UNIVERSITY OF CALIFORNIA

Los Angeles

A Machine Learning-Based Approach to Cybersecurity and Safety of Model Predictive Control Systems

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Chemical Engineering

by

Siyao Chen

2022

ABSTRACT OF THE DISSERTATION

A Machine Learning-Based Approach to Cybersecurity and Safety of Model Predictive Control Systems

by

Siyao Chen

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2022

Professor Panagiotis D. Christofides, Chair

Automated real-time operations of industrial process control systems depend heavily on accurate information and reliable communication of decision variables, state variables, and measurement data. Augmentation in sensor information and network-based communication increases the complexity of the control problem in terms of modeling structure and accuracy, process safety, as well as vulnerability to cybersecurity threats. Production plants collect a large amount of operational and instrumentation data to be used for monitoring, control, and troubleshooting. The potential application of these data goes beyond preventative maintenance and fault detection, especially with increased digital connectivity and increased computing power. With the rise of big data, researchers are equipped to explore more robust systems that will improve production and computation efficiency, operational process safety as well as cybersecurity in many industrial applications. As various machine-learning algorithms have demonstrated success in a wide range of engineering applications, the development of rigorous and systematic integration of nonlinear process control and machine-learning methods has become the focus of this research.

Large-scale industrial processes face many control and operating challenges such as high dimensionality, information structure constraints, complex interacting process dynamics, and

uncertainties in the system. To this end, the need of accounting for multivariable interactions and input/state constraints has motivated the development of model predictive control (MPC), and subsequently highlights the need for a process model that describes the process dynamics accurately. More specifically, distributed and decentralized control structures demonstrate better computational efficiency compared to the centralized control framework. On the other hand, traditional approaches to process safety through process design considerations and hazard analysis are independent from the design of control systems. MPC provides a framework to account for process operational safety constraints, and is able to provide simultaneous closed-loop stability and safety. Furthermore, cybersecurity has become another leading cause of process safety incidents, and is becoming increasingly important in chemical process industries as cyber-attacks have grown in sophistication and frequency. As intelligent cyber-attacks have access to control system information, researchers are motivated to develop cyber-attack detection and resilient operation strategies to address cyber-security issues beyond fault diagnosis.

This dissertation presents the use of machine learning techniques in MPC, and provides various methods of designing MPC systems for improved cyber-security and operational safety for nonlinear chemical processes. Integrated detect-control architectures for Lyapunov-based MPC, economic MPC, and distributed and decentralized MPC systems are presented to address several types of intelligent cyber-attacks with machine-learning-based detection algorithms and resilient control and mitigation strategies. Data-driven nonlinear dynamic models are developed for large-scale processes consisting of multiple subsystems, and are used as the predictive model in distributed and decentralized MPC systems. Distributed MPC systems designed with control Lyapunov-barrier functions (CLBF) to guarantee closed-loop stability and safety properties are presented, and machine-learning methods to characterize the barrier function used in a CLBF-MPC are developed with statistical stability and safety analyses. Nonlinear chemical process examples are numerically simulated to demonstrate the effectiveness and performance of the proposed control methods throughout the dissertation.

The dissertation of Siyao Chen is approved.

Philippe Sautet

James F. Davis

Elisa Franco

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2022

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Panagiotis D. Christofides, for his support, encouragement, and guidance on both my technical work as well as professional development over the past four years. Professor Christofides sets an example of excellence as a researcher, mentor, instructor, and role model. I am lucky to be a student of his, as this Ph.D. experience has laid out a strong foundation for my future career. His mentoring on control theory, process/systems engineering, and life in general has made a profound impact on both my character and my career goals.

I am also grateful for my family and friends – for their encouragement, love, and support throughout my graduate career and throughout my life. In particular, I would like to thank my parents, Ken Chen and Aiping He, my grandparents, and my fiancé Jinming Wang. They have always believed in me, and it is their unconditional and unwavering support and affirmation that give me the strength and motivation to overcome the next challenge in life.

In addition, I would like to thank all of my colleagues with whom I have worked over the years in the Christofides research group, including Dr. Zhihao Zhang, Dr. Yangyao Ding, Dr. Yichi Zhang, Yi Ming Ren, Mohammed Alhajeri, Michael Park, Fahim Abdullah, Aisha Alnajdi, Junwei Luo, Sungil Yun, Matthew Tom, Vito Canuso, Berkay Citmaci. I would particularly like to thank Professor Zhe Wu, Dr. David Rincon, with whom I have collaborated extensively and spent long hours working on papers together.

I would also like to thank Professor Philippe Sautet, Professor James Davis, and Professor Elisa Franco for serving on my doctoral committee.

Financial support from UCLA Graduate Division Fellowship, UCLA Doctoral Dissertation Year Fellowship, the US Department of Energy (DOE) and the US National Science Foundation (NSF) is gratefully acknowledged, and my work could not have been done without this support.

With regard to the research that forms the foundation for this work:

Chapter 2 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "A Cyber-secure Control-Detector Architecture for Nonlinear Processes," *AIChE J.*, 66, e16907, 2020.

Chapter 3 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "Cyber-attack Detection and Resilient Operation of Nonlinear Processes under Economic Model Predictive Control," *Comp. & Chem. Eng.*, 136, 106806, 2020.

Chapter 4 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "Cyber-Security of Centralized, Decentralized, and Distributed Control-Detector Architectures for Nonlinear Processes," *Chem. Eng. Res. & Des.*, 165, 25-39, 2021.

Chapter 5.1 and 5.2 contain versions of: Chen, S., Wu, Z., Rincon, D. , and Christofides, P. D., "Machine Learning-Based Distributed Model Predictive Control of Nonlinear Processes," *AIChE J.*, 66, e17013, 2020; Chen, S., Wu, Z., and Christofides, P. D., "Decentralized Machine Learning-Based Predictive Control of Nonlinear Processes," *Chem. Eng. Res. & Des.*, 162, 45-60, 2020.

Chapter 6 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "Machine-learning-based construction of barrier functions and models for safe model predictive control," *AIChE J.*, e17456, 2021.

Chapter 7 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "Barrier-Function-Based Distributed Predictive Control for Operational Safety of Nonlinear Processes," *Comp. & Chem. Eng.*, 159, 107690, 2022.

Chapter 8 contains versions of: Chen, S., Wu, Z., and Christofides, P. D., "Statistical Machine-Learning-based Predictive Control Using Barrier Functions for Process Operational Safety," *Comp. & Chem. Eng.*, Submitted, 2022.

Curriculum Vitae

Education

University of Alberta **Sep. 2013 - June 2018**

B.S., Chemical Engineering Computer Process Control Co-op **Edmonton, Canada**

Journal Publications

1. **Chen, S.**, Wu, Z., and Christofides, P. D., "Statistical Machine-Learning-based Predictive Control Using Barrier Functions for Process Operational Safety," *Comp. & Chem. Eng.*, Submitted, 2022.

2. **Chen, S.**, Wu, Z., and Christofides, P. D., "Barrier-Function-Based Distributed Predictive Control for Operational Safety of Nonlinear Processes," *Comp. & Chem. Eng.*, 159, 107690, 2022.

3. **Chen, S.**, Wu, Z., and Christofides, P. D., "Machine-learning-based construction of barrier functions and models for safe model predictive control," *AIChE J.*, e17456, 2021.

4. **Chen, S.**, Wu, Z., and Christofides, P. D., "Cyber-Security of Centralized, Decentralized, and Distributed Control-Detector Architectures for Nonlinear Processes," *Chem. Eng. Res. & Des.*, 165, 25-39, 2021.

5. **Chen, S.**, Wu, Z., Rincon, D. , and Christofides, P. D., "Machine Learning-Based Distributed Model Predictive Control of Nonlinear Processes," *AIChE J.*, 66, e17013, 2020.

6. **Chen, S.**, Wu, Z., and Christofides, P. D., "Decentralized Machine Learning-Based Predictive Control of Nonlinear Processes," *Chem. Eng. Res. & Des.*, 162, 45-60, 2020.

7. **Chen, S.**, Wu, Z., and Christofides, P. D., "Cyber-attack Detection and Resilient Operation of Nonlinear Processes under Economic Model Predictive Control," *Comp. & Chem. Eng.*, 136, 106806, 2020.

8. **Chen, S.**, Wu, Z., and Christofides, P. D., "A Cyber-secure Control-Detector Architecture for Nonlinear Processes," *AIChE J.*, 66, e16907, 2020.

9. Wu, Z., **Chen, S.**, Rincon, D., and Christofides, P. D., "Post Cyber-Attack State Reconstruction for Nonlinear Processes Using Machine Learning," *Chem. Eng. Res. & Des.*, 159, 248-261, 2020.

10. **Chen, S.**, Kumar, A., Wong, W. C., Chiu, M. S. , and Wang, X., "Hydrogen value chain and fuel cells within hybrid renewable energy systems: Advanced operation and control strategies," *Applied Energy*, 233, 321-337, 2019.

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

Chapter 1

Introduction

1.1 Motivation

The ubiquitous deployment of sensors, the expansion of wireless communication, as well as the improved and cheaper computing power all contribute to the leap towards Industry 4.0. An explosive growth of data and computing power has been witnessed in the last decade that allows the development of big data analytics and intelligent machines [32, 111, 133]. Production plants collect and archive huge operational and instrumentation data used for monitoring, control, and troubleshooting. The potential application of these data goes beyond preventative maintenance and fault detection, especially with increased digital connectivity and increased computing power. This new digital revolution allows researchers to explore more robust systems that will improve operational stability, process safety, production quality, computation speed, economic gain, as well as cybersecurity in many industrial applications. In the following paragraphs, some critical challenges in the field of process control that motivate the research work presented in this dissertation will be outlined.

Stable and secure operation of cyber-physical systems require accurate information and reliable communication technologies. In recent years, the cyber-security of cyber-physical systems has become increasingly important as more communication networks are replaced or complemented

by wireless networks in addition to point-to-point communications [2, 26]. While these new developments increase operation efficiency and performance, they also increase the system's vulnerability to cyber-attacks. One example use of these "big data" is anomaly detection, including the detection of cyber-attacks and other abnormal process behavior. Due to the close interactions between cyber and physical components, operational cyber-security of control systems would mandate a different strategy than traditional information technology (IT) approaches – one that combines robust control strategies with an advanced detection scheme using the process data at hand. Stealthy, intelligent cyber-attack diagnosis and defense span a much broader scope than classical fault detection problems because intelligent adversaries can modify the actuator, the sensor, or the control implementation using process and control system information. With knowledge of the plant model and control formulation, cyber-attacks are strategically programmed with the goal of disruption, and are fundamentally different from ordinary sensor and actuator faults. Furthermore, the effects of these attacks may only be observed in changes of the dynamic behavior (runtime variables) of the closed-loop system; thus, using hardware performance counters to track code modifications is not feasible [61]. While conventional detection methods have demonstrated their effectiveness in detecting suspicious process variable deviations, most of these methods are model-based – either dependent on network and computer system models, or on physical process models. Certain classes of intelligent cyber-attacks either render traditional detection methods ineffective, or remain undetected until the system experiences a significant deviation and reaches an undesirable operating point, at which the existing alarm systems could be triggered. The goal of a robust cyber-attack detector is to identify attacks from subtle variations in real-time process state measurements and mitigate the risk before an operation alarm is triggered. Therefore, without explicit knowledge on the process model, adopting a data-based detection approach utilizing machine-learning algorithms provides a promising path for the detection of unknown intelligent cyber-attacks. The integration of existing advanced control techniques (e.g., Model Predictive Control (MPC)) and online machine-learning-based detection algorithms adds another protective safeguard to the multi-layer cyber-defense strategy

that is standard to next-generation smart manufacturing. Machine learning techniques, such as support vector machines [115], as well as more advanced deep learning methods, such as recurrent neural networks [50, 94], have demonstrated effectiveness in plant anomaly detection [12, 91, 97]. Motivated by this, machine-learning methodologies can be readily adopted in the context of control theory and cyber-physical security. In addition to having an adequate detection mechanism, control and operation strategies can be designed or adjusted accordingly if a process is vulnerable to cyber-attacks.

Another control challenge faced by many applications involves the scale and complexity of the process. Most existing methods for analyzing and controlling dynamical systems hinge on the common assumption of centrality, meaning that all the information available about the system is collected and received by a single controller to perform relevant calculations. Many industrial systems, such as chemical production plants, power distribution grids, urban traffic networks, and cyber-physical facilities such as data centers, are considered large-scale systems in which the assumption of centrality cannot hold due to the absence of a centralized information-gathering platform and the lack of centralized computing capabilities. Challenges such as high dimensionality of the system, uncertainties and delays in the communication network, and geographical separation of components, combined with the rapid development of microprocessor technologies, are factors that push for the shift from centralized control to decentralized decision-making and distributed computations [11, 27]. Amongst many advanced control techniques for large-scale systems, MPC is well known for its ability to handle multi-variable control problems with constraints. Centralized MPC is generally unsuited for the control of large-scale networked systems because of difficulties associated with scalability as well as the maintenance of global models [11]. Therefore, the formulation of decentralized and distributed MPC algorithms has naturally emerged to address these challenges [10, 27]. The original optimization problem in the centralized controller is broken down into a number of smaller optimization problems, which are solved in separate decentralized or distributed local controllers. Thus, decentralized and distributed control structures provide a practical solution

to decoupling large-scale processes and reducing the computational complexity of a centralized control problem by having multiple MPCs that work iteratively and cooperatively to achieve a common control objective applicable to the entire system [23, 71]. The advancement in computing and communication technologies has provided an adequate platform for networked control systems to handle the control problem of large-scale, complex processes in a distributed or decentralized structure. However, the existence of networked communication also implies inherent vulnerabilities to cyber-security risks. To this end, cyber-security threats on centralized, decentralized, and distributed control systems also need to be assessed and addressed with proper detection mechanisms.

For many large-scale industrial processes and/or novel processes, developing an accurate and comprehensive model that captures the dynamic behavior of the system can be difficult. Even in the case where a deterministic first-principles model is developed based on fundamental understandings, there may be inherent simplifying assumptions involved. Furthermore, during process operation, the model that is employed in model-based control systems to predict the future evolution of the system state may not always remain accurate as time progresses due to unforeseen process changes or large disturbances, causing plant model mismatch that degrades the performance of the control algorithm [39]. Given these considerations, the model identification of a nonlinear process is crucial for safe and robust model-based feedback control. Nonlinear system identification and an efficient approach to such is a crucial step to designing nonlinear model predictive controllers. While there is no systematic way to parameterize general nonlinear dynamic systems, among existing techniques, the universal approximation properties of neural networks make them a powerful method for modeling nonlinear systems using data [40]. There are many different structures of neural networks, such as feedforward neural networks (FNN), convolutional neural networks (CNN), and recurrent neural networks (RNN). RNN has proven its effectiveness in representing temporal dynamic behaviors of sequential data by taking advantage of feedback signals stored in its network units [40,57]. Many classes and variants of RNNs have been proposed and utilized in various applications, such as model identification, speech recognition,

signal processing, and intrusion detection. The integration of machine-learning-based modeling methods and various advanced control architectures is a broad field with expanding research scope. Using state-of-the-art machine-learning methods to address the issues of model uncertainties in a distributed and decentralized control structure is also a research topic addressed in this dissertation.

Moreover, the improvement of process operational safety has also been a long-standing research problem in the field of optimal control of dynamic processes. The severity of the potential hazards involved and the close interaction between human lives and the environment make safety a top priority in any industrial plant operation. The catastrophic outcomes of these incidents alarm us of the importance of maintaining, designing, and implementing stricter and more robust process and operational safety measures [30]. One way to prevent safety incidents is by designing a comprehensive and robust process control system that not only maintains stable production and economic optimality, but also handles unexpected production scenarios that could lead to unsafe operating conditions and environmental hazards. In addition to configuring alarming thresholds on process variables, the interactions between multiple process variables in a large-scale complex plant and their impact on the operational safety of the system should also be considered. Hence, MPC has been proposed as an advanced control methodology to account for multi-variable interactions, variable and safety constraints, and nonlinearities in industrial plants. Machine-learning-based methods can aid in the formulation of control algorithms that are incorporated in the design of MPC systems that will be able to ensure operational safety while achieving desired control performance. This approach can be extended to large-scale processes with multiple sub-systems under a distributed or decentralized control system. Lastly, rigorous statistical analysis is provided for the machine-learning-based methods, which provides an upper bound on the generalization performance of the machine-learning-based MPC designs in guaranteeing closed-loop stability and safety.

1.2 Background

Recent incidents of cyber-attacks on various industrial plants, such as the Iranian nuclear plant attack in 2010 and the Ukrainian electric power grid attack in 2015, demonstrated the capability of cyber-attacks in infiltrating cyber-physical systems (CPS) and the severity of their consequences. Malicious cyber-attacks could target any device or communication channels in the control network to modify control actions and jeopardize operational cost, stability, integrity, and other safety considerations. With access to technical details of the control system, these targeted cyber-attacks are intelligently designed to disrupt process operation and compromise fundamental process safety. As cyber-attacks pose severe threats to the control system, safety measures addressing cyber-security need to be carefully considered and incorporated in plant-wide risk assessments. Cyber-attacks can target actuators, sensors, communication channels between devices, and the control system algorithms; they modify the control implementation using process and control system information in an attempt to disrupt closed-loop performances. A comprehensive review in [9] included analysis on security issues, requirements, and possible solutions at various layers of the CPS architecture. Moreover, a survey on cyber-physical systems security from the security perspective (taxonomy of threats, attacks, and controls), the cyber-physical components perspective, and from a holistic systems perspective was explored in [54], where representative systems such as smart grids, medical CPS, and smart cars were studied. A review of possible weaknesses in corporate networks, in the Supervisory Control and Data Acquisition (SCADA) and Distributed Control System (DCS) systems, and in production environments was presented in [8]. Amongst sensor cyber-attacks, some common attack types are denial-of-service attacks, replay attacks, and deception attacks – such as min-max, geometric, and surge attacks [16]. For instance, for replay attacks commonly occurring on wireless sensor networks, a wormhole tunnel can be created between two end points to replay messages observed in different regions [65]. The detection and control of replay attacks in smart grid systems specifically have also been studied in [107, 140]. Moreover, adversaries may launch other deception attacks through hacking remote terminal units (RTUs), such as sensors in substations in a power grid transmission system. In [6],

a hierarchical attack in automated canal systems was described with various deception attacks in different cyber layers, and a field-operational test attack was reported on the Gignac canal system located in Southern France. Due to the sophistication of cyber-attacks and their accessibility to control system information, they are intended to disrupt the closed-loop system while avoiding being detected by conventional detection methods or by control engineers, thus making them fundamentally different from sensor or actuator faults. Situations where conventional model-based detection schemes may be rendered ineffective by intelligent cyber-attacks can be potentially tackled by data-based detection methods [16].

Machine-learning methods deployed for attack detection were presented in a number of works [14, 86, 108]. Using various machine-learning classification methods, cyber-attacks on power systems were distinguished from process disturbances in [48], and a behavior-based intrusion detection algorithm was developed to identify the type of attack [59]. Similarly, detection of cyber-attacks in a chemical process was realized via development of feed-forward artificial neural networks in [118], where compromised signals were rerouted to a secure sensor upon detection. One prominent use of machine-learning algorithms is to solve classification problems. For example, in [95], a hybrid approach using support vector machines and genetic algorithms was implemented and compared to existing network intrusion detection systems in the industry. An overview of recent research directions for applying supervised and unsupervised machine learning techniques to address the problem of anomaly detection was presented in [85]. Neural network and many of its variances have demonstrated remarkable performance. For instance, a Long Short-Term Memory (LSTM) recurrent neural network was used to build a classifier model for the intrusion detection system in [62]. The anomaly detection algorithm outlined in [42] also used a LSTM network as a predictor to model normal behavior of a water treatment testbed, and used the Cumulative Sum (CUSUM) method to identify anomalies. FNNs were used to construct detectors in [21], which were implemented in an economic model predictive control system with combined open-loop and closed-loop control modes to monitor and stay resilient against cyber-attacks. Many variants of CNN with different topologies, parameters, and structures were analyzed for the task

of intrusion detection in cyber-security of network traffic in [112], which have shown significant improvement than conventional classifiers. These recent literature contributions have demonstrated the feasibility of machine-learning algorithms in anomaly detection. At any large-scale chemical production plant, tremendous amount of data is being collected and archived daily in the historian. Using neural-network algorithms, these data can be put to use to train effective detection devices for monitoring and guarding the system against malicious cyber-attacks.

In many practical problems, a mathematical model based on physical first-principles is not fully known or too complicated to be used as the internal model for model-based control. A continuous-time RNN is developed in [3] and used in the context of nonlinear model predictive control where automatic differentiation techniques are used in both training the neural network model as well as in solving the online optimization problem in the controller. A combination of FNN and RNN is used in [132] where unmodeled dynamics as well as unknown higher-order terms from the decomposition of the nonlinear system are modeled by an FNN via supervised learning, and the optimization problem of the nonlinear model predictive control is iteratively solved using a single-layer RNN called the simplified dual network. Three deep neural network structures are trained and evaluated in [84] to demonstrate their effectiveness in modeling underlying characteristics of dynamical systems from input-output data. Moreover, in [121], an ensemble of RNN models is developed and used in the design of a Lyapunov-based economic model predictive control to address economic optimality of nonlinear systems. The integration of machine-learning-based modeling methods and various advanced control architectures is a broad field with expanding research scope.

Chemical process operation has extensively relied on automated control systems, and the need of accounting for multivariable interactions and input/state constraints has motivated the development of MPC. Moreover, augmentation in sensor information and network-based communication increases the number of decision variables, state variables, and measurement data, which in turn increases the complexity of the control problem and the computation time if a centralized MPC is used. Many industrial systems, such as power distribution grids, mechanical

systems and chemical processes, supply chains, and urban traffic networks, are large-scale systems that cannot be handled by using a centralized controller because the system to be controlled is too large and the control problem to be solved is too complex [10]. These challenges cannot be simply solved by using faster computers with larger memory. With these considerations in mind, distributed control systems have been developed, where multiple controllers with inter-controller communication are used to cooperatively calculate the control actions and achieve closed-loop plant objectives. In this context, MPC is a natural control framework to implement due to its ability to account for input and state constraints while also considering the actions of other control actuators. In other words, the controllers communicate with each other to calculate their distinct set of manipulated inputs that will collectively achieve the control objectives of the closed-loop system. Many distributed MPC methods have been proposed in the literature addressing the coordination of multiple MPCs that communicate to calculate the optimal input trajectories in a distributed manner (the reader may refer to [27, 92, 101] for reviews of results on distributed MPC, and to [31] for a review of network structure-based decomposition of control and optimization problems). A robust distributed control approach to plant-wide operations based on dissipativity was proposed in [114, 131]. Depending on the communication network, i.e., whether is one-directional or bi-directional, two distributed architectures, namely sequential and iterative distributed MPCs, were proposed in [70]. Furthermore, distributed MPC method was also used in [71] to address the problem of introducing new control systems to pre-existing control schemes. In a recent work [105], a fast and stable non-convex constrained distributed optimization algorithm was developed and applied to distributed MPC. As distributed MPC systems also depend on an accurate process model, the development and implementation of RNN models in distributed MPCs is an important area yet to be explored.

Decentralized control systems have also been proposed to address the concerns associated with control of large-scale systems, including challenges posed by high dimensionality, information structure constraints, uncertainties and delays in the system [10]. The analysis of the overall system is divided into independent sub-problems that are weakly related to each other to be regulated

by separate controllers, which together constitute a decentralized control structure. As such, decentralized control structures provide a practical solution to decoupling large-scale processes and reducing the computational complexity of a centralized control problem. Many recent works have been done on the subject of decentralized control [49, 53, 73]. For example, augmented estimators for subsystems were designed in [135] to form a distributed state estimation network from decentralized estimators. A quasi-decentralized control framework was developed in [103] where the network utilization and communication costs were minimized. With respect to earlier works, in [75], a stabilizing decentralized MPC algorithm was presented for nonlinear discrete-time systems subject to decaying disturbances. Moreover, a dynamically-coupled decentralized MPC system for large-scale linear processes subject to input constraints was presented in [5], where the global model of the process was approximated by multiple smaller subsystem models and the degree of decoupling was a tunable parameter in the design. Furthermore, a decentralized control strategy using the overlapping decomposition method and using H_∞ controllers was applied to a web transport system in [63]. By only acquiring local output measurements and deciding local control inputs, the main advantages of a decentralized control scheme are the reduced communications and parallel computations.

Amongst many MPC formulations, Lyapunov-based MPC (LMPC) ensures feasibility and stabilizability within an explicitly defined stability region using a Lyapunov-based stabilizing control law [79, 82]. Previously, safety considerations have been incorporated in the design of LMPC algorithms to specify unsafe regions of operation in state space characterized by the relative safeness of process states [4], as well as to ensure that unsafe regions are avoided at all times by utilizing a Control Lyapunov-Barrier Function (CLBF) [90, 122]. CLBFs are developed from the combination of a Control Lyapunov Function (CLF) and a Control Barrier Function (CBF), and can be used in control algorithms to account for both stability and safety, respectively [56, 83, 106]. Barrier functions, or barrier certificates, serve as an important tool in safety-critical systems where multi-objective control is involved [129]. CLBF-MPC has been proposed in [119, 122], where the stability and safety analysis for the closed-loop system in the presence of both bounded

and unbounded unsafe sets have been provided. Many other recent works [76, 136] have also explored MPC with discrete-time control barrier functions, as well as optimal control based on reinforcement learning with the inclusion of control barrier functions.

Incorporating CLBF in the design of distributed MPC in controlling multiple subsystems is a strategy that has not been explored in literature. This is essential to the operation of complex industrial processes where the overall system may encounter regions in state-space for which they would like to avoid, and the sub-controllers for each subsystem need to work cooperatively to achieve the stability and safety objectives. To this end, an analytical representation of the unsafe operating points in state-space is used to specify the CLBFs that will be used for the design of distributed MPC systems. The unsafe operating regions may be specified for each subsystem individually, or if these unsafe points are interdependent across subsystems, the unsafe regions may be specified holistically with respect to the overall process.

One challenge of implementing CLBF-based controllers is whether the unsafe operating region can be explicitly and accurately represented in closed form as a function of process states. While this may be possible to do for simple shapes or patterns of the unsafe region, it is practically difficult to express such a barrier function for real industrial processes with complex unsafe regions that cannot be readily described with common explicit functions. To this end, FNN can be used to model barrier functions based on data samples collected from the safe and the unsafe operating regions.

There has been previous works on characterizing a barrier function using machine learning methods, such as using support vector machines [100] and neural networks [58, 139]. Moreover, in [68, 89], optimization-based approaches are used to learn CBFs from data for nonlinear continuous control affine dynamical systems as well as hybrid systems. In [130], an imitation learning framework is proposed to learn neural network-based feedback controllers with CBF constraints for systems under disturbances. The work in [58] uses neural networks to jointly learn a Lyapunov-like function and a barrier function and obtains a safe and goal-reaching control policy. Similarly, in [139], barrier functions are synthesized using neural networks that use

a devised activation function Bent-ReLu and checked against the barrier function criteria as a formal guarantee. Although formal proofs of guaranteed safety and stability have been provided either from *a priori* theoretical development or posterior empirical verification, the question of generalization accuracy of machine learning techniques has not been addressed.

Many research works have been conducted on the probabilistic safety certification of barrier functions, but the probability analysis is with respect to uncertainties that exist in the process dynamics (e.g., [60, 72, 74]), and not in the sense of analyzing the generalization error of the modeling method. For example, in [72], a Gaussian process is used to model the projection of unknown residual dynamics onto a CBF; similarly in [60], the Gaussian process approach is used to obtain a distribution over the system dynamics, which is then used to ensure safety with high probability by specifying a chance constraint on a CBF. The work in [29] develops barrier functions for stochastic systems with sufficient conditions for safety with probability.

On the other hand, probably approximately correct (PAC) learning theory provides a framework for analyzing the generalization ability of machine learning models, and provides the conditions under which a learning algorithm is probably able to yield an output that is approximately correct [81, 110]. One way to characterize the machine learning model's capability to generalize new unseen data based on learned data is to examine the generalization error. In [37], a tighter error bound on the performance of classification via support vector machine is characterized by exploiting domain knowledge. A bound on the generalization error of feed-forward neural networks has been developed by providing a bound on the Rademacher complexity of the network [43]. In [126], a similar bound is provided for recurrent neural networks, and statistical stability analysis of Lyapunov-based MPC using the recurrent neural network model was introduced. Generalization error in deep learning algorithms has been surveyed in [55] with discussions on different measures to assess generalization capabilities of deep neural networks, such as PAC-Bayes theory, algorithm stability, algorithm robustness, and compression-based approach. In this dissertation, statistical analysis on the FNN-based CBF construction method proposed in our previous work in [25] is also provided, and the FNN-based CBF will be used to design a

CLBF-based model predictive control system.

1.3 Dissertation Objectives and Structure

This dissertation presents machine learning approaches to address challenges in modern control systems such as operational safety, cybersecurity, and large-scale process control, and provides control theoretic analyses as well as applications on nonlinear chemical process examples. The objectives of this dissertation can be summarized as follows:

1. To present a framework of integrating machine-learning-based detection algorithms with resilient control methods implemented in the contexts of two-tier LMPC, EMPC, distributed and decentralized MPC, to ensure cyber-security of industrial processes

2. To develop machine-learning-based distributed and decentralized model predictive control schemes with rigorous theoretical analysis on their closed-loop stability properties

3. To investigate the safety and stability properties of distributed MPC systems designed based on Control Lyapunov-Barrier Functions

4. To present machine-learning-based methods of characterizing Control Lyapunov-Barrier Functions and provide control theoretic proof as well as statistical analysis on the stability and safety properties of CLBF-based MPC algorithms

5. To illustrate the applications of the proposed control methods to nonlinear chemical process examples

The remainder of this dissertation is organized as follows. Chapter 2 presents a detector-integrated two-tier control architecture capable of identifying the presence of various types of cyber-attacks, and ensuring closed-loop system stability upon detection of the cyber-attacks. Working with a general class of nonlinear systems, an upper-tier Lyapunov-based Model Predictive Controller, using networked sensor measurements to improve closed-loop

performance, is coupled with lower-tier cyber-secure explicit feedback controllers to drive a nonlinear multivariable process to its steady-state. Although the networked sensor measurements may be vulnerable to cyber-attacks, the two-tier control architecture ensures that the process will stay immune to destabilizing malicious cyber-attacks. Data-based attack detectors are developed using sensor measurements via artificial neural networks, where various types of cyber-attacks are introduced to the process under nominal and noisy operating conditions. The detectors are applied online to a simulated reactor-reactor-separator process. Simulation results demonstrate the effectiveness of these detection algorithms in detecting and distinguishing between multiple classes of intelligent cyber-attacks. Upon successful detection of cyber-attacks, the two-tier control architecture allows convenient reconfiguration of the control system to stabilize the process to its operating steady-state.

Chapter 3 proposes resilient operation strategies for nonlinear processes that are vulnerable to targeted cyber-attacks, as well as detection and handling of standard types of cyber-attacks. Working with a general class of nonlinear systems, a modified Lyapunov-based Economic Model Predictive Controller (LEMPC) using combined closed-loop and open-loop control action implementation schemes is proposed to optimize economic benefits in a time-varying manner while maintaining closed-loop process stability. Although sensor measurements may be vulnerable to cyber-attacks, the proposed controller design and operation strategy ensure that the process will maintain stability and stay resilient against particular types of destabilizing cyber-attacks. Data-based cyber-attack detectors are developed using sensor data via machine-learning methods, and these detectors are periodically activated and applied online in the context of process operation. Using a continuously stirred tank reactor example, simulation results demonstrate the effectiveness of the resilient control strategy in maintaining stable and economically optimal operation in the presence of cyber-attacks. The detection results demonstrate the capability of the proposed method in identifying the presence of a cyber-attack, as well as in differentiating between different types of cyber-attacks. Upon successful detection of the cyber-attacks, the impact of cyber-attacks can be mitigated by replacing the attacked sensors by secure back-up sensors, and secure operation will

resume with the process operated under the proposed resilient LEMPC control strategy.

In Chapter 4, cyber-security in large-scale complex processes consisting of multiple sub-systems under the control of decentralized and distributed controllers is studied. With the expansion in communication networks, vulnerability to cyber intrusions also increases. This work investigates the effect of different types of standard cyber-attacks on the operation of nonlinear processes under centralized, decentralized, and distributed model predictive control systems. The robustness of the decentralized control architecture over distributed and centralized control architectures is analyzed. Moreover, a machine-learning-based detector is trained using sensor data to monitor and ensure the cyber-security of the overall system. Specifically, detectors built with feed-forward neural networks are used to detect the presence of an attack or identify the subsystem being attacked. A nonlinear chemical process example is simulated to demonstrate the robustness of decentralized control architectures and the effectiveness of the neural-network detection scheme in maintaining the closed-loop stability of the system.

Chapter 5.1 and Chapter 5.2 explore the design of distributed MPC and decentralized MPC systems for nonlinear processes using machine learning models to predict nonlinear dynamic behavior. Firstly, the distributed MPC scheme is discussed, where sequential and iterative distributed model predictive control systems are designed and analyzed with respect to closed-loop stability and performance properties. Extensive open-loop data within a desired operating region are used to develop LSTM recurrent neural network models with a sufficiently small modeling error from the actual nonlinear process model. Subsequently, these LSTM models are utilized in Lyapunov-based distributed MPC to achieve efficient real-time computation time while ensuring closed-loop state boundedness and convergence to the origin. Using a nonlinear chemical process network example, the simulation results demonstrate the improved computational efficiency when the process is operated under sequential and iterative distributed MPCs while the closed-loop performance is very close to the one of a centralized MPC system. Secondly, the dencetralized MPC system for nonlinear processes is presented, where the nonlinear process is broken down into multiple, yet coupled subsystems and the dynamic behavior of each subsystems is described by

machine learning models. One decentralized MPC is designed and used to control each subsystem while accounting for the interactions between subsystems through feedback of the entire process state. The closed-loop stability of the overall nonlinear process network and the performance properties of the decentralized model predictive control system using machine-learning prediction models are analyzed. More specifically, multiple recurrent neural network models suited for each different subsystem need to be trained with a sufficiently small modeling error from their respective actual nonlinear process models to ensure closed-loop stability. These recurrent neural network models are subsequently used as the prediction model in decentralized Lyapunov-based MPCs to achieve efficient real-time computation time while ensuring closed-loop state boundedness and convergence to the origin. The simulation results of a nonlinear chemical process network example demonstrate the effective closed-loop control performance when the process is operated under the decentralized MPCs using the independently-trained recurrent neural network models, as well as the improved computational efficiency compared to the closed-loop simulation of a centralized MPC system.

In Chapter 6, a Control Lyapunov-Barrier Function-based Model Predictive Control method is developed utilizing a feed-forward neural network specified Control Barrier Function and a recurrent neural network predictive model to stabilize nonlinear processes with input constraints, and to guarantee that safety requirements are met for all times. The nonlinear system is first modeled using recurrent neural network techniques, and a Control Barrier Function is characterized by constructing a feed-forward neural network model with unique structures and properties. The FNN model for the CBF is trained based on data samples collected from safe and unsafe operating regions, and the resulting FNN model is verified to demonstrate that the safety properties of the CBF are satisfied. Given sufficiently small bounded modeling errors for both the FNN and the RNN models, the proposed control system is able to guarantee closed-loop stability while preventing the closed-loop states from entering unsafe regions in state-space under sample-and-hold control action implementation. We provide the theoretical analysis for both bounded unsafe sets in state-space, and demonstrate the effectiveness of the proposed control strategy using a nonlinear

chemical process example with a bounded unsafe region.

Chapter 7 focuses on the design of distributed model predictive control (DMPC) systems for nonlinear processes with input constraints using a Control Lyapunov-Barrier Function to achieve simultaneous closed-loop stability and process safety. Specifically, a constrained CLBF is first used to design explicit control laws for each subsystem and to characterize a set of initial conditions, starting from which the closed-loop states of the overall nonlinear system are guaranteed to converge to the operating steady-state under the CLBF-based control laws while avoiding unsafe regions in state space. We then propose the CLBF-based DMPC, and prove its feasibility and effectiveness in ensuring the stability and avoidance of unsafe regions under sample-and-hold implementation of DMPC control actions. The CLBF-based DMPC is applied to both sequential and iterative DMPC designs in the general sense, and a modification to the DMPC formulation is presented for special cases of systems where the coupling between subsystems is in a one-way cascading manner. The proposed CLBF-DMPC method is demonstrated via a nonlinear chemical process example consisting of two subsystems.

In Chapter 8, we present statistical model predictive control with Control Lyapunov-Barrier Functions built using machine learning approaches, and analyze closed-loop stability and safety properties in probability using statistical machine learning theory. A FNN is used to construct the Control Barrier Function, and a generalization error bound can be obtained for this FNN via the Rademacher complexity method. The FNN Control Barrier Function is incorporated in a CLBF-based MPC, which is used to control a nonlinear process subject to input constraints. The stability and safety properties of the closed-loop system under the sample-and-hold implementation of FNN-CLBF-MPC are evaluated in a statistical sense. A chemical process example is used to demonstrate the relation between various factors of building an FNN model and the generalization error, as well as the probabilities of closed-loop safety and stability for both bounded and unbounded unsafe sets.

Chapter 9 summarizes the main results of the dissertation.

Chapter 2

A Cyber-secure Control-Detector Architecture for Nonlinear Processes

This chapter presents a detector-integrated two-tier control architecture capable of identifying the presence of various types of cyber-attacks, and ensuring closed-loop system stability upon detection of the cyber-attacks. Working with a general class of nonlinear systems, an upper-tier Lyapunov-based Model Predictive Controller (LMPC), using networked sensor measurements to improve closed-loop performance, is coupled with lower-tier cyber-secure explicit feedback controllers to drive a nonlinear multivariable process to its steady-state. Although the networked sensor measurements may be vulnerable to cyber-attacks, the two-tier control architecture ensures that the process will stay immune to destabilizing malicious cyber-attacks. Data-based attack detectors are developed using sensor measurements via machine-learning methods, namely artificial neural networks (ANN), under nominal and noisy operating conditions, and applied online to a simulated reactor-reactor-separator process. Simulation results demonstrate the effectiveness of these detection algorithms in detecting and distinguishing between multiple classes of intelligent cyber-attacks. Upon successful detection of cyber-attacks, the two-tier control architecture allows convenient reconfiguration of the control system to stabilize the process to its operating steady-state.

Despite current literature efforts on stealthy attack analysis and machine-learning-based detection, there is a lack of an integration of the two, as well as a broader application of detection schemes across stealthy attack classes and nonlinear chemical processes. Furthermore, feasible mitigation practices using control strategies after the occurrence of attacks have not yet been explored. In light of these gaps, the contributions of this work are as follows: 1) construction of data-based machine-learning detection algorithms which can effectively detect multiple classes of intelligent cyber-attacks; 2) design of a robust control architecture to promptly contain and eliminate the impact of cyber-attacks by reconfiguring the control system; and 3) application of the proposed detection and mitigation schemes to a benchmark multivariable nonlinear process example, which is a process example widely used in literature to test the performance of new control system designs [87, 101, 134]. This Chapter is organized as follows: notation and the class of nonlinear process systems considered are presented in Section 2.1; the cyber-secure control architecture is formulated in Section 2.2; the design and detection mechanism of cyber-attacks are presented in Section 2.3; and the application of the proposed methodology to a nonlinear chemical process network is presented in Section 2.4.

2.1 Preliminaries

2.1.1 Nonlinear System Formulation

In this work, $|\cdot|$ is used to denote the Euclidean norm of a vector; x^T denotes the transpose of x; $\mathbf{R}^{\mathbf{n}}_{+}$ denotes the set of vector functions of dimension n whose domain is $[0,\infty)$. Class $\mathscr{K}$ functions $\alpha(\cdot) : [0,a) \to [0,\infty]$ are defined as strictly increasing scalar functions with $\alpha(0) = 0$. The class of continuous-time nonlinear systems considered is described by the following state-space form:

$$\dot{x}(t) = f(x(t), u_c(t), u_a(t)) \tag{2.1a}$$

$$y_c(t) = h_c(x(t)),\ y_a(t) = h_a(x(t)) \tag{2.1b}$$

where $x \in \mathbf{R}^{n_x}$ is the state vector, $y_c(t) \in \mathbf{R}^{n_{y_c}}$ represents the vector of state measurements that are sampled continuously (e.g., reactor temperature), and $y_a(t) \in \mathbf{R}^{n_{y_a}}$ represents the vector of networked state measurements that may be sampled asynchronously at $t = t_k$ (e.g., reactor product concentration); u_c and u_a are the manipulated input vectors, which are constrained by $[u_c \in \mathbf{R}^{m_{u_c}}, u_a \in \mathbf{R}^{m_{u_a}}] \in U := \{u_i^{min} \leq u_i \leq u_i^{max}, i = 1, ..., m_{u_c} + m_{u_a}\}$. Through y_c and y_a, we assume measurement of the full state vector x can be obtained at t_k. Without loss of generality, the initial time t_0 is taken to be zero ($t_0 = 0$). It is assumed that $f(\cdot)$ is a sufficiently smooth vector function of its arguments, and $h_c(\cdot)$ and $h_a(\cdot)$ are sufficiently smooth vector functions of x where $f(0,0,0) = 0$, $h_c(0) = 0$, $h_a(0) = 0$. Thus, the origin is an equilibrium point of the system of Eq. 8.1 under $u_c(t) = 0$ and $u_a(t) = 0$.

2.2 Cyber-secure Two-tier Control Architecture

We propose a cyber-secure control architecture that unites a lower-tier control system that uses the dedicated sensor measurements, $y_c(t)$, to ensure stability of the steady-state of the closed-loop system and an upper-tier, high-performance control system (in this work, model predictive control) that uses both dedicated ($y_c(t)$) and networked ($y_a(t)$) sensor measurements to improve closed-loop performance significantly above what could be achieved with the lower-tier control system. Below we present in detail the lower-tier and upper-tier control systems.

2.2.1 Lower-tier Control System

We assume that there exists an explicit feedback controller of the form $u_c(t) = \phi_c(y_c(x(t))) \in U$ that can stabilize the closed-loop system of Eq. 8.1. This controller, using only the continuous measurements $y_c(t)$ is termed the lower-tier controller, and is designed such that the origin of the nominal closed-loop system of Eq. 8.1 with the input $u_a(t) = 0$ is rendered asymptotically stable. Therefore, there exist class $\mathscr{K}$ functions $\alpha_i(\cdot)$, $i = 1,2,3,4$, and a positive definite control

Lyapunov function $V(x)$ that satisfy the following conditions:

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|), \tag{2.2a}$$

$$\frac{\partial V(x)}{\partial x} f(x, \phi_c(y_c(x)), 0) \leq -\alpha_3(|x|), \tag{2.2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq \alpha_4(|x|) \tag{2.2c}$$

for all $x \in D \subseteq \mathbf{R}^{n_x}$, where D is an open neighbourhood around the origin. We construct a subset defined as a level set of $V(x)$ inside D, $\Omega_\rho := \{x \in D \mid V(x) \leq \rho, \rho > 0\}$, to represent an estimate of the stability region of the closed-loop system of Eq. 8.1 under $\phi_c(y_c)$. Ω_ρ is an invariant set for the closed-loop system. Therefore, starting from any initial state in Ω_ρ, $\phi_c(y_c)$ guarantees that the state trajectory of the closed-loop system remains within Ω_ρ and asymptotically converges to the origin. Thus, given that the sensor measurements received by the lower-tier controller are secure and reliable, the lower-tier controller is able to stabilize the process to the origin for any initial conditions inside Ω_ρ.

2.2.2 Upper-tier Model Predictive Control System

To fully utilize the networked (potentially asynchronous) state measurements $y_a(t)$ and to compute $u_a(t)$ that improves the overall closed-loop performance over what can be achieved with $\phi_c(y_c)$, a Lyapunov-based MPC (LMPC) is used as the upper-tier controller with its contractive constraint defined based on the stability region of the lower-tier controller such that the asymptotic stability of the closed-loop system will not be jeopardized by the contributions of $u_a(t)$. The optimization

problem of LMPC is as follows:

$$\mathscr{J} = \min_{u_a \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), \tilde{u}_c(t), u_a(t))dt \tag{2.3a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = f(\tilde{x}(t), \phi_c(y_c(\tilde{x}(t)), u_a(t)) \tag{2.3b}$$

$$\dot{\hat{x}}(t) = f(\hat{x}(t), \phi_c(y_c(\hat{x}(t)), 0) \tag{2.3c}$$

$$\tilde{x}(t_k) = \hat{x}(t_k) = x(t_k) \tag{2.3d}$$

$$[u_c(t), u_a(t)] \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{2.3e}$$

$$V(\tilde{x}(t_k)) \leq V(\hat{x}(t_k)), \text{if } V(\tilde{x}(t_k)) > \rho_{min} \tag{2.3f}$$

$$V(\tilde{x}(t)) \leq \rho_{min}, \ \forall \ t \in [t_k, t_{k+N}), \text{if } V(\tilde{x}(t_k)) \leq \rho_{min} \tag{2.3g}$$

where u_a belongs to a family of piece-wise constant functions $S(\Delta)$ with sampling period Δ, N is the number of sampling periods in the prediction horizon, and the LMPC optimization problem presented in Eq. 2.3 optimizes u_a over the prediction horizon $t \in [t_k, t_{k+N})$ when full state measurement is received at time instance t_k. The optimal solution is denoted $u_a^*(t)$. The first control action of $u_a^*(t)$, i.e., $u_a(t) = u_a^*(t_k)$, is applied in open loop until a new full-state measurement $x(t_k)$ obtained from $y_c(t_k)$ and $y_a(t_k)$ in Eq. 3.1b becomes available to the LMPC and the optimization problem is solved again. In the meantime, the lower-tier controller continuously calculates and applies $u_c(t) = \phi_c(y_c(t))$ based on continuous measurement feedback $y_c(t)$. If the time between two consecutive asynchronous measurements is longer than the prediction horizon $N \cdot \Delta$, then $u_a(t)$ is set to zero for the remainder of the asynchronous sampling interval past the prediction horizon, such that it does not act as an additional disturbance to the lower-tier controller before the next y_a arrives. In Eqs. 2.3b and 2.3c, $\tilde{x}(t)$ and $\hat{x}(t)$ are the predicted state trajectories of the two-tier nominal system using control actions $\phi_c(y_c(\tilde{x}(t))$ coupled with $u_a(t)$ computed by LMPC, and control actions $\phi_c(y_c(\hat{x}(t))$ coupled with $u_a(t) = 0$, respectively. As shown in Eq. 2.3d, full-state measurements received at t_k are used as the initial conditions of the predicted trajectories in the optimization problem of LMPC. Both upper-tier and lower-tier controller inputs are subject to

their respective constraints defined by U in Eq. 2.3e.

Given that the lower-tier controller is able to stabilize the system independently as the Lyapunov function under lower-tier control satisfies the conditions in Eq. 3.2, the contractive constraint in Eq. 2.3f ensures that the value of the Lyapunov function of the closed-loop system under two-tier control, $V(\tilde{x}(t_k))$, is lower than or equal to that under lower-tier control alone $V(\hat{x}(t_k))$. Therefore, Ω_ρ is the closed-loop stability region under two-tier control. In other words, the upper-tier maintains the closed-loop stability of the system while improving the overall closed-loop performance. In order to avoid oscillations when the states approach the equilibrium point, the Lyapunov function is bounded as seen in Eq. 2.3g once the system enters a small region around the equilibrium point characterized by a level set $\Omega_{\rho_{min}}$, where $0 < \rho_{min} < \rho$. It is important to note that the LMPC is only executed when full-state information is received from both the continuous and asynchronous measurements as they become available at time t_k. The continuous measurements are measured and transmitted by sensors in a point-to-point network, and are used by the lower-tier control system to compute stabilizing control actions continuously. Thus, the continuous measurements will be available and readily used as state feedback in addition to the asynchronous measurements when the LMPC is activated. This two-tier control design, where the networked sensor measurements, $y_a(t)$, used only by the upper-tier controller may be under potential cyber-attack, is illustrated in Fig. 2.1(a).

2.3 Cyber-attack Design and Detection

2.3.1 Attack Scenarios

The upper-tier control system – where networked sensor information is incorporated and part of its measurement feedback is asynchronous – is vulnerable to cyber-attacks. Due to these irregular and sparse measurements, suspicious disparities between consecutive state measurements may be less apparent or susceptible to detection by the control engineer or classical fault-detection schemes. Furthermore, deviations caused by intelligent cyber-attacks and their dynamic impact

on the process may be less detectable when multiple states are attacked. While the networked asynchronous measurements used only in the upper-tier controller are more susceptible to attacks, the continuous measurements used in both upper-tier and lower-tier controllers must remain intact for a few reasons. Firstly, we assume that the process is stabilized under lower-tier controllers, and the upper-tier controller is designed such that its control Lyapunov function is contained inside that of the stabilizing lower-tier controllers. Therefore, the closed-loop stability under two-tier control is ensured by the stabilizing lower-tier controllers, and the closed-loop stability under lower-tier control is only guaranteed if the continuous measurements feeding into lower-tier controllers are secure and reliable. Secondly, having a secure stabilizing lower-tier controller allows quick mitigation measures by changing the control structure once a cyber-attack is identified in the networked measurements. Since the process can be driven to its operating steady-state using only the lower-tier control system, in the case of a confirmed cyber-attack detection, the upper-tier controller which uses the corrupted networked measurements will be shut off. If the continuous measurements are also tampered, then the closed-loop stability under the lower-tier controllers is no longer guaranteed, and this mitigation plan is rendered ineffective. Therefore, having secure continuous sensor measurements is instrumental to maintaining functional stabilizing lower-tier controllers, which in turn ensures robustness of the closed-loop system to cyber-attacks.

To capture realistic sensor variance and to differentiate cyber-attacks from normal device fluctuations, bounded sensor noise is also considered. Thus, in our formulation, two scenarios are considered:

1. Nominal model is as presented in the nonlinear system outlined in Eq. 8.1 where output sensors do not encounter any sensor noise.

2. Noise model adopts the same dynamic system model in Eq. 3.1a, but with bounded Gaussian noise $w(t) \in W$ added to all sensor measurements, where $W = \{w \in \mathbf{R}^{n_{yc}+n_{ya}} \mid |w| \leq w_{max}\}$. Depending on the range of the outputs, the standard deviation of noise distribution for each

sensor is adjusted accordingly. Therefore, Eq. 3.1b is modified to the following form:

$$y_c(t) = h_c(x(t)) + w(t),\ y_a(t) = h_a(x(t)) + w(t),\ w(t) \in\ W \tag{2.4}$$

To collect closed-loop data used for machine-learning detector training, attacks with varying durations L_a are introduced at random times i_0 during the simulation period. In both cases (with and without noise), signals without attack interceptions are classified as "no attack". Attacks on single sensor and on multiple sensors are both considered, where the data collected from single-sensor tampering is used for detector training, the multiple-sensor attacks are simulated for online testing of the effectiveness of the detector algorithms. Data collected from single-sensor tampering also allows for sensor isolation using machine-learning-based models. For clarity, we consider that only one type of cyber-attack will occur at a time, i.e., there will not be a hybrid of multiple attack types within each attack duration.

Remark 2.1. *The continuous state measurements used by lower-tier controllers are also used by the upper-tier controller despite the asynchronous execution frequency of the upper-tier controller. Since these continuous state measurements cannot be tampered, the same measurements fed into the upper-tier controller remain intact when an attack occurs. Thus, even in the case where multiple sensors are under attack, only those sending sampled asynchronous state measurements to the upper-tier controller will be corrupted. Moreover, it is not meaningful to simulate an intelligent cyber-attack that targets the two separate communication channels going into the two tiers of controllers such that the continuous measurements received by lower-tier controllers are accurate while the continuous measurements received by the upper-tier controller are falsified. This is because a simple tracker that examines the deviation between the same measurements received by both controllers would identify the presence of this abnormality. Therefore, the continuous state measurements remain unattacked in both controllers.*

2.3.2 Types of Intelligent Cyber-attacks

As intelligent cyber-attacks are adaptive to the process and control system behavior, we may assume that they are as powerful as having access to the measurement feedback signals (sensor

attack), the control command signals (actuator attack), or auxiliary information such as the threshold and bias parameters in detection methods such as CUSUM [16, 80]. Being process and controller behavior aware, the attacks will therefore have information on the stability region of the process under two-tier control, as well as existing alarm triggers on the ideal operating window imposed on the input and output variables. In this work, we only consider attacks on sensor measurements. During normal operation, these sensor feedback measurements need to accurately reflect the true state of the plant, otherwise any falsified measurement may result in control actions that no longer guarantee closed-loop stability, and may eventually drive the process away from its steady-state and outside of Ω_ρ. Intelligent cyber-attacks are designed such that the controller is able to compute feasible control actions (i.e., the falsified state is not outside the closed-loop stability region Ω_ρ), but have large enough magnitude of variations such that the control system will not be able to drive the process to its operating steady-state. The four most important types [16] of such attacks are considered below.

2.3.2.1 Min-Max Cyber-attack

Min-max attacks are designed to induce maximum destabilizing impact within shortest time without being detected. In order to stay undetectable by classical detection methods, min-max attacks are introduced based on the more conservative value of the following two conditions: 1) a window around the equilibrium point of the attacked state(s) reflecting reasonable physical operating conditions; 2) state values furthest from the equilibrium point (minimum or maximum) such that the system does not exit the closed-loop stability region Ω_ρ. Attacks generated based on these two conditions ensure that the attacked state measurements fed to the control system do not exit the stability region or the configured operating window, and do not trigger any conventional detection alarms designed based on these boundary values. The min-max attack can be formulated as follows:

$$\bar{x}(t_i) = \min\{\arg\max_{x\in\mathbf{R^{n_x}}}\{V(x(t_i)) \leq \rho\}, \arg\max_{x\in\mathbf{R^{n_x}}}\{x(t_i) \in \chi\}\}, \ \ \forall\, i \in [i_0, i_0 + L_a] \tag{2.5}$$

where ρ defines the level set of the Lyapunov function $V(x)$ that characterizes the stability region of the closed-loop system of Eq. 8.1 under the two-tier control system, $\chi := \{x_l \leq x \leq x_u\}$ represents the ideal state operating window, $\bar{x}$ is the compromised sensor measurement at each sampling step, i_0 marks the time instant that attack is added, and L_a denotes the time duration of the attack in terms of sampling periods.

2.3.2.2 Replay Cyber-attack

In a replay attack, the attacker first records segments of the system output corresponding to a nominal operating condition where large oscillations occur. The attacker then intercepts and resets the current process state measurements to these pre-recorded values. Replay attacks can be represented by the following equations:

$$\bar{x}(t_i) = x(t_k),\ \forall\, k \in [k_0, k_0 + L_a],\ \ \forall\, i \in [i_0, i_0 + L_a] \tag{2.6}$$

where $x(t_k)$ is the true plant measurement, L_a represents the length of the attack in terms of sampling periods, and $\bar{x}$ is the series of replay attacks introduced at time t_{i_0} duplicating previous plant measurements that are recorded starting from time t_{k_0}. As previous plant outputs are obtained from legitimate closed-loop measurements and given by secure sensors, these state values are supposedly inside the stability region and the operating envelope. Therefore, by replicating these values and feeding them back to the controller, classical detectors will not be able to recognize the abnormality caused by replay cyber-attacks.

2.3.2.3 Geometric Cyber-attack

Geometric cyber-attacks aim to deteriorate the closed-loop system stability slowly at the beginning, then geometrically increase their impact as time progresses, with its maximum damage achieved at the end of the attack duration. Initially, the attacker adds a small constant β to the true measured output (β is well below the maximum allowable value as defined in a min-max attack). At each subsequent time step, this offset is multiplied by $(1+\alpha)$, where $\alpha \in (0,1)$, until it reaches the

maximum allowable attack value. The two parameters α and β are therefore selected based on the stability region, the operating envelope, and the attack duration. Geometric attacks can be written in the form:

$$\bar{x}(t_i) = x(t_i) + \beta \times (1+\alpha)^{i-i_0}, \ \ \forall\, i \in [i_0, i_0 + L_a] \tag{2.7}$$

where $\bar{x}$ is the compromised sensor measurement, β and α are parameters that define the magnitude and speed of the geometric attack, i_0 signifies the time instant at which the attack starts, and L_a is the duration of the attack in terms of sampling periods.

2.3.2.4 Surge Cyber-attack

Surge attacks act similarly as min-max attacks initially to maximize the disruptive impact for a short period of time, then they are reduced to a lower value. In our case, the duration of the initial surge in terms of sampling times is selected as $L_s \in [2,5]$ to differentiate itself from a min-max attack. Moreover, the initial surge period L_s is chosen to be in this range such that the potential time delays on the detection alarm will not be longer than L_s where the impact is most severe, and in turn, may cause a missed alarm from the detector. After the initial surge, the reduced constant value – at which the attack stays at – is chosen considering the impact of the initial surge and the total duration of the attack such that the cumulative error between state measurements and their steady-state values will not exceed the threshold defined in some statistic-based detection methods (e.g., CUSUM).The formulation of a surge attack is presented below:

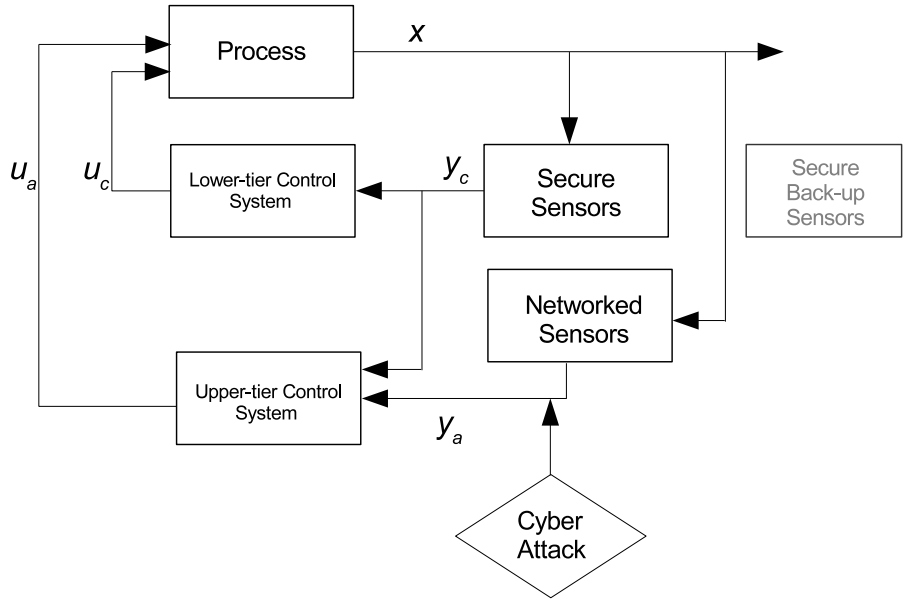
$$\begin{aligned} \bar{x}(t_i) &= \min\{\arg\max_{x \in \mathbf{R^{n_x}}} \{V(x(t_i)) \leq \rho\}, \arg\max_{x \in \mathbf{R^{n_x}}} \{x(t_i) \in \chi\}\}, \text{ if } i_0 \leq i \leq i_0 + L_s \\ \bar{x}(t_i) &= \arg\max_{x \in \mathbf{R^{n_x}}} \{|x(t_i)|, 0 \leq i \leq i_0\}, \text{ if } i_0 + L_s < i \leq i_0 + L_a \end{aligned} \tag{2.8}$$

where i_0 is the start time of the attack, L_s is the duration of the initial surge, and L_a is the total duration of the attack in terms of sampling periods. After the initial surge, the attack is reduced to a lower constant value, which is obtained by examining the secure state measurements prior to the occurrence of the surge attack, and taking the value that is furthest away from the origin.

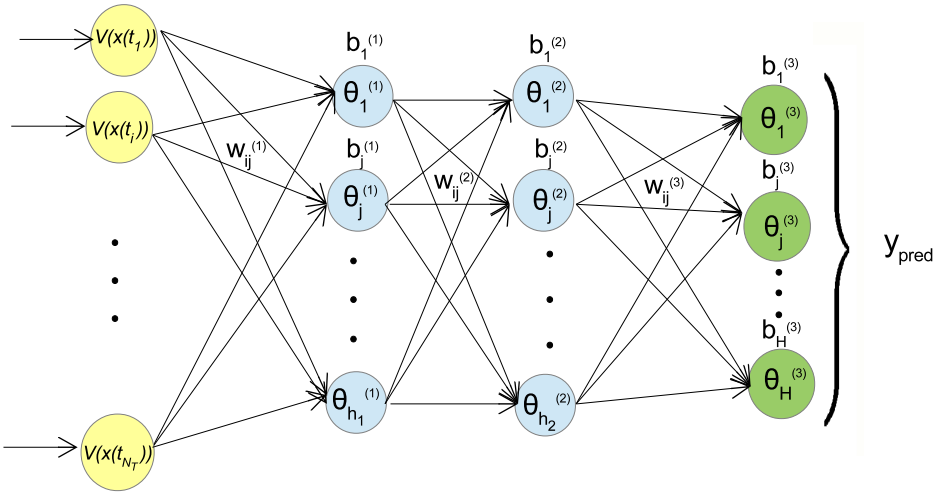
2.3.3 Machine-Learning-Based Detection of Cyber-attacks

There are many advantages to using a data-based approach to develop the cyber-attack detector [1, 52, 85]. Firstly, with attacks having possible access to information on process behavior (stability region and variable operating window), physical-model-based detection methods where the statistics threshold and false alarm bias are selected based on process operation are rendered ineffective [16]. Secondly, during real-life operation, plant model structure and parameters may be subject to modifications due to a changing operating environment. Therefore, using a data-based (physical model independent) method to train a detection mechanism for cyber-attacks is resilient to both process changes and intelligently designed attacks.

Within well-practiced machine-learning methods, neural networks (NN) have proven their effectiveness in both supervised and unsupervised classification problems [44]. Depending on the training data and target number of classes the algorithm aims to identify, neural networks can be used to distinguish between "attack" and "no attack" (two classes), or to identify the type of attack (multiple classes). While under attack, data collected from individual sensors can also be used to locate the corruption, where the neural network model distinguishes between multiple classes with each class representing one problematic sensor. In our study, a feed-forward artificial neural network is used for supervised classification. Through a series of nonlinear transformations, neurons in the first hidden layer are derived from the inputs, and hidden neurons in subsequent layers are derived from their precedent layer, with the output calculated from neurons in the last hidden layer. These nonlinear transformations are in the form of an activation function of biases and weighted sum of inputs (or neurons in the previous layer). The structure of a basic neural network model employed here is shown in Fig. 2.1(b), with each input representing the control Lyapunov function of the full state measurements at each asynchronous sampling time instant, and an output vector for predicted class label.



(a)



(b)

Figure 2.1: Two-tier control-detector architecture showing (a) Lower-tier controllers using continuous secure sensor measurements and an upper-tier model predictive controller using both continuous (secure) and networked (vulnerable to cyber-attacks) sensor measurements, and (b) Feed-forward neural network structure with 2 hidden layers with inputs being the full-state Lyapunov function at each sampling time of the model predictive controller within the detection window, and output being the probability of each class label for the examined trajectory indicating the status and/or type of cyber-attack.

The mathematical formulation of the two-hidden-layer feed-forward neural network is as

follows:

$$\theta_j^{(1)} = g_1(\sum_{i=1}^{N_T} w_{ij}^{(1)} V(x(t_i)) + b_j^{(1)}) \tag{2.9a}$$

$$\theta_j^{(2)} = g_2(\sum_{i=1}^{h_1} w_{ij}^{(2)} \theta_i^{(1)} + b_j^{(2)}) \tag{2.9b}$$

$$\theta_j^{(3)} = g_3(\sum_{i=1}^{h_2} w_{ij}^{(3)} \theta_i^{(2)} + b_j^{(3)}), \;\; y_{pred} = [\theta_1^{(3)}, \theta_2^{(3)}, ..., \theta_H^{(3)}]^T \tag{2.9c}$$

with $\theta_j^{(1)}$ and $\theta_j^{(2)}$ representing neurons in the first and second hidden layer, respectively, where $j = 1, ..., h_l$ is the number of neurons in layer $l = 1$ and $l = 2$. $\theta_j^{(3)}$ represents neurons in the output layer ($l = 3$), where $j = 1, ..., H$, and H is the number of class labels. In this study, the number of hidden layers is 2; however, the formulation of neurons can be similarly applied to multiple hidden layers as well. In the input layer, input variables $V(x(t_i))$ are the control Lyapunov function of the full state measurements at time t_i, where $i = 1, ..., N_T$ is the length of the time-varying trajectory for each input sample. The weight associated with the connections between neurons i and j in consecutive layers (from $l-1$ to l) is denoted by $w_{ij}^{(l)}$, and the bias placed on the j^{th} neuron in the l^{th} layer is denoted by $b_j^{(l)}$. Each layer receives information from its previous layer, and computes an output based on the optimized weights, biases, and its nonlinear activation function – denoted g_l (e.g., hyperbolic tangent sigmoid transfer function $g(z) = \frac{2}{1+e^{-2z}} - 1$, and softmax function $g(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{H} e^{z_i}}$ where H is the number of class labels). Performances of different common activation functions including ReLu, sigmoid, radial basis functions were analyzed in [96]. In the output layer, y_{pred} is a vector providing the predicted probabilities of each class label for the examined sample, where the neuron with the highest probability indicates the predicted class label. Depending on the type of classification problem the neural network is intended for, the predicted class label provides information on either the status or the type of a cyber-attack. The weights and biases are optimized by minimizing the Bayesian regularized mean squared error cost function. The cost function used in the optimization problem is of the form:

$$S(w) = \gamma \sum_{k=1}^{N_s} (y_{pred,k} - y_{true,k})^2 + \zeta \sum_{p=1}^{N_w} w_p^2 \tag{2.10}$$

where $k = 1, ..., N_s$ represents the number of samples in the training dataset, $p = 1, ..., N_w$ represents the number of weights and biases in the neural network, y_{true} is the vector of target class label associated with each sample, y_{pred} is the vector of the predicted probabilities associated with each class label derived from the neural network, and γ and ζ are the regularization hyper-parameters.

The minimization of $S(w)$ with respect to the weights and biases is a nonlinear optimization problem solved using the Levenberg-Marquardt algorithm, in which the gradient and the Hessian matrix of $S(w)$ are calculated using the backpropagation method. Assuming the weights and the data have Gaussian prior probability distributions, the regularization hyper-parameters, γ and ζ, are updated by maximizing their posterior probability distribution given the data, which is equivalent to maximizing the likelihood of evidence by Bayes' Theorem. Within each epoch, the cost function $S(w)$ is minimized with respect to w, and the likelihood of evidence is maximized with respect to γ and ζ. This is carried out iteratively until self-consistency is achieved, at which point the optimal distribution of weights and biases in the Bayesian regularized artificial neural network is obtained. Bayesian regularized artificial neural networks can effectively avoid over-training and over-fitting. Evidence procedures provide an objective Bayesian criterion for early stopping and remove the need for lengthy cross validations. Furthermore, the less relevant weights are turned off during the training process and Bayesian regularization effectively prunes the network [15]. Training and testing accuracies are calculated using the ratio between number of correctly classified samples and total number of samples in the training and testing sets, respectively. To develop a neural network detection model, closed-loop measurement data, both y_c and y_a, under two-tier control are collected. For better detection accuracy, training data needs to be collected starting at a broad range of initial conditions within the stability region Ω_ρ, such that various state evolutions under different operating conditions are covered. Full state measurements are recorded along the time-varying trajectory, and the Lyapunov function $V(x)$ is computed. As it captures the dynamic features of all states, $V(x)$ is an effective one-dimensional input feature for the attack detection problem. To ensure training accuracy, equal number of samples within each class are collected, with each sample corresponding to a different set of initial conditions for the closed-loop system simulation.

After data collection and adequate training, the NN detector is implemented online with the process controlled by the two-tier control system. The feed-forward NN model is a static model receiving inputs of fixed dimension, N_T, which is the length of the time-varying trajectory. Therefore, the detection window of the NN detector while implemented online also matches the trajectory length of the training data, N_T. The detector is activated every time full state measurements become available, and uses a moving horizon detection window, receiving latest sequences of $x(t_k)$ of fixed length N_T. Moreover, as the NN detector does not have perfect classification accuracy, false alarms may occur where large oscillatory data within normal ranges may be misclassified as a cyber-attack. To reduce false alarm rates, a sliding alarm verification window is also implemented, where the number of positive attack detections within this window

need to surpass a threshold before a cyber-attack alarm is confirmed. The size of this verification window and the threshold value are determined based on the closed-loop evolution of the process, as these two parameters have a direct impact on the detection time and alarm rate. If sensor isolation is required, then all upper-tier state trajectories need to be fed into the neural network individually, as the output class labels depend on changes in each sensor. Each sample consists of a two-dimensional matrix $n_x \times N_T$, where n_x is the full state dimension, and N_T is the length of each state trajectory within the training simulation period. Similarly, equal number of samples in each class (i.e., one class representing each networked sensor measurement being attacked) are collected for various initial conditions in the stability region. These samples are used to train a sensor-isolation NN algorithm outputting multiple classes, where each class corresponds to each of the networked sensors being attacked. During online implementation, given that the system is under attack, this sensor isolator examines all states in the most recent N_T sampling periods and outputs which sensor is experiencing abnormalities.

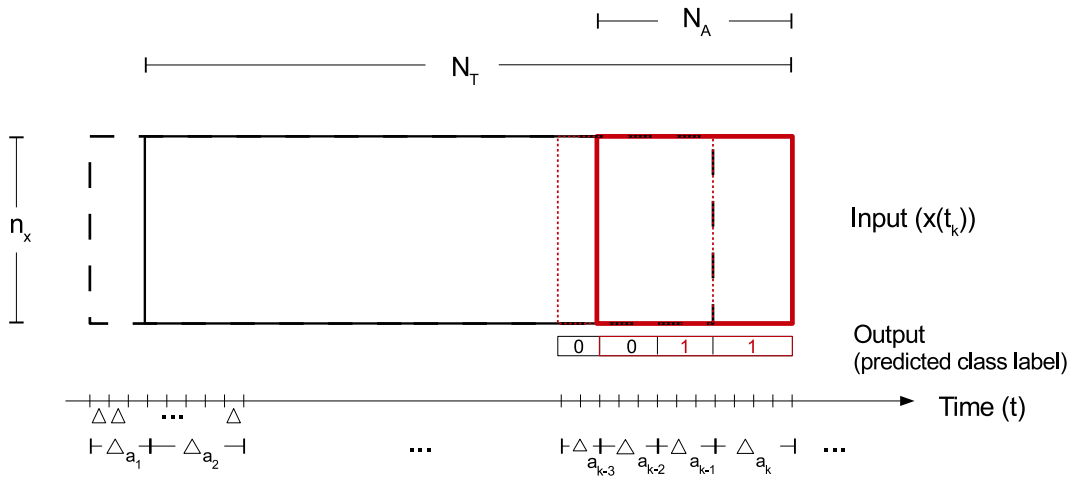


Figure 2.2: Online implementation of NN detector with moving horizon detection window N_T and alarm verification window N_A, where the detector reads past inputs $x(t_k)$ of length N_T and dimension n_x, and computes the predicted class label.

Remark 2.2. *In the sliding verification window, we examine the number of positive detections out of the total number of detector activations; the two parameters, size of verification window and threshold for alarms, are different from a threshold number that is often examined in statistical methods such as Cumulative Sum. Long-term attacks such as geometric and surge attacks may*

be designed such that the cumulative error of the attacked measurements stay just below the statistical detection threshold, thus remain undetectable. However, they are detectable by neural network detectors, given that the extent and pattern of the attacked measurements are similar to the anomalous behavior learned by the neural net during training. Furthermore, neural network detectors trained with noisy sensor data are able to differentiate cyber-attacks from normal device fluctuations. However, in the case that measurement noise is so significant that it is similar to attacked oscillations (like in a replay attack), then the neural network detector may flag these noisy measurements as replay cyber-attacks. If significant noise is bound to be observed, then new neural network detectors can be readily trained based on these new noisy data to reflect the changed nominal operating conditions.

2.3.4 Mitigation Measures via Control System Reconfiguration

Upon detection of an attack on the sensors providing networked asynchronous state measurements to the two-tier control system, the control system reconfiguration logic allows for two mitigation plans. First, the control system can deactivate the upper-tier controller completely and operate the system under the stabilizing lower-tier control system only, which uses cyber-secure, dedicated sensor measurements. Since the lower-tier controllers are capable of driving the process to its operating steady-state with secure continuous measurements, the effect of the cyber-attacks is fully eliminated in the closed-loop system in this case and the process is stabilized to the operating steady-state. Second, if a sensor isolation detector is also implemented, it will be activated once a sensor attack is verified. Subsequently, the upper-tier controller can choose to switch the compromised sensor to its redundant back-up sensor with secure readings. By abandoning the corrupted sensor and using its back-up sensor using a secure sensor-controller communication, the upper-tier controller remains functional and is able to drive the process to its steady-state with better closed-loop performance.

In the extreme case that both continuous and asynchronous sensor measurements are attacked, the upper-tier controller will be shut off and the lower-tier controllers will reroute their continuous measurement signals from the corrupted sensors to their respective secure back-up sensors. The robustness of the proposed two-tier control architecture against intelligent cyber-attacks is demonstrated in Section 2.4 below through a reactor-reactor-separator process.

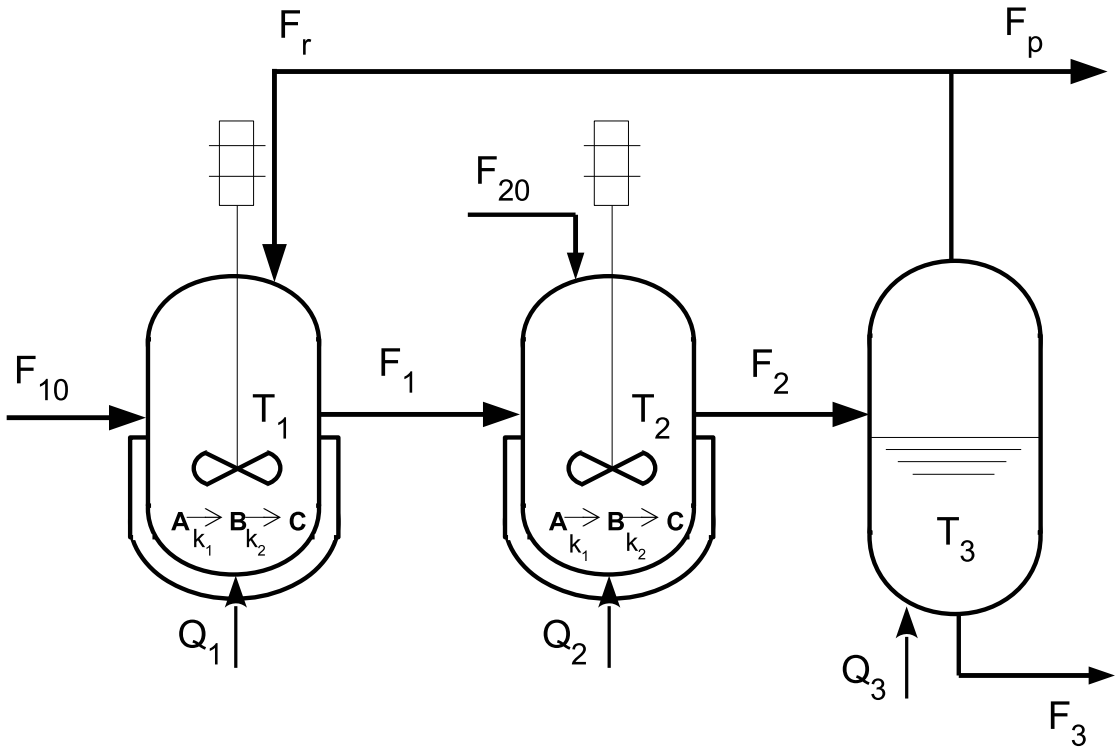


Figure 2.3: Process schematic consisting of two CSTRs and a flash drum separator.

