized to digitalize the electrochemical reactor in this work. LabVIEW is a graphical programming language that allows a user to develop a user interface to monitor the system and develop control systems to implement the control actions in the working equipment. Although LabVIEW also allows

users to develop simple programs (e.g., PI controller), it is technically challenging to implement the aforementioned workflow that involves using the NN model, linearization, and optimization in LabVIEW. Therefore, a data pipeline was developed to allow information to flow between a Python script operating the workflow and the LabVIEW controlling the operating equipment.

The options for pipeline design include reading the data from a real-time updated csv file or data transfer through a database. Since opening a real-time updated csv file to read data might disturb the process of writing data, this option is not optimal. On the other hand, sending data from LabVIEW to a database is an easy task that is already combined into our automation scheme using the Clean Energy Smart Manufacturing Innovation Institute's (CESMII) Smart Manufacturing Innovation Platform (SMIP) as discussed in [30]. Specifically, the SMIP can work as a database to store and organize our data at defined endpoints for each piece of equipment, and the use of start and end dates for query and mutation of SMIP's data transfer protocol, GraphQL, makes it a perfect candidate for data transfer application. In short, this data flow is designed to use LabVIEW as the edge device performing process control and monitor tasks, SMIP as a cloud database for data management, and a high-performance computer running Python interpreters as a back-end server. The data transfer protocol is shown in Fig. 5.7.

When performing an experiment, constant physical properties, such as solution resistance, open circuit potential, etc., are measured before the electrolysis and sent to the SMIP at the beginning of the experiment. Process data collected through LabVIEW, such as applied potential, surface potential, rotation speed, and current, are mutated to the SMIP every 2 seconds. The reactor was put to run in open-loop for the first 7000 seconds of the experiment because, at the beginning state of the experiment, the rector does not reach the equilibrium giving higher variance

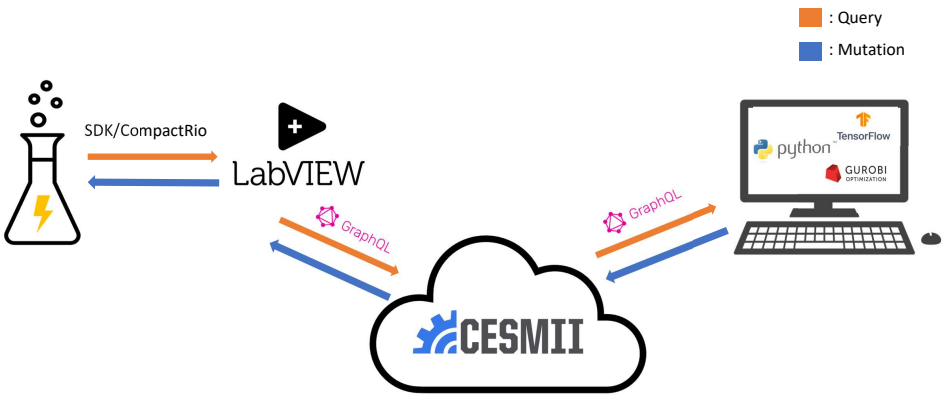


Figure 5.7: Data flow between the experimental setup and local Python script through SMIP for MPC calculations.

in its productivity. Thus, data collected at this stage is expected to have a different distribution from the rest of the experiment and is excluded from the LSTM training. Control actions applied to the reactor at this stage are fixed to be $-1.22\ V$ for potential and 300 RPM for rotation speed.

After letting the reactor run in open-loop for the first 6298 seconds, the MPC will be activated, and the Python script queries the last one hour of process data every 100 seconds to form the initial input for the LSTM model. Subsequently, the LSTM was linearized to compute the first control action, while the original LSTM model estimations along with input values are mutated to SMIP. LabVIEW script also queries those values from the SMIP to feed the new input values to the potentiostat and modulated speed rotator to implement the new applied potential and rotation speed. From then on, the experiment was run in closed-loop.

5.5.2 Closed-loop Experiments

The result of the closed-loop experiment is summarized in Fig. 5.8, where the dots are the GC measurements collected in the experiment and the dashed curves are the approximated output states with probable experimental trajectory method employing 3rd-order polynomials. The productivity evolution of C_2H_4 and CO_2 is shown in the top figure, and their set-points, which are 147 and 478 ppm, respectively, are denoted with the dotted straight lines. A reference control action $u_r = [-1.28\ V,\ 600$ RPM$]$, stated in Eq. 5.2c, was used in the MPC objective function to achieve better control performance. The reference control action u_r was found using the trained LSTM model. As discussed above, the LSTM model is stable, such that it can be used to find the theoretical control actions that can give the targeted steady state. The weight matrix was chosen to be $Q = \text{diag}(0.01, 0.01, 1 \times 10^{-6})$ and $R = \text{diag}(1 \times 10^4, 1.0/1200)$. The weight parameters were tuned based on the simulation and experiment results.

The design of the weight matrices considered scaling their importance on the cost function. For example, when designing the matrix Q, the first two parameters are the weight of C_2H_4 and CO_2, respectively, which are equally important in our control scheme. The value 0.01 in the Q matrix was used to prevent the cost value from becoming too big. On the other hand, the cost for the H_2 is much less than the other two weights because driving the outputs to the set-point is the first priority for the control system. Reducing the productivity of the side product H_2 can maximize the energy efficiency of our reactor. However, it is physically impossible to eliminate the H_2 production, which means if the weight parameter for the H_2 is too high, the MPC will allow the two target states to be away from the set-point as the trade-off to reach the optimum defined by the