2.4 Application to a Reactor-Reactor-Separator Process

2.4.1 Process Description and Control System Design

To simulate a chemical process application where multiple manipulated inputs are regulated by both the upper-tier and lower-tier controllers, a process network consisting of two CSTRs followed by a flash tank separator is considered [137]. A schematic diagram of this process network can be found in Fig. 2.3. Two reactions in series take place ($A \rightarrow B \rightarrow C$) in both reactors, and the overhead vapor from the flash tank is recycled to the first CSTR. All three vessels are assumed to have constant holdup.

Using mass and energy balances, the process model can be obtained, which includes 9 nonlinear ordinary differential equations as shown below:

$$\frac{dx_{A1}}{dt} = \frac{F_{10}}{V_1}(x_{A10} - x_{A1}) + \frac{F_r}{V_1}(x_{Ar} - x_{A1}) - k_1 e^{\frac{-E_1}{RT_1}} x_{A1} \tag{2.11a}$$

$$\frac{dx_{B1}}{dt} = \frac{F_{10}}{V_1}(x_{B10} - x_{B1}) + \frac{F_r}{V_1}(x_{Br} - x_{B1}) + k_1 e^{\frac{-E_1}{RT_1}} x_{A1} - k_2 e^{\frac{-E_2}{RT_1}} x_{B1} \tag{2.11b}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{(-\Delta H_1)}{\rho C_p} C_M k_1 e^{\frac{-E_1}{RT_1}} x_{A1} + \frac{(-\Delta H_2)}{\rho C_p} C_M k_2 e^{\frac{-E_2}{RT_1}} x_{B1} + \frac{Q_1}{\rho C_p V_1} + \frac{F_r}{V_1}(T_3 - T_1) \tag{2.11c}$$

$$\frac{dx_{A2}}{dt} = \frac{F_1}{V_2}(x_{A1} - x_{A2}) + \frac{F_{20}}{V_2}(x_{A20} - x_{A2}) - k_1 e^{\frac{-E_1}{RT_2}} x_{A2} \tag{2.11d}$$

$$\frac{dx_{B2}}{dt} = \frac{F_1}{V_2}(x_{B1} - x_{B2}) + \frac{F_{20}}{V_2}(x_{B20} - x_{B2}) + k_1 e^{\frac{-E_1}{RT_2}} x_{A2} - k_2 e^{\frac{-E_2}{RT_2}} x_{B2} \tag{2.11e}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2}(T_{20} - T_2) + \frac{(-\Delta H_1)}{\rho C_p} C_M k_1 e^{\frac{-E_1}{RT_2}} x_{A2} + \frac{(-\Delta H_2)}{\rho C_p} C_M k_2 e^{\frac{-E_2}{RT_2}} x_{B2} + \frac{Q_2}{\rho C_p V_2} + \frac{F_1}{V_2}(T_1 - T_2) \tag{2.11f}$$

$$\frac{dx_{A3}}{dt} = \frac{F_2}{V_3}(x_{A2} - x_{A3}) - \frac{F_r + F_p}{V_3}(x_{Ar} - x_{A3}) \tag{2.11g}$$

$$\frac{dx_{B3}}{dt} = \frac{F_2}{V_3}(x_{B2} - x_{B3}) - \frac{F_r + F_p}{V_3}(x_{Br} - x_{B3}) \tag{2.11h}$$

$$\frac{dT_3}{dt} = \frac{F_2}{V_3}(T_2 - T_3) + \frac{Q_3}{\rho C_p V_3} + \frac{(F_r + F_p) C_M}{\rho C_p V_3}(x_{Ar}\Delta H_{vapA} + x_{Br}\Delta H_{vapB} + x_{Cr}\Delta H_{vapC}) \tag{2.11i}$$

where the state variables include the temperatures of the three vessels T_1, T_2, T_3, respectively, which are measured securely and continuously, and the mass fractions of species A and B in the three vessels x_{A1}, x_{A2}, x_{A3} and x_{B1}, x_{B2}, x_{B3}, whose measurements are available at asynchronous time instants and are sent to the upper-tier control system over a digital network that may be subjected to cyber-attacks. The upper-tier control system involves an LMPC that receives both asynchronous and continuous state measurements, and it is executed when full state information becomes available. Each of the three vessels has an external heat input. Three PI controllers are used to manipulate the heat inputs to the three vessels, Q_1, Q_2 and Q_3, each to regulate vessel temperature at a desired set-point value, and the LMPC manipulates the feed stream flow rate to second CSTR, F_{20}, to improve the speed of the closed-loop response. Assuming that there is negligible reaction in the separator tank and the relative volatility of each species remains constant within the operating temperature range, the composition of the recycle stream are: $x_{Ar} = \frac{\alpha_A x_{A3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}}$, $x_{Br} = \frac{\alpha_B x_{B3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}}$, $x_{Cr} = \frac{\alpha_C x_{C3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}}$, where α represents the constant relative volatility of each species. Each of the six mass fraction measurements can be subject to the cyber-attacks, which are designed based on the current value of the true states at the time the attack occurs, as discussed in Section 2.3. With the integration of a machine-learning-based cyber-attack detector, the control objective is to track all 9 states to an unstable equilibrium point while meeting all imposed constraints and staying immune to intelligent cyber-attacks. All process

Table 2.1: Values and descriptions of process parameters and steady-states of state and input variables.

Parameter/Value	Description
$F_{10} = 5.04\ m^3/hr$	Feed flow rate of CSTR 1
$F_r = 50.4\ m^3/hr$	Recycle stream flow rate
$F_p = 5.04\ m^3/hr$	Purge stream flow rate
$T_{10} = 300\ K$, $T_{20} = 300\ K$	Feed temperatures of CSTR 1 & 2
$V_1 = 1.0\ m^3$, $V_2 = 0.5\ m^3$, $V_3 = 1.0\ m^3$	Volume of 3 vessels
$k_1 = 9.972 \times 10^6\ h^{-1}$,	Pre-exponential factors for reactions 1 & 2
$k_2 = 9.36 \times 10^6\ h^{-1}$	
$E_1 = 5.0 \times 10^4\ kJ/kmol$,	Activation energy for reactions 1 & 2
$E_2 = 6.0 \times 10^4\ kJ/kmol$	
$\Delta H_1 = -1.2 \times 10^5\ kJ/kmol$,	Heat of reaction for reactions 1 & 2
$\Delta H_2 = -1.4 \times 10^5\ kJ/kmol$	
$\Delta H_{vapA} = -3.53 \times 10^4\ kJ/kmol$,	Heat of vaporization for A, B, C
$\Delta H_{vapB} = -1.57 \times 10^4\ kJ/kmol$,	
$\Delta H_{vapC} = -4.068 \times 10^4\ kJ/kmol$	
$C_p = 4.2\ kJ/(kg\ K)$	Heat capacity
$R = 8.314\ kJ/(kmol\ K)$	Gas constant
$\rho = 1000\ kg/m^3$	Liquid solution density
$\alpha_A = 3.5$, $\alpha_B = 1.0$, $\alpha_C = 0.5$	Relative volatility of A, B, C
$C_M = 2\ kmol/m^3$	Total molar concentration
$x_{A1s} = 0.1762$, $x_{A2s} = 0.1965$, $x_{A3s} = 0.0651$,	Steady-state values of state variables
$x_{B1s} = 0.6731$, $x_{B2s} = 0.6536$, $x_{B3s} = 0.6703$,	
$T_{1s} = 480.32\ K$, $T_{2s} = 472.79\ K$, $T_{3s} = 474.89\ K$	
$Q_{1s} = 2.9 \times 10^9\ kJ/hr$, $Q_{2s} = 1.9 \times 10^9\ kJ/hr$,	Steady-state values of input variables
$Q_{3s} = 2.9 \times 10^9\ kJ/hr$, $F_{20s} = 5.04\ m^3/hr$	

parameter values, the steady-state values, and the corresponding steady-state input values are given in Table 7.1.

Deviation variables are used to present the simulation results, where the state vector and the input vector are represented as the difference between their values and their steady-states. By using deviation variables, the equilibrium point of the process (i.e., the operating steady-state) is at the origin of the state space. The input variables in deviation variable form are subject to the following operating constraints: $-4.04\ m^3/hr \leq \Delta F_{20} \leq\ 3.96\ m^3/hr$, $|\Delta Q_1| \leq\ 5 \times 10^7\ kJ/hr$, $|\Delta Q_2| \leq\ 5 \times 10^7\ kJ/hr$, $|\Delta Q_3| \leq\ 5 \times 10^7\ kJ/hr$.

Classical controllers are used in the lower-tier control system; specifically, proportional-integral (PI) controllers are used. The formulation of PI controller is presented

as below:

$$u_{c_i}(t) = K_{c_i}(e_{c_i}(t) + \frac{1}{\tau_i}\int_0^t e_{c_i}(\tau)d\tau), \;\; e_{c_i}(t) = y_{c_i}^{REF}(t) - y_{c_i}(t) \tag{2.12a}$$

where $e_{c_i}(t)$ is the error between the measured output values y_{c_i} and their operating set-points $y_{c_i}^{REF}$ (defined based on the operating steady-state), and K_{c_i} and τ_i are the proportional gain and integral time constant of each PI controller $i = 1,2,3$, respectively. In order to ensure closed-loop stability under PI control, K_{c_i} and τ_i are selected by first linearizing the model in Eq. 8.1 around the steady-state, and then assessing the eigenvalues of the linearized model $\dot{x} = Ax + Bu_c$. The proportional gain and time constant of the three PI controllers are chosen to be $[K_{c_1}\ K_{c_2}\ K_{c_3}]^T = [-8\times10^5,\ -8\times10^5,\ -8\times10^5]^T$ and $[\tau_1\ \tau_2\ \tau_3]^T = [5000,\ 5000,\ 5000]^T$, respectively. An initial set of the PI controller parameters are determined using the Cohen-Coon tuning method, and then further optimized from closed-loop simulations, to make sure that the closed-loop response is smooth with reasonable control actions. With these tuning parameters, closed-loop stability under P-only control is ensured as the eigenvalues of the linearized model are $\Lambda = [-2.599,\ -56.97,\ -99.98-26.28i,\ -99.98+26.28i,\ -27.93-149.2i,\ -27.93+149.2i,\ -257.8-26.93i,\ -257.8+26.93i,\ -758.8]$, all of which having negative real parts, and the integral term aims to eliminate the offset while having minimal impact on the control action. An anti-windup mechanism is also implemented inside each PI controller to avoid integral wind-up effects which involves eliminating the integral term when the control action hits constraints. The upper-tier LMPC used in this simulation adopts the formulation shown in Eq. 2.3. The objective function used in the optimization problem of LMPC is defined by a positive definite function, $L(x,u_a) = x^TQ_cx + u_a^TR_cu_a$, where R_c and Q_c are weighting matrices to penalize u_a and x, and have the following values: $R_c = 1.0$ and $Q_c = diag([5000,\ 10,\ 0.001,\ 5000,\ 10,\ 0.001,\ 5000,\ 10,\ 0.001])$. The quadratic control Lyapunov function used in the contractive constraints of LMPC has the form $V(x) = x^TPx$, where P is a positive definite matrix: $P = diag([3228.31,\ 220.79,\ 4.334\times10^{-4},\ 2576.72,\ 233.80,\ 4.474\times10^{-4},\ 23675.92,\ 222.77,\ 4.434\times10^{-4}])$. The family of piece-wise constant function $S(\Delta)$ which u_a belongs to uses a sampling period of $\Delta = 0.02\ hr$, and the prediction horizon of the LMPC is $N = 10$. The nonlinear optimization problem of LMPC is solved using the OPTI-Toolbox in MATLAB. To numerically simulate the dynamic process model in Eq. 2.11, explicit Euler method is used with an integration step of $h_c = 10^{-4}hr$. The time sequence at which asynchronous measurements are sampled and received by the upper-tier controller is modeled after a lower-bounded random Poisson process, with each unequal interval

between two consecutive asynchronous measurements being at least $\Delta_{a_k} \geq \Delta$ for all $k \in [1, N_T]$. The sequence of asynchronous intervals used in this simulation in which the LMPC calculations are executed is as follows: $\Delta_a = [0.04,\ 0.08,\ 0.1,\ 0.06,\ 0.12,\ 0.08,\ 0.02]$ for every 1.5 *hr*; alternative calculations of the asynchronous time instants may be considered with similar conclusions. After a simulation grid search, we use $\rho = 120$ as a level set of Lyapunov function to characterize the stability region and $\rho_{min} = 0.1$ to ensure convergence close to the steady-state. The safe operating envelope of the 9 states in deviation variable form is as follows: $x_l = [-0.1763,\ -0.6731,\ -50,\ -0.1965,\ -0.6536,\ -50,\ -0.0651,\ -0.6703,\ -50]^T$ denotes the lower bounds of the states and $x_u = [0.7237,\ 0.2269,\ 50,\ 0.7035,\ 0.2464,\ 50,\ 0.8349,\ 0.2297,\ 50]^T$ denotes the upper bounds of the states. The stability region and the operating envelope are key parameters to generating intelligent cyber-attacks. The simulation period used for collecting training data is 3 *hr*, within which the lower-tier PI controllers are executed 150 times, and the upper-tier LMPC is executed 42 times. With the upper-tier controller receiving full-state measurements 42 times, the time-varying trajectories of state measurements have a length of $N_T = 43$, accounting for the initial condition measurements. Closed-loop simulations under two-tier and under PI-only control are carried out to compare the closed-loop performances; the initial conditions used to evaluate the performance metrics are $x_0 = [0.0176,\ 0.067299,\ 48.032,\ 0.0197,\ 0.0654,\ 47.279,\ 0.006499,\ 0.067,\ 47.489]$. Performance metrics in terms of settling time and normalized cumulative mean squared error along the state trajectories are calculated for closed-loop control under only lower-tier PI controllers, and under the two-tier LMPC/PI control scheme. It is shown that it takes 2.46 *hr* for lower-tier PI controllers, and 0.6 *hr* for two-tier LMPC/PI to settle to the operating steady-state. The normalized cumulative mean squared errors are 4.1203 and 0.8014 for lower-tier PI and two-tier LMPC/PI, respectively. The two-tier control architecture achieves significantly better closed-loop performance by stabilizing the process within shorter time and eliminating process overshoots and offset effectively.

2.4.2 Cyber-attacks and Detector Training

Min-max attacks are used to train the neural-network-based detector with and without sensor noise. If the neural network detector is trained with only one type of attack, the resulting output will have 2 classes – attacked and not attacked. In addition, replay attacks are also used to train a neural network detector capable of identifying the type of attack, where the output classes consist of 3 labels: not attacked, attacked by min-max attacks, and attacked by replay attacks. In the first 5

sampling steps, more extreme oscillations with larger magnitudes in state feedback are observed. Therefore, these aggressively oscillatory measurements with length $L_a = 5$ are recorded and used as replay attacks. Other attacks with varying lengths can be introduced at random time instants between $i_0 \in [6, 42]$ to simulate cyber-attacks of various durations and occurring at various times during operation. With extensive closed-loop simulations, equal number of samples are collected for each output class, with each sample either consisting of a 1×43 array of $V(x)$ values (in attack identification), or a 9×43 matrix of x values along the dynamic trajectory (in sensor isolation), where the 9×43 matrix in each sample is then collapsed into a 1×387 array to be fed into the feed-forward neural network detector. Four NN detectors are trained to carry out detection: 1) a 2-class model with nominal operation under min-max attack (12000 samples per class label, training time 24.05 seconds), 2) a 2-class model with noisy sensors under min-max attack (1044 samples per class label, training time 4.332 seconds), 3) a 3-class model with noisy sensors and 2 attack types – min-max and replay (1044 samples per class label, training time 5.265 seconds), and 4) a 6-class model with noisy sensors under min-max attack for corrupted sensor isolation (2800 samples per class label, training time 6211.52 seconds). To train the compromised sensor isolation detector, we also used a min-max attack to simulate the abnormal behaviors. Noisy sensors are simulated by adding bounded Gaussian white noise on each sensor. The lower and upper bounds of the sensor noises are as follows: $|w_1| \leq 7.5 \times 10^{-5}$, $|w_2| \leq 5.5 \times 10^{-5}$, $|w_3| \leq 0.032\ K$, $|w_4| \leq 7.5 \times 10^{-5}$, $|w_5| \leq 5.5 \times 10^{-5}$, $|w_6| \leq 0.032\ K$, $|w_7| \leq 3.5 \times 10^{-5}$, $|w_8| \leq 5.5 \times 10^{-5}$, $|w_9| \leq 0.032\ K$. These Gaussian noise distributions have a mean of $\mu = 0$ and standard deviations $\sigma_1 = \sigma_4 = 0.0002$, $\sigma_2 = \sigma_5 = \sigma_8 = 0.001$, $\sigma_3 = \sigma_6 = \sigma_9 = 0.1\ K$, and $\sigma_7 = 0.0001$. Feed-forward neural networks with two hidden layers having 12 and 10 neurons, respectively, are built using the MATLAB Machine Learning and Deep Learning Toolboxes. Both hidden layers use a *tansig* activation function, which is in the form $g_{1,2}(z) = \frac{2}{1+e^{-2z}} - 1$, and is commonly known as the hyperbolic tangent sigmoid transfer function. The output layer uses a $softmax$ function to provide a predicted probability of the class labels, which is in the form of $g_3(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{H} e^{z_i}}$ where H denotes the number of class labels. The NN detector trained with nominal conditions has a training accuracy of 99.6% and a testing accuracy of 92.2%, while the NN detector trained with noisy sensors achieves a training accuracy of 99.9% and testing accuracy of 100%. The NN algorithm trained with noisy sensors achieves a higher accuracy than the nominal case because the addition of noise contributes more variance to the training dataset, thereby making the learning process harder and yielding a more robust NN detector. Moreover, the training and testing accuracy of the NN detector trained with noisy sensors under two types of cyber-attacks are 98.2% and 91.4%,

respectively, and the NN algorithm to isolate the compromised noisy sensor achieves a training and testing accuracy of 99.6% and 99.0%, respectively.

2.4.3 Cyber-attack Detection Results

The NN detectors are implemented online with the process operated under two-tier control with initial conditions at the operating steady-state; i.e., $x_0 = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]^T$ (the conclusions are similar for other initial conditions in Ω_ρ). Therefore, attacks are introduced when the process is stabilized at its operating steady-state. As the NN detectors are trained using a fixed input dimension of 43 (with 42 sampling steps), to ensure that input data with sufficient length is collected, the detector is activated at time instant $k = 42$ in the asynchronous time sequence, which corresponds to real time $t = 3.0\ hr$. The detector reads state measurements in the previous 42 sampling periods, analyzes their behaviors, and computes an output on which class these time-series data resembles. The window of this fixed-length segment of time-sequence data rolls forward in time as the upper-tier LMPC and the attack detector are executed in real time. The alarm verification window is chosen to be three sampling periods of the upper-tier LMPC, where two positive detections within every three consecutive sampling steps will confirm the presence of an attack. Once an attack is confirmed, at the same time instant, the detection alarm will be triggered and the LMPC will be deactivated. Furthermore, to examine whether the detector will misclassify not-attacked signals as being under attack, attacks are introduced a few sampling periods after the detector has been activated at $t = 3.0\ hr$, such that the first few outputs by the detector are based on normal operation data. Cyber-attacks with a duration of $L_a = 40$ sampling periods are introduced at time instant $i_0 = 45$, which corresponds to $t = 3.22\ hr$; thus, the compromised sensor will stay corrupted until the end of the $6 - hr$ simulation period. To illustrate the pattern and effect of the four cyber-attack types, Fig. 2.4 shows the true state values and the sensor values of state 1 when min-max, replay, geometric, and surge attacks target only the sensor measuring mass fraction x_{A1} with bounded noise. Although Fig. 2.4 only shows the true state progression of state 1, all 9 states experience similar deviating patterns after the cyber-attacks. Under min-max attack, the true state settles at an offset of similar magnitude as the initial jump. Replay attack results in aggressive oscillations in true plant states around an offset. Geometric attacks drive process states increasingly away from the operating steady-state before reaching an offset due to the increasing magnitude of the attack with time until the attack reaches the boundary of the stability region. Surge attack causes an initial jump similarly seen in min-max attacks; with the reduction of attack severity, states are driven closer to the setpoint, but still reach an offset that is smaller than that in min-max.

The upper-tier LMPC receives falsified information on the values of these process states, and in turn, computes a control action that is unable to drive the true states back to the steady-state. States do not continue to diverge and do not exit the stability region, but instead settle at an offset value, due to the stabilizing contributions from the three lower-tier PI controllers, which use secure sensor measurements. Regardless, the attack has successfully targeted the system and the two-tier control system fails to drive the states back to their operating steady-states without using any data-based detection algorithms.

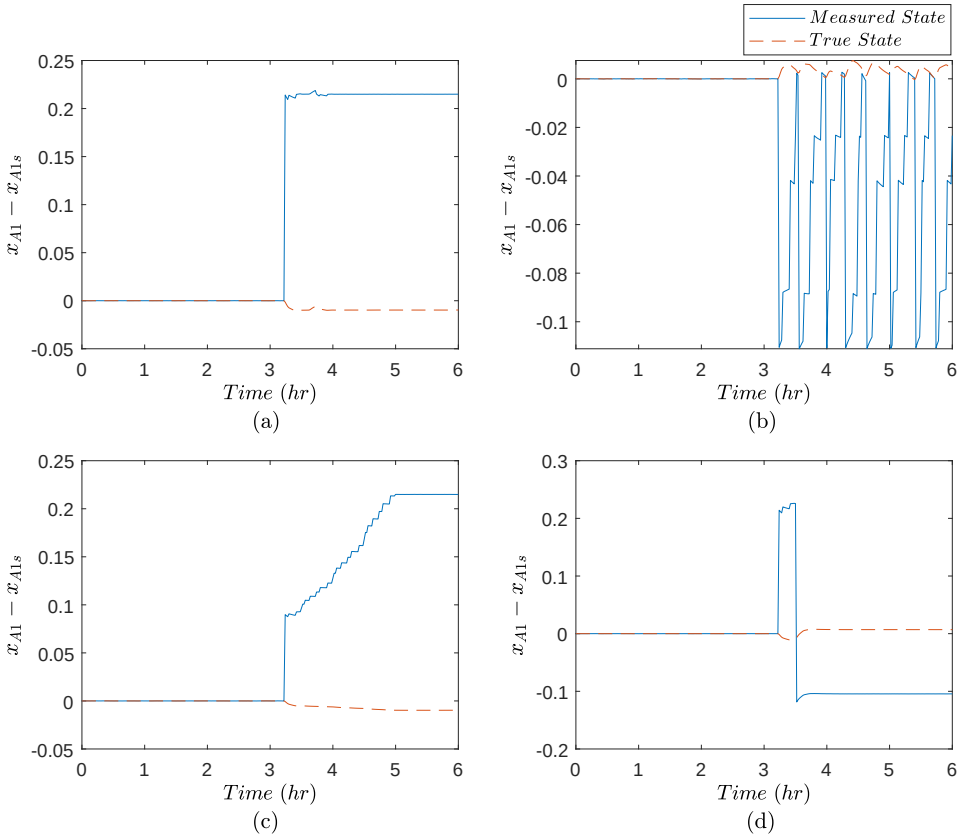


Figure 2.4: True and measured values of x_{A1} in deviation variable form without detection or mitigation mechanisms when (a) min-max, (b) replay, (c) geometric, and (d) surge cyber-attacks are introduced at 3.22 *hr* on the concentration sensor measuring x_{A1}.

When the trained NN detectors are applied during online operation, time delays are observed due to the configured alarm verification window, which requires at least two positive detections out of three detection instances to confirm an attack. The time delay is defined as the number of sampling steps between when the attack is inserted and when the attack is confirmed. In the cases of the first two detectors trained using min-max attack only (i.e., nominal and with noise operations), replay, geometric and surge attacks are unknown attacks which have not been learned

by the NN detector. All four types of cyber-attacks, despite the latter three being unknown to the NN detector, are captured by the NN detector trained under nominal condition. The NN detection algorithm detects min-max, replay, and surge attacks successfully with a time delay of 1 sampling period. This is because two out of three detections need to be positive to confirm a detection; in other words, as soon as two consecutive positive detections occur, the detection is confirmed. Therefore, the detection of these cyber-attacks is delayed by 1 sampling period, at which time the second consecutive positive detection is received by the control system. A time delay of 2 sampling periods is observed when a geometric attack is introduced due to the initial small magnitude of change induced, therefore causing the NN detector a delay in predicting the correct class label. As time progresses, the attack increases exponentially towards a point where the deviation is on par with the other three attacks, at which point the detector captures the irregular deviation. The potential time delay of NN detectors trained with min-max attacks in detecting geometric attacks will vary depending on the geometric parameters, i.e., β and α in Eq. 4.12. Meanwhile, the NN detector trained with noisy sensor measurements is able to detect min-max, geometric, and surge attacks successfully, but with a time delay of 7 sampling periods when the geometric attack is introduced. Moreover, this detector fails to detect replay attacks due to the oscillatory nature of the replay signals. Unlike the other 3 attacks where the attacked measurement stays at the attack target for at least 2 sampling periods, replay attacks oscillate at every sampling period, which is different from the min-max behavior that this NN detector is trained based on. Given the relatively smaller magnitude of the oscillations (not reaching the min-max window) and oscillatory behavior of replay attacks, the NN detector trained with added noises is not able to differentiate replay attacks from sensor noise. Thus, a third NN detector is trained, where both replay and min-max attacks are accounted in the training process. This detection algorithm outputs 3 classes, where min-max and replay attacks are classified correctly and the detection is confirmed after a time delay of 1 sampling period, and geometric and surge attacks are classified as replay attacks with the detection confirmed after 1 sampling period.

Cyber-attacks could target multiple sensors at once, and the detection algorithms are tested on these cases where more than one sensor could be under attack. We first consider the extreme case where min-max cyber-attacks are applied on all 9 sensors to simulate the impact of cyber-attacks when all state measurements are compromised without using any online detectors; this scenario allows to demonstrate the value of the proposed two-tier control architecture. The true state trajectories are shown in Fig. 2.5 where a min-max attack is introduced at 3.22 hr with a duration covering until the end of the simulation period. With the continuous temperature measurements

also under cyber-attack, closed-loop stability under the lower-tier controllers is no longer achieved. As a result of the cyber-attack, the true state evolution exits the stability region when no detection algorithms are being used; moreover, the mass fractions of species A and the temperatures in all three vessels exceed their operating boundaries, violating the safety limits on their states in deviation variable form. Under the circumstance that continuous temperature measurements are jeopardized, the only cyber-attack countermeasure is to reroute measured temperature signals received by lower-tier controllers from the corrupted sensors to a new set of redundant sensors with secure readings. This extreme scenario demonstrates the severity of the destabilizing impacts of cyber-attacks to all sensors, and thus, the necessity of having secure and reliable feedback measurements for the lower-tier controllers in order to maintain the robustness of the overall control architecture.

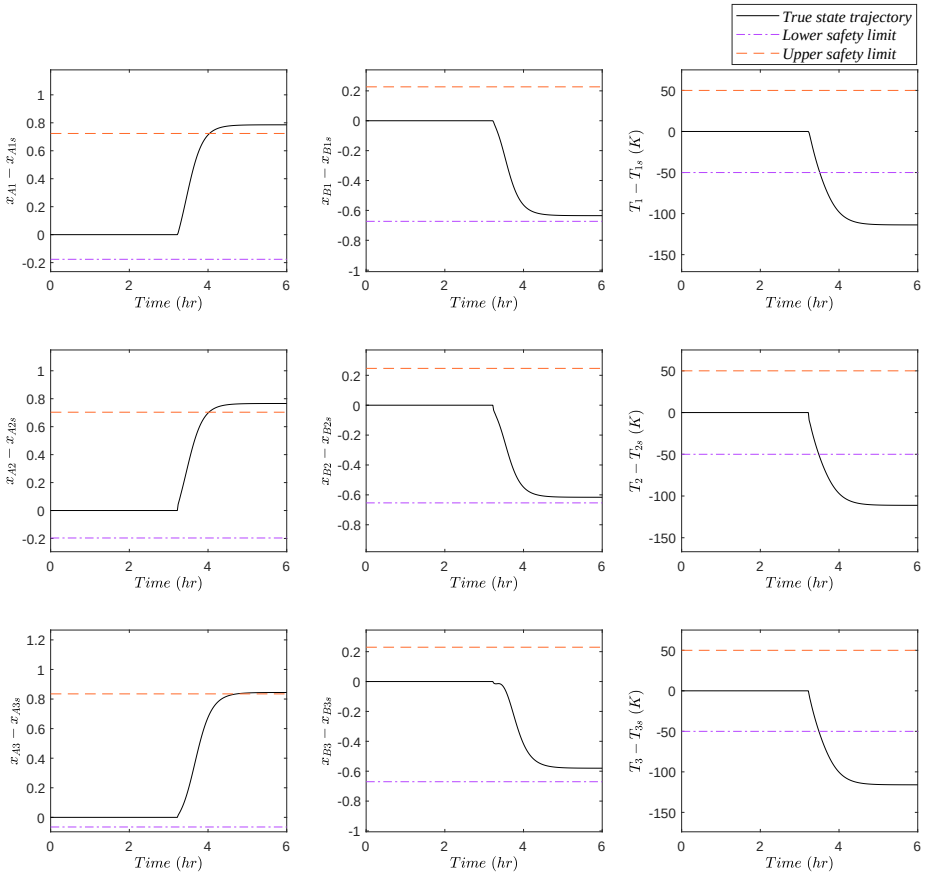


Figure 2.5: Evolution of true process states when min-max cyber-attacks on all 9 state measurement sensors are introduced at 3.22 *hr* when the process network operates under the two-tier control architecture but no detection or control system reconfiguration mechanisms are implemented.

To mitigate the impact of the cyber-attacks on all 9 sensors, the neural-network detection algorithm trained with noisy measurements and two cyber-attack types is applied online. With the alarm verification window to reduce false alarms, the min-max attack is introduced at 3.22 *hr* and the detection is confirmed at 3.28 *hr*, from which point the upper-tier LMPC is turned off and the continuous temperature measurements used by lower-tier PI controllers are obtained from a set of secure back-up sensors. By doing this, the closed-loop stability of the process under lower-tier control is re-established and the process is driven back to its operating steady-state, as shown in Fig. 2.6.

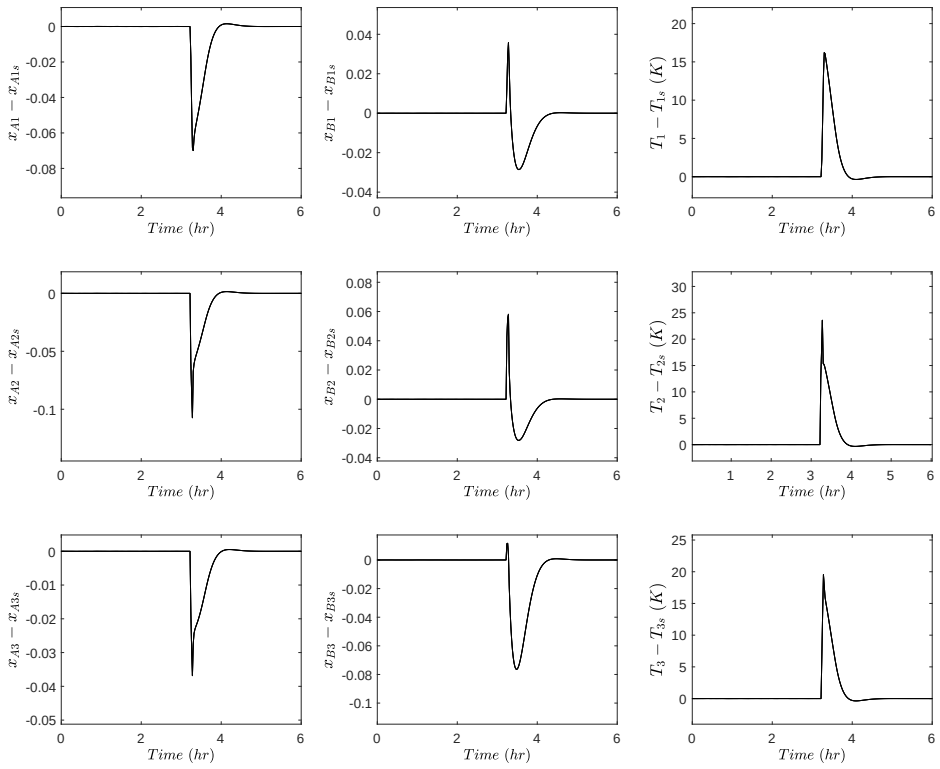


Figure 2.6: Evolution of true process states under min-max cyber-attacks on all nine state measurements. The min-max cyber-attacks are introduced at 3.22 *hr* and are detected at 3.28 *hr*, at which time the upper-tier LMPC is turned off and the temperature measurements used by the lower-tier PI controllers are taken from secure back-up temperature sensors and the process is driven back to the steady-state.

To ensure the robustness of the lower-tier controller against cyber-attacks, we now consider the case where only the networked mass fraction measurements fed into the upper-tier LMPC are attacked while the continuous temperature measurements used by both the lower-tier PIs and

the upper-tier LMPC remain secure. Therefore, with the stabilizing lower-tier PI controllers, the upper-tier LMPC can be turned off once the detection of an attack is confirmed (i.e., $u_a = 0$ for the remainder of the closed-loop simulation) such that the false control actions calculated by the upper-tier LMPC will not act as a disturbance to the closed-loop system. The effectiveness of this mitigation strategy is illustrated in our simulation results, where the true plant states are driven back to their operating steady-states using the stabilizing lower-tier PI controllers despite the sudden jumps or gradual deviations caused by cyber-attacks. The re-stabilized state trajectories after min-max, replay, geometric, and surge attacks are shown respectively in Figs. 2.7, 2.8, 2.9, and 2.10. When the attacks are introduced at time 3.22 *hr*, the detection algorithm confirms that the measured state trajectory is under attack at 3.28 *hr*, at which point the LMPC is turned off, and the process is re-stabilized to its operating steady-state using the lower-tier PI controllers. Despite the minor degradation in closed-loop performance with only lower-tier controllers, the reconfigured control system succeeds in maintaining closed-loop stability in the presence of cyber-attacks.

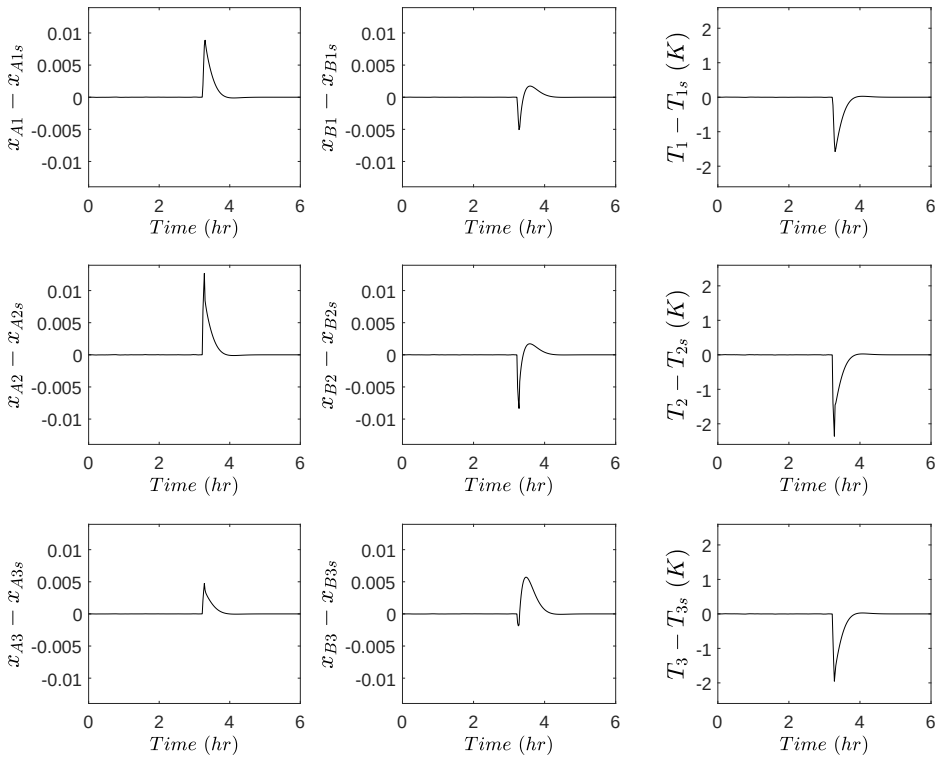


Figure 2.7: Evolution of true process states under min-max cyber-attacks on all six mass fraction sensors. The min-max cyber-attacks are introduced at 3.22 *hr* and are detected at 3.28 *hr*, at which time the upper-tier LMPC is turned off and the process is driven back to the steady-state under the lower-tier PI controllers.

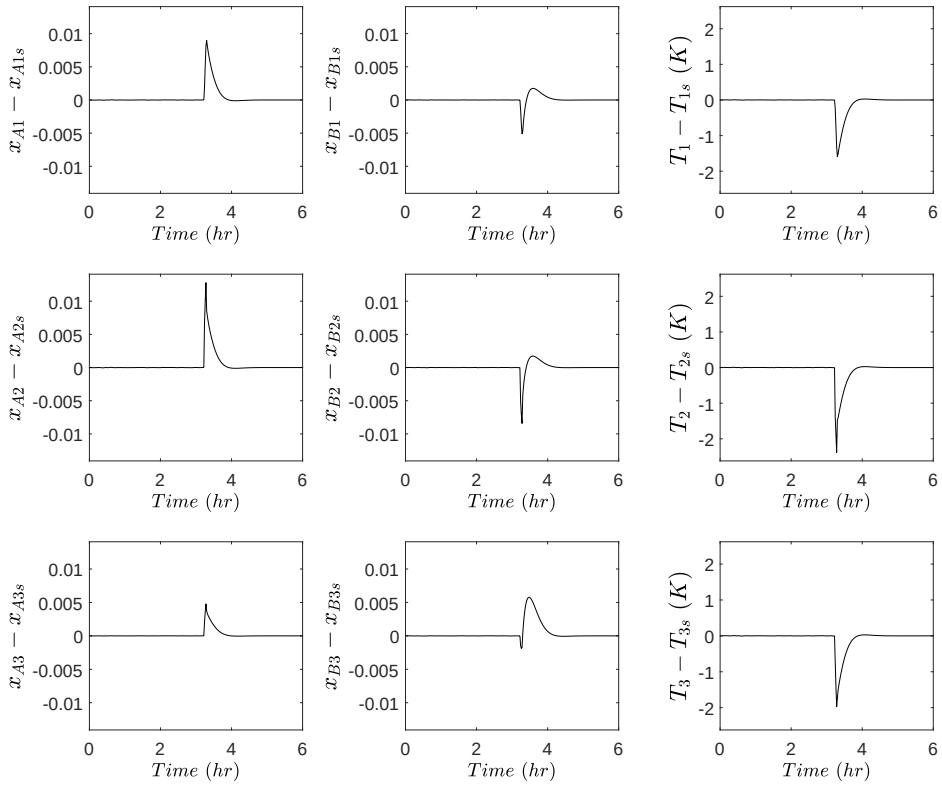


Figure 2.8: Evolution of true process states under replay cyber-attacks on all six mass fraction sensors. The replay cyber-attacks are introduced at 3.22 *hr* and are detected at 3.28 *hr*, at which time the upper-tier LMPC is turned off and the process is driven back to the steady-state under the lower-tier PI controllers.

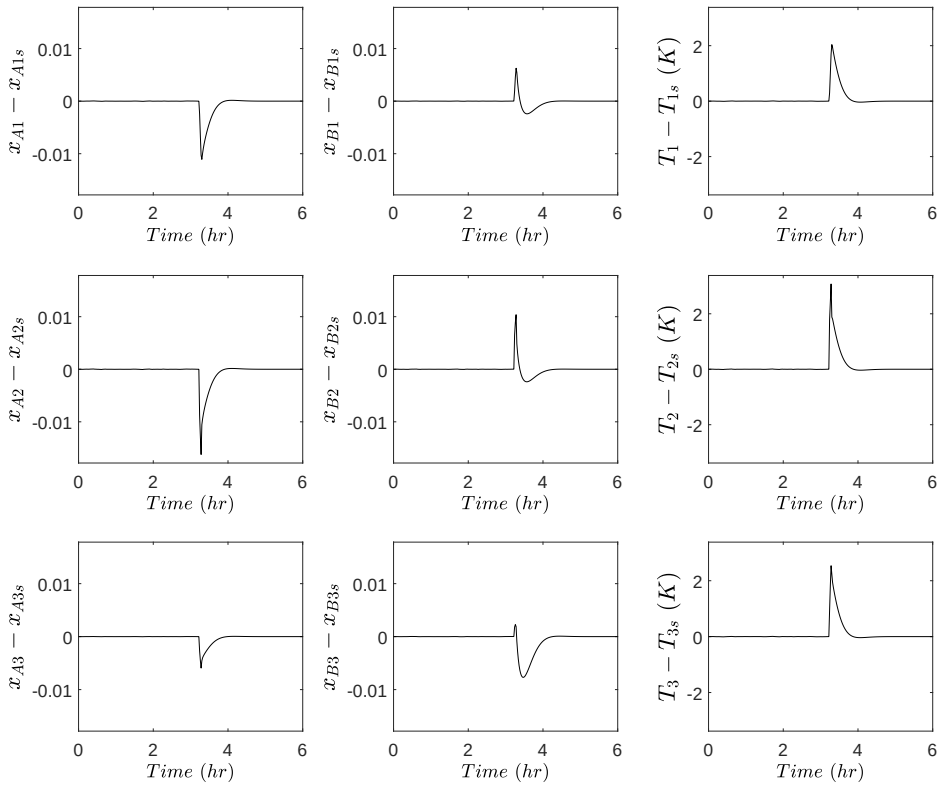


Figure 2.9: Evolution of true process states under geometric cyber-attacks on all six mass fraction sensors. The geometric cyber-attacks are introduced at 3.22 *hr* and are detected at 3.28 *hr*, at which time the upper-tier LMPC is turned off and the process is driven back to the steady-state under the lower-tier PI controllers.

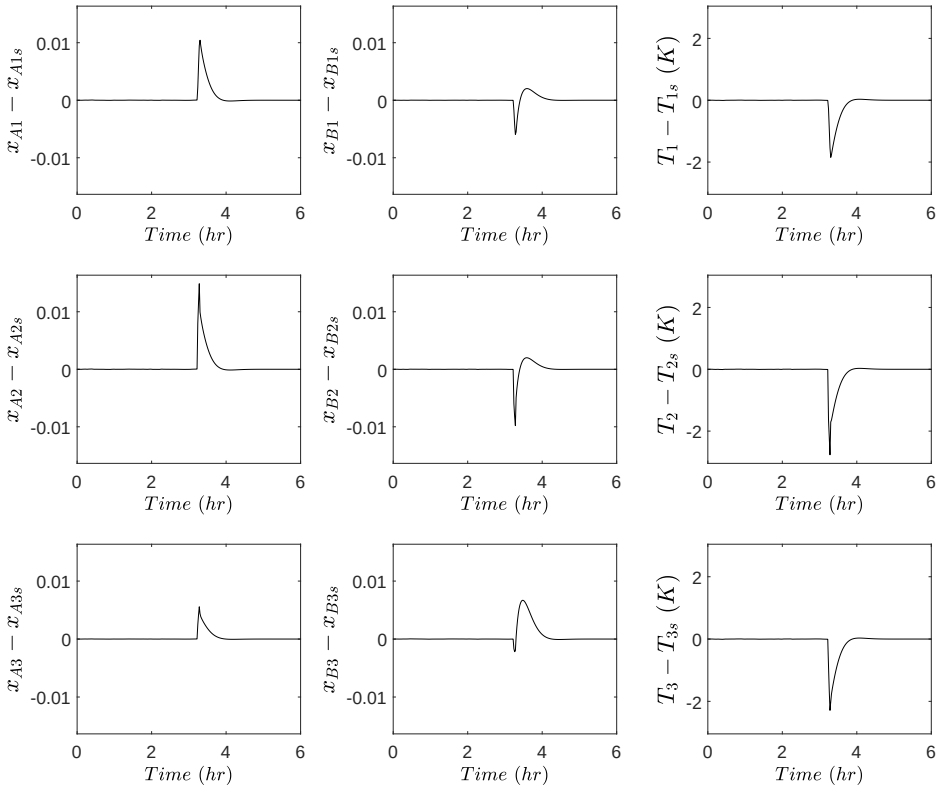


Figure 2.10: Evolution of true process states under surge cyber-attacks on all six mass fraction sensors. The surge cyber-attacks are introduced at 3.22 *hr* and are detected at 3.28 *hr*, at which time the upper-tier LMPC is turned off and the process is driven back to the steady-state under the lower-tier PI controllers.

Chapter 3

Cyber-attack Detection and Resilient Operation of Nonlinear Processes under Economic Model Predictive Control

This chapter proposes resilient operation strategies for nonlinear processes that are vulnerable to targeted cyber-attacks, as well as detection and handling of standard types of cyber-attacks. Working with a general class of nonlinear systems, a modified Lyapunov-based Economic Model Predictive Controller (LEMPC) using combined closed-loop and open-loop control action implementation schemes is proposed to optimize economic benefits in a time-varying manner while maintaining closed-loop process stability. Although sensor measurements may be vulnerable to cyber-attacks, the proposed controller design and operation strategy ensure that the process will maintain stability and stay resilient against particular types of destabilizing cyber-attacks. Data-based cyber-attack detectors are developed using sensor data via machine-learning methods, and these detectors are periodically activated and applied online in the context of process operation. Using a continuously stirred tank reactor example, simulation results demonstrate the effectiveness of the resilient control strategy in maintaining stable and economically optimal operation in the presence of cyber-attacks. The detection results produced by the detection algorithm demonstrate the capability of the proposed method in identifying the presence of a cyber-attack, as well as in differentiating between different types of cyber-attacks. Upon successful detection of the cyber-attacks, the impact of cyber-attacks can be mitigated by replacing the attacked sensors by secure back-up sensors, and secure operation will resume with the process operated under the proposed resilient LEMPC control strategy.

Machine-learning methodologies can be readily adopted in the context of control theory and cyber-physical security. In addition to having an adequate detection mechanism, control and operation strategies can be designed or adjusted accordingly if a process is vulnerable to cyber-attacks. Prior to developing control frameworks to address cyber-attacks in cyber-physical systems, there has been robust model-based control frameworks proposed to address uncertainties in the process. In [46], it was assumed that the uncertain process variables were bounded, and the robustness of the controller was established with respect to the worst-case values of the uncertain variables such that the state of the closed-loop system stays within a well-characterized region of the state-space given that the uncertain variables are within sufficiently small bounds. Moreover, other tube-based model predictive controller approaches have been developed to achieve robustness against unstructured uncertainties [38, 78]. In recent years, increasing research efforts have been dedicated to developing system and control designs to address cyber-attacks [35]. For instance, novel methods and tools to support effective preliminary design efforts for new cyber-physical systems were presented in [17], which addressed the integration of required defense and resilience solutions. A robust event-triggered model predictive control problem was investigated in [104] when the process is subject to bounded disturbances and denial-of-service cyber-attacks. Cumulative Sum (CUSUM) detection method was used in [18] in conjunction with model predictive control to operate a nonlinear system under false data injection attacks. Moreover, a robust two-tier control architecture was proposed in [22] that provided convenient system reconfiguration strategies to maintain cyber-security. In light of these considerations, the contributions of this work are as follows: 1) a cyber-secure operation mode of economic model predictive control, 2) the construction of a data-based machine-learning detection algorithm, and 3) the application of the proposed operation and detection schemes to a benchmark nonlinear chemical process example. The remainder of this chapter is organized as follows: the notation, the class of nonlinear process systems considered, as well as the formulation of Lyapunov-based economic model predictive control are shown in Section 3.1; the modified cyber-secure LEMPC formulations are presented in Section 3.2; the design of adapted intelligent cyber-attacks is shown in Section 4.3; the attack-resilient control strategies are developed in Section 3.4; the machine-learning-based detection algorithm is explained in Section 4.4; and the application of the proposed methodology to a nonlinear chemical process network is presented in Section 3.6.

3.1 Preliminaries

3.1.1 Nonlinear System Formulation

In this work, $|\cdot|$ is used to denote the Euclidean norm of a vector; x^T denotes the transpose of x; $\mathbf{R}^{\mathbf{n}}_{+}$ denotes the set of vector functions of dimension n whose domain is $[0,\infty)$. Set subtraction is denoted by "\", i.e., $A\backslash B := \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. Class $\mathscr{K}$ functions $\alpha(\cdot) : [0,a) \to [0,\infty]$ are defined as strictly increasing scalar functions with $\alpha(0) = 0$. The class of continuous-time nonlinear systems considered is described by the following state-space form:

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

$$\bar{x}(t) = h(x(t)) \tag{3.1b}$$

where $x(t) \in \mathbf{R}^n$ is the state vector, and $u(t) \in \mathbf{R}^m$ is the manipulated input vector, which is constrained by $u \in U := \{u_i^{min} \leq u_i \leq u_i^{max}, i = 1,...,m\} \subset \mathbf{R}^m$, where u_i^{min} and u_i^{max} are the lower and upper bounds for the input vector. We will denote the vector of state measurements from sensors, which may be compromised by sensor cyber-attacks, with $\bar{x}(t) \in \mathbf{R}^n$. When no cyber-attacks are present in the system, $\bar{x}(t) = x(t)$. Without loss of generality, the initial time t_0 is taken to be zero ($t_0 = 0$). It is assumed that $f(\cdot)$ is a sufficiently smooth vector function of its arguments, and $h(\cdot)$ is a sufficiently smooth vector function of x where $f(0,0) = 0$, $h(0) = 0$. Thus, the origin is an equilibrium point of the system of Eq. 8.1 under $u(t) = 0$.

We assume that there exists an explicit feedback controller of the form $u(t) = \phi(x(t)) \in U$ that can render the origin of the nonlinear closed-loop system of Eq. 8.1 asymptotically stable. The stabilizability assumption implies the existence of a positive definite control Lyapunov function $V(x)$ that satisfies the following conditions:

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|), \tag{3.2a}$$

$$\frac{\partial V(x)}{\partial x} f(x, \phi(x), 0) \leq -\alpha_3(|x|), \tag{3.2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq \alpha_4(|x|) \tag{3.2c}$$

for all $x \in D \subseteq \mathbf{R}^n$, where D is an open neighborhood around the origin, and $\alpha_i(\cdot)$, $i = 1,2,3,4$, are class $\mathscr{K}$ functions. Based on the universal Sontag control law [67], a candidate controller $\phi(x)$ is given by the saturated control law accounting for the input constraint $u \in U$, which is shown as

follows:

$$\varphi_i(x) = \begin{cases} -\dfrac{p+\sqrt{p^2+|q|^4}}{|q|^2}q, & \text{if} \quad q \neq 0 \\ 0, & \text{if} \quad q = 0 \end{cases} \tag{3.3a}$$

$$\phi_i(x) = \begin{cases} u_i^{min}, & \text{if} \quad \varphi_i(x) < u_i^{min} \\ \varphi_i(x), & \text{if} \quad u_i^{min} \leq \varphi_i(x) \leq u_i^{max} \\ u_i^{max}, & \text{if} \quad \varphi_i(x) > u_i^{max} \end{cases} \tag{3.3b}$$

where p denotes $L_f V(x)$ and q denotes $(L_g V(x))^T = [L_{g_1} V(x) \cdots L_{g_m} V(x)]^T$. $\varphi_i(x)$ of Eq. 7.7a represents the i^{th} component of the control law $\phi(x)$ without considering saturation of the control action at the input bounds. $\phi_i(x)$ of Eq. 7.7b represents the i^{th} component of the saturated control law $\phi(x)$ that accounts for the input constraints $u \in U$.

We first characterize a set of states D, in which the time-derivative of the Lyapunov function $V(x)$ under $u = \phi(x) \in U$ is negative for $x \neq 0$. Then we construct a level set of $V(x)$ inside D as $\Omega_\rho := \{x \in D \mid V(x) \leq \rho, \rho > 0\}$, which represents an estimate of the stability region of the closed-loop system of Eq. 8.1, and Ω_ρ is an invariant set for the closed-loop system. Therefore, starting from any initial state $x_0 := x(t_0)$ in Ω_ρ, $\phi(x(t))$ guarantees that the state trajectory of the closed-loop system of Eq. 8.1 remains within Ω_ρ and asymptotically converges to the origin. Thus, given that the sensor measurements received by the controller are secure and reliable (i.e., $\bar{x}(t) = x(t)$), the control law $\phi(x(t))$ is able to stabilize the process at the origin for any initial conditions $x_0 \in \Omega_\rho$.

3.1.2 Lyapunov-based Economic Model Predictive Control

Within the traditional paradigm of process control, a two-layer architecture is utilized to increase process economic profits where the tracking model predictive control (MPC) is coupled with an optimizer referred to as a real-time optimizer (RTO) that computes economically optimal steady-states for the MPC to track by solving a nonlinear optimization problem with a detailed steady-state plant model and a possibly nonlinear and nonquadratic objective function representing the process economics.

However, as operational efficiency and increasing energy consumption are becoming critical issues in the chemical and petrochemical industry, a model-based feedback control strategy, economic model predictive control (EMPC), was proposed to operate the system off steady-state by dynamically optimizing an economic cost function while accounting for stability constraints. It

has been repeatedly shown in the chemical process control literature that a number of industrially relevant processes can achieve higher profits when operated in a time-varying fashion than when operated at steady-state for all times; therefore, EMPC has been proposed as an efficient method to address process control problems integrated with dynamic economic optimization of the process (e.g., [7, 36, 46]).

Specifically, Lyapunov-based Economic Model Predictive Control (LEMPC) design is represented by the following optimization problem:

$$\mathscr{J} = \max_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} l_e(\tilde{x}(t), u(t))dt \tag{3.4a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = f(\tilde{x}(t), u(t)) \tag{3.4b}$$

$$u(t) \in U, \ \forall \, t \in [t_k, t_{k+N}) \tag{3.4c}$$

$$\tilde{x}(t_k) = \bar{x}(t_k) \tag{3.4d}$$

$$V(\tilde{x}(t)) \leq \rho_e, \ \forall \, t \in [t_k, t_{k+N}), \quad \text{if } \bar{x}(t_k) \in \Omega_{\rho_e} \tag{3.4e}$$

$$\dot{V}(\bar{x}(t_k), u) \leq \dot{V}(\bar{x}(t_k), \phi(\bar{x}(t_k))), \quad \text{if } \bar{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_e} \tag{3.4f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. $\dot{V}(x,u)$ is used to represent $\frac{\partial V(x)}{\partial x} f(x,u)$. The optimal input trajectory computed by the EMPC is denoted by $u^*(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*(t_k)$ is sent by the EMPC to be applied over the next sampling period in a sample-and-hold manner, and the EMPC is solved again in a rolling horizon fashion. The EMPC of Eq. 3.4 is solved by optimizing the time integral of the cost function $l_e(\tilde{x}(t), u(t))$ of Eq. 3.4a that accounts for process economic benefits over the prediction horizon subject to the constraints of Eqs. 3.4b-3.4f. Eq. 3.4c defines the input constraints applied over the entire prediction horizon. Eq. 3.4d defines the initial condition $\tilde{x}(t_k)$ of Eq. 3.4b, which is the state measurement $\bar{x}(t)$ at $t = t_k$. The constraint of Eq. 3.4e maintains the closed-loop state predicted by Eq. 3.4b in Ω_{ρ_e} over the prediction horizon if the state $\bar{x}(t_k)$ is inside Ω_{ρ_e}, where Ω_{ρ_e} is a conservative region within the closed-loop stability region Ω_ρ to make it an invariant set in the presence of sufficiently small bounded disturbances. However, if $\bar{x}(t_k)$ leaves Ω_{ρ_e} but still remains in Ω_ρ, the contractive constraint of Eq. 3.4f drives the state towards the origin for the next sampling

period such that the state will eventually enter Ω_{ρ_e} within finite sampling periods. Therefore, under the LEMPC of Eq. 3.4, the closed-loop state is maintained within the closed-loop stability region Ω_ρ for all times while optimal economic profits can be achieved via time-varying operation. The closed-loop stability proof can be found in [36].

3.2 Cyber-secure LEMPC Operation Strategies

Given that EMPC operates the system in an off steady-state manner, cyber-attacks that target EMPC systems can be designed to compromise both closed-loop stability and process economic benefits. Specifically, similar to the cyber-attacks that have been designed for tracking MPC [118], cyber-attacks for EMPC systems can be designed to drive states out of the stability region as fast as possible (e.g., min-max cyber-attack). The EMPC receiving falsified state measurements will compute incorrect control actions that will eventually cause the true process states to exit the stability region. The unstable evolution of state trajectory may occur even sooner in a system operated under EMPC than under tracking MPC since the system is operated off steady-state. Therefore, the selection of operating region, design of operating strategies, and integration of detection schemes need to be carefully considered.

3.2.1 Operation within Secure Operating Region

Considering that sensors are vulnerable to cyber-attacks, the process will be operated within a smaller region, $\Omega_{\rho_{secure}}$, where $0 < \rho_{secure} < \rho_e$, to avoid the system from immediately losing stability when under malicious cyber-attacks.

Although economic benefits will not be maximized when operated based on $\Omega_{\rho_{secure}}$ compared to operation around the original region Ω_{ρ_e}, it allows the system leeway to detect and mitigate the cyber-attack before closed-loop stability is lost (i.e., before true process states $x(t)$ exit Ω_ρ). The goal of detection is to identify the occurrence of a cyber-attack before the true process states exit the closed-loop stability region Ω_ρ, such that the process can be eventually driven back to and bounded within the secure region $\Omega_{\rho_{secure}}$ under LEMPC after eliminating the impact of cyber-attacks. The

modified LEMPC formulation is presented as follows:

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

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

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

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

$$V(\tilde{x}(t)) \leq \rho_{secure}, \ \forall \, t \in [t_k, t_{k+N}),$$
$$\text{if } \bar{x}(t_k) \in \Omega_{\rho_{secure}} \tag{3.5e}$$

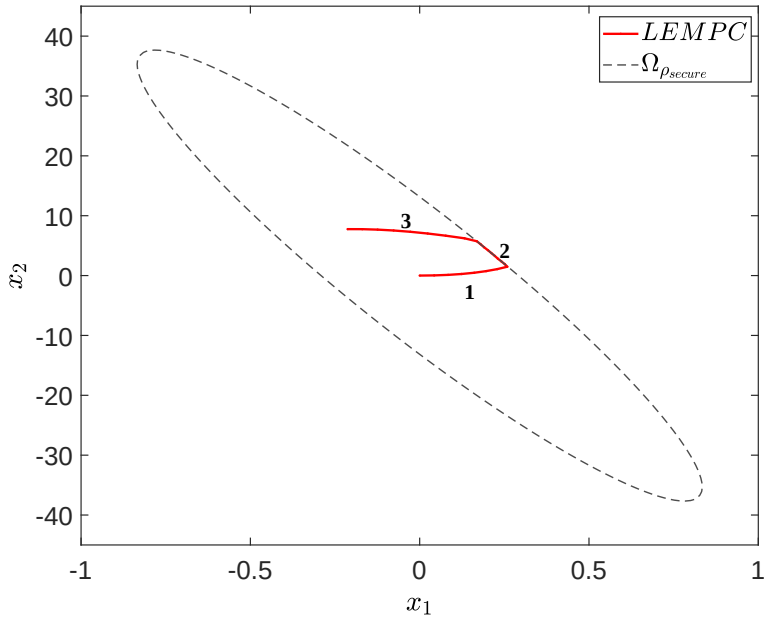
$$\dot{V}(\bar{x}(t_k), u) \leq \dot{V}(\bar{x}(t_k), \phi(\bar{x}(t_k))),$$
$$\text{if } \bar{x}(t_k) \in \Omega_{\rho} \backslash \Omega_{\rho_{secure}} \tag{3.5f}$$

where the notations follow those in Eq. 3.4. The constraint of Eq. 3.5e maintains the closed-loop states within $\Omega_{\rho_{secure}}$ over the prediction horizon if current state $\bar{x}(t_k)$ is inside $\Omega_{\rho_{secure}}$, and the contractive constraint of Eq. 3.5f will be activated when process states are outside of the neighborhood $\Omega_{\rho_{secure}}$. Since $\Omega_{\rho_{secure}}$ is characterized as a subset of Ω_{ρ_e} (i.e., $\rho_{secure} < \rho_e < \rho$), when the process state vector $\bar{x}(t_k)$ is inside $\Omega_{\rho_{secure}}$, it is guaranteed that under sufficiently small bounded disturbances $x(t_{k+1})$ will not exit Ω_{ρ}. Therefore, Ω_{ρ_e} is characterized accounting for bounded disturbances and sample-and-hold implementation of control actions to ensure the invariance of Ω_{ρ}. In the presence of bounded disturbances, Ω_{ρ_e} can be found computationally [46].

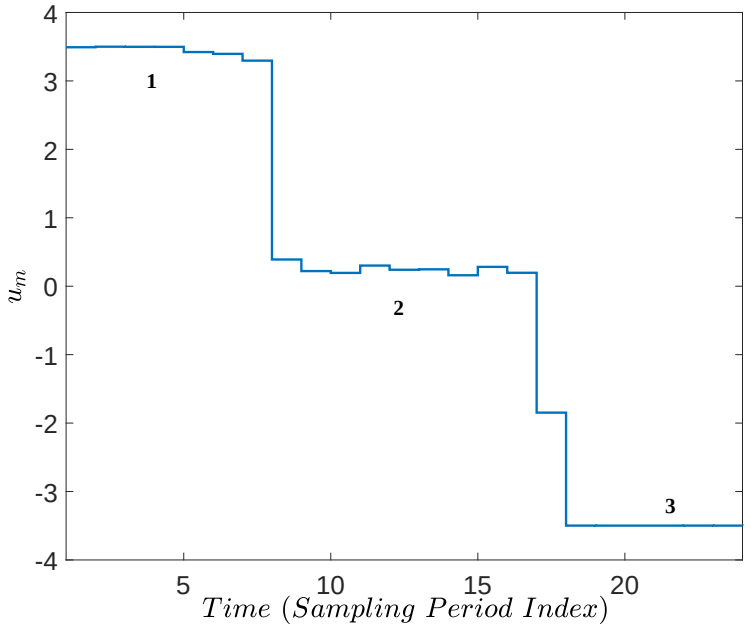
Furthermore, it is common that chemical processes are subject to periodic feed stock constraints, which are specified as part of the input constraint set U, where the quantity of feed materials is limited within a fixed period of time t_{N_p}. During this period of time, the total feed material is constrained to a constant value C, i.e., $\frac{1}{t_{Np}} \int_{t_0}^{t_{N_p}} u_m(\tau) d\tau = C$, where u_m represents feed material used at every sampling period. Therefore, the material consumption constraint renews every t_{N_p}. If the total operation time is longer than one material constraint period, this material consumption constraint results in cyclic operation of the plant, and consequently, cyclic behavior of the state-space trajectory. At the start of a new material constraint period, the total consumption limit is renewed, as new feed materials become available to be used again for the next constraint period.

Fig. 3.1 illustrates the trajectories of the states and the input constrained by feed materials under normal operation of LEMPC over one material constraint period. Assuming the process starts from the operating steady-state (e.g., the origin), since EMPC maximizes the economic benefits

while maintaining closed-loop stability during operation, it will drive the process states in the direction where economic benefit is optimized using large inputs until process states reach the boundary of the secure region $\Omega_{\rho_{secure}}$, as shown in Segment 1 in Fig. 3.1. Following this, the optimized state trajectory will progress along the boundary of $\Omega_{\rho_{secure}}$, as illustrated in Segment 2 in Fig. 3.1. Process states will remain on the boundary until the input materials start to exhaust and the input consumption constraints start restricting process states from further progressing along the boundary. With restricted inputs, the process states will be driven away from operating on the boundary – this is shown in Segment 3 in Fig. 3.1.



(a)



(b)

Figure 3.1: Trajectories of (a) process states x_1 and x_2, and (b) manipulated input u_m, under normal LEMPC operation over one material constraint period.

Remark 3.1. *Additionally, stealthy cyber-attacks can be designed with the aim of decreasing process economic benefits by driving the state to the region with low economic profits (but still in the closed-loop stability region). Since cyber-attacks targeting process economic profits will not*

cause physical damage or accidents, they are more difficult for process engineers to detect. Under the assumption that attackers know the process model and the stability region, such cyber-attacks will compromise sensor measurements such that the control actions calculated by the optimization problem of EMPC will not increase process economic profits for the next sampling period as much as it would do under nominal operations. In this study, we only consider attacks that intend to compromise process stability.

3.3 Intelligent Cyber-Attacks

Intelligent cyber-attacks are designed to intentionally destroy the control objectives of the system, disrupting system stability and degrading control performance. Cyber-attacks could compromise sensors, actuators, and/or the communication channels between them. In this work, we only consider attacks on sensor measurements. Sensor feedback measurements must accurately report the true state of the process to ensure closed-loop stability; falsified measurements may result in control actions that will no longer achieve maximum economic benefit and may ultimately drive the true process states outside of the stability region. There are some standard types of cyber-attacks considered in literature [97]. Min-max cyber-attacks aim to achieve maximum disruptive impact within shortest amount of time. Surge attacks cause maximum deviation for an initial "surge" period, and then the attacked value is set to a reduced value for the remainder of the attack duration such that the cumulative deviation will not exceed a certain threshold that will trigger alarms in conventional detection methods such as Cumulative Sum [16,80]. Geometric attacks geometrically increase the deviation of the attacked value from its true value until it reaches the alarming threshold. Details on the formulations of the four attack types can be found in Section 4.3.1. Being process and controller behavior aware, the cyber-attacks will have access to information on the operating region of the process under LEMPC $\Omega_{\rho_{secure}}$, and existing alarms configured on the input and output ranges. Specifically, when attacks intend to induce maximum disruption (i.e., in min-max or surge attacks), the attacked value will be set to the maximum or minimum value beyond which an alarm monitoring the current state measurement will be immediately triggered. These intelligent cyber-attacks are designed such that no alarms will be sounded (i.e., the falsified state measurement is not outside the operating stability region or the alarm window) and the controller is still able to compute feasible control actions, but have large enough variations such that economic optimality and closed-loop stability will be lost.

To train a machine-learning-based detector, closed-loop data will be collected where attacks

with varying durations L_a are introduced at random times i_0 during the simulation period. If no attacks occur within the simulation period (or the detection window), then the measurement signals are classified as "no attack". Furthermore, we consider a system where some sensors are attacked and some remain intact. For clarity, only one type of cyber-attack will occur at a time during the simulation period.

3.3.1 Design of Cyber-attacks Adapted to Secure LEMPC Operation

The system is now operated under the modified LEMPC of Eq. 3.5, where the operating region is set to be a smaller level set of $V(x)$, $\Omega_{\rho_{secure}}$, within the stability region, where $0 < \rho_{secure} < \rho_e < \rho$. Thus, the cyber-attacks imposed on the sensors also need to be adapted to prevent having a falsified measurement beyond the operating region $\Omega_{\rho_{secure}}$ and to avoid triggering any immediate alarms based on the values of the state measurements. The adapted mathematical formulations of min-max, surge, geometric, and replay attacks are presented in the following sections.

3.3.1.1 Min-Max Cyber-attack

While avoiding triggering any alarms, min-max attacks result in maximum destabilizing impact within a short time period. Therefore, the falsified state measurements take values that are furthest from the equilibrium point (minimum or maximum) but not outside of the secure operating region $\Omega_{\rho_{secure}}$. The min-max attack can be formulated as follows:

$$\bar{x}(t_i) = \arg\min / \max_{x \in \mathbf{R^n}} \{V(x(t_i)) \leq \rho_{secure}\}, \quad \forall\, i \in [i_0, i_0 + L_a] \tag{3.6}$$

where ρ_{secure} defines the level set of the Lyapunov function $V(x)$ that characterizes the secure operating region of the closed-loop system of Eq. 8.1 under LEMPC, $\bar{x}$ is the compromised sensor measurement, i_0 is the time instant that the attack is introduced, and L_a is the total duration of the attack in terms of sampling periods.

3.3.1.2 Surge Cyber-attack

Surge attacks maximize the disruptive impact for an initial short period of time, then they remain at a lower value for the rest of the attack duration. The maximum or minimum attack value is also defined based on the secure operating region, $\Omega_{\rho_{secure}}$. The length of the initial surge period and the reduced value after the surge can be designed in many ways as long as the cumulative error from t_{i_0} to $t_{i_0+L_a}$ between state measurements and their predicted true values does not exceed the threshold

defined in some statistic-based detection methods (e.g., CUSUM). In our study, the reduced value after the surge is set to act as a sufficiently small bounded noise imposed on the attacked sensor. The formulation of the surge attack is presented below:

$$\begin{aligned} \bar{x}(t_i) &= \arg\min / \max_{x \in \mathbf{R^n}} \{V(x(t_i)) \leq \rho_{secure}\}, \text{ if } i_0 \leq i \leq i_0 + L_s \\ \bar{x}(t_i) &= x(t_i) + \eta(t_i), \text{ if } i_0 + L_s < i \leq i_0 + L_a \end{aligned} \tag{3.7}$$

where i_0 is the start time of the attack, L_s is the duration of the initial surge, and $\eta_l \leq \eta(t_u) \leq \eta_u$ is the bounded noise added on the sensor measurement after the initial surge period, where η_l and η_u are the lower and upper bounds of the noise, respectively.

3.3.1.3 Geometric Cyber-attack

Under geometric cyber-attacks, closed-loop system stability deteriorates at a geometric speed until the cyber-attack reaches the maximum or minimum allowable value as characterized by the secure operating region. At the start of the attack t_{i_0}, a small constant $\beta \in \mathbf{R}$ is added to the true measured output $x(t_{i_0})$, where $x(t_{i_0}) + \beta$ is well below the alarm threshold. Following that, at each subsequent time step, β is multiplied by a factor $(1+\alpha)$, where $\alpha \in (0,1)$, until $\bar{x}$ reaches the maximum allowable attack value bounded by $\Omega_{\rho_{secure}}$. Thus, attackers will choose the two parameters α and β based on $\Omega_{\rho_{secure}}$ and the attack duration. Geometric attacks can be written in the form as follows:

$$\bar{x}(t_i) = x(t_i) + \beta \times (1+\alpha)^{i-i_0}, \ \ \forall\, i \in [i_0, i_0 + L_a] \tag{3.8}$$

where β and α are parameters that define the magnitude and speed of the geometric attack.

3.3.1.4 Replay Cyber-attack

Replay cyber-attacks have access to all previous system outputs corresponding to secure nominal operating conditions where no cyber-attacks are present. The attacker extracts segments of these previous state measurements and injects them into the current measurement readings. As the replayed values are given by secure sensors and supposedly inside the secure operating bounds, classical detectors will not be able to recognize any abnormalities. Replay attacks can be represented by the following equations:

$$\bar{x}(t_i) = x(t_k), \ \forall\, k \in [k_0, k_0 + L_a], \ \ \forall\, i \in [i_0, i_0 + L_a] \tag{3.9}$$

where $x(t_k)$ is the true plant measurement, L_a represents the length of the attack (which is also the length of the replay segment) in terms of sampling periods, and $\bar{x}$ is the series of replay attacks

added at time t_{i_0} duplicating previous state measurements that are recorded starting from time t_{k_0}. The duration of the attack could be exactly the length of one or more material constraint periods. Therefore, the tampered state trajectory would look identical to the nominal state trajectory of one (or more) complete cycle(s) of operation starting from a different set of initial conditions.

3.4 Attack-Resilient Combined Open-loop and Closed-loop Control

Due to the LEMPC constraints of Eq. 3.5e and Eq. 3.5f, for any initial condition $x_0 \in \Omega_\rho$, the evolution of state trajectory $x(t)$ will be driven towards but ultimately bounded inside the secure region $\Omega_{\rho_{secure}}$. As the economic benefit of the process is maximized with respect to the state vector, it is likely that during one material constraint period, the optimized states will reach, and evolve along the boundary of the secure region $\Omega_{\rho_{secure}}$, which is a level set of the control Lyapunov function $V(x)$. Assuming that the attacker has knowledge on the stability region as well as the secure region that the LEMPC operates based on, in order to induce maximum destructive impact on the system (e.g., in a min-max or surge cyber-attack) without triggering any alarms, the tampered state measurements will be near or on the boundary of the secure region $\Omega_{\rho_{secure}}$. Therefore, regardless of the presence of a cyber-attack, the measured process states will likely reach the boundary of $\Omega_{\rho_{secure}}$ where $V(\bar{x}) = \rho_{secure}$ during the operation of one material constraint period. In other words, when measured process states yield $V(\bar{x}) = \rho_{secure}$, there could be two reasons: 1) following optimized control actions $u^*(t_k)$, the measured process states reach the boundary of the bounded secure region $\Omega_{\rho_{secure}}$ at time t_k under the normal operation with no cyber-attacks, or 2) the measured states are compromised by a cyber-attack (e.g., min-max, or surge) at time t_k. Therefore, when measured states $\bar{x}(t_k)$ provide $V(\bar{x}(t_k)) = \rho_{secure}$, this measurement can no longer be trusted due to the ambiguous cause of this observation, and closed-loop control can no longer be carried out.

To combat the ambiguity of state measurements when they are on the boundary of $\Omega_{\rho_{secure}}$, open-loop control actions will be used in conjunction with closed-loop control. Assuming that the states measured at the beginning of each material constraint period, $t = t_{N_0}$, (or initial conditions at $t = t_0$) are secure and correct, the open-loop control actions are computed at the beginning of the

material constraint period by solving the following nonlinear optimization problem:

$$\mathscr{J} = \max_{u' \in S(\Delta)} \int_{t_{N_0}}^{t_{N_0+N_p}} l_e(\tilde{x}(t), u'(t))dt \tag{3.10a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = f(\tilde{x}(t), u(t)) \tag{3.10b}$$

$$u'(t) \in U, \ \forall \ t \in [t_{N_0}, t_{N_0+N_p}) \tag{3.10c}$$

$$\tilde{x}(t_{N_0}) = \bar{x}(t_{N_0}) \tag{3.10d}$$

$$V(\tilde{x}(t)) \leq \rho_{secure}, \ \forall \ t \in [t_{N_0}, t_{N_0+N_p}),$$
$$\text{if } \bar{x}(t_{N_0}) \in \Omega_{\rho_{secure}} \tag{3.10e}$$

$$\dot{V}(\bar{x}(t_{N_0}), u) \leq \dot{V}(\bar{x}(t_{N_0}), \phi(\bar{x}(t_{N_0}))),$$
$$\text{if } \bar{x}(t_{N_0}) \in \Omega_{\rho} \backslash \Omega_{\rho_{secure}} \tag{3.10f}$$

where N_p is the number of sampling periods in one material constraint period, which is the prediction horizon for open-loop control. At time t_k, a new material constraint period begins, the EMPC in open-loop control mode receives state measurement $x(t_k)$ and computes the optimal trajectory of N_p control actions that will be applied in a sample-and-hold manner until the end of this material constraint period. In the case that there are no cyber-attacks or process disturbances, this optimal trajectory of control actions would yield maximum economic benefits while meeting all input and state constraints.

While at closed-loop operation, if feedback measurement is no longer reliable and cannot be used for closed-loop control, the open-loop control actions that were calculated at the beginning of the material constraint period will be used as a substitute until the end of the material constraint period. At the end of the material constraint period, a cyber-attack detector is activated to determine any occurrence of an attack, and the reliability of the control system is re-assessed. The detector will provide information on the security status of the feedback measurements over the latest material constraint period. Upon mitigating the impact of a confirmed attack and/or confirming the security of the control system, closed-loop control with secure feedback measurement can be reactivated as a new material constraint period starts.