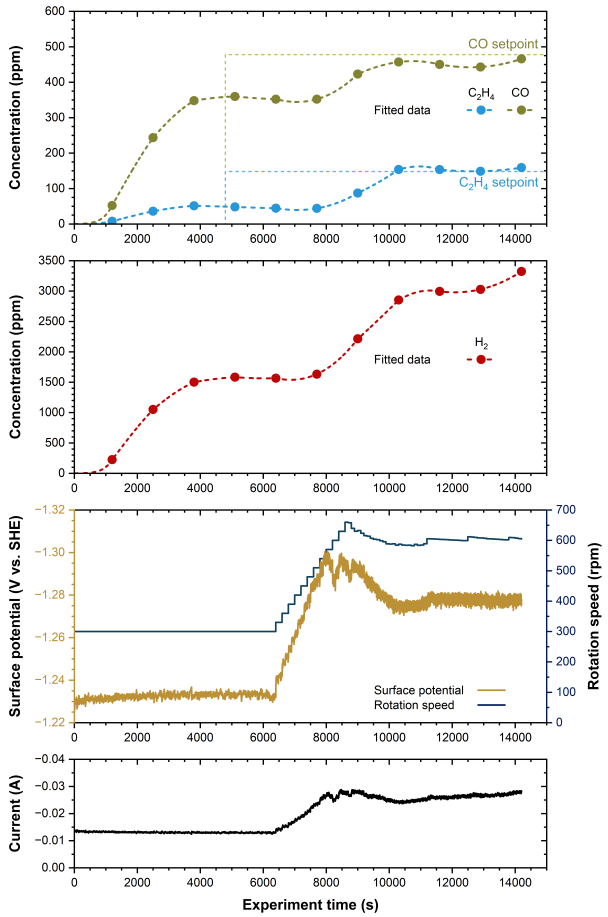


Figure 5.8: Output responses and control actions in the closed-loop experiment controlled by the MPC using the linearization of the LSTM model.

MPC objective function. Drawn from the understanding of the reactor, we considered the reactor operated energy efficiently if the productivity of H_2 was kept below 4000 ppm. Since the states are squared in the objective function, using 1×10^{-6} will give a cost equal to 16 if the productivity of H_2 is at 4000 ppm, which will dominate the MPC with the outputs approaching the set-points, and thus, the productivity of H_2 was not included in the objective function. The weight matrix R was calculated to balance the speed of the convergence of the states to the steady state and the magnitude of the control actions.

The prediction horizon of the MPC (N_h) is 8 times steps (i.e., 800 seconds in the future), and

the length of the time sequence, N_t, collected for linearization is designed to be equal to N_h+2. The number of linearization samples, N_s, is taken to be 30. Theoretically, linear regression can be more accurate with increasing amounts of data. On the other hand, increasing N_t and N_s also requires more time to collect the sample which makes the linearization more computationally expensive. Notably, the sampling step can be processed in parallel, which means the computational time is independent of the size of N_s if there are sufficient amounts of parallel processors. However, since the processing of the time sequence is iterative, the computation time for linearization is bounded by the size of N_t. In this work, the maximum allowable step changes in u reduces the required number of sequences in our linearization sample. In our implementation, 30 sequences collected with the Monte Carlo method turned out to be sufficient for the linearization task. With this choice, the MPC successfully drives the outputs to the set-point while maintaining the H_2 production rate below 4000 ppm.

5.5.3 Model Retrain

The MPC design in this work created a feedback loop by using real-time measurements to re-initiate the LSTM model with measurement feedback and improve closed-loop system robustness. However, a correction algorithm that uses the feedback information to improve the LSTM model was not implemented, which means the control scheme demonstrated in this work is based on the assumption that the LSTM model can capture the real process accurately and that the process behavior does not change significantly. However, in real-world applications, the system is very likely to perform differently from the model prediction due to the variance of the application. The reported control scheme in this work should be able to handle this slight variance, as long as

the variance is not significant enough to have the steady state shifting on a very different condition. However, sometimes the process may have very different behavior than the data collected to develop the neural network model. This problem is usually called the data (process) shift problem (for example, due to catalyst activity variation as a new catalyst is introduced every certain number of experiments), and more actions need to be taken to account for the data shift problem in model update.

To this end, we introduced a model retrain procedure based on the transfer learning concept to update the process model efficiently when the data shift problem is detected. To imitate this data shift problem in our reactor, we changed the polycrystalline Cu RCE to a new one and followed the same procedure to synthesize nanopores, but the resulting performance was different. Specifically, the new catalyst was more active and had a selectivity toward the C_2H_4 production. Various experiments were conducted with the new catalyst, which gave the result that with the control action at -1.28 V and 600 RPM, the output for C_2H_4 increased to about 200 ppm but remained unchanged for CO.

Subsequently, the LSTM model was retrained based on the data collected from the new experiments. Of course, the new data set is smaller in size compared to the original data set. Therefore, if we just train a new model after including the new data into the original data set, the newly trained LSTM is very likely to count heavily on the old data set and represent less of the performance of the new catalyst. To account for this, we used the idea of transfer learning, which is a scheme to fine-tune a pre-trained model to make it fit better to a new data set. Since the underlying physico-chemical phenomena do not change with the catalyst change, the available LSTM is a good pre-trained model that captures the critical dynamic relations from the previous training.

Specifically, the training of all layers in the pre-trained LSTM model except the output layer was frozen with the assumption that the ground truth physical relationship is captured in the LSTM layer. Subsequently, the model is trained with only the new experimental results. The model is trained with 10 epochs because it is common in transfer learning to train the model with a small number of epochs to prevent it from overfitting the new data, especially when the size of the new data set is small. Eventually, the retrained model preserved a stable behavior and predicted the new u_r to be -1.26 V and 650 RPM. This result matches our expectation for the new catalyst since the C_2H_4 productivity is more correlated to the applied potential while CO productivity is more correlated to the rotation speed. Based on the experiment observation, the new u_r should decrease the potential to reduce the productivity of C_2H_4 to better approach the set-point. However, reducing the potential will also reduce the productivity of CO even if it is more correlated to rotation speed. The rotation speed then needs to be increased slightly to compensate for the loss in CO productivity. Thus, we concluded that the transfer learning-based retrain process calibrates the LSTM model in the correct direction and moved on to using it to perform closed-loop MPC experiments of controlling the reactor with the new catalyst. The result of the closed-loop experiment is shown in Fig. 5.9 which demonstrates that the MPC with the retrained model can stabilize the reactor outputs to the desired set-points.