Although the absence of feedback may result in minor performance degradation in the case that process disturbances and modeling error exists and no cyber-attack is present, this strategy also completely eliminates the impact of a min-max or surge attack on the sensor measurements. The implementation strategy is illustrated in a logic flow diagram in Fig. 3.2, and the specific steps are outlined as follows:

1. At the start of a material constraint period ($t = t_{N_0}$), open-loop control actions over the course of the material constraint period are computed following Eq. 3.10. Closed-loop control is active, calculating the optimal control action over the next sampling period following Eq. 3.5.

2. If $\rho_{secure} - V(\bar{x}(t_k)) \leq c$, (where $c > 0$ quantifies the distance from the boundary of secure region to categorize a state measurement as being untrustworthy), then closed-loop control (i.e., the modified LEMPC of Eq. 3.5) will be deactivated and open-loop control action $u'(t_k)$ calculated by the LEMPC of Eq. 3.10 will be used as an substitute.

3. Open-loop control actions $u'(t_k)$ will be used until $t_{N_0+N_p}$.

4. At $t_{N_0+N_p}$, the cyber-attack detector is activated to examine past full-state measurements $\bar{x}(t_k)$ for $k \in [N_0, N_0+N_p]$. If an attack is detected, then disconnect the tampered sensors, reroute these measurement signals to a set of secure back-up sensors, and go to Step 5. If detection indicates no attack, go to Step 5.

5. At $t_{N_0+N_p}$, a new material constraint period starts, and closed-loop control is reactivated. Repeat Steps 1 – 4.

Remark 3.2. *In some cases, the system may never reach the boundary of $\Omega_{\rho_{secure}}$ depending on the initial condition, the size of $\Omega_{\rho_{secure}}$, and the length of the material constraint period. If this is the case, and cyber-attacks wrongfully set the measured states to be on the boundary of $\Omega_{\rho_{secure}}$, then closed-loop control will still be deactivated following the implementation of Step 2, and open-loop control actions will be used.*

Remark 3.3. *In our study, we do not consider systems under large disturbances. Since open-loop control actions over the entire operating period are calculated by MPC at the beginning of the simulation period based on the nominal system, closed-loop stability is guaranteed using open-loop control if there are no disturbances or model uncertainties in the real process given that the process is open-loop stable. In the presence of process disturbances other than cyber-attacks, estimations of true process states are needed and closed-loop control can be applied based on these estimated states to stabilize the system within a bounded region. However, in our manuscript, we assume that the actual nonlinear process is disturbance-free; therefore, open-loop control is able to ensure closed-loop stability until the end of the material constraint period (i.e., the time instant at which NN detection will be activated and closed-loop control will resume).*

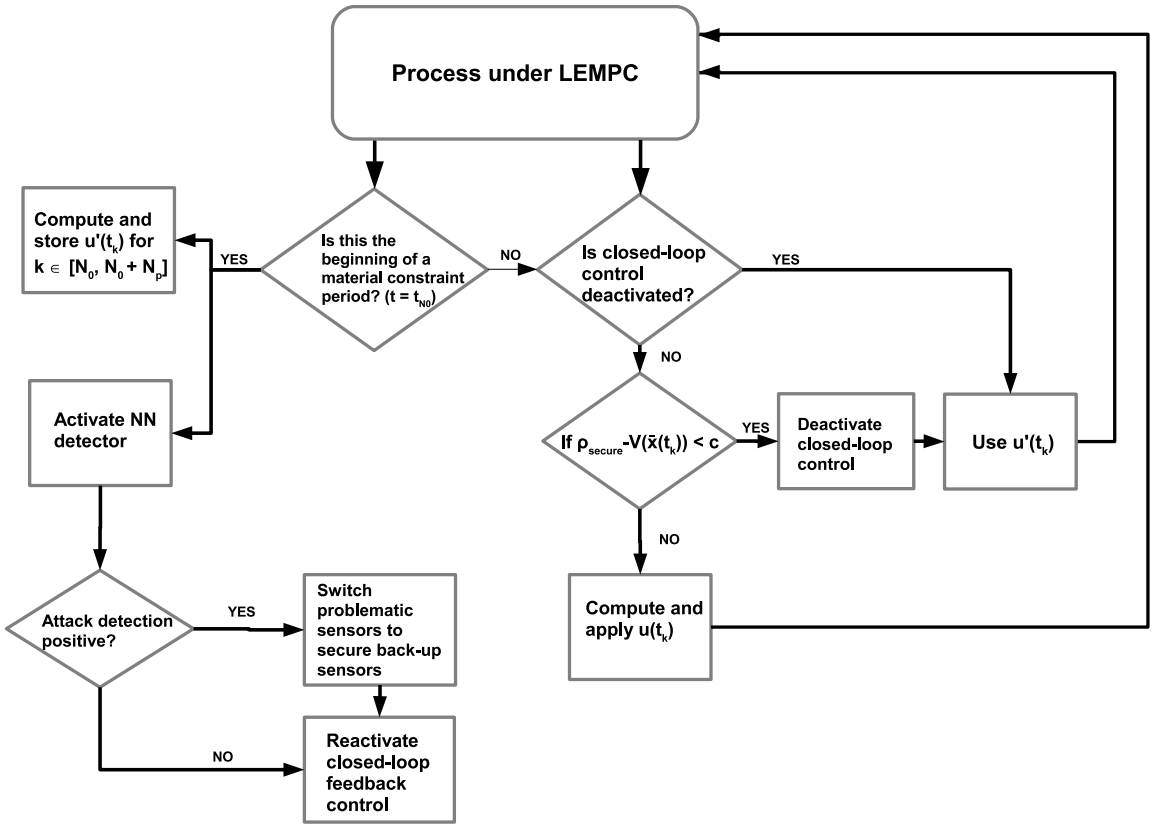


Figure 3.2: Logic flowchart outlining the implementation steps of the attack-resilient operation of LEMPC using combined closed-loop and open-loop control actions when operating within a secure region $\Omega_{\rho_{secure}}$.

3.5 Detection of Cyber-Attacks Targeting EMPC

Cyber-attack detection carried out using data-based approaches, and more specifically, machine-learning methods, have been studied in many literature (e.g., [1,52,85]). Using data-based methods to train a detection algorithm for cyber-attacks separates the detector from the physical process model, and therefore makes the detector resilient to both process changes and intelligent stealthy attacks designed based on process behavior.

Amongst advanced machine-learning methods, neural networks (NN) have been successful in a wide range of applications for both supervised and unsupervised classifications [44]. There are also other types of state-of-the-art machine-learning classification methods that have been used in a variety of applications in recent literature, such as k-nearest-neighbors, random forest, and

support vector machine [45]. Amongst these machine-learning algorithms, the advantage of neural network is that it provides a broad class of tuning parameters and a variety of nonlinear activation functions to optimize the overall model. Furthermore, neural networks can be developed using multiple different training algorithms, providing more alternatives during training to obtain better performance results [109]. In a supervised classification problem, by training the neural network with labeled data corresponding to each target class, the neural network can be used to classify new data into classes that share similar characteristics. Depending on the training data, the neural network can distinguish between two (i.e., "attack" or "no attack") or multiple classes (each class representing a known type of attack).

We use a feed-forward artificial neural network for supervised classification in our study. Each layer in the neural network consists of a series of nonlinear functions, yielding values for the neurons in the subsequent layer from the previous layer. Specifically, the neurons in the first hidden layer are derived from the inputs, and the neurons in the output layer are calculated from those in the last hidden layer. These nonlinear functions are activation functions of the weighted sum of inputs (or neurons in the previous layer) with an added bias term.

The structure of a basic neural network model employed here is shown in Fig. 3.3, with each input representing a nonlinear function $p(\cdot)$ of the full state measurements at each sampling time, and an output vector for predicted class label. The mathematical formulation of a two-hidden-layer feed-forward neural network is as follows:

$$\theta_j^{(1)} = g_1\left(\sum_{i=1}^{N_T} w_{ij}^{(1)} p(\bar{x}(t_i)) + b_j^{(1)}\right) \tag{3.11a}$$

$$\theta_j^{(2)} = g_2\left(\sum_{i=1}^{h_1} w_{ij}^{(2)} \theta_i^{(1)} + b_j^{(2)}\right) \tag{3.11b}$$

$$\theta_j^{(3)} = g_3\left(\sum_{i=1}^{h_2} w_{ij}^{(3)} \theta_i^{(2)} + b_j^{(3)}\right), \;\; y_{pred} = [\theta_1^{(3)}, \theta_2^{(3)}, ..., \theta_H^{(3)}]^T \tag{3.11c}$$

with $\theta_j^{(1)}$ and $\theta_j^{(2)}$ representing neurons in the first and second hidden layer, respectively, where $j = 1, ..., h_l$ is the number of neurons in layer $l = 1$ and $l = 2$. $\theta_j^{(3)}$ represents neurons in the output layer ($l = 3$), where $j = 1, ..., H$, and H is the number of class labels. In this study, we use two hidden layers for the cyber-attack detector design; however, multiple hidden layers can also be developed using similar formulations. For each sample, the input layer consists of variables $p(\bar{x}(t_i))$, which is a nonlinear function of the full-state measurements at time t_i, where $i = 1, ..., N_T$ is the length of the time-varying trajectory. The weights connecting neurons i and j in

consecutive layers (from $l-1$ to l) are $w_{ij}^{(l)}$, and the bias term on the j^{th} neuron in the l^{th} layer is $b_j^{(l)}$. Each layer calculates an output based on the information received from the previous layer, as well as the optimized weights, biases, and the nonlinear activation function g_l (some examples include hyperbolic tangent sigmoid transfer function $g(z) = \frac{2}{1+e^{-2z}} - 1$, and softmax function $g(z_j) = \frac{e^{z}j}{\sum_{i=1}^{H} e^{z_i}}$ where H is the number of class labels). Various common activation functions including ReLu, sigmoid, radial basis functions were presented and their performances were analyzed in [96]. Furthermore, while Bayesian regularization is a powerful regularization method to avoid over-fitting and over-training, there are also other regularization algorithms such as L2 and L1 regularization, both of which add a parameter penalty in the objective function in an effort to reduce the generalization error (thus, the testing error) of the trained model. The advantage of Bayesian regularization is that it provides a probability distribution of optimal parameters instead of a single optimal value, thereby effectively dropping out trivial nodes to speed up the training process. In the output layer, y_{pred} is a vector giving the predicted probabilities of each class label. The predicted class label for the examined sample is indicated by the neuron with the highest probability, which in turn provides information on either the presence of a cyber-attack, or the type of the cyber-attack, depending on the classification problem the neural network is trained to solve.

To obtain an optimal set of weights and biases in Eq. 6.6, the Levenberg-Marquardt algorithm [41] is used to minimize a Bayesian regularized mean squared error cost function, which has the following form:

$$S(w) = \gamma \sum_{k=1}^{N_s} (y_{pred,k} - y_{true,k})^2 + \zeta \sum_{p=1}^{N_w} w_p^2 \tag{3.12}$$

where $k = 1, ..., N_s$ represents the number of samples in the training dataset, $p = 1, ..., N_w$ represents the number of weights and biases in the neural network, y_{true} is the vector of target class labels of each sample, y_{pred} is the vector of the predicted probabilities associated with each class label, and γ and ζ are the regularization hyper-parameters. Within the Levenberg-Marquardt algorithm, the gradient and the Hessian matrix of $S(w)$ are calculated using the backpropagation method. The weights and the data are assumed to have Gaussian prior probability distributions. Then, the regularization hyper-parameters, γ and ζ, are updated by maximizing their posterior probability distribution provided the data, which is equivalent to maximizing the likelihood of evidence by Bayes' Theorem. Within each epoch, two sequential procedures are carried out: the cost function $S(w)$ is minimized with respect to w, and the likelihood of evidence is maximized with respect to γ and ζ. Detailed formulation of this procedure can be found in [15]. Training and testing accuracies are calculated, which are the ratios between the number of correctly classified samples and total number of samples in the training and testing sets, respectively.

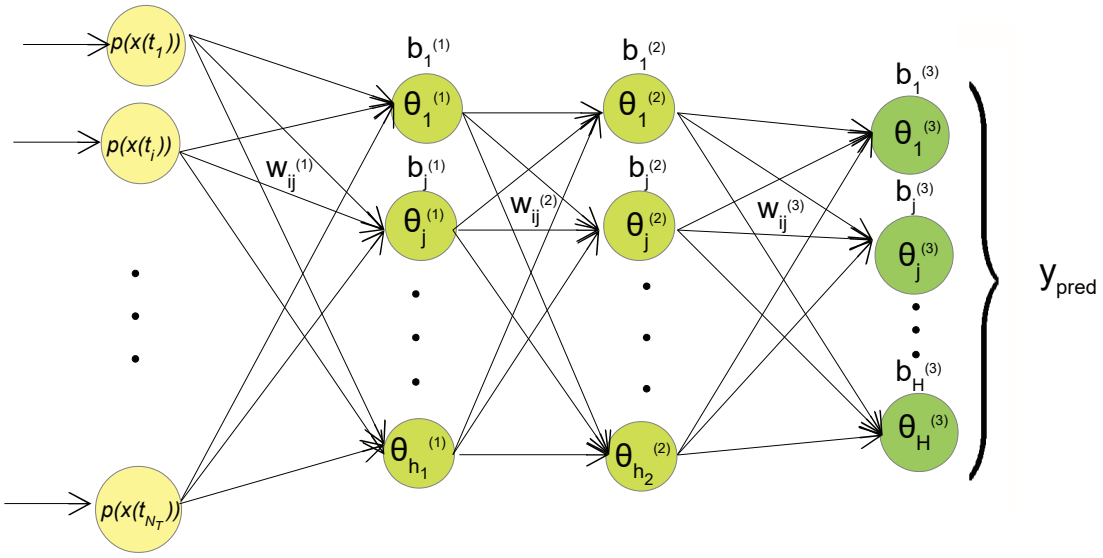


Figure 3.3: Feed-forward neural network structure with 2 hidden layers with inputs being a nonlinear function $p(\bar{x})$ at each sampling time of the model predictive controller within the detection window N_T, and output being the probability of each class label for the examined trajectory indicating the status and/or type of cyber-attack.

To develop an NN detector, state measurement data are collected while the system is operated under the modified LEMPC of Eq. 3.5. For better detection accuracy, various state evolutions within the stability region under different operating conditions need to be accounted for; therefore, training data is collected for a broad range of initial conditions within the stability region Ω_ρ. Full state measurements $\bar{x}(t)$ are recorded along the time-varying trajectory for $t \in [t_0, t_{N_T}]$, and a nonlinear function denoted by $p(\bar{x})$ is computed. In order to provide an effective one-dimensional input feature for the detection problem, the function $p(\bar{x})$ needs to capture the dynamic behavior of all states. The selection of this input variable, $p(\bar{x})$, is discussed in Section 4.4.1.

After data collection and adequate training, the NN detector is implemented online and activated at the end of each material constraint period, with the process controlled by the modified LEMPC in Eq. 3.5 with combined open-loop and closed-loop control described in Section 3.4. Since the feed-forward NN model is a static model that receives inputs of fixed dimension, N_T (which is the length of the time-varying trajectory over one material constraint period N_p), the detection window of the NN detector when activated online is $N_T = N_p$. The detector will receive the entire sequence of full state measurements $\bar{x}(t_k)$ over the latest material constraint period with a fixed length N_T. Each sample consists of a two-dimensional matrix $n \times N_T$, where n is the full state dimension, and N_T is the length of each state trajectory within the detection window. Each training sample corresponds to a different set of initial conditions for the closed-loop system simulation,

and equal number of samples within each class labels are collected to ensure training accuracy.

3.5.1 Choice of Detection Input Variable

The nonlinear system of Eq. 8.1 is operated in an off steady-state manner under LEMPC by maximizing a nonlinear function of process state vector with respect to the control actions, which are subject to their respective lower and upper bounds, and material consumption constraints. Considering this, the exact trajectory of each individual state variable is not predictable and does not follow a general expected trend even under nominal operation. Therefore, assessing the trajectory of the measured state vector is not an effective method of detecting the occurrence of a cyber-attack.

Moreover, if the goal of a cyber-attack is to destabilize the closed-loop system within the shortest amount of time, the attacker will choose to set the current state measurement to the maximum/minimum allowable attack value characterized by the boundary of the secure operating region $\Omega_{\rho_{secure}}$ such that no alarms will be triggered. Therefore, the falsified sensor measurements will also yield a Lyapunov function that is equal to ρ_{secure}. Unlike the case of operation under tracking MPC where the Lyapunov function decreases as the process states are driven towards the origin, off steady-state operation of LEMPC results in a state trajectory that remains on the boundary of the secure operating region $\Omega_{\rho_{secure}}$ where $V(\bar{x}) = \rho_{secure}$ for the majority of each material constraint period as discussed in Section 3.2.1. Therefore, the trajectory of the Lyapunov function $V(\bar{x})$ under nominal operation and under cyber-attacks can be too similar to differentiate. For these reasons, the control Lyapunov function of the full-state measurements $V(\bar{x})$, which is used as an input variable for the detection algorithm used together with a tracking MPC [22], is no longer a good measure of input for the detection algorithm when the system is operated under LEMPC.

Given that EMPC optimizes the economic benefit in its cost function, the progression of economic benefit is a measure that effectively reflects the time-varying operation under LEMPC; hence, information derived from the economic benefit provides a good comparison for attacked and not-attacked scenarios. Therefore, we will be monitoring the evolution of economic benefits during closed-loop operation. The cumulative economic benefit increases monotonically as operation time progresses. The first derivative of cumulative economic benefit (i.e., incremental economic benefit, which can be analogous to the reaction rate of desired product, at each sampling period) displays varying patterns depending on the initial conditions and on the material consumption constraint. The rate of change in the incremental economic benefit, or the change in the production

reaction rate between sampling periods, provides information on the rate of change in the optimized cost function l_e inside the integral in Eq. 3.5a. This rate of change, which is also the second derivative of the cumulative economic benefit, will be used as the input parameters $p(\bar{x})$ for the neural-network-based detection algorithm.

Remark 3.4. *Material constraints on the feed stock are commonly seen in the industry. Moreover, operating the process in an off-steady-state manner with material constraint periods imposed is a common practice for economic model predictive control because we would like to compare its control performance to that of the process operated at steady-state for all times. While EMPC aims to maximize economic benefits by computing optimal sets of control actions, we impose this constraint on EMPC such that the sum of the calculated control actions will be the same as the sum used in steady-state operation. Depending on the initial conditions, the state trajectory may exhibit different patterns when the material constraint is removed. Despite this, the proposed neural-network detection approach does generalize to systems without material constraint periods. This is because the neural network detector, with adequate training, is still able to distinguish the attacked trajectory from the nominal trajectory of the examined detection variable (in our case, the second time-derivative of the economic objective function).*

3.6 Application to a Nonlinear Chemical Process

3.6.1 Process Description and Control System Design

The application of the modified LEMPC of Eq. 3.5, the resilient control strategy presented in Section 3.4, as well as the training and online detection of NN cyber-attack detectors are demonstrated on a chemical process example. Specifically, the process considered is a well-mixed, non-isothermal continuous stirred tank reactor (CSTR), within which an irreversible second-order exothermic reaction takes place. The second-order reaction, $A \rightarrow B$, transforms reactant A to product B at a reaction rate $r_B = k_0 e^{-E/RT} C_A^2$. The CSTR is equipped with a heating jacket that supplies or removes heat at a rate Q. The dynamic model of this CSTR process is described by the following material and energy balance equations:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{3.13a}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{3.13b}$$

where C_A is the concentration of reactant A in the reactor, V is the volume of the reacting liquid in the reactor (assuming the vessel has constant holdup), T is the temperature of the reactor and Q denotes the heat input rate. The concentration of reactant A in the feed is C_{A0}. The feed temperature and volumetric flow rate are T_0 and F, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p. ΔH, k_0, R, and E represent the enthalpy of reaction, pre-exponential constant, ideal gas constant, and activation energy, respectively. A complete list of the process parameter values are shown in Table 5.2.1.

Table 3.1: Parameter values of the CSTR.

$T_0 = 300\ K$	$F = 5\ m^3/hr$
$V = 1\ m^3$	$E = 5 \times 10^4\ kJ/kmol$
$k_0 = 8.46 \times 10^6\ m^3/kmol\ hr$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$C_p = 0.231\ kJ/kg\ K$	$R = 8.314\ kJ/kmol\ K$
$\rho_L = 1000\ kg/m^3$	$C_{A0_s} = 4\ kmol/m^3$
$Q_s = 0.0\ kJ/hr$	$C_{A_s} = 1.95\ kmol/m^3$
$T_s = 401.87\ K$	

The CSTR is initially operated at the unstable steady-state $[C_{As},\ T_s] = [1.9537\ kmol/m^3,\ 401.87\ K]$, and $[C_{A0_s}\ Q_s] = [4\ kmol/m^3,\ 0\ kJ/hr]$. The manipulated inputs are the inlet concentration of reactant A and the heat input rate, which are represented by the deviation variables $\Delta C_{A0} = C_{A0} - C_{A0_s}$, $\Delta Q = Q - Q_s$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{A0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q| \leq 5 \times 10^5\ kJ/hr$. Therefore, the states and the inputs of the closed-loop system are $x^T = [C_A - C_{As}\ T - T_s]$ and $u^T = [\Delta C_{A0}\ \Delta Q]$, respectively, such that the equilibrium point of the system is at the origin of the state-space, (i.e., $x_s^T = [0,0], u_s^T = [0,0]$). We assume that at time $t = t_0$, the system is at the equilibrium point (i.e., the initial conditions of the system are $x_0 = [0,0]^T$).

The control objective is to maximize the economic profit of the CSTR process of Eq. 8.45 by manipulating the inlet concentration ΔC_{A0} and the heat input rate ΔQ, while maintaining the closed-loop state trajectories in the stability region Ω_ρ for all times under LEMPC. The objective function of the LEMPC optimizes the production rate of B as follows:

$$l_e(\tilde{x}, u) = r_B(C_A, T) = k_0 e^{-E/RT} C_A^2 \tag{3.14}$$

The dynamic model of Eq. 8.45 is numerically simulated using the explicit Euler method with an

integration time step of $h_c = 2.5 \times 10^{-5}\ hr$. The nonlinear optimization problem of the LEMPC of Eq. 3.5 is solved using the MATLAB OPTI Toolbox with the sampling period $\Delta = 2.5 \times 10^{-3}\ hr$.

The modified LEMPC of Eq. 3.5 uses the following material constraint to make the averaged reactant material available within one operating period $t_{N_p} = 0.06\ hr$ to be its steady-state value, C_{A0s} (i.e., the averaged reactant material in deviation form, u_1, is equal to 0).

$$\frac{1}{t_{N_p}} \int_0^{t_{N_p}} u_1(\tau) d\tau = 0\ kmol/m^3 \tag{3.15}$$

The control Lyapunov function $V(x) = x^T P x$ is designed with the following positive definite P matrix:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{3.16}$$

The closed-loop stability region Ω_ρ for the CSTR with $\rho = 320$ is characterized as a level set of Lyapunov function inside the region D, from which the origin can be rendered asymptotically stable under the controller $u = \phi(x) \in U$ of Eq. 7.7. The secure operating region $\Omega_{\rho_{secure}}$ for the LEMPC in Eq. 3.5 is selected to have $\rho_{secure} = 90$. The matrix P in $V = x^T P x$ and the stability region Ω_ρ are determined through simulations when determining the largest invariant set Ω_ρ in state-space (i.e., the level set of V) in which $\dot{V}$ is rendered negative ($\dot{V} \leq -\alpha_3(|x|)$, where α_3 is a class $\mathscr{K}$ function) for all states within Ω_ρ under the stabilizing controller $u = \phi(x) \in U$. Different values of P will generate different set of states where $\dot{V} \leq -\alpha_3(|x|)$, resulting in a different size and shape of the invariant set Ω_ρ.

Remark 3.5. *The closed-loop system exhibits periodic patterns due to the periodic reactant material constraint imposed on the control actions. The process itself is not periodic; however, the material constraint imposed on the control actions renews periodically.*

Remark 3.6. *The design of the secure operating region $\Omega_{\rho_{secure}}$ can be adjusted depending on the system dynamics, the desired threshold for economic benefits, the magnitude and type of cyber-attacks, as well as whether the detector experiences time delay in correctly identifying the attacks. If the process dynamics are very fast, then more room needs to be vacated between Ω_ρ and $\Omega_{\rho_{secure}}$ to accommodate for the fast changes in process states when under cyber-attacks. However, designing a conservative secure operating region $\Omega_{\rho_{secure}}$ is at the expense of compromising economic benefits, since the maximum economic gain under normal operation is bounded by $\Omega_{\rho_{secure}}$. Therefore, the determination of the size of $\Omega_{\rho_{secure}}$ comes from a balance between operational stability and economic performance. These were all factors taken into consideration*

when running extensive closed-loop simulations to determine the value of ρ_{secure}.

3.6.2 Resilient Operation of LEMPC

With initial conditions $x_0 = [0,0]^T$, the closed-loop operation of the CSTR process in Eq. 8.45 over one material constraint period t_{N_p} under the modified LEMPC in Eq. 3.5, and under the resilient control of LEMPC with combined open-loop and closed-loop control actions as described in Section 3.4 around the secure operating region $\Omega_{\rho_{secure}}$ are both carried out. Fig. 3.4 presents the state-space plot showing the trajectory of the measured process states using the modified LEMPC of Eq. 3.5 and using the resilient LEMPC control strategy when the process is under no attack. The switching from using closed-loop to open-loop control actions happens at $t_s = 0.0175\ hr$. For $t_0 \leq t_k < t_s$, measured process states are well within the secure operating region $\Omega_{\rho_{secure}}$, and closed-loop control using the modified LEMPC of Eq. 3.5 is used with state feedback updates. The LEMPC of Eq. 3.5 is deactivated at $t_s = 0.0175\ hr$ when the measured process states first reach the boundary of the secure operating region, and can no longer be trustworthy as this may be a result of a cyber-attack, i.e., when $\rho_{secure} - V(\bar{x}(t_k)) \leq c$, where $c = 0.5$ for this case study. The distance from the secure region boundary, c, is determined to account for computational error in designing and inserting the attacked sensor measurements. It provides a buffer zone for which resilient LEMPC can be activated accurately and preemptively. Therefore, for $t_s \leq t_k \leq t_{N_p}$, control actions $u'(t_k)$ from the open-loop optimization of Eq. 3.10 that are solved based on the initial condition x_0 will be applied.

Even in the case that no process disturbance, no model mismatch, and no cyber-attack is present, the resulting state trajectories under LEMPC (closed-loop only), and the resilient LEMPC (closed-loop followed by open-loop control actions after the switching time t_s) are slightly different. This is because the prediction horizon used in the ordinary LEMPC with periodic closed-loop feedback has a length of $N = 8$ and rolls forward in time as feedback signal updates are received, whereas the open-loop optimization problem computed at the beginning of the material constraint period accounts for $N_p = 24$. Therefore, the control actions computed from the open-loop optimization, $u'(t_k)$, will be slightly different from $u(t_k)$ calculated from online optimization, resulting in slightly different state trajectories.

Despite the subtle differences in the state trajectory, using open-loop control actions following closed-loop control still maintains the process states within the secure operating region (hence the stability region) for all times. It is important to note that, if the process is operated at steady-state, the total economic benefit in the form of $\int_{t_0}^{t_{N_p}} l_e(\bar{x}(t))dt$ is 0.6397 $kmol/m^3$, which is much less

than that achieved under time-varying EMPC operation. The total economic benefit from t_0 to t_{N_p} using closed-loop-only control actions from the LEMPC of Eq. 3.5 is 0.8192 $kmol/m^3$, and using the resilient control strategy outlined in Section 3.4 is similarly 0.8203 $kmol/m^3$. Under no disturbances or model mismatch, the total economic benefit achieved by the resilient LEMPC using open-loop control actions is marginally higher. In closed-loop operation, we used a shorter prediction horizon to speed up the computation to ensure the real-time implementation of EMPC. Since the optimization problem of EMPC is essentially non-convex, the solutions we obtained from closed-loop operation may not be as good as the solutions calculated at the beginning, which uses a sufficiently long prediction horizon that covers the entire operating period as per material constraints. This shows the effectiveness of the resilient control strategy when the system is under no attack as it does not compromise system stability and economic performance. Furthermore, the similarity in the two trajectories also suggests that, if a cyber-attack is present and the resilient control strategy is utilized, the evolution of true process states will highly resemble that under closed-loop control in the absence of cyber-attacks. Under min-max attacks with LEMPC operation, the total economic benefit that the true process states provide is 1.4939 $kmol/m^3$; the higher economic benefit is a result of the min-max attacks driving the true states outside of the stability region. With resilient LEMPC operation and under min-max attacks, the true process states also yield a total economic benefit of 0.8203 $kmol/m^3$ over one operating period, which is the same as the case under no attacks. Since NN detection is activated at the end of the first operating period, the total economic benefit with integrated NN detection is also 0.8203 $kmol/m^3$. This shows that when the process operates under resilience LEMPC, the addition of cyber-attacks does not alter the economic performance over one material constraint period.

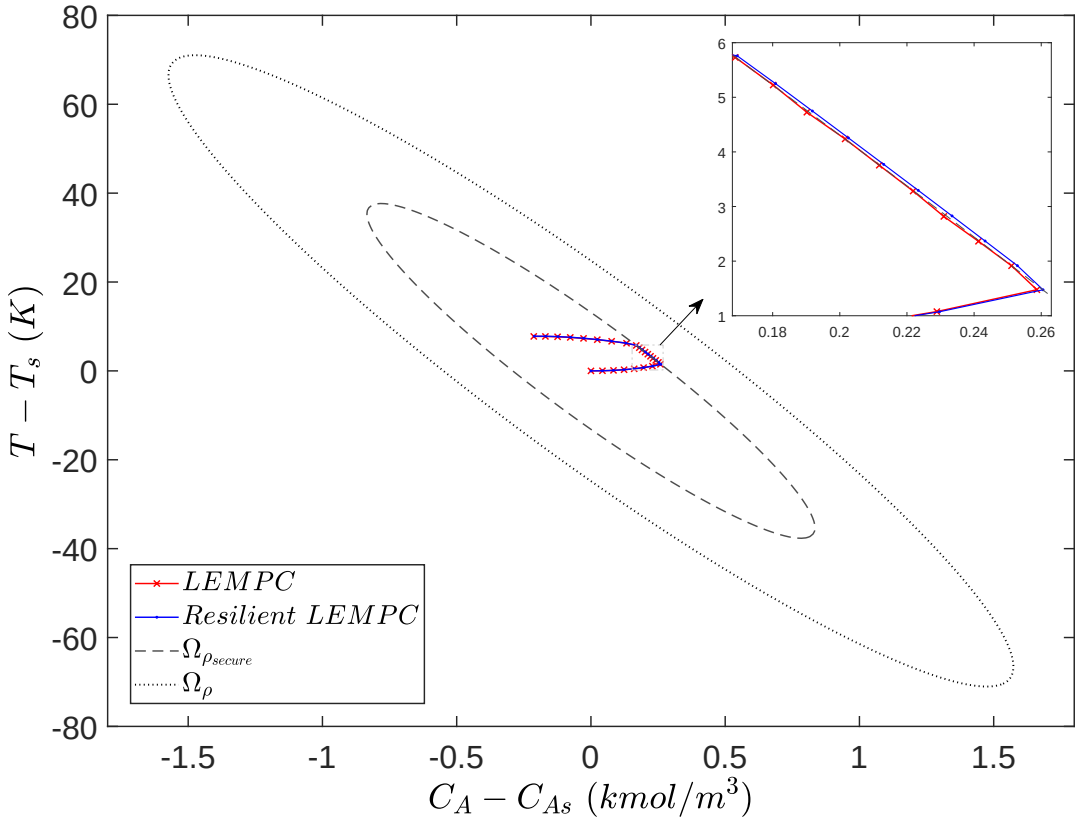


Figure 3.4: State-space plot showing the evolution of measured process states over one material constraint period under LEMPC (red trajectory) and under resilient LEMPC (blue trajectory).

3.6.3 Cyber-attack Resiliency Assessment

The purpose of using the resilient control strategy outlined in Section 3.4 is to prevent true process states from exiting the stability region Ω_ρ when under sensor cyber-attacks. Fig. 3.5 shows the state-space plot of the evolution of true process states and attacked state measurements from initial conditions $x_0 = [0, 0]^T$ over one material constraint period under LEMPC and under resilient LEMPC when the temperature sensor is attacked by min-max, geometric, replay and surge attacks, respectively. In all cases, once the specified cyber-attack starts, it will continue until it has been successfully detected; the detection results and process simulation after the detection are shown in Section 3.6.5. Here, the simulation results over only one material constraint period are shown. After a cyber-attack has tampered the sensor, the resulting falsified state measurements will not exit the secure operating region $\Omega_{\rho_{secure}}$ so as to stay inconspicuous to the control engineer.

Min-max and surge cyber-attacks are added at $t = t_s = 0.0175\ hr$ such that there will be no suspicious deviation in the Lyapunov function of the system. At $t = 0.0175\ hr$, both the true process state and the attacked state measurement will reach the boundary of the secure operating region, $V(x(t_s)) = V(\bar{x}(t_s)) = \rho_{secure}$. As shown in Fig. 3.5(a) and Fig. 3.5(d), when the temperature sensor is under min-max and surge attacks respectively, true process states will exit $\Omega_{\rho_{secure}}$ and eventually Ω_{ρ} if only closed-loop control actions from the online LEMPC optimization in Eq. 3.5 are used. However, when the resilient LEMPC control strategy is implemented, closed-loop control is deactivated at $t = 0.0175\ hr$, and the falsified feedback measurements can no longer impact the control system. Open-loop control actions, which are calculated based on a correctly measured set of initial conditions, are used starting at $t = 0.0175\ hr$ until the end of the material constraint period when $t = t_{N_p} = 0.06\ hr$. As a result, the true process states will not exit $\Omega_{\rho_{secure}}$, and the evolution of the true process states is almost identical to that under secure closed-loop control (as demonstrated in Section 3.6.2). The system stays resilient to min-max and surge attacks, with protected stability and comparable control performance.

However, the resilient control strategy may not be effective when the system is under other types of attacks, particularly in situations where the falsified state measurement does not approach the boundary of $\Omega_{\rho_{secure}}$. To illustrate this, geometric attacks on the temperature measurements as shown in Fig. 3.5(b) start at $t = 0.01\ hr$ following Eq. 4.12, where $\beta = x(t) * (1.001)$ and $\alpha = 0.1$. As cyber-attacks could happen at any time instant during operation, geometric attacks are designed and inserted as such to demonstrate the incapability of the resilient control strategy in handling geometric attacks or attacks alike. At $t = 0.01\ hr$, the states have not reached the boundary of $\Omega_{\rho_{secure}}$, therefore not satisfying the condition for deactivating closed-loop control. Geometric attacks starting at $t = 0.01\ hr$ resulted in state measurements that did not reach the boundary of $\Omega_{\rho_{secure}}$ for the entire duration of cyber-attack. Hence, closed-loop control continued with these false measurements, and the true process states exited $\Omega_{\rho_{secure}}$ during operation. Despite having a correct array of open-loop control actions computed at $t = 0\ hr$ using the correctly measured initial conditions, these control actions were not used. As a result, the resilient control strategy fails to ensure that the true process states are maintained within the secure operating region $\Omega_{\rho_{secure}}$.

Moreover, there may be situations where, even when closed-loop control is deactivated and feedback measurements are no longer used, the true process states still exit $\Omega_{\rho_{secure}}$ because the open-loop control actions are calculated based on false sensor measurements. To illustrate this scenario, replay attacks as shown in Fig. 3.5(c) start at $t_0 = 0\ hr$, and the replayed signals span the duration of one material constraint period. In other words, the replayed signals are real

closed-loop state measurements when the system started from a different set of initial conditions, $\bar{x}_0 = [-0.2107\ kmol/m^3; 7.8047\ K]$. Since the initial conditions $\bar{x}_0$ are incorrect, open-loop control actions optimized over the prediction horizon of N_p based on $\bar{x}_0$ are also not correct. As a result, despite the falsified state measurements also reaching the boundary of $\Omega_{\rho_{secure}}$ at $t = 0.0175\ hr$ and deactivating closed-loop control, these incorrect open-loop control actions applied on the process still resulted in true process states exiting the secure operating region.

In this example, when under geometric and replay attacks, the true process states did not exit the stability region Ω_ρ; however, this may not be the case for a different geometric attack with larger α (geometric factor), a different replay attack that yielded more aggressive open-loop control actions, or for a faster process. In other words, system stability cannot be guaranteed by using the resilient control strategy, and an effective cyber-attack detection mechanism needs to be included.

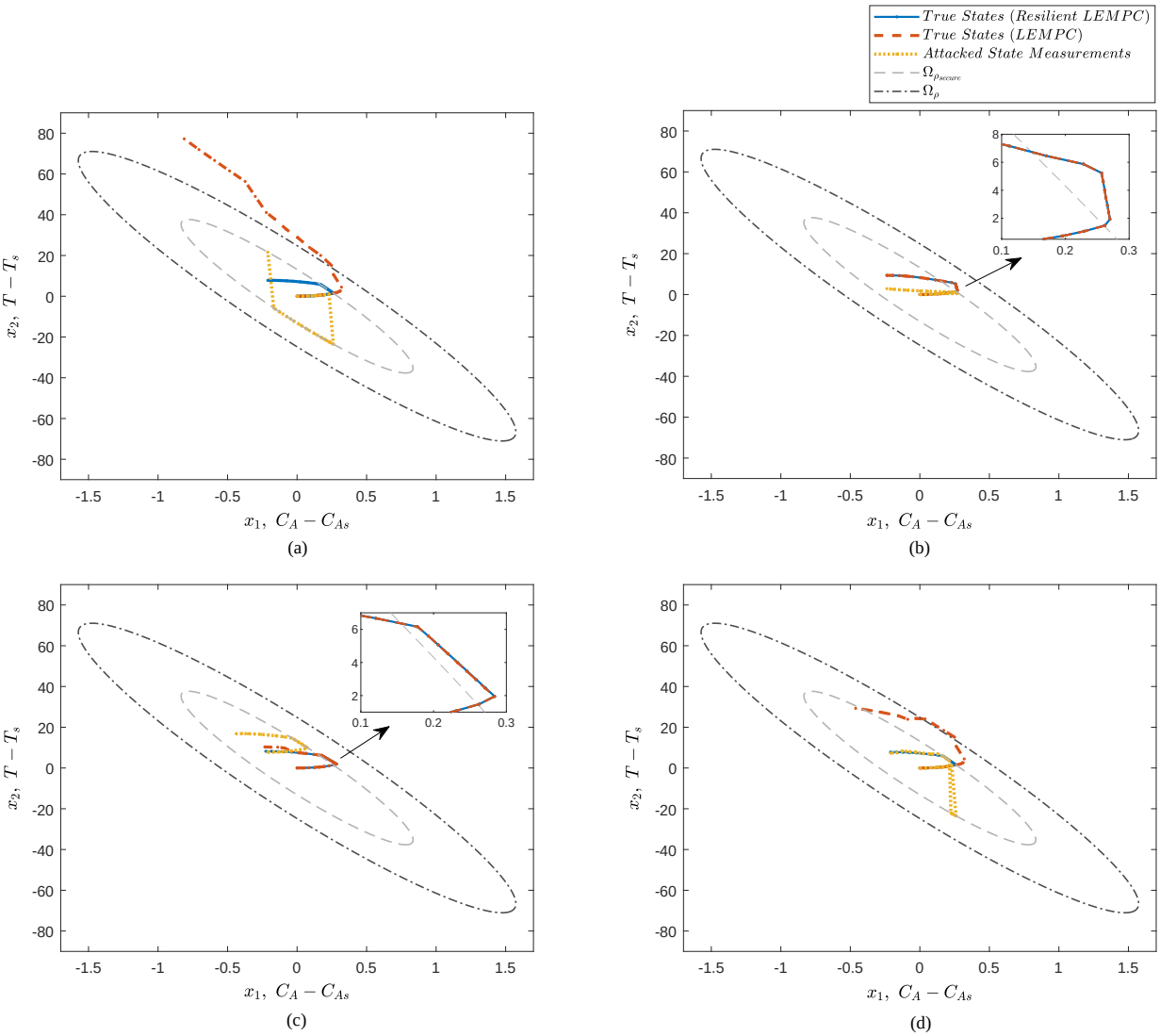


Figure 3.5: State-space plot showing the evolution of true process states and attacked state measurements (yellow trajectories) over one material constraint period under LEMPC (red trajectories) and under resilient LEMPC (blue trajectories) when (a) min-max, (b) geometric, (c) replay, and (d) surge attacks, are targeting the temperature sensor, where the dash-dotted ellipse is the stability region Ω_ρ and the dashed ellipse is $\Omega_{\rho_{secure}}$.

3.6.4 Detectors Training and Testing

To train neural-network detectors, training data will be collected under closed-loop operation with the secure LEMPC outlined in Eq. 3.5. Simulation period is one material constraint period $t_{N_p} = 0.06\ hr$ with $N_p = 24$. Cyber-attacks are added at random times and last until the end of the simulation period. Neural network models are constructed and trained using the MATLAB

Machine Learning and Deep Learning Toolboxes.

The reaction rate to yield product B, $r_B(\bar{x})$ can be calculated from full-state measurements $\bar{x}(t)$ at each time instant t_k from $k = 0$ to $k = N_p$ following Eq. 3.14, where $C_A = \bar{x}_1 + C_{A_s}$ and $T = \bar{x}_2 + T_s$. The input parameters used for neural network training are the time-varying trajectory of the rate of change in $r_B(\bar{x})$ over the simulation period of one material constraint period$N_p = 24$, which is denoted as $p(\bar{x})$, shown as follows:

$$p(\bar{x}(t)) = \frac{dr_B(\bar{x})}{dt} \tag{3.17}$$

The evolution of $p(\bar{x})$ when the temperature sensor is under no attack, and under min-max, geometric, replay, and surge attacks, are shown in Fig. 3.6. Each sample consists of a 1×24 array of $p(\bar{x})$, started from a different initial condition within Ω_ρ. With extensive closed-loop simulations, equal number of samples are collected for each output label, from which 70% are used for training, and 30% are used for testing.

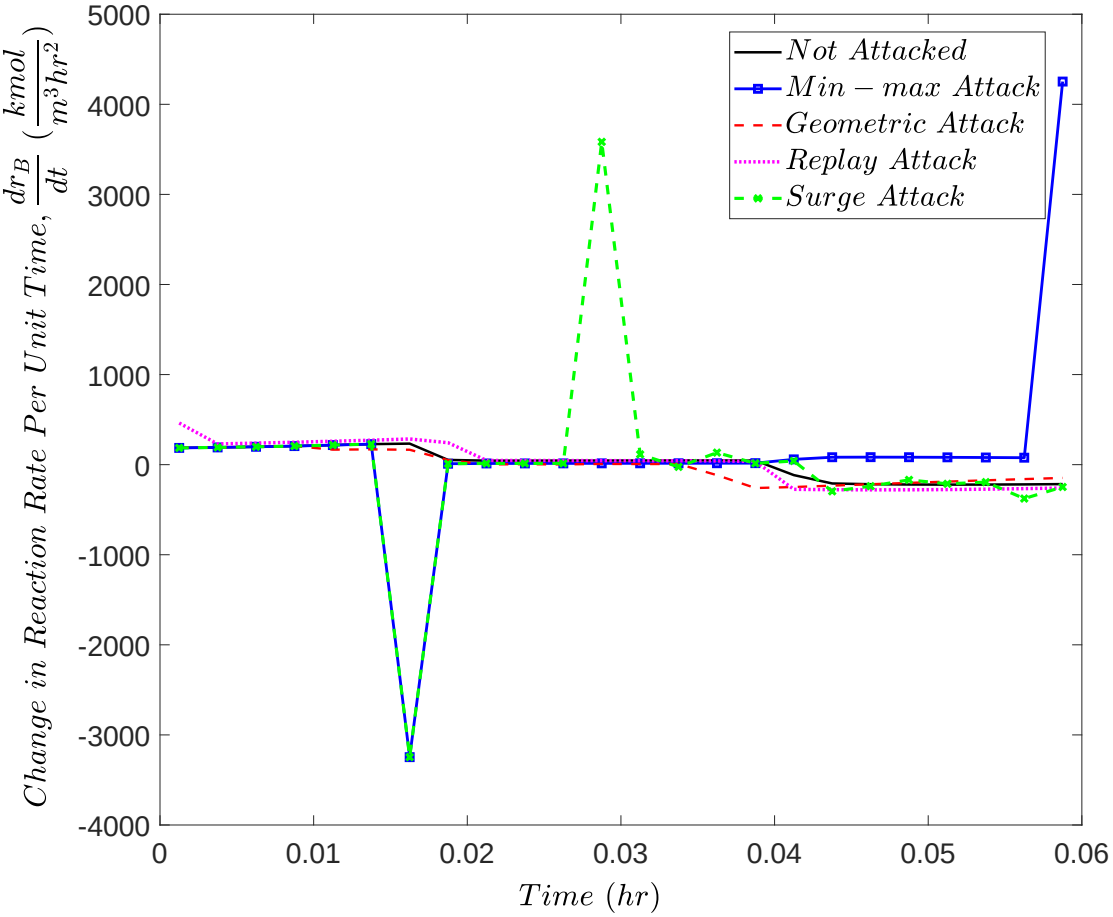


Figure 3.6: Time-derivative of the reaction rate r_B of Eq. 3.14 based on measured process states over one material constraint period, when the temperature sensor is under no attack, and under min-max, geometric, replay, and surge attacks, respectively.

First, min-max attacks are used to train a neural-network-based detector. This feed-forward neural network model has two hidden layers with 12 and 10 neurons in each layer respectively. Both hidden layers use a *tansig* activation function, which is in the form $g_{1,2}(z) = \frac{2}{1+e^{-2z}} - 1$. The output layer uses a $softmax$ function to provide a predicted probability of the class labels, which is in the form of $g_3(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{H} e^{z_i}}$ where H denotes the number of class labels. Bayesian regularized mean squared error cost function $S(w)$ are minimized with respect to the weights and biases using the Levenberg-Marquardt algorithm, in which the gradient and the Hessian matrix of $S(w)$ are calculated using the back-propagation method. A total of 750 samples are collected for each class label. The training time for this 2-class detector is 2.05 seconds, undergoing 70 epochs, and the detector achieves a training accuracy of 98.9%. The testing accuracy of this detector against the

different attack types is shown in Table 4.3. Note that geometric attacks are not identified as being attacked due to the vast difference in the trends of $p(\bar{x})$ when under geometric attack compared to min-max attacks as shown in Fig. 3.6.

A second detector is trained with min-max and geometric attacks. The detector is able to classify between 3 classes: not attacked, attacked by min-max cyber-attacks, and attacked by geometric cyber-attacks. Thus, the detector is capable of differentiating the types of cyber-attacks in addition to indicating the presence of one. This detector is trained because geometric attacks exhibit very different behavior than min-max attacks, and therefore the testing accuracy by the 2-class detector is very low. This 3-class feed-forward neural network detector has two hidden layers with 15 and 12 neurons each, using the same activation functions and cost function in Eq. 6.6, which is minimized using the Levenberg-Marquardt algorithm. The training time for this 3-class detector is 39.48 seconds with 300 epochs. This 3-class detector achieves an overall training accuracy of 91.8%, and its testing accuracies in response to min-max, geometric, and surge attacks are shown in Table 4.3. The detector accurately identifies min-max and geometric attacks as their respective labels, and it classifies 71.0% of surge attacks as min-max, 10.0% as geometric, and the remaining 19% are wrongly classified as "not attacked".

Table 3.2: Detection accuracies of NN detectors in response to min-max, geometric, and surge attacks.

	Detector 1 (Attacked vs. Not Attacked)	Detector 2 (Min-max vs. Geometric vs. Not Attacked)
Min-max	98.3%	89.7%
Geometric	2.4% (Attacked)	71.1%
Surge	87.0% (Attacked)	71.0% (Min-max); 10.0% (Geometric)
Not Attacked	98.4%	95.6%

Remark 3.7. *Since replay signals could mimic the secure operation of one entire material constraint period starting at a different initial condition, they are essentially a different sample that belongs to the class of "not attacked", and will be rightfully classified as being "not attacked". At the end of the material constraint period, the falsified signals follow exactly the trajectory of previous secure measurements of one period, thus they will remain undetectable by the NN detectors. Since the NN approach used in this study is based on supervised classification, it is heavily dependent on labeled data from distinct classes. Being duplicates of nominal signals,*

replay signals are not in a distinct class of signals that is different from the nominal data, thus the proposed NN approach is an unsuitable detection method for replay attacks. Purely data-based approaches examining sensor measurements in the case of replay attacks will not be sufficient, and model-based prediction approaches may be a possible future research direction. The reader may refer to other works on the detection of replay attacks in [51, 107].

3.6.5 Online Detection

Detector 1 is used to detect min-max and surge attacks, whereas detector 2 is used to detect geometric attacks. The corresponding detector is activated at the end of the material constraint period, and examines state measurements received over the last material constraint period. Since replay attacks cannot be detected, the online detection results are also not shown.

Fig. 3.7 shows the evolution of true process states and measured process states attacked by min-max, geometric, and surge cyber-attacks when the process is controlled by the resilient LEMPC with combined open-loop and closed-loop control. The figures show the trajectories over two material constraint periods, where NN-based detection occurs twice – once at the end of the first period, and once at the end of the second period.

Min-max and surge attacks are correctly detected by detector 1 at the end of the first constraint period $t = 0.06\ hr$ by examining the trajectory of $p(\bar{x}(t))$ from $t = 0\ hr$ to $t = 0.06\ hr$, after which the sensor devices are switched to a secure set of redundant sensors and operation continues with these secure sensor measurements. During the second period, the attacked old set of sensors are no longer connected to the control system, and the newly switched set of sensors are not tampered by cyber-attacks. At the end of the second material constraint period $t = 0.12\ hr$, detector 1 is activated again, and it correctly classifies the secure measurements as "not attacked".

Furthermore, if a particular attack type is trained as a separate class (i.e., "geometric") from other attack types (i.e., "min-max"), then the detector is also capable of identifying the type of cyber-attack. As shown in Fig. 3.7(b), although the true process states exited $\Omega_{\rho_{secure}}$ during the first material constraint period (closed-loop control based on false feedback signals was not deactivated), the state measurements attacked by geometric attacks were still correctly identified as geometric by detector 2 at the end of the first material constraint period. After switching the sensor devices to the respective secure back-up sensors, detector 2 correctly identifies the trajectory of $p(\bar{x}(t))$ over the second material constraint period from $t = 0.06\ hr$ to $t = 0.12\ hr$ as "not attacked". This means that, although the resilient control strategy cannot ensure stability over one material constraint period if the attacked measurement deliberately avoids approaching the

boundary of $\Omega_{\rho_{secure}}$, the attack can still be detected at the end of the material constraint period, and mitigation measures can be taken following the successful detection to terminate the impact of the cyber-attacks. One method to avoid the true states from exiting the stability region when under geometric attacks is to adjust the size of $\Omega_{\rho_{secure}}$ such that the resilient control strategy could come into effect earlier. Moreover, setting a shorter material constraint period in addition to operating within a conservative secure region could be another preventative method to consider, so that the cyber-attack detection can happen more frequently.

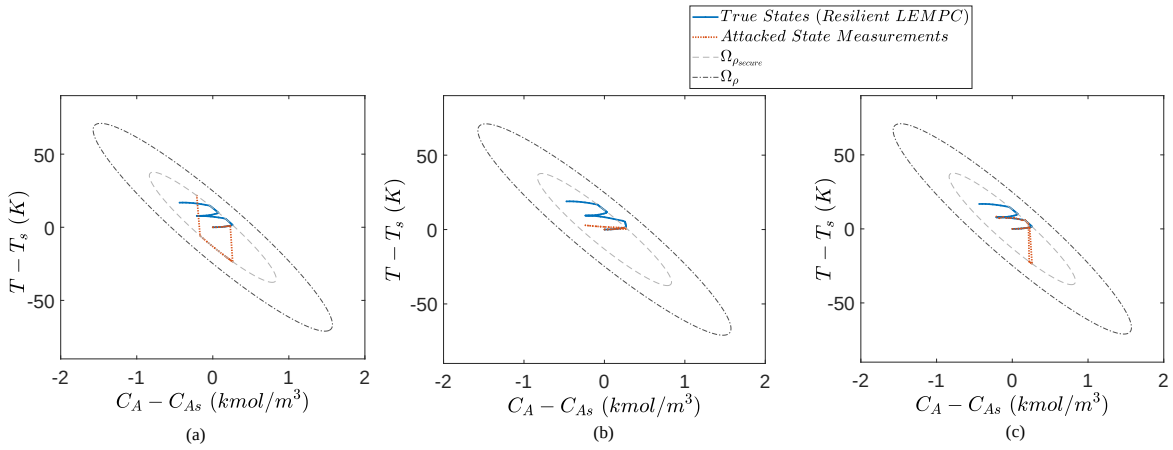


Figure 3.7: State-space plot showing the evolution of true process states (blue trajectories) and attacked state measurements (red trajectories) over two material constraint periods under the resilient LEMPC when (a) min-max, (b) geometric, and (c) surge attacks, targeting the temperature sensor are successfully detected by a NN detector at the end of the first material constraint period, $t = 0.06\,hr$, where the dash-dotted ellipse is the stability region Ω_ρ and the dashed ellipse is $\Omega_{\rho_{secure}}$.

Chapter 4

Cyber-Security of Centralized, Decentralized, and Distributed Control-Detector Architectures for Nonlinear Processes

Decentralized and distributed control systems provide an efficient solution to many challenges of controlling large-scale industrial processes. With the expansion in communication networks, vulnerability to cyber intrusions also increases. This work investigates the effect of different types of standard cyber-attacks on the operation of nonlinear processes under centralized, decentralized, and distributed model predictive control (MPC) systems. The robustness of the decentralized control architecture over distributed and centralized control architectures is analyzed. Moreover, a machine-learning-based detector is trained using sensor data to monitor the cyber security of the overall system. Specifically, detectors built using feed-forward neural networks are used to detect the presence of an attack or identify the subsystem being attacked. A nonlinear chemical process example is simulated to demonstrate the robustness of decentralized control architectures and the effectiveness of the neural-network detection scheme in maintaining the closed-loop stability of the system.

This work explores the impact of standard types of sensor cyber-attacks on centralized, decentralized, and distributed Lyapunov-based model predictive control (LMPC) systems, and shows the robustness of decentralized control system against certain cyber-attacks when compared to centralized and distributed control systems. We examine Lyapunov-based model predictive

control systems in this study, however, the proposed control-detector architecture and detection methodology can be extended to other applications of model predictive control, or other methods of advanced control systems in general. Then, a neural-network-based detector is trained and implemented online to monitor sensor behaviors when the process is operated under the decentralized control system. Section 2 presents the preliminaries on notation, the general class of nonlinear systems considered, and the stabilizability assumptions. Section 3 includes the formulations of the centralized, decentralized, and distributed LMPCs. The design of intelligent cyber-attacks is shown in Section 4, and the development of the cyber-attack detector using neural networks is explained in Section 5. In Section 6, closed-loop simulations and analyses of a two-CSTR-in-series chemical process are presented.

4.1 Preliminaries

4.1.1 Notation

Throughout the manuscript, $|\cdot|$ is used to denote the Euclidean norm of a vector. The notation x^T is used to denote the transpose of x. Set subtraction is denoted by "\", i.e., $A\backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. A continuous function $\alpha : [0,a) \to [0,\infty)$ is said to belong to class $\mathscr{K}$ if it is strictly increasing and is zero only when evaluated at zero. The function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable in its domain. Lastly, $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$.

4.1.2 Class of Systems

Consider a general class of continuous-time nonlinear systems in which multiple distinct sets of manipulated inputs are used, with each set of manipulated inputs regulating a specific subsystem of the process. We consider $j = 1,...,N_{sys}$ subsystems, with each subsystem j consisting of states x_j which are regulated by only u_j and potentially impacted by states in other subsystems due to coupling between subsystems. The continuous-time nonlinear dynamics of the subsystem j is described as follows:

$$\dot{x}_j = F_j(x,u_j) := f_j(x) + g_j(x)u_j,\ x_j(t_0) = x_{j_0},\ \forall j = 1,...,N_{sys} \tag{4.1}$$

where N_{sys} represents the number of subsystems, $x_j \in \mathbf{R}^{n_j}$ represents the state vector for subsystem j, and x represents the vector of all states $x = [{x_1}^T \cdots {x_{N_{sys}}}^T]^T \in \mathbf{R}^n$, where $n = \sum_{j=1}^{N_{sys}} n_j$. $u_j \in \mathbf{R}^{m_j}$ is the set of manipulated input vectors for each subsystem j, which together constitute the vector of all manipulated inputs $u \in \mathbf{R}^m$ with $m = \sum_{j=1}^{N_{sys}} m_j$. The manipulated input vector constraints are defined by $u_j \in U_j := \{u_{j_i}^{\min} \leq u_{j_i} \leq u_{j_i}^{\max}, i = 1,...,m_j\} \subset \mathbf{R}^{m_j}, \forall j = 1,...,N_{sys}$. Therefore, the set that bounds the manipulated input vector u for the overall system is denoted by U, which is the union of all U_j, $j = 1,...,N_{sys}$. $f_j(\cdot)$ and $g_j(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n_j \times 1$ and $n_j \times m_j$, respectively. The initial time t_0 is taken to be zero ($t_0 = 0$). We assume that $f_j(0) = 0$, $\forall j = 1,...,N_{sys}$, thus, the origin is a steady-state of the nominal system of Eq. 8.1 (i.e., $u(t) \equiv 0$). Therefore, we have $(x_s, u_s) = (0,0)$, where x_s and u_s are the steady-state state and input vectors for the overall system, respectively. The overall system is described as follows:

$$\dot{x} = F(x, u_1, ..., u_{N_{sys}}) := f(x) + \sum_{j=1}^{N_{sys}} g_j(x) u_j \tag{4.2a}$$

$$\bar{x} = h(x) \tag{4.2b}$$

where $f(\cdot)$ represents the vector function of dimension $n \times 1$ for all states of the two subsystems $f = [{f_1}^T \ \cdots \ {f_{N_{sys}}}^T]^T$. $\bar{x} \in \mathbf{R}^n$ denotes the vector of full state measurements from sensors, which may be compromised by sensor cyber-attacks. When no cyber-attacks are present in the system, $\bar{x} = x$.

4.1.3 Stability Assumptions

Depending on the partitioning of the overall large-scale system, there may exist interacting dynamics between the subsystems, where the states of one subsystem may be impacted by the states of other subsystems. We assume that there exist stabilizing control laws $u_j = \Phi_j(x) \in U_j$ which regulate the individual subsystems $j = 1,...,N_{sys}$ and will be applied to the control actuators in the respective subsystems such that the origin of the overall system of Eq. 8.1 is rendered asymptotically stable. This implies that there exists a $\mathscr{C}^1$ control Lyapunov function $V(x)$ such that the following inequalities hold for all $x \in \mathbf{R}^n$ in an open neighborhood D around the origin:

$$c_1(|x|) \leq V(x) \leq c_2(|x|), \tag{4.3a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \mathbf{\Phi}(x)) \leq -c_3(|x|), \tag{4.3b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq c_4(|x|) \tag{4.3c}$$

where c_1, c_2, c_3 and c_4 are class $\mathscr{K}$ functions. $F(x,u)$ represents the overall nonlinear system of Eq. 8.1. $\mathbf{\Phi}(x) = [\Phi_1(x)^T \cdots \Phi_{N_{sys}}(x)^T]^T$ represents the vector containing the candidate control laws for each subsystem j, i.e., $\Phi_j(x) \in \mathbf{R}^{m_j}$, for $j = 1,...,N_{sys}$. The candidate controller for each subsystem j is given in the following form:

$$\phi_{j_i}(x) = \begin{cases} -\dfrac{p+\sqrt{p^2+q^4}}{q^T q} q & \text{if} \quad q \neq 0 \\ 0 & \text{if} \quad q = 0 \end{cases} \tag{4.4a}$$

$$\Phi_{j_i}(x) = \begin{cases} u_{j_i}^{min} & \text{if} \quad \phi_{j_i}(x) < u_{j_i}^{min} \\ \phi_{j_i}(x) & \text{if} \quad u_{j_i}^{min} \leq \phi_{j_i}(x) \leq u_{j_i}^{max} \\ u_{j_i}^{max} & \text{if} \quad \phi_{j_i}(x) > u_{j_i}^{max} \end{cases} \tag{4.4b}$$

Depending on whether the control system is decentralized or distributed, the candidate control laws of Eq. 7.7 may be calculated differently. In a decentralized control system, each controller has knowledge of the dynamics of the local subsystem that it regulates, but does not have access to the model dynamics of the entire process. Therefore, when the candidate controller of Eq. 7.7 is implemented, p in Eq. 7.7a denotes $\frac{\partial V(x)}{\partial x_j} f_j(x)$ and q denotes $\frac{\partial V(x)}{\partial x_j} g_{j_i}(x)$. Here, $f_j \in \mathbf{R}^{n_j}$ and $g_{j_i} \in \mathbf{R}^{n_j \times m_j}$ for $j = 1,...,N_{sys}$, and $i = 1,...,m_j$ for subsystem j corresponding to the vector of control actions $\Phi_j(x) \in \mathbf{R}^{m_j}$. The control Lyapunov function $V(x)$ can be a linear combination of multiple control Lyapunov functions V_j. Each V_j is designated for the subsystem j and is a function of x_j only, i.e., $V(x) = \sum_{j=1}^{N_{sys}} V_j(x_j)$. Thus, $\frac{\partial V(x)}{\partial x_j} = \frac{\partial V_j(x_j)}{\partial x_j}$, $\forall\ j = 1,...,N_{sys}$, and the time-derivative of V can be represented as $\dot{V}(x) = L_f V + L_g V u = \sum_{j=1}^{N_{sys}} \frac{\partial V(x)}{\partial x_j}(f_j + \sum_{i=1}^{m_j} g_{j_i} u_{j_i})$. On the other hand, in a distributed control system, each controller has knowledge of the dynamics of the overall process, and calculates the control actions for each corresponding subsystem accordingly. Thus, when the candidate controller $\Phi_{j_i}(x)$ of Eq. 7.7 is calculated, p denotes $L_f V(x) = \frac{\partial V(x)}{\partial x} f(x) = \sum_{j=1}^{N_{sys}} \frac{\partial V(x)}{\partial x_j} f_j(x)$ and q denotes $L_{g_{j_i}} V(x) = \frac{\partial V(x)}{\partial x_j} g_{j_i}$ for subsystems $j = 1,...,N_{sys}$, and $i = 1,...,m_j$ number of inputs for subsystem j; here, we only consider g_{j_i} because $\Phi_j(x) \in \mathbf{R}^{m_j}$ regulates the states $x_j \in \mathbf{R}^{n_j}$ of the subsystem j only.

$\phi_{j_i}(x)$ of Eq. 7.7a represents the i^{th} component of the control law $\phi_j(x)$. $\Phi_{j_i}(x)$ of Eq. 7.7 represents the i^{th} component of the saturated control law $\Phi_j(x)$ that accounts for the input constraints $u_j \in U_j$ for subsystem j. Note that the candidate control law $\Phi_j(x)$ is calculated based on the nonlinear dynamics of the subsystem j of Eq. 5.2.2, and the set of candidate control laws

for the overall system is denoted as $\mathbf{\Phi}(x) = [\Phi_1(x)^T \cdots \Phi_{N_{sys}}(x)^T]^T \in U$, which together can render the overall system of Eq. 8.1 asymptotically stable.

Based on Eq. 7.5, we first characterize a region where the time-derivative of the Control Lyapunov function V is rendered negative definite under the candidate control laws $\mathbf{\Phi}(x) \in U$ as $D = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V + L_g V u \leq -c_3(|x|), u = \mathbf{\Phi}(x) \in U\} \cup \{0\}$. Then, the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 8.1 is defined as a level set of V, which is an invariant set for the closed-loop system inside D: $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset D$. Therefore, starting from any initial state $x_0 := x(t_0)$ in Ω_ρ, $\mathbf{\Phi}(x(t))$ guarantees that the state trajectory of the closed-loop system of Eq. 8.1 remains within Ω_ρ and asymptotically converges to the origin. Thus, given that the sensor measurements received by the controller are secure and reliable (i.e., $\bar{x} = x$), the control law $\mathbf{\Phi}(x)$ is able to stabilize the process at the origin for any initial conditions $x_0 \in \Omega_\rho$.

4.2 Centralized, Decentralized, and Distributed Lyapunov-based Model Predictive Control

4.2.1 Centralized LMPC

Traditionally, when the overall process is regulated by a centralized controller, the control problem that is solved incorporates all the manipulated inputs and state measurements of the process. Specifically, the centralized Lyapunov-based Model Predictive Control (LMPC) is represented by the following optimization problem:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) dt \tag{4.5a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u(t)) \tag{4.5b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{4.5c}$$

$$\tilde{x}(t_k) = \bar{x}(t_k) \tag{4.5d}$$

$$\dot{V}(\bar{x}(t_k), u) \leq \dot{V}(\bar{x}(t_k), \mathbf{\Phi}(\bar{x}(t_k))), \quad \text{if } \bar{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_s} \tag{4.5e}$$

$$V(\tilde{x}(t)) \leq \rho_s, \ \forall \ t \in [t_k, t_{k+N}), \quad \text{if } \bar{x}(t_k) \in \Omega_{\rho_s} \tag{4.5f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. $\dot{V}(x,u)$ is used to represent $\frac{\partial V(x)}{\partial x}F(x,u)$. The optimal input trajectory computed by the centralized LMPC is denoted by $u^*(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*(t_k)$ is sent by the LMPC to be applied over the next sampling period in a sample-and-hold manner, and the centralized LMPC is solved again in a rolling horizon fashion. The LMPC of Eq. 4.5 is solved by minimizing the time integral of the objective function $L(\tilde{x}(t), u(t))$ of Eq. 4.5a over the prediction horizon N, subject to the constraints of Eqs. 4.5b-4.5f. The objective function is generally in a quadratic form of $\tilde{x}^T Q\tilde{x} + u^T Ru$ such that the states can be driven to the operating steady-state of the process without exhausting too much inputs; here, Q and R are the weighting matrices for the states and the inputs, respectively. Eq. 4.5c defines the input constraints applied over the entire prediction horizon, and Eq. 4.5d defines the initial condition $\tilde{x}(t_k)$ of Eq. 4.5b, which is the state measurement $\bar{x}(t)$ at $t = t_k$. The constraint of Eq. 4.5e forces the closed-loop state to move towards the origin at a minimum rate characterized by the Lyapunov function V and the Lyapunov-based control law $\mathbf{\Phi}(\bar{x}(t_k))$, if $\bar{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_s}$, where Ω_ρ is the stability region and Ω_{ρ_s} is the ultimate bounded region around the operating steady-state which the closed-loop states of Eq. 4.5b will converge to. If $\bar{x}(t_k)$ enters Ω_{ρ_s}, Eq. 4.5f ensures that the states predicted by Eq. 4.5b will be maintained in Ω_{ρ_s} for the entire prediction horizon.

4.2.2 Decentralized LMPC

When the optimization problem of a centralized MPC is too complex to solve within a reasonable time period (i.e., the sampling period), the control problem may be decoupled into smaller local optimization problems that are solved in separate processors/controllers to achieve improved computational efficiency. In a decentralized LMPC system, no communication is established between the different local controllers, therefore each controller does not have any knowledge on the control actions calculated by the other controllers.

We design separate $j = 1, ..., N_{sys}$ LMPCs, each designated to regulate the states x_j of one subsystem j, and compute the respective control actions. The trajectories of control actions computed by LMPC j are denoted by u_{d_j}, which are applied to the corresponding control actuators in subsystem j. Each decentralized LMPC may receive full-state feedback measurements, but they only have information on the dynamic behavior of their respective subsystem. The mathematical

formulation of each decentralized LMPC j, $j = 1, ..., N_{sys}$, is shown as follows:

$$\mathscr{J}_j = \min_{u_{d_j} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}_j(t), u_{d_j}(t))dt \tag{4.6a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}_j(t) = F_j(\hat{x}(t), u_{d_j}(t)) \tag{4.6b}$$

$$\hat{x}(t) = [\bar{x}_1(t_k)^T \cdots \bar{x}_{j-1}(t_k)^T \;\; \tilde{x}_j(t)^T \;\; \bar{x}_{j+1}(t_k)^T \cdots \bar{x}_{N_{sys}}(t_k)^T]^T \tag{4.6c}$$

$$u_{d_j}(t) \in U_j, \; \forall \, t \in [t_k, t_{k+N}) \tag{4.6d}$$

$$\tilde{x}_j(t_k) = \bar{x}_j(t_k) \tag{4.6e}$$

$$\frac{\partial V(\bar{x}(t_k))}{\partial x_j}(F_j(\bar{x}(t_k), u_{d_j}(t_k))) \leq \frac{\partial V(\bar{x}(t_k))}{\partial x_j}(F_j(\bar{x}(t_k), \Phi_j(\bar{x}(t_k)))),$$

$$\text{if } \bar{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_s} \tag{4.6f}$$

$$V(\tilde{x}(t)) \leq \rho_s, \; \forall \, t \in [t_k, t_{k+N}), \text{ if } \bar{x}(t_k) \in \Omega_{\rho_s} \tag{4.6g}$$

where $\tilde{x}_j$ is the predicted state trajectory for subsystem j, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon for subsystem j. The optimal input trajectory computed by this LMPC j is denoted by $u^*_{d_j}(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*_{d_j}(t_k)$ is sent by LMPC j to its control actuators in subsystem j to be applied over the next sampling period. The objective function of Eq. 5.2.13a is the integral of $L(\tilde{x}_j(t), u_{d_j}(t))$ over the prediction horizon, where $L(x_j, u_j)$ is typically in a quadratic form of $x_j^T Q_j x_j + u_j^T R_j u_j$, and Q_j and R_j are the weighting matrices of the states and the inputs of subsystem j respectively. The states of each subsystem will be driven towards the origin by minimizing this objective function. The constraint of Eq. 5.2.13b is the first-principles model of Eq. 5.2.2 used to predict the states of the closed-loop subsystem j. $\hat{x}(t)$ is a vector containing the predicted states of subsystem j, $\tilde{x}_j(t)$, and the measured states of all other subsystems at $t = t_k$, $\bar{x}_i(t_k)$, $\forall \, i = 1, ..., N_{sys}$, and $i \neq j$. Eq. 5.2.13d is the input constraints on u_{d_j} applied over the entire prediction horizon, and Eq. 5.2.13e defines the initial conditions of Eq. 5.2.13b for subsystem j, which is the state measurement $\bar{x}_j(t)$ at $t = t_k$. Eq. 5.2.13f is a constraint that forces the closed-loop state of subsystem j to move towards the origin at a minimum rate characterized by the Lyapunov function V and the Lyapunov-based control law $\Phi_j(\bar{x}(t_k))$ if $\bar{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_s}$. If $\bar{x}(t_k)$ enters the terminal set region Ω_{ρ_s}, then the states $\tilde{x}_j$ predicted by Eq. 5.2.13b will be maintained in Ω_{ρ_s} for the entire prediction horizon.

Since the decentralized LMPCs solve different optimization problems specific to their

respective subsystems and are computed in separate processors in parallel, the computation time for solving one iteration of the entire decentralized LMPC design in one sampling period (assuming that the feedback measurements are available to both controllers at synchronous intervals) will be the maximum time out of the two LMPCs.

4.2.3 Distributed LMPC

To achieve better closed-loop control performance, some level of communication may be established between the different controllers. In this study, we consider an iterative distributed MPC system, which allows signal exchanges between all controllers, thereby allowing each controller to have full knowledge of the predicted state evolution along the prediction horizon and yielding better closed-loop performance via multiple iterations at the cost of more computational time. In the context that two LMPCs are designed, both controllers communicate with each other to cooperatively optimize the control actions. The controllers solve their respective optimization problems independently in a parallel structure, and solutions to each control problem are exchanged at the end of each iteration. More specifically, the following implementation strategy is used:

1. At each sampling instant t_k, each LMPC j, $j = 1,...,N_{sys}$, receives the state measurement $\bar{x}(t)$ at $t = t_k$ from all the sensors.
2. At iteration $c = 1$, each LMPC j evaluates future trajectories of $u_{d_j}(t)$ assuming the control actions of all other subsystemsare calculated by the Lyapunov-based control law, $u_i(t) = \Phi_i(\bar{x}(t_k)), \forall t \in [t_k, t_{k+N})$, $i = 1,...,N_{sys}, i \neq j$. The LMPCs then exchange their future input trajectories, calculate and store the value of their own objective function.
3. At iteration $c > 1$:
   (a) Each LMPC evaluates its own future input trajectory based on state measurement $\bar{x}(t_k)$ and the latest received input trajectories from the other LMPCs.
   (b) The LMPCs exchange their future input trajectories. Each LMPC calculates and stores the value of the cost function.
4. When a termination criterion is satisfied, each LMPC sends its entire future input trajectory corresponding to the smallest value of the cost function to its actuators. If the termination criterion is not satisfied, go to back Step 3 ($c \leftarrow c+1$).
5. When a new state measurement $\bar{x}$ is received, go to Step 1 ($k \leftarrow k+1$).

Following the same variables and constraints as defined in a decentralized LMPC design, the optimization problem of LMPC 1 in an iterative distributed LMPC at iteration $c = 1$ is presented as follows:

$$\mathscr{J} = \min_{u_{d_j} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_j}(t), \Phi_i(\tilde{x}(t)))dt \tag{4.7a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_j}(t), \Phi_i(\tilde{x}(t))) \tag{4.7b}$$

$$u_{d_j}(t) \in U_j, \ \forall\, t \in [t_k, t_{k+N}) \tag{4.7c}$$

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

$$\frac{\partial V(\bar{x}(t_k))}{\partial x}(F(\bar{x}(t_k), u_{d_j}(t_k), \Phi_i(\bar{x}(t_k)))) \leq \frac{\partial V(\bar{x}(t_k))}{\partial x}(F(\bar{x}(t_k), \mathbf{\Phi}(\bar{x}(t_k)))),$$

$$\text{if } \bar{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s} \tag{4.7e}$$

$$V(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } \bar{x}(t_k) \in \Omega_{\rho_s} \tag{4.7f}$$

where for each control action j corresponding to subsystem j, $i = 1, ..., N_{sys}, i \neq j$, which refers to the control actions of all other subsystems except for j. At iteration $c > 1$, after the exchange of the optimized input trajectories $u^*_{d_j}(t), \forall t \in [t_k, t_{k+N})$ between all LMPCs $j = 1, ..., N_{sys}$, the optimization problem of LMPC j is as follows:

$$\mathscr{J} = \min_{u_{d_j} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_j}(t), u^*_{d_i}(t))dt \tag{4.8a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_j}(t), u^*_{d_i}(t)) \tag{4.8b}$$

$$u_{d_j}(t) \in U_j, \ \forall\, t \in [t_k, t_{k+N}) \tag{4.8c}$$

$$\tilde{x}(t_k) = \bar{x}(t_k) \tag{4.8d}$$

$$\frac{\partial V(\bar{x}(t_k))}{\partial x}(F(\bar{x}(t_k), u_{d_j}(t_k), u^*_{d_i}(t_k)) \leq \frac{\partial V(\bar{x}(t_k))}{\partial x}(F(\bar{x}(t_k), \mathbf{\Phi}(\bar{x}(t_k)))),$$

$$\text{if } \bar{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s} \tag{4.8e}$$

$$V(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } \bar{x}(t_k) \in \Omega_{\rho_s} \tag{4.8f}$$

At each iteration $c \geq 1$, the two LMPCs can be solved simultaneously via parallel computing in separate processors. Therefore, the total computation time required for iterative distributed LMPC is the maximum solving time out of the two controllers accounting for all the iterations required before the termination criterion is met. The termination condition can be of many forms, e.g., c must not exceed a maximum iteration number c_{max} (i.e., $c \leq c_{max}$), the computational time for solving each LMPC must not exceed a maximum time period, or the difference in the cost function

or of the solution trajectory between two consecutive iterations is smaller than a threshold value. During implementation, when one such criterion is met, the iterations will be terminated.