Remark 33. *In addition to the retraining method that corrects the model off-line, on-line correction may be implemented to improve the MPC performance using real-time measurements. For*

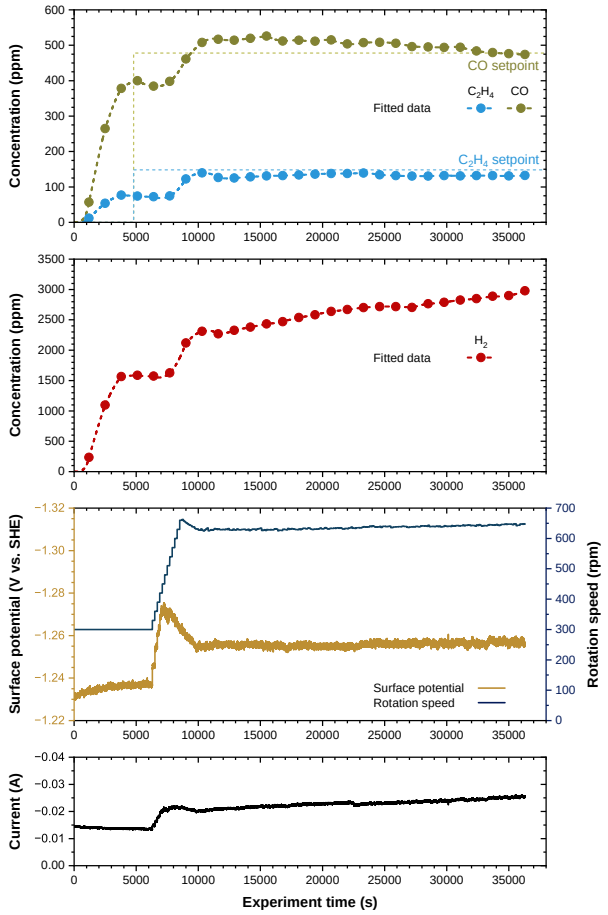


Figure 5.9: Output responses and control actions with new catalyst controlled by the MPC using the retrained model.

example, the extended Kalman filter method can be a good candidate to be considered for this task. To implement this method, the Kalman correction factor should be added to the LSTM model for real-time estimation. Furthermore, since the Koopman method can be applied to any nonlinear model, the overall workflow of the model linearization and MPC implementation does not need to be changed to include the Kalman filter correction. Developing this correction step for an MPC of an electrochemical reactor is one of our future objectives.

Remark 34. *The retraining correction requires collecting new data from the process, which may introduce a certain delay to update the MPC. Consider the case where the data shift problem is*

just detected, and the collected data is not enough to retrain the process model, the MPC should be deactivated and switched to a backup controller (e.g., classical proportional integral controller) to ensure that the process operates safely.

Chapter 6

Conclusions

This dissertation discussed the application of ML/AI technologies to develop an advanced MPC scheme implementing MIMO control to an experimental electrochemical reactor that could capture and reduce CO_2 to valuable chemical products. Prior to engineering the ML/AI-based process control system of the electrochemical reactor, various novel NN structures and algorithms were investigated to account for the effect of noisy data. Notably here, noisy data was not only referring to the data set that was corrupted by sensor or process noise, but also considered the irregularly sampled data, such as missing data points in a time-series data sequence. Mathematical evaluations were performed to understand how those novel NN approaches could be used to account for noisy data and to ensure they were suitable to be applied to chemical engineering applications. Subsequently, MPCs were developed using the novel noise-robust NN models and evaluated via simulations using dynamic process models and large-scale chemical process simulators (e.g., Aspen Plus and Aspen Dynamic). Next, a weighted-FNN model was developed based on the historical experiment data to predict the output states of the reactor at steady states. This weighted-

FNN model also provided insights towards facilitating the development of a first principles-based model to describe the physical-chemical phenomena of the reactor and can be used in the RTO to compute the optimum set point. Lastly, an LSTM model was developed to capture the dynamic behavior of the reactor. The Koopman operator method was adopted to linearize the LSTM model in real-time to reduce the computational cost, which made the MPC feasible to be applied in real-time operation.

In **Chapter 2**, LSTM models using the Monte Carlo dropout method and co-teaching technique were developed to predict underlying process dynamics (ground truth) from noisy data. A chemical reactor was modeled and simulated in a large-scale process simulator, Aspen Plus, to demonstrate the application of dropout and co-teaching methods with noisy and noise-free data generated from Aspen simulations and first principles model solutions, respectively. Then, the closed-loop simulation of the reactor was carried out in Aspen Plus Dynamics under an LSTM-based MPC to demonstrate the dropout and co-teaching LSTM methods were more robust to noisy data than the standard LSTM modeling approach in terms of open-loop prediction accuracy and closed-loop performance.

In **Chapter 3**, a Lyapunov-based MPC (LMPC) system using a neural ordinary differential equation (NODE) model, which was constructed from process data, was developed. The results demonstrated the application of the NODE as the process model in an LMPC. Specifically, the core model of the NODE could capture the time derivatives of the states, which could be considered an additional method to compute the derivative information other than numerical approximation methods. The state derivatives predicted by the NODE model were used in the LMPC to enforce the constraints and optimize the control objective. Moreover, the NODE model imposed

fewer restrictions on the structure of the training data than RNN models, which permitted the use of the subsampling method to account for noisy data. NODE models developed with different subsampling factors were used to account for Gaussian and non-Gaussian measurement noise. It was found that using subsampling could not significantly reduce the training loss under Gaussian noise, but could reduce the LMPC cost function in closed-loop simulations by up to 15%. For non-Gaussian noise, using subsampling reduced the training loss by up to 24% and 26% in the case of the weak and strong noises considered in this work, respectively. As for the closed-loop simulations, using subsampling improved the closed-loop performance of the LMPC under non-Gaussian noise by up to 34% in terms of the LMPC cost function.

In **Chapter 4**, the application of neural network modeling to an electrochemical reactor was demonstrated. Specifically, an FNN model was developed to model the experimental reactor data over a broad range of operating conditions. Additionally, weighted-FNN model, inspired by the maximum likelihood estimation method, was developed to account for the variability of experimental data, and its predictive performance was demonstrated over a broad range of operating conditions. Lastly, an algorithm was proposed to improve the empirical, first principles (EFP) model by utilizing the weighted-FNN insight, which decreased the MSE of the EFP model predictions for three reaction rates by 75%, 79%, and 32%, respectively.

In **Chapter 5**, a procedure to apply neural network model-based MPC to perform real-time multivariable control for an experimental electrochemical reactor was presented; the approach involved on-line linearization of the neural network model and was applicable to broad classes of chemical processes. Specifically, an LSTM neural network model was used to capture the nonlinear dynamic input-output relationship to control an electrochemical reactor. The Koopman opera-

tor method was found to be able to linearize the LSTM model efficiently (in terms of computational effort) and effectively (in terms of model performance). Based on that method, a systematic approach was developed to linearize a neural network model using the Scikit-learn linear regression function, which was efficient and easy to implement. Open-loop simulations were performed to evaluate the performance of the original LSTM and linearized LSTM models, and the MPC developed based on the linearization of the LSTM model was applied to control the experimental electrochemical reactor. As the closed-loop results demonstrated, the MPC calculated the optimal control actions with a reasonable computation cost and successfully drove the process outputs to desired set-point values. Furthermore, a transfer-learning scheme was introduced to account for the data shift problem (owing to catalyst activity variability every time a new catalyst was introduced) by updating the LSTM model using new process measurement data. The transfer-learning method was demonstrated to be able to update the original LSTM model with a limited amount of new data and computational cost. Finally, the updated LSTM model and the resulting MPC were demonstrated to resolve the data shift problem by driving the process outputs to the desired set-points in a closed-loop experiment under the new experimental (catalyst) conditions.

Bibliography

[1] D. Abdul, J. Wenqi, and A. Tanveer. Prioritization of renewable energy source for electricity generation through ahp-vikor integrated methodology. *Renewable Energy*, 184:1018–1032, 2022.

[2] F. Abdullah and P. D. Christofides. Data-based modeling and control of nonlinear process systems using sparse identification: An overview of recent results. *Computers & Chemical Engineering*, 174:108247, 2023.

[3] F. Abdullah, Z. Wu, and P. D. Christofides. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Computers & Chemical Engineering*, 153: 107411, 2021.

[4] F. Abdullah, 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*, 61:17976–17992, 2022.

[5] 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.

[6] A. Agarwal, Y. 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.

[7] M. S. Alhajeri, A. Alnajdi, F. Abdullah, and P. D. Christofides. On generalization error of neural network models and its application to predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 189:664–679, 2023.

[8] T.-Z. Ang, M. Salem, M. Kamarol, H. S. Das, M. A. Nazari, and N. Prabaharan. A comprehensive study of renewable energy sources: Classifications, challenges and suggestions. *Energy Strategy Reviews*, 43:100939, 2022.

[9] H. Arbabi and I. Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16:2096–2126, 2017.

[10] H. Arbabi, M. Korda, and I. Mezić. A data-driven koopman model predictive control framework for nonlinear partial differential equations. In *Proceedings of IEEE Conference on Decision and Control*, pages 6409–6414, Miami, FL, 2018.

[11] 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.

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

[13] M. Bangi and J. Kwon. Deep reinforcement learning control of hydraulic fracturing. *Computers and Chemical Engineering*, 154:107489, 2021.

[14] A. Bard and L. Faulkner. *Electrochemical Methods: Fundamentals and Applications*. John Wiley & Sons, New York, 2nd edition, 2001.

[15] S. E. Benattia, S. Tebbani, and D. Dumur. A linearized robust model predictive control applied to bioprocess. In *Proceedings of 55th Conference on Decision and Control*, pages 4046–4052, Las Vegas, Nevada, 2016.

[16] B. Bhadriraju, A. Narasingam, and J. S.-I. Kwon. Machine learning-based adaptive model identification of systems: Application to a chemical process. *Chemical Engineering Research and Design*, 152:372–383, 2019.

[17] S. A. Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[18] A. Boulamanti, J. A. Moya, et al. Energy efficiency and ghg emissions: Prospective scenarios for the chemical and petrochemical industry. *Report 9789279657344, EU Science Hub*, 2017.

[19] G. Box and N. Draper. *Empirical model-building and response surfaces.* John Wiley & Sons, New York, 1987.

[20] W. Bradley and F. Boukouvala. Two-stage approach to parameter estimation of differential equations using neural odes. *Industrial & Engineering Chemistry Research*, 60:16330–16344, 2021.

[21] T. Brüdigam, M. Olbrich, D. Wollherr, and M. Leibold. Stochastic model predictive control with a safety guarantee for automated driving. *IEEE Transactions on Intelligent Vehicles*, 2021.

[22] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS One*, 11: e0150171, 2016.

[23] Y. Cao, S. Chen, S. Bo, W. Fan, J. Li, C. Jia, Z. Zhou, Q. Liu, L. Zheng, and F. Zhang. Single Atom Bi Decorated Copper Alloy Enables C-C Coupling for Electrocatalytic Reduction of CO_2 into C_{2+} Products. *Angew. Chemie - Int. Ed.,* in press, 2023.

[24] S. Chavan, N. Birnale, and A. S. Deshpande. Design and simulation of model predictive control for multivariable distillation column. In *Proceedings of 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 764–768, 2018.

[25] K. Y. Chee, M. A. Hsieh, and N. Matni. Learning-enhanced nonlinear model predictive control using knowledge-based neural ordinary differential equations and deep ensembles. *arXiv preprint arXiv:2211.13829*, 2022.

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[27] 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.

[28] M. Cilimkovic. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15, 2015.

[29] B. Çıtmacı, J. Luo, J. B. Jang, V. Canuso, D. Richard, Y. M. Ren, C. G. Morales-Guio, and P. D. Christofides. Machine learning-based ethylene concentration estimation, real-time optimization and feedback control of an experimental electrochemical reactor. *Chemical Engineering Research and Design*, 185:87–107, 2022.

[30] B. Çıtmacı, J. Luo, J. B. Jang, P. Korambath, C. G. Morales-Guio, J. F. Davis, and P. D. Christofides. Digitalization of an experimental electrochemical reactor via the smart manufacturing innovation platform. *Digital Chemical Engineering*, 5:100050, 2022.