Once the optimization problems of all subsystems $j = 1, ..., N_{sys}$ are solved, the optimal control actions of the proposed decentralized or distributed LMPC systems are defined as follows:

$$u_j(t) = u_{d_j}^*(t_k),\ j = 1, ..., N_{sys},\ \forall\, t \in [t_k, t_{k+1}) \tag{4.9}$$

4.3 Intelligent Cyber-Attacks

With the intention of destroying the control objectives of the system, cyber-attacks can jeopardize system stability. Intelligent cyber-attacks could target sensors, actuators, and/or the communication channels in the system [16]. In this work, we only consider sensor cyber-attacks. Falsified sensor feedback measurements that do not accurately report the true states of the process will result in incorrect control actions being calculated by the controllers, which may ultimately drive the true process states outside of the stability region. The cyber-attacks may have access to the stability region which the controllers are designed to operate the process within, and correspondingly they may have knowledge on any existing alarms configured on the input and output variables such that no alarms will be triggered to bring human attention and intervention, and the controllers are still able to compute feasible control actions based on the falsified sensor measurements.

4.3.1 Design of Cyber-Attacks on Sensors

Sensor cyber-attacks can target any sensor in the overall system, regardless of the structure of the system being centralized, decentralized, or distributed. There are some standard types of cyber-attacks considered in literature [22, 97], and the formulation details of three of them are shown in this section. For simplicity, the following cyber-attacks will be introduced in the context of a decentralized LMPC system where the Lyapunov function of each subsystem V_j is assessed independently. However, the same form of cyber-attacks can be applied to systems under centralized and distributed control as well. Note that V_j is a function of x_j exclusively. Therefore, the design of the cyber-attacks is with respect to the stability region of each subsystem respectively.

Remark 4.1. *In this work, we only consider sensor cyber-attacks. However, cyber-attacks can also happen in actuators or in any communication channels. Specifically, if the attacker*

has knowledge on the architecture of the control system being distributed, the attacks could be injected in the inter-controller communication channels. The controllers communicate their future input trajectories with each other, which gives information on the future predicted states of other subsystems. If such information is falsified, the controllers will calculate incorrect control actions based on the received wrongful information, and these incorrect control actions will get communicated to other controllers again through iterations, and the impact of the cyber-attack will propagate and magnify. Investigating the inter-controller communication channel attacks and detecting such attacks based on sensor data or other forms of metrics will be an interesting future research direction.

4.3.1.1 Min-Max Cyber-attack

Maximum disruption impact is achieved by a min-max attack within a short period of time without triggering any alarms. The falsified sensor measurement will be set to a value furthest away from the value of the current true state without exiting the stability region, and thus the resulting falsified measurements $\bar{x}_j$ will be on the boundary of the stability region, i.e., $V_j(\bar{x}_j) = \rho_j$. Depending on the value of x_j, the furthest value from the current state might be the maximum or the minimum as characterized by the boundary of the stability region. Therefore, the min-max attack on one particular subsystem j can be formulated as follows:

$$\bar{x}_j(t_i) = \arg(\min / \max)_{x_j}\{x_j \in \mathbf{R^{n_j}} \mid V_j(x_j(t_i)) = \rho_j\}, \ \ \forall\, i \in [i_0, i_0 + L_a] \tag{4.10}$$

where i_0 is the time instant that the attack is injected, $\bar{x}_j$ denotes the compromised sensor measurement, L_a is the total duration of the attack in terms of sampling periods, ρ_j refers to the level set of the Lyapunov function $V_j(x_j)$ that characterizes the stability region of the closed-loop subsystem of Eq. 5.2.2 under the decentralized LMPC.

4.3.1.2 Surge Cyber-attack

Surge attacks induce maximum deviation for an initial short period, and then the attacked value is set to a reduced value thereafter such that the cumulative deviation will not exceed a certain threshold, which is typically examined by conventional detection methods such as Cumulative Sum [16, 80]. The initial surge value that causes maximum impact is also defined based on the stability region of the subsystem j, Ω_{ρ_j}. Designing the length of the initial surge period and the attacked sensor measurement after the surge can be of many forms, as long as the cumulative error between state measurements and their predicted true values (usually given by an estimation

algorithm) for the entire attack duration does not exceed the statistics-based threshold defined in other detection methods (e.g., CUSUM). For our study, the attack after the initial surge is designed to act as a sufficiently small bounded noise imposed on the attacked sensor. The formulation of the surge attack is as follows:

$$\begin{aligned} \bar{x}_j(t_i) &= \arg(\min / \max)_{x_j}\{x_j \in \mathbf{R^{n_j}} \mid V_j(x_j(t_i)) = \rho_j\}, \text{ if } i_0 \leq i \leq i_0 + L_s \\ \bar{x}_j(t_i) &= x_j(t_i) + \eta(t_i),\ V_j(\bar{x}_j(t_i)) < \rho_j, \text{ if } i_0 + L_s < i \leq i_0 + L_a \end{aligned} \tag{4.11}$$

where L_s is the duration of the initial surge, and $\eta_l \leq \eta(t_i) \leq \eta_u$ is the bounded noise added on the sensor measurement after the initial surge period, where η_l and η_u are the lower and upper bounds of the noise, respectively. Here, the reduced attack value is designed to have much lesser magnitude and lower impact compared to the min/max value achieved during the initial surge so that the attack could last for a long time without being detected by conventional detection methods. Therefore, the Lyapunov function of the reduced attack measurement should not exceed the boundary of the stability region, i.e., $V_j(\bar{x}_j) < \rho_j$. In the case that the additive noise does generate $\bar{x}_j(t_i)$ such that $V_j(\bar{x}_j(t_i)) > \rho_j$ due to the true state x_j already outside of the stability region, $\bar{x}_j$ after the surge would be set to the closest point on the boundary of the stability region to the true state value until the true state returns inside the stability region, after which $\bar{x}_j$ would take the form of having a sufficiently small bounded noise added onto the true state measurement.

4.3.1.3 Geometric Cyber-attack

Under geometric cyber-attacks, closed-loop system stability deteriorates at a geometric speed until the attacked value reaches the maximum or minimum allowable limit as characterized by the stability region of the subsystem. Geometric attacks can be written as follows:

$$\bar{x}_j(t_i) = x_j(t_i) + \beta \times (1+\alpha)^{i-i_0},\ \ \forall\, i \in [i_0, i_0 + L_a] \tag{4.12}$$

where β and α define the initial magnitude and the speed of the geometric attack, respectively. A small constant $\beta \in \mathbf{R}$ is added to the true measured output $x_j(t_{i_0})$ at the start of the attack such that $x_j(t_{i_0}) + \beta$ is well below the alarm threshold. At each subsequent time step after t_{i_0}, β is multiplied by a factor $(1+\alpha)$, $\alpha \in (0,1)$, until $\bar{x}_j$ reaches the maximum allowable attack value bounded by Ω_{ρ_j}. $\bar{x}_j$ is increasing or decreasing geometrically depending on the sign of the parameter β. Attackers will choose the two parameters α and β based on Ω_{ρ_j} and the attack duration L_a.

4.3.2 Robustness of Decentralized LMPC against Cyber-Attacks

Both decentralized and distributed LMPC systems are designed to alleviate the computational complexity and effort of a centralized control problem regulating multiple subsystems. Considering the inherent structure and operating requirement of both systems, the decentralized control system exhibits greater robustness against potential cyber-attacks. Firstly, there must exist inter-controller communication in a distributed control system to exchange information on the control actions calculated by each distributed local controller. With additional communication channels between controllers built for this purpose, it creates a greater exposure surface that is vulnerable to cyber-attacks targeting communication channels. Controllers in the decentralized control system do not share any information, and therefore, eliminating this layer of potential threat. Secondly, every local controller in a distributed control system has knowledge on how the overall process dynamically evolves, and receives full-state measurements of the entire process at every sampling period as required. This means that, if any one of the many sensors of the entire system is falsified, it would result in erroneous calculations in all local controllers, and the error would propagate as the incorrect control actions calculated by the controllers are exchanged. On the other hand, decentralized controllers only have knowledge of the process dynamics of the local subsystem, and are designed to only regulate the states of the respective subsystem. Depending on how the overall process is partitioned, local controllers in the decentralized control system may not need information on feedback measurements of the process states of other subsystems. Therefore, in the case that only one or a few subsystems are attacked, the decentralized controllers regulating those un-attacked subsystems will not be affected at all, and are still able to maintain local subsystem stability. This will be demonstrated in the simulation studies in Section 4.5.

4.4 Detection of Cyber-Attacks

Since physical processes may be prone to structural or parameter changes, and sophisticated stealthy attacks may have knowledge on the process behavior and the underlying model, model-based detection algorithms may be less effective than data-based detection methods [16]. Most model-based detection methods – such as the Cumulative Sum method – rely on the availability of an accurate model that describes the dynamics of the process, and they determine the presence of an anomaly or a cyber-attack in the system by examining the discrepancy between a model-based predictive estimate and the actual process output. There are some disadvantages to these model-based methods. One disadvantage is that during real-time operation, the process

model is subject to structural and parameter changes, and therefore the model used to construct the detector also needs to be adjusted to reflect the same changes. Another disadvantage is that, the intelligent cyber-attacks may have information on the setup of the control system including the design of the model-based detector, and therefore, could adopt attack tactics that counteract the detection scheme. For example, a Cumulative Sum method would not be able to detect surge attacks or any other forms of attacks that have prior knowledge on the detection threshold imposed on the cumulative error in the state measurement. Therefore, these potential shortcomings provide motivation for developing data-based detection methods which are independent from the physical process model and are resilient to process changes and stealthy attacks designed based on process behavior. Amongst data-based methods, machine learning algorithms provide many advanced and flexible capabilities of classification and regression; more specifically, by using different categories of training data and constructing different training algorithms, the machine-learning-based detector may be designed and optimized to perform a variety of different detection tasks. The interested readers may also refer to [118] for a numerical analysis on the detection accuracy of a machine-learning-based detection method and a model-based statistics detection method. Neural networks (NN), as one of the advanced machine-learning techniques, have proven to be successful in a variety of applications, solving both supervised and unsupervised classification problems. One advantage of NN over other classification methods such as k-nearest-neighbors, random forest, and support vector machine, is that a large number of nonlinear activation functions, tuning parameters, and alternative training algorithms can be used to optimize the overall model accuracy [109].

In our study, we use a feed-forward neural network (FNN) for supervised classification. In supervised classification, the NN will be trained using data with labels corresponding to each target class, and its task will be to classify new data samples into the respective known classes that the new data identifies the most similarities with. A conventional FNN consists of an input layer, an output layer, and a customizable number of hidden layers in between. Each layer undergoes a series of nonlinear functions, which are activation functions of the weight sum of inputs (or neurons in the previous layer) with a bias term, and provides the values for the neurons in the current layer. The neurons in the first hidden layer are derived from the inputs, and the outputs are calculated from the neurons in the last hidden layer.

As the accuracy of the FNN model depends heavily on the types and quality of the training data, selecting the input features for the FNN model that effectively and concisely capture the evolution of the process states is critical. Considering the control objective of the system, the states of the nonlinear subsystems of Eq. 5.2.2 are driven towards the operating steady-state

under the decentralized or the distributed LMPC subject to Lyapunov-based constraints. The Lyapunov function of the subsystem, $V_j(\bar{x}_j)$, which is a function of the measured process states $\bar{x}_j$ of the respective subsystem j only, captures the dynamic features of all states. Thus, $V_j(\bar{x}_j)$ is an effective one-dimensional parameter used as the input variables for the attack detection problem. More specifically, since multiple subsystems are involved, we record the Lyapunov function of all subsystems using the state measurements recorded along the time-varying trajectory, i.e., $V_1(\bar{x}_1(t_i))$, $V_2(\bar{x}_2(t_i))$, $i = 1,...,N_T$, and concatenate V_1 and V_2 as a single one-dimensional vector of dimension $1 \times (2N_T)$. The resulting vector contains information on the evolution of the process states of the two subsystems independently, and will be used as the input vector into the feed-forward NN detector model to determine whether abnormalities exist in any of the sensor measurements in the latest time window of $\Delta \times N_T$, where Δ is the sampling period of the LMPC system. Since the information of the state measurements of the two subsystems are recorded and presented in the detector input vector independently, depending on the structure of the NN detector model and the classes of the training data, the FNN detector can be trained to either detect the presence of an attack anywhere in the overall system, or identify the location of an attack (i.e., which subsystem the problematic sensors are located).

The structure of a basic one-hidden-layer FNN model for cyber-attack detection is shown in Fig. 6.1. On the input layer, each neuron represents the Lyapunov function $V_j(\bar{x}_j(t_i))$ at time instant t_i, for $j = 1,2$, $i = 1,...,N_T$, and resulting in a total of $p = 1,...,2N_T$ neurons. The output layer provides an output vector the the predicted class label, where the number of neurons corresponds to the number of possible classes. The mathematical formulation of a one-hidden-layer FNN model is shown as follows:

$$\theta_k^{(1)} = g_1\Big(\sum_{p=1}^{2N_T} w_{pk}^{(1)} V_j(\bar{x}_j(t_i)) + b_k^{(1)}\Big) \tag{4.13a}$$

$$\theta_k^{(2)} = g_2\Big(\sum_{p=1}^{h} w_{pk}^{(2)} \theta_p^{(1)} + b_k^{(2)}\Big) \tag{4.13b}$$

$$y_{pred} = [\theta_1^{(2)}, \theta_2^{(2)}, ..., \theta_H^{(2)}]^T \tag{4.13c}$$

with $\theta_k^{(1)}$ representing neurons in the hidden layer layer, where $k = 1,...,h$ is the number of neurons in hidden layer. $\theta_k^{(2)}$ represents neurons in the output layer, where $k = 1,...,H$, and H is the number of class labels. In this work, we use one hidden layer for this FNN detector model. Using the similar formulations as Eq. 6.6, multiple hidden layers can also be constructed. The input layer for each sample consists of variables $V_1(\bar{x}_1(t_i))$ and $V_2(\bar{x}_2(t_i))$ for $i = 1,...,N_T$. The weights

connecting neurons p and k in consecutive layers (from layer $l-1$ to layer l) are $w_{pk}^{(l)}$, and the bias term on the k^{th} neuron in the l^{th} layer is $b_k^{(l)}$. Each layer calculates an output using the information received from the previous layer, the optimized weights, biases, which are all passed into a nonlinear activation function g_l (some examples include hyperbolic tangent transfer function $g(z) = \frac{2}{1+e^{-2z}} - 1$, sigmoid function $g(z) = \frac{1}{1+e^{-z}}$, and softmax function $g(z_k) = \frac{e^{z_k}}{\sum_{i=1}^{H} e^{z_i}}$ where H is the number of class labels). The output layer consists of a vector, y_{pred}, which gives the predicted probabilities of each class label, where the neuron with the highest probability indicates the final predicted class label for the examined sample. Depending on the classification problem that the neural network is trained to solve, the predicted class label can provide information on either the presence of a cyber-attack, where the cyber-attack occurs, or the type of cyber-attack.

To obtain an optimal set of weights and biases in Eq. 6.6, the solver *Adam* is used to minimize a binary cross-entropy loss function, which has the following form:

$$S(w) = (N_s \cdot \sum_{q=1}^{N_s} y_{actual,q} \cdot ln(y_{pred,q}))^{-1} \tag{4.14}$$

where $q = 1, ..., N_s$ represents the number of samples in the training dataset, y_{actual} is the vector of target class labels of each sample, and y_{pred} is the vector of the predicted probabilities associated with each class label.

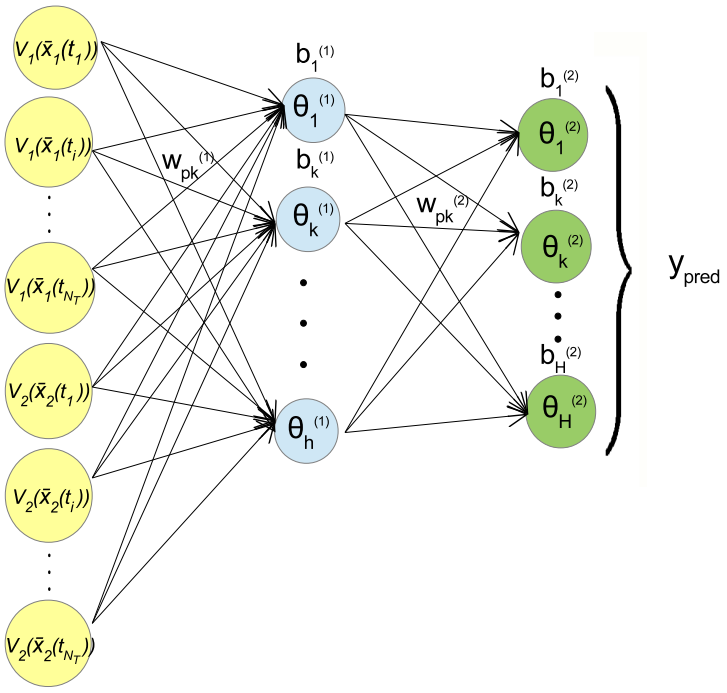


Figure 4.1: Feed-forward neural network structure with 1 hidden layer with inputs being the vector of Lyapunov functions of two subsystems $V_1(\bar{x}_1(t_i))$ and $V_2(\bar{x}_2(t_i))$ with a detection window $i = 1,...,N_T$, and output being the probability of each class label for the examined trajectory of length N_T indicating the status and/or location of cyber-attack.

To build an FNN detector model, training data samples are collected, which consist of closed-loop time-varying trajectories of the specific input vector variable suited for the detector model, generated from simulations under different attack scenarios. Sensor cyber-attacks with varying duration L_a are introduced at random times t_{i_0} on specific sensor(s) during the simulation period $N_T \times \Delta$, which is the same length as the detection window. We consider a system where some sensors are attacked and some remain intact. If no attacks occur anywhere in the system within $N_T \times \Delta$, then the measurement signals are labeled and should be classified as "no attack". In order to improve the detection accuracy, various closed-loop state evolutions within the stability region Ω_ρ need to be accounted for; therefore, training data is collected for a broad range of initial conditions within the stability region. Full state measurements $\bar{x}(t)$ are recorded along the time-varying trajectory for $t \in [t_0, t_{N_T}]$, and the Lyapunov functions of both subsystems are computed. Each training sample reflects a different set of initial conditions, and within each class labels, equal number of samples are collected to ensure training accuracy. Training and testing

accuracies are the ratios between the number of correctly classified samples and total number of samples in the training and testing sets, respectively.

4.4.1 Online Detection

When the FNN detector is implemented online with the process controlled by the centralized, the decentralized, or the distributed LMPC system, it uses a moving horizon detection window of a fixed length N_T, which is the same length as the time-varying trajectories of the training data. The detector is activated every time full-state measurements become available; it receives the latest sequences of the process state measurements of all subsystems with a fixed length N_T, and computes the Lyapunov function V_j of each subsystem j respectively. The values of V_j along the time-varying trajectory of length N_T will then be concatenated into a single one-dimensional vector to be used as the input vector for the FNN detector. If the FNN detector is trained to differentiate between "not attacked" and "attacked", then at every sampling period, it makes a decision on whether the sensor measurements over the latest time period of $\Delta \times N_T$ have been tampered. In addition, if the FNN detector is trained with multiple classes where each class represents one particular subsystem being attacked, then by examining the concatenated input vector containing all $V_j, j = 1, ..., N_{sys}$ along the detection window N_T, the detector will determine the particular subsystem where the tampered sensors are located. When the occurrence of a sensor cyber-attack has been confirmed, the sensor measurements can no longer be trusted, and all sensors will be switched for secure back-up sensors to completely mitigate the attack.

Remark 4.2. *The availability of back-up sensors is always necessary in case of instrument failure, or in this case, sensor cyber-attacks. These redundant sensors are not connected to the online system until a necessary scenario arises, and therefore will remain secure to any cyber-attacks that target control system in real time. Here, we propose one strategy for mitigating the impact of the cyber-attack by physically switching out the problematic sensors; other mitigation measures have also been proposed in our previous work in [120] where we use a recurrent neural network model to reconstruct tampered state measurements and restore system stability by using these state observers.*

4.5 Application to a Two-CSTR-in-Series Process

In this study, we simulate a chemical process consisting of two well-mixed, non-isothermal continuous stirred tank reactors (CSTRs) that operate in series. An irreversible second-order exothermic reaction takes place in each reactor, where the feed reactant A is transformed into product B. Reactant material A is fed into each of the two reactors $j = 1,2$, with inlet concentrations C_{Aj0}, inlet temperatures T_{j0} and the reactor feed volumetric flow rates F_{j0}. On each CSTR, a heating jacket is installed to supply and remove heat at a rate Q_j, $j = 1,2$. Considering the material and energy balances of the overall process, the dynamic models of this two-CSTR-in-series process can be represented as follows:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_{L_1}}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{4.15a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_{L_1}}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_{L_1}} \tag{4.15b}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_{L_2}} C_{A20} + \frac{F_{10}}{V_{L_2}} C_{A1} - \frac{F_{10} + F_{20}}{V_{L_2}} C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{4.15c}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_{L_2}} T_{20} + \frac{F_{10}}{V_{L_2}} T_1 - \frac{F_{10} + F_{20}}{V_{L_2}} T_2 + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_{L_2}} \tag{4.15d}$$

where C_{Aj}, V_{L_j}, T_j and Q_j, $j = 1,2$ are the concentration of reactant A, the volume of the reacting liquid, the temperature, and the heat input rate in the first and the second reactor, respectively. The reacting liquid has a constant density of ρ_L and a constant heat capacity of C_p for both reactors. E, R, ΔH, and k_0 represent the activation energy, ideal gas constant, enthalpy of the reaction, and pre-exponential constant, respectively. Table 5.2.1 lists all parameter values of this process.

For both CSTRs, the manipulated inputs are the inlet concentration of species A and the heat input rate supplied by the heating jacket, which are represented by the deviation variables $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1,2$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{Aj0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q_j| \leq 5 \times 10^5\ kJ/hr$, $j = 1,2$. The states of the closed-loop system are the concentration of species A and the temperature in the first and the second reactor, which are also represented by their deviation variables such that the equilibrium point of the system is at the origin of the state-space. Therefore, the vector of closed-loop states is $x = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}\ \ C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]^T$, where C_{A1_s}, C_{A2_s}, T_{1_s} and T_{2_s} are the steady-state values of concentration of A and temperature in each of the two tanks, respectively.

We analyze and compare three different control architectures in this example. In a centralized

Table 4.1: Parameter values of the two CSTRs in series.

$T_{10} = 300\ K$	$T_{20} = 300\ K$
$F_{10} = 5\ m^3/hr$	$F_{20} = 5\ m^3/hr$
$V_{L_1} = 1\ m^3$	$V_{L_2} = 1\ m^3$
$T_{1_s} = 401.9\ K$	$T_{2_s} = 401.9\ K$
$C_{A1_s} = 1.954\ kmol/m^3$	$C_{A2_s} = 1.954\ kmol/m^3$
$C_{A10_s} = 4\ kmol/m^3$	$C_{A20_s} = 4\ kmol/m^3$
$Q_{1_s} = 0.0\ kJ/hr$	$Q_{2_s} = 0.0\ kJ/hr$
$k_0 = 8.46 \times 10^6\ m^3/kmol\ hr$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$C_p = 0.231\ kJ/kg\ K$	$R = 8.314\ kJ/kmol\ K$
$\rho_L = 1000\ kg/m^3$	$E = 5 \times 10^4\ kJ/kmol$

framework, the centralized controller receives feedback measurements of all states x, and computes the manipulated inputs for the entire system, $u = [\Delta C_{A10}\ \ \Delta Q_1\ \ \Delta C_{A20}\ \ \Delta Q_2]^T$. The objective function in the centralized LMPC optimization problem is $L(x,u) = x^T Q x + u^T R u$, where $Q = diag[2\times 10^3\ \ 1\ \ 2\times 10^3\ \ 1], R = diag[8\times 10^{-13}\ \ 0.001\ \ 8\times 10^{-13}\ \ 0.001]$. In a decentralized scheme, two LMPCs are designed for the two subsystems respectively, and they are solved in separate processors independently without any inter-controller communications. Due to the structure and the interactions between the two CSTRs, the dynamic behvaior of the second CSTR is impacted by that of the first CSTR, but not vice versa. Thus, LMPC 1 receives only local state feedback measurements of the first CSTR as other state measurements in the system are not needed to compute the manipulated control actions for the first CSTR. To predict the process states of the second CSTR, LMPC 2 needs feedback information on all states, and therefore receives feedback measurements of both CSTRs to calculate the control actions for the second CSTR. The objective function in the decentralized LMPC j optimization problem has the form $L_j(x_j,u_j) = x_j^T Q_j x_j + u_j^T R_j u_j$, where $Q_j = diag[2\times 10^3\ \ 1]$, $R_j = diag[8\times 10^{-13}\ \ 0.001]$, for $j = 1,2$, and $u_j, j = 1,2$ denote the manipulated input vectors of each subsystem j. In a distributed system, both LMPCs are designed to predict the state evolution of the entire process, and then calculate the respective control actions for each CSTR accordingly. Therefore, feedback measurements of both CSTRs are received by both LMPCs in the distributed system. The objective function in each distributed LMPC is $L(x,u_1,u_2) = x^T Q x + u_1^T R_1 u_1 + u_2^T R_2 u_2$, where $Q = diag[2\times 10^3\ \ 1\ \ 2\times 10^3\ \ 1]$, $R_1 = R_2 = diag[8\times 10^{-13}\ \ 0.001]$, and $u_1 = [\Delta C_{A10}\ \ \Delta Q_1]^T$, $u_2 = \Delta C_{A20}\ \ \Delta Q_2]^T$. We assume that the

feedback measurements of all states are available at the same synchronous intervals $\Delta = 10^{-2}$ *hr*. The control Lyapunov function for each decentralized LMPC j is $V_j(x_j) = x_j^T P_j x_j$, for $j = 1,2$, and the control Lyapunov function for the centralized and the distributed LMPCs is the sum of the control Lyapunov functions for the two CSTRs, i.e., $V(x) = V_1(x_1) + V_2(x_2) = x_1^T P_1 x_1 + x_2^T P_2 x_2$, with the following positive definite P_j matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{4.16}$$

The closed-loop stability region Ω_{ρ_j} for each subsystem $j = 1,2$, defined by the largest level sets of the control Lyapunov functions for subsystem 1 and 2 respectively, are $\rho_1 = \rho_2 = 392$. The ultimate bounded regions $\Omega_{\rho_{s_j}}$ are chosen to be $\rho_{s_j} = 7$, for $j = 1,2$; the positive constants ρ_j and ρ_{s_j} are determined via extensive closed-loop simulations with $u_j \in U_j$.

To numerically simulate the dynamic model of Eq. 8.45, we use the explicit Euler method with an integration time step of $h_c = 10^{-4}$ *hr*. The nonlinear optimization problems of the centralized, the decentralized, and the distributed LMPCs are solved with the same sampling period of $\Delta = 10^{-2}$ *hr* using PyIpopt, which is a Python module of the IPOPT software package.

4.5.1 Closed-loop Performance without Detection

To illustrate the impact of the three types of sensor cyber-attacks considered for this system, we compare the closed-loop performance of the decentralized LMPC system when attacked and when not attacked. Fig. 4.2 shows in state-space plot the closed-loop evolution of the process states of the second CSTR (i.e., $C_{A2} - C_{A2s}$ and $T_2 - T_{2s}$) under the decentralized LMPC system when un-attacked and when one type of sensor attack has been injected during operation. The three types of attacks – min-max, geometric, and surge attacks – are respectively added on the temperature sensor of the second CSTR at time $t = 0.04$ *hr* and lasted until the end of the simulation period of 0.3 *hr*. The tampered sensor measurements, as well as the true process states are shown in the same plots. We can see that the falsified sensor measurements remain within the stability region boundaries, communicating to the controller that the solution to the optimization problem of the system is still feasible, causing the controller to calculate incorrect control actions with respect to the true process state. As a result, the true process states of the system are eventually driven outside of the stability region without triggering any alarms in the absence of an adequate detection mechanism.

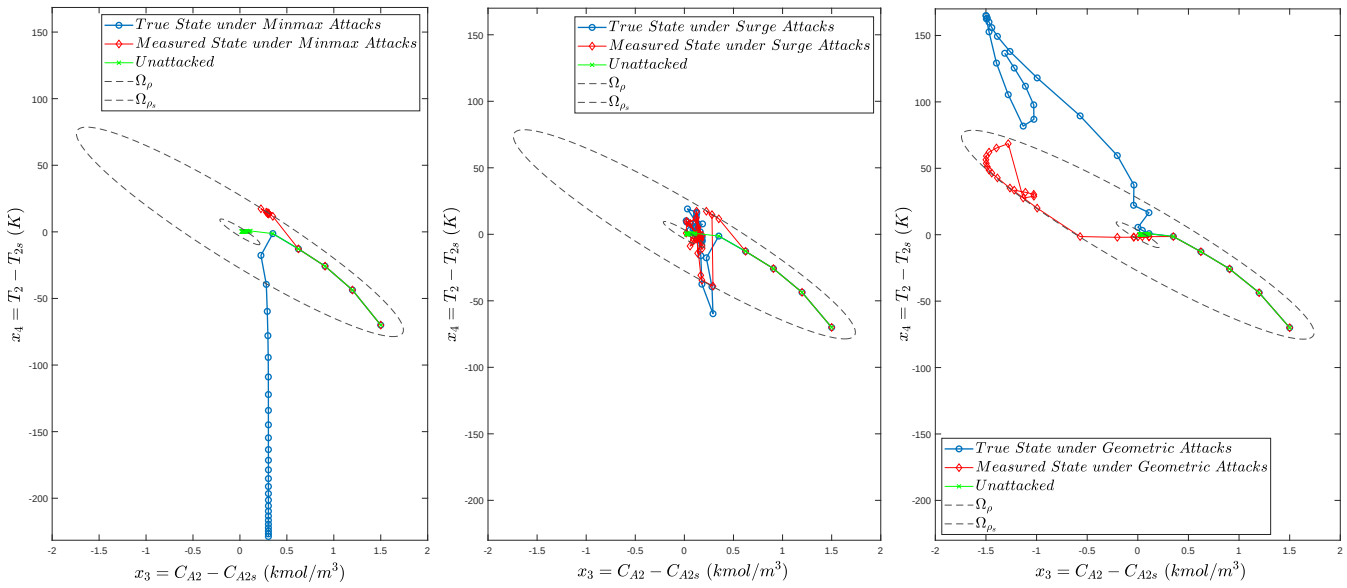


Figure 4.2: State-space closed-loop trajectories of the true, the measured, and the un-attacked process states of the second CSTR under the decentralized LMPC system when the temperature sensor T_2 is attacked by min-max, surge, and geometric attacks, respectively.

While sensors of one subsystem may be compromised, its impact on other subsystems depends on the structure and the resulting actions of the control system. Figs. 4.3 – 4.5 show the true state profiles of the two-CSTR process of Eq. 8.45 operated under the centralized, the decentralized, and the distributed LMPC systems when under min-max, surge, and geometric sensor attacks, respectively. All types of attacks are added on the temperature sensor T_2 of the second CSTR starting at $t = 0.05$ hr. The measured states from attacked sensors are not shown in these figures because depending on the location of the true process state, the attacked sensor measurement will adjust and show different patterns as well. Since the problematic sensor is in the second CSTR subsystem, the true states of the second CSTR (x_3 and x_4) under the centralized, decentralized, and distributed LMPC systems all exhibit destabilized behavior and are no longer driven to the operating steady-state as a result of the cyber-attack. The true states x_1 and x_2 of the first CSTR subsystem, however, may or may not be impacted by the sensor attacks in the second CSTR subsystem depending on the closed-loop control system. The decentralized LMPC system demonstrates robustness against sensor cyber-attacks on the temperature sensor of the second CSTR, and ensures the stability of the intact subsystem (i.e., x_1 and x_2 which belong to the first CSTR), which has not experienced cyber-attacks. However, when the process is operated under the centralized and the distributed LMPC systems, varying degrees of oscillations and deviations are seen in the true state profiles of the first CSTR. Moreover, the true state profiles under the

centralized LMPC demonstrate significantly worse performance for both CSTR subsystems when under min-max and surge attacks. Although the true state trajectories under the distributed LMPC system may seem overlapped with that under the decentralized LMPC system, the closed-loop performance under the distributed LMPC system is inferior than the decentralized LMPC system when subject to sensor attacks on a particular subsystem. To further illustrate the robustness of the decentralized LMPC system compared to the distributed LMPC system, the sum of squared percentage error of the two CSTRs in the form of $SSE_1 = \int_0^{t_{N_T}} ((\frac{C_{A1}-C_{A1_s}}{C_{A1_s}})^2 + (\frac{T_1-T_{1_s}}{T_{1_s}})^2)dt$ and $SSE_2 = \int_0^{t_{N_T}} ((\frac{C_{A2}-C_{A2_s}}{C_{A2_s}})^2 + (\frac{T_2-T_{2_s}}{T_{2_s}})^2)dt$ over the simulation period of $t_{N_T} = 0.3\ hr$ are calculated for the decentralized and distributed LMPC systems when the temperature sensor T_2 is under the three attack types and when under no attacks. The *SSE* results are shown in Table 4.2. We can see that the SSE of the decentralized LMPC 1 stays constant regardless of the presence of an attack; when either min-max, surge, or geometric attacks are injected into the second CSTR, while the decentralized LMPC 2 does yield higher SSE values as a result of the sensor attack, the decentralized LMPC 1 yields SSE values identical to that of the "not attacked" scenario. However, the SSE values of the distributed LMPC 1 when under the three types of attacks are inconsistent and higher than that under no attacks.

The decentralized LMPC system outperforms the centralized and the distributed LMPC systems because the objective function used in each decentralized LMPC only depends on local state feedback measurements of its own subsystem, and is not impacted by state measurements of other subsystems. Therefore, if a particular subsystem is not attacked, the decentralized LMPC for that subsystem will ensure and sustain its stability. On the other hand, due to the centralized LMPC and the distributed LMPC system's requirements of using full-state measurements, the destabilizing impact of sensor cyber-attacks on one particular subsystem will propagate to other subsystems as well. In both centralized LMPC and distributed LMPC structures, the objective function in the LMPC optimization problem accounts for variations in the feedback measurements of all states in order to bring all states to the operating steady-state. Therefore, if one sensor of the overall process is attacked and reflects false information, then the control actions calculated by the centralized LMPC or the distributed LMPC system will deviate from the supposedly optimal value, and cause destabilizing behavior in the true process states of all subsystems.

4.5.2 FNN Detector Modeling

Training data are collected under closed-loop operation for a simulation period of $t_{N_T} = 0.3\ hr$ with $N_T = 30$ and $\Delta = 0.01\ hr$. Cyber-attacks are added at random times during the simulation period

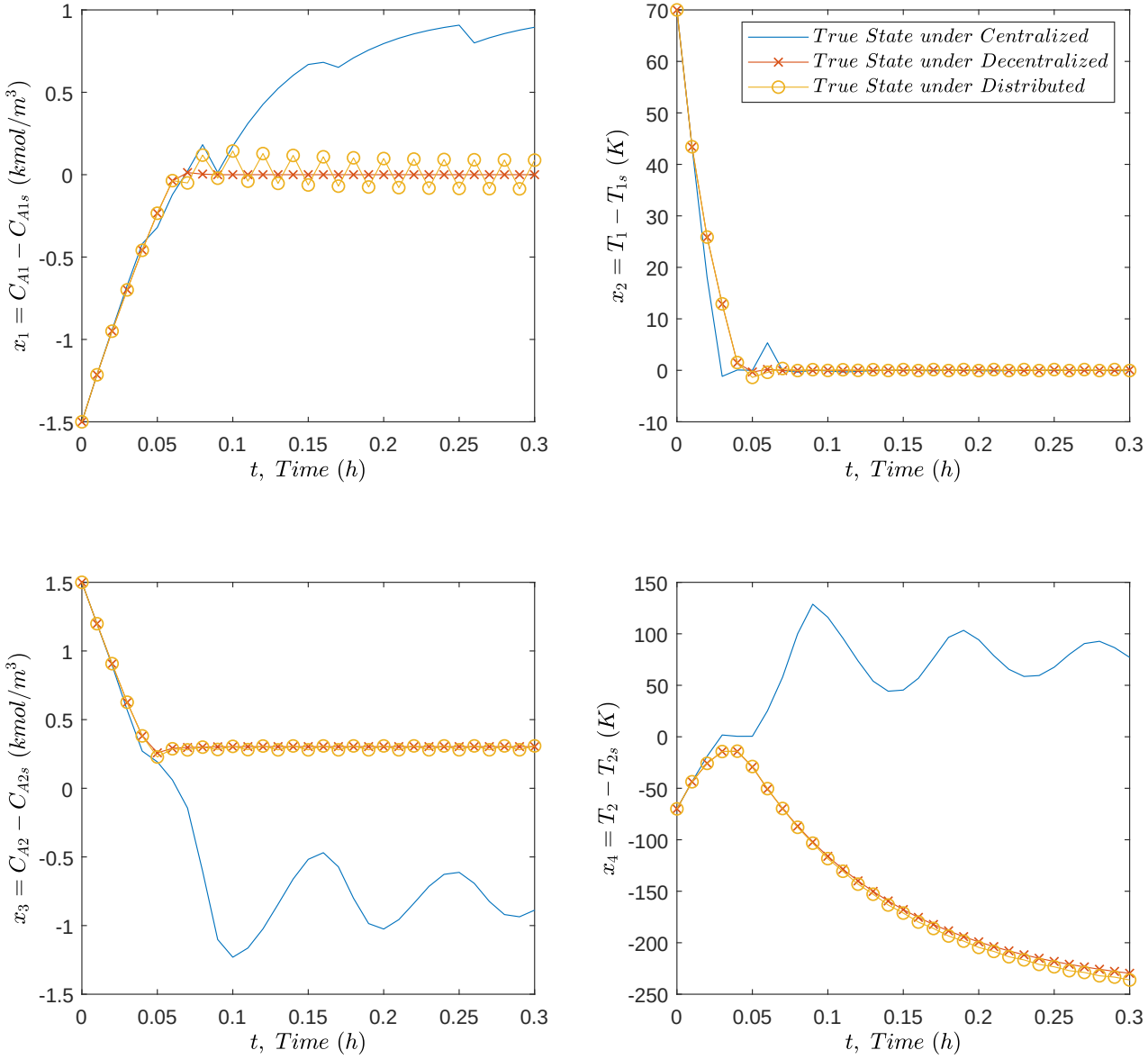


Figure 4.3: Closed-loop trajectories of true states the two-CSTR process operated under centralized, decentralized, and distributed LMPC systems when min-max attacks are added on the temperature sensor T_2 of the second CSTR at $t = 0.05\ hr$.

and last until the end of the period. The state measurements $\bar{x}_j, j = 1,2$ of both CSTRs at every sampling period are recorded, then the Lyapunov function of each subsystem $V_j(\bar{x}_j), j = 1,2$ is calculated along the time-varying trajectory of length $N_T = 30$ to be concatenated and used as the input vector for the FNN model. Such state measurements were collected for un-attacked scenarios as well as scenarios in which either the temperature sensor of the first CSTR T_1, or the temperature

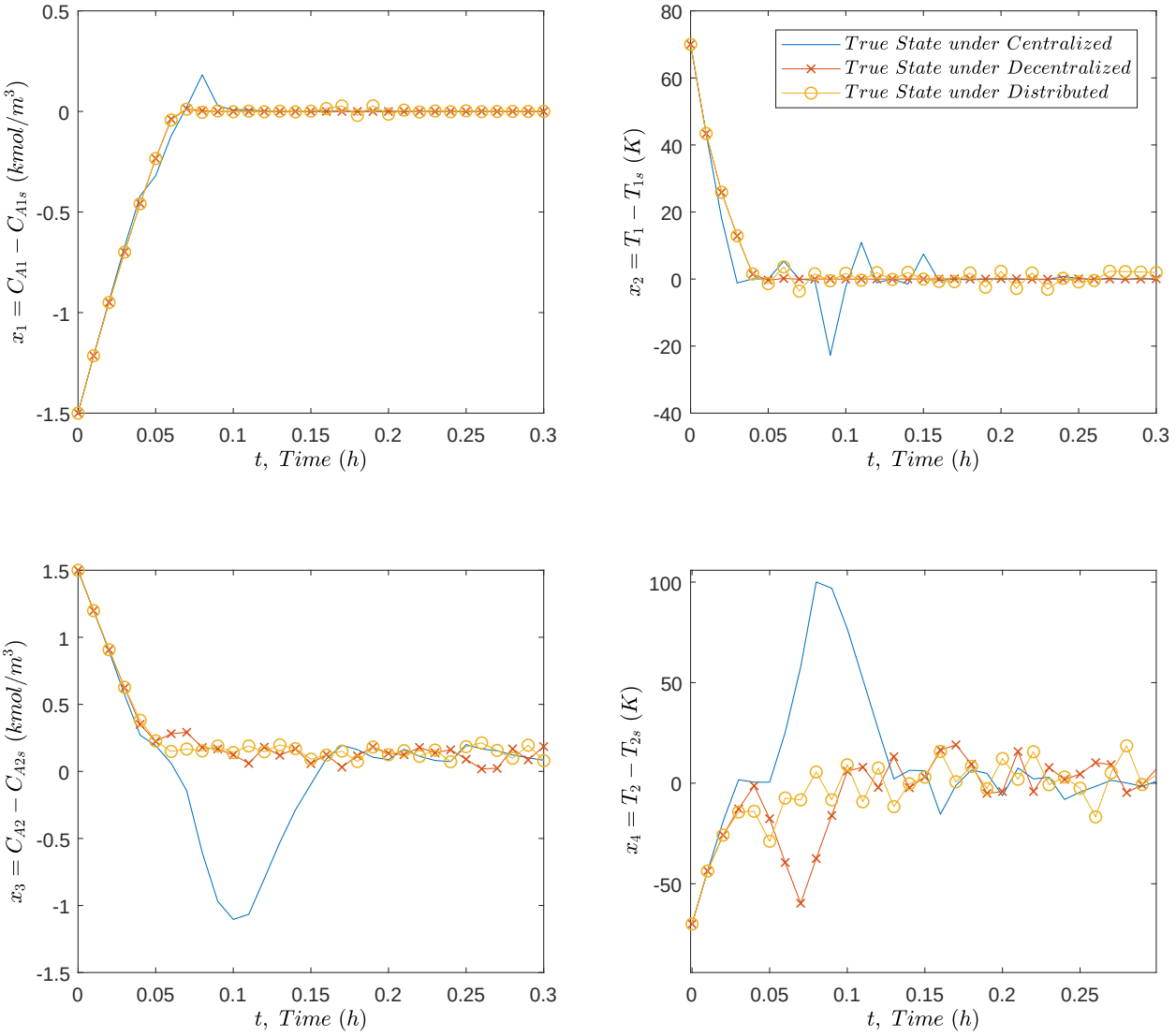


Figure 4.4: Closed-loop trajectories of true states the two-CSTR process operated under centralized, decentralized, and distributed LMPC systems when surge attacks are added on the temperature sensor T_2 of the second CSTR at $t = 0.05\ hr$.

sensor of the second CSTR T_2, has been targeted by min-max attacks. Within each scenario, the training data samples are obtained by simulating the closed-loop process under the decentralized LMPC system starting from different set of initial conditions in the stability region. For training data collection, we only simulate min-max sensor attacks; after obtaining a model with adequate classification accuracy, we test this detector model against surge and geometric sensor attacks to examine the accuracy of the detector against unknown types of sensor cyber-attacks. Depending on whether the detector is designed to identify the presence of a sensor cyber-attack where 2 classes

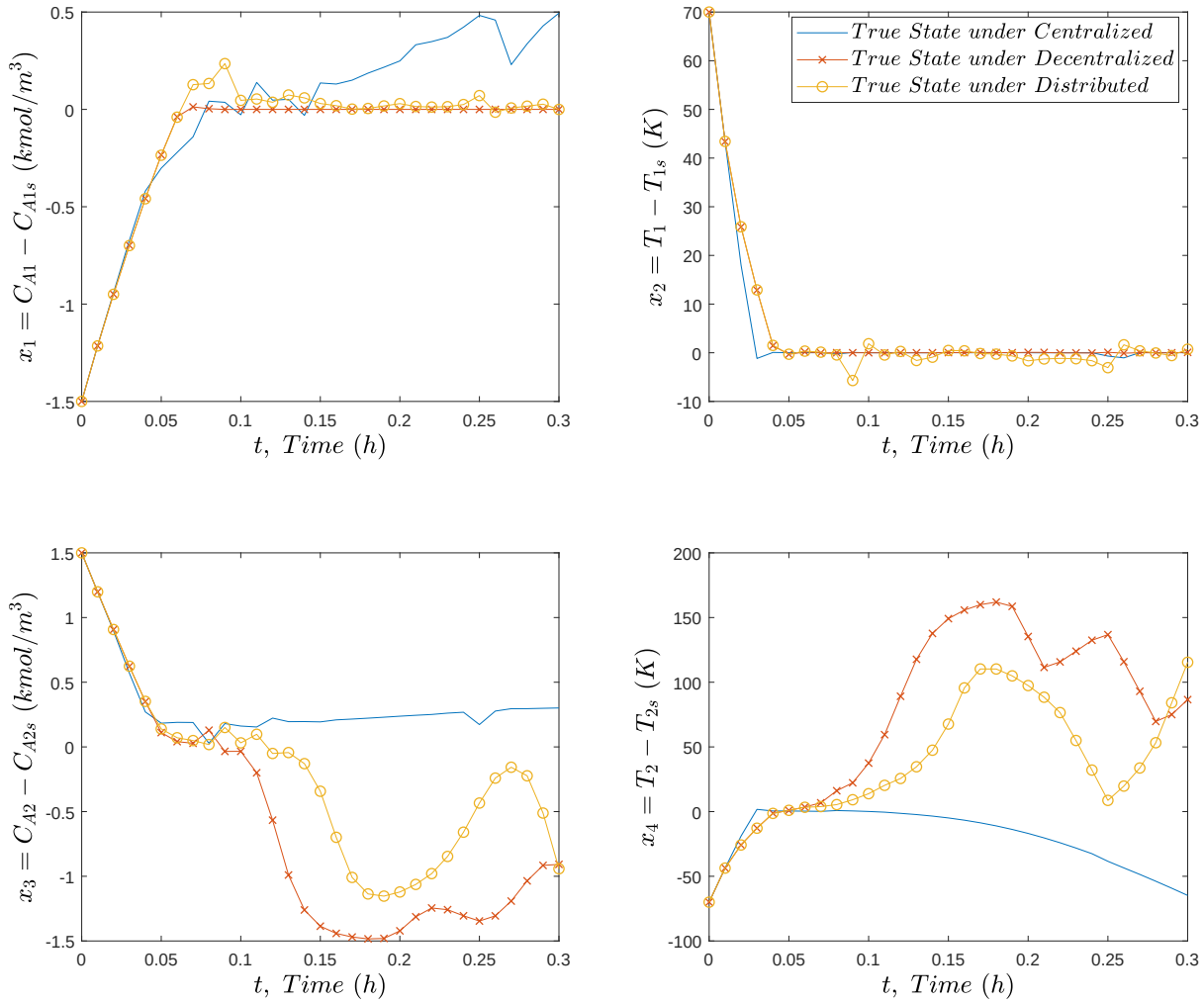


Figure 4.5: Closed-loop trajectories of true states the two-CSTR process operated under centralized, decentralized, and distributed LMPC systems when geometric attacks are added on the temperature sensor T_2 of the second CSTR at $t = 0.05\ hr$.

are involved (attacked vs. not attacked), or to identify which subsystem the sensor cyber-attack has occurred where 3 classes are involved in this example (CSTR-1 attacked vs. CSTR-2 attacked vs. not attacked), the collected training data are labeled differently. Both the 2-class detector and the 3-class detector are FNN models consisting of one hidden layer with 10 neurons, which uses a *tanh* activation function in the form of $g_1(z) = \frac{2}{1+e^{-2z}} - 1$. Then, on the output layer, a *sigmoid* activation function in the form of $g_2(z) = \frac{1}{1+e^{-z}}$ is used to provide an output value between 0 and 1, which represents the predicted probability of the class labels. The number of layers and the number of neurons in each layer is determined via a grid search method. There are some

Table 4.2: Sum of squared percentage error of the first and the second CSTR controlled by decentralized and distributed LMPC systems when under no attacks and when the temperature sensor T_2 is attacked by min-max, surge, geometric cyber-attacks.

	Decentralized SSE-1	Decentralized SSE-2	Distributed SSE-1	Distributed SSE-2
Not Attacked	1.4560	1.3815	1.4578	1.3956
Min-max	1.4560	6.9806	1.5081	7.1882
Surge	1.4560	1.5631	1.4586	1.5540
Geometric	1.4560	11.0521	1.4878	4.7149

standard nonlinear activation functions used in feedforward neural networks and available in the *Keras* library; in our simulation, we chose to use *tanh* on the hidden layer, and *sigmoid* on the output layer, as this combination produced the best prediction outcome and model accuracy. A total of 1500 samples are collected for each class label (i.e., attack scenario), where 70% is used for training and 30% is used for testing. The optimizer function used when minimizing the cost function of the network is *Adam* and 50 epochs are carried out during training for both the 2-class and the 3-class FNN detector models. The testing accuracies against the three different attack types (min-max, surge, geometric) on either the temperature sensor of the first CSTR (i.e., $\bar{x}_2$) or the temperature sensor of the second CSTR (i.e., $\bar{x}_4$) are shown in Table 4.3. Regardless of where the attack happens, the 2-class FNN detector is able to identify the presence of a cyber sensor attack accurately. Both detectors are able to achieve a testing accuracy of above 96% against all attack scenarios. Moreover, the classification accuracy for the "not attacked" scenario is 100% for both detectors, showing that the occurrence of false alarms is very low.

4.5.3 Closed-loop Operation with FNN Detector

After obtaining the FNN detector models with an adequate classification accuracy, we apply the detector online. At each sampling instant after the state measurements are collected from the sensors and before these measurements are communicated to the controllers, they are sent to the detector. A moving horizon detection window of size $N_T \times \Delta$ is implemented on the detector, where the detector examines the Lyapunov functions of the latest measured state trajectories of length N_T for both subsystems $j = 1,2$ as the input vector for the FNN model (i.e., $V_j(\bar{x}_j(t_k))$ for $j = 1,2$ and $k = 1,...,N_T$). Therefore, as the new state measurements are received by the detector, the Lyapunov functions for these latest measurements are calculated and added to the input vector,

Table 4.3: Detection accuracies of NN detectors in response to min-max, geometric, and surge attacks.

	Detector 1 (2-class) (Attacked vs. Not Attacked)	Detector 2 (3-class) (T_1 Attacked vs. T_2 Attacked vs. Not Attacked)
Min-max on T_1	99.93%	99.87%
Min-max on T_2	99.94%	99.88%
Surge on T_1	99.29%	98.57%
Surge on T_2	98.22%	100.00%
Geometric on T_1	96.88%	97.73%
Geometric on T_2	99.47%	98.93%
Not Attacked	100.00%	100.00%

and the detection window moves forward one sampling step Δ. If the prediction provided by the detector indicates that an attack has occurred within the detection window $N_T \times \Delta$, then all sensors in the process will be deemed un-trustworthy and should be switched to their back-up sensors. In the case that a 3-class detector is used and is able to identify which subsystem is experiencing sensor cyber-attacks, then a possible mitigation response is to only replace the sensors of the targeted subsystem. However, knowing that cyber-attacks has entered the system and leaving the overall network vulnerable, it is still good practice to examine all the sensors and act accordingly. We simulate closed-loop operation of the two-CSTR process under the more robust decentralized LMPC system integrated with the 2-class FNN detector. The total simulation period is 0.6 *hr* with a sampling period of $\Delta = 0.01$ *hr*, where either the temperature sensor for T_1 or T_2 starts experiencing one of the three types of cyber-attacks at $t = 0.3$ *hr*. The measured and the true states of both CSTRs in state-space under different attack scenarios are shown in Figs. 4.6 – 4.8. After the detector indicates the occurrence of a cyber-attack, all sensors of the system are deemed unreliable and are switched to redundant back-up sensors to ensure the security of measurement data. The detector experiences a slight time delay when detecting the presence of certain sensor attacks. More specifically, a detection time delay of 2 sampling periods are observed when min-max and surge attacks are added on the sensor for T_1, and a detection time delay of 5 sampling periods for geometric attacks on T_1. When attacks target T_2, the detection time delays for min-max, surge, and geometric attacks are 0, 0, and 4 sampling periods, respectively. This means that as soon as the sensor for T_2 is attacked by min-max or surge attacks, and the attacked measurements are sent to

the detector, the detector flags it immediately and the sensors are switched to their secure back-up sensors before the controller receives these incorrect measurements. Despite the time delays in some cases, all true process states are maintained inside the stability region and eventually driven back to the terminal set around the operating steady-state within 6 sampling periods (i.e., 0.06 hr).

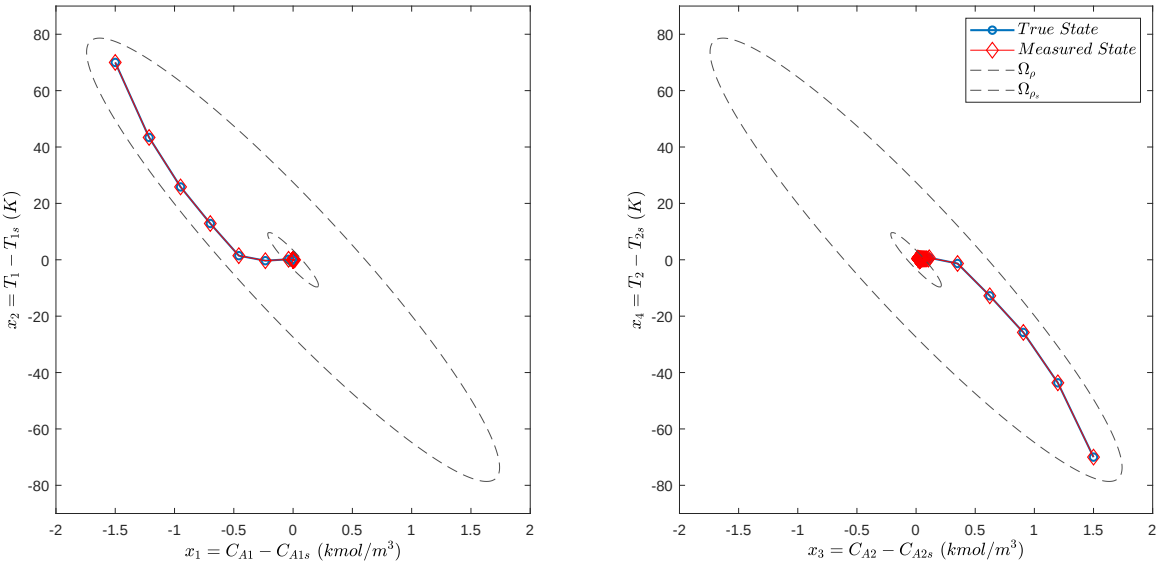


Figure 4.6: Closed-loop trajectories of true states the two-CSTR process operated under the decentralized LMPC system when min-max attacks are added on the temperature sensor T_2 of the second CSTR at $t = 0.30$ hr, and detected by the 2-class FNN detector at $t = 0.30$ hr.

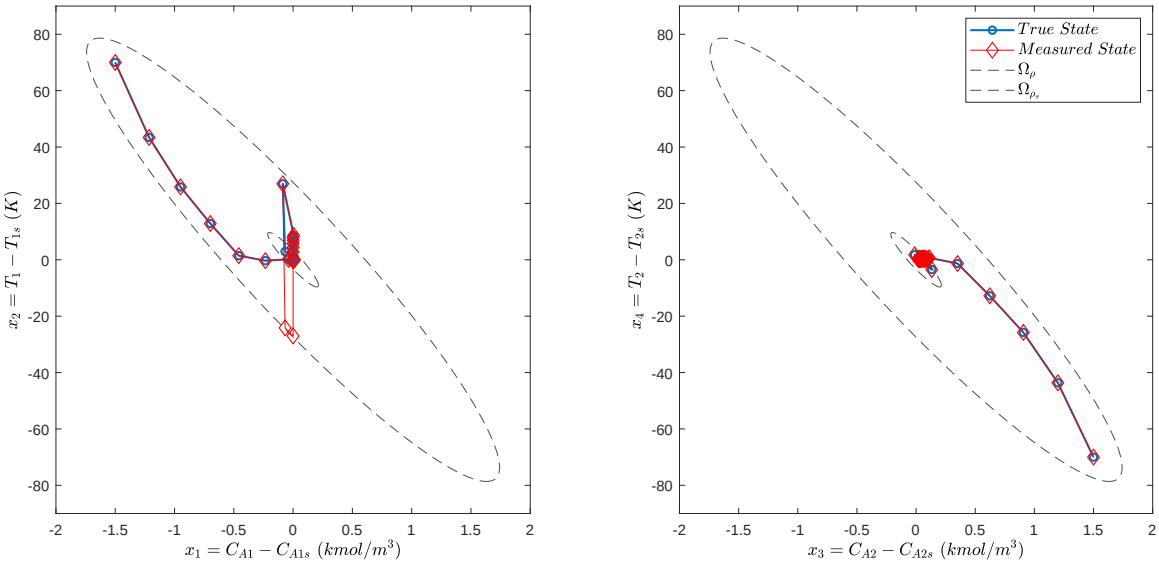


Figure 4.7: Closed-loop trajectories of true states the two-CSTR process operated under the decentralized LMPC system when surge attacks are added on the temperature sensor T_1 of the first CSTR at $t = 0.30$ hr and detected by the 2-class FNN detector at $t = 0.32$ hr, after which all sensors are switched to their secure back-up sensors and the true process states are driven back to the ultimate bounded region Ω_{ρ_s} around the operating steady-state.

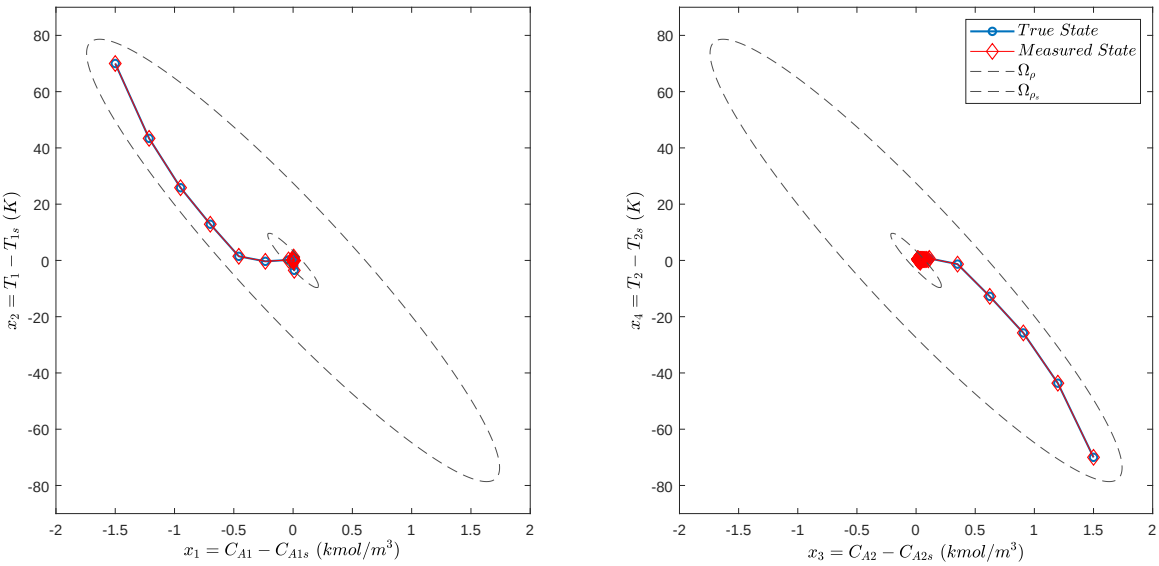


Figure 4.8: Closed-loop trajectories of true states the two-CSTR process operated under the decentralized LMPC system when geometric attacks are added on the temperature sensor T_1 of the first CSTR at $t = 0.30$ hr and detected by the 2-class FNN detector at $t = 0.35$ hr, after which all sensors are switched to their secure back-up sensors and the true process states are maintained within the ultimate bounded region Ω_{ρ_s} around the operating steady-state.

Chapter 5.1

Machine Learning-Based Distributed Model Predictive Control of Nonlinear Processes

This chapter explores the design of distributed model predictive control (DMPC) systems for nonlinear processes using machine learning models to predict nonlinear dynamic behavior. Specifically, sequential and iterative distributed model predictive control systems are designed and analyzed with respect to closed-loop stability and performance properties. Extensive open-loop data within a desired operating region are used to develop Long Short-Term Memory (LSTM) recurrent neural network models with a sufficiently small modeling error from the actual nonlinear process model. Subsequently, these LSTM models are utilized in Lyapunov-based DMPC to achieve efficient real-time computation time while ensuring closed-loop state boundedness and convergence to the origin. Using a nonlinear chemical process network example, the simulation results demonstrate the improved computational efficiency when the process is operated under sequential and iterative DMPCs while the closed-loop performance is very close to the one of a centralized MPC system.

With the rise of big data analytics, machine learning methodologies have gained increasing recognition and demonstrated successful implementation in many traditional engineering fields. One exemplar use of machine learning techniques in chemical engineering is the identification of process models using recurrent neural networks (RNN), which has shown effectiveness in modeling nonlinear dynamic systems. RNN is a class of artificial neural networks that, by using feedback loops in its neurons and passing on past information derived from earlier inputs to the current network, can represent temporal dynamic behaviors.

On the other hand, chemical process operation has extensively relied on automated control

systems, and the need of accounting for multivariable interactions and input/state constraints has motivated the development of model predictive control (MPC). Moreover, augmentation in sensor information and network-based communication increases the number of decision variables, state variables, and measurement data, which in turn increases the complexity of the control problem and the computation time if a centralized MPC is used. With these considerations in mind, distributed control systems have been developed, where multiple controllers with inter-controller communication are used to cooperatively calculate the control actions and achieve closed-loop plant objectives. In this context, MPC is a natural control framework to implement due to its ability to account for input and state constraints while also considering the actions of other control actuators. In other words, the controllers communicate with each other to calculate their distinct set of manipulated inputs that will collectively achieve the control objectives of the closed-loop system. Many distributed MPC methods have been proposed in the literature addressing the coordination of multiple MPCs that communicate to calculate the optimal input trajectories in a distributed manner (the reader may refer to [27, 92, 101] for reviews of results on distributed MPC, and to [31] for a review of network structure-based decomposition of control and optimization problems). A robust distributed control approach to plant-wide operations based on dissipativity was proposed in [114, 131]. Depending on the communication network, i.e., whether is one-directional or bi-directional, two distributed architectures, namely sequential and iterative distributed MPCs, were proposed in [70]. Furthermore, distributed MPC method was also used in [71] to address the problem of introducing new control systems to pre-existing control schemes. In a recent work [105], a fast and stable non-convex constrained distributed optimization algorithm was developed and applied to distributed MPC. As distributed MPC systems also depend on an accurate process model, the development and implementation of RNN models in distributed MPCs is an important area yet to be explored. In the present work, we introduce distributed control frameworks that employ a Long Short-Term-Memory (LSTM) network, which is a particular type of RNN. The distributed control systems are designed via Lyapunov-based model predictive control (LMPC) theory. Specifically, we explore both sequential distributed LMPC systems and iterative distributed LMPC systems, and compare the closed-loop performances with that of a centralized LMPC system.

The remainder of the chapter is organized as follows. Preliminaries on notation, the general class of nonlinear systems, and the stabilizing Lypuanov-based controller for the nonlinear process are given in Section 2. The structure and development of recurrent neural network and specifically LSTM, as well as Lyapunov-based control using LSTM models are outlined in Section 3. In

Section 4, the formulation and proof for recursive feasibility and closed-loop stability of the distributed Lyapunov-based model predictive control systems using an LSTM model as the prediction model are presented. Lastly, Section 5 includes the application to a two-CSTR-in-series process, demonstrating guaranteed closed-loop stability and enhanced computational efficiency of the proposed distributed LMPC systems with respect to the centralized LMPC.

5.1.1 Preliminaries

5.1.1.1 Notation

For the remainder of this manuscript, the notation x^T is used to denote the transpose of x. $|\cdot|$ is used to denote the Euclidean norm of a vector. $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. $\emptyset$ signifies the null set. The function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable in its domain. A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathscr{K}$ if it is strictly increasing and is zero only when evaluated at zero.

5.1.1.2 Class of Systems

In this work, we consider a general class of nonlinear systems in which several distinct sets of manipulated inputs are used, and each distinct set of manipulated inputs is responsible for regulating a specific subsystem of the process. For the simplicity of notation, throughout the manuscript, we consider two subsystems, subsystem 1 and subsystem 2, consisting of states x_1 and x_2 respectively, which are regulated by only u_1 and u_2 respectively. However, extending the analysis to systems with more than two sets of distinct manipulated input vectors (i.e., having more than two subsystems, with each one regulated by one distinct input vector $u_j,\ j = 1, ..., M,\ M > 2$) is conceptually straight-forward. The class of continuous-time nonlinear systems considered is represented by the following system of first-order nonlinear ordinary differential equations:

$$\dot{x} = F(x, u_1, u_2, w) := f(x) + g_1(x)u_1 + g_2(x)u_2 + v(x)w,\ x(t_0) = x_0 \tag{5.1.1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u_1 \in \mathbf{R}^{m_1}$ and $u_2 \in \mathbf{R}^{m_2}$ are two separate sets of manipulated input vectors, and $w \in W$ is the disturbance vector with $W := \{w \in \mathbf{R}^r \mid |w| \leq w_m,\ w_m \geq 0\}$. The control action constraints are defined by $u_1 \in U_1 := \{u_{1_i}^{\min} \leq u_{1_i} \leq u_{1_i}^{\max}, i = 1, ..., m_1\} \subset \mathbf{R}^{m_1}$, and

$u_2 \in U_2 := \{u_{2_i}^{\min} \leq u_{2_i} \leq u_{2_i}^{\max}, i = 1,...,m_2\} \subset \mathbf{R}^{m_2}$. $f(\cdot)$, $g_1(\cdot)$, $g_2(\cdot)$, and $v(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, $n \times m_1$, $n \times m_2$, and $n \times r$, respectively. Throughout the manuscript, the initial time t_0 is taken to be zero ($t_0 = 0$), and it is assumed that $f(0) = 0$, and thus, the origin is a steady-state of the nominal (i.e., $w(t) \equiv 0$) system of Eq. 8.1 (i.e., $(x_s, u_{1s}, u_{2s}) = (0,0,0)$, where x_s, u_{1s} and u_{2s} represent the steady-state state and input vectors).

5.1.1.3 Stability Assumptions

We assume that there exist stabilizing control laws $u_1 = \Phi_1(x) \in U_1, u_2 = \Phi_2(x) \in U_2$ (e.g., the universal Sontag control law [67]) such that the origin of the nominal system of Eq. 8.1 with $w(t) \equiv 0$ is rendered exponentially stable in the sense that there exists a $\mathscr{C}^1$ Control Lyapunov function $V(x)$ such that the following inequalities hold for all x in an open neighborhood D around the origin:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{5.1.2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi_1(x), \Phi_2(x), 0) \leq -c_3|x|^2, \tag{5.1.2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{5.1.2c}$$

where c_1, c_2, c_3 and c_4 are positive constants. $F(x, u_1, u_2, w)$ represents the nonlinear system of Eq. 8.1. A set of candidate controllers $\Phi_1(x) \in \mathbf{R}^{m_1}$ and $\Phi_2(x) \in \mathbf{R}^{m_2}$, both denoted by $\Phi_k(x)$ where $k = 1,2$, is given in the following form:

$$\phi_{k_i}(x) = \begin{cases} -\dfrac{p + \sqrt{p^2 + q^4}}{q^T q} q & \text{if} \quad q \neq 0 \\ 0 & \text{if} \quad q = 0 \end{cases} \tag{5.1.3a}$$

$$\Phi_{k_i}(x) = \begin{cases} u_{k_i}^{min} & \text{if} \quad \phi_{k_i}(x) < u_{k_i}^{min} \\ \phi_{k_i}(x) & \text{if} \quad u_{k_i}^{min} \leq \phi_{k_i}(x) \leq u_{k_i}^{max} \\ u_{k_i}^{max} & \text{if} \quad \phi_{k_i}(x) > u_{k_i}^{max} \end{cases} \tag{5.1.3b}$$

where $k = 1,2$ represents the two candidate controllers, p denotes $L_f V(x)$ and q denotes $L_{g_{k_i}} V(x)$, $f = [f_1 \cdots f_n]^T$, $g_{k_i} = [g_{k_{i1}}, ..., g_{k_{in}}]^T$, ($i = 1,2,...,m_1$ for $k = 1$ corresponding to the vector of control actions $\Phi_1(x)$, and $i = 1,2,...,m_2$ for $k = 2$ corresponding to the vector of control actions $\Phi_2(x)$.) $\phi_{k_i}(x)$ of Eq. 7.7a represents the i_{th} component of the control law $\phi_k(x)$. $\Phi_{k_i}(x)$ of Eq. 7.7 represents the i_{th} component of the saturated control law $\Phi_k(x)$ that accounts for the input constraints $u_k \in U_k$.

Based on Eq. 7.5, we can first characterize a region where the time-derivative of V is rendered negative under the controller $\Phi_1(x) \in U_1,\ \Phi_2(x) \in U_2$ as $D = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V + L_{g_1} V u_1 + L_{g_2} V u_2 < -c_3|x|^2, u_1 = \Phi_1(x) \in U_1, u_2 = \Phi_2(x) \in U_2\} \cup \{0\}$. Then the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 8.1 is defined as a level set of the Lyapunov function, which is inside D: $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset D$. Also, the Lipschitz property of $F(x, u_1, u_2, w)$ combined with the bounds on u_1, u_2 and w implies that there exist positive constants $M, L_x, L_w, L_x^{'}, L_w^{'}$ such that the following inequalities hold $\forall x, x' \in \Omega_\rho, u_1 \in U_1, u_2 \in U_2$ and $w \in W$:

$$|F(x, u_1, u_2, w)| \leq M \tag{5.1.4a}$$

$$|F(x, u_1, u_2, w) - F(x', u_1, u_2, 0)| \leq L_x|x - x'| + L_w|w| \tag{5.1.4b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x, u_1, u_2, w) - \frac{\partial V(x')}{\partial x} F(x', u_1, u_2, 0) \right| \leq L_x^{'}|x - x'| + L_w^{'}|w| \tag{5.1.4c}$$

5.1.2 Long Short-Term Memory Network

In this work, we develop an LSTM network model with the following form:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u_1, u_2) := A\hat{x} + \Theta^T y \tag{5.1.5}$$

where $\hat{x} \in \mathbf{R}^n$ is the predicted state vector and $u_1 \in \mathbf{R}^{m_1}$ and $u_2 \in \mathbf{R}^{m_2}$ are the two separate sets of manipulated input vectors. $y^T = [y_1, ..., y_n, y_{n+1}, ..., y_{n+m_1}, y_{n+m_1+1} ..., y_{n+m_1+m_2}, y_{n+m_1+m_2+1}] = [H(\hat{x}_1), ..., H(\hat{x}_n), u_{1_1}, ..., u_{1_{m_1}}, u_{2_1}, ..., u_{2_{m_2}}, 1] \in \mathbf{R}^{n+m_1+m_2+1}$ is a vector of the network state $\hat{x}$, where $H(\cdot)$ represents a series of interacting nonlinear activation functions in each LSTM unit, the inputs u_1 and u_2, and the constant 1 which accounts for the bias term. A is a diagonal coefficient matrix, i.e., $A = diag\{-\alpha_1, ..., -\alpha_n\} \in \mathbf{R}^{n \times n}$, and $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(n+m_1+m_2+1) \times n}$ with $\theta_i = \beta_i[\omega_{i1}, ..., \omega_{i(n+m_1+m_2)}, b_i]$, $i = 1, ..., n$. α_i and β_i are constants, and ω_{ij} is the weight connecting the jth input to the ith neuron where $i = 1, ..., n$ and $j = 1, ..., (n + m_1 + m_2)$, and b_i is the bias term for $i = 1, ..., n$. We use x to represent the state of actual nonlinear system of Eq. 8.1 and use $\hat{x}$ for the state of the LSTM model of Eq. 6.3. Here, α_i is assumed to be positive such that each state $\hat{x}_i$ is bounded-input bounded-state stable.

Instead of having one-way information flow from the input layer to the output layer in a feed-forward neural network (FNN), RNNs introduce feedback loops into the network and allow information exchange in both directions between modules. Unlike feed-forward neural networks,

RNNs take advantage of the feedback signals to store outputs derived from past inputs, and together with the current input information, give a more accurate prediction of the current output. By having access to information of the past, RNN is capable of representing dynamic behaviors of time-series samples, therefore it is an effective method used to model nonlinear processes. Based on the universal approximation theorem, it can be shown that the RNN model with sufficient number of neurons is able to approximate any nonlinear dynamic system on compact subsets of the state-space for finite time [64, 99]. However, in a standard RNN model, the problem of vanishing gradient phenomena often arises due to the network's difficulty to capture long term dependencies; this is because of multiplicative gradients that can be exponentially decaying with respect to the number of layers. Therefore, the stored information over extended time intervals is very limited in a short term memory manner. Due to these considerations, Hochreiter and Schmidhuber [50] proposed the Long Short-Term Memory (LSTM) network, which is a type of RNN that uses three gated units (the forget gate, the input gate, and the output gate) to protect and control the memory cell state, $c(k)$, where $k = 1, ..., T$, such that information will be stored and remembered for long periods of time [50]. For these reasons, LSTM networks may perform better when modeling processes where inputs towards the beginning of a long time-series sequence are crucial to the prediction of outputs towards the end of the sequence. This may be more prevalent in large-scale systems where there may exist inherent time delays between subsystems, causing discrepancies in the speed of the dynamics between the subsystems. The basic architecture of an LSTM network is illustrated in Fig. 5.2.1. We develop an LSTM network model to approximate the class of continuous-time nonlinear processes of Eq. 8.1.

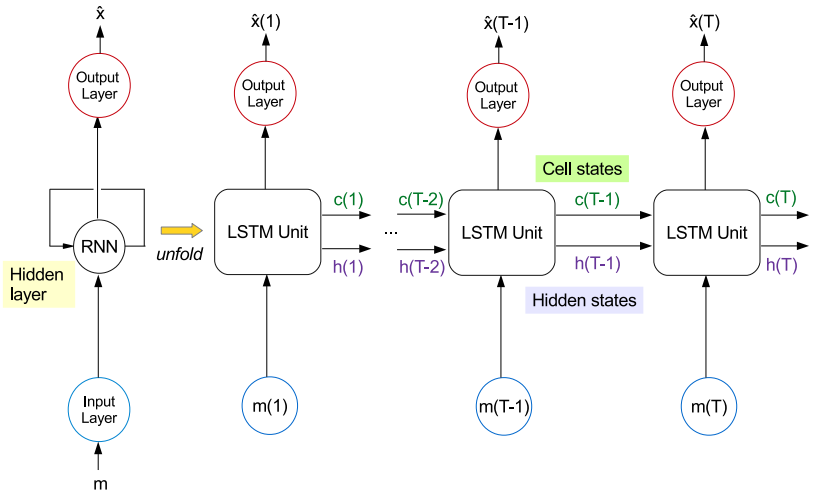


Figure 5.1.1: A long short-term memory recurrent neural network and its unfolded structure, where m is the input vector and $\hat{x}$ is the output vector, c is the cell state vector, and h is the hidden state vector.