[31] B. Çıtmacı, J. Luo, J. B. Jang, C. G. Morales-Guio, and P. D. Christofides. Machine learning-based ethylene and carbon monoxide estimation, real-time optimization, and multivariable feedback control of an experimental electrochemical reactor. *Chemical Engineering Research and Design*, 191:658–681, 2023.

[32] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[33] W. Dai, Y. Song, and D. Wang. A subsampling method for regression problems based on minimum energy criterion. *Technometrics*, 65:192–205, 2023.

[34] R. De, S. Gonglach, S. Paul, M. Haas, S. Sreejith, P. Gerschel, U.-P. Apfel, T. H. Vuong, J. Rabeah, S. Roy, et al. Electrocatalytic reduction of CO_2 to acetic acid by a molecular manganese corrole complex. *Angewandte Chemie*, 132:10614–10621, 2020.

[35] P. De Luna, C. Hahn, D. Higgins, S. A. Jaffer, T. F. Jaramillo, and E. H. Sargent. What would it take for renewably powered electrosynthesis to displace petrochemical processes? *Science*, 364:eaav3506, 2019.

[36] 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.

[37] Y. Ding, Y. Zhang, Y. Ren, G. Orkoulas, and P. D. Christofides. Machine learning-based modeling and operation for ALD of SiO_2 thin-films using data from a multiscale CFD simulation. *Chemical Engineering Research and Design*, 151:131–145, 2019.

[38] M. R. Dobbelaere, P. P. Plehiers, R. Van de Vijver, C. V. Stevens, and K. M. Van Geem. Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering*, 7:1201–1211, 2021.

[39] A. Dongare, R. Kharde, A. D. Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology*, 2:189–194, 2012.

[40] S. Dorling, R. Foxall, D. Mandic, and G. Cawley. Maximum likelihood cost functions for neural network models of air quality data. *Atmospheric Environment*, 37:3435–3443, 2003.

[41] E. Dupont, A. Doucet, and Y. W. Teh. Augmented neural ODEs. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] R. M. Errico. What is an adjoint model? *Bulletin of the American Meteorological Society*, 78:2577–2592, 1997.

[43] L. Fan, Y. Zhao, L. Chen, J. Chen, J. Chen, H. Yang, Y. Xiao, T. Zhang, J. Chen, and L. Wang. Selective production of ethylene glycol at high rate via cascade catalysis. *Nat. Catal.*, 2023. ISSN 25201158.

[44] Z. Feng, W. Shen, G. P. Rangaiah, and L. Dong. Closed-loop identification and model predictive control of extractive dividing-wall column. *Chemical Engineering and Processing-Process Intensification*, 142:107552, 2019.

[45] M. Feurer and F. Hutter. Hyperparameter optimization. In *Automated machine learning: Methods, systems, challenges*, pages 3–33. Springer, 2019.

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

[47] 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.

[48] R. Galvelis and Y. Sugita. Neural network and nearest neighbor algorithms for enhancing sampling of molecular dynamics. *Journal of Chemical Theory and Computation*, 13:2489–2500, 2017.

[49] P. Giesl and S. Hafstein. Review on computational methods for lyapunov functions. *Discrete and Continuous Dynamical Systems-B*, 20:2291–2331, 2015.

[50] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[51] P. Goyal and P. Benner. Neural ODEs with irregular and noisy data. *arXiv preprint arXiv:2205.09479*, 2022.

[52] B. Grosman and D. R. Lewin. Automatic generation of lyapunov functions using genetic programming. *IFAC Proceedings Volumes*, 38:75–80, 2005.

[53] N. M. Haegel, R. Margolis, T. Buonassisi, D. Feldman, A. Froitzheim, R. Garabedian, M. Green, S. Glunz, H.-M. Henning, B. Holder, et al. Terawatt-scale photovoltaics: Trajectories and challenges. *Science*, 356:141–143, 2017.

[54] 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. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 8527–8537, Montreal, Canada, 2018.

[55] H.-G. Han, L. Zhang, Y. Hou, and J.-F. Qiao. Nonlinear model predictive control based on a self-organizing recurrent neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 27:402–415, 2015.

[56] L. Han, C. Yu, K. Xiao, and X. Zhao. A new method of mixed gas identification based on a convolutional neural network for time series classification. *Sensors*, 19:1960, 2019.

[57] C. D. Hansen and C. R. Johnson. *Visualization handbook*. Elsevier, Burlington, 2011.

[58] 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.

[59] L. Hewing, J. Kabzan, and M. N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28:2736–2743, 2019.

[60] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

[61] K. Holkar and L. M. Waghmare. An overview of model predictive control. *International Journal of Control and Automation*, 3:47–63, 2010.

[62] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982. ISSN 0027-8424.

[63] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[64] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

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

[66] Q. J. Huys and L. Paninski. Smoothing of, and parameter estimation from, noisy biophysical recordings. *PLoS Computational Biology*, 5:e1000379, 2009.

[67] V. Huzurbazar. The likelihood equation, consistency and the maxima of the likelihood function. *Annals of Eugenics*, 14:185–200, 1947.

[68] R. Inapakurthi, S. Miriyala, and K. Mitra. Recurrent neural networks based modelling of industrial grinding operation. *Chemical Engineering Science*, 219:115585, 2020.

[69] R. Inapakurthi, S. Miriyala, and K. Mitra. Deep learning based dynamic behavior modelling and prediction of particulate matter in air. *Chemical Engineering Journal*, 426:131221, 2021.

[70] M. Jadid and D. Fairbairn. Neural-network applications in predicting moment-curvature parameters from experimental data. *Engineering Applications of Artificial Intelligence*, 9: 309–319, 1996.

[71] J. Jang, M. Rüscher, M. Winzely, and C. G. Morales-Guio. Gastight rotating cylinder electrode: Toward decoupling mass transport and intrinsic kinetics in electrocatalysis. *AIChE Journal*, 68:e17605, 2022.

[72] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66:952–964, 2016.

[73] N. L. Jian, H. Zabiri, and M. Ramasamy. Control of the multi-timescale process using

multiple timescale recurrent neural network-based model predictive control. *Industrial & Engineering Chemistry Research*, 62:6176–6195, 2023.