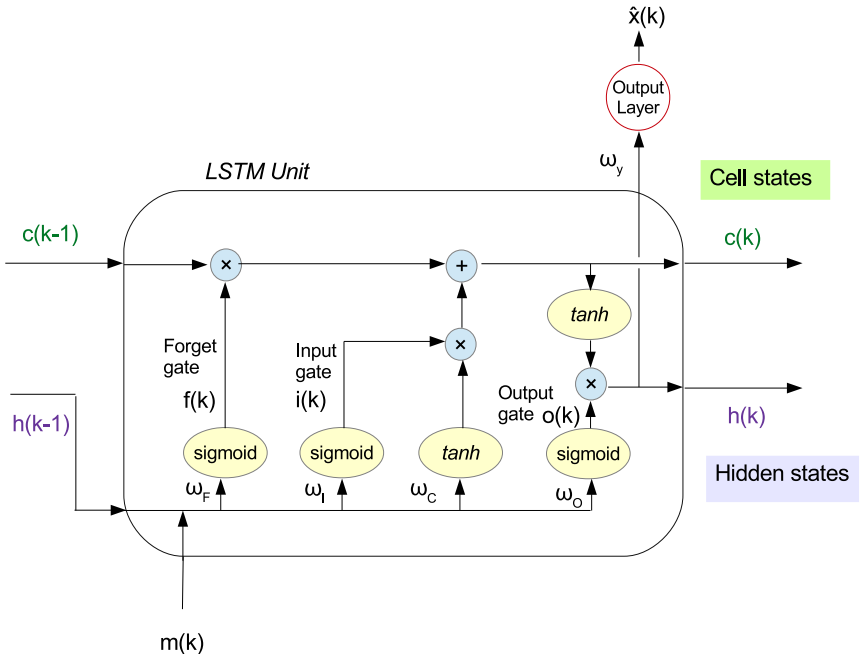


Figure 5.1.2: The internal structure of an LSTM unit showing the input gate, the forget gate, and the output gate layers, where the cell state vector $c(k-1)$, hidden state vector $h(k-1)$, and the input vector $m(k)$ are used to obtain $c(k)$, $h(k)$, as well as the network output vector $y(k)$ via an additional output activation layer.

We use $m \in \mathbf{R}^{(n+m_1+m_2)\times T}$ to denote the matrix of input sequences to the LSTM network, and $\hat{x} \in \mathbf{R}^{n\times T}$ to denote the matrix of network output sequences. The output from each repeating module that is passed onto the next repeating module in the unfolded sequence is the hidden state, and the vector of hidden states is denoted by h. The network output $\hat{x}$ at the end of the prediction period is dependent on all internal states $h(1),...,h(T)$, where the number of internal states T (i.e., the number of repeating modules) corresponds to the length of the time-series input sample. The LSTM network calculates a mapping from the input sequence m to the output sequence $\hat{x}$ by calculating the following equations iteratively from $k = 1$ to $k = T$:

$$i(k) = \sigma(\omega_i^m m(k) + \omega_i^h h(k-1) + b_i) \tag{5.1.6a}$$

$$f(k) = \sigma(\omega_f^m m(k) + \omega_f^h h(k-1) + b_f) \tag{5.1.6b}$$

$$c(k) = f(k)c(k-1) + i(k)tanh(\omega_c^m m(k) + \omega_c^h h(k-1) + b_c) \tag{5.1.6c}$$

$$o(k) = \sigma(\omega_o^m m(k) + \omega_o^h h(k-1) + b_o) \tag{5.1.6d}$$

$$h(k) = o(k)tanh(c(k)) \tag{5.1.6e}$$

$$\hat{x}(k) = \omega_y h(k) + b_y \tag{5.1.6f}$$

where $\sigma(\cdot)$ is the sigmoid function, $tanh(\cdot)$ is the hyperbolic tangent function; both of which are activation functions. $h(k)$ is the internal state, and $\hat{x}(k)$ is the output from the repeating LSTM module with ω_y and b_y denoting the weight matrix and bias vector for the output, respectively. The outputs from the input gate, the forget gate, and the output gate are represented by $i(k)$, $f(k)$, $o(k)$, respectively; correspondingly, ω_i^m, ω_i^h, ω_f^m, ω_f^h, ω_o^m, ω_o^h are the weight matrices for the input vector m and the hidden state vectors h within the input gate, the forget gate, and the output gate respectively, and b_i, b_f, b_o represent the bias vectors within each of the three gates, respectively. Furthermore, $c(k)$ is the cell state which stores information to be passed down the network units, with ω_c^m, ω_c^h and b_c representing the weight matrices for the input and hidden state vectors, and the bias vector in the cell state activation function, respectively. The series of interacting nonlinear functions carried out in each LSTM unit, outlined in Eq. 5.2.8, can be represented by $H(\hat{x})$. The internal structure of a repeating module within an LSTM network where the iterative calculations of Eq. 5.2.8 are carried out is shown in Fig. 5.2.2.

The closed-loop simulation of the continuous-time nonlinear system of Eq. 8.1 is carried out in a sample-and-hold manner, where the feedback measurement of the closed-loop state x is received by the controller every sampling period Δ. Furthermore, state information of the simulated nonlinear process is obtained via numerical integration methods, e.g., explicit Euler, using an

integration time step of h_c. Since the objective of developing the LSTM model is its eventual utilization in a controller, the prediction period of the LSTM model is set to be the same as the sampling period Δ of the model predictive controller. The time interval between two consecutive internal states within the LSTM can be chosen to be a multiple q_{nn} of the integration time step h_c used in numerical integration of the nonlinear process, with the minimum time interval being $q_{nn} = 1$, i.e., $1 \times h_c$. Therefore, depending on the choice of q_{nn}, the number of internal states, T, will follow $T = \frac{\Delta}{q_{nn} \cdot h_c}$. Given that the input sequences fed to the LSTM network are taken at time $t = t_k$, the future states predicted by the LSTM network, $\hat{x}(t)$, at $t = t_k + \Delta$, would be the network output vector at $k = T$, i.e., $\hat{x}(t_k + \Delta) = \hat{x}(T)$. The LSTM learning algorithm is developed to obtain the optimal parameter matrix Γ^*, which includes the network parameters ω_i, ω_f, ω_c, ω_o, ω_y, b_i, b_f, b_c, b_o, b_y. Under this optimal parameter matrix, the error between the actual state $x(t)$ of the nominal system of Eq. 8.1 (i.e., $w(t) \equiv 0$) and the modeled states $\hat{x}(t)$ of the LSTM model of Eq. 6.3 is minimized. The LSTM model is developed using a state-of-the-art application program interface, i.e., Keras, which contains open-source neural network libraries. The mean absolute percentage error between $x(t)$ and $\hat{x}(t)$ is minimized using the adaptive moment estimation optimizer, i.e., *Adam* in Keras, in which the gradient of the error cost function is evaluated using back-propagation. Furthermore, in order to ensure that the trained LSTM model can sufficiently represent the nonlinear process of Eq. 8.1, which in turn ascertains that the LSTM model can be used in a model-based controller to stabilize the actual nonlinear process at its steady-state with guaranteed stability properties, a constraint on the modeling error is also imposed during training, where $|\nu| = |F(x,u_1,u_2,0) - F_{nn}(x,u_1,u_2)| \leq \gamma|x|$, with $\gamma > 0$. Additionally, to avoid over-fitting of the LSTM model, the training process is terminated once the modeling error falls below the desired threshold and the error on the validation set stops decreasing. One way to assess the modeling error $\nu = F(x(t_k),u_1,u_2,0) - F_{nn}(x(t_k),u_1,u_2)$ is through numerical approximation using the forward finite difference method. Given that the time interval between internal states of the LSTM model is a multiple of the integration time step $q_{nn} \times h_c$, the time derivative of the LSTM predicted state $\hat{x}(t)$ at $t = t_k$ can be approximated by $\dot{\hat{x}}(t_k) = F_{nn}(x(t_k),u_1,u_2) \approx \frac{\hat{x}(t_k+q_{nn}h_c)-\hat{x}(t_k)}{q_{nn}h_c}$. The time derivative of the actual state $x(t)$ at $t = t_k$ can be approximated by $\dot{x}(t_k) = F(x(t_k),u_1,u_2,0) \approx \frac{x(t_k+q_{nn}h_c)-x(t_k)}{q_{nn}h_c}$.

At time $t = t_k$, $\hat{x}(t_k) = x(t_k)$, the constraint $|\nu| \leq \gamma|x|$ can be written as follows:

$$|\nu| = |F(x(t_k), u_1, u_2, 0) - F_{nn}(x(t_k), u_1, u_2)| \tag{5.1.7a}$$

$$\approx |\frac{x(t_k + q_{nn}h_c) - \hat{x}(t_k + q_{nn}h_c)}{q_{nn}h_c}| \tag{5.1.7b}$$

$$\leq \gamma|x(t_k)| \tag{5.1.7c}$$

which will be satisfied if $|\frac{x(t_k+q_{nn}h_c)-\hat{x}(t_k+q_{nn}h_c)}{x(t_k)}| \leq \gamma q_{nn}h_c$. Therefore, the mean absolute percentage error between the predicted states $\hat{x}$ and the targeted states x in the training data will be used as a metric to assess the modeling error of the LSTM model. While the error bounds that the LSTM network model and the actual process should satisfy to ensure closed-loop stability are difficult to calculate explicitly and are, in general, conservative, they provide insight into the key network parameters that will need to be tuned to reduce the error between the two models as well as the amount of data needed to build a suitable LSTM model.

In order to gather adequate training data to develop the LSTM model for the nonlinear process, we first discretize the desired operating region in state-space with sufficiently small intervals as well as discretize the range of manipulated inputs based on the control actuator limits. We run open-loop simulations for the nonlinear process of Eq. 8.1 starting from different initial conditions inside the desired operating region, i.e., $x_0 \in \Omega_\rho$, for finite time using combinations of the different manipulated inputs $u_1 \in U_1, u_2 \in U_2$ applied in a sample-and-hold manner, and the evolving state trajectories are recorded at time intervals of size $q_{nn} \times h_c$. We obtain enough samples of such trajectories to sweep over all the values that the states and the manipulated inputs (x, u_1, u_2) could take to capture the dynamics of the process. These time-series data can be separated into samples with a fixed length T, which corresponds to the prediction period of the LSTM model, where $\Delta = T \times q_{nn} \times h_c$. The time interval between two time-series data points in the sample $q_{nn} \times h_c$ corresponds to the time interval between two consecutive memory units in the LSTM network. The generated dataset is then divided into training and validation sets.

Remark 5.1.1. *The actual nonlinear process is a continuous-time model that can be represented using Eq. 8.1; therefore, to characterize the modeling error ν between the LSTM network and the nonlinear process of Eq. 8.1, the LSTM network is represented as a continuous-time model of Eq. 6.3. However, the series of interacting nonlinear operations in the LSTM memory unit is carried out recursively akin to a discrete-time model. The time interval $q_{nn} \times h_c$ between two LSTM memory units is given by the time interval between two consecutive time-series data points*

in the training samples. Since the LSTM network provides a predicted state at each time interval $q_{nn} \times h_c$ calculated by each LSTM memory unit, similarly to how we can use numerical integration methods to obtain the state at the same time instance using the continuous-time model, we can use the predicted states from the LSTM network to compare with the predicted states from the nonlinear model of Eq. 8.1 to assess the modeling error. The modeling error is subject to the constraint of Eq. 5.2.9 to ensure that the LSTM model can be used in the model-based controller with guaranteed stability properties.

5.1.2.1 Lyapunov-based Control using LSTM Models

Once we obtain an LSTM model with a sufficiently small modeling error, we can design a stabilizing feedback controller $u_1 = \Phi_{nn_1}(x) \in U_1$ and $u_2 = \Phi_{nn_2}(x) \in U_2$ that can render the origin of the LSTM model of Eq. 6.3 exponentially stable in an open neighborhood $\hat{D}$ around the origin in the sense that there exists a $\mathscr{C}^1$ Control Lyapunov function $\hat{V}(x)$ such that the following inequalities hold for all x in $\hat{D}$:

$$\hat{c}_1|x|^2 \leq \hat{V}(x) \leq \hat{c}_2|x|^2, \tag{5.1.8a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, \Phi_{nn_1}(x), \Phi_{nn_2}(x)) \leq -\hat{c}_3|x|^2, \tag{5.1.8b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{c}_4|x| \tag{5.1.8c}$$

where $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$, $\hat{c}_4$ are positive constants, and $F_{nn}(x, u_1, u_2)$ represents the LSTM network model of Eq. 6.3. Similar to the characterization method of the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 8.1, we first search the entire state-space to characterize a set of states $\hat{D}$ where the following inequality holds: $\dot{\hat{V}}(x) = \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, u_1, u_2) < -\hat{c}_3|x|^2$, $u_1 = \Phi_{nn_1}(x) \in U_1$, $u_2 = \Phi_{nn_2}(x) \in U_2$. The closed-loop stability region for the LSTM network model of Eq. 6.3 is defined as a level set of Lyapunov function inside $\hat{D}$: $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. Starting from $\Omega_{\hat{\rho}}$, the origin of the LSTM network model of Eq. 6.3 can be rendered exponentially stable under the controller $u_1 = \Phi_{nn_1}(x) \in U_1$, and $u_2 = \Phi_{nn_2}(x) \in U_2$. It is noted that the above assumption of Eq. 5.2.10 is the same as the assumption of Eq. 7.5 for the general class of nonlinear systems of Eq. 8.1 since the LSTM network model of Eq. 6.3 can be written in the form of Eq. 8.1 (i.e., $\dot{\hat{x}} = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u$, where $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ are obtained from coefficient matrices A and Θ in Eq. 6.3). However, due to the complexity of the LSTM structure and the interactions of the nonlinear activation functions, $\hat{f}$ and $\hat{g}$ may be hard to compute explicitly. For computational convenience, at $t = t_k$, given a set of control actions $u_1(t_k) \in U_1 \backslash \{0\}$ and $u_2(t_k) \in U_2 \backslash \{0\}$ that are

applied in a sample-and-hold fashion for the time interval $t \in [t_k, t_k + h_c)$ (h_c is the integration time step), $\hat{f}$ and $\hat{g}$ can be numerically approximated as follows:

$$\hat{f}(x(t_k)) \approx \frac{\int_{t_k}^{t_k+h_c} F_{nn}(x,0,0)dt - x(t_k)}{h_c} \tag{5.1.9a}$$

$$\hat{g}_1(x(t_k)) \approx \frac{\int_{t_k}^{t_k+h_c} F_{nn}(x,u_1(t_k),0)dt - \int_{t_k}^{t_k+h_c} F_{nn}(x,0,0)dt}{h_c u_1(t_k)} \tag{5.1.9b}$$

$$\hat{g}_2(x(t_k)) \approx \frac{\int_{t_k}^{t_k+h_c} F_{nn}(x,0,u_2(t_k))dt - \int_{t_k}^{t_k+h_c} F_{nn}(x,0,0)dt}{h_c u_2(t_k)} \tag{5.1.9c}$$

The integral $\int_{t_k}^{t_k+h_c} F_{nn}(x,u_1,u_2)dt$ gives the predicted state $\hat{x}(t)$ at $t = t_k + h_c$ under the sample-and-hold implementation of the inputs $u_1(t_k)$ and $u_2(t_k)$; $\hat{x}(t_k + h_c)$ is the first internal state of the LSTM network, given that the time interval between consecutive internal states of the LSTM network is chosen as the integration time step h_c. After obtaining $\hat{f}$, $\hat{g}_1$ and $\hat{g}_2$, the stabilizing control law $\Phi_{nn_1}(x)$ and $\Phi_{nn_2}(x)$ can be computed similarly as in Eq. 7.7, where f, g_1, and g_2 are replaced by $\hat{f}$, $\hat{g}_1$, and $\hat{g}_2$, respectively. Subsequently, $\hat{V}$ can also be computed using the approximated $\hat{f}$, $\hat{g}_1$, and $\hat{g}_2$. The assumptions of Eq. 7.5 and Eq. 5.2.10 are the stabilizability requirements of the first-principles model of Eq. 8.1 and the LSTM network model of Eq. 6.3, respectively. Since the dataset for developing the LSTM network model is generated from open-loop simulations for $x \in \Omega_\rho$, $u_1 \in U_1$, and $u_2 \in U_2$, the closed-loop stability region of the LSTM system is a subset of the closed-loop stability region of the actual nonlinear system, $\Omega_{\hat{\rho}} \subseteq \Omega_\rho$. Additionally, there exist positive constants M_{nn} and L_{nn} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$, $u_1 \in U_1$ and $u_2 \in U_2$:

$$|F_{nn}(x,u_1,u_2)| \leq M_{nn} \tag{5.1.10a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x,u_1,u_2) - \frac{\partial \hat{V}(x')}{\partial x} F_{nn}(x',u_1,u_2) \right| \leq L_{nn}|x - x'| \tag{5.1.10b}$$

5.1.3 Distributed LMPC using LSTM Network Models

To achieve better closed-loop control performance, some level of communication may be established between the different controllers. In a distributed Lyapunov-based model predictive controller (LMPC) framework, we design two separate LMPCs – LMPC 1 and LMPC 2 – to compute control actions u_1 and u_2 respectively; the trajectories of control actions computed by LMPC 1 and LMPC 2 are denoted by u_{d_1} and u_{d_2}, respectively. We consider two types of

distributed control architectures: sequential and iterative distributed MPCs. Having only one-way communication, the sequential distributed MPC architecture carries less computational load with transmitting inter-controller signals. However, it must assume the input trajectories along the prediction horizon of the other controllers downstream of itself in order to make a decision. The iterative distributed MPC system allows signal exchanges between all controllers, thereby allowing each controller to have full knowledge of the predicted state evolution along the prediction horizon and yielding better closed-loop performance via multiple iterations at the cost of more computational time.

5.1.3.1 Sequential Distributed LMPC using LSTM Network Models

The communication between two LMPCs in a sequential distributed LMPC framework is one-way only; i.e., the optimal control actions obtained from solving the optimization problem of one LMPC will be relayed to the other LMPC, which will use this information to carry on with its own optimization problem. A schematic diagram of the structure of a sequential distributed LMPC system is shown in Fig. 5.1.3. In a sequential distributed LMPC system, the following implementation strategy is used:

1. At each sampling instant $t = t_k$, both LMPC 1 and LMPC 2 receive the state measurement $x(t)$, $t = t_k$ from the sensors.

2. LMPC 2 evaluates the optimal trajectory of u_{d_2} based on the state measurement $x(t)$ at $t = t_k$, sends the control action of the first sampling period $u^*_{d_2}(t_k)$ to the corresponding actuators, and sends the entire optimal trajectory to LMPC 1.

3. LMPC 1 receives the entire optimal input trajectory of u_{d_2} from LMPC 2, and evaluates the optimal trajectory of u_{d_1} based on state measurement $x(t)$ at $t = t_k$ and the optimal trajectory of u_{d_2}. LMPC 1 then sends $u^*_{d_1}(t_k)$, the optimal control action over the next sampling period to the corresponding actuators.

4. When a new state measurement is received ($k \leftarrow k+1$), go to Step 1.

We first define the optimization problem of LMPC 2, which uses the LSTM network model as its prediction model. LMPC 2 depends on the latest state measurement, but does not have any information on the value that u_{d_1} will take. Thus, to make a decision, LMPC 2 must assume a trajectory for u_{d_1} along the prediction horizon. An explicit nonlinear control law, $\Phi_{nn_1}(x)$, is used

to compute the assumed trajectory of u_{d_1}. To inherit the stability properties of $\Phi_{nn_j}(x),\ j=1,2$, u_{d_2} must satisfy a Lyapunov-based contractive constraint that guarantees a minimum decrease rate of the Lyapunov function $\hat{V}$. The optimization problem of LMPC 2 is given as follows:

$$\mathscr{J} = \min_{u_{d_2}\in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), \Phi_{nn_1}(\tilde{x}(t)), u_{d_2}(t))dt \tag{5.1.11a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), \Phi_{nn_1}(\tilde{x}(t)), u_{d_2}(t)) \tag{5.1.11b}$$

$$u_{d_2}(t) \in U_2,\ \forall\ t \in [t_k, t_{k+N}) \tag{5.1.11c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.11d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), u_{d_2}(t_k))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), \Phi_{nn_2}(x(t_k)))),$$
$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.11e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn},\ \forall\ t \in [t_k, t_{k+N}),\ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.11f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. The optimal input trajectory computed by this LMPC 2 is denoted by $u^*_{d_2}(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. This information is sent to LMPC 1. The control action computed for the first sampling period of the prediction horizon $u^*_{d_2}(t_k)$ is sent by LMPC 2 to its control actuators to be applied over the next sampling period. In the optimization problem of Eq. 5.1.11, the objective function of Eq. 5.1.11a is the integral of $L(\tilde{x}(t), \Phi_{nn_1}(t), u_{d_2}(t))$ over the prediction horizon. Note that $L(x, u_1, u_2)$ is typically in a quadratic form, i.e., $L(x, u_1, u_2) = x^T Q x + u_1^T R_1 u_1 + u_2^T R_2 u_2$, where Q, R_1, and R_2 are positive definite matrices, and the minimum of the objective function of Eq. 5.1.11a is achieved at the origin. The constraint of Eq. 5.1.11b is the LSTM network model of Eq. 6.3 that is used to predict the states of the closed-loop system. Eq. 5.1.11c defines the input constraints on u_{d_2} applied over the entire prediction horizon. Eq. 5.1.11d defines the initial condition $\tilde{x}(t_k)$ of Eq. 5.1.11b, which is the state measurement at $t = t_k$. The constraint of Eq. 5.1.11e forces the closed-loop state to move towards the origin if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the LSTM network model of Eq. 5.1.11b will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon.

The optimization problem of LMPC 1 depends on the latest state measurement as well as the control action computed by LMPC 2 (i.e., $u^*_{d_2}(t), \forall t \in [t_k, t_{k+N})$). This allows LMPC 1 to compute a control action u_{d_1} such that the closed-loop performance is optimized while guaranteeing the stability properties of the Lyapunov-based controllers using LSTM network models, $\Phi_{nn_j}(x),\ j=$

1,2, are preserved. Specifically, LMPC 1 uses the following optimization problem:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), u^*_{d_2}(t))dt \tag{5.1.12a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u_{d_1}(t), u^*_{d_2}(t)) \tag{5.1.12b}$$

$$u_{d_1}(t) \in U_1, \ \forall \ t \in [t_k, t_{k+N}) \tag{5.1.12c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.12d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), u_{d_1}(t_k), u^*_{d_2}(t_k))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), u^*_{d_2}(t_k))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.12e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.12f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. The optimal input trajectory computed by LMPC 1 is denoted by $u^*_{d_1}(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*_{d_1}(t_k)$ is sent by LMPC 1 to be applied over the next sampling period. In the optimization problem of Eq. 5.1.12, the objective function of Eq. 5.1.12a is the integral of $L(\tilde{x}(t), u_{d_1}(t), u^*_{d_2}(t))$ over the prediction horizon. The constraint of Eq. 5.1.12b is the LSTM model of Eq. 6.3 that is used to predict the states of the closed-loop system. Eq. 5.1.12c defines the input constraints on u_{d_1} applied over the entire prediction horizon. Eq. 5.1.12d defines the initial condition $\tilde{x}(t_k)$ of Eq. 5.1.12b, which is the state measurement at $t = t_k$. The constraint of Eq. 5.1.12e forces the closed-loop state to move towards the origin if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the LSTM model of Eq. 5.1.12b will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon. Since the execution of LMPC 1 depends on the results of LMPC 2, the total computation time to execute the sequential distributed LMPC design would be the sum of the time taken to solve each optimization problem in LMPC 1 and LMPC 2 respectively.

5.1.3.2 Iterative Distributed LMPC using LSTM Network Models

In an iterative distributed LMPC framework, both controllers communicate with each other to cooperatively optimize the control actions. The controllers solve their respective optimization problems independently in a parallel structure, and solutions to each control problem are exchanged at the end of each iteration. The schematic diagram of an iterative distributed LMPC

system is shown in Fig. 5.1.4.

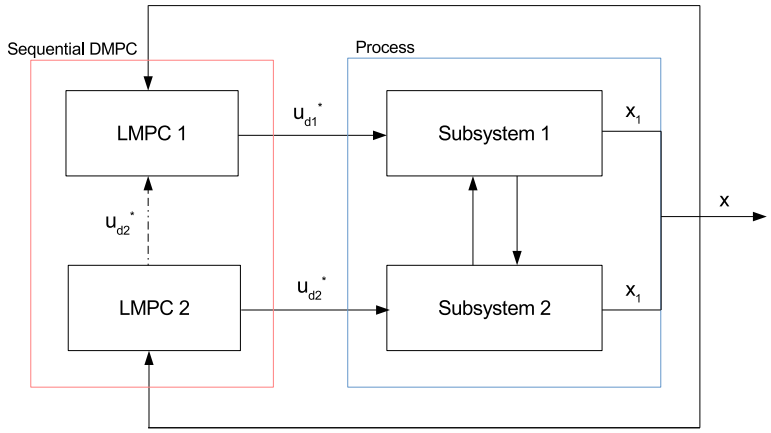


Figure 5.1.3: A schematic showing the flow of information of the sequential distributed LMPC system with the overall process.

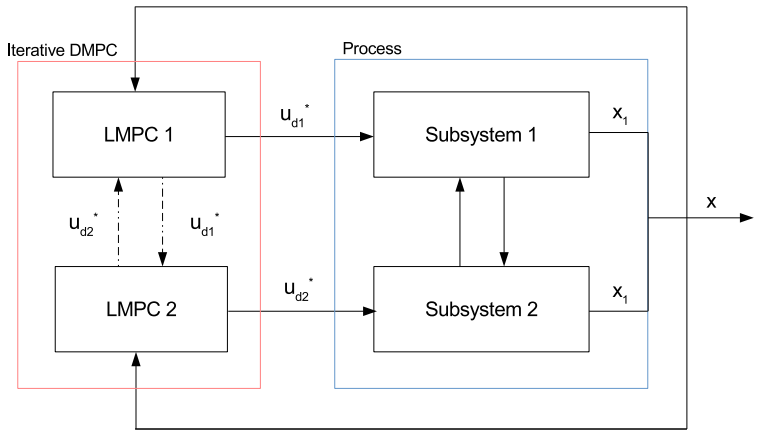


Figure 5.1.4: A schematic showing the flow of information of iterative distributed LMPC system with the overall process.

More specifically, the following implementation strategy is used:

1. At each sampling instant t_k, both LMPC 1 and LMPC 2 receive the state measurement $x(t)$

at $t = t_k$ from the sensors.

2. At iteration $c = 1$, LMPC 1 evaluates future trajectories of $u_{d_1}(t)$ assuming $u_2(t) = \Phi_{nn_2}(t), \forall t \in [t_k, t_{k+N})$. LMPC 2 evaluates future trajectories of $u_{d_2}(t)$ assuming $u_1(t) = \Phi_{nn_1}(t), \forall t \in [t_k, t_{k+N})$. The LMPCs exchange their future input trajectories, calculate and store the value of their own cost function.

3. At iteration $c > 1$:

   (a) Each LMPC evaluates its own future input trajectory based on state measurement $x(t_k)$ and the latest received input trajectories from the other LMPC.

   (b) The LMPCs exchange their future input trajectories. Each LMPC calculates and stores the value of the cost function.

4. If a termination criterion is satisfied, each LMPC sends its entire future input trajectory corresponding to the smallest value of the cost function to its actuators. If the termination criterion is not satisfied, go to Step 3 ($c \leftarrow c+1$).

5. When a new state measurement is received, go to Step 1 ($k \leftarrow k+1$).

To preserve the stability properties of the Lyapunov-based controllers $\Phi_{nn_j}(x),\ j = 1,2$, the optimized u_{d_1} and u_{d_2} must satisfy the contractive constraint that guarantees a minimum decrease rate of the Lyapunov function $\hat{V}$ given by $\Phi_{nn_j}(x),\ j = 1,2$. Following the same variables and constraints as defined in a sequential distributed LMPC design, the optimization problem of LMPC 1 in an iterative distributed LMPC at iteration $c = 1$ is presented as follows:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), \Phi_{nn_2}(\tilde{x}(t)))dt \tag{5.1.13a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u_{d_1}(t), \Phi_{nn_2}(\tilde{x}(t))) \tag{5.1.13b}$$

$$u_{d_1}(t) \in U_1,\ \forall\ t \in [t_k, t_{k+N}) \tag{5.1.13c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.13d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), u_{d_1}(t_k), \Phi_{nn_2}(x(t_k)))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), \Phi_{nn_2}(x(t_k)))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.13e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn},\ \forall\ t \in [t_k, t_{k+N}),\ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.13f}$$

At iteration $c = 1$, the optimization problem of LMPC 2 is shown as follows:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), \Phi_{nn_1}(\tilde{x}(t)), u_{d_2}(t))dt \tag{5.1.14a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), \Phi_{nn_1}(\tilde{x}(t)), u_{d_2}(t)) \tag{5.1.14b}$$

$$u_{d_2}(t) \in U_2, \ \forall\, t \in [t_k, t_{k+N}) \tag{5.1.14c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.14d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), u_{d_2}(t))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), \Phi_{nn_2}(x(t_k)))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.14e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.14f}$$

At iteration $c > 1$, following the exchange of the optimized input trajectories $u_{d_1}^*(t)$ and $u_{d_2}^*(t)$ between the two LMPCs, the optimization problem of LMPC 1 is modified as follows:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), u_{d_2}^*(t))dt \tag{5.1.15a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u_{d_1}(t), u_{d_2}^*(t)) \tag{5.1.15b}$$

$$u_{d_1}(t) \in U_1, \ \forall\, t \in [t_k, t_{k+N}) \tag{5.1.15c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.15d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), u_{d_1}(t_k), u_{d_2}^*(t_k)) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), \Phi_{nn_2}(x(t_k)))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.15e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.15f}$$

And the optimization problem of LMPC 2 becomes:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}^*(t), u_{d_2}(t))dt \tag{5.1.16a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u_{d_1}^*(t), u_{d_2}(t)) \tag{5.1.16b}$$

$$u_{d_2}(t) \in U_2, \ \forall \ t \in [t_k, t_{k+N}) \tag{5.1.16c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.1.16d}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), u_{d_1}^*(t), u_{d_2}(t))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x}(F_{nn}(x(t_k), \Phi_{nn_1}(x(t_k)), \Phi_{nn_2}(x(t_k)))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.1.16e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.1.16f}$$

At each iteration $c \geq 1$, the two LMPCs can be solved simultaneously via parallel computing in separate processors. Therefore, the total computation time required for iterative distributed LMPC would be the maximum solving time out of the two controllers accounting for all the iterations required before the termination criterion is met.

Remark 5.1.2. *One consideration that applies to any MPC system is that the computation time to calculate the solutions to the MPC optimization problem(s) must be less than the sampling time of the actual nonlinear process of Eq. 8.1. One of the main advantages of distributed MPC systems is the reduced computational complexity of the optimization problems, and thus, reduced total computational time compared to solving the optimization problem in a centralized MPC system. Therefore, running more iterations to achieve a more optimal set of solutions (i.e., lower value of the cost function) should be balanced with reducing total computation time, and there should be an upper bound enforced on the maximum number of iterations at all times to ensure calculation of control actions within the sampling time.*

Remark 5.1.3. *It is important to note that the number of iterations c could vary and will not affect the closed-loop stability of the iterative distributed LMPC system. The number of iterations c depends on the termination conditions, which can be of many forms, e.g., c must not exceed a maximum iteration number c_{max} (i.e., $c \leq c_{max}$), the computational time for solving each LMPC must not exceed a maximum time period, or the difference in the cost function or of the solution trajectory between two consecutive iterations is smaller than a threshold value. During implementation, when one such criterion is met, the iterations will be terminated.*

Remark 5.1.4. *In general, there is no guaranteed convergence of the optimal cost or solution*

of an iterative distributed LMPC system to the optimal cost or solution of a centralized LMPC. This is due to the non-convexity of the MPC optimization problems. However, the proposed implementation strategy guarantees that the optimal cost of the distributed optimization is upper bounded by the cost of the Lyapunov-based control laws $\Phi_{nn_1}(x) \in U_1$, $\Phi_{nn_2}(x) \in U_2$.

5.1.3.3 Sample-and-hold implementation of Distributed LMPC

Once both optimization problems of LMPC 1 and LMPC 2 are solved, the optimal control actions of the proposed distributed LMPC design (both sequential and iterative distributed LMPC systems) are defined as follows:

$$\begin{aligned} u_1(t) &= u_{d_1}^*(t_k), \forall t \in [t_k, t_{k+1}) \\ u_2(t) &= u_{d_2}^*(t_k), \forall t \in [t_k, t_{k+1}) \end{aligned} \tag{5.1.17}$$

The control actions computed by each LMPC will be applied in a sample-and-hold manner to the process, which may be subject to bounded disturbances (i.e., $|w(t)| \leq w_m$). In this section, we present the stability properties of the distributed LMPC design, accounting for sufficiently small bounded modeling error of the LSTM network and bounded disturbances. Following Lyapunov arguments, this property will guarantee practical stability of the closed-loop system, i.e., the closed-loop state $x(t)$ of the nominal process of Eq. 8.1 is bounded in $\Omega_{\hat{\rho}}$ at all times, and ultimately driven to a small neighborhood $\Omega_{\rho_{min}}$ around the origin under the control actions in the distributed LMPC design of Eq. 7.19 implemented in a sample-and-hold manner. First, we will present propositions demonstrating the existence of an upper bound on the state error $|e(t)| = |x(t) - \hat{x}(t)|$ provided that the modeling error $|\nu|$ and process disturbances $|w|$ are bounded, followed by propositions that demonstrate the boundedness and convergence of the LSTM system of Eq. 6.3 and of the actual nonlinear system of Eq. 8.1 under the sample-and-hold implementation of $u_1 = \Phi_{nn_1}(x) \in U_1$ and $u_2 = \Phi_{nn_2}(x) \in U_2$. Both propositions have been previously proved in [127]. Then, we will extend the proof to show the boundedness and convergence of the nonlinear system of Eq. 8.1 under the sample-and-hold implementation of $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ from the distributed LMPC design of Eq. 7.19 in the presence of sufficiently small bounded disturbances and modeling error.

Proposition 5.1.1. *Consider the nonlinear system* $\dot{x} = F(x, u_1, u_2, w)$ *of Eq. 8.1 in the presence of bounded disturbances* $|w(t)| \leq w_m$ *and the LSTM model* $\dot{\hat{x}} = F_{nn}(\hat{x}, u_1, u_2)$ *of Eq. 6.3 with the same initial condition* $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$ *and sufficiently small modeling error* $|\nu| \leq \nu_m$. *There exists a class* $\mathscr{K}$ *function* $f_w(\cdot)$ *and a positive constant* κ *such that the following inequalities hold* $\forall x, \hat{x} \in \Omega_{\hat{\rho}}$

and $w(t) \in W$*:*

$$|e(t)| = |x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + \nu_m}{L_x}(e^{L_x t} - 1) \tag{5.1.18a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{5.1.18b}$$

It has also been established that under the controller $u_1(t) = \Phi_{nn_1}(x) \in U_1$, $u_2(t) = \Phi_{nn_2}(x) \in U_2$ implemented in a sample-and-hold fashion, the closed-loop state $x(t)$ of the actual process of Eq. 8.1 and the closed-loop state $\hat{x}(t)$ of the LSTM system of Eq. 6.3 are bounded in the stability region and ultimately driven to a small neighborhood around the origin, given that the conditions of Eq. 5.2.10 are satisfied, and the modeling error $|\nu| \leq \gamma|x| \leq \nu_m$, where γ is chosen to satisfy $\gamma < \hat{c}_3/\hat{c}_4$. This is shown in the following proposition. The full proof of the following proposition can be found in [127].

Proposition 5.1.2. *Consider the system of Eq. 8.1 under the controllers* $u_j = \Phi_{nn_j}(\hat{x}) \in U_j$, $j = 1,2$*, which meet the conditions of Eq. 5.2.10. The controllers* $u_j = \Phi_{nn_j}(\hat{x}) \in U_j$, $j = 1,2$ *are designed to stabilize the LSTM system of Eq. 6.3, developed with a modeling error* $|\nu| \leq \gamma|x| \leq \nu_m$*, where* $\gamma < \hat{c}_3/\hat{c}_4$*. The control actions are implemented in a sample-and-hold fashion, i.e.,* $u_j(t) = \Phi_{nn_j}(\hat{x}(t_k))$, $j = 1,2$, $\forall t \in [t_k, t_{k+1})$*, where* $t_{k+1} := t_k + \Delta$*. Let* ε_s, $\varepsilon_w > 0$, $\Delta > 0$, $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma > 0$*, and* $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ *satisfy*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}M_{nn}\Delta \leq -\varepsilon_s \tag{5.1.19a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}M\Delta + L_w^{'}w_m \leq -\varepsilon_w \tag{5.1.19b}$$

and

$$\rho_{nn} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u_1 \in U_1, u_2 \in U_2\} \tag{5.1.20a}$$

$$\rho_{min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{5.1.20b}$$

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

$$\hat{V}(\hat{x}(t)) \leq \hat{V}(\hat{x}(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{5.1.21a}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{5.1.21b}$$

and if $x_0 \in \Omega_{\hat{\rho}}$, the state $\hat{x}(t)$ of the LSTM modeled system of Eq. 6.3 is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in $\Omega_{\rho_{nn}}$, and the state $x(t)$ of the nonlinear system of Eq. 8.1 is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in $\Omega_{\rho_{min}}$.

Proposition 6.3 demonstrates that, if $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the closed-loop state of the LSTM system of Eq. 6.3 and of the actual nonlinear process of Eq. 8.1 are both bounded in the stability region $\Omega_{\hat{\rho}}$ and they move towards the origin under $u_1(t) = \Phi_{nn_1}(x) \in U_1$ and $u_2(t) = \Phi_{nn_2}(x) \in U_2$ implemented in a sample-and-hold fashion. If $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$, the closed-loop state of the LSTM model is maintained in $\Omega_{\rho_{nn}}$ within one sampling period for all $t \in [t_k, t_{k+1})$, and the closed-loop state of the actual nonlinear system is maintained in $\Omega_{\rho_{min}}$ within one sampling period.

In the following theorem, we will prove that the optimization problem of LMPC 1 and of LMPC 2 in the distributed LMPC network can be solved with recursive feasibility, and the closed-loop stability of the nonlinear system of Eq. 8.1 is guaranteed under the sample-and-hold implementation of the optimal control actions $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ given by the distributed LMPC design of Eq. 7.19.

Theorem 5.1.1. *Consider the closed-loop system of Eq. 8.1 under $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ in the distributed LMPC design of Eq. 7.19, which are calculated based on the controllers $\Phi_{nn_j}(x),\ j = 1,2$ that satisfy Eq. 5.2.10. Let $\Delta > 0$, $\varepsilon_s > 0$, $\varepsilon_w > 0$ and $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ satisfy Eq. 6.18 and 6.19. Then, given any initial state $x_0 \in \Omega_{\hat{\rho}}$, if the conditions of Proposition 5.1.1 and Proposition 6.3 are satisfied, and the LSTM model of Eq. 6.3 has a modeling error $|\nu| \leq \gamma|x| \leq \nu_m$, $0 < \gamma < \hat{c}_3/\hat{c}_4$, then there always exists a feasible solution for the optimization problem of Eq. 5.1.11, Eq. 5.1.12, and of Eq. 5.1.15, Eq. 5.1.16. Additionally, it is guaranteed that under the distributed LMPC design $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ of Eq. 7.19, $x(t) \in \Omega_{\hat{\rho}},\ \forall t \geq 0$, and $x(t)$ ultimately converges to $\Omega_{\rho_{min}}$ for the closed-loop system of Eq. 8.1.*

Proof. The proof consists of three parts. In *Part* 1, We first prove that the optimization problem of each LMPC in the distributed LMPC network is feasible for all states $x \in \Omega_{\hat{\rho}}$. In *Part* 2, we prove the boundedness and convergence of the state in $\Omega_{\rho_{nn}}$ for the closed-loop LSTM system of Eq. 6.3 under the distributed LMPC design $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ in Eq. 7.19. Lastly, in *Part* 3, we prove the boundedness and convergence of the closed-loop state to $\Omega_{\rho_{min}}$ for the actual nonlinear system of Eq. 8.1 under the distributed LMPC design $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ in Eq. 7.19. The following proof is provided in reference to the formulations of the sequential distributed LMPC of Eq. 5.1.11 – Eq. 5.1.12, but the same result also applies to the iterative distributed LMPC of Eq. 5.1.15 – Eq. 5.1.16.

Part 1: We prove that the optimization problem of each LMPC in the distributed LMPC network is recursively feasible for all $x \in \Omega_{\hat{\rho}}$. If $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, the input trajectories $u_{d_j}(t) = \Phi_{nn_j}(x(t_k))$, $j = 1,2$, for $t \in [t_k, t_{k+1}]$ are feasible solutions to the optimization problem of LMPC j since such trajectories satisfy the input constraint on u_{d_j} of Eq. 5.1.11c in LMPC 2 and of Eq. 5.1.12c in LMPC 1 respectively, as well as the Lyapunov-based contractive constraint of Eq. 5.1.11e in LMPC 2 and of Eq. 5.1.12e in LMPC 1. Additionally, if $x(t_k) \in \Omega_{\rho_{nn}}$, the control actions given by $\Phi_{nn_j}(\tilde{x}(t_{k+i}))$, $i = 0, 1, ..., N-1$ satisfy the input constraint on u_{d_2} of Eq. 5.1.11c and the Lyapunov-based constraint of Eq. 5.1.11f in LMPC 2, and the input constraint on u_{d_1} of Eq. 5.1.12c and the Lyapunov-based constraint of Eq. 5.1.12f in LMPC 1, since it is shown in Proposition 6.3 that the states predicted by the LSTM model of Eq. 5.1.11b and of Eq. 5.1.12b remain inside $\Omega_{\rho_{nn}}$ under the controller $\Phi_{nn_j}(\tilde{x}), j = 1,2$. Therefore, for all $x_0 \in \Omega_{\hat{\rho}}$, the optimization problems of both Eq. 5.1.12 and Eq. 5.1.11 can be solved with recursive feasibility if $x(t) \in \Omega_{\hat{\rho}}$ for all times.

Part 2: Next, we prove that given any $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, the state of the closed-loop LSTM system of Eq. 6.3 is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately converges to a small neighborhood around the origin $\Omega_{\rho_{nn}}$ defined by Eq. 6.19a under the sample-and-hold implementation of the distributed LMPC design $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$ of Eq. 7.19. First, we consider $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ at $t = t_k$, therefore activating the contractive constraints of Eq. 5.1.11e and Eq. 5.1.12e. Based on the definition of ρ_{nn} in Eq. 6.19a, this means $x(t_k)$ also belongs to the region $\Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. With the conditions of Eq. 5.2.10 on $\Phi_{nn_1}(\hat{x}(t_k))$ and $\Phi_{nn_2}(\hat{x}(t_k))$ satisfied, the contractive constraints are activated such that the optimal control actions $u_{d_2}^*$, and sequentially $u_{d_1}^*$, are calculated to decrease the value of the Lyapunov function based on the states predicted by the LSTM model of Eq. 5.1.11b and Eq. 5.1.12b over the next sampling period, respectively. This is shown as follows:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t_k)) =& \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k)) \\
\leq& \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn_1}(\hat{x}(t_k)), u_{d_2}^*(t_k)) \\
\leq& \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn_1}(\hat{x}(t_k)), \Phi_{nn_2}(\hat{x}(t_k))) \\
\leq& -\hat{c}_3 |\hat{x}(t_k)|^2
\end{aligned} \tag{5.1.22}$$

The time derivative of the Lyapunov function along the trajectory of $\hat{x}(t)$ of the LSTM model of

Eq. 6.3 in $t \in [t_k, t_{k+1})$ is given by:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) =& \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k)) \\ =& \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k)) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k)) \\ & - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k))\end{aligned} \tag{5.1.23}$$

After adding and subtracting $\frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k))$, and taking into account the conditions of Eq. 5.2.10, we obtain the following inequality:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) \leq & -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k)) \\ & - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k))\end{aligned} \tag{5.1.24}$$

Based on the Lipschitz condition of Eq. 5.2.12 and that $\hat{x} \in \Omega_{\hat{\rho}}$, $u_1 \in U_1$, and $u_2 \in U_2$, the upper bound of $\dot{\hat{V}}(\hat{x}(t))$ is derived $\forall t \in [t_k, t_{k+1})$:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) \leq & -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}|\hat{x}(t) - \hat{x}(t_k)| \\ \leq & -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}M_{nn}\Delta\end{aligned} \tag{5.1.25}$$

Therefore, if Eq. 5.2.15a is satisfied, the following inequality holds $\forall \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(\hat{x}(t)) \leq -\varepsilon_s \tag{5.1.26}$$

By integrating the above equation over $t \in [t_k, t_{k+1})$, it is obtained that $\hat{V}(\hat{x}(t_{k+1})) \leq \hat{V}(\hat{x}(t_k)) - \varepsilon_s\Delta$. Therefore, $\hat{V}(\hat{x}(t)) \leq \hat{V}(\hat{x}(t_k)), \forall t \in [t_k, t_{k+1})$. We have proved that for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the state of the closed-loop LSTM system of Eq. 6.3 is bounded in the closed-loop stability region $\Omega_{\hat{\rho}}$ for all times and moves towards the origin under $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ implemented in a sample-and-hold fashion.

Next, we consider when $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$ and Eq. 5.1.26 may not hold. According to Eq. 6.19a, $\Omega_{\rho_{nn}}$ is designed to ensure that the closed-loop state $\hat{x}(t)$ of the LSTM model does not leave $\Omega_{\rho_{nn}}$ for all $t \in [t_k, t_{k+1})$, $u_1 \in U_1$, $u_2 \in U_2$, and $\hat{x}(t_k) \in \Omega_{\rho_s}$ within one sampling period. If the state $\hat{x}(t_{k+1})$ leaves Ω_{ρ_s}, the controller $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)] \in U$ will drive the state

towards Ω_{ρ_s} over the next sampling period since Eq. 5.1.26 is satisfied again at $t = t_{k+1}$. Therefore, the convergence of the state to $\Omega_{\rho_{nn}}$ for the closed-loop LSTM system of Eq. 6.3 is proved for all $\hat{x}_0 \in \Omega_{\hat{\rho}}$.

Part 3: We have proven that the closed-loop state of the LSTM system of Eq. 6.3 are bounded in $\Omega_{\hat{\rho}}$ and ultimately converge to $\Omega_{\rho_{nn}}$ under the controller $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ computed by the distributed LMPC design of Eq. 7.19 for all $\hat{x} \in \Omega_{\hat{\rho}}$. We will now prove that the controllers $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ computed by the distributed LMPC design of Eq. 7.19 are able to stabilize the actual nonlinear system of Eq. 8.1 while accounting for bounded modeling error $|\nu|$ and disturbances $|w|$. If there exists a positive real number $\gamma < \hat{c}_3/\hat{c}_4$ that constrains the modeling error $|\nu| = |F(x,u_1,u_2,0) - F_{nn}(x,u_1,u_2)| \leq \gamma|x|$ for all $x \in \Omega_{\hat{\rho}}$, $u_1 \in U_1$, $u_2 \in U_2$, then the origin of the closed-loop nominal system of Eq. 8.1 can be rendered exponentially stable under the controller $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$. This is shown by proving that $\dot{\hat{V}}$ for the nominal system of Eq. 8.1 can be rendered negative under $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$. Based on the conditions on the Lyapunov functions of Eq. 5.2.19 as derived in *Part* 2, and Eq. 5.2.10c, the time derivative of the Lyapunov function is derived as follows:

$$\begin{aligned}
\dot{\hat{V}}(x) =& \frac{\partial \hat{V}(x)}{\partial x} F(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) \\
=& \frac{\partial \hat{V}(x)}{\partial x} (F_{nn}(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k)) + F(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) - F_{nn}(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k))) \\
\leq& -\hat{c}_3|x|^2 + \hat{c}_4|x|(F(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) - F_{nn}(x, u_{d_1}^*(t_k), u_{d_2}^*(t_k))) \\
\leq& -\hat{c}_3|x|^2 + \hat{c}_4\gamma|x|^2 \\
\leq& -\tilde{c}_3|x|^2
\end{aligned} \tag{5.1.27}$$

When γ is chosen to satisfy $\gamma < \hat{c}_3/\hat{c}_4$, it holds that $\dot{\hat{V}} \leq -\tilde{c}_3|x|^2 \leq 0$ where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma > 0$. Therefore, the closed-loop state of the nominal system of Eq. 8.1 converges to the origin under $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$, $\forall\, x_0 \in \Omega_{\hat{\rho}}$ if the modeling error is sufficiently small, i.e., $|\nu| \leq \gamma|x|$.

Additionally, considering the presence of bounded disturbances (i.e., $|w| \leq w_m$), we will now prove that the closed-loop state $x(t)$ of the actual nonlinear system of Eq. 8.1 (i.e., $\dot{x} = F(x,u,w)$) is bounded in $\Omega_{\hat{\rho}}$ and ultimately converges to $\Omega_{\rho_{min}}$ under the sample-and-hold implementation of the control actions $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ as computed by the distributed LMPC design of Eq. 7.19.

Similarly, we first consider $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, which means $x(t_k)$ also belongs to the region $\Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$.We derive the time-derivative of $\hat{V}(x)$ for the nonlinear system of Eq. 8.1 with

bounded disturbances as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k), w) \\ &= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k), w) \\ &\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0)\end{aligned} \tag{5.1.28}$$

From Eq. 5.2.21, we know that $\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) \leq -\tilde{c}_3 |x(t_k)|^2$ holds for all $x \in \Omega_{\hat{\rho}}$. Based on Eq. 5.2.10a and the Lipschitz condition in Eq. 5.2.5, the following inequality is obtained for $\dot{\hat{V}}(x(t))$ $\forall t \in [t_k, t_{k+1})$ and $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_{d_1}^*(t_k), u_{d_2}^*(t_k), w) - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_{d_1}^*(t_k), u_{d_2}^*(t_k), 0) \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} |x(t) - x(t_k)| + L_w^{'} |w| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} M \Delta + L_w^{'} w_m\end{aligned} \tag{5.1.29}$$

Therefore, if Eq. 5.2.15b is satisfied, the following inequality holds $\forall x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\varepsilon_w \tag{5.1.30}$$

Integrating Eq. 5.2.22 will show that Eq. 5.2.17b holds; hence, the closed-loop state of the actual nonlinear process of Eq. 8.1 is maintained in $\Omega_{\hat{\rho}}$ for all times, and can be driven towards the origin in every sampling period under the controller $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$. Additionally, if $x(t_k) \in \Omega_{\rho_s}$, considering the sample-and-hold implementation of control actions, it has been shown in *Part* 2 that the state of the LSTM model of Eq. 6.3 is maintained in $\Omega_{\rho_{nn}}$ within one sampling period. Considering the bounded error between the state of the LSTM of Eq. 6.3 model and the state of the nonlinear system of Eq. 8.1 given by Eq. 5.1.18a, there exists a compact set $\Omega_{\rho_{min}} \supset \Omega_{\rho_{nn}}$ that satisfies Eq. 6.19b such that the state of the actual nonlinear system of Eq. 8.1 does not leave $\Omega_{\rho_{min}}$ during one sampling period if the state of the LSTM model of Eq. 6.3 is bounded in $\Omega_{\rho_{nn}}$. If the state $x(t)$ enters $\Omega_{\rho_{min}} \backslash \Omega_{\rho_s}$, we have shown that Eq. 5.2.22 holds, and thus, the state will be driven towards the origin again under $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ during the next sampling period.

Consider $x(t) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ at $t = t_k$ where the contractive constraints of Eq. 5.1.11e and Eq. 5.1.12e are activated. Since $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, it follows that $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, hence Eq. 5.2.22 holds, implying that the closed-loop state will be driven towards the origin in every sampling step under $[u_1 \;\; u_2] = [u_{d_1}^*(t_k) \;\; u_{d_2}^*(t_k)]$ and can be driven into $\Omega_{\rho_{nn}}$ within finite sampling steps. After the state enters $\Omega_{\rho_{nn}}$, the constraint of Eq. 5.1.11f and Eq. 5.1.12f are activated to maintain the predicted states of the LSTM model of Eq. 5.1.11b and Eq. 5.1.12b in $\Omega_{\rho_{nn}}$ over the entire prediction horizon. As we characterize a region $\Omega_{\rho_{min}}$ that satisfies Eq. 6.19b, the closed-loop state $x(t)$ of the nonlinear system of Eq. 8.1, $\forall t \in [t_k, t_{k+1})$ is guaranteed to be bounded in $\Omega_{\rho_{min}}$ if the predicted state by the LSTM model of Eq. 5.1.11b and Eq. 5.1.12b remains in $\Omega_{\rho_{nn}}$. Therefore, at the next sampling step $t = t_{k+1}$, if the state $x(t_{k+1})$ is still bounded in $\Omega_{\rho_{nn}}$, the constraint of Eq. 5.1.11f and Eq. 5.1.12f maintains the predicted state $\hat{x}$ of the LSTM model of Eq. 5.1.11b and Eq. 5.1.12b in $\Omega_{\rho_{nn}}$ such that the actual state x of the nonlinear system of Eq. 8.1 stays inside $\Omega_{\rho_{min}}$. However, if $x(t_{k+1}) \in \Omega_{\rho_{min}} \backslash \Omega_{\rho_{nn}}$, following the proof we have shown for the case that $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, the contractive constraint of Eq. 5.1.11e and Eq. 5.1.12e will be activated instead to drive it towards the origin. This completes the proof of boundedness of the states of the closed-loop system of Eq. 8.1 in $\Omega_{\hat{\rho}}$ and convergence to $\Omega_{\rho_{min}}$ for any $x_0 \in \Omega_{\hat{\rho}}$. □

Remark 5.1.5. *Although there exists approximation error induced by Eq. 5.2.11, it does not jeopardize the stability analysis. This is because the same numerical approximations are used universally whenever the calculations for $\hat{f}$, $\hat{g}_1$ and $\hat{g}_2$ are invoked. This includes when calculating $\Phi_{nn_1}(x), \Phi_{nn_2}(x)$, when characterizing the stability region, and when calculating the time-derivative of the control Lyapunov function in the contractive constraint for the optimization problems of the distributed MPC system. Therefore, $[u_1 \;\; u_2] = [\Phi_{nn_1}(x) \;\; \Phi_{nn_2}(x)]$ still remains a feasible solution to the distributed MPC that will render the origin of the nonlinear process exponentially stable for all initial conditions $x_0 \in \Omega_{\hat{\rho}}$.*

5.1.4 Application to a Two-CSTR-in-Series Process

A chemical process example is utilized to demonstrate the application of sequential distributed and iterative distributed model predictive control using the proposed LSTM model, the results of which will be compared to that of centralized model predictive control. Specifically, two well-mixed, non-isothermal continuous stirred tank reactors (CSTRs) in series are considered where an irreversible second-order exothermic reaction takes place in each reactor as shown in Fig. 5.2.5.

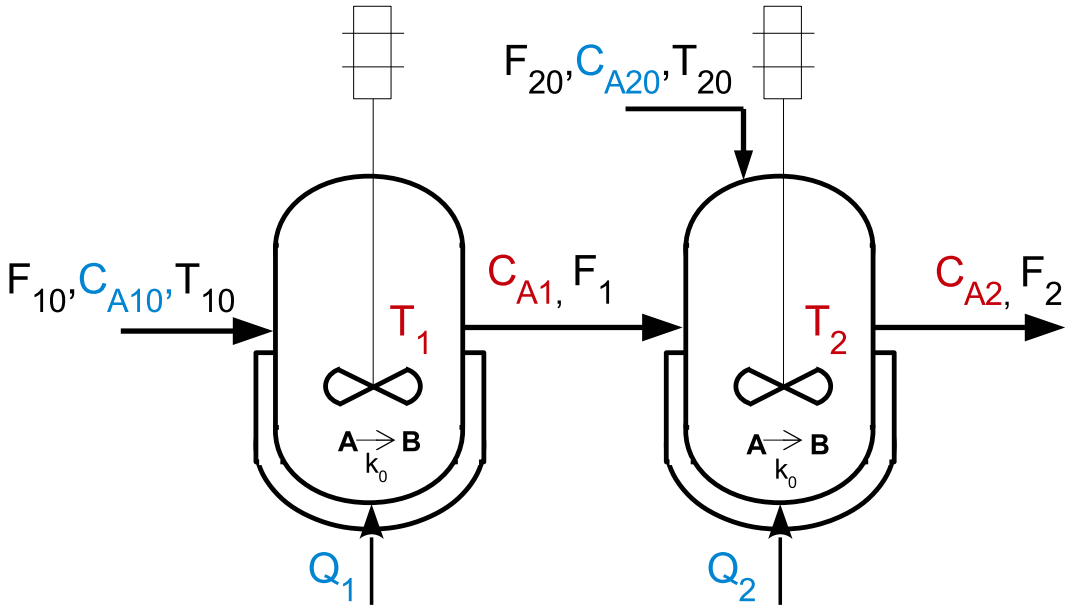


Figure 5.1.5: Process flow diagram of two CSTRs in series.

The reaction transforms a reactant A to a product B ($A \rightarrow B$). Each of the two reactors are fed with reactant material A with the inlet concentration C_{Aj0}, the inlet temperature T_{j0} and feed volumetric flow rate of the reactor F_{j0}, $j = 1,2$, where $j = 1$ denotes the first CSTR and $j = 2$ denotes the second CSTR. Each CSTR is equipped with a heating jacket that supplies/removes heat at a rate Q_j, $j = 1,2$. The CSTR dynamic models is obtained by the following material and

energy balance equations:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_1}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{5.1.31a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_1} \tag{5.1.31b}$$

$$\frac{dC_{B1}}{dt} = -\frac{F_{10}}{V_1} C_{B1} + k_0 e^{\frac{-E}{RT_1}} C_{A_1}^2 \tag{5.1.31c}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_2} C_{A20} + \frac{F_{10}}{V_2} C_{A1} - \frac{F_{10}+F_{20}}{V_2} C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{5.1.31d}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2} T_{20} + \frac{F_{10}}{V_2} T_1 - \frac{F_{10}+F_{20}}{V_2} T_2 + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_2} \tag{5.1.31e}$$

$$\frac{dC_{B2}}{dt} = \frac{F_{10}}{V_2} C_{B1} - \frac{F_{10}+F_{20}}{V_2} C_{B2} + k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{5.1.31f}$$

where C_{Aj}, V_j, T_j and Q_j, $j = 1,2$ are the concentration of reactant A, the volume of the reacting liquid, the temperature, and the heat input rate in the first and the second reactor, respectively. The reacting liquid has a constant density of ρ_L and a constant heat capacity of C_p for both reactors. ΔH, k_0, E, and R represent the enthalpy of the reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively. Process parameter values are listed in Table 5.2.1. The manipulated inputs for both CSTRs are the inlet concentration of species A and the heat input rate, which are represented by the deviation variables $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1,2$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{Aj0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q_j| \leq 5 \times 10^5\ kJ/hr,\ j = 1,2$. Therefore, the states of the closed-loop system are $x^T = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}\ \ C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]$, where C_{A1_s}, C_{A2_s}, T_{1_s} and T_{2_s} are the steady-state values of concentration of A and temperature in the first and the second reactor, such that the equilibrium point of the system is at the origin of the state-space. It is noted that the states of the first CSTR can be separately denoted as $x_1^T = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}]$ and the states of the second CSTR are denoted as $x_2^T = [C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]$. In a centralized MPC framework, feedback measurement on all states x is received by the controller, and the manipulated inputs for the entire system, $u^T = [\Delta C_{A10}\ \ \Delta Q_1\ \ \Delta C_{A20}\ \ \Delta Q_2]$, are computed by one centralized controller. In a distributed LMPC system, both LMPCs have access to full-state information as well as the overall model of the two-CSTR process. Both LMPC 1 and LMPC 2 receive feedback on $x(t)$; LMPC 1 optimizes u_1^T and LMPC 2 optimizes u_2^T. The common control objective of the model predictive controllers is to stabilize the two-CSTR process at the unstable operating steady-state $x_s^T = [C_{A1_s}\ \ C_{A2_s}\ \ T_{1_s}\ \ T_{2_s}]$, whose values are presented in Table 5.2.1.

Table 5.1.1: Parameter values of the CSTRs.

$T_{10} = 300\ K$	$T_{20} = 300\ K$
$F_{10} = 5\ m^3/hr$	$F_{20} = 5\ m^3/hr$
$V_1 = 1\ m^3$	$V_2 = 1\ m^3$
$T_{1_s} = 401.9\ K$	$T_{2_s} = 401.9\ K$
$C_{A1_s} = 1.954\ kmol/m^3$	$C_{A2_s} = 1.954\ kmol/m^3$
$C_{A10_s} = 4\ kmol/m^3$	$C_{A20_s} = 4\ kmol/m^3$
$Q_{1_s} = 0.0\ kJ/hr$	$Q_{2_s} = 0.0\ kJ/hr$
$k_0 = 8.46 \times 10^6\ m^3/kmol\ hr$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$C_p = 0.231\ kJ/kg\ K$	$R = 8.314\ kJ/kmol\ K$
$\rho_L = 1000\ kg/m^3$	$E = 5 \times 10^4\ kJ/kmol$

The explicit Euler method with an integration time step of $h_c = 10^{-4}\ hr$ is used to numerically simulate the dynamic model of Eq. 8.45. The nonlinear optimization problems of the distributed LMPCs of Eq. 5.1.11 – Eq. 5.1.12, and of Eq. 5.1.15 – Eq. 5.1.16 are solved using the Python module of the IPOPT software package [113], named PyIpopt with a sampling period $\Delta = 10^{-2}\ hr$. The objective function in the distributed LMPC optimization problem has the form $L(x,u_1,u_2) = x^T Qx + u_1^T R_1 u_1 + u_2^T R_2 u_2$, where $Q = diag[2\times 10^3\ \ 1\ \ 2\times 10^3\ \ 1]$, $R_1 = R_2 = diag[8\times 10^{-13}\ \ 0.001]$; the same objective function is used in both LMPC 1 and LMPC 2 in all distributed LMPC systems. The overall control Lyapunov function is the sum of the control Lyapunov functions for the two CSTRs, i.e., $V(x) = V_1(x_1) + V_2(x_2) = x_1^T P_1 x_1 + x_2^T P_2 x_2$, with the following positive definite P matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{5.1.32}$$

Given that the modeling error between the LSTM model and the first-principles model is sufficiently small, the control Lyapunov function for the first-principles model of the nonlinear process V can be used as the control Lyapunov function for the LSTM model $\hat{V}$ as well.

5.1.4.1 LSTM Network Development

Open-loop simulations are conducted for finite sampling steps for various initial conditions inside Ω_ρ, where $\rho = 392$, using the nonlinear system of Eq. 8.1 under various $u_1 \in U_1,\ u_2 \in U_2$ applied in a sample-and-hold manner. These trajectories which consist of training sample points are collected

with a time interval of $5 \times h_c$. The LSTM model is then developed to predict future states over one sampling period Δ. This LSTM model captures the dynamics of the overall two-CSTR process of Eq. 8.45, and can be used in all individual distributed LMPCs or in the centralized LMPC. The LSTM network is developed using *Keras* with 1 hidden layer consisting of 50 units, where *tanh* function is used as the activation function, and *Adam* is used as the optimizer. The stopping criteria for the training process include that the mean squared modeling error being less than 5×10^{-7} and the mean absolute percentage of the modeling error being less than 4.5×10^{-4}. After 50 epochs of training, with each epoch taking on average 200 *s*, the mean squared error between the predicted states of the LSTM network model and of the first-principles models is 4.022×10^{-7} and the mean absolute error is 4.254×10^{-4}. After obtaining an LSTM model with sufficiently small modeling error, the Lyapunov function of the LSTM model, $\hat{V}$, is chosen to be the same as $V(x)$. Subsequently, the set $\hat{D}$ can be characterized using the controllers $[u_1 \;\; u_2] = [\Phi_{nn_1}(x) \;\; \Phi_{nn_2}(x)]$, from which the closed-loop stability region $\Omega_{\hat{\rho}}$ for the LSTM system can be characterized as the largest level set of $\hat{V}$ in $\hat{D}$ while also being a subset of Ω_{ρ}. The positive constants $\hat{\rho}_1$ and $\hat{\rho}_2$, which are used to define the largest level sets of the control Lyapunov functions for the first and the second CSTR respectively, are $\hat{\rho}_1 = \hat{\rho}_2 = 380$. Additionally, the ultimate bounded region $\Omega_{\rho_{nn}}$, and subsequently, $\Omega_{\rho_{min}}$, are chosen to be $\rho_{nn} = 10$ and $\rho_{min} = 12$, determined via extensive closed-loop simulations with $u_1 \in U_1, u_2 \in U_2$. Readers interested in more computational details on the development of a recurrent neural network model can refer to [128].

In this study, we simulate three different types of control systems to compare their closed-loop control performances: a centralized LMPC, an iterative distributed LMPC system, and a sequential distributed LMPC system. We develop an LSTM model for the overall two-CSTR process, which is the same model used in both centralized LMPC system and distributed LMPC systems. When this LSTM network model is implemented online during closed-loop simulations, the inputs to the LSTM network are $x(t)$ and $u(t)$ at $t = t_k$, and the outputs are the predicted future states $\hat{x}(t)$ at $t = t_{k+1}$; more examples on this overall LSTM model can be found in [125]. It should be noted that, depending on the different architectures of the control systems, the choice of inputs and outputs as well as the structure of the LSTM model used in the control system may be different.

5.1.4.2 Closed-loop Model Predictive Control Simulations

To demonstrate the efficacy of the distributed model predictive control network using LSTM models, the following simulations are carried out. First, we simulate a centralized LMPC using the LSTM network for the overall two-CSTR process as its prediction model, where the four

manipulated inputs are $u^T = [\Delta C_{A10} \;\; \Delta Q_1 \;\; \Delta C_{A20} \;\; \Delta Q_2]$, and it receives feedback on all states $x^T = [C_{A1} - C_{A1_s} \;\; T_1 - T_{1_s} \;\; C_{A2} - C_{A2_s} \;\; T_2 - T_{2_s}]$. Then, we simulate sequential distributed LMPCs and iterative distributed LMPCs, where LMPC 1 and LMPC 2 in both distributed frameworks use the same LSTM model for the overall two-CSTR process as used in a centralized LMPC system. The closed-loop control performances of the aforementioned control networks are compared, the comparison metrics include the computation time of calculating the solutions to the LMPC optimization problem(s), as well as the sum squared error of the closed-loop states $x(t)$ for a total simulation period of $t_p = 0.3\ hr$. It should be noted that, since iterative distributed LMPC systems allow parallel computing of the individual controllers, the computation time for obtaining the final solutions to the optimization problems of Eq. 5.1.15 – Eq. 5.1.16 should be the maximum time of the two controllers, accounting for all iterations carried out before the termination criterion is reached. The termination criterion used was that the computation time for solving each LMPC must not exceed the sampling period, Δ. On the other hand, in a sequential distributed LMPC system, since the computation of LMPC 1 depends on the optimal trajectory of control action calculated by LMPC 2, the total computation time taken to obtain the solutions to the optimization problems of Eq. 5.1.11 – Eq. 5.1.12 must be the sum of the time taken by the two controllers.

Table 5.2.2 shows the average computation time for solving the optimization problem(s) of the distributed and centralized LMPC systems, as well as the sum of squared percentage error of all states in the form of $SSE = \int_0^{t_p} (\frac{C_{A1} - C_{A1_s}}{C_{A1_s}})^2 + (\frac{T_1 - T_{1_s}}{T_{1_s}})^2 + (\frac{C_{A2} - C_{A2_s}}{C_{A2_s}})^2 + (\frac{T_2 - T_{2_s}}{T_{2_s}})^2 dt$. It is shown in Table 5.2.2 that when the two-CSTR process is operated under distributed LMPC systems, the sum squared error and the average computation time are reduced compared to the case where a centralized LMPC system is used. Moreover, it is shown that the iterative distributed MPC using the LSTM model has a lower mean computation time and a lower sum squared error than the sequential distributed MPC, and the distributed MPC in general achieve a lower sum squared error than the centralized MPC. It should also be noted that the computation time of all simulated control systems are lower than the sampling period used in the two-CSTR process such that the proposed control system can be implemented without computational issues.

Table 5.1.2: Average LMPC computation time in one sampling period and the sum of squared percentage error of all states along the closed-loop trajectory under iterative distributed, sequential distributed, and centralized LMPC systems using their respective LSTM models with a total simulation time of 0.3 *hr*.

	Ave. Computation Time (s)	Sum Squared Error
Iterative Distributed LMPC	26.70	2.85
Sequential Distributed LMPC	29.55	3.04
Centralized LMPC	35.26	3.08

In this work, we develop machine-learning-based models for the two-CSTR process of Eq. 8.45 assuming that the first-principles model of Eq. 8.45 is unknown. However, in order to have a reasonable baseline for comparison, we show the simulation results of each distributed control framework using the first-principles model of the nonlinear process of Eq. 8.45. Furthermore, in real-life scenarios where the first-principles model of an industrial-scale chemical plant is not available, the comparison of closed-loop control performances using machine-learning-based models can be conducted against plant data. To further illustrate the closed-loop performances of sequential and iterative LMPC systems using the LSTM model, the closed-loop state evolution showing the convergence of closed-loop states from the initial conditions $x_0^T = [-1.5kmol/m^3 \quad 70K \quad 1.5kmol/m^3 \quad -70K]$ under the sequential and iterative LMPC using the LSTM model are plotted in Figs. 5.1.6 – 5.1.7 along with the closed-loop trajectories under the respective distributed LMPCs using the first-principles model as a baseline for comparison. All states converge to $\Omega_{\rho_{min}}$ within 0.10 *hr* under the sequential and iterative distributed MPCs using the LSTM model. It is reported that using the LSTM model, the sum squared error of an operation period of 0.3 *hr* under iterative distributed LMPC and under sequential distributed LMPC are 2.85 and 3.04, respectively. Using the first-principles model, the sum squared error of the same operation period with the same initial conditions under iterative and sequential distributed LMPC systems are 2.96 and 2.98, respectively, which is on par with the sum squared error achieved using LSTM network, with the iterative distributed LMPC obtaining even a lower sum squared error using LSTM network than using first-principles. Through closed-loop simulations and performance metrics comparisons, we have demonstrated the efficacy of distributed LMPC systems using LSTM network models.

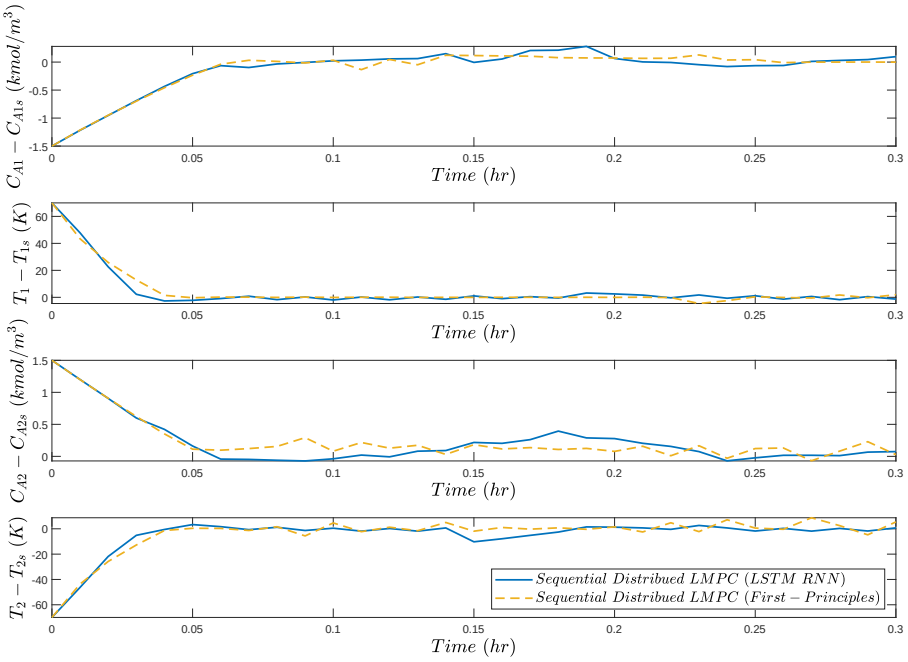


Figure 5.1.6: Closed-loop state trajectories of the sequential distributed LMPC systems using LSTM model and first-principles models respectively.

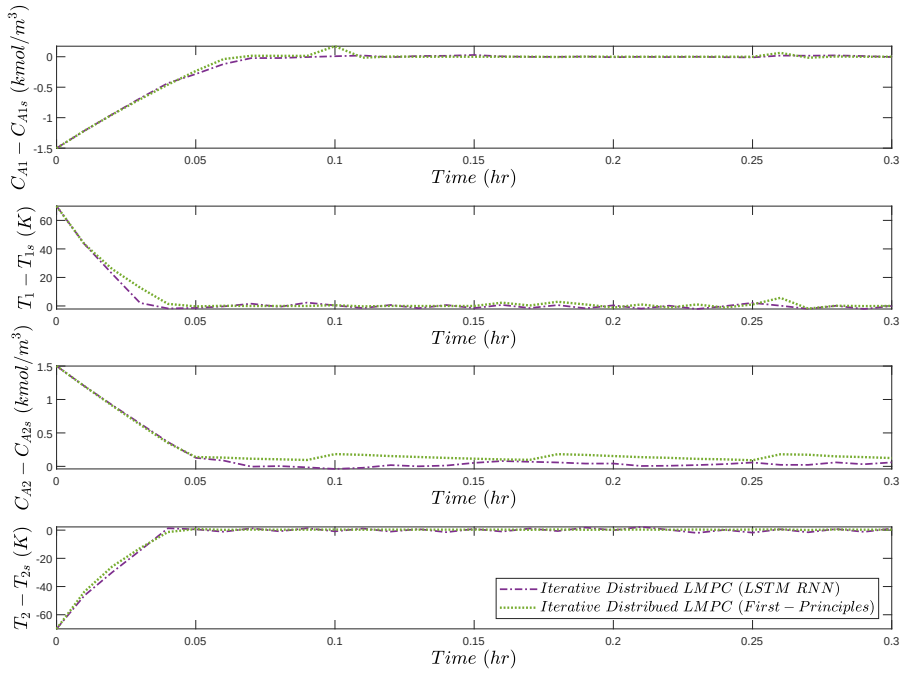


Figure 5.1.7: Closed-loop state trajectories of the iterative distributed LMPC systems using LSTM model and first-principles models respectively.

Chapter 5.2

Decentralized Machine-Learning-Based Predictive Control of Nonlinear Processes

This chapter focuses on the design of decentralized model predictive control (MPC) systems for nonlinear processes, where the nonlinear process is broken down into multiple, yet coupled subsystems and the dynamic behavior of each subsystems is described by machine learning models. One decentralized MPC is designed and used to control each subsystem while accounting for the interactions between subsystems through feedback of the entire process state. The closed-loop stability of the overall nonlinear process network and the performance properties of the decentralized model predictive control system using machine-learning prediction models are analyzed. More specifically, multiple recurrent neural network models suited for each different subsystem need to be trained with a sufficiently small modeling error from their respective actual nonlinear process models to ensure closed-loop stability. These recurrent neural network models are subsequently used as the prediction model in decentralized Lyapunov-based MPCs to achieve efficient real-time computation time while ensuring closed-loop state boundedness and convergence to the origin. The simulation results of a nonlinear chemical process network example demonstrate the effective closed-loop control performance when the process is operated under the decentralized MPCs using the independently-trained recurrent neural network models, as well as the improved computational efficiency compared to the closed-loop simulation of a centralized MPC system.

The integration of machine-learning-based modeling methods and various advanced control architectures is a broad field with expanding research scope. Using state-of-the-art machine-learning methods to address the issues of model uncertainties in a decentralized

control structure highlights the research interest of this study. In this chapter, we introduce decentralized model-based control frameworks, where each decentralized controller employs a Long-Short-Term-Memory (LSTM) network – a particular class of RNN. One decentralized controller is designed and designated to one subsystem of the overall process, and each decentralized controller is designed via Lyapunov-based model predictive control (LMPC) theory ([79]). We analyze the stability properties of the decentralized LMPC system that uses LSTM network models as the prediction model for each subsystem, and then compare the closed-loop performances of the decentralized LMPCs with those using first-principles models as the prediction model, and lastly compare with the closed-loop performance of a centralized LMPC system. The remainder of the paper is organized as follows. Section 2 presents preliminaries on notation, the general class of nonlinear systems considered, and the stabilizability assumptions. An introduction on RNN, specifically the structure and development of LSTM networks, as well as Lyapunov-based control using LSTM networks are presented in Section 3. The formulation and stability proofs of the decentralized LMPC systems using LSTM models are outlined in Section 4. In Section 5, closed-loop simulations of a two-CSTR-in-series process under the decentralized LMPC system are presented.

5.2.1 Preliminaries

5.2.1.1 Notation

Throughout the manuscript, the notation x^T is used to denote the transpose of x. $|\cdot|$ is used to denote the Euclidean norm of a vector. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. A continuous function $\alpha : [0,a) \to [0,\infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. The function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain. $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$.

5.2.1.2 Class of Systems

Consider a general class of continuous-time nonlinear systems in which several distinct sets of manipulated inputs are used, with each set of manipulated inputs regulating a specific subsystem of the process. The class of the overall continuous-time nonlinear system is described as follows:

$$\dot{x} = F(x,u,w) := f(x) + g(x)u + v(x)w, \;\; x(t_0) = x_0 \tag{5.2.1}$$

where $x \in \mathbf{R}^n$ is the vector of all states of the nonlinear system, $u \in \mathbf{R}^m$ is the vector of all manipulated inputs of the system, and $w \in \mathbf{R}^l$ is the disturbance vector for the entire system. $f(\cdot)$, $g(\cdot)$, and $v(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, $n \times m$, and $n \times l$, respectively. We refer to each subpart of the process as a subsystem. Throughout the manuscript, we consider $j = 1,...,N_{sys}$ subsystems, with each subsystem j consisting of states x_j which are regulated by only u_j and potentially impacted by states in other subsystems due to coupling between subsystems. The continuous-time nonlinear dynamics of the subsystem j is described as follows:

$$\dot{x}_j = F_j(x,u_j,w) := f_j(x) + g_j(x)u_j + v_j(x)w,\ x_j(t_0) = x_{j_0},\ \forall j = 1,...,N_{sys} \tag{5.2.2}$$

where N_{sys} represents the number of subsystems, $x_j \in \mathbf{R}^{n_j}$ represents the state vector for subsystem j, and x represents the vector of all states $x = [{x_1}^T \cdots {x_{N_{sys}}}^T]^T \in \mathbf{R}^n$, where $n = \sum_{j=1}^{N_{sys}} n_j$. $u_j \in \mathbf{R}^{m_j}$ is the set of manipulated input vectors for each subsystem j, which together constitute the vector of all manipulated inputs $u \in \mathbf{R}^m$ with $m = \sum_{j=1}^{N_{sys}} m_j$. The manipulated input vector constraints are defined by $u_j \in U_j := \{u_{j_i}^{\min} \leq u_{j_i} \leq u_{j_i}^{\max}, i = 1,...,m_j\} \subset \mathbf{R}^{m_j}, \forall j = 1,...,N_{sys}$. Therefore, the set that bounds the manipulated input vector u for the overall system is denoted by U, which is the union of all $U_j,\ j = 1,...,N_{sys}$. $w \in W$ is the disturbance vector with $W := \{w \in \mathbf{R}^l \mid |w| \leq w_m,\ w_m \geq 0\}$. $f_j(\cdot)$, $g_j(\cdot)$, and $v_j(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n_j \times 1$, $n_j \times m_j$, and $n_j \times l$, respectively. The initial time t_0 is taken to be zero ($t_0 = 0$). We assume that $f_j(0) = 0,\ \forall j = 1,...,N_{sys}$, thus, the origin is a steady-state of the nominal system of Eq. 8.1 (i.e., $u(t) \equiv 0,\ w(t) \equiv 0$). Therefore, we have $(x_s,u_s) = (0,0)$, where x_s and u_s are the steady-state state and input vectors, respectively.

5.2.1.3 Stability Assumptions

Depending on the partitioning of the overall large-scale system, there may exist interacting dynamics between the subsystems, where the states of one subsystem may be impacted by the states of other subsystems. Accounting for the coupling effects between these subsystems, we assume that there exist stabilizing control laws $u_j = \Phi_j(x) \in U_j$ which regulate the individual subsystems $j = 1,...,N_{sys}$ and will be applied to the control actuators in the respective subsystems such that the origin of the overall system of Eq. 8.1 with $w(t) \equiv 0$ is rendered exponentially stable. This implies that there exists a $\mathscr{C}^1$ control Lyapunov function $V(x)$ such that the following

inequalities hold for all $x \in \mathbf{R}^n$ in an open neighborhood D around the origin:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{5.2.3a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \mathbf{\Phi}(x), 0) \leq -c_3|x|^2, \tag{5.2.3b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{5.2.3c}$$

where c_1, c_2, c_3 and c_4 are positive constants. $F(x,u,w)$ represents the overall nonlinear system of Eq. 8.1. $\mathbf{\Phi}(x) = [\Phi_1(x)^T \cdots \Phi_{N_{sys}}(x)^T]^T$ represents the vector containing the candidate controllers for each subsystem j, i.e., $\Phi_j(x) \in \mathbf{R}^{m_j}$, for $j = 1,...,N_{sys}$. The candidate controller for each subsystem j is given in the following form:

$$\phi_{j_i}(x) = \begin{cases} -\dfrac{p + \sqrt{p^2 + q^4}}{q^T q} q & \text{if} \quad q \neq 0 \\ 0 & \text{if} \quad q = 0 \end{cases} \tag{5.2.4a}$$

$$\Phi_{j_i}(x) = \begin{cases} u_{j_i}^{min} & \text{if} \quad \phi_{j_i}(x) < u_{j_i}^{min} \\ \phi_{j_i}(x) & \text{if} \quad u_{j_i}^{min} \leq \phi_{j_i}(x) \leq u_{j_i}^{max} \\ u_{j_i}^{max} & \text{if} \quad \phi_{j_i}(x) > u_{j_i}^{max} \end{cases} \tag{5.2.4b}$$

where p denotes $\frac{\partial V(x)}{\partial x_j} f_j(x)$ and q denotes $\frac{\partial V(x)}{\partial x_j} g_{j_i}(x)$. Here, $f_j \in \mathbf{R}^{n_j}$ and $g_{j_i} \in \mathbf{R}^{n_j \times m_j}$ for $j = 1,...,N_{sys}$, and $i = 1,...,m_j$ for subsystem j corresponding to the vector of control actions $\Phi_j(x) \in \mathbf{R}^{m_j}$. It should be noted that the control Lyapunov function $V(x)$ can be a linear combination of multiple control Lyapunov functions V_j, where each V_j is designated for the subsystem j and is a function of x_j only, i.e., $V(x) = \sum_{j=1}^{N_{sys}} V_j(x_j)$. Thus, $\frac{\partial V(x)}{\partial x_j} = \frac{\partial V_j(x_j)}{\partial x_j}$, $\forall\ j = 1,...,N_{sys}$, and the time-derivative of V can be represented as $\dot{V}(x) = L_f V + L_g V u = \sum_{j=1}^{N_{sys}} \frac{\partial V(x)}{\partial x_j} (f_j + \sum_{i=1}^{m_j} g_{j_i} u_{j_i})$. $\phi_{j_i}(x)$ of Eq. 7.7a represents the i^{th} component of the control law $\phi_j(x)$. $\Phi_{j_i}(x)$ of Eq. 7.7 represents the i^{th} component of the saturated control law $\Phi_j(x)$ that accounts for the input constraints $u_j \in U_j$ for subsystem j. Note that the candidate control law $\Phi_j(x)$ is calculated based on the nonlinear dynamics of the subsystem j of Eq. 5.2.2, and the set of candidate control laws for the overall system is denoted as $\mathbf{\Phi}(x) = [\Phi_1(x)^T \cdots \Phi_{N_{sys}}(x)^T]^T \in U$, which together can render the overall system of Eq. 8.1 with $w \equiv 0$ exponentially stable.

Based on Eq. 7.5, we can first characterize a region where the time-derivative of the Control Lyapunov function V is rendered negative definite under the controllers $\mathbf{\Phi}(x) =$

$[\Phi_1(x)^T \cdots \Phi_{N_{sys}}(x)^T]^T \in U$ as $D = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V + L_g V u \leq -c_3|x|^2, u = \mathbf{\Phi}(x) \in U\} \cup \{0\}$. Subsequently, the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 8.1 is defined as a level set of V, which is inside D: $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset D$. Also, the Lipschitz property of $F(x,u,w)$ combined with the bounds on u and w imply that there exist positive constants $M, L_x, L_w, L_x^{'}, L_w^{'}$ such that the following inequalities hold for all $x, x' \in \Omega_\rho, u \in U$ and $w \in W$:

$$|F(x,u,w)| \leq M \tag{5.2.5a}$$

$$|F(x,u,w) - F(x',u,0)| \leq L_x|x-x'| + L_w|w| \tag{5.2.5b}$$

$$\left|\frac{\partial V(x)}{\partial x}F(x,u,w) - \frac{\partial V(x')}{\partial x}F(x',u,0)\right| \leq L_x^{'}|x-x'| + L_w^{'}|w| \tag{5.2.5c}$$

Note that the controller $\Phi_{j_i}(x),\ i = 1,...,m_j$ of Eq. 7.7 is a candidate stabilizing control law for the $u_j \in \mathbf{R}^{m_j}$ inputs regulating subsystem j accounting for the interacting dynamics between subsystems, and the nominal system of Eq. 8.1 is rendered exponentially stable under the set of all such stabilizing control laws for N_{sys} subsystems, $\mathbf{\Phi}(x) \in U \subset \mathbf{R}^m$.

Remark 5.2.1. *Before we proceed, it is important to elaborate on the structure and differences between decentralized and distributed control systems. Both decentralized and distributed control systems aim to alleviate the computational complexity and effort of a centralized control problem used to regulate multiple subsystems. More specifically, there exists inter-controller communication in a distributed control system, sharing information on the control actions calculated by each local controller. This means that additional communication channels need to be established between the local controllers, which contributes to additional communication network traffic. Correspondingly, the computation time may increase (e.g., in a sequential distributed control system, the controllers need to wait for the calculated results from the precedent controllers and are executed in a sequential manner; in an iterative distributed control system, while the calculated control actions are exchanged simultaneously past the first iteration, the additional iterations also contribute to longer computation time). On the other hand, in a decentralized control system, each local controller concurrently calculates the optimal control actions based on state feedback measurements, thereby significantly reducing the computational time for control action calculation. However, without information on the control actions taken by the other controllers, the cost function minimized by each local controller only includes the relevant control actions and the corresponding predicted states of the respective subsystem, thereby yielding inferior closed-loop performance.*

Remark 5.2.2. *It is important to note that, regardless of the open-loop stability property of the equilibrium point in which the process is operated at, the stabilizability assumption ensures the existence of a controller that achieves desired decay rate of the closed-loop state to the origin and the Lyapunov stability constraints in MPC ensure closed-loop stability and closed-loop state convergence to the origin at a rate that is as fast or faster than the one achieved by a Lyapunov-based controller. If the process steady-state of operation is open-loop stable, the closed-loop stability region may be bigger given the limitation imposed in the control actions by the control actuator constraints.*

5.2.2 Long Short-Term Memory Neural Network

It has been demonstrated in numerous examples in the literature that recurrent neural network (RNN) models are capable of modeling dynamic behaviors of time-series data and are an effective method to represent nonlinear processes [40, 57, 102]. With the use of feedback loops in the network, RNNs store outputs derived from past inputs, and use these previous output information together with current inputs to obtain a more accurate prediction of the current output. With sufficient number of neurons in the RNN model, it can be shown that RNNs are capable of approximating any nonlinear dynamic systems on compact subsets of the state-space for finite time based on the universal approximation theorem [64, 99].

There are many types and variations of recurrent neural networks suited for different purposes. Classic RNNs have neurons which pass on historical information across time, and they are prevalent in the fields of natural language processing and speech recognition [28]. However, they face difficulty of accessing information from a long time ago due to the vanishing/exploding gradient phenomena, and they cannot consider any future input for the prediction of the current state since the feedback loops are in the forward direction in time only. To this end, many variations of RNNs have been proposed. For example, in order to address the vanishing/exploding gradient problem, specific gates are used in each repeating network unit to assess and regulate the information transferred down the network. More specifically, Long Short-Term Memory (LSTM) networks use three gates (the forget gate, the input gate, and the output gate) to protect and control the memory cell state, thus information will be stored and remembered for long periods of time, making LSTM networks well suited for data sequences that exhibit long time lags between relevant data points [50,93]. Gated Recurrent Unit (GRU), which is a variation of the LSTM unit, combines the gating functions of the input gate and the forget gate in an LSTM unit, thereby significantly

reducing the training and execution time [33]. On the other hand, Bidirectional RNN (BRNN) was proposed in [94] where a hidden layer with forward connections in time and a hidden layer with backward connections in time were used together such that both past outputs and future outputs were used along with current inputs to predict the current output. Moreover, Deep RNN (DRNN) stacks multiple hidden recurrent layers on top of each other, where each hidden state is continuously passed to both the next time step of the current layer, as well as the current time step of the next layer [93]. LSTM networks could also adopt the configurations of BRNN and DRNN for more complex applications. For each subsystem j, $j = 1, ..., N_{sys}$, we develop a classic LSTM model to approximate the continuous-time nonlinear processes of each subsystem of Eq. 5.2.2. Combining N_{sys} such LSTM network models, the nonlinear process of Eq. 8.1 can be represented as follows:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) = [\dot{\hat{x}}_1^T \cdots \dot{\hat{x}}_{N_{sys}}^T]^T \tag{5.2.6}$$

and the LSTM network model for each subsystem j, modeling the nonlinear dynamics of Eq. 5.2.2, is represented in the following form:

$$\dot{\hat{x}}_j = F_{nn_j}(\hat{x}, u_j) := \lambda_j A_j \hat{x} + \Theta_j^T y_j, \ j = 1, ..., N_{sys} \tag{5.2.7}$$

where $\hat{x} \in \mathbf{R}^n$ is the predicted state vector for the overall system, $\hat{x}_j \in \mathbf{R}^{n_j}$ is the predicted state vector given by the LSTM model for subsystem j, and $u_j \in \mathbf{R}^{m_j}$ is the manipulated input vector for subsystem j. $y_j = [y_1 \cdots y_{n_j+m_j+1}]^T = [H(\hat{x}_1) \cdots H(\hat{x}_{n_j}) \ \ u_1 \cdots u_{m_j} \ \ 1]^T \in \mathbf{R}^{n_j+m_j+1}$ is a vector of both the network states $\hat{x}_j$ and the inputs u_j, where $H(\cdot)$ represents a series of interacting nonlinear activation functions in each LSTM unit. $\lambda_j \in \mathbf{R}^{n_j \times n}$ is a coefficient matrix representing the interacting dynamic behavior between the states of the overall system x and the states of the subsystem j, x_j. A_j is a diagonal coefficient matrix, where $A_j = diag\{-\alpha_{j_1} \cdots -\alpha_{j_n}\} \in \mathbf{R}^{n \times n}$, and $\Theta_j = [\theta_1 \cdots \theta_{n_j+m_j+1}] \in \mathbf{R}^{(n_j+m_j+1) \times n_j}$ with $\theta_i = \beta_{j_i}[\omega_{i1} \cdots \omega_{i(n_j+m_j)} \ \ b_{j_i}]$, $i = 1, ..., n_j$. α_{j_i} and β_{j_i} are constants, and ω_{ik} is the weight connecting the k^{th} input to the i^{th} neuron where $i = 1, ..., n_j$ and $k = 1, ..., (n_j + m_j)$, and b_{j_i} is the bias term for $i = 1, ..., n_j$. We assume that α_{j_i} are positive constants such that each state in the vector $\hat{x}_j$ is bounded-input bounded-state stable. We use x_j to represent the state of the actual nonlinear subsystem of Eq. 5.2.2 and use $\hat{x}_j$ for the state of the subsystem j modeled by the LSTM network of Eq. 5.2.7. The basic architecture of an LSTM network is illustrated in Fig. 5.2.1.

The matrix of input sequences to the LSTM network is denoted by $m \in \mathbf{R}^{(n+m_j) \times T}$ containing the measured states $x \in \mathbf{R}^n$ at each model predictive controller (MPC) sampling period Δ and the

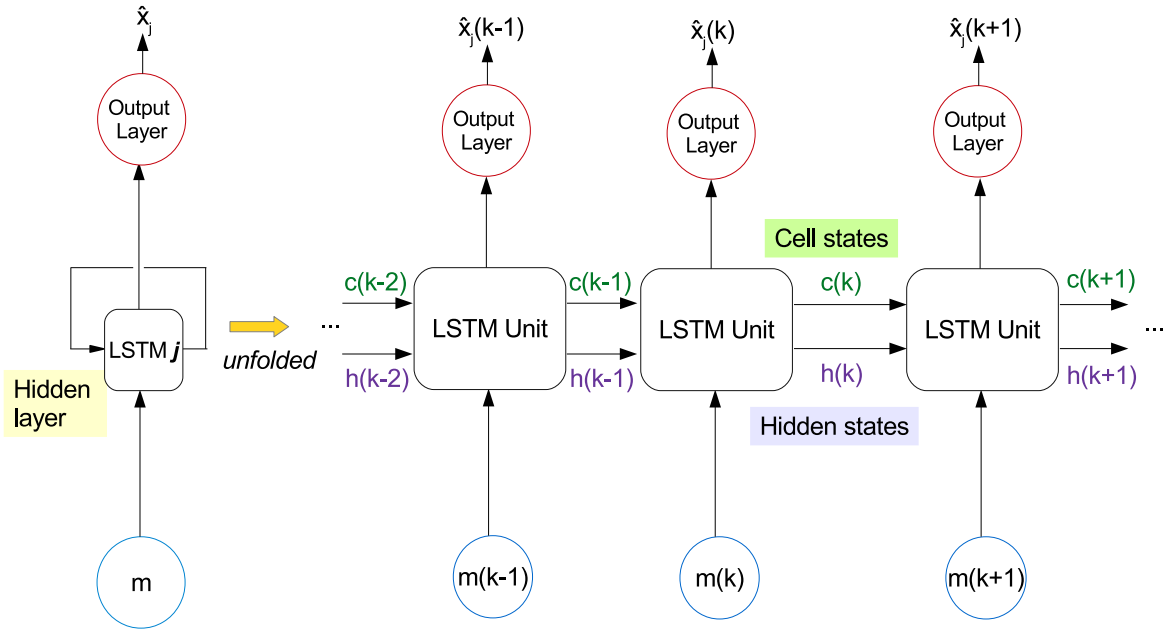


Figure 5.2.1: A long short-term memory (LSTM) recurrent neural network for subsystem j and the series of its unfolded structure, where m is the input vector consisting of the state measurement x at each MPC sampling period and the control action u_j to be optimized over the next sampling period, c is the cell state vector, h is the hidden state vector, and $\hat{x}_j$ is the output vector.

manipulated inputs $u_j \in \mathbf{R}^{m_j}$ to be optimized over the next sampling period Δ, both with a sequence length of T. The matrix of the network output sequences is the predicted states for subsystem j denoted by $\hat{x}_j \in \mathbf{R}^{n_j \times T}$. From each repeating LSTM unit, the network state that is passed onto the next LSTM unit in the unfolded sequence is the hidden state, denoted by $h(1),...,h(T)$, where the number of internal states T (i.e., the number of repeating LSTM units) corresponds to the length of the input sequence. The LSTM network calculates the predicted output sequence $\hat{x}_j$ from the

input sequence m by computing the following calculations iteratively from $k = 1$ to $k = T$:

$$i(k) = \sigma(\omega_i^m m(k) + \omega_i^h h(k-1) + b_i) \tag{5.2.8a}$$

$$f(k) = \sigma(\omega_f^m m(k) + \omega_f^h h(k-1) + b_f) \tag{5.2.8b}$$

$$c(k) = f(k)c(k-1) + i(k)tanh(\omega_c^m m(k) + \omega_c^h h(k-1) + b_c) \tag{5.2.8c}$$

$$o(k) = \sigma(\omega_o^m m(k) + \omega_o^h h(k-1) + b_o) \tag{5.2.8d}$$

$$h(k) = o(k)tanh(c(k)) \tag{5.2.8e}$$

$$\hat{x}_j(k) = \omega_y h(k) + b_y \tag{5.2.8f}$$

where $m(k)$ denotes the k^{th} element in the input sequence, $h(k)$ and $\hat{x}_j(k)$ are the internal state and the output computed by the k^{th} LSTM unit in the unfolded sequence, respectively. $\sigma(\cdot)$ is the sigmoid activation function, $tanh(\cdot)$ is the hyperbolic tangent activation function. ω_y and b_y denote the weight matrix and bias vector for the output, respectively. The outputs from the input gate, the forget gate, and the output gate are represented by $i(k)$, $f(k)$, $o(k)$, respectively; ω_i^m, ω_i^h, ω_f^m, ω_f^h, ω_o^m, ω_o^h are the weight matrices for the input vector m and the hidden state vector h within the input gate, the forget gate, and the output gate respectively, and b_i, b_f, b_o represent the bias vectors within each of the three gates, respectively. $c(k)$ is the cell state which stores memory and passes information down the LSTM units, with ω_c^m, ω_c^h and b_c representing the weight matrices for the input and hidden state vectors, and the bias vector, respectively. The series of interacting nonlinear functions carried out in each LSTM unit, outlined in Eq. 5.2.8, can be represented by $H(\cdot)$. The internal structure of an LSTM unit showing the gating functions is shown in Fig. 5.2.2.

Remark 5.2.3. *Since the class of nonlinear systems we consider in this work is continuous-time, the stability analysis of the data-based LSTM model is also conducted using the continuous-time representation. However, given the inherent internal structure of the LSTM model and that the LSTM model is constructed based on data arranged in sequences of uniform intervals, the LSTM model works as a discrete-time model with a uniform sampling time. While the LSTM model and the first-principles model of the process are both represented as a continuous-time models, the computation of the LSTM model occurs at discrete time instants, similar to how the first-principles continuous-time model in the form of ordinary differential equations can be numerically integrated via explicit Euler method, but maybe with a different integration time step.*

The continuous-time nonlinear system of Eq. 8.1 is operated under the proposed decentralized Lyapunov-based model predictive control (LMPC) system in a sample-and-hold manner, where

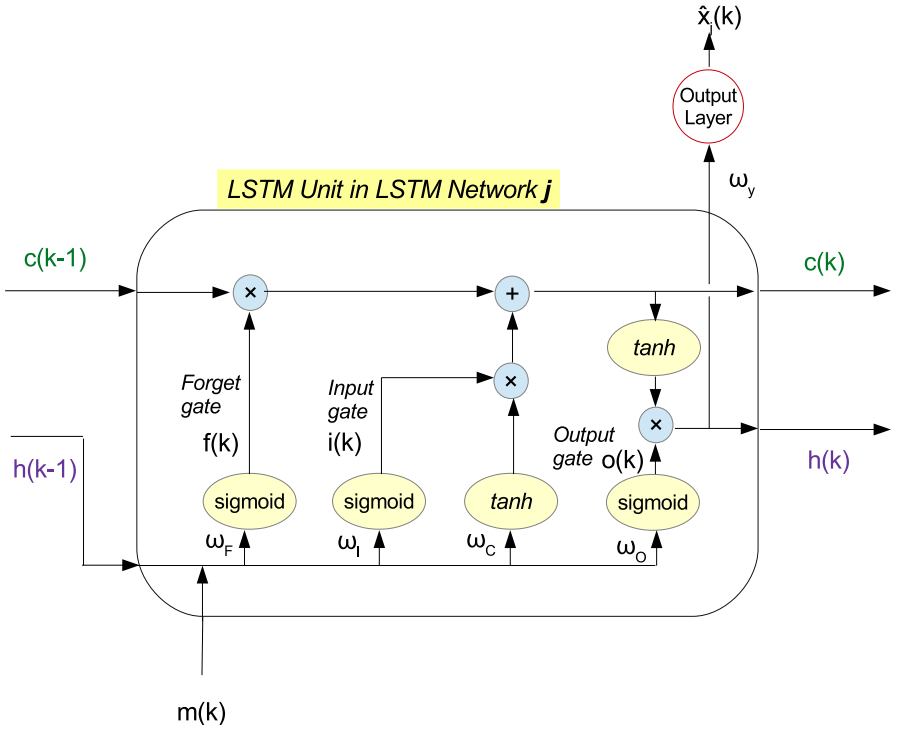


Figure 5.2.2: The internal structure of an LSTM unit inside the LSTM network j where the past cell state vector $c(k-1)$, past hidden state vector $h(k-1)$, and the current input vector $m(k)$ are used to obtain $c(k)$, $h(k)$, and the network output vector $\hat{x}_j(k)$ for subsystem j via the input gate, the forget gate, the output gate, and an output layer.

the feedback measurement of the closed-loop state x for the nonlinear system is received by the designated local controller, LMPC j, every sampling period Δ. Using numerical integration methods such as the explicit Euler method, we can obtain the state information of the simulated nonlinear process with an integration time step of h_c. We develop the LSTM network model for each subsystem j such that it can be used as the prediction model in the decentralized LMPC; to this end, the total prediction period of the LSTM network model is also set to be Δ (i.e., the last output state vector is obtained at the end of every Δ), and the time interval between two consecutive internal states h in the LSTM network for subsystem j can be chosen as a multiple q_{nn} of the integration time step h_c, with the minimum time interval being the integration time step with $q_{nn} = 1$. Depending on the choice of q_{nn}, the number of internal states T will follow $T = \frac{\Delta}{q_{nn} \cdot h_c}$. Therefore, given that the input sequence is fed to the LSTM network at $t = t_k$, the LSTM network provides a sequence of T future predicted states following Eq. 5.2.8, with the last network output vector corresponding to the predicted state $\hat{x}_j(t)$ at $t = t_k + \Delta$. Since input sequences with a length of T are needed to produce T number of internal states, the input time-series samples

will be of a fixed length of T. Fig. 5.2.3 illustrates the different time steps used in this work, i.e., the integration time step for the numerical simulation (h_c), the time interval between the internal states in the developed LSTM network models ($q_{nn} \times h_c$), and the sampling period in the model predictive control algorithm (Δ).

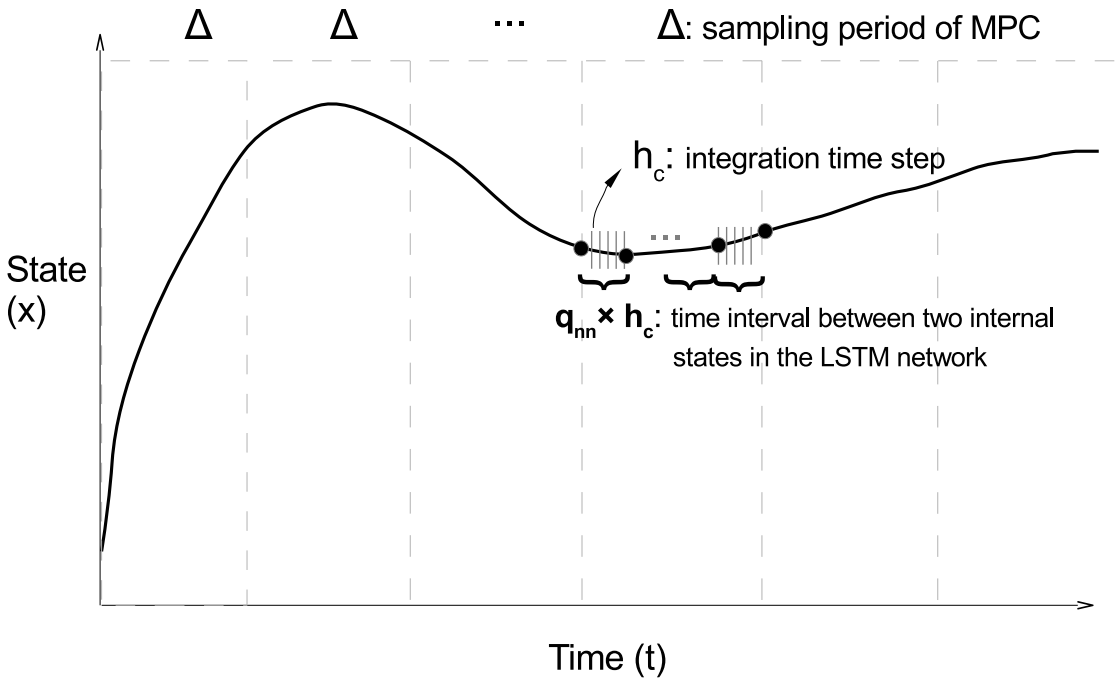


Figure 5.2.3: An illustration showing the integration time step h_c used in the numerical integration of the process states x, the time interval between internal states in the LSTM network $q_{nn} \times h_c$, and the sampling period for the model predictive controller.

To collect training data for developing an LSTM network for each subsystem j, we first discretize the targeted region in state-space with sufficiently small intervals; then, open-loop simulations are carried out where the first-principles model of Eq. 5.2.2 is numerically integrated at a fixed integration time step of h_c, and these simulations are conducted for various initial conditions $x_0 \in \Omega_\rho$ under various input sequences $u_j \in U_j,\ j = 1,...,N_{sys}$ implemented in a sample-and-hold manner with a sampling period of Δ. We obtain enough samples of trajectories for finite time to sweep over all the values that (x,u) can take in the targeted region in state-space. Then, we extract the state measurements every $q_{nn} \times h_c$ interval to use as the target predicted internal states during training. As there are T such internal states within each controller sampling period $\Delta = T \times q_{nn} \times h_c$, the last internal state corresponds to the output predicted state at the end of every Δ. The time interval between two data points in the sample of time-series sequence is $q_{nn} \times h_c$, and

it corresponds to the time interval between two consecutive LSTM units in the LSTM network. We separate the dataset into the corresponding input and output vectors for each subsystem j, i.e., the input vector includes data on all states of the overall system x and the distinct set of manipulated inputs for the subsystem, u_j, and the output vector includes the states of the subsystem, x_j. The generated dataset is then divided into training and validation sets.

Remark 5.2.4. *We could use all the numerically integrated state values every h_c step (i.e., $q_{nn} = 1$) as the internal states in the LSTM network, however, this would increase the computational load associated with training and implementing the model. Therefore, we choose a q_{nn} value such that there are sufficient state values within each sampling period of the controller Δ to capture the dynamic state evolution while not overburdening the computational effort. While the control actions are computed every sampling period Δ (i.e., the model predictive controllers are executed once every Δ), the objective function of the optimization problem of the model predictive controller includes the integral of all the predicted states over the prediction horizon of $N \times \Delta$, which includes all the predicted internal states intervaled at $q_{nn} \times h_c$ given by the LSTM model.*

With sufficient data in the training dataset, the LSTM network is developed using an application program interface which contains open-source neural network libraries, e.g., *Keras*. The optimal parameter matrix of the LSTM network, Γ^*, which includes the network parameters ω_i, ω_f, ω_c, ω_o, ω_y, b_i, b_f, b_c, b_o, b_y, is optimized by minimizing the mean absolute percentage error between $x_j(t)$ and $\hat{x}_j(t)$ for each subsystem j. The minimization of the state error is carried out using the adaptive moment estimation optimizer, i.e., *Adam* in Keras, in which the gradient of the error cost function is calculated using back-propagation. A constraint on the modeling error ν_j for each LSTM network j is imposed during training, $|\nu_j| = |F_j(x,u_j,0) - F_{nn_j}(x,u_j)| \leq \gamma_j |x_j|$, with $\gamma_j > 0$, such that the modeling error for the overall nonlinear system is sufficiently small, i.e., $|\nu| = \sum_{j=1}^{N_{sys}} |\nu_j| \leq \gamma |x|,\ \gamma > 0$. With a sufficiently small modeling error, the trained LSTM network model can adequately represent the nonlinear process of subsystem j and can be used in the proposed model predictive controller to stabilize the actual nonlinear process of Eq. 8.1 at its steady-state with guaranteed stability properties.

The modeling error can be numerically approximated using the forward finite difference method. Note that the time interval $q_{nn} \cdot h_c$ between two LSTM memory units is given by the time interval between two consecutive time-series data points in the training data sequences. Since the LSTM network provides a predicted state at each time interval $q_{nn} \cdot h_c$ calculated by each LSTM memory unit, similarly to how we can use numerical integration methods to obtain the state at the same time instance using the continuous-time model of Eq. 8.1, we can use the predicted states

from the LSTM network to compare with the predicted states from the nonlinear model of Eq. 8.1 to assess the modeling error. The time derivative of the LSTM predicted state $\hat{x}_j(t)$ at $t = t_k$ can be approximated by $\dot{\hat{x}}_j(t_k) = F_{nn_j}(x(t_k), u_j) \approx \frac{\hat{x}_j(t_k+q_{nn}\cdot h_c)-\hat{x}_j(t_k)}{q_{nn}\cdot h_c}$. The time derivative of the actual state $x_j(t)$ at $t = t_k$ can be approximated by $\dot{x}_j(t_k) = F_j(x(t_k), u_j, 0) \approx \frac{x_j(t_k+q_{nn}\cdot h_c)-x_j(t_k)}{q_{nn}\cdot h_c}$. At time $t = t_k$, $\hat{x}_j(t_k) = x_j(t_k)$, the constraint $|\nu_j| \leq \gamma|x_j|$ for the LSTM network of subsystem j can be written as follows:

$$|\nu_j| = |F_j(x(t_k), u_j, 0) - F_{nn_j}(x(t_k), u_j)| \tag{5.2.9a}$$

$$\approx |\frac{x_j(t_k + q_{nn}\cdot h_c) - \hat{x}_j(t_k + q_{nn}\cdot h_c)}{q_{nn}\cdot h_c}| \tag{5.2.9b}$$

$$\leq \gamma_j|x_j(t_k)| \tag{5.2.9c}$$

which will be satisfied if $|\frac{x_j(t_k+q_{nn}\cdot h_c)-\hat{x}_j(t_k+q_{nn}\cdot h_c)}{x_j(t_k)}| \leq \gamma_j \cdot q_{nn} \cdot h_c$. Therefore, we can use the mean absolute percentage error between the predicted states $\hat{x}_j$ and the targeted states x_j in the training data to assess the modeling error of the LSTM model for subsystem j. To avoid over-fitting of the LSTM models, the training process is terminated once the modeling error falls below the desired threshold and the error on the validation set stops decreasing. The modeling error of the overall system of Eq. 5.2.6 can be represented as a sum of the modeling errors of the individual LSTM networks, i.e., $|\nu| = \sum_{j=1}^{N_{sys}} |\nu_j|$.

Remark 5.2.5. *For the theoretical analysis and the construction of the LSTM model in this work, we assume that a first-principles model for the nominal process is available in the form of ordinary differential equations. In the case that such models are not available, the modeling error approximation in Eq. 5.2.9 can be calculated based on real plant data.*

5.2.2.1 Lyapunov-based Control using LSTM Models

After obtaining a sufficiently small modeling error between the trained LSTM network and the actual nonlinear model of Eq. 8.1 for subsystem j, we assume that there exists a set of stabilizing feedback controllers $u = \mathbf{\Phi}_{nn}(x) \in U$, where $\mathbf{\Phi}_{nn}(x) = [\Phi_{nn_1}(x)^T \cdots \Phi_{nn_{N_{sys}}}(x)^T]^T$, that can render the origin of the LSTM model of Eq. 5.2.6 exponentially stable in the sense that there exists a $\mathscr{C}^1$ control Lyapunov function $\hat{V}(x)$ such that the following inequalities hold for all $x \in \mathbf{R}^n$ in an open neighborhood around the origin $\hat{D}$:

$$\hat{c}_1|x|^2 \leq \hat{V}(x) \leq \hat{c}_2|x|^2, \tag{5.2.10a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, \mathbf{\Phi}_{nn}(x)) \leq -\hat{c}_3 |x|^2, \tag{5.2.10b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{c}_4 |x| \tag{5.2.10c}$$

where $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$, $\hat{c}_4$ are positive constants, and $F_{nn}(x,u)$ is represented by the system of LSTM models Eq. 5.2.6. While there are many forms that the Lyapunov function $\hat{V}$ can take, it is a function that captures all the states of the overall system of Eq. 5.2.6. Similar to the Lyapunov function $V(x)$ for the nonlinear system of Eq. 8.1 introduced in Section 5.2.1.3, one example of $\hat{V}$ is a linear combination of the Lyapunov functions of the states of individual subsystems, i.e., $\hat{V}(x) = \sum_{j=1}^{N_{sys}} \hat{V}_j(x_j)$. Here, we assume that $\hat{V}_j(\cdot)$ is a function of x_j only.

Similar to the characterization method of the closed-loop stability region Ω_ρ for the nonlinear subsystem of Eq. 8.1, we first search the entire state-space to characterize a set of states $\hat{D}$ where the following inequality holds: $\dot{\hat{V}}(x) = \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x,u) \leq -c_3 |x|^2$, $u = \mathbf{\Phi}_{nn}(x) \in U$. The closed-loop stability region for the LSTM model of Eq. 5.2.6 is defined as a level set of Lyapunov function inside $\hat{D}$: $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. Starting from $\Omega_{\hat{\rho}}$, the origin of the LSTM model of Eq. 5.2.6 can be rendered exponentially stable under the set of controllers $u_j = \Phi_{nn_j}(x) \in U_j$, for $j = 1, ..., N_{sys}$. The assumptions of Eq. 5.2.10 are the stabilizability conditions for the LSTM model of Eq. 5.2.6, and they are analogous to the ones of Eq. 7.5 for the general class of nonlinear systems of Eq. 8.1 since the LSTM model of Eq. 5.2.7 for each subsystem can be written in the form of Eq. 5.2.2 (i.e., for each subsystem, $\dot{\hat{x}}_j = \hat{f}_j(\hat{x}) + \hat{g}_j(\hat{x}) u_j$, where $\hat{f}_j(\cdot)$ and $\hat{g}_j(\cdot)$ are obtained from coefficient matrices λ_j, A_j and Θ_j in Eq. 5.2.7). However, due to the complexity of the LSTM structure and the interactions of the nonlinear activation functions, $\hat{f}_j$ and $\hat{g}_j$ may be hard to compute explicitly. For computational convenience, we will numerically approximate $\hat{f}_j$ and $\hat{g}_j$ using the forward finite difference method similar to Eq. 5.2.9. At $t = t_k$, we compute two sets of predicted states $\hat{x}_j(t)$ at $t = t_k + q_{nn} \cdot h_c$ using the LSTM network under the sample-and-hold implementation of two sets of inputs respectively: first, we use $u_{j_i} = 0$, $\forall\, i = 1, ..., m_j$, applied in a sample-and-hold manner for the time interval $[t_k, t_k + q_{nn} \cdot h_c)$ to obtain the predicted state $\hat{x}_j(t)$ at $t = t_k + q_{nn} \cdot h_c$ under zero inputs, denoted as $\int_{t_k}^{t_k + q_{nn} \cdot h_c} F_{nn_j}(x, 0) dt$; then, we use $u_j(t_k) = [0 \cdots u_{j_i}(t_k) \cdots 0] \in \mathbf{R}^{m_j}$, where $u_{j_i}^{min} \leq u_{j_i}(t_k) \leq u_{j_i}^{max}$, $u_{j_i}(t_k) \neq 0$, applied in a sample-and-hold manner for the interval $[t_k, t_k + q_{nn} \cdot h_c)$ to obtain the predicted state $\hat{x}_j(t)$ at $t = t_k + q_{nn} \cdot h_c$ under a non-zero input $u_{j_i}(t_k)$, denoted as $\int_{t_k}^{t_k + q_{nn} \cdot h_c} F_{nn_j}(x, u_j(t_k)) dt$. Therefore, given

the state measurement $x(t_k)$ at $t = t_k$, $\hat{f}_j$ and $\hat{g}_{j_i}$ can be numerically approximated as follows:

$$\hat{f}_j(t_k) = \frac{\int_{t_k}^{t_k+q_{nn}\cdot h_c} F_{nn_j}(x,0)dt - x(t_k)}{q_{nn}\cdot h_c} \tag{5.2.11a}$$

$$\hat{g}_{j_i}(t_k) = \frac{\int_{t_k}^{t_k+q_{nn}\cdot h_c} F_{nn_j}(x,u_j(t_k))dt - \int_{t_k}^{t_k+q_{nn}\cdot h_c} F_{nn_j}(x,0)dt}{q_{nn}\cdot h_c\cdot u_{j_i}(t_k)} \tag{5.2.11b}$$

Here, $q_{nn}\cdot h_c$ is the time interval between two internal states within the LSTM network; thus, the predicted state $\hat{x}(t_k + q_{nn}\cdot h_c)$ represented by the integral $\int_{t_k}^{t_k+q_{nn}\cdot h_c} F_{nn_j}(x,\cdot)dt$ in Eq. 5.2.11, is the first internal state of the LSTM network. After obtaining $\hat{f}_j$ and $\hat{g}_{j_i}$, $i = 1,...,m_j$, the stabilizing control law $\Phi_{nn_j}(x_j)$ can be computed similarly as in Eq. 7.7 and repeated for all subsystems $j = 1,...,N_{sys}$, with $\hat{f}_j$ and $\hat{g}_{j_i}$ replacing f_j and g_{j_i} respectively. Subsequently, $\dot{\hat{V}}$ can also be computed using the approximated values of $\hat{f}_j$ and $\hat{g}_{j_i}$.

Since the dataset for developing the LSTM model for subsystem j is generated from open-loop simulations for $x \in \Omega_\rho$, $u_j \in U_j$, the closed-loop stability region of the LSTM system is a subset of the closed-loop stability region of the actual nonlinear system, i.e., $\Omega_{\hat{\rho}} \subseteq \Omega_\rho$. Moreover, there exist positive constants M_{nn} and L_{nn} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$, $u \in U$:

$$|F_{nn}(x,u)| \leq M_{nn} \tag{5.2.12a}$$

$$\left|\frac{\partial \hat{V}(x)}{\partial x}F_{nn}(x,u) - \frac{\partial \hat{V}(x')}{\partial x}F_{nn}(x',u)\right| \leq L_{nn}|x-x'| \tag{5.2.12b}$$

Remark 5.2.6. *Depending on how the overall system of Eq. 8.1 is partitioned, the extent of the coupling effect between the different subsystems j may be different. The Lyapunov-based control law $\Phi_{nn_j}(x(t))$ for each subsystem j are designed accounting for the coupling effect between the partitioned subsystems, and $\mathbf{\Phi}_{nn}(x)$, which is the vector containing all subsystem control laws, can render the origin of the LSTM model of Eq. 5.2.6 exponentially stable for all $\hat{x}_0 = x_0 \in \Omega_{\hat{\rho}}$.*

5.2.3 Decentralized LMPC using LSTM Models

When the optimization problem of a centralized MPC is too complex to solve within a reasonable time period (i.e., the sampling period), the control problem may be decoupled into smaller local optimization problems that are solved in separate processors/controllers to achieve improved computational efficiency. In a decentralized LMPC system, there is no communication between

the different local controllers, therefore each controller does not have any knowledge on the control actions calculated by the other controllers. A schematic diagram showing the decentralized LMPC architecture of a process consisting of N_{sys} subsystems is shown in Fig. 5.2.4.

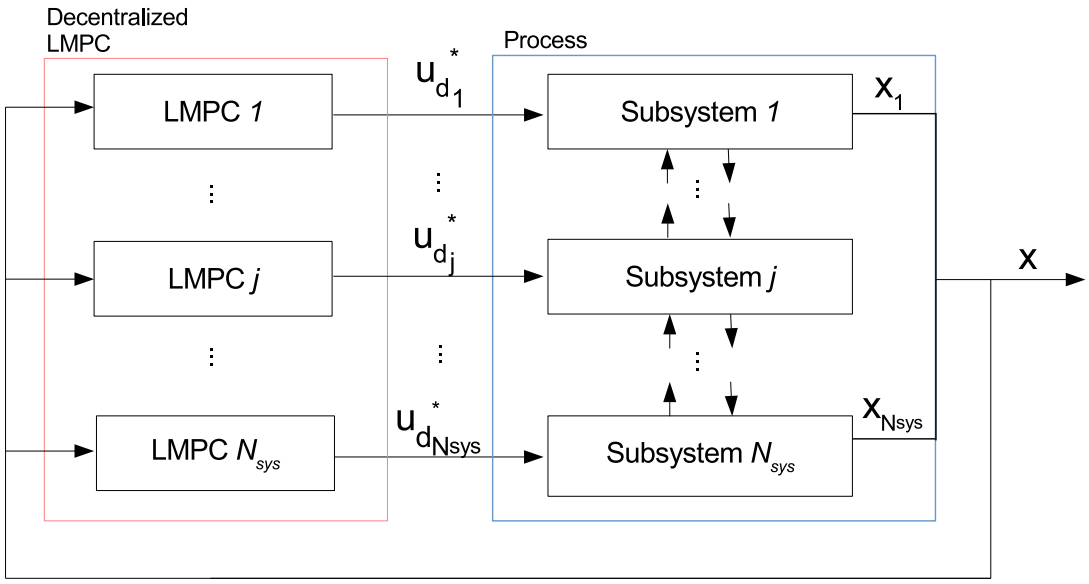


Figure 5.2.4: A schematic showing the flow of information of N_{sys} number of decentralized LMPCs with the overall process subdivided into N_{sys} number of subsystems.

Consider $j = 1, ..., N_{sys}$ subsystems, each with distinct sets of states x_j and manipulated inputs u_j. We design separate LMPCs, each designated for one subsystem to compute the respective control actions. The trajectories of control actions computed by LMPC j are denoted by u_{d_j}. Each decentralized LMPC may receive full-state feedback measurements, but they only have information on the dynamic behavior of their respective subsystem. Therefore, separate LSTM models are developed, one for each subsystem j. For instance, LMPC j uses the LSTM network model developed for subsystem j as its prediction model to predict the states of subsystem j (i.e., x_j), and calculates the control actions u_{d_j} which are applied to the corresponding control actuators in subsystem j. To inherit the stability properties of Φ_{nn_j}, $j = 1, ..., N_{sys}$, the optimized control actions u_{d_j} must satisfy a Lyapunov-based contractive constraint that guarantees a minimum decrease rate of the Lyapunov function $\hat{V}$. The mathematical formulation of each decentralized

LMPC j, $j = 1, ..., N_{sys}$, is shown as follows:

$$\mathscr{J}_j = \min_{u_{d_j} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}_j(t), u_{d_j}(t)) dt \tag{5.2.13a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}_j(t) = F_{nn_j}(\bar{x}(t), u_{d_j}(t)) \tag{5.2.13b}$$

$$\bar{x}(t) = [x_1(t_k)^T \cdots x_{j-1}(t_k)^T \;\; \tilde{x}_j(t)^T \;\; x_{j+1}(t_k)^T \cdots x_{N_{sys}}(t_k)^T]^T \tag{5.2.13c}$$

$$u_{d_j}(t) \in U_j, \; \forall \, t \in [t_k, t_{k+N}) \tag{5.2.13d}$$

$$\tilde{x}_j(t_k) = x_j(t_k) \tag{5.2.13e}$$

$$\frac{\partial \hat{V}(x(t_k))}{\partial x_j}(F_{nn_j}(x(t_k), u_{d_j}(t_k))) \leq \frac{\partial \hat{V}(x(t_k))}{\partial x_j}(F_{nn_j}(x(t_k), \Phi_{nn_j}(x(t_k)))),$$

$$\text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{5.2.13f}$$

$$\hat{V}(\bar{x}(t)) \leq \rho_{nn}, \; \forall \, t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{5.2.13g}$$

where $\tilde{x}_j$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon for subsystem j. The optimal input trajectory computed by this LMPC j is denoted by $u^*_{d_j}(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*_{d_j}(t_k)$ is sent by LMPC j to its control actuators to be applied over the next sampling period. In the optimization problem of Eq. 5.2.13, the objective function of Eq. 5.2.13a is the integral of $L(\tilde{x}_j(t), u_{d_j}(t))$ over the prediction horizon. Note that $L(x_j, u_j)$ is typically in a quadratic form, i.e., $L(x_j, u_j) = x_j^T Q_j x_j + u_j^T R_j u_j$, where Q_j and R_j are the weighting matrices of the states and the inputs of subsystem j respectively, such that each subsystem will be steered towards the origin by minimizing the objective function. The constraint of Eq. 5.2.13b is the LSTM model of Eq. 5.2.7 that is used to predict the states of the closed-loop subsystem j using $\bar{x}(t)$ and $u_{d_j}(t)$ as the input vector to the LSTM model. $\bar{x}(t)$ is a vector containing the predicted states of subsystem j, $\tilde{x}_j(t)$, and the measured states of all other subsystems at $t = t_k$, $x_i(t_k)$, $\forall \, i = 1, ..., N_{sys}$, and $i \neq j$. Eq. 5.2.13d defines the input constraints on u_{d_j} applied over the entire prediction horizon. Eq. 5.2.13e defines the initial condition $\tilde{x}_j(t_k)$ of Eq. 5.2.13b, which is the state measurement $x_j(t)$ at $t = t_k$. The constraint of Eq. 5.2.13f forces the closed-loop state to move towards the origin at a minimum rate characterized by the Lyapunov function $\hat{V}$ and the Lyapunov-based control law $\Phi_{nn_j}(x(t_k))$ if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the LSTM model of Eq. 5.2.13b will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon.

Since the decentralized LMPCs examine different optimization problems specific to their respective subsystems and they are computed in separate processors in parallel, the computation time for solving one iteration of the decentralized LMPC design in one sampling period (assuming that the feedback measurements are available to both controllers at synchronous intervals) will be the maximum time out of the two LMPCs.

Remark 5.2.7. *The computation time to calculate the solutions to any MPC optimization problem(s) must be less than the sampling time of the nonlinear process. One of the main advantages of decentralized MPC systems is the reduced computational complexity of the optimization problems where a large-scale system is broken down into local subsystems, and thus, reduced total computational time is needed compared to solving the optimization problem in a centralized MPC system.*

Remark 5.2.8. *In the case that full-state feedback is not available to the controllers, decentralized estimators can be used to reconstruct state information. Provided that the state estimation error is bounded and is rendered sufficiently small in a short time, it can be considered as part of the bounded process disturbance, and the closed-loop stability analysis of the estimator-based control system may be established.*

5.2.3.1 Sample-and-hold implementation of Decentralized LMPC

Once the optimization problems of all subsystems $j = 1,...,N_{sys}$ are solved, the optimal control actions of the proposed decentralized LMPC design are defined as follows:

$$u_j(t) = u_{d_j}^*(t_k),\ j = 1,...,N_{sys},\ \forall\, t \in [t_k, t_{k+1}) \tag{5.2.14}$$

The stability of the overall system of Eq. 8.1 is ensured by the inclusion of a contractive constraint Eq. 5.2.13f in the formulation of each of the decentralized LMPCs, which ascertains that the optimized control actions of Eq. 7.19 guarantee a minimum decrease rate of the Lyapunov function $\hat{V}$ characterized by the Lyapunov-based control law using LSTM models $\Phi_{nn_j}(x(t))$.

We have previously proven the stability properties of a centralized MPC using RNN models where the Lyapunov-based control actions are applied in a sample-and-hold manner, given that the process disturbances and the modeling error are sufficiently small; detailed proof can be found in [127]. More specifically, given that the stabilizability conditions of Eq. 5.2.10 are satisfied and the modeling error $|\nu| \leq \gamma|x| \leq \nu_m$, where γ is chosen to satisfy $\gamma < \hat{c}_3/\hat{c}_4$ ($\hat{c}_3$ and $\hat{c}_4$ are the

positive constants in Eq. 5.2.10), the closed-loop state $x(t)$ of the actual process of Eq. 8.1 and the closed-loop state $\hat{x}(t)$ of the LSTM system of Eq. 5.2.6 are bounded in the stability region and driven to a small neighborhood around the origin under the set of Lyapunov-based controllers $\mathbf{\Phi}_{nn}(x) \in U$ implemented in a sample-and-hold manner. This is established in the following proposition:

Proposition 5.2.1. *Consider the system of Eq. 8.1 under the Lyapunov-based controllers* $u_j = \Phi_{nn_j}(\hat{x}) \in U_j$, $j = 1,...,N_{sys}$, *which meet the conditions of Eq. 5.2.10, and are designed to stabilize the LSTM system of Eq. 5.2.6 and implemented in a sample-and-hold fashion, i.e.,* $u_j(t) = \Phi_{nn_j}(\hat{x}(t_k))$, $j = 1,...,N_{sys}$, $\forall t \in [t_k, t_{k+1})$, *where* $t_{k+1} := t_k + \Delta$. *The LSTM system is developed with an overall modeling error* $|\nu| \leq \gamma|x| \leq \nu_m$, *where* $\gamma < \hat{c}_3/\hat{c}_4$. *Let* ε_s, $\varepsilon_w > 0$, $\Delta > 0$, $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma > 0$, *and* $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ *satisfy*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}M_{nn}\Delta \leq -\varepsilon_s \tag{5.2.15a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}M\Delta + L_w^{'}w_m \leq -\varepsilon_w \tag{5.2.15b}$$

and

$$\rho_{nn} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{5.2.16a}$$

$$\rho_{min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{5.2.16b}$$

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

$$\hat{V}(\hat{x}(t)) \leq \hat{V}(\hat{x}(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{5.2.17a}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{5.2.17b}$$

and if $x_0 \in \Omega_{\hat{\rho}}$, *the state* $\hat{x}(t)$ *of the LSTM modeled system of Eq. 5.2.6 is bounded in* $\Omega_{\hat{\rho}}$ *for all times and ultimately bounded in* $\Omega_{\rho_{nn}}$, *and the state* $x(t)$ *of the nonlinear system of Eq. 8.1 is bounded in* $\Omega_{\hat{\rho}}$ *for all times and ultimately bounded in* $\Omega_{\rho_{min}}$.

Consider each closed-loop subsystem of Eq. 5.2.2 under $u_j = u_{d_j}^*$ in the decentralized LMPC design of Eq. 7.19, which are calculated based on the controllers $\Phi_{nn_j}(x)$, $j = 1,...,N_{sys}$ that collectively satisfy Eq. 5.2.10 for the overall system. The control actions computed by each LMPC will be applied in a sample-and-hold manner to the process. The proof for recursive

feasibility of each of the decentralized LMPCs, and the closed-loop stability of the overall nonlinear process of Eq. 8.1 under the sample-and-hold implementation of the optimal control actions $u_j = u^*_{d_j},\ j = 1,...,N_{sys}$ of Eq. 7.19 will be established in the following theorem.

Theorem 5.2.1. *Consider the nonlinear process of Eq. 8.1 and the LSTM network system of Eq. 5.2.6. In the presence of bounded disturbances $|w(t)| \leq w_m$ and a sufficiently small modeling error $|\nu| \leq \gamma|x| \leq \nu_m$, $\gamma > 0$, there exists a class $\mathscr{K}$ function $f_w(\cdot)$ and a positive constant κ such that $\forall x, \hat{x} \in \Omega_{\hat{\rho}}$ and $w(t) \in W$, $|e(t)| = |x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + \nu_m}{L_x}(e^{L_x t} - 1)$. Let $\Delta > 0$, $\varepsilon_s > 0$, $\varepsilon_w > 0$, $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma > 0$, and $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ satisfy the conditions of Eq. 5.2.15a – Eq. 6.19b, then given any initial state $x_0 \in \Omega_{\hat{\rho}}$, there always exists a feasible solution for the optimization problems of Eq. 5.2.13 for each decentralized LMPC, and it is guaranteed that under the optimized control actions $u_j = u^*_{d_j},\ j = 1,...,N_{sys}$, $x(t) \in \Omega_{\hat{\rho}},\ \forall t \geq 0$, and $x(t)$ ultimately converges to $\Omega_{\rho_{min}}$ for the closed-loop system of Eq. 8.1.*

Proof. This proof consists of three parts. First, we will prove the recursive feasibility of the optimization problem in each decentralized LMPC in *Part* 1. Then, under the optimized control actions of the decentralized LMPC design of Eq. 7.19, we will prove the boundedness and convergence of the closed-loop state of the LSTM network system of Eq. 5.2.6 in a compact set $\Omega_{\rho_{nn}}$ in *Part* 2. And lastly, in *Part* 3, we will prove the boundedness and convergence of the closed-loop state of the nonlinear system of Eq. 8.1 in a compact set $\Omega_{\rho_{min}}$ under the sample-and-hold implementation of the control actions from the decentralized LMPC design of Eq. 7.19.

Part 1: In this part, we prove that the optimization problem in Eq. 5.2.13 of each decentralized LMPC for subsystem j is feasible for all states $x \in \Omega_{\hat{\rho}}$. First, we consider $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$. The input trajectories $u_{d_j}(t) = \Phi_{nn_j}(x(t_k)),\ j = 1,...,N_{sys}$ for $t \in [t_k, t_{k+1})$ are feasible solutions to the optimization problem of Eq. 5.2.13 as the input constraint of Eq. 5.2.13d and the Lyapunov-based contractive constraint of Eq. 5.2.13f are both satisfied. Now we consider $x(t_k) \in \Omega_{\rho_{nn}}$. The input trajectories $u_{d_j}(t_{k+i}) = \Phi_{nn_j}(\tilde{x}(t_{k+i})),\ i = 0,1,...,N-1,\ j = 1,...,N_{sys}$ satisfy the constraints on the inputs in Eq. 5.2.13d and the Lyapunov-based constraint of Eq. 5.2.13g. This is because it has been shown in Proposition 6.3 that the states predicted by the LSTM model of Eq. 5.2.13b under the Lyapunov-based controllers $\Phi_{nn_j}(\tilde{x}), j = 1,...,N_{sys}$ remain inside $\Omega_{\rho_{nn}}$. Therefore, we have proven that for all $x_0 \in \Omega_{\hat{\rho}}$, the optimization problem of each decentralized LMPC can be solved with recursive feasibility.

Part 2: We will now prove that given $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, the state of the closed-loop LSTM system of Eq. 5.2.6 is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately converges to $\Omega_{\rho_{nn}}$ defined

by Eq. 6.19a under the sample-and-hold implementation of the decentralized LMPC design $u_j = u^*_{d_j},\ j = 1,...,N_{sys}$ of Eq. 7.19. First, we consider the case where $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ at $t = t_k$, and the contractive constraint of Eq. 5.2.13f is activated. The following inequality holds for the derivative of the Lyapunov function based on the states predicted by the LSTM model of Eq. 5.2.13b of each decentralized LMPC at $t = t_k$:

$$\frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}_j} F_{nn_j}(\hat{x}(t_k), u^*_{d_j}(t_k)) \leq \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}_j} F_{nn_j}(\hat{x}(t_k), \Phi_{nn_j}(\hat{x}(t_k))),\ \forall j = 1,...,N_{sys} \tag{5.2.18}$$

Since the Lyapunov function for the overall system $\hat{V}(x)$ can be a linear combination of the Lyapunov functions for each subsystem, $\hat{V}(x) = \sum_{j=1}^{N_{sys}} \hat{V}_j(x_j)$, where $\hat{V}_j$ is assumed to be a function of x_j only, Eq. 5.2.18 can be extended to the overall system of Eq. 5.2.6 to give the following inequality:

$$\sum_{j=1}^{N_{sys}} \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}_j} F_{nn_j}(\hat{x}(t_k), u^*_{d_j}(t_k)) \leq \sum_{j=1}^{N_{sys}} \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}_j} F_{nn_j}(\hat{x}(t_k), \Phi_{nn_j}(\hat{x}(t_k))) \tag{5.2.19a}$$

$$\Rightarrow \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u^*_d(t_k)) \leq \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))) \leq -\hat{c}_3 |\hat{x}(t_k)|^2 \tag{5.2.19b}$$

where Eq. 5.2.19b is given by the stabilizability condition of Eq. 5.2.10b. By the definition of ρ_{nn} in Eq. 6.19a, $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ also belongs to the set $\Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. Considering Eq. 5.2.19b, the conditions of Eq. 5.2.10, and the Lipschitz conditions of Eq. 5.2.12, the time derivative of the Lyapunov function along the trajectory of $\hat{x}(t)$ of the LSTM model of Eq. 5.2.6 in $t \in [t_k, t_{k+1})$ is given by:

$$\begin{aligned}
\frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u^*_d(t_k)) =& \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u^*_d(t_k)) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u^*_d(t_k)) \\
&- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u^*_d(t_k)) \\
\leq& -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), u^*_d(t_k)) - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), u^*_d(t_k)) \\
\leq& -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn}|\hat{x}(t) - \hat{x}(t_k)| \\
\leq& -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{nn} M_{nn} \Delta \\
\leq& -\varepsilon_s
\end{aligned} \tag{5.2.20}$$

By integrating the above equation over $t \in [t_k, t_{k+1})$, it is obtained that $V(\hat{x}(t_{k+1})) \leq V(\hat{x}(t_k)) - \varepsilon_s \Delta$, hence $V(\hat{x}(t)) \leq V(\hat{x}(t_k)), \forall t \in [t_k, t_{k+1})$. Thus, for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the state of the closed-loop LSTM system of Eq. 5.2.6 is bounded in the closed-loop stability region $\Omega_{\hat{\rho}}$ for all times and moves towards the origin under the sample-and-hold implementation of $u_j = u_{d_j}^*,\ j = 1, ..., N_{sys}$. Next, we consider when $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$. $\Omega_{\rho_{nn}}$ as defined in Eq. 6.19a ensures that when $\hat{x}(t_k) \in \Omega_{\rho_s}$, under $u \in U$, the closed-loop state $\hat{x}(t)$ of the LSTM model does not leave $\Omega_{\rho_{nn}}$ for all $t \in [t_k, t_{k+1})$. If $\hat{x}(t_{k+1})$ leaves $\Omega_{\rho_{nn}}$, then Eq. 5.2.20 is satisfied at $t = t_{k+1}$, and the control actions of the decentralized LMPC design of Eq. 7.19 will drive $\hat{x}$ towards the origin over the next sampling period. If $\hat{x}(t_k)$ is outside of Ω_{ρ_s} but inside of $\Omega_{\rho_{nn}}$, the constraint of Eq. 5.2.13g will ensure that $\hat{x}(t_{k+i}),\ i = 1, ..., N-1$ will also remain inside $\Omega_{\rho_{nn}}$. This concludes *Part* 2 of the proof showing the convergence of the closed-loop state $\hat{x}$ of the LSTM system of Eq. 5.2.6 to $\Omega_{\rho_{nn}}$ for all $\hat{x}_0 \in \Omega_{\hat{\rho}}$.

Part 3: Now we will prove the boundedness and convergence of the closed-loop state of the actual nonlinear system of Eq. 8.1 under the sample-and-hold implementation of the optimized control actions $u_j = u_{d_j}^*,\ j = 1, ..., N_{sys}$ of the decentralized LMPC design, accounting for bounded overall modeling error $|\nu| \leq \gamma |x|, 0 < \gamma < \hat{c}_3/\hat{c}_4$ and disturbances $|w| \leq w_m$. We first consider the scenario where $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ (which implies that $x(t_k)$ also belongs to the region $\Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$), and the contractive constraint of Eq. 5.2.13f is activated. Based on the Lyapunov conditions of Eq. 5.2.10c and the inequality derived in Eq. 5.2.19b, the time derivative of the Lyapunov function of the nominal system of Eq. 8.1 is as follows:

$$\begin{aligned}
\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_d^*(t_k), 0) =& \frac{\partial \hat{V}(x(t_k))}{\partial x} (F_{nn}(x(t_k), u_d^*(t_k)) + F(x(t_k), u_d^*(t_k), 0) - F_{nn}(x(t_k), u_d^*(t_k))) \\
&\leq -\hat{c}_3 |x(t_k)|^2 + \hat{c}_4 |x(t_k)| (F(x(t_k), u_d^*(t_k), 0) - F_{nn}(x(t_k), u_d^*(t_k))) \\
&\leq -\hat{c}_3 |x(t_k)|^2 + \hat{c}_4 \gamma |x(t_k)|^2 \\
&\leq -\tilde{c}_3 |x(t_k)|^2
\end{aligned} \tag{5.2.21}$$

where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4 \gamma > 0$. Therefore, the closed-loop state of the nominal system of Eq. 8.1 with $w \equiv 0$ converges to the origin under $u_j = u_{d_j}^*,\ j = 1, ..., N_{sys},\ \forall\ x_0 \in \Omega_{\hat{\rho}}$ if the modeling error $|\nu| \leq \gamma |x|$. Considering bounded disturbances and the effect of sample-and-hold implementation, based on Eq 5.2.21, Eq. 5.2.10a, and the Lipschitz condition in Eq. 5.2.5, the following inequality

is obtained for the time-derivative of $\hat{V}(x(t))$ $\forall t \in [t_k, t_{k+1})$ and $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$\begin{aligned}
\frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_d^*(t_k), w) =& \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_d^*(t_k), 0) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_d^*(t_k)), w) \\
& - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_d^*(t_k), 0) \\
\leq & - \frac{\tilde{c}_3}{\hat{c}_2} \rho_s + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), u_d^*(t_k), w) - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), u_d^*(t_k), 0) \\
\leq & - \frac{\tilde{c}_3}{\hat{c}_2} \rho_s + L_x^{'} |x(t) - x(t_k)| + L_w^{'} |w| \\
\leq & - \frac{\tilde{c}_3}{\hat{c}_2} \rho_s + L_x^{'} M \Delta + L_w^{'} w_m \\
\leq & - \varepsilon_w
\end{aligned} \tag{5.2.22}$$

Integrating Eq. 5.2.22 for $t \in [t_k, t_{k+1})$ will show that Eq. 5.2.17b holds. Therefore, the closed-loop state of the actual nonlinear process of Eq. 8.1 is maintained in $\Omega_{\hat{\rho}}$ for all $t \geq t_0$, and can be driven towards the origin in every sampling period under the decentralized LMPC control actions $u_j = u_{d_j}^*,\ j = 1, ..., N_{sys}$. Next, we consider the case where $x(t_k) \in \Omega_{\rho_s}$. Given that the state error is bounded by $|e(t)| = |x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + \nu_m}{L_x}(e^{L_x t} - 1)$, there exists a compact set $\Omega_{\rho_{min}} \supset \Omega_{\rho_{nn}}$ satisfying Eq. 6.19b such that the state of the actual nonlinear system of Eq. 8.1 at $t = t_{k+1}$ (i.e., $x(t_{k+1})$) does not leave $\Omega_{\rho_{min}}$ if the predicted state of the LSTM model of Eq. 5.2.6 at $t = t_{k+1}$ (i.e., $\hat{x}(t_{k+1})$) is bounded in $\Omega_{\rho_{nn}}$, which is guaranteed if $x(t_k) \in \Omega_{\rho_s}$, as shown in *Part* 2. Therefore, we have proven that the state $x(t)$ of the nonlinear system of Eq. 8.1 is bounded in $\Omega_{\hat{\rho}}$ for all $x_0 \in \Omega_{\hat{\rho}},\ t \geq t_0$ and ultimately bounded in $\Omega_{\rho_{min}}$ under the sample-and-hold implementation of the decentralized LMPC control design of Eq. 7.19. After the state $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the constraint of Eq. 5.2.13g is activated to maintain the predicted states $\hat{x}$ of the LSTM model of Eq. 5.2.13b inside $\Omega_{\rho_{nn}}$ over the entire prediction horizon, such that the closed-loop state $x(t)$ of the nonlinear system of Eq. 8.1 remains in $\Omega_{\rho_{min}}$ under the sample-and-hold implementation of the optimized decentralized control action $u_j = u_{d_j}^*,\ j = 1, ..., N_{sys}$ for $t \in [t_k, t_{k+1})$. At the next sampling step, if $x(t_{k+1}) \in \Omega_{\rho_{min}} \backslash \Omega_{\rho_{nn}}$, then Eq. 5.2.22 holds, and the contractive constraint of Eq. 5.2.13f will be activated instead to drive the state towards the origin during the next sampling period. Therefore, with the overall system stabilized under the control actions $u_j = u_{d_j}^*, j = 1, ..., N_{sys}$ of the decentralized LMPC design of Eq. 7.19, we have proven that the overall system can be stabilized under the control actions of the decentralized LMPC design of Eq. 7.19 implemented in a sample-and-hold fashion.

□

5.2.4 Application to a Two-CSTR-in-Series Process

We use a chemical process example to demonstrate the closed-loop simulation of the nonlinear process under the decentralized Lyapunov-based model predictive control (LMPC) system using the trained LSTM models, and the simulation results will be compared to that of decentralized LMPC system using first-principles models, as well as a centralized model predictive control system using LSTM models. More specifically, two well-mixed, non-isothermal continuous stirred tank reactors (CSTRs) that operate in series are considered, where an irreversible second-order exothermic reaction transforming a reactant A to a product B ($A \rightarrow B$) takes place in each reactor. A schematic diagram of the process is shown in Fig. 5.2.5. Reactant material A with inlet concentration C_{Aj0}, inlet temperature T_{j0} and feed volumetric flow rate of the reactor F_{j0}, are fed into each of the two reactors $j = 1,2$, where $j = 1$ denotes the first CSTR and $j = 2$ denotes the second CSTR. A heating jacket is installed on each CSTR to supply/remove heat at a rate Q_j, $j = 1,2$. The dynamic models of the two-CSTR-in-series process are obtained by the following material and energy balance equations:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_1}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{5.2.23a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_1} \tag{5.2.23b}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_2} C_{A20} + \frac{F_{10}}{V_2} C_{A1} - \frac{F_{10} + F_{20}}{V_2} C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{5.2.23c}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2} T_{20} + \frac{F_{10}}{V_2} T_1 - \frac{F_{10} + F_{20}}{V_2} T_2 + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_2} \tag{5.2.23d}$$

where C_{Aj}, V_j, T_j and Q_j, $j = 1,2$ are the concentration of reactant A, the volume of the reacting liquid, the temperature, and the heat input rate in the first and the second reactor, respectively. The reacting liquid has a constant density of ρ_L and a constant heat capacity of C_p for both reactors. E, R, ΔH, and k_0 represent the activation energy, ideal gas constant, enthalpy of the reaction, and pre-exponential constant, respectively. The process parameter values are listed in Table 5.2.1.

For both CSTRs, the manipulated inputs are the inlet concentration of species A and the heat input rate supplied by the heating jacket, which are represented by the deviation variables $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1,2$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{Aj0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q_j| \leq 5 \times 10^5\ kJ/hr$, $j = 1,2$. The states of the closed-loop system are the concentration of species A and the temperature in the first and the second reactor, which are also represented by their deviation variables such that the equilibrium point of the system

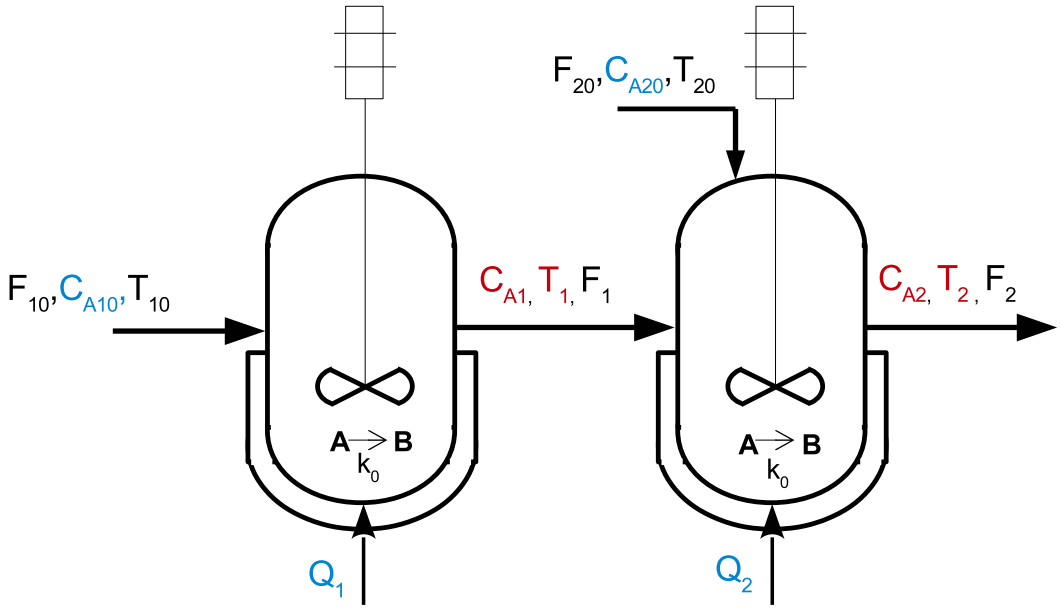


Figure 5.2.5: Process flow diagram of two CSTRs in series.

is at the origin of the state-space. Therefore, the vector of closed-loop states is $x = [C_{A1} - C_{A1_s} \;\; T_1 - T_{1_s} \;\; C_{A2} - C_{A2_s} \;\; T_2 - T_{2_s}]^T$, where C_{A1_s}, C_{A2_s}, T_{1_s} and T_{2_s} are the steady-state values of concentration of A and temperature in each of the two tanks, respectively.

The states of the first CSTR can be separately denoted as $x_1 = [C_{A1} - C_{A1_s} \;\; T_1 - T_{1_s}]^T$ and the states of the second CSTR are denoted as $x_2 = [C_{A2} - C_{A2_s} \;\; T_2 - T_{2_s}]^T$. Correspondingly, the manipulated inputs that regulate the state of the first CSTR can be denoted as $u_1 = [\Delta C_{A10} \;\; \Delta Q_1]^T$, and the manipulated inputs for the second CSTR are $u_2 = [\Delta C_{A20} \;\; \Delta Q_2]^T$. We let the states and the manipulate inputs of the first CSTR constitute subsystem 1, and that of the second CSTR constitute subsystem 2.

In a centralized LMPC framework, feedback measurement of all states x is received by the controller, and the manipulated inputs for the entire system, $u = [\Delta C_{A10} \;\; \Delta Q_1 \;\; \Delta C_{A20} \;\; \Delta Q_2]^T$, are computed by one centralized controller. In a decentralized LMPC scheme, we design two LMPCs, each one is responsible for one subsystem. The two LMPCs are solved independently in separate processors without any inter-communications. Due to the specific dynamic behavior and interactions between the two CSTRs, the states of the first CSTR play a part in the dynamic evolution of the states of the second CSTR, but not vice versa. Therefore, LMPC 1 receives

Table 5.2.1: Parameter values of the two CSTRs in series.

$T_{10} = 300\ K$	$T_{20} = 300\ K$
$F_{10} = 5\ m^3/hr$	$F_{20} = 5\ m^3/hr$
$V_1 = 1\ m^3$	$V_2 = 1\ m^3$
$T_{1_s} = 401.9\ K$	$T_{2_s} = 401.9\ K$
$C_{A1_s} = 1.954\ kmol/m^3$	$C_{A2_s} = 1.954\ kmol/m^3$
$C_{A10_s} = 4\ kmol/m^3$	$C_{A20_s} = 4\ kmol/m^3$
$Q_{1_s} = 0.0\ kJ/hr$	$Q_{2_s} = 0.0\ kJ/hr$
$k_0 = 8.46 \times 10^6\ m^3/kmol\ hr$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$C_p = 0.231\ kJ/kg\ K$	$R = 8.314\ kJ/kmol\ K$
$\rho_L = 1000\ kg/m^3$	$E = 5 \times 10^4\ kJ/kmol$

feedback measurements of x_1, predicts $\hat{x}_1$ in its internal model, and manipulates the inputs for the first CSTR, $u_1 = [\Delta C_{A10}\ \ \Delta Q_1]^T$. We assume that LMPC 2 has access to full-state measurement signals but does not have any knowledge on how x_1 dynamically evolves. Therefore, LMPC 2 receives feedback measurements of x, predicts the states of the second CSTR, $\hat{x}_2$, and only manipulates the inputs for the second CSTR, $u_2 = [\Delta C_{A20}\ \ \Delta Q_2]^T$. Furthermore, we assume that the feedback measurements of all states are available to both controllers at the same synchronous intervals $\Delta = 10^{-2}\ hr$.