[74] S. Johari, M. Yaghoobi, and H. R. Kobravi. Nonlinear model predictive control based on hyper chaotic diagonal recurrent neural network. *Journal of Central South University*, 29: 197–208, 2022.

[75] M. Jouny, W. Luc, and F. Jiao. General techno-economic analysis of CO_2electrolysis systems. *Industrial & Engineering Chemistry Research*, 57:2165–2177, 2018.

[76] P. Kadlec, R. Grbić, and B. Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & Chemical Engineering*, 35:1–24, 2011.

[77] I. Kamal and A. Malah. *Aspen Plus Chemical Engineering Applications*. John Wiley & Sons, Inc., 2017.

[78] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

[79] B. Kim, Y. C. Tan, Y. Ryu, K. Jang, H. G. Abbas, T. Kang, H. Choi, K.-s. Lee, S. Park, W. Kim, P.-p. Choi, S. Ringe, and J. Oh. Trace-Level Cobalt Dopants Enhance CO_2 Electroreduction and Ethylene Formation on Copper. *ACS Energy Lett.*, 8:3356–3364, 2023.

[80] G. Kimaev and L. A. Ricardez-Sandoval. Nonlinear model predictive control of a multiscale thin film deposition process using artificial neural networks. *Chemical Engineering Science*, 207:1230–1245, 2019.

[81] G. Kimaev and L. A. Ricardez-Sandoval. Artificial neural network discrimination for parameter estimation and optimal product design of thin films manufactured by chemical vapor deposition. *The Journal of Physical Chemistry C*, 124:18615–18627, 2020.

[82] P. Kittisupakorn, P. Thitiyasook, M. A. Hussain, and W. Daosud. Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19:579–590, 2009.

[83] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17:315–318, 1931.

[84] B. O. Koopman and J. V. Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18:255–263, 1932.

[85] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[86] J. Krishnaiah, C. Kumar, and M. 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.

[87] M. Kumar, D. Garg, and R. Zachery. Intelligent sensor modeling and data fusion via neural network and maximum likelihood estimation. *Proceedings of ASME International Mechanical Engineering Congress and Exposition*, 42169:1759–1768, 2005.

[88] Z. Lai, C. Mylonas, S. Nagarajaiah, and E. Chatzi. Structural identification with physics-

informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508: 116196, 2021.

[89] M. Ławryńczuk. *Computationally efficient model predictive control algorithms: A Neural Network Approach*, volume 3. Studies in Systems, Decision and Control Series, Springer, Warsaw, 2014.

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

[91] J. H. Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9:415–424, 2011.

[92] K. Lee and E. J. Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A*, 477:20210162, 2021.

[93] M. Li, H. Wang, W. Luo, P. C. Sherrell, J. Chen, and J. Yang. Heterogeneous single-atom catalysts for electrochemical CO_2 reduction reaction. *Advanced Materials*, 32:2001848, 2020.

[94] Z. Li, X. Hong, K. Hao, L. Chen, and B. Huang. Gaussian process regression with heteroscedastic noises–a machine-learning predictive variance approach. *Chemical Engineering Research and Design*, 157:162–173, 2020.

[95] R. Liao, C. W. Chan, J. Hromek, G. H. Huang, and L. He. Fuzzy logic control for a

petroleum separation process. *Engineering Applications of Artificial Intelligence*, 21:835–845, 2008.

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

[97] 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.

[98] M. Lin, J. G. Carlsson, D. Ge, J. Shi, and J. Tsai. A review of piecewise linearization methods. *Mathematical Problems in Engineering*, 2013:101376, 2013.

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

[100] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C. Hsieh. Neural SDE: Stabilizing neural ODE networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

[101] J. Lötsch, S. Malkusch, and A. Ultsch. Optimal distribution-preserving downsampling of large biomedical data sets (opdisdownsampling). *Plos One*, 16:e0255838, 2021.

[102] Y. Ma, D. Noreña-Caro, A. Adams, T. Brentzel, J. Romagnoli, and M. 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.

[103] A. Malek, Q. Wang, S. Baumann, O. Guillon, M. Eikerling, and K. Malek. A data-driven

framework for the accelerated discovery of CO_2 reduction electrocatalysts. *Frontiers in Energy Research*, 9:52, 2021.

[104] P. Mendis, C. Wickramasinghe, M. Narayana, and C. Bayer. Adaptive model predictive control with successive linearization for distillate composition control in batch distillation. In *Proceedings of 2019 Moratuwa Engineering Research Conference (MERCon)*, pages 366–369, 2019.

[105] D. F. Mendoza, L. M. Palacio, J. E. Graciano, C. A. Riascos, A. S. Vianna Jr, and G. C. Le Roux. Real-time optimization of an industrial-scale vapor recompression distillation process. model validation and analysis. *Industrial & Engineering Chemistry Research*, 52: 5735–5746, 2013.

[106] S. Miriyala, P. Mittal, S. Majumdar, and K. Mitra. Comparative study of surrogate approaches while optimizing computationally expensive reaction networks. *Chemical Engineering Science*, 140:44–61, 2016.

[107] S. Mohanty. Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19:991–999, 2009.

[108] C. G. Morales-Guio, E. Cave, S. Nitopi, J. Feaster, L. Wang, K. Kuhl, A. Jackson, N. Johnson, D. Abram, T. Hatsukade, and et al. Improved CO_2 reduction activity towards C_{2+} alcohols on a tandem gold on copper electrocatalyst. *Nature Catalysis*, 1:764–771, 2018.

[109] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23:667–682, 1999.

[110] I. Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47:90–100, 2003.

[111] S. Natarajan and J. Behler. Neural network molecular dynamics simulations of solid–liquid interfaces: water at low-index copper surfaces. *Physical Chemistry Chemical Physics*, 18: 28704–28725, 2016.

[112] R. Nian, J. Liu, and B. Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.

[113] Y. Nishimura, H. Peng, S. Nitopi, M. Bajdich, L. Wang, C. Morales-Guio, F. Abild-Pedersen, T. Jaramillo, and C. Hahn. Guiding the catalytic properties of copper for electrochemical CO_2 reduction by metal atom decoration. *ACS Applied Materials & Interfaces*, 13:52044–52054, 2021.