The dynamic model of Eq. 8.45 is numerically simulated using the explicit Euler method with an integration time step of $h_c = 10^{-4}\ hr$. The nonlinear optimization problems of each of the decentralized LMPCs of Eq. 5.2.13 are solved using the Python module of the IPOPT software package, PyIpopt [113], with the same sampling period $\Delta = 10^{-2}\ hr$. The objective function in the decentralized LMPC optimization problem for subsystem j has the form $L_j(x_j, u_j) = x_j^T Q_j x_j + u_j^T R_j u_j$, where $Q_j = diag[2 \times 10^3\ \ 1]$, $R_j = diag[8 \times 10^{-13}\ \ 0.001]$, for $j = 1, ..., N_{sys}$. Here, we have two subsystems, $N_{sys} = 2$. The control Lyapunov function for each decentralized LMPC j is $V_j(x_j) = x_j^T P_j x_j$, for $j = 1, 2$, with the following positive definite P_j matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{5.2.24}$$

5.2.4.1 LSTM Network Development

To collect the training data for building the LSTM network models, we carry out open-loop simulations for finite sampling steps for various initial conditions inside Ω_{ρ_j}, where $\rho_j = 392,\ j = 1,2$, for both subsystems using the nonlinear system of Eq. 8.45. We apply various $u_1 \in U_1$, $u_2 \in U_2$ in a sample-and-hold manner to the nonlinear process and collect trajectories with a time interval of $q_{nn} \cdot h_c = 5h_c$. We separate these trajectories of manipulated input vector u_j and the state vector x_j into segments with a fixed length $T = 20$, such that the time period that the training samples cover, $T \times q_{nn} \times h_c = 20 \times 5 \times 10^{-4}\ hr$, is the same as the prediction period of the LSTM network $\Delta = 10^{-2}\ hr$. We also normalize both the input vector and state vector samples with respect to their means and standard deviations. The trajectories for $T = 20$ of the state vectors x_j over the prediction period Δ are used as the target state vectors when training the LSTM networks. Using the normalized samples of input and state vector sequences for each subsystem j, we develop a separate LSTM network j for each subsystem to predict the normalized future states over one sampling period Δ, which are then re-scaled to the actual future states $\hat{x}_1$ and $\hat{x}_2$ as described in the previous section. Each LSTM network j captures the dynamic behavior of its respective subsystem j, but does not have any information on the dynamic behaviors of all other subsystems; the LSTM netowrk j can then be used in the decentralized LMPC j as the prediction model.

It should be noted that, depending on the different architectures of the control systems, the choice of inputs and outputs as well as the structure of the LSTM model used in the control system may be different. In some cases, the nonlinear process, such as the one outlined in Eq. 8.45, cannot be completely decoupled into two separate subsystems. For instance, the states of the first CSTR x_1 are completely independent from the states and inputs of the second CSTR; however, the evolution of the states of the second CSTR, x_2, depends on the values of the states of the first CSTR, x_1. With knowledge on only the first CSTR (subsystem 1) and the first CSTR being independent from information on the second CSTR, LSTM network 1 receives information on $x_1(t_k)$ and $u_1(t_k)$ as the inputs to the neural network, and is able to predict $\hat{x}_1(t_{k+1})$, where $t_{k+1} := t_k + \Delta$. Although we assume that full-state feedback information is available to all controllers at the same time intervals, due to the unique process dynamics in this case, decentralized LMPC 1 for the first CSTR (subsystem 1) requires feedback measurements of x_1 only. Therefore, LSTM network 1, which is built to be used as the internal prediction model in decentralized LMPC 1, only uses $x_1(t_k)$ and $u_1(t_k)$ as the inputs to the neural network, and does not need information on $x_2(t_k)$. On the other hand, since the prediction of $\hat{x}_2(t_{k+1})$ depends on $x_1(t_k)$ and $x_2(t_k)$, decentralized LMPC 2 will take full advantage of the feedback measurements of all states. Therefore, the inputs to

the LSTM network 2 are $x_1(t_k)$, $x_2(t_k)$, and $u_2(t_k)$, and the outputs of LSTM network 2 are the predicted future states of the second CSTR, $\hat{x_2}(t)$ at $t = t_{k+1}$. Note that since LSTM network 2 does not have any information on u_1 and only predicts closed-loop states of the second CSTR x_2, the LSTM network model is only accounting for the impact of $x_1(t_k)$, $x_2(t_k)$, and $u_2(t_k)$ on the future state $x_2(t_{k+1})$ analogous to only having knowledge on the dynamic behavior of subsystem 2.

Both LSTM networks $j = 1,2$ are developed with 1 hidden layer, where *tanh* function is used as the activation function, and *Adam* is used as the optimizer to minimize the error between the sequences of target states and predicted states. With the normalized training data, the mean squared modeling error needs to be less than 5×10^{-7} to terminate the training process. After 10 epochs of training, each epoch taking on average 700 s, the training mean squared error between the predicted states $\hat{x}_1$ of the LSTM network 1 and the first-principles model of the first CSTR is 6.70×10^{-7}, and the validation mean squared error is 6.43×10^{-4}. Similarly, the mean squared error between the predicted states $\hat{x}_2$ of the LSTM network 2 and the first-principles model of the second CSTR in the training dataset is 7.02×10^{-7}, and the mean squared error in the validation dataset is 1.14×10^{-6}, after 10 epochs of training with each epoch taking on average 720 s. The Lyapunov function of the LSTM network model for both subsystems, $\hat{V}_j,\ j = 1,2$, is chosen to be the same as V_j. The set $\hat{D}_j$ can be characterized using the stabilizing control laws $u_j = \Phi_{nn_j}(x_j),\ j = 1,2$ of Eq. 7.7, computed using the numerically approximated $\hat{f}_j(t_k),\ \hat{g}_{j_i}(t_k)$ in Eq. 5.2.11. Subsequently, the closed-loop stability region $\Omega_{\hat{\rho}_j}$ for subsystem j can be characterized as the largest level set of $\hat{V}_j$ in $\hat{D}_j$ while also being a subset of Ω_{ρ_j}. The positive constants $\hat{\rho}_j,\ j = 1,2$, which are used to define the largest level sets of the control Lyapunov functions for subsystem 1 and 2 respectively, are $\hat{\rho}_1 = \hat{\rho}_2 = 380$. Additionally, the ultimate bounded region $\Omega_{\rho_{nn_j}}$, and subsequently, $\Omega_{\rho_{min_j}}$, are chosen to be $\rho_{nn_j} = 10$ and $\rho_{min_j} = 12$, for $j = 1,2$; the positive constants ρ_{nn_j} and ρ_{min_j} are determined via extensive closed-loop simulations with $u_j \in U_j$. More computational details on the development of a recurrent neural network model can be found in [128].

Remark 5.2.9. *The machine-learning-based model of each subsystem can be of different structures and variants of the recurrent neural network, and can be trained using different algorithms, depending on the complexity and dynamic behavior of the nonlinear process. In general, if the machine learning algorithm is capable of capturing temporal relationship between its input and output variables, then it can be used for model construction from data for dynamic systems. For example, a classic recurrent neural network model is fitted for a single-CSTR process in [128], and a recurrent neural network whose design depends on the a priori structural process knowledge is*

fitted for the two-CSTR-in-series process in [125]. Besides neural networks, hidden Markov models and support vector machines have also been applied in modeling and control of dynamic systems (e.g., [20, 69]). In this example, we constructed two LSTM network models for the two CSTRs individually. However, similar studies can be done where the first CSTR can be modeled using a classic RNN with simpler network structures to improve computational efficiency during training and execution, and the second CSTR can be modeled using a different class of recurrent neural networks (e.g., LSTM or GRU) with more units (i.e., more nonlinear activation functions and fitted parameters) to account for the relatively higher complexity in the second subsystem.

5.2.4.2 Closed-loop Model Predictive Control Simulations

We run closed-loop simulations on the two-CSTR-in-series process of Eq. 8.45 operated under a centralized LMPC system and under two decentralized LMPC systems. First, to design a centralized LMPC system, we use an LSTM network model developed for the overall two-CSTR process of Eq. 8.45 where, at time instant $t = t_k$, the inputs to this overall LSTM model are the four manipulated inputs $u(t) = [\Delta C_{A10} \;\; \Delta Q_1 \;\; \Delta C_{A20} \;\; \Delta Q_2]^T$ applied in a sample-and-hold fashion for $t \in [t_k, t_{k+1})$, as well as the four states of the overall process $x(t) = [C_{A1} - C_{A1_s} \;\; T_1 - T_{1_s} \;\; C_{A2} - C_{A2_s} \;\; T_2 - T_{2_s}]^T$ at $t = t_k$, and the outputs of this LSTM network model are the predicted states $\hat{x}(t_{k+1})$. The centralized LMPC uses this two-CSTR process model as the prediction model to optimize the four manipulated inputs $u^*(t_k)$ while receiving feedback of all states x. The closed-loop performance of this centralized LMPC using an LSTM network model will be used as a benchmark for comparison. Next, with the two LSTM networks developed for subsystem 1 and 2 as shown in Section 5.2.4.1, we design two decentralized LMPCs, where LMPC 1 uses the LSTM network model for subsystem 1, and LMPC 2 uses the LSTM network model for subsystem 2. LMPC 1 receives feedback of $x_1 = [C_{A1} - C_{A1_s} \;\; T_1 - T_{1_s}]^T$ and optimizes $u_1 = [\Delta C_{A10} \;\; \Delta Q_1]^T$; LMPC 2 receives feedback of all states x and optimizes $u_2 = [\Delta C_{A20} \;\; \Delta Q_2]^T$. The two decentralized LMPCs are independent of each other, and they can be solved simultaneously in separate processors in parallel to save computation time. Therefore, the computation time for solving one LMPC iteration is the maximum time of the two decentralized controllers. To further demonstrate the efficacy of the decentralized LMPC system using LSTM models, we compare the closed-loop performance of the decentralized LMPC system using LSTM models with that of the same decentralized LMPC system using first-principles models. The closed-loop performances of the centralized and the decentralized control networks using their respective LSTM models, as well as using the first-principles models are compared in terms of the computation time of solving one MPC

iteration and the sum of squared percentage error of the closed-loop states x for a total simulation period of $t_p = 0.3$ hr, which are both shown in Table 5.2.2. The sum of squared percentage error is in the form of $SSPE = \int_0^{t_p} \left[(\frac{C_{A1}-C_{A1_s}}{C_{A1_s}})^2 + (\frac{T_1-T_{1_s}}{T_{1_s}})^2 + (\frac{C_{A2}-C_{A2_s}}{C_{A2_s}})^2 + (\frac{T_2-T_{2_s}}{T_{2_s}})^2 \right] dt$. It is shown that the average computation time is significantly reduced when the process of Eq. 8.45 is operated under two decentralized LMPCs compared with the result of a centralized LMPC, and both control systems generate comparable sum of squared percentage errors. Furthermore, the decentralized LMPC system using LSTM models achieves similar average computation time and sum of squared percentage error as the decentralized LMPC system using first-principles models. The computation time of all simulated control systems are lower than the sampling period of the process; therefore, the proposed control system using LSTM models can be implemented without computational issues.

Table 5.2.2: Average LMPC computation time in one sampling period and the sum of squared percentage error of all states along the closed-loop trajectory under the centralized and the decentralized systems using their respective LSTM models and first-principles models, all with a total operation time of 0.3 hr.

	Ave. Computation Time (s)	Sum of Squared Percentage Error
Decentralized (LSTM)	5.41	2.94
Decentralized (First-Principles)	5.18	2.84
Centralized (LSTM)	35.26	3.08
Centralized (First-Principles)	27.93	2.96

The closed-loop state evolution of the two-CSTR-in-series process under the two decentralized LMPCs using the LSTM model and using the first-principles model are shown in Fig. 5.2.6, and the corresponding input profiles are shown in Fig. 5.2.7. The closed-loop state trajectories in state-space under the centralized LMPC using an LSTM model, the decentralized LMPC system using two LSTM models, and the decentralized LMPC system using the respective first-principles model for each subsystem, are shown in Fig. 5.2.8 to show the boundedness and convergence of the closed-loop states. Starting from initial conditions $x_0 = [-1.5 \;\; 70 \;\; 1.5 \;\; -70]^T$, all states of each subsystem j, $j = 1,2$, converge to $\Omega_{\rho_{min_j}}$ within 0.07 hr and are bounded in $\Omega_{\hat{\rho}_j}$ under the two decentralized LMPCs using their respective LSTM network models for all time. Therefore, through closed-loop simulations and assessing their performance metrics, we have illustrated the effectiveness of the proposed decentralized LMPC design using separate LSTM network models in stabilizing the overall nonlinear process.

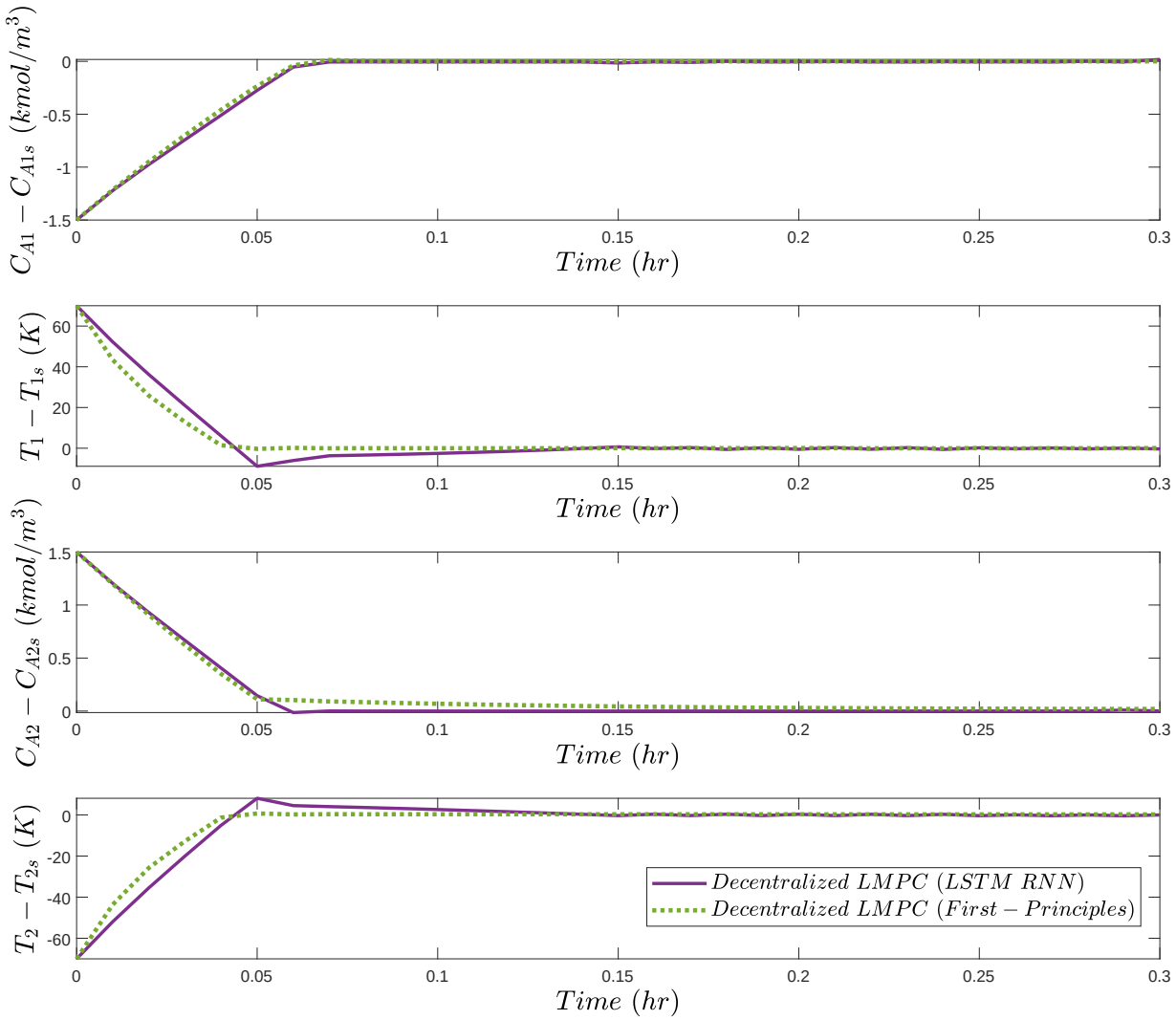


Figure 5.2.6: Closed-loop state trajectories of the decentralized LMPC systems using LSTM model and first-principles (FP) model, respectively.

Remark 5.2.10. *The absence of a first-principles model provides the motivation for developing machine-learning-based models for nonlinear processes. However, we show the simulation results of the control frameworks using first-principles models of the two-CSTR-in-series process of Eq. 8.45 to adequately compare with the results of the same process operated under the same control frameworks using LSTM network models. In real-life scenarios where the first-principles model of the studied process is not attainable, the same comparison can be carried out with respect to real plant data.*

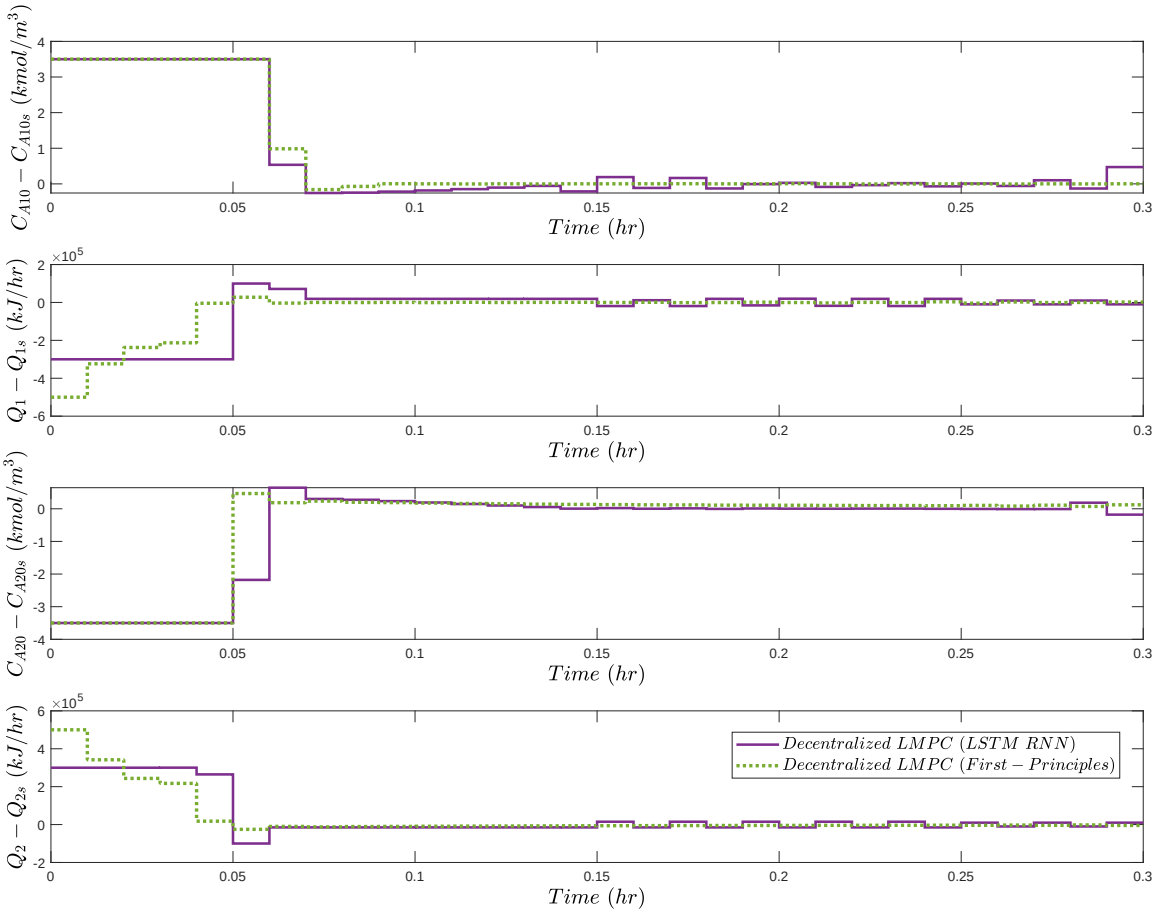


Figure 5.2.7: Closed-loop input trajectories of the decentralized LMPC systems using LSTM model and first-principles (FP) model, respectively.

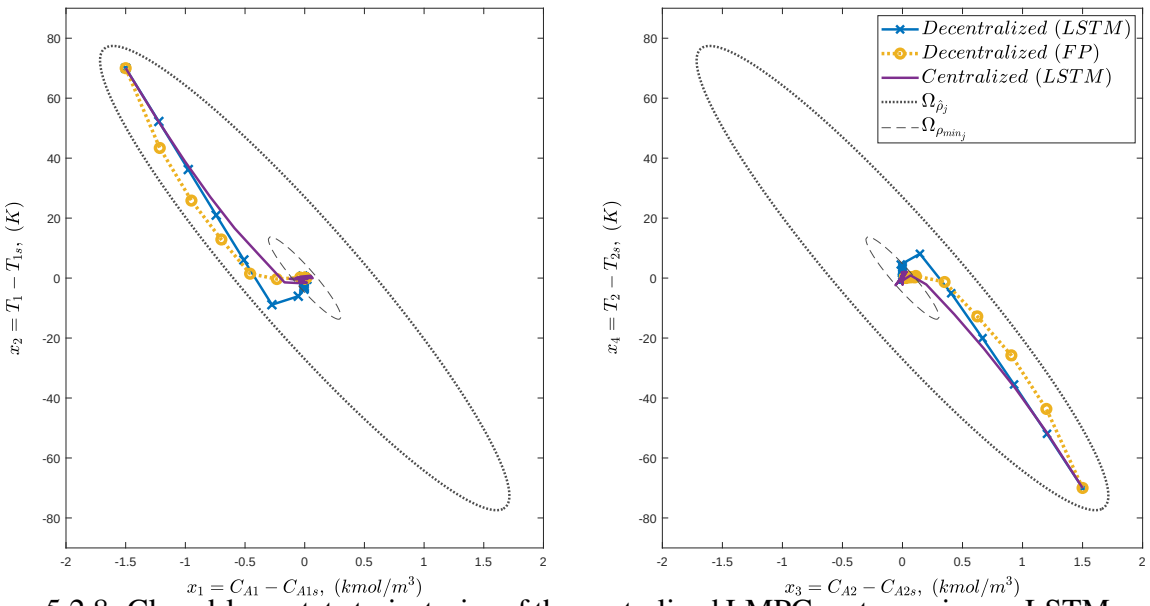


Figure 5.2.8: Closed-loop state trajectories of the centralized LMPC system using an LSTM model, and the decentralized LMPC systems using LSTM models and first-principles (FP) models in state space, showing the boundedness of the states of each subsystem $j,\ j = 1,2$ in $\Omega_{\hat{\rho}_j}$ for all operation time $t_p = 0.03\ hr$, and the convergence of the states of each subsystem j in $\Omega_{\rho_{min_j}}$ after $t \geq 0.07\ hr$.

Chapter 6

Machine-Learning-Based Construction of Barrier Functions and Models for Safe Model Predictive Control

In this chapter, we propose a Control Lyapunov-Barrier Function-based Model Predictive Control method utilizing a feed-forward neural network specified Control Barrier Function and a recurrent neural network predictive model to stabilize nonlinear processes with input constraints, and to guarantee that safety requirements are met for all times. The nonlinear system is first modeled using recurrent neural network (RNN) techniques, and a Control Barrier Function is characterized by constructing a feed-forward neural network (FNN) model with unique structures and properties. The FNN model for the CBF is trained based on data samples collected from safe and unsafe operating regions, and the resulting FNN model is verified to demonstrate that the safety properties of the CBF are satisfied. Given sufficiently small bounded modeling errors for both the FNN and the RNN models, the proposed control system is able to guarantee closed-loop stability while preventing the closed-loop states from entering unsafe regions in state-space under sample-and-hold control action implementation. We provide the theoretical analysis for both bounded unsafe sets in state-space, and demonstrate the effectiveness of the proposed control strategy using a nonlinear chemical process example with a bounded unsafe region.

Many previous works have been developed to incorporate various machine learning modeling approaches with the design of MPC (e.g., [34, 47, 117]). In this work, in addition to using a a recurrent neural network (RNN) as the prediction model in the MPC, we also characterize the CLBF using an FNN model. Provided with extensive training data which are labeled, NN models

can be constructed with strategically chosen architectures, activation and objective functions, and evaluation metrics, and ultimately trained with supervision to approximate the barrier function. The FNN-specified barrier function can be proven to satisfy all required conditions of a barrier function, and can then be applied to the CLBF-based controllers. In our study, we consider a CLBF-MPC, where the barrier function is found using feed-forward neural network structures.

The chapter is organized as follows. Preliminaries on the class of systems considered, the stabilizability assumptions and safety considerations given by CLBF are described in Section 2. We introduce the structure and the development of the NN model in Section 3, along with proofs of its efficacy when applied in the CLBF-based controllers. In Section 4, the formulation of the CLBF-MPC with NN-specified BF is presented, where the proof for recursive feasibility of the optimization problem, as well as the boundedness and convergence of the closed-loop state while always avoiding the unsafe region is shown, given bounded modeling error of the NN-BF, sample-and-hold implementation of control actions, and a well-characterized set of initial conditions. Lastly, in Section 5, the control method proposed in this work is applied to an example chemical process to illustrate its effectiveness.

6.1 Preliminaries

6.1.1 Notation

We use $|\cdot|$ to denote the Euclidean norm of a vector. $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$ denotes the standard Lie derivative . Furthermore, a scalar continuous function $V : \mathbf{R}^n \to \mathbf{R}$ is proper if for all $k \in \mathbf{R}$, the set $\{x \in \mathbf{R}^n \mid V(x) \leq k\}$ is a compact set. x^T denotes the transpose of x. $\mathscr{B}_\beta(\varepsilon) := \{x \in \mathbf{R}^n \mid |x-\varepsilon| < \beta\}$ is an open ball around ε with radius of β, with positive real numbers β and ε. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. $\emptyset$ signifies the null set. Lastly, a function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable.

6.1.2 Class of Systems

The class of continuous-time nonlinear systems considered is described by the following state-space form:

$$\dot{x} = f(x) + g(x)u + h(x)w,\ x(t_0) = x_0 \tag{6.1}$$

where $x \in \mathbf{R}^n$ represents the state vector, $u \in \mathbf{R}^m$ represents the input vector, and $w \in \mathbf{W}$ is the bounded disturbance vector, where $\mathbf{W} := \{w \in \mathbf{R}^l \mid |w| \leq \theta,\ \theta \geq 0\}$. The input control actions are constrained by their lower and upper bounds, $u \in U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^m$. $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are vector and matrix functions of dimensions $n \times 1$, $n \times m$, and $n \times l$, respectively, and we assume that they are sufficiently smooth. Without loss of generality, we take the initial time t_0 to be zero, i.e., $t_0 = 0$. It is assumed that $f(0) = 0$. Thus, the system of Eq. 8.1 with $w(t) \equiv 0$ has a steady state at the origin. Additionally, it is assumed the feedback measurement of $x(t)$ is available at synchronous sampling times, t_k.

6.1.3 Stabilizability Assumptions Expressed via Lyapunov-based Control

For the nominal system of Eq. 8.1 with $w(t) \equiv 0$, we assume that there exists a positive definite and proper Control Lyapunov Function (CLF), V, that satisfies the small control property as well as the following conditions:

$$L_f V(x) < 0, \forall x \in \{z \in \mathbf{R}^n \backslash \{0\} \mid L_g V(z) = 0\} \tag{6.2}$$

The small control property states that for every $\varepsilon > 0$, $\exists\ \delta > 0$, s.t. $\forall\ x \in \mathscr{B}_\delta(0)$, there exists u that satisfies $|u| < \varepsilon$ and $L_f V(x) + L_g V(x) \cdot u < 0$ [98]. The existence of such CLF implies that there exists a stabilizing feedback control law $\Phi(x) \in U$ for the nominal system of Eq. 8.1 such that Eq. 7.2 holds for $u = \Phi(x) \in U$, and the origin of the closed-loop system is rendered asymptotically stable for all x in a neighborhood around the origin under $u = \Phi(x) \in U$. A candidate of a stabilizing feedback control law is shown in [67]. We can characterize a region ϕ_u where the time derivative of $V(x)$ is rendered negative using $u = \Phi(x) \in U$ as: $\phi_u = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V(x) + L_g V(x) \cdot u < 0, u = \Phi(x) \in U\} \cup \{0\}$. Within this region ϕ_u, we define a level set of $V(x)$, $\Omega_b := \{x \in \phi_u \mid V(x) \leq b,\ b > 0\}$, which is a forward invariant set such that the closed-loop trajectory $x(t),\ t \geq 0$ of the nominal system of Eq. 8.1 with $w(t) \equiv 0$ under $u = \Phi(x) \in U$ remains in Ω_b, for any initial condition $x_0 \in \Omega_b$.

6.1.4 Process Modeled Using Recurrent Neural Network

When first-principles models of a process are not available or may not be accurate, one way to model the process is to use data-based machine-learning methods. A recurrent neural network (RNN) is an effective algorithm that is capable of modeling the dynamics of the nonlinear system

of Eq. 8.1, and its general formulation is shown as follows:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T y \tag{6.3}$$

where $\hat{x} \in \mathbf{R}^n$ is the state vector of the RNN model and $u \in \mathbf{R}^m$ is the manipulated input vector. $y = [y_1, ..., y_n, y_{n+1}, ..., y_{m+n}] = [\sigma(\hat{x}_1), ..., \sigma(\hat{x}_n), u_1, ..., u_m] \in \mathbf{R}^{n+m}$ is a vector that contains both the network state $\hat{x}$ and the input u, where $\sigma(\cdot)$ is the nonlinear activation function. A is a diagonal coefficient matrix, i.e., $A = diag\{-a_1, ..., -a_n\} \in \mathbf{R}^{n \times n}$, and $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(m+n) \times n}$ with $\theta_i = b_i[w_{i1}, ..., w_{i(m+n)}]$, $i = 1, ..., n$. a_i and b_i are constants. w_{ij} represents the weight connecting the jth input to the ith neuron where $i = 1, ..., n$ and $j = 1, ..., (m+n)$. a_i is assumed to be positive for each state $\hat{x}_i$ to be bounded-input bounded-state stable. For the remainder of the manuscript, x will be used to denote the state of the nonlinear system of Eq. 8.1, and $\hat{x}$ will be used to denote the state of the RNN model of Eq. 6.3.

As the RNN model of Eq. 6.3 is an input-affine system, it can be also written in the form that is similar to the general nonlinear system of Eq. 8.1:

$$\dot{x} = \hat{f}(x) + \hat{g}(x)u \tag{6.4}$$

where $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ can be derived from the coefficient matrices A and Θ in Eq. 6.3 and are assumed to be sufficiently smooth. The construction of RNN models including procedures on data generation, model training and validation, as well as developing an ensemble of models have been outlined in [127]. Note that $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ can be approximated via numerical methods. The modeling error of the RNN, $|\nu|$, needs to be below a certain threshold ν_m during training, and is bounded as follows: $|\nu| = |F(x, u, 0) - F_{nn}(x, u)| \leq \gamma|x| \leq \nu_m$, where $\gamma > 0$. The bounded modeling error is a requirement to ensure that the nonlinear system of Eq. 8.1 and the RNN model of Eq. 6.3 have the same steady-state within the operating region considered, and is a requirement used in subsequent stability and safety proofs. Furthermore, we assume that there exists a CLF $\hat{V}$ and a Lyapunov-based stabilizing control law $u = \Phi_{nn}(x) \in U$ that renders the origin of the RNN modeled system of Eq. 6.3 asymptotically stable.

6.1.5 Control Barrier Function

We assume that there exists an open set $\mathscr{D}$ in state-space that should be avoided during operations; for example, the operating conditions within this region may result in process safety risks. We also

characterize a set of safe states, $\mathscr{X}_0 := \{x \in \mathbf{R}^n \backslash \mathscr{D}\}$ where $\{0\} \in \mathscr{X}_0$ and $\mathscr{X}_0 \cap \mathscr{D} = \emptyset$. The set of initial conditions to be considered in this study will be developed from $\mathscr{X}_0$.

Two types of unsafe regions are generally considered in literature – bounded and unbounded sets – the details of which can be found in [122]. We denote bounded unsafe set and unbounded unsafe set as $\mathscr{D}_b$ and $\mathscr{D}_u$, respectively. Due to the data-driven approach of constructing the CBF, there are relevant restrictions with collecting finite samples from compact sets of safe and unsafe data. Therefore, only bounded unsafe sets can be handled in this approach. Details of the limitations on the compactness of the unsafe set will be further presented in Section 3.2.2. We address process operational safety in the sense of not entering any unsafe sets. The formal definition of process operational safety is defined as follows:

Definition 6.1. *Consider the nominal system of Eq. 8.1 with $w(t) \equiv 0$ and input constraints $u \in U$. If there exists a control law $u = \Phi(x) \in U$ such that, for any initial state $x(t_0) = x_0 \in \mathscr{X}_0$, the origin of the closed-loop system of Eq. 8.1 is rendered asymptotically stable, and the state trajectories of the system do not enter the unsafe region, i.e., $x(t) \in \mathscr{X}_0$, $x(t) \notin \mathscr{D}$, $\forall\, t \geq 0$, then the control law $u = \Phi(x)$ maintains the process state within a safe operating region $\mathscr{X}_0$ for all times.*

Following the definition of safe operation, the definition of a valid Control Barrier Function (CBF) is as follows [116]:

Definition 6.2. *Given a set of unsafe points in state-space $\mathscr{D}$, a $\mathscr{C}^1$ function $B(x) : \mathbf{R}^n \to \mathbf{R}$ is a CBF if it satisfies the following properties:*

$$B(x) > 0, \quad \forall\, x \in \mathscr{D} \tag{6.5a}$$

$$L_f B(x) \leq 0,\ \forall\, x \in \{z \in \mathbf{R}^n \backslash \mathscr{D} \mid L_g B(z) = 0\} \tag{6.5b}$$

$$\mathscr{X}_B := \{x \in \mathbf{R}^n \mid B(x) \leq 0\} \neq \emptyset \tag{6.5c}$$

Remark 6.1. *In many chemical processes, unbounded unsafe sets represent unsafe operations where process variables exceed their safety envelopes, e.g., when temperature is above a threshold that can lead to overheating, or when concentration is below a threshold which could lead to incomplete reaction. Bounded unsafe sets are more common in mechanical processes; e.g., robotics navigation to avoid obstacles in its trajectory. In chemical processes, many mid-range operating conditions are sub-optimal to achieving high yields of reactions. For example, low pressure steam could be used as a coolant at low temperature, or as a heat source at high temperature. However, if its temperature is in the middle ranges, then it is not fit for either purpose and might be discarded as waste.*

Remark 6.2. *For many industrial operations where the dynamics of the process is not well understood, it is difficult to model the intertwined relations between multitudes of variables. Although it is possible to specify certain operating envelopes within which individual process variables should operate within, the impact of these variables on other variables, and vice versa, may not be pre-assessed and therefore, cannot be explicitly described. It is common for plant operators to provide data points at or near which operation would be avoided. With these data points, we can use the approach discussed in this manuscript to model a CBF.*

6.2 Construction of Barrier Function using Neural Networks

6.2.1 Neural Network Structure and Training

In our study, we use a feed-forward artificial neural network (FNN) to synthesize the control barrier function $\hat{B}(x)$. A conventional FNN consists of an input layer, an output layer, and any number of hidden layers in between that can be customized depending on network complexity and computational need. Each layer undergoes nonlinear transformations, which consists of activation functions of a bias term plus the weighted sum of neurons in the previous layer. In turn, the results of these activation functions provide the values of the neurons in the current layer. The neurons in the first hidden layer are derived from the input layer, and the outputs are calculated based on the neurons in the last hidden layer. The input layer contains the state vector x of the nonlinear system of Eq. 8.1 with a dimension of $\mathbf{R}^n$, and the single output in the output layer provides the predicted barrier function $\hat{B}(x)$ for the particular input data sample x.

Without having prior knowledge on an explicit formulation of the barrier function $B(x)$, training data for the NN will be collected for both the safe and unsafe regions with target values of $B(x)$ that satisfy the conditions of Eq. 8.3a and Eq. 8.3c for each region respectively. We choose nonlinear activation functions that will best fit the dichotomous nature of the barrier function, which will aid in obtaining better prediction accuracy. Furthermore, we encode custom loss function and evaluation metric for the FNN in order to ensure that the condition of Eq. 8.3b is also satisfied. Since this approach is data-driven and dependent on the sampling of training data generation, we also provide formal proof for the verification of the FNN-learned barrier function $\hat{B}(x)$, proving that the $\hat{B}(x)$ indeed satisfies all conditions of Eq. 8.3. The structure of a 2-hidden-layer FNN are presented in Fig. 6.1 and in Eq. 6.6 below:

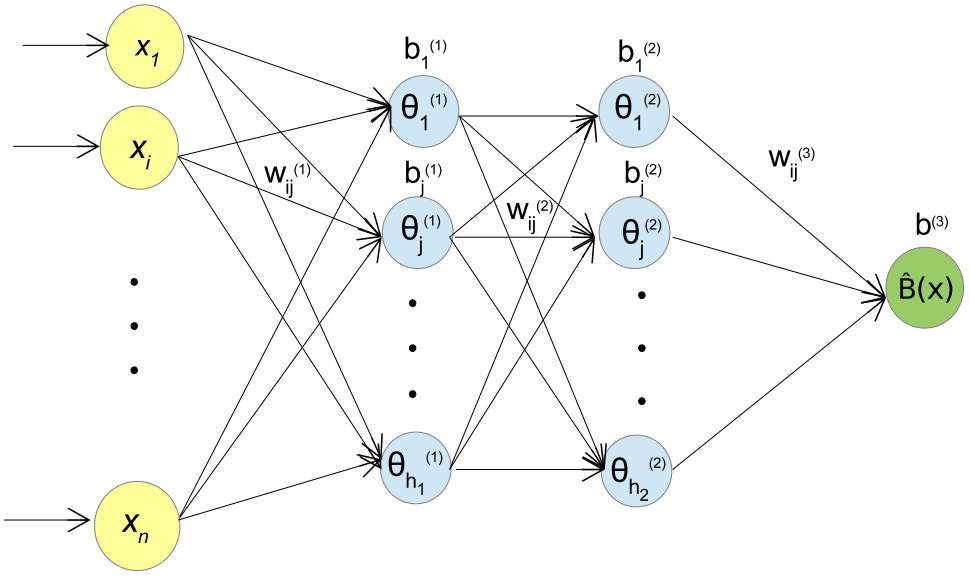


Figure 6.1: Structure of a 2-hidden-layer feedforward neural network with the state vector $x \in \mathbf{R}^n$ as inputs and the CBF $\hat{B}(x)$ as the output.

$$\theta_j^{(1)} = g_1\left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}\right) \tag{6.6a}$$

$$\theta_j^{(2)} = g_2\left(\sum_{i=1}^{h_1} w_{ij}^{(2)} \theta_i^{(1)} + b_j^{(2)}\right) \tag{6.6b}$$

$$\hat{B} = g_3\left(\sum_{i=1}^{h_2} w_i^{(3)} \theta_i^{(2)} + b^{(3)}\right) \tag{6.6c}$$

with $\theta_j^{(1)}$ and $\theta_j^{(2)}$ representing neurons in the first and second hidden layer, respectively, where $j = 1, ..., h_l$ is the number of neurons in layer $l = 1$ and $l = 2$. The weight associated with the connections between neurons i and j, which are in consecutive layers (from $l-1$ to l), is denoted by $w_{ij}^{(l)}$, and $b_j^{(l)}$ represents the bias term added to the weighted sum for each neuron in hidden

layers $l = 1,\ 2$ and output layer $l = 3$. Upon receiving the information from the previous layer, neurons in the current layer l then computes an output via a nonlinear activation function, denoted g_l. There are many choices of activation functions, e.g., sigmoid function, $g(z) = \frac{1}{1+e^{-z}}$, hyperbolic tangent sigmoid function $g(z) = \frac{2}{1+e^{-2z}} - 1$, and ReLu function, $g(z) = max(0, z)$; interested readers may refer to [96] for more details on the different activation functions and their characteristics. The two-hidden-layer representation in this section can be similarly extended to multiple hidden layers for better fitting suited for other applications.

To train the FNN, training data are generated by sampling points from the operating region of the system (i.e., $x \in \mathscr{X} \subset \mathbf{R}^n$ where $\mathscr{X}$ is a compact set). In order to ensure that the FNN developed from discrete data samples is able to meet the conditions of $B(x)$ in a continuous sense (for reasons that will be further explained in Section 8.2.2), the safe and the unsafe operating regions we consider need to be compact and connected sets within $\mathscr{X}$. Therefore, we first characterize a compact and connected set $\mathscr{H}$, that encloses the open set $\mathscr{D}$ such that a key condition used for designing CBF, as shown in Eq. 8.32, is satisfied. These design guidelines are explained in detail in Section 4.1. Then, we use $\mathscr{H}'$, which encloses $\mathscr{H}$ with sufficient margin, to represent the unsafe region. Samples from the unsafe region $\mathscr{H}'$ and the safe region $\mathscr{X} \setminus \mathscr{H}'$ are collected by discretizing the respective regions with a fixed mesh size $(\delta x)_{\mathscr{H}'}$ and $(\delta x)_{\mathscr{X}\setminus\mathscr{H}'}$ respectively. We denote the finite sampled data set of the unsafe region as $S_{\mathscr{H}'}$, and the finite sampled data set of the safe region as $S_{\mathscr{I}}$. To achieve best training results, equal number of samples for each set are obtained, where N_D and N_I represents the number of sampled data points in the unsafe and the safe regions respectively.

Due to the dichotomous condition of $B(x)$ as specified by Eq. 8.3a and Eq. 8.3c (depending on whether the particular point x belongs to the safe or the unsafe region in state-space), the activation functions of the two hidden layers and the output layer are all chosen to be the hyperbolic tangent sigmoid function (i.e., $tanh(z)$) due to the nature of $tanh(z)$ functions settling at 1 as z approaches $+\infty$, and -1 as z approaches $-\infty$, effectively polarizing the outputs and allowing the outputs of the FNN to take either relatively constant positive values, or relatively constant negative values. According to the requirement of conditions Eq. 8.3a and Eq. 8.3c, we can then label safe data points in $S_{\mathscr{I}}$ as having an output value $B(x)$ of -1, and unsafe data points in $S_{\mathscr{H}'}$ as having an output value $B(x)$ of $+1$. These labeled target output values can be then compared to the predicted output values given by the layers of neurons and $tanh$ activation functions; more specifically, we use mean squared error in the objective function to track the error between the target $B(x)$ and the predicted $\hat{B}(x)$ values. Minimizing the mean squared error between the target $B(x)$ and the

predicted $\hat{B}(x)$ values will address the conditions of Eq. 8.3a and Eq. 8.3c. Furthermore, we add an additional term in the cost function, which uses the *ReLu* function and penalizes sample points that violate the condition of Eq. 8.3b. To obtain an optimal set of weights and biases that will produce an output $\hat{B}(x)$ that meets all three conditions of Eq. 8.3, we use an optimization algorithm to minimize the cost function, which has the following form:

$$\begin{aligned} Cost &= Cost_1 + Cost_2 \\ Cost_1 &= \alpha \frac{1}{N_s} \sum_{k=1}^{N_s} (\hat{B}_k - B_k)^2 \\ Cost_2 &= \beta \sum_{j=1}^{N_I} ReLu(L_{\hat{f}} \hat{B}_j + \tau_I) \end{aligned} \tag{6.7}$$

where $Cost_1$ represents the mean squared error between the target and the predicted outputs for all samples in the operating region, and $Cost_2$ represents the penalizing term to ensure that $L_{\hat{f}}\hat{B} \leq 0$ for all $x \in S_{\mathscr{H}}$. $k = 1,...,N_s$ represents the total number of samples in the training dataset, i.e., $N_s = N_D + N_I$, and $j = 1,...,N_I$ represents all sample points in the safe operating region. In $Cost_2$, τ_I is a small positive constant. Since *ReLu* is defined to take the maximum between its argument and 0, we penalize any occurrences of data samples producing $L_{\hat{f}}\hat{B}_j + \tau_I > 0$, thereby forcing $L_{\hat{f}}\hat{B}_j$ to be negative for all points in the safe region. Positive constants α and β are hyper-parameters representing the weights of the two terms in the cost function. During training, when $\sum_{j=1}^{N_I} ReLu(L_{\hat{f}}\hat{B}_j + \tau_I)$ has reached 0, then we have arrived at a predicted barrier function $\hat{B}(x)$ that satisfies the condition Eq. 8.3b. In order to ensure the efficacy of the predicted barrier function $\hat{B}(x)$ at the end of the network training, we evaluate and monitor $Cost_1$ and $Cost_2$ separately during training, and implement stopping criteria that would require both $Cost_1$ and $Cost_2$ to reach below their respective thresholds to ensure bounded modeling error for $\hat{B}(x)$ as well as negative semi-definiteness of $L_{\hat{f}}\hat{B} \leq 0$ for all $x \in S_{\mathscr{J}}$.

6.2.2 Effectiveness of NN-based Barrier Function

The definition given in Definition 8.2 presents the properties and characteristics of an adequate barrier function. In this section, we will show how FNN-based barrier function can be verified to satisfy Definition 8.2 and be applied to continuous nonlinear systems of Eq. 8.1.

6.2.2.1 Continuity and Differentiability

By Definition 8.2, the barrier function is a continuously differentiable function, thus we need to show that $\hat{B}(x)$ and $\dot{\hat{B}}(x)$ are continuous. By the universal approximation theorem, feed-forward artificial neural networks are able to model any continuous nonlinear functions on compact subsets of the state space $\mathbf{R}^n$ with sufficient number of neurons [99]. Furthermore, $\hat{B}(x)$ is the output of a series of nonlinear activation functions of inputs, weights and biases. We choose activation functions that are Lipschitz continuous in the compact subset within which the FNN training data is collected, such as *tanh*. All hidden layers and output layer of the FNN model for approximating $B(x)$ use *tanh* as the activation function, therefore making $\hat{B}(x)$ also Lipschitz continuous.

6.2.2.2 Verification

Minimizing the cost function of Eq. 8.6 aims to minimize the error between the values of $B(x)$ and $\hat{B}(x)$ as well as to penalize violations of the decrease condition $L_{\hat{f}}\hat{B}(x) \leq 0,\ \forall x \in S_{\mathscr{J}}$, but does not enforce the conditions of Eq. 8.3 in a continuous sense. Therefore, we must verify that these conditions hold over the compact subsets for which the respective data samples are collected from. Many verification techniques can be used, such as the Satisfiability Modulo Theories (SMT) algorithm in [19] and the Lipschitz method in [58, 88]. More specifically, the work in [13] has shown the verification of the decrease condition for a candidate Lyapunov function on a finite sampling of a bounded set of initial conditions. The following theorem is adapted from the work in [13], in which the full proof of the theorem is presented in details.

Theorem 6.1. *Let S_s be a finite set sampled from a compact set $S \subset \mathbf{R}^n$ such that for all $x \in S$, there exists at least a pair $(x_s, \delta_{x_s}) \in S_s \times \mathbf{R}_+$ s.t. $|x - x_s| \leq \delta x_s$. If for all $x_s \in S_s$ it holds that $F(x_s) \leq -L_F \cdot \delta x_s$ (or respectively $F(x_s) < -L_F \cdot \delta x_s$), where $L_F > 0$ is the Lipschitz constant for function F, then $F(x) \leq 0$ (respectively $F(x) < 0$) holds for all $x \in S$. [13]*

Therefore, by Theorem 8.1, we can show that $L_{\hat{f}}\hat{B}(x) \leq 0,\ \forall\, x \in \mathscr{X} \setminus \mathscr{H}'$ by checking the tightened condition $L_{\hat{f}}\hat{B}(x) \leq -L' \cdot \delta x_{\mathscr{X}\setminus\mathscr{H}'},\ \forall\, x \in S_{\mathscr{J}}$, where the sampled finite set $S_{\mathscr{J}}$ is a discretization of the compact set $\mathscr{X} \setminus \mathscr{H}'$, $L' > 0$ is the Lipschitz constant for $L_{\hat{f}}\hat{B}(x)$, and $\delta x_{\mathscr{X}\setminus\mathscr{H}'} > 0$ is the discretization mesh size in the safe region $\mathscr{X} \setminus \mathscr{H}'$. Similarly, we can show that $\hat{B}(x) \leq 0,\ \forall\, x \in \mathscr{X} \setminus \mathscr{H}'$ by showing that $\hat{B}(x) \leq -L \cdot \delta x_{\mathscr{X}\setminus\mathscr{H}'}\ \forall\, x \in S_{\mathscr{J}}$, where L is the Lipschitz constant for $\hat{B}$. Once this tightened condition is verified, it is sufficient to show that the condition of Eq. 8.3c is satisfied. Lastly, we show that the condition of Eq. 8.3a is satisfied by verifying that $-\hat{B}(x) < -L \cdot \delta x_{\mathscr{H}'},\ \forall\, x \in S_{\mathscr{H}'}$, which means $-\hat{B}(x) < 0\ \forall\, x \in \mathscr{H}'$, and equivalently $\hat{B}(x) > 0\ \forall\, x \in \mathscr{H}'$.

6.2.2.3 Characterization of Unsafe Data

It is generally difficult to describe the exact unsafe operating conditions of nonlinear processes as the actual unsafe set $\mathscr{D}$ can be open and not connected. For example, unsafe sets are not connected if there are multiple clusters of unsafe operating regions located within close proximity such that navigating around them would be nearly impossible. Therefore, in order to proceed with designing an adequate CBF, we first characterize a compact, connected set, denoted as $\mathscr{H}$, to embed the unsafe set $\mathscr{D}$. This approach is similarly applied in the design of constrained CLBF $W_c(x)$ proposed in [119], where an explicit form of the CBF was constructed. In our study, we use a similar compact and connected set $\mathscr{H}$, such that $\mathscr{D} \subset \mathscr{H}$, to characterize the set of unsafe states considered.

To obtain a FNN model for the CBF, we need to supply the model with training data samples from the unsafe and the safe operating regions in state-space. As there always exists inherent modeling error in the approximation of the CBF, a contingency margin should be considered when generating these training data. More specifically, we use a larger compact set, $\mathscr{H}'$ where $\mathscr{H} \subset \mathscr{H}'$, to distinguish the different labels assigned to safe and unsafe data samples. Data samples obtained from a discretization of the region $\mathscr{H}'$ will be labeled as "unsafe", and data samples obtained from a discretization of the set $\mathscr{X} \setminus \mathscr{H}'$ will be labeled as "safe". Upon verification of the trained model with regards to the definition of CBF (Eq. 8.3) and with regards to the classification accuracy, we ensure that the resulting unsafe region as predicted by the FNN-modeled CBF, denoted as $\hat{\mathscr{H}}$, should be as close to $\mathscr{H}'$ as possible and always be a superset of the compact unsafe region $\mathscr{H}$. We leave sufficient margin between $\mathscr{H}$ and $\mathscr{H}'$ so that, with bounded modeling error in the FNN model for CBF (Eq. 6.6) and in the RNN model for the nonlinear process (Eq. 6.3), it is guaranteed that the closed-loop state will not enter $\mathscr{H}$ given any initial condition $x_0 \in \mathscr{X} \setminus \mathscr{H}'$.

Remark 6.3. *Despite rigorous training and extensive validation, there may still exist modeling error in the testing phase or in the implementation of the NN model that we cannot eliminate completely. Without knowing an explicit analytical form of $B(x)$, it is difficult to quantify such modeling error as well. We assume that the contingency margin that we leave when characterizing the set of unsafe points for which training data will be generated from is able to account for the inherent modeling error of the FNN-modeled CBF $\hat{B}(x)$. Hence, while the FNN output $\hat{B}$ aims to characterize the unsafe region boundary as close to $\mathscr{H}'$ as possible, in the presence of modeling error, the predicted $\hat{B}(x)$ will satisfy all conditions on CBF and CLBF with respect to the actual unsafe closed and compact set $\mathscr{H}$.*

Remark 6.4. *To verify that the FNN-modeled barrier function $\hat{B}(x)$ satisfies the properties of a*

CBF in a continuous sense, the finite sets of safe and unsafe data used to build the FNN must be sampled from a compact (i.e., closed and bounded) safe set $\mathscr{X} \setminus \mathscr{H}'$, and a compact unsafe set $\mathscr{H}'$, respectively, as shown in Section 3.2.2. It should be noted that the unsafe set $\mathscr{H}'$ is a set characterized by the user to enclose the compact set $\mathscr{H}$ to account for the error margin in the neural network model. Moreover, the compact set $\mathscr{H}$ is a set characterized by the user to enclose the actual unsafe region $\mathscr{D}$. In this study, we focus on bounded unsafe sets, $\mathscr{D}_b$. Bounded unsafe sets in the middle of the operating region could obstruct the state trajectories, and are therefore the more difficult case to handle. In the case of unbounded unsafe sets, $\mathscr{D}_u$, they must be first approximated by a sufficiently large compact set within a reasonable physical range, $\mathscr{D}_{\tilde{b}}$. Based on this approximated unsafe set, we can then characterize the compact set $\mathscr{H}' \supset \mathscr{D}_{\tilde{b}}$ from which we will collect finite samples of unsafe data used for training the FNN, and subsequently, the analysis and design of CLBF will follow that of the bounded unsafe set.

6.3 Stabilization and safety via Control Lyapunov-Barrier Function

A Control Lyapunov-Barrier Function (CLBF) was proposed in [90], which is a weighted average of a CLF and a CBF, where it was shown that if a CLBF exists for the system of Eq. 8.1 with $w(t) \equiv 0$, there exists a controller $u = \Phi(x)$ that will maintain the closed-loop state with $x_0 \in \mathscr{X}_0$ within a level set of the CLBF and outside of $\mathscr{D}$ at all times. The work in [119, 122] extends the analysis to constrained CLBFs, accounting for physical constraints on manipulated inputs $u \in U$. In the recent work in [123], a constrained CLBF-MPC is analyzed where the MPC uses a prediction model built from an ensemble of RNN models, and the stability and safety properties of this approach were guaranteed using a control law $u = \Phi_{nn}(x) \in U$. The CLF needs to meet the conditions outlined in Section 8.1.3 and the CBF needs to meet the conditions of Eq. 8.3. As we have shown in Section 8.2.2, upon successful verification of $\hat{B}(x)$ against the conditions of Eq. 8.3, it is a valid CBF which CLBF-based controllers can take in. Therefore, the theoretical results shown in [123] can be similarly applied to a CLBF constructed with a FNN-specified CBF $\hat{B}(x)$, where closed-loop stability and safe operation can be achieved under the CLBF-based control law $u = \Phi_{nn}(x) \in U$ for the RNN system of Eq. 6.3.

The definition of a constrained CLBF constructed using the FNN-CBF $\hat{B}$, denoted as $W_{nn}(x)$ with respect to the RNN model of Eq. 6.3 is as follows:

Definition 6.3. *Given a set of unsafe points in state-space $\mathscr{D}$, a proper, lower-bounded and $\mathscr{C}^1$ function $W_{nn}(x) : \mathbf{R}^n \to \mathbf{R}$ is a constrained CLBF if $W_{nn}(x)$ has a minimum at the origin and also satisfies the following properties:*

$$W_{nn}(x) > \rho, \ \ \forall\, x \in \mathscr{D} \subset \phi_{uc} \tag{6.8a}$$

$$L_{\hat{f}} W_{nn}(x) < 0,$$

$$\forall\, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathscr{X}_e) \mid L_{\hat{g}} W_{nn}(z) = 0\} \tag{6.8b}$$

$$\mathscr{U}_\rho := \{x \in \phi_{uc} \mid W_{nn}(x) \leq \rho\} \neq \emptyset \tag{6.8c}$$

$$\overline{\phi_{uc} \backslash (\mathscr{D} \cup \mathscr{U}_\rho)} \cap \overline{\mathscr{D}} = \emptyset \tag{6.8d}$$

where $\rho \in \mathbf{R}$ is a constant, $\mathscr{X}_e := \{x \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\}) \mid \partial W_{nn}(x)/\partial x = 0\}$ is a set of states for the RNN model of Eq. 6.4 where $L_{\hat{f}} W_{nn}(x) = 0$ (for $x \neq 0$) due to $\partial W_{nn}(x)/\partial x = 0$. $\hat{f}$ and $\hat{g}$ are from the RNN model in Eq. 6.4. Under a stabilizing control law $u = \Phi_{nn}(x) \in U$, ϕ_{uc} is defined to be the union of the set where the time-derivative of $W_{nn}(x)$ is negative with constrained inputs, the origin, and the set $\mathscr{X}_e$: $\phi_{uc} = \{x \in \mathbf{R}^n \mid \dot{W_{nn}}(x(t), \Phi_{nn}(x)) = L_{\hat{f}} W_{nn} + L_{\hat{g}} W_{nn} \cdot u < -\alpha_W |W_{nn}(x) - W_{nn}(0)|, u = \Phi_{nn}(x) \in U\} \cup \{0\} \cup \mathscr{X}_e$, and α_W is a positive real number used to characterize the set ϕ_{uc}. A control law $u = \Phi_{nn}(x) \in U$ that renders the origin exponentially stable within ϕ_{uc} is assumed to exist for the RNN system of Eq. 6.3 in the sense that there exists a $\mathscr{C}^1$ constrained CLBF $W_{nn}(x)$. The CLBF function satisfies the following conditions $\forall\, x \in \phi_{uc}$ and has a minimum at the origin:

$$\hat{c}_1 |x|^2 \leq W_{nn}(x) - \rho_0 \leq \hat{c}_2 |x|^2, \tag{6.9a}$$

$$\frac{\partial W_{nn}(x)}{\partial x} F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{c}_3 |x|^2, \ \forall\, x \in \phi_{uc} \backslash \mathscr{B}_\delta(x_e) \tag{6.9b}$$

$$\left| \frac{\partial W_{nn}(x)}{\partial x} \right| \leq \hat{c}_4 |x| \tag{6.9c}$$

where $\hat{c}_j(\cdot)$, $j = 1,\ 2,\ 3,\ 4$ are positive real numbers, $W_{nn}(0) = \rho_0$ is the global minimum value of $W_{nn}(x)$, and $\mathscr{B}_\delta(x_e)$ is a small neighborhood around $x_e \in \mathscr{X}_e$. $F_{nn}(x,u)$ is the RNN system of Eq. 6.3.

In addition, in the nonlinear system of Eq. 8.1, we assumed that functions f, g and h are sufficiently smooth, by continuity, there exist positive constants L_x, L_w, $L_x^{'}, L_w^{'}$, M, such that for

all $x, x' \in \mathscr{U}_\rho$, $w \in W$, and $u \in U$, the following conditions will hold:

$$|F(x,u,w)| \leq M \tag{6.10a}$$

$$|F(x,u,w) - F(x',u,0)| \leq L_x|x-x'| + L_w|w| \tag{6.10b}$$

$$\left|\frac{\partial W_{nn}(x)}{\partial x}F(x,u,w) - \frac{\partial W_{nn}(x')}{\partial x}F(x',u,0)\right| \leq L_x^{'}|x-x'| + | \leq L_w^{'}|w_m| \tag{6.10c}$$

In [122], an exemplar stabilizing control law $\Phi_{nn}(x)$ is shown. The Lyapunov function $V(x)$ can be replaced with the CLBF $W_{nn}(x)$ within the Lyapunov-based control law that is presented in the form of the universal Sontag controller [67].

6.3.1 Design of Constrained CLBF

We first design CLF and CBF separately to meet their respective conditions, and we follow the practical design guidelines presented in [119] to construct the CLBF. We present the design method for choosing the CLF, the CBF, and the corresponding weights in this section, and show that the $\hat{B}(x)$ is able to meet all the conditions on CBF, through which $W_{nn}(x)$ is able to meet all its required properties of Eq. 8.31 and Eq. 7.5 and has a global minimum at the origin.

Proposition 6.1. *Given an open set $\mathscr{D}$ of unsafe states for the system of Eq. 8.1 with $w(t) \equiv 0$, assume that there exists a $\mathscr{C}^1$ CLF $V : \mathbf{R}^n \to \mathbf{R}_+$, and a $\mathscr{C}^1$ CBF $\hat{B} : \mathbf{R}^n \to \mathbf{R}$, such that the following conditions hold:*

$$c_1 |x|^2 \leq V(x) \leq c_2 |x|^2, \ \forall x \in \mathbf{R}^n, c_2 > c_1 > 0 \tag{6.11}$$

$$\mathscr{D} \subset \mathscr{H} \subset \mathscr{H}' \subset \phi_{uc}, \ 0 \notin \mathscr{H}, \ 0 \notin \mathscr{H}' \tag{6.12}$$

$$\hat{B}(x) = -\eta < 0, \ \forall x \in \mathbf{R}^n \backslash \mathscr{H}'; \ \hat{B}(x) > 0, \ \forall x \in \mathscr{H}' \tag{6.13}$$

where $\mathscr{H}$ is a compact and connected set within ϕ_{uc}, and $\mathscr{H}'$ is a compact and connected set within ϕ_{uc} that encloses $\mathscr{H}$ with sufficient margin to account for modeling errors in $\hat{B}(x)$, $\hat{f}$, and $\hat{g}$. Define $W_{nn}(x)$ to have the form $W_{nn}(x) := V(x) + \mu\hat{B}(x) + \nu$, where:

$$\left|\frac{\partial W_{nn}(x)}{\partial x}\right| \leq \hat{c}_4|x| \tag{6.14}$$

$$\begin{gathered} L_{\hat{f}}W_{nn}(x) < 0, \\ \forall \, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathscr{X}_e) \mid L_{\hat{g}}W_{nn}(z) = 0\} \end{gathered} \tag{6.15}$$

$$\mu > \frac{c_2 c_3 - c_1 c_4}{\eta}, \tag{6.16a}$$

$$\nu = \rho - c_1 c_4, \tag{6.16b}$$

$$c_3 := \max_{x \in \partial \mathscr{H}'} |x|^2, \tag{6.16c}$$

$$c_4 := \min_{x \in \partial \mathscr{D}} |x|^2 \tag{6.16d}$$

then the control law $\Phi_{nn}(x)$ (Lyapunov-based Sontag controller with $W_{nn}(x)$ replacing $V(x)$) guarantees that the closed-loop state is bounded in $\phi_{uc} \backslash \mathscr{H}$ and does not enter the unsafe region $\mathscr{H}$ for all times, for any initial state $x_0 \in \phi_{uc} \backslash \mathscr{D}_{\mathscr{H}'}$, where $\mathscr{D}_{\mathscr{H}'} := \{x \in \mathscr{H}' \mid W_{nn}(x) > \rho\}$.

Proof. By the construction of the FNN model for the CBF, $\hat{B}(x)$ meets the condition of Eq. 8.33 despite modeling error due to the characterization of $\mathscr{H}' \supset \mathscr{H}$, where the margin between $\mathscr{H}'$ and $\mathscr{H}$ accounts for the modeling error of $\hat{B}(x)$, and of $\hat{f}$ and $\hat{g}$ of the RNN model of Eq. 6.4. It was proven in [119] and [123] that a constrained CLBF designed following these guidelines satisfies the properties of Eq. 8.31 and Eq. 7.5c; the proofs will be omitted here.

In addition, we also need to prove that the constrained CLBF $W_{nn}(x)$ designed using a CLF $V(x)$ and a CBF $\hat{B}(x)$ satisfies the additional properties of Eq. 7.5a – Eq. 7.5b, which are required for $u = \Phi_{nn}(x) \in U$ to render the origin of the RNN system of Eq. 6.3 exponentially stable. In order to make sure Eq. 7.5a holds, both $|V(x) - V(0)|$ and $|\hat{B}(x) - \hat{B}(0)|$ need to be bounded. From Eq. 6.11, we know that $c_1 |x|^2 \leq V(x) - V(0) \leq c_2 |x|^2$, $\forall\, x \in \mathbf{R}^n$ since $V(0) = 0$. Based on the construction and the training objectives of the FNN-modeled CBF, we also know that $|\hat{B}(x) - \hat{B}(0)| \leq 2$ within a sufficiently small bounded error that includes modeling inaccuracies and numerical error in the $\hat{B}$ predictions. Therefore, the resulting CLBF, $W_{nn}(x) - W_{nn}(0)$, which is a linear combination of the bounded $V(x)$ and $\hat{B}(x)$, is also bounded by its respective lower and upper bounds as shown in Eq. 7.5a.

The condition of Eq. 7.5b holds due to the definition of ϕ_{uc} as well as the boundedness of $|W_{nn}(x) - W_{nn}(0)|$, where $\hat{c}_3 = \alpha_W \hat{c}_2$. Furthermore, $V(x)$ has a global minimum at the origin: $V(0) = 0$ and $V(x) > 0$ for all $x \in \mathbf{R}^n \backslash \{0\}$. With a sufficiently small bounded numerical error and modeling error, $\hat{B}(x) = -1$ for all $x \in \phi_{uc} \backslash \mathscr{H}'$, where $\{0\} \in \phi_{uc} \backslash \mathscr{H}'$, and $\hat{B}(x) = +1$ for all $x \in \mathscr{H}'$. Therefore, $\hat{B}(x)$ also has a global minimum at the origin within bounded numerical error. Since $W_{nn}(x)$ is a weighted average of $V(x)$ and $\hat{B}(x)$, the global minimum of $W_{nn}(x)$ is also at the origin. Therefore, we have demonstrated, a CLBF $W_{nn}(x)$ and a controller $u = \Phi_{nn}(x) \in U$ exist that together satisfy all conditions of Eq. 8.31 and Eq. 7.5, and will guarantee exponential stability

for all $x_0 \in \phi_{uc} \backslash \mathscr{D}_{\mathscr{H}'}$.

In the rest of our paper, we will focus on initial conditions in $\mathscr{U}_\rho$, which is a forward invariant set of $W_{nn}(x)$ as defined in Eq. 8.31d. Furthermore, closed-loop stability and safety for the RNN system of Eq. 6.3 are analyzed with respect to bounded unsafe sets similar to Theorem 1 in [122]. Specifically, in the presence of bounded unsafe sets, there exist stationary points $x_e \in \mathscr{X}_e$ in state-space other than the origin that can be treated as saddle points. When states reach these stationary points, the continuous control law of $u = \Phi_{nn}(x) \in U$ is unable to drive the states away from them. We design discontinuous control actions $u = \bar{u}(x) \in U$, $\bar{u}(x) \neq \Phi_{nn}(x)$, to drive the states away from these saddle points in the direction of decreasing $W_{nn}(x)$. The theorem below provides the sufficient conditions under which the controller $u = \Phi_{nn}(x) \in U$ designed based on the CLBF $W_{nn}(x)$ is able to fulfill stability and safety for the closed-loop RNN system of Eq. 6.3.

Theorem 6.2. *Consider a constrained CLBF $W_{nn}(x)$: $\mathbf{R}^n \to \mathbf{R}$ built using $\hat{B}(x)$, that has a minimum at the origin and satisfies the conditions of Eq. 8.31, exists for the RNN system of Eq. 6.3. The controller $u = \Phi_{nn}(x) \in U$ that satisfies Eq. 7.5 guarantees that the closed-loop state stays within $\mathscr{U}_\rho$ for all times for any $x_0 \in \mathscr{U}_\rho$. In the presence of a bounded unsafe region in state-space, the origin can be rendered exponentially stable under $u = \Phi_{nn}(x) \in U$ (if x is not near a saddle point x_e) and under discontinuous control actions $u = \bar{u}(x) \in U$ that decrease $W_{nn}(x)$ (if x is near a saddle point x_e) for all $x_0 \in \mathscr{U}_\rho$.*

Proof. It has been proven in [119, 122, 123] that the universal Sontag controller [98] with the CLBF $W_{nn}(x)$ replacing the Lyapunov function $V(x)$ gives a valid $u = \Phi_{nn}(x) \in U$ that ensures $\dot{W}_{nn}(x) \leq 0$ for all $x \in \mathscr{U}_\rho$, therefore ensuring that for $x_0 \in \mathscr{U}_\rho$, x is bounded in $\mathscr{U}_\rho$ for all times. Furthermore, since $\mathscr{U}_\rho$ is a level set of $W_{nn}(x)$ in ϕ_{uc} (ϕ_{uc} is a set within which Eq. 7.5 is met), the origin is rendered exponentially stable under $u = \Phi_{nn}(x) \in U$. In the presence of bounded unsafe regions, the saddle points at which $\dot{W}_{nn}(x) = 0$ can be handled by discontinuous control actions $u = \bar{u}(x) \in U, \bar{u}(x) \neq \Phi_{nn}(x)$ that decrease $W_{nn}(x)$. The detailed proofs for handling bounded unsafe sets can be referenced from Thereom 1 in [122], and will be omitted here.

6.4 CLBF-based MPC using FNN CBF and RNN Prediction Model

In this work, we propose a CLBF-based MPC which is designed based on a CLBF-based controller that ensures simultaneous closed-loop stability and process safety for the nonlinear system of

Eq. 8.1. The CLBF-based controller $u = \Phi_{nn}(x) \in U$, which uses a $W_{nn}(x)$ incorporating an FNN-modeled CBF ($\hat{B}(x)$), is designed based on the $\hat{f}$ and $\hat{g}$ of the RNN system of Eq. 6.4. Then, the CLBF-MPC is developed to optimize process performance while driving the process states to a small ball around the origin. So far, we have shown that a valid CLBF $W_{nn}(x)$ can be constructed using $\hat{B}(x)$, from which the controller $u = \Phi_{nn}(x) \in U$ exponentially stabilizes the origin of the RNN system of Eq. 6.3 while keeping closed-loop states in a safe region of operation $\mathscr{U}_\rho$.

The control actions of the CLBF-MPC are implemented in a sample-and-hold manner to the original nonlinear system of Eq. 8.1, i.e., for any $t \in [t_k, t_{k+1})$, $u(t) = u(t_k)$, where $t_{k+1} := t_k + \Delta$. Note that Δ is the sampling period of the MPC. Due to the presence of bounded disturbances in the nonlinear system of Eq. 8.1, as well as the modeling mismatch between the RNN system of Eq. 6.3 and the first-principles system of Eq. 8.1, we must investigate the safety and stability properties of the system with these considerations in mind.

In Proposition 1 of [123], given that the modeling error of the RNN model of Eq. 6.3 is bounded by $|\nu| = |F(x,u,0) - F_{nn}(x,u)| \leq \gamma|x| \leq \nu_m$, and the nonlinear system of Eq. 8.1 has bounded disturbances $|w| \leq w_m$, the boundedness of the state error $|x - \hat{x}|$ and the difference between $|W_c(x) - W_c(\hat{x}))|$ was shown, where W_c is a CLBF that uses an explicitly defined CBF $B(x)$. More specifically, $|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + \nu_m}{L_x}(e^{L_x t} - 1)$, and $W_c(x) \leq W_c(\hat{x}) + \kappa|x - \hat{x}|^2 + \frac{\hat{c}_4\sqrt{\rho - \rho_0}}{\sqrt{\hat{c}_1}}|x - \hat{x}|$, where $\kappa > 0$. Since $W_{nn}(x)$ can be also shown to be continuous and bounded on a compact set and behaves the same as $W_c(x)$, the same proofs apply on $W_{nn}(x)$, and we can conclude that $W_{nn}(x) \leq W_{nn}(\hat{x}) + \frac{\hat{c}_4\sqrt{\rho - \rho_0}}{\sqrt{\hat{c}_1}} f_w(t) + \kappa f_w(t)^2$, where $\hat{c}_1$, and $\hat{c}_4$ are positive real constants in Eq. 7.5 for $W_{nn}(x)$.

All subsequent proofs on the stability and safety of the nominal system of Eq. 8.1 under the CLBF-based control law designed based on $W_{nn}(x)$ follow the same proofs in [123] (Proposition 2 and Proposition 3), with W_{nn} replacing W_c. This is shown for bounded unsafe regions, where the CLBF-based control law designed using the RNN model of Eq. 6.3 can also guarantee closed-loop exponential stability and safety for the nominal system of Eq. 8.1. We will show that the combination of the CLBF-based control law $u = \Phi_{nn}(x) \in U$ along with discontinuous control actions that yield decreasing $W_{nn}(x)$ will provide exponential stability and safety in the case of bounded unsafe sets.

We consider the nominal system of Eq. 8.1 with a bounded unsafe set, where saddle points $x_e \in \mathscr{X}_e$ are present in $\mathscr{U}_\rho$. We provide sufficient conditions, under which the continuous CLBF-based control actions $u = \Phi_{nn}(x) \in U$ and the control actions $u = \bar{u}(x) \in U$ designed in a discontinuous manner, can ensure closed-loop stability and safety. The proof for the following

adapted proposition can be found in [123].

Proposition 6.2. *If the RNN model is developed such that for all $x \in \mathscr{U}_\rho$ and $u \in U$, the modeling error is constrained by $|\nu| = |F(x,u,0) - F_{nn}(x,u)| \leq \gamma|x|$, where γ is a positive real number that satisfies $\gamma < \hat{c}_3/\hat{c}_4$, and furthermore, Eq. 6.17 is satisfied under discontinuous control actions $u = \bar{u}(x) \in U$ when $x(t_k) = \hat{x}(t_k) \in \mathscr{B}_\delta(x_e)$,*

$$W_{nn}(\hat{x}(t)) < W_{nn}(\hat{x}(t_k)) - f_e(t - t_k),\ \forall t > t_k \tag{6.17}$$

where

$$f_e(t - t_k) := \frac{\hat{c}_4\sqrt{\rho - \rho_0}}{\sqrt{\hat{c}_1}} f_w(t - t_k) - \kappa f_w(t - t_k)^2$$

and $f_w(t)$ is the upper bound on the state error $|x(t) - \hat{x}(t)| \leq f_w(t)$, then the stability and safety properties outlined in Theorem 6.2 also apply to the nominal system of Eq. 8.1 with a bounded unsafe region $\mathscr{D}_b$ under $u = \Phi_{nn}(x) \in U$ and $u = \bar{u}(x) \in U$.

In the presence of bounded disturbances (i.e., $|w(t)| \leq w_m$), now we show that the nonlinear system of Eq. 8.1 can be rendered exponentially stable and maintained within the safe region $\mathscr{U}_\rho$. Under the sample-and-hold implementation of the control actions, the state of the closed-loop system of Eq. 8.1 is always bounded in $\mathscr{U}_\rho$, and converges to a small neighborhood $\mathscr{U}_{\rho_{min}}$. Given that the set of initial conditions $\mathscr{U}_\rho$ for which exponential stability and safety of the RNN system of Eq. 6.3 can be guaranteed under the CLBF-based control laws is characterized using $W_{nn}(x)$, the following proposition has been adapted from Proposition 4 in [123].