[114] S. Nitopi, E. Bertheussen, S. Scott, X. Liu, A. Engstfeld, S. Horch, B. Seger, I. Stephens, K. Chan, C. Hahn, and et al. Progress and perspectives of electrochemical CO_2 reduction on copper in aqueous electrolyte. *Chemical Reviews*, 119:7610–7672, 2019.

[115] F. Núñez, S. Langarica, P. Díaz, M. Torres, and J. C. Salas. Neural network-based model predictive control of a paste thickener over an industrial internet platform. *IEEE Transactions on Industrial Informatics*, 16:2859–2867, 2019.

[116] P. Osofisan and O. Obafaiye. Fuzzy logic modeling of the fluidized catalytic cracking unit of a petrochemical refinery. *The Pacific Journal of Science and Technology*, 8:59–67, 2007.

[117] O. Owoyele and P. Pal. Chemnode: A neural ordinary differential equations framework for efficient chemical kinetic solvers. *Energy and AI*, 7:100118, 2022.

[118] A. Ozden, Y. Wang, F. Li, M. Luo, J. Sisler, A. Thevenon, A. Rosas-Hernández, T. Burdyny, Y. Lum, H. Yadegari, T. Agapie, J. C. Peters, E. H. Sargent, and D. Sinton. Cascade CO_2 electroreduction enables efficient carbonate-free production of ethylene. *Joule*, 5:706–719, 2021.

[119] 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.

[120] L. S. Pontryagin. *Mathematical theory of optimal processes*. CRC press, New York, 1987.

[121] S. Popović, M. Smiljanić, P. Jovanovič, J. Vavra, R. Buonsanti, and N. Hodnik. Stability and degradation mechanisms of copper-based catalysts for electrochemical CO_2 reduction. *Angewandte Chemie*, 132:14844–14854, 2020.

[122] A. Pozzi, M. Zambelli, A. Ferrara, and D. M. Raimondo. Balancing-aware charging strategy for series-connected lithium-ion cells: A nonlinear model predictive control approach. *IEEE Transactions on Control Systems Technology*, 28:1862–1877, 2020.

[123] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15:142–161, 2016.

[124] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Generalizing Koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17:909–930, 2018.

[125] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[126] P. Raccuglia, K. Elbert, P. Adler, C. Falk, M. Wenny, A. Mollo, M. Zeller, S. Friedler, J. Schrier, and A. Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533:73–76, 2016.

[127] D. Raciti and C. Wang. Recent advances in CO_2 reduction electrocatalysis on copper. *ACS Energy Letters*, 3:1545–1556, 2018.

[128] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker. A predictive controller for autonomous vehicle path tracking. *IEEE Transactions on Intelligent Transportation Systems*, 10:92–102, 2009.

[129] M. Ramdin, B. De Mot, A. R. Morrison, T. Breugelmans, L. J. Van Den Broeke, J. P. Trusler, R. Kortlever, W. De Jong, O. A. Moultos, P. Xiao, P. A. Webley, and T. J. Vlugt. Electroreduction of CO_2/CO to C_2 Products: Process Modeling, Downstream Separation, System Integration, and Economic Analysis. *Ind. Eng. Chem. Res.*, 60:17862–17880, 2021. ISSN 15205045.

[130] M. Ramdin, O. A. Moultos, L. J. van den Broeke, P. Gonugunta, P. Taheri, and T. J. Vlugt. Carbonation in Low-Temperature CO_2 Electrolyzers: Causes, Consequences, and Solutions. *Ind. Eng. Chem. Res.*, 62:6843–6864, 2023. ISSN 15205045.

[131] D. Rangel-Martinez, K. Nigam, and L. A. Ricardez-Sandoval. Machine learning on sustain-

able energy: A review and outlook on renewable energy systems, catalysis, smart grid and energy storage. *Chemical Engineering Research and Design*, 174:414–441, 2021.

[132] Y. M. Ren, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, page 107956, 2022.

[133] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29:1251–1274, 1993.

[134] D. Richard, J. Jang, B. Çıtmacı, J. Luo, V. Canuso, P. Korambath, O. Morales-Leslie, J. F. Davis, H. Malkani, P. D. Christofides, et al. Smart manufacturing inspired approach to research, development, and scale-up of electrified chemical manufacturing systems. *Iscience*, 26, 2023.

[135] F. S. Roberts, K. P. Kuhl, and A. Nilsson. High Selectivity for Ethylene from Carbon Dioxide Reduction over Copper Nanocube Electrocatalysts. *Angew. Chemie*, 127:5268–5271, 2015.

[136] S. Rohani, M. Haeri, and H. Wood. Modeling and control of a continuous crystallization process part 2. model predictive control. *Computers & Chemical Engineering*, 23:279–286, 1999.

[137] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[138] Y. Rubanova, R. T. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing systems*, 32, 2019.

[139] E. Ruiz-López, J. Gandara-Loe, F. Baena-Moreno, T. R. Reina, and J. A. Odriozola. Electrocatalytic CO_2 conversion to c_2 products: Catalysts design, market perspectives and techno-economic aspects. *Renewable and Sustainable Energy Reviews*, 161:112329, 2022.

[140] M. R. Sarmasti Emami. Fuzzy logic applications in chemical processes. *J. Math. Comput. Sci*, 1:339–348, 2019.

[141] T. S. Schei. A finite-difference method for linearization in nonlinear estimation algorithms. *Automatica*, 33:2053–2058, 1997.

[142] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[143] M. Schwenzer, M. Ay, T. Bergs, and D. Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117:1327–1349, 2021.

[144] 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.

[145] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104:148–175, 2015.

[146] M. Sharifzadeh. Integration of process design and control: A review. *Chemical Engineering Research and Design*, 91:2515–2549, 2013.

[147] Q. Shen, B. Jiang, P. Shi, and C. C. Lim. Novel neural networks-based fault tolerant control scheme with fault alarm. *IEEE Transactions on Cybernetics*, 44:2190–2201, 2014.

[148] Y. Shin, R. Smith, and S. Hwang. Development of model predictive control system using an artificial neural network: A case study with a distillation column. *Journal of Cleaner Production*, 277:124124, 2020.