Proposition 6.3. *Consider the nonlinear system of Eq. 8.1 under the CLBF-based controller $u = \Phi_{nn}(x) \in U$ (under sample-and-hold implementation), which is built using a valid W_{nn} following Proposition 8.1 and satisfies Eq. 7.5. If Eq. 6.17 is satisfied under the controller $u = \bar{u}(x) \in U$ in a sample-and-hold fashion for $x \in \mathscr{B}_\delta(x_e)$, and there exist $\varepsilon_w > 0$, $\Delta > 0$ and $\rho_s < \rho_{nn} < \rho_{min} < \rho$ that satisfy*

$$-\frac{\tilde{c}_3}{\hat{c}_2}(\rho_s - \rho_0) + L_x^{'} M\Delta + L_w^{'} w_m \leq -\varepsilon_w \tag{6.18}$$

and

$$\rho_{nn} := \max\{W_{nn}(\hat{x}(t+\Delta)) \mid u \in U,\ \hat{x}(t) \in \mathscr{U}_{\rho_s}\} \tag{6.19a}$$

$$\rho_{nn} + f_e(\Delta) \leq \rho_{min} \tag{6.19b}$$

where $f_e(t)$ is given by Eq. 6.17, then for any $x(t_k) \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_s}$, $W_{nn}(x(t))$ is guaranteed to decrease

within every sampling period, and can be bounded in $\mathscr{U}_\rho$ for all times and ultimately bounded in $\mathscr{U}_{\rho_{min}}$.

6.4.1 Formulation of CLBF-MPC

The following optimization problem represents the CLBF-MPC design:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) dt \tag{6.20a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{6.20b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{6.20c}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{6.20d}$$

$$\dot{W}_{nn}(x(t_k), u(t_k)) \leq \dot{W}_{nn}(x(t_k), \Phi_{nn}(t_k)) \quad \text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W_{nn}(x(t_k)) > \rho_{nn} \tag{6.20e}$$

$$W_{nn}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \text{ if } W_{nn}(x(t_k)) \leq \rho_{nn} \tag{6.20f}$$

$$W_{nn}(\tilde{x}(t)) < W_{nn}(x(t_k)) - f_e(t - t_k), \ \forall \ t \in (t_k, t_{k+N}), \quad \text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{6.20g}$$

where $\tilde{x}(t)$ is the predicted state trajectory, N is the number of sampling periods in the prediction horizon, $S(\Delta)$ represents the set of piece-wise constant functions with sampling period Δ. The CLBF-MPC optimization problem has an objective function of Eq. 8.44a, which is the integral of $L(\tilde{x}(t), u(t))$ over the prediction horizon typically in a quadratic form, i.e., $L(\tilde{x}(t), u(t)) = \tilde{x}^T Q \tilde{x} + u^T R u$, where Q, R are positive definite weighting matrices, and the minimum of this objective function is achieved at the origin. In Eq. 8.44b, the predicted state trajectory $\tilde{x}(t)$, $t \in [t_k, t_{k+N})$ are calculated using the RNN model F_{nn} of Eq. 6.3. $\dot{W}_{nn}(x,u)$ represents $\frac{\partial W_{nn}(x)}{\partial x}(\hat{f}(x) + \hat{g}(x)u)$, where $\hat{f}$ and $\hat{g}$ are the approximated nonlinear functions of the RNN model of Eq. 6.4. The input constraints of Eq. 8.44d are applied over the entire prediction horizon. We assume that the measured states of the closed-loop system are available at each sampling time. For the predicated state trajectory of Eq. 8.44b, the initial condition is obtained from the feedback measurement of Eq. 8.44c at $t = t_k$. To ensure closed-loop stability and process operational safety, the constraints of Eqs. 8.44e-8.44g are utilized. When $x(t_k) \notin \mathscr{B}_\delta(x_e)$ and $W_{nn}(x(t_k)) > \rho_{nn}$, the constraint of Eq. 8.44e forces $W_{nn}(\tilde{x})$ to decrease along at a rate less than or equal to that under the CLBF-based control law $u = \Phi_{nn}(x) \in U$. If $W_{nn}(x(t_k)) \leq \rho_{nn}$, the constraint of Eq. 8.44f maintains the predicted state of the RNN system of

Eq. 6.3 within $\mathscr{U}_{\rho_{nn}}$ such that the closed-loop state of the nonlinear system of Eq. 8.1 is bounded in $\mathscr{U}_{\rho_{min}}$. Furthermore, if $x(t_k) \in \mathscr{B}_\delta(x_e)$, the constraint of Eq. 8.44g decreases $W_{nn}(x)$ over the predicted state trajectory such that the closed-loop state can escape from saddle points x_e within a finite number of sampling periods. Once the state leaves $\mathscr{B}_\delta(x_e)$, it will be driven to smaller level sets of $W_{nn}(x)$ under the constraint of Eq. 8.44e, therefore guaranteeing that the state does not go back to $\mathscr{B}_\delta(x_e)$ afterwards. After solving the optimal solution $u^*(t)$, the control action at the first time instant, $u^*(t_k)$, is applied over the next sampling period in a sample-and-hold manner. The horizon will be moved forward one sampling period, and the above process is repeated.

The following theorem and proof will show that safety and stability can be established for the closed-loop nonlinear system of Eq. 8.1 using the CLBF-based MPC.

Theorem 6.3. *Consider the system of Eq. 8.1 with a constrained CLBF W_{nn} built using a NN-BF $\hat{B}(x)$ following the procedures in Section 8.2. The constrained NN-based CLBF $W_{nn}(x)$ satisfies Eq. 8.31 and has a minimum at the origin. Given any initial state $x_0 \in \mathscr{U}_\rho$, it is guaranteed that the CLBF-MPC optimization problem of Eq. 8.44 can be solved with recursive feasibility for all times. Additionally, under the sample-and-hold implementation of CLBF-MPC based on an RNN prediction model that satisfies $|\nu| = |F(x,u,0) - F_{nn}(x,u)| \leq \gamma|x| \leq \nu_m$ and the conditions in Proposition 6.3, it is guaranteed that for any $x_0 \in \mathscr{U}_\rho$, the state is bounded in $\mathscr{U}_\rho$, $\forall\ t \geq 0$, and ultimately converges to $\mathscr{U}_{\rho_{min}}$ as $t \to \infty$.*

Proof.

Part 1: The optimization problem of Eq. 8.44 for the CLBF-MPC has a feasible solution at all times since the CLBF-MPC constraints of Eqs. 8.44d-8.44g can be satisfied by the sample-and-hold control laws $u = \bar{u}(x) \in U$, $\forall x \in \mathscr{B}_\delta(x_e)$ and $u = \Phi_{nn}(x) \in U$, $\forall x \in \mathscr{U}_\rho \backslash \mathscr{B}_\delta(x_e)$. This has been demonstrated in Propositions 6.2 and 6.3 with detailed proofs outlined in [123]. More specifically, the control laws $u = \bar{u}(x) \in U$, $\forall x \in \mathscr{B}_\delta(x_e)$ and $u = \Phi_{nn}(x) \in U$, $\forall x \in \mathscr{U}_\rho \backslash \mathscr{B}_\delta(x_e)$ are already constrained by $u \in U$, therefore the input constraint of Eq. 8.44d can be met over the prediction horizon. By letting $u(t_k) = \Phi_{nn}(x(t_k))$ when $x(t_k) \in \mathscr{U}_\rho \backslash (\mathscr{B}_\delta(x_e) \cup \mathscr{U}_{\rho_{nn}})$, Eq. 8.44e is also satisfied. It has been shown in Proposition 6.3 that once the closed-loop state is inside $\mathscr{U}_{\rho_s}$ under the control law $u = \Phi_{nn}(x) \in U$, it will not leave $\mathscr{U}_{\rho_{nn}}$ for any $u \in U$ within one sampling period. Thus, the CLBF-based control law $u(t) = \Phi_{nn}(x(t_{k+i})) \in U$, $\forall t \in [t_{k+i}, t_{k+i+1})$ with $i = 0, ..., N-1$ provides a feasible trajectory of control actions that meet the constraint of Eq. 8.44f. Lastly, as the controller $u = \bar{u}(x) \in U$ satisfies Eq. 6.17, the control action $u(t) = \bar{u}(x(t_{k+i})) \in U$, $\forall t \in [t_{k+i}, t_{k+i+1})$ with $i = 0, ..., N-1$ will satisfy the constraint of Eq. 8.44g

and drive the state away from saddle points if $x(t_k) \in \mathscr{B}_\delta(x_e)$. The proof for recursive feasibility of the optimization problem of Eq. 8.44 is complete.

Part 2: Now we will prove that the optimized solution of Eq. 8.44 will guarantee simultaneous safety and stability for the closed-loop nonlinear system of Eq. 8.1. For any $x_0 \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_{nn}}$, the constraint of Eq. 8.44e ensures that the optimized CLBF-MPC control action u^* will drive the closed-loop state of the RNN system towards the origin and into $\mathscr{U}_{\rho_{nn}}$ within finite sampling periods. After the state enters $\mathscr{U}_{\rho_{nn}}$, the constraint of Eq. 8.44f ensures the boundedness of the closed-loop state of the RNN model in $\mathscr{U}_{\rho_{nn}}$ for the remaining time. With the impact of the RNN modeling error, bounded disturbances, and sample-and-hold implementation of control actions, it has been shown in Proposition 6.3 that when the closed-loop state of the RNN system is bounded in $\mathscr{U}_{\rho_{nn}}$, the actual state of the nonlinear system of Eq. 8.1 is ultimately bounded in $\mathscr{U}_{\rho_{min}}$. Furthermore, since the safe operating region $\mathscr{U}_\rho$ has no intersection with the unsafe region $\mathscr{D}$, the closed-loop state will be bounded in $\mathscr{U}_\rho$ for any $x_0 \in \mathscr{U}_\rho$, and thus will not enter $\mathscr{D}$ at all times.

In addition, avoiding convergence to saddle points needs to be considered. Saddle points are points in state-space at which the CLBF W_{nn} has a local minima. Starting from an initial condition $x_0 \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_{nn}}$, the constraint of Eq. 8.44e pulls the state towards the origin. When the closed-loop state reaches a neighborhood around the saddle point where $x(t_k) \in \mathscr{B}_\delta(x_e)$, the constraint of Eq. 8.44g will drive the state away from the neighborhood of saddle point in a direction of decreasing $W_{nn}(x)$. Once the state escapes $\mathscr{B}_\delta(x_e)$, then the constraints of Eqs. 8.44e-8.44f will ensure operational safety and closed-loop stability, and the closed-loop state ultimately converges to the origin and is bounded in $\mathscr{U}_{\rho_{min}}$. Therefore, the presence of saddle points have been addressed, and closed-loop stability and safety under the CLBF-MPC for the nonlinear system of Eq. 8.1 with bounded unsafe sets have been proven.

6.5 Application to a Chemical Process Example

In this section, we apply the proposed CLBF-MPC on a chemical process example. The process considered consists of a well-mixed, non-isothermal continuous stirred tank reactor (CSTR) where an irreversible first-order exothermic reaction $A \rightarrow B$ takes place. There is a heating jacket installed on the reactor to supply and remove heat. The material and energy balances of this CSTR system

is as follows:

$$\frac{dC_A}{dt} = \frac{F}{V_L}(C_{A0} - C_A) - k_0 e^{-E/RT} C_A \tag{6.21a}$$

$$\frac{dT}{dt} = \frac{F}{V_L}(T_0 - T) - \frac{\Delta H k_0}{\rho_L C_p} e^{-E/RT} C_A + \frac{Q}{\rho_L C_p V_L} \tag{6.21b}$$

where T is the temperature in the reactor, C_A represents the concentration of reactant A, Q is the heat rate, and V_L is the volume of the reacting liquid in the reactor. The reactor feed contains the reactant A at a concentration C_{A0}, temperature T_0, and volumetric flow rate F. ρ_L, C_p, k_0, E and ΔH are the liquid density, heat capacity, reaction pre-exponential factor, activation energy and the enthalpy of the reaction, respectively. Process parameter values can be found in [119]. The control objective is to operate the CSTR at the steady-state point $(C_{As},\ T_s) = (0.57\ kmol/m^3,\ 395.3\ K)$ and maintain the state in a safe region by manipulating the inlet concentration of species A, $\Delta C_{A0} = C_{A0} - C_{A0_s}$, and the heat input rate $\Delta Q = Q - Q_s$. The input constraints for ΔQ and ΔC_{A0} are $|\Delta Q| \leq 0.0167\ kJ/min$ and $|\Delta C_{A0}| \leq 1\ kmol/m^3$, respectively.

Deviation variables are used such that the equilibrium point of the system is at the origin of the state-space. $x^T = [C_A - C_{As}\ T - T_s]$, $u^T = [\Delta C_{A0}\ \Delta Q]$ represent the state vector and the manipulate input vector in deviation variable forms, respectively. As the focus of the current work is on the machine-learning construction of CBF and its application on an RNN-MPC, we do not consider bounded disturbances. Further simulations can be run with added disturbances to assess the robustness of the proposed control system.

We construct a Control Lyapunov Function using the standard quadratic form $V(x) = x^T P x$ with $P = \begin{bmatrix} 9.35 & 0.41 \\ 0.41 & 0.02 \end{bmatrix}$. The P matrix of the control Lyapunov function is determined via extensive closed-loop simulations of the process. With the goal of finding the largest stability and safety region in state space, we carry out an iterative search where we start with an initial guess of the P matrix, then find the region in state space within which the time derivative of CLBF can be rendered negative under the Sontag control law, and characterize the largest forward invariant set within this region to be considered as the stability and safety region. We define the unsafe region, $\mathscr{D}$, as a region embedded fully within the closed-loop system stability region. The unsafe region is located in the middle of the stability region such that the state trajectory will intersect the unsafe region on its converging route towards the origin. Such a bounded unsafe set poses both theoretical as well as implementation challenges for CLBF-MPC as the controller has to drive the state around the unsafe region and to the steady-state.

6.5.1 Development of the RNN Model for the CSTR Process

We follow similar procedures of data generation, training and validation as outlined in [123] to obtain an RNN model for the nonlinear process of Eq. 8.1. To generate training data sufficiently large to represent the entire operating region, open-loop simulations are run for finite sampling steps starting at various initial conditions within the safe and stabilizable set $\mathscr{U}_\rho$ with various control actions $u \in U$. The RNN model constructed takes the form of a Long-Short-Term-Memory network, which is a special kind of RNN known for its superior performance in remembering longer-intervaled temporal relationships. The RNN model uses one input layer, one hidden layer consisting of 20 recurrent units, and one output layer. State measurements $x(t_k)$ and the control actions $u(t_k)$ are the inputs to the RNN model, and the RNN model has the outputs of the predicted state trajectory over one sampling period $\hat{x}(t)$ for $t \in [t_k, t_{k+1}]$. The number of recurrent units in the hidden layer corresponds with the number of internal states within each sampling period. In our simulations, the time progression of states are simulated using an Euler integration method at an integration time step of h_c, and the sampling period of MPC is $\Delta = 100h_c$. In order to predict the states at the end of each sampling period, we could choose to have a maximum of 100 internal states, with a time interval of h_c between each internal state. In order to provide the RNN with sufficient neurons to achieve adequate accuracy and to also reduce computational effort, we have conducted a grid search between various numbers of internal states, and have chosen the design of 20 internal states with a time interval of $5h_c$ between each internal state. An early stopping criterion of achieving a validation mean squared error (MSE) of below 1×10^{-6} is implemented to avoid over-fitting and to ensure that the modeling error is rendered sufficiently small. After 65 epochs, early stopping is triggered and the obtained RNN model achieves a training MSE of 4.17 $\times 10^{-6}$ and a validation MSE of 9.03×10^{-7}.

6.5.2 Development of the FNN Model for Barrier Function

In this example, we define the unsafe region as follows: $\mathscr{D} := \{x \in \mathbf{R}^2 \mid F(x) = \frac{(x_1+0.22)^2}{1} + \frac{(x_2-4.6)^2}{1\times 10^4} < 2 \times 10^{-4}\}$. $\mathscr{H}$ is defined as $\mathscr{H} := \{x \in \mathbf{R}^2 \mid F(x) < 2.5 \times 10^{-4}\}$ such that it satisfies $\mathscr{D} \subset \mathscr{H} \subset \phi_{uc}$ in Proposition 8.1. We define the unsafe region to be an ellipse as an illustrative example of a challenging case of bounded unsafe set embedded in the operating region. In practice, the bounded unsafe set can be of any bounded form in state-space, and may not be easily described explicitly. For example, operating at certain mid-ranges of temperature and concentration could lead to material corrosion, incomplete reactions, or generation of byproducts from side reactions.

There are also circumstances where specific ranges of operation are sub-optimal to efficiency and productivity. To generate training data for the FNN model, we specify $\mathscr{H}' := \{x \in \mathbf{R}^2 \mid F(x) < 5.6 \times 10^{-4}\}$. The set of initial conditions considered $\mathscr{U}_\rho$ is characterized with $\rho = 0$ as per Eq. 8.31d. The CLBF $W_{nn}(x)$ is constructed with the following parameters: $c_1 = 0.001,\ c_2 = 10,\ c_3 = 48.269,\ c_4 = 16.85,\ \nu = \rho - c_1 c_4 = -1.685 \times 10^{-2}$, and $\mu = 5000$. The safe region, $\mathscr{U}_\rho \backslash \mathscr{H}'$, and the unsafe region $\mathscr{H}'$, are discretized into 18,000 data samples, respectively. The data samples are assigned a target label of "+1" if they belong in the unsafe region, and "-1" if they belong in the safe region. The FNN model is constructed with 2 hidden layers of 12 and 10 neurons respectively. The inputs to the FNN model are the state measurement vector, and the output of the FNN model gives the predicted class of the data point in state-space indicating whether it is located inside the safe or the unsafe region. Both hidden layers use an activation function of *tanh*, and the cost function of Eq. 8.6 has the following weighting parameters: $\alpha = 1.1,\ \beta = 0.005$. The validation metric examines the magnitude of $Cost_2$ of Eq. 8.6, and early stopping is triggered if $Cost_2$ has reached 0. After 700 epochs of training, the MSE ($Cost_1$) is 0.0155, and $Cost_2$ has a cumulative value of 0.4233. The classification accuracy over the testing dataset is 99.5%. The predicted $\hat{B}$ values are shown in Fig. 6.2, and the misclassified data points in the testing set are shown in Fig. 6.3. $Cost_2$ has not reached 0 as required by the algorithm within the specified number of epochs; however, the classification accuracy has reached an acceptable level to cease training. Then, the model can be assessed in terms of its misclassification rate, as well as the conditions on the values of $\hat{B}(x)$ and $L_{\hat{f}}\hat{B}(x)$ as specified by Eq. 8.3. There are 171 out of 36,000 data points being misclassifed, all of them are safe data classified as being unsafe. This does not cause any problems as the controller will simply be prompted to act sooner due to this misclassification when the closed-loop state approaches the boundary of the unsafe region. This also means that the predicated unsafe region given by the FNN-modeled CBF, denoted as $\hat{\mathscr{H}}$, is larger than $\mathscr{H}'$ as specified by the training data samples, therefore more conservative than what was intended. Moreover, to verify that the FNN model satisfies the safety conditions of CBF of Eq. 8.3, we verify that the tighter conditions hold for all discretized data points in their respective regions. It is shown that all predicted $\hat{B}(x) > 0.0197$ for all discretized x points in $\hat{\mathscr{H}}$, and the predicted $\hat{B}(x) < -0.0241$ for all discretized x points in $\mathscr{U}_\rho \backslash \hat{\mathscr{H}}$. Since $\hat{\mathscr{H}}$ is a superset of $\mathscr{D}$, it is proven that conditions Eq. 8.3a and Eq. 8.3c hold, respectively. We also examine $L_{\hat{f}}\hat{B}(x)$ values for all discretized x points outside of the unsafe region and where $L_{\hat{g}}\hat{B}(x) = 0$. Although both the FNN and the RNN models can be expressed in continuous forms, for simplicity, we use numerical approximation to calculate $L_{\hat{f}}\hat{B}(x)$ and $L_{\hat{g}}\hat{B}(x)$ respectively. Due to the dichotomous nature of $\hat{B}$

having nearly constant values close to $+1$ or -1, all discretized points in $x \in \mathscr{U}_\rho \backslash \mathscr{D}$ such that $L_{\hat{g}}\hat{B}(x) = 0$ have $L_{\hat{f}}\hat{B}(x) = 0$. Although we cannot conclude that $\forall\, x \in \{z \in \mathbf{R}^n \backslash \mathscr{D} \mid L_{\hat{g}}\hat{B}(x) = 0\}$, $L_{\hat{f}}\hat{B}(x) \leq 0$ holds, we know that the FNN model with a high accuracy can achieve $L_{\hat{f}}\hat{B}(x) = 0$ within the discretized region. Therefore, we proceed with this FNN model for CBF and apply it in the CLBF-MPC to assess its closed-loop performance.

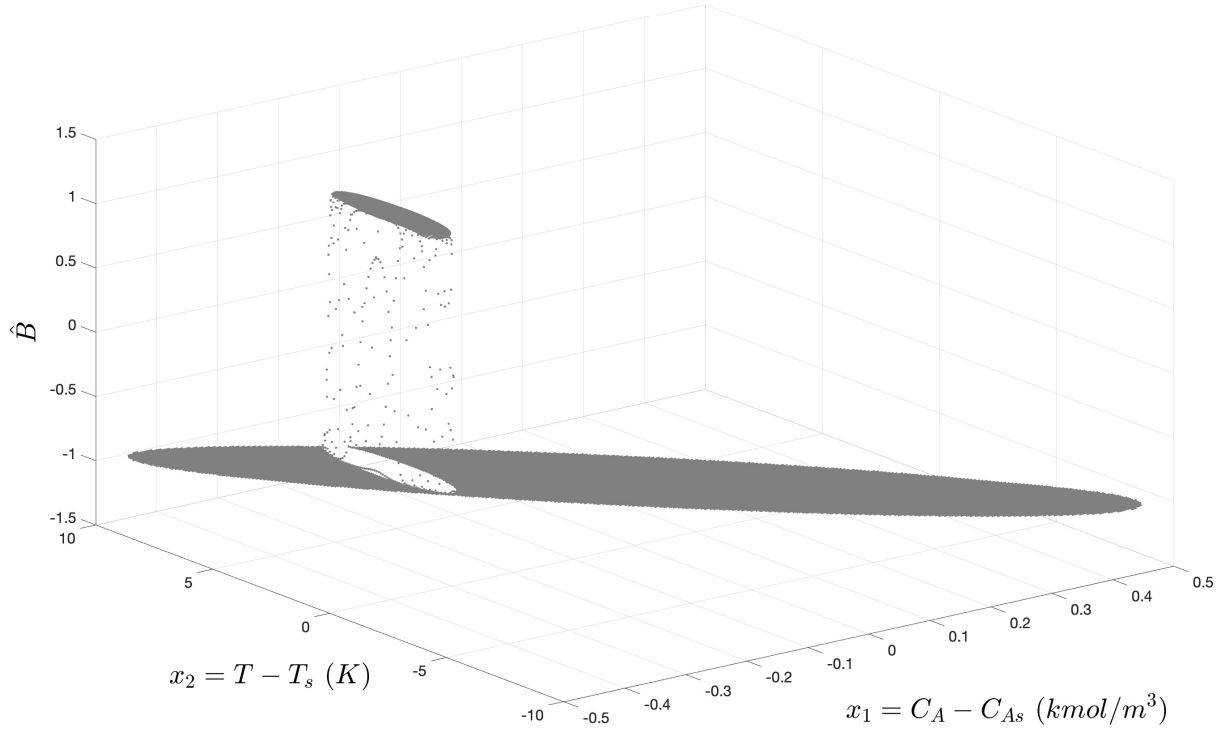


Figure 6.2: FNN-predicted barrier function $\hat{B}(x)$ for all data points in the training and the testing datasets.

Remark 6.5. *When training the FNN model, one may find that the weighting parameters α and β need to be chosen based on a grid-search approach as these two parameters indicate the trade-off between classification accuracy and enforcing $L_{\hat{f}}\hat{B}(x) \leq 0$ for all discretized data points in the safe region. In our simulations, striving for high classification accuracy whilst minimizing $Cost_2$ yielded a good model with its safety requirements met. In the case that the verification against the safety requirements of Eq. 8.3 are not met, the FNN needs to be retrained with more weighting on $Cost_2$ and more epochs.*

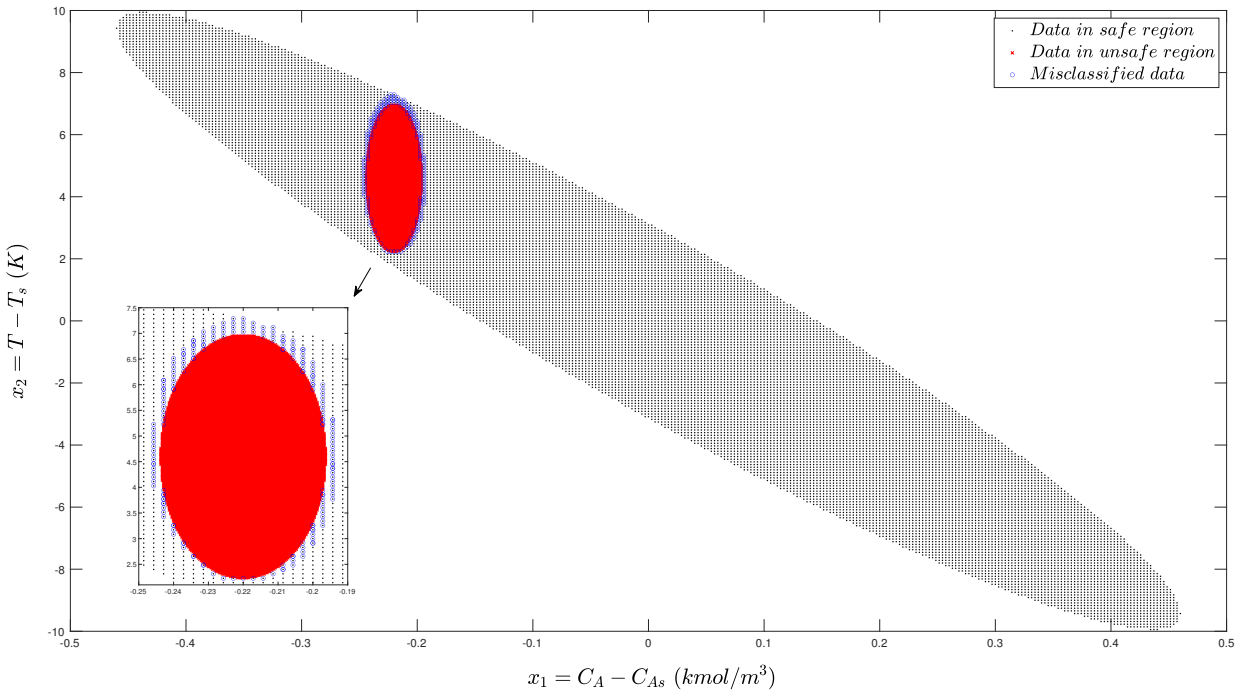


Figure 6.3: State values in the safe (black) and unsafe (red) operating regions, with misclassified data points (blue circles) showing that all inaccuracies are safe points misclassified as unsafe points.

6.5.3 Closed-loop Simulations

To demonstrate that the closed-loop state trajectory does not reach the unsafe region $\mathscr{D}$ when being driven towards the origin, we choose various initial conditions within $\mathscr{U}_\rho$ to start the simulation. It is demonstrated that the stabilization of the closed-loop system can be achieved when the simulation starts at an initial condition (x_1, x_2) = (0.18, -4.5), which is on the opposite side far from the unsafe region. More initial conditions near the unsafe region within the ellipse $\mathscr{U}_\rho$ are selected, from which the closed-loop state would have encountered the unsafe region if a conventional tracking controller were to be implemented. It is demonstrated that although the state enters the region $\mathscr{H}'$ due to inevitable modeling error within the FNN model for the CBF and the RNN model for the nonlinear process, the state never reaches the border of $\mathscr{D}$. Note that $\mathscr{D}$ represents the actual unsafe set in state-space from physical law, $\mathscr{H}$ is the closed and compact set that encloses $\mathscr{D}$, $\mathscr{H}'$ is the set within which training data collected are deemed as "unsafe". In addition, we use $\hat{\mathscr{H}}$ to denote the unsafe set predicted by $\hat{B}(x)$, which as shown in the previous section, encloses the unsafe set given by the training dataset $\mathscr{H}'$. All trajectories demonstrate that the states can successfully converge to the terminal set $\mathscr{U}_{\rho_{min}}$ while not entering the unsafe region

$\mathscr{D}$, as shown in Fig. 6.4.

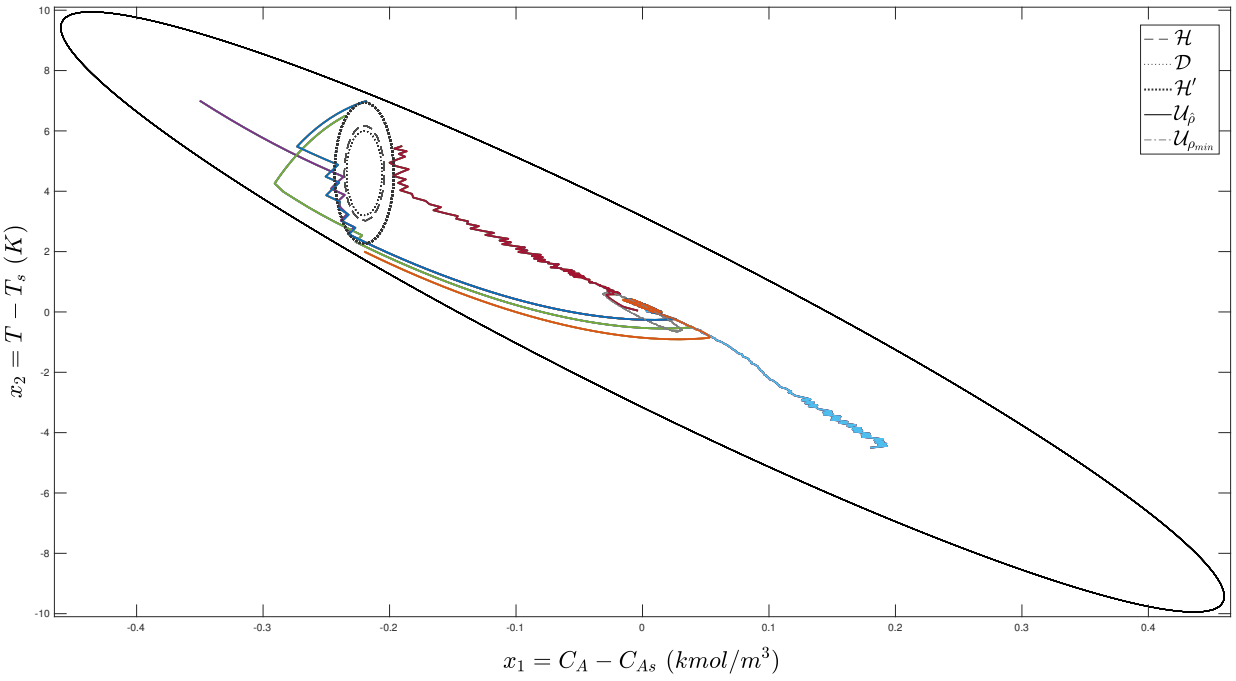


Figure 6.4: State trajectories originated from 6 different initial conditions in the safe operating regions under the closed-loop control of the CLBF-MPC using the RNN predictive model and the FNN-based CBF.

We also compare the closed-loop performance of the proposed machine-learning-based CLBF-MPC with other CLBF-MPC's with various levels of machine-learning implemented as part of the formulation. The trajectories are shown in Fig. 6.5. As shown, all trajectories successfully avoided the unsafe region, bounded in $\mathscr{U}_{\rho}$, and ultimately converged to $\mathscr{U}_{\rho_{min}}$. The trajectories using the analytical CBF, which is designed based on the region $\mathscr{H}$, enter and trespass the $\mathscr{H}'$ region (as they should in order to converge to the origin faster) while not entering the $\mathscr{H}$ region. The trajectories using the FNN-modeled CBF do oscillate around the boundary of the $\mathscr{H}'$ region due to modeling error, but remain distanced from the $\mathscr{H}$ region with the conservative contingency margin considered.

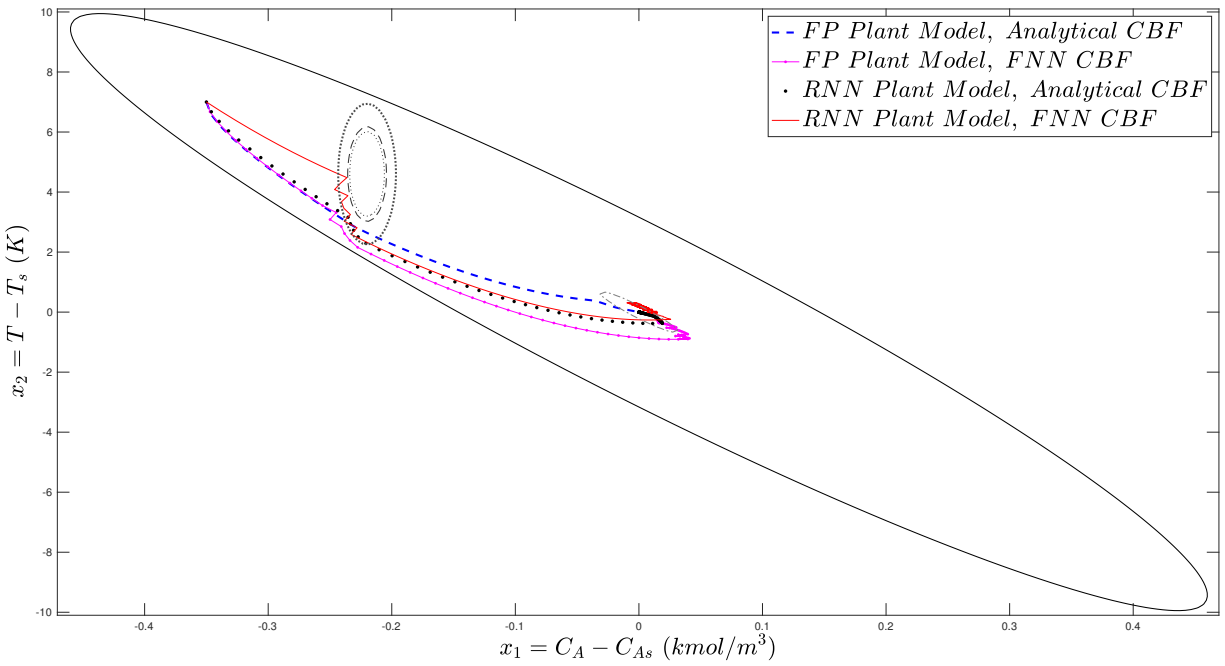


Figure 6.5: Closed-loop state trajectories under the CLBF-MPC using different combinations of first-principles (FP) process model or RNN process model, and analytical Control Barrier Function (CBF) or FNN-based CBF.

Chapter 7

Barrier-Function-Based Distributed Predictive Control for Operational Safety of Nonlinear Processes

This chapter focuses on the design of distributed model predictive control (DMPC) systems for nonlinear processes with input constraints using a Control Lyapunov-Barrier Function (CLBF) to achieve simultaneous closed-loop stability and process safety. Specifically, we first use a constrained CLBF to design explicit control laws for each subsystem and to characterize a set of initial conditions, starting from which the closed-loop states of the overall nonlinear system are guaranteed to converge to the operating steady-state under the CLBF-based control laws while avoiding unsafe regions in state space. We then propose the CLBF-based DMPC, and prove its feasibility and effectiveness in ensuring the stability and avoidance of unsafe regions under sample-and-hold implementation of DMPC control actions. The CLBF-based DMPC is applied to both sequential and iterative DMPC designs in the general sense, and a modification to the DMPC formulation is presented for special cases of systems where the coupling between subsystems is in a one-way cascading manner. The proposed CLBF-DMPC method is demonstrated via a nonlinear chemical process example consisting of two subsystems.

Process safety is inarguably a top priority in industrial engineering given the involvement of operators with potential hazards and exposure to the environment. During each stage of design, operation, and maintenance, risk assessment and analysis is an irreplaceable part of engineering and implementation in order to prevent catastrophic events from happening. Process control systems not only enable automated control, operation, and monitoring of the plant, but also allow

safe, stable, and optimal production if robust control designs are implemented. The work in [66] provides a control-inspired approach for the engineering of safe processes, and by defining process safety within a system-theoretic framework, allows for a comprehensive treatment of process safety.

CLBF-MPC has been proposed in [119, 122], where the stability and safety analysis for the closed-loop system in the presence of both bounded and unbounded unsafe sets have been provided. Many other recent works [76, 136] have also explored MPC with discrete-time control barrier functions, as well as optimal control based on reinforcement learning with the inclusion of control barrier functions.

In this work, we introduce CLBF to the design of DMPC in controlling multiple subsystems. This contribution is essential to the operation of complex industrial processes where the overall system may encounter regions in state-space for which they would like to avoid, and the sub-controllers for each subsystem need to work cooperatively to achieve the stability and safety objectives. In this work, we use an analytical representation of the unsafe operating points in state-space to specify the CLBFs. However, interested readers may also refer to previous works in [25] for machine-learning-based methods of characterizing such regions and designing MPC algorithms based on a feedforward-neural-network-based control barrier function. The unsafe operating regions may be specified for each subsystem individually, or if these unsafe points are interdependent across subsystems, the unsafe regions may be specified holistically with respect to the overall process.

The remainder of the paper is organized as follows. We address the class of systems considered, the stabilizability assumptions, and the definition of Control Lyapunov-Barrier Functions in Section 2. In Section 3, we provide the formulation of DMPCs, and develop a CLBF-based DMPCs that guarantee recursive feasibility, closed-loop stability and safety under the sample-and-hold control action implementation for the general case. We also provide a modified DMPC framework for special cases of coupled subsystems in order to demonstrate its advantages and drawbacks. In Section 4, we demonstrate the applicability of the proposed control scheme using a nonlinear chemical process example.

7.1 Preliminaries

7.1.1 Notation

We use $|\cdot|$ to denote the Euclidean norm of a vector. x^T denotes the transpose of x. If a function $f(\cdot)$ is continuously differentiable, it is of class $\mathscr{C}^1$. $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$ represents the Lie derivative. We say that a continuous scalar function $V : \mathbf{R}^n \to \mathbf{R}$ is a proper function, if the set $\{x \in \mathbf{R}^n \mid V(x) \leq k\}$ is a compact set $\forall\, k \in \mathbf{R}$. With positive real numbers β and ε, we use $\mathscr{B}_\beta(\varepsilon) := \{x \in \mathbf{R}^n \mid |x - \varepsilon| < \beta\}$ to represent an open ball around ε with radius of β. $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$ denotes set subtraction.

7.1.2 Class of Systems

A general class of nonlinear systems is considered in which multiple distinct sets of manipulated inputs exist. Each set of inputs regulates a specific subsystem. Throughout the manuscript, we consider two subsystems – subsystem-1 and subsystem-2 – for the simplicity of notation. Subsystem-1 and subsystem-2 consist of states x_1 and x_2 respectively, which are controlled by and only by u_1 and u_2 respectively. The general class of system under consideration can be represented by nonlinear ordinary differential equations as follows:

$$\begin{aligned} &\dot{x} = F(x, u_1, u_2, w) := f(x) + g_1(x)u_1 + g_2(x)u_2 + v(x)w, \\ &x(t_0) = x_0 \end{aligned} \tag{7.1}$$

where $x \in \mathbf{R}^n$ denotes the state vector, $u_1 \in \mathbf{R}^{m_1}$ and $u_2 \in \mathbf{R}^{m_2}$ are the two distinct sets of input vectors, and the disturbance is denoted by $w \in W$ with $W := \{w \in \mathbf{R}^r \mid |w| \leq w_m,\ w_m \geq 0\}$. There are constraints on the control actions as defined by $u_1 \in U_1 := \{u_{1_i}^{\min} \leq u_{1_i} \leq u_{1_i}^{\max}, i = 1, ..., m_1\} \subset \mathbf{R}^{m_1}$, and $u_2 \in U_2 := \{u_{2_i}^{\min} \leq u_{2_i} \leq u_{2_i}^{\max}, i = 1, ..., m_2\} \subset \mathbf{R}^{m_2}$. $f(\cdot)$, $g_1(\cdot)$, $g_2(\cdot)$, and $v(\cdot)$ are matrix and vector functions of dimensions $n \times 1$, $n \times m_1$, $n \times m_2$, and $n \times r$, respectively, which are assumed to be sufficiently smooth. The initial time t_0 is taken to be zero ($t_0 = 0$), and we assume that $f(0) = 0$. Therefore, the origin is an equilibrium point of the nominal system of Eq. 8.1 with $w(t) \equiv 0$ (i.e., $(x_s, u_{1s}, u_{2s}) = (0, 0, 0)$, where x_s, u_{1s} and u_{2s} represent the steady-state state and input vectors).

7.1.3 Control Lyapunov Function

With the nominal system of Eq. 8.1 with $w(t) \equiv 0$ in consideration, it is assumed that a Control Lyapunov Function (CLF) V exists, which is positive definite and proper; the CLF meets the small control property, which indicates that for every positive ε, there exists a positive δ, such that $\forall\ x \in \mathscr{B}_\delta(0)$, $\exists\ u^T = [u_1^T,\ u_2^T]$ that meet the conditions of $|u| < \varepsilon$ and $L_f V(x) + L_{g_1} V(x) \cdot u_1 + L_{g_2} V(x) \cdot u_2 < 0$ [98]. In addition, the CLF also satisfies the following conditions:

$$\begin{aligned} &L_f V(x) < 0, \\ &\forall x \in \{z \in \mathbf{R}^n \backslash \{0\} \mid L_{g_1} V(z) = 0,\ L_{g_2} V(z) = 0\} \end{aligned} \tag{7.2}$$

The existence of V implies the existence of explicit feedback control laws $\Phi_1(x) \in U_1,\ \Phi_2(x) \in U_2$ such that Eq. 7.2 holds for the nominal system of Eq. 8.1 under $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$, and for all x in an explicitly defined neighborhood around the origin, the closed-loop system is rendered asymptotically stable. The Sontag control law in [67] is one example of such stabilizing feedback control laws. A region ϕ_u can be characterized around the origin where the time derivative of the Lyapunov function $V(x)$ is negative under $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$ as: $\phi_u = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V(x) + L_{g_1} V(x) \cdot u_1 + L_{g_2} V(x) \cdot u_2 < 0, u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2\} \cup \{0\}$. Within ϕ_u, we define $\Omega_b := \{x \in \phi_u \mid V(x) \leq b,\ b > 0\}$, which is a level set of $V(x)$ and a forward invariant set. For the closed-loop system under $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$, Ω_b is considered as the stability region in the sense that, for any $x_0 \in \Omega_b$, the closed-loop trajectory $x(t),\ t \geq 0$ of the nominal system of Eq. 8.1 (i.e., $w(t) \equiv 0$) remains in Ω_b.

7.1.4 Control Barrier Function

During operation, there are undesirable regions within state-space that must be avoided for safety and/or other considerations related to cost, environment, and optimality. Let us assume that an open set $\mathscr{D}$ exists, and it sufficiently describes the region to be avoided. In the remainder of the manuscript, the notation $\mathscr{D}$ is used to represent the unsafe set. A safe set can be subsequently defined as $\mathscr{X}_0 := \{x \in \mathbf{R}^n \backslash \mathscr{D}\}$ where $\mathscr{X}_0 \cap \mathscr{D} = \emptyset$, $\{0\} \in \mathscr{X}_0$. $\mathscr{X}_0$ will include the set of initial conditions that we consider. Both bounded and unbounded unsafe regions have been studied in literature; in this manuscript, bounded unsafe set is denoted as $\mathscr{D}_b$, unbounded unsafe set is denoted as $\mathscr{D}_u$, respectively.

The definition of process operational safety studied in this manuscript entails closed-loop states not entering any unsafe sets. Formally, operational safety has a definition described as follows:

Definition 7.1. *The nominal system of Eq. 8.1 under input constraints $u \in U$ and with $w(t) \equiv 0$ is considered. If a set of constrained control actions $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$ exists such that, for any initial state $x(t_0) = x_0 \in \mathscr{X}_0$, the process state trajectories do not enter the unsafe region and converge to the origin asymptotically (i.e., $x(t) \in \mathscr{X}_0,\ x(t) \notin \mathscr{D},\ \forall\ t \geq 0$), then the control actions $u_1 = \Phi_1(x),\ u_2 = \Phi_2(x)$ are able to maintain the closed-loop state within a safe operating region $\mathscr{X}_0$ at all times.*

Subsequently, with the introduction of safe and unsafe operating regions in state-space, we can define a valid Control Barrier Function (CBF) in the following definition [116]:

Definition 7.2. *With a set of unsafe points $\mathscr{D}$ in state-space, a $\mathscr{C}^1$ function $B(x) : \mathbf{R}^n \to \mathbf{R}$ is a Control Barrier Function if it satisfies these properties:*

$$B(x) > 0, \quad \forall\, x \in \mathscr{D} \tag{7.3a}$$

$$L_f B(x) \leq 0,\ \forall\, x \in \{z \in \mathbf{R}^n \backslash \mathscr{D} \mid L_g B(z) = 0\} \tag{7.3b}$$

$$\mathscr{X}_B := \{x \in \mathbf{R}^n \mid B(x) \leq 0\} \neq \emptyset \tag{7.3c}$$

7.2 Stabilization and Safety via Control Lyapunov-Barrier Function

The work in [90] proposed a Control Lyapunov-Barrier Function (CLBF) and proved that if a valid CLBF exists for the nominal system of Eq. 8.1, then for any initial condition $x_0 \in \mathscr{X}_0$, a control law exists which maintains the closed-loop state outside of $\mathscr{D}$ and within an explicitly characterized region around the steady-state (which is a level set of CLBF) at all times. In [119, 122], this work is extended to including constraints on the manipulated inputs $u \in U$ in the design of CLBFs. In all three works, the CLBF was designed using a weighted sum of a CBF and a CLF, where the CBF satisfies the properties outlined in Eq. 8.3, and the CLF meets the relevant conditions in Section 8.1.3. Then, a practical design guideline is presented in [119] to construct this CLBF. We can reference and utilize the same guidelines, applied on the nonlinear system of Eq. 8.1 consisting of multiple subsystems to design the CLBF for the overall process.

The definition of a constrained CLBF $W(x)$ with respect to the overall process as represented by the nonlinear model of Eq. 8.1 is shown as below:

Definition 7.3. *Considering an unsafe set in state-space $\mathscr{D}$, a lower-bounded, proper, and $\mathscr{C}^1$ function $W(x) : \mathbf{R}^n \to \mathbf{R}$ is a constrained CLBF if $W(x)$ satisfies the following properties and has*

a minimum at the origin:

$$W(x) > \rho, \ \ \forall\, x \in \mathscr{D} \subset \phi_{uc} \tag{7.4a}$$

$$\begin{aligned} &L_f W(x) < 0, \\ &\forall\, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathscr{X}_e) \mid L_{g_1} W(z) = 0, \\ &\qquad L_{g_2} W(z) = 0\} \end{aligned} \tag{7.4b}$$

$$\mathscr{U}_\rho := \{x \in \phi_{uc} \mid W(x) \leq \rho\} \neq \emptyset \tag{7.4c}$$

$$\overline{\phi_{uc} \backslash (\mathscr{D} \cup \mathscr{U}_\rho)} \cap \overline{\mathscr{D}} = \emptyset \tag{7.4d}$$

where $\mathscr{X}_e := \{x \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\}) \mid \frac{\partial W(x)}{\partial x} = 0\}$ represents a set of states where $L_f W(x) = 0$ (for $x \neq 0$) due to $\frac{\partial W(x)}{\partial x} = 0$. $\rho \in \mathbf{R}$ is a real constant. f, g_1, g_2 are the vector and matrix functions from Eq. 8.1. Using a set of explicit control laws subject to their lower and upper bounds $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$, ϕ_{uc} is defined to be the union of the origin, the set $\mathscr{X}_e$, and the set where the time-derivative of $W(x)$ is negative: $\phi_{uc} = \{0\} \cup \mathscr{X}_e \cup \{x \in \mathbf{R}^n \mid \dot{W}(x(t), \Phi_1(x), \Phi_2(x)) = L_f W + L_{g_1} W \cdot u_1 + L_{g_2} W \cdot u_2 < -\alpha_W |W(x) - W(0)|,\ u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2,\ \alpha_W > 0\}$. For the nominal system of Eq. 8.1 with $w(t) \equiv 0$, if a $\mathscr{C}^1$ constrained CLBF $W(x)$ exists, then there exists a set of control laws $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$ that together render the origin asymptotically stable within ϕ_{uc}. The CLBF function has a minimum at the origin and is able to satisfy the following properties $\forall\, x \in \phi_{uc}$:

$$c_1 |x|^2 \leq W(x) - \rho_0 \leq c_2 |x|^2, \tag{7.5a}$$

$$\frac{\partial W(x)}{\partial x} F(x, \Phi_1(x), \Phi_2(x)) \leq -c_3 |x|^2, \ \forall\, x \in \phi_{uc} \backslash \mathscr{B}_\delta(x_e) \tag{7.5b}$$

$$\left| \frac{\partial W(x)}{\partial x} \right| \leq c_4 |x| \tag{7.5c}$$

where $F(x, u_1, u_2)$ is the nominal system of Eq. 8.1 with $w(t) \equiv 0$, $c_j(\cdot) > 0$, $j = 1,\ 2,\ 3,\ 4$ are real numbers, ρ_0 represents the global minimum of $W(x)$ at the origin (i.e., $W(0) = \rho_0$), and $\mathscr{B}_\delta(x_e)$ denotes a neighborhood surrounding a saddle point in state-space, $x_e \in \mathscr{X}_e$.

Within the nonlinear system described by Eq. 8.1, the functions f, g_1, g_2 and v are assumed to be sufficiently smooth, thus positive constants L_x, L_w, $L_x^{'}, L_w^{'}$, and M exist (by continuity) s.t.

$\forall\, x, x' \in \mathscr{U}_\rho$, $w \in W$, and $u_1 \in U_1,\ u_2 \in U_2$, the conditions below will hold:

$$|F(x,u_1,u_2,w)| \leq M \tag{7.6a}$$

$$|F(x,u_1,u_2,w) - F(x',u_1,u_2,0)| \leq L_x|x-x'| + L_w|w| \tag{7.6b}$$

$$\left|\frac{\partial W(x)}{\partial x}F(x,u_1,u_2,w) - \frac{\partial W(x')}{\partial x}F(x',u_1,u_2,0)\right| \leq L_x^{'}|x-x'| + L_w^{'}|w_m| \tag{7.6c}$$

Remark 7.1. *When designing local controllers, we could consider designing a CLBF for individual subsystems $W_j(x_j)$, where x_j are the states of the subsystem j. We can characterize the region ϕ_{uc_j} for each subsystem j, which includes the set for which under a set of constrained control laws $u_j = \Phi_j(x) \in U_j$, the CLBF satisfies $\dot{W}_j(x_j, \Phi_j(x_j)) < -\alpha_{W_j}|W_j(x_j) - W_j(0)|$. However, in the context of DMPC where multiple controllers work collaboratively to achieve a collective control objective of guaranteeing safety and stability for each subsystem, some initial conditions may result in trajectories where the control objective for each controller in the DMPC network may lead to a conflict. For example, one controller may encounter an unsafe region for its respective local subsystem and attempts to navigate the closed-loop states away from the unsafe region, consequently increasing the Lyapunov function $V(x)$ for the overall system. On the other hand, another controller may be attempting to navigate the process state of the other subsystem to the origin, thereby decreasing $V(x)$ for the overall system. Therefore, the set of initial conditions for which the conditions of Eq. 7.5 are satisfied will only be a subset of the combination of sets for which the CLBF conditions on $W_j(x_j)$ are satisfied for each subsystem individually. In other words, if the stability and safety region for each subsystem j is defined with respect to $W_j(x_j)$ as $\mathscr{U}_{\rho_j},\ j = 1,2$, then $\mathscr{U}_\rho \subseteq (\mathscr{U}_{\rho_1} \cup \mathscr{U}_{\rho_2})$.*

An example of such CLBF-based controllers $\Phi_1(x) \in U_1 \subset \mathbf{R}^{m_1}$ and $\Phi_2(x) \in U_2 \subset \mathbf{R}^{m_2}$, is given as follows:

$$\phi_{j_i}(x) = \begin{cases} -\dfrac{p + \sqrt{p^2 + q^4}}{q^T q} q & \text{if} \quad q \neq 0 \\ 0 & \text{if} \quad q = 0 \end{cases} \tag{7.7a}$$

$$\Phi_{j_i}(x) = \begin{cases} u_{j_i}^{min} & \text{if} \quad \phi_{j_i}(x) < u_{j_i}^{min} \\ \phi_{j_i}(x) & \text{if} \quad u_{j_i}^{min} \leq \phi_{j_i}(x) \leq u_{j_i}^{max} \\ u_{j_i}^{max} & \text{if} \quad \phi_{j_i}(x) > u_{j_i}^{max} \end{cases} \tag{7.7b}$$

where $j = 1,2$ represents the two candidate controllers for the two subsystems, p denotes $L_f W(x)$ where $f = [f_1 \cdots f_n]^T$, and q denotes $L_{g_{j_i}} W(x)$, where $g_{j_i} = [g_{j_{i1}}, ..., g_{j_{in}}]^T$, ($i = 1,2,...,m_1$ for $j = 1$

corresponding to $\Phi_1(x)$, and $i = 1,2,...,m_2$ for $j = 2$ corresponding to $\Phi_2(x)$.) $\phi_{j_i}(x)$ of Eq. 7.7a denotes the i_{th} component of the control action $\phi_j(x)$. After accounting for the input constraints $u_j \in U_j$, $\Phi_{j_i}(x)$ of Eq. 7.7 represents the i_{th} component of the saturated control law $\Phi_j(x)$.

7.2.1 Design of Constrained CLBF

The set $\mathscr{U}_\rho$, as defined in Eq. 8.31d, is a forward invariant set of $W(x)$, and it will define the set of initial conditions we consider in the rest of the manuscript. We analyze the scenario of bounded unsafe sets first following similar logic as presented in Theorem 1 in [122], and present theoretical analysis on the closed-loop stability and safety for the nonlinear system of Eq. 8.1. Specifically in the case of bounded unsafe regions, stationary points in addition to the origin $x_e \in \mathscr{X}_e$ are present in state-space which can be considered as saddle points. The continuous control actions $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$ are not able to help the states escape the stationary points once the states reach them. Therefore, discontinuous control actions $u_1 = \bar{u}_1(x) \in U_1,\ u_2 = \bar{u}_2(x) \in U_2$ that decrease $W(x)$ are designed, where $[\bar{u}_1(x), \bar{u}_2(x)] \neq [\Phi_1(x), \Phi_2(x)]$, to navigate x away from these stationary points along the direction of $W(x)$ decreasing. On the other hand, in the presence of unbounded unsafe sets, the only unique stationary point in state-space is the origin, therefore $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$ are able to render the origin asymptotically stable and simultaneously guaranteeing process safety. More work on handling bounded and unbounded unsafe sets are detailed in [122] for further reference.

7.3 CLBF-Based Control Law

7.3.1 Effect of Bounded Disturbance and Sample-and-hold Implementation of Control Actions

As the CLBF will be used in the design of DMPC, which implements its control actions in a sample-and-hold manner, we need to consider the impact of sample-and-hold control as well as the presence of disturbances in the nonlinear system of Eq. 8.1 which are sufficiently small and bounded when analyzing stability and safety properties of the closed-loop system. We provide proof for these considerations in the next proposition.

Proposition 7.1. *Consider the nominal system of Eq. 8.1 with $w(t) \equiv 0$ and a constrained CLBF $W(x)$ that meets the requirements of Definition 8.6 and has a minimum at the origin. Subsequently,*

we characterize the set of initial conditions $\mathscr{U}_\rho \subset \mathscr{X}_0$. Let $u_1(t) = \Phi_1(x(t_k)) \in U_1,\ u_2(t) = \Phi_2(x(t_k)) \in U_2$ for all $t_k \le t < t_{k+1}$, $x(t_k) \in \mathscr{U}_\rho \backslash \mathscr{B}_\delta(x_e)$ where $\delta > 0$, $x_e \in \mathscr{X}_e$ and t_k represents the time stamp $t = k\Delta$, $k = 0,1,2,...$, and $u_1(t) = \bar{u}_1(x) \in U_1,\ u_2(t) = \bar{u}_2(x) \in U_2$ such that if $x(t_k) \in \mathscr{B}_\delta(x_e)$, then $W(x(t_{k+1})) < W(x(t_k))$ for any $\Delta > 0$. Then, there exists a real valued Δ^*, such that, if $\Delta \in (0, \Delta^*]$ and $x_0 \in \mathscr{U}_\rho$, then $x(t) \in \mathscr{U}_\rho$, and $\lim_{t\to\infty} |x(t)| \le d$, for any positive real number d.

Proof. For any initial conditions in $\mathscr{U}_\rho$, we will first show that $x(t)$ converges to a terminal level set around the origin $\mathscr{U}_{\rho_{min}} := \{x \in \phi_{uc} \mid W(x) \le \rho_{min}\}$ as $t \to \infty$ where $\rho_{min} < \rho$. Then by the continuity of $W(x)$, we prove that $\lim_{t\to\infty} |x(t)| \le d$ as $t \to \infty$. First, we consider the case when $x(t_k) \in \mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_s} \cup \mathscr{B}_\delta(x_e))$, where $\rho_s < \rho_{min} < \rho$, and demonstrate that $\dot{W}(x(t), u_1(t), u_2(t)) < -\varepsilon$ holds in the set $\mathscr{Z} := \{x \in \phi_{uc} \backslash \mathscr{B}_\delta(x_e) \mid \rho_s \le W(x) \le \rho\}$ under $u_1(t) = u_1(t_k) = \Phi_1(x(t_k))$, $u_2(t) = u_2(t_k) = \Phi_2(x(t_k))$, $\forall\ t \in [t_k, t_k + \Delta^*)$. The time derivative of the CLBF can be represented as follows:

$$
\begin{aligned}
&\dot{W}(x(t), u_1(t), u_2(t)) \\
&\quad = \dot{W}(x(t_k), u_1(t_k), u_2(t_k)) + (\dot{W}(x(t), u_1(t), u_2(t)) \\
&\quad\ - \dot{W}(x(t_k), u_1(t_k), u_2(t_k)) \\
&\quad = L_f W(x(t_k)) \\
&\quad\ + L_{g_1} W(x(t_k)) u_1(t_k) + L_{g_2} W(x(t_k)) u_2(t_k) \\
&\quad\ + (L_f W(x(t)) - L_f W(x(t_k))) \\
&\quad\ + (L_{g_1} W(x(t)) - L_{g_1} W(x(t_k))) u_1(t) \\
&\quad\ + (L_{g_2} W(x(t)) - L_{g_2} W(x(t_k))) u_2(t)
\end{aligned}
\tag{7.8}
$$

Since $W(x)$ is a $\mathscr{C}^1$ function that satisfies Eq. 8.31, and $f(\cdot)$, $g_1(\cdot)$ and $g_2(\cdot)$ are assumed to be smooth, there exist positive real numbers k_f, k_{g_1} and k_{g_2}, such that $|(L_f W(x(t)) - L_f W(x(t_k))| \le k_f |x(t) - x(t_k)|$, $|(L_{g_1} W(x(t)) - L_{g_1} W(x(t_k))) u_1(t))| \le k_{g_1} |x(t) - x(t_k)|$, $|(L_{g_2} W(x(t)) - L_{g_2} W(x(t_k))) u_2(t))| \le k_{g_2} |x(t) - x(t_k)|$. In addition, since $f(x)$, $g_1(x)$ and $g_2(x)$ are continuous, and $\mathscr{Z}$ is bounded, there exists a positive real number k_s and a sampling period Δ' such that $|x(t) - x(t_k)| \le k_s \Delta'$ for all $t \in [t_k, t_k + \Delta')$. According to how ϕ_{uc} is defined, it is given that $\dot{W}(x(t_k)) < -\alpha_W |W(x) - W(0)| < -\alpha_W \rho_m$ holds for all $x \in \mathscr{Z}$, where $\rho_m := \min_{x \in \mathscr{Z}} |W(x) - W(0)|$. We choose $\Delta' < \frac{\alpha_W \rho_m - \varepsilon}{k_s (k_f + k_{g_1} + k_{g_2})}$ and $0 \le \varepsilon < \alpha_W \rho_m$, where α_W is used to characterize ϕ_{uc}. Using these inequalities derived from Lipschitz conditions, Eq. 8.42 can be

written as:

$$\begin{aligned}\dot{W}(x(t),u_1(t),u_2(t)) &\leq \dot{W}(x(t_k),u_1(t_k),u_2(t_k)) \\ &\quad + k_s(k_f+k_{g_1}+k_{g_2})\Delta' \\ &< -\alpha_W\rho_m + k_s(k_f+k_{g_1}+k_{g_2})\Delta' \\ &< -\varepsilon\end{aligned} \tag{7.9}$$

which implies that for any initial conditions in $\mathscr{U}_\rho$, $W(x(t)) < W(x(t_k)) \leq \rho, \forall\ t > t_k$ and the closed-loop state $x(t)$ will enter a terminal set $\mathscr{U}_{\rho_s}$ within finite steps. We have proven that $x(t)$ is bounded in $\mathscr{U}_\rho\ \forall t \in [t_k, t_k+\Delta')$.

In addition, we discuss the case where the closed-loop state is in the neighborhood of saddle points, $x(t_k) \in \mathscr{B}_\delta(x_e)$ where x_e are saddle points. Since $\bar{u}_1(x), \bar{u}_2(x)$ are a set of control actions that decrease $W(x)$, as a result, $W(x(t_{k+1})) < W(x(t_k))$ as $x(t_{k+1})$ moves to a smaller level set of $W(x)$ and the closed-loop state eventually leaves $\mathscr{B}_\delta(x_e)$ within finite time steps. After it leaves the saddle point neighborhood, $x(t)$ will not come back to $\mathscr{B}_\delta(x_e)$ as Eq. 8.43 (i.e., $W(x(t)) < W(x(t_k)), \forall\ t > t_k$) holds thereafter.

We will now address the effect of sample-and-hold control and bounded disturbance on the convergence and boundedness of the closed-loop state. First, we will show that given $x(t_k) \in \mathscr{U}_{\rho_s}$, the trajectory of $x(t)$ will stay in $\mathscr{U}_{\rho'_{min}}, \forall\ t \in [t_k, t_k+\Delta'')$. Consider Δ'' such that

$$\rho'_{min} = \max_{\Delta\in[0,\Delta'')} \{W(x(t_k+\Delta)) \mid x(t_k) \in \mathscr{U}_{\rho_s},\ u \in U\}. \tag{7.10}$$

There exists a sufficiently small Δ'' such that Eq. 7.10 holds. Therefore, let $\Delta^* = \min\{\Delta', \Delta''\}$, and we have shown that for any $x(t_k) \in \mathscr{U}_{\rho_s}$, the closed-loop state $x(t)$ under sample-and-hold control implementation will remain in $\mathscr{U}_{\rho'_{min}}$ during one sampling period $\Delta \in (0, \Delta^*]$. When taking the bounded disturbance $|w(t)| \leq w_m$ into account and the CLBF-based controller applied in a sample-and-hold fashion, we can show that Proposition 7.1 still holds for the system of Eq. 8.1 subject to the bounded disturbance. Given the local Lipschitz property of $v(\cdot)$, we can derive the following inequality for $L_vW(x)$: $\exists\ k_d > 0$, s.t. $|(L_vW(x(t)) - L_vW(x(t_k))| \leq k_d|x(t) - x(t_k)|$. Therefore, similar results can be shown for $\dot{W}(x(t),u_1(t),u_2(t))$ and ρ_{min} that account for $w(t)$ as follows:

$$\begin{aligned}\dot{W}(x(t),u_1(t),u_2(t)) &\leq \dot{W}(x(t_k),u_1(t_k),u_2(t_k)) \\ &\quad + k_s(k_f+k_{g_1}+k_{g_2}+k_dw_m)\Delta' \\ &< -\alpha_W\rho_m + k_s(k_f+k_{g_1}+k_{g_2}+k_dw_m)\Delta' \\ &< -\varepsilon\end{aligned} \tag{7.11}$$

$$\rho_{min} = \max_{\Delta \in [0, \Delta'')} \{W(x(t_k + \Delta), u_1, u_2, w) \mid x(t_k) \in \mathscr{U}_{\rho_s}, \\ u \in U,\ |w| \leq w_m\}. \tag{7.12}$$

where $\Delta' < \frac{\alpha_W \rho_m - \varepsilon}{k_s(k_f + k_{g_1} + k_{g_2} + k_d w_m)}$ and $0 \leq \varepsilon < \alpha_W \rho_m$, respectively. Hence, when sufficiently small bounded disturbance $|w| \leq w_m$ is present, $\dot{W} < 0$ still holds within each sampling period if Δ' and ε are chosen. Furthermore, if $x(t_k) \in \mathscr{B}_\delta(x_e)$, the discontinuous control laws $\bar{u}_1(x), \bar{u}_2(x)$ are assumed to exist and satisfy $W(x(t_{k+1})) < W(x(t_k))$, $\forall\ |w| \leq w_m$. By the definition of ρ_{min} of Eq. 7.12, it is shown that for any $x(t_k) \in \mathscr{U}_{\rho_s}$, the trajectory of $x(t)$ will stay in $\mathscr{U}_{\rho_{min}}, \forall\ t \in [t_k, t_k + \Delta'')$. The proof above shows the robustness of the CLBF-based control law against the sample-and-hold execution in the presence of sufficiently small bounded disturbances, and serves as an underlying foundation for proving that the CLBF-DMPC is also robust to sample-and-hold control execution and bounded disturbances.

7.4 CLBF-DMPC Formulations and Analysis

DMPC has proven to provide improved computational time and closed-loop control performance, where some level of communication may be established between the different controllers. In this framework, two separate MPCs are designed to compute control actions u_1 and u_2 respectively; the control law trajectories computed by MPC-1 and MPC-2 are denoted by u_{d_1} and u_{d_2}, respectively. In the following few sections, we will discuss a sequential distributed MPC design and an iterative distributed MPC design; interested readers may refer to [24, 27] for other distributed and decentralised MPC architectures.

7.4.1 Sequential Distributed MPC System

Between two MPCs in a sequential DMPC structure, the communication is one-way. In other words, the set of the optimal control laws calculated by one MPC will be relayed to the other MPC, which will utilize this additional knowledge to optimize its corresponding set of control laws. In a sequential DMPC framework, the following implementation strategy is used:

1. At each sampling instant $t = t_k$, sensor measurements on the states $x(t),\ t = t_k$ are sent to both MPC-1 and MPC-2.

2. The optimal trajectory of u_{d_1} is calculated by MPC-1 and sent to MPC-2, and the first value of the input trajectory $u^*_{d_1}(t_k)$ is sent to the corresponding actuators.

3. MPC-2 calculates the optimal trajectory of u_{d_2} based on state measurement $x(t)$ at $t = t_k$ and the optimal trajectory of u_{d_1} received from MPC-1, then sends the first optimal control action over the next sampling period $u^*_{d_2}(t_k)$ to the corresponding control actuators.

4. At the next sampling instance, when an updated state measurement is available ($k \leftarrow k+1$), go to Step 1.

In the calculation of MPC-1, it first assumes a trajectory for u_{d_2} along the prediction horizon, which is computed using the explicit nonlinear CLBF-based control law, $\Phi_2(x)$. In addition, we incorporate a contractive constraint in the optimization problem of the MPC in order to ensure that u_{d_1} will inherit the stability and safety properties of $\Phi_j(x),\ j = 1,2$, and decrease the CLBF $W(x)$ at a minimum rate of that of the CLBF-based control laws $\Phi_j(x),\ j = 1,2$. The optimization problem of MPC-1 is given as follows:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t)))dt \tag{7.13a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t))) \tag{7.13b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.13c}$$

$$u_{d_1}(t) \in U_1,\ \forall\, t \in [t_k, t_{k+N}) \tag{7.13d}$$

$$\dot{W}(x(t_k), u_{d_1}(t_k), \Phi_2(x(t_k))) \leq \dot{W}(x(t_k), \Phi_1(x(t_k)), \Phi_2(x(t_k))), \tag{7.13e}$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.13f}$$

$$W(\tilde{x}(t)) \leq \rho_s,\ \forall\, t \in [t_k, t_{k+N}),\ \text{if } W(x(t_k)) \leq \rho_s \tag{7.13g}$$

$$W(\tilde{x}(t)) < W(x(t_k)),\ \forall\, t \in (t_k, t_{k+N}),\ \text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.13h}$$

where $S(\Delta)$ is the set of piece-wise constant functions with sampling period Δ, N is the number of sampling periods in the prediction horizon, and $\tilde{x}$ represents the predicted state trajectory. The optimal input trajectory calculated over the prediction horizon $t \in [t_k, t_{k+N})$ by MPC-1 is denoted as $u^*_{d_1}(t)$. The cost function of Eq. 7.13a is the integral of $L(\tilde{x}(t), u_{d_1}(t), \Phi_2(t))$ over the prediction horizon; here, $L(x, u_1, u_2)$ typically takes on a quadratic form, i.e., $L(x, u_1, u_2) = x^T Q x + u_1^T R_1 u_1 + u_2^T R_2 u_2$, where Q, R_1, and R_2 are positive definite weighting matrices. The minimum value of the objective function of Eq. 7.13a is at the origin. The constraint of Eq. 7.13b is

the nominal system of Eq. 8.1 with $w(t) \equiv 0$ and predicts the closed-loop state trajectory. Eq. 7.13d defines the input variable constraints on u_{d_1}. The initial condition $\tilde{x}(t_k)$ of Eq. 7.13b is taken as the state sensor measurement at $t = t_k$ defined in Eq. 7.13c. The constraints of Eqs. 7.13f-7.13h are activated depending on the location of the process state in state-space, and they work together to make certain of operational safety and stability. When $x(t_k) \notin \mathscr{B}_\delta(x_e)$ and $W(x(t_k)) > \rho_s$, the constraint Eq. 7.13f ensures that $W(\tilde{x})$ decreases at least as fast as the rate achieved by the CLBF-based control laws $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$. If $W(x(t_k)) \leq \rho_s$, the constraint of Eq. 7.13g maintains the predicted state within $\mathscr{U}_{\rho_s}$, so that in the presence of sufficiently bounded disturbances in the nonlinear system of Eq. 8.1, the closed-loop state still remains in $\mathscr{U}_{\rho_{min}}$. Furthermore, if $x(t_k)$ enters a neighborhood of a saddle point $\mathscr{B}_\delta(x_e)$, the constraint Eq. 7.13h ensures that $W(x)$ decreases over the predicted trajectory, and with decreasing $W(x)$, the closed-loop process state can eventually escape x_e within finite steps. Once the state escapes from the saddle points $\mathscr{B}_\delta(x_e)$, the constraint of Eq. 7.13f will drive it towards the origin into smaller level sets of the CLBF $W(x)$, thus guaranteeing the state will not return to $\mathscr{B}_\delta(x_e)$ thereafter. Each time MPC-1 is executed, it communicates the entire trajectory of $u^*_{d_1}(t),\ t \in [t_k, t_{k+N})$ to MPC-2 and sends the first value of the input trajectory $u^*_{d_1}(t_k)$ to its actuators. The horizon rolls one sampling time step forward while the above optimization problem is solved again.

MPC-2 computes control actions u_{d_2} based on the latest received state measurement, and in addition, the control action computed by MPC-1 (i.e., $u^*_{d_1}(t), \forall t \in [t_k, t_{k+N})$). By utilizing the optimal input trajectory of MPC-1 as well as the CLBF-based control law $\Phi_2(x(t_k))$, the closed-loop performance is optimized while guaranteeing that the stability and safety properties of the CLBF-based control laws are preserved. Specifically, MPC-2 calculates the following optimization problem:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u^*_{d_1}(t), u_{d_2}(t))dt \tag{7.14a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u^*_{d_1}(t), u_{d_2}(t)) \tag{7.14b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.14c}$$

$$u_{d_2}(t) \in U_2, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.14d}$$

$$\dot{W}(x(t_k), u^*_{d_1}(t_k), u_{d_2}(t_k)) \tag{7.14e}$$

$$\leq \dot{W}(x(t_k), u^*_{d_1}(t_k), \Phi_2(x(t_k))),$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.14f}$$

$$W(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W(x(t_k)) \leq \rho_s \tag{7.14g}$$

$$W(\tilde{x}(t)) < W(x(t_k)), \ \forall\, t \in (t_k, t_{k+N}),$$

$$\text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.14h}$$

The notation and the explanation of the optimization problem of MPC-2 are akin to that of MPC-1 and will be omitted here for brevity. To account for the total computation time for the sequential DMPC framework, one would add the times taken to solve each MPC problem respectively, since the solution of MPC-2 depends on the MPC-1 results.

7.4.2 Iterative Distributed MPC System

The communication between two MPCs in an iterative framework is two-ways. The optimal control actions calculated by each MPC are exchanged to better predict future states, and the optimization problem in each MPC is solved independently in a parallel structure until an iteration criterion has been met. The implementation strategy is as follows:

1. MPC-1 and MPC-2 receive the state sensor measurement $x(t)$ at $t = t_k$ at each sampling instant t_k.

2. At iteration $c = 1$, MPC-1 calculates $u_{d_1}(t)$ over the prediction horizon assuming $u_2(t) = \Phi_2(t), \forall t \in [t_k, t_{k+N})$. MPC-2 calculates $u_{d_2}(t)$ over the prediction horizon assuming $u_1(t) = \Phi_1(t), \forall t \in [t_k, t_{k+N})$. The future trajectories of $u_{d_1}(t)$ and $u_{d_2}(t)$ are exchanged between the two MPCs, and each MPC calculates and stores the value of its own cost function.

3. At iteration $c > 1$:

(a) Based on state measurement $x(t_k)$ as well as the latest input trajectories received from the other MPC, each MPC evaluates its own future input trajectory again.

(b) The MPCs cross-communicate their newest calculated future input trajectories. Each MPC computes then stores the value of its cost function.

4. If a termination criterion is met, each MPC selects the input trajectory corresponding to the smallest cost function value, and sends the first control action of this optimal trajectory to its actuators. If the termination criterion is not satisfied, go to Step 3 ($c \leftarrow c+1$).

5. At the next sampling instance, when an updated state measurement is available, go to Step 1 ($k \leftarrow k+1$).

The optimization problem of MPC-1 in an iterative distributed LMPC at iteration $c = 1$ is presented as follows. Readers may refer to the formulations of sequential DMPC design for detailed definitions of the same variables and constraints.

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t)))dt \tag{7.15a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t))) \tag{7.15b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.15c}$$

$$u_{d_1}(t) \in U_1, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.15d}$$

$$\dot{W}(x(t_k), u_{d_1}(t_k), \Phi_2(x(t_k))) \leq \dot{W}(x(t_k), \Phi_1(x(t_k)), \Phi_2(x(t_k))), \tag{7.15e}$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.15f}$$

$$W(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W(x(t_k)) \leq \rho_s \tag{7.15g}$$

$$W(\tilde{x}(t)) < W(x(t_k)), \ \forall\, t \in (t_k, t_{k+N}), \text{ if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.15h}$$

At iteration $c = 1$, the optimization problem of MPC-2 is shown as follows:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), \Phi_1(\tilde{x}(t)), u_{d_2}(t)) dt \tag{7.16a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), \Phi_1(\tilde{x}(t)), u_{d_2}(t)) \tag{7.16b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.16c}$$

$$u_{d_2}(t) \in U_2, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.16d}$$

$$\dot{W}(x(t_k), \Phi_1(x(t_k)), u_{d_2}(t_k)) \tag{7.16e}$$

$$\leq \dot{W}(x(t_k), \Phi_1(x(t_k)), \Phi_2(x(t_k))),$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.16f}$$

$$W(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W(x(t_k)) \leq \rho_s \tag{7.16g}$$

$$W(\tilde{x}(t)) < W(x(t_k)), \ \forall\, t \in (t_k, t_{k+N}),$$

$$\text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.16h}$$

At iteration $c > 