[149] B. Singh, P. Sihag, and K. Singh. Modelling of impact of water quality on infiltration rate of soil by random forest regression. *Modeling Earth Systems and Environment*, 3:999–1004, 2017.

[150] 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.

[151] I. Sullivan, A. Goryachev, I. A. Digdaya, X. Li, H. A. Atwater, D. A. Vermaas, and C. Xiang. Coupling electrochemical CO_2 conversion with CO_2 capture. *Nature Catalysis*, 4:952–958, 2021.

[152] M. Surtsukov. Neural ODEs. https://github.com/msurtsukov/neural-ode, 2019.

[153] M. Tom, S. Yun, H. Wang, F. Ou, G. Orkoulas, and P. D. Christofides. Machine learning-based run-to-run control of a spatial thermal atomic layer etching reactor. *Computers & Chemical Engineering*, 168:108044, 2022.

[154] G. Tran and R. Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15:1108–1129, 2017.

[155] J. H. Tu. *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.

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

[157] 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.

[158] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066. PMLR, 2013.

[159] X. Wang, P. Ou, A. Ozden, S.-F. Hung, J. Tam, C. M. Gabardo, J. Y. Howe, J. Sisler, K. Bertens, F. P. García de Arquer, et al. Efficient electrosynthesis of n-propanol from carbon monoxide using a ag–ru–cu catalyst. *Nature Energy*, 7:170–176, 2022.

[160] Y. Wang, Z. Wang, C.-T. Dinh, J. Li, A. Ozden, M. Golam Kibria, A. Seifitokaldani, C.-S. Tan, C. M. Gabardo, M. Luo, et al. Catalyst synthesis under CO_2 electroreduction favours faceting and promotes renewable fuels electrosynthesis. *Nature Catalysis*, 3:98–106, 2020.

[161] P. Whittle. Tests of fit in time series. *Biometrika*, 39:309–318, 1952.

[162] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

[163] R. H. Wiser and M. Bolinger. 2018 wind technologies market report. 2019.

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

[165] J. Wu, X. Chen, H. Zhang, L. Xiong, H. Lei, and S. Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17:26–40, 2019.

[166] 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.

[167] 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.

[168] 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.

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

[170] 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.

[171] X.-C. Xi, A.-N. Poo, and S.-K. Chou. Support vector regression model predictive control on a hvac plant. *Control Engineering Practice*, 15:897–908, 2007.

[172] R. Xia, S. Overa, and F. Jiao. Emerging electrochemical processes to decarbonize the chemical industry. *JACS Au*, 2:1054–1070, 2022.

[173] T. Xiao, Z. Wu, P. D. Christofides, A. Armaou, and D. Ni. Recurrent neural-network-based model predictive control of a plasma etch process. *Industrial & Engineering Chemistry Research*, 61:638–652, 2021.

[174] K. Xie, A. Ozden, R. K. Miao, Y. Li, D. Sinton, and E. H. Sargent. Eliminating the need for anodic gas separation in CO_2 electroreduction systems via liquid-to-liquid anodic upgrading. *Nat. Commun.*, 13:1–9, 2022.

[175] S. Xie and J. Ren. Linearization of recurrent-neural-network-based models for predictive control of nano-positioning systems using data-driven Koopman operators. *IEEE Access*, 8: 147077–147088, 2020.

[176] W. Xie, I. Bonis, and C. Theodoropoulos. Data-driven model reduction-based nonlinear mpc for large-scale distributed parameter systems. *Journal of Process Control*, 35:50–58, 2015.

[177] S. Yaacob, R. Nagarajan, and K. T. T. Kin. Application of predictive fuzzy logic controller in temperature control of phenol-formaldehyde manufacturing: using matlab-simulink

methodology. In *Intelligent Systems in Design and Manufacturing IV*, volume 4565, pages 101–109, 2001.

[178] F. Yang, K. Li, Z. Zhong, Z. Luo, X. Sun, H. Cheng, X. Guo, F. Huang, R. Ji, and S. Li. Asymmetric co-teaching for unsupervised cross-domain person re-identification. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 12597–12604, New York City, New York, 2020.

[179] S. Yang and M. P. Wan. Machine-learning-based model predictive control with instantaneous linearization–a case study on an air-conditioning and mechanical ventilation system. *Applied Energy*, 306:118041, 2022.

[180] K. Yeo. Short note on the behavior of recurrent neural network for noisy dynamical system. *arXiv preprint arXiv:1904.05158*, 2019.

[181] S. Yun, M. Tom, J. Luo, G. Orkoulas, and P. D. Christofides. Microscopic and data-driven modeling and operation of thermal atomic layer etching of aluminum oxide thin films. *Chemical Engineering Research & Design*, 177:96–107, 2022.

[182] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

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

[184] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[185] S. Zhang, P. Ocłoń, J. J. Klemeš, P. Michorczyk, K. Pielichowska, and K. Pielichowski. Renewable energy systems for building heating, cooling and electricity production with thermal energy storage. *Renewable and Sustainable Energy Reviews*, 165:112560, 2022.

[186] T. Zhang, S. Li, and Y. Zheng. Implementable stability guaranteed lyapunov-based data-driven model predictive control with evolving gaussian process. *Industrial & Engineering Chemistry Research*, 61:14681–14690, 2022.

[187] Z. Zhang, E. W. Lees, S. Ren, B. A. Mowbray, A. Huang, and C. P. Berlinguette. Conversion of Reactive Carbon Solutions into CO at Low Voltage and High Carbon Efficiency. *ACS Cent. Sci.*, 8:749–755, 2022. ISSN 23747951.

[188] Y. Zheng, A. Vasileff, X. Zhou, Y. Jiao, M. Jaroniec, and S.-Z. Qiao. Understanding the roadmap for electrochemical reduction of CO_2 to multi-carbon oxygenates and hydrocarbons on copper-based catalysts. *Journal of the American Chemical Society*, 141:7646–7659, 2019.

[189] Y. Zheng, T. Zhang, S. Li, G. Zhang, and Y. Wang. Gp-based mpc with updating tube for safety control of unknown system. *Digital Chemical Engineering*, 4:100041, 2022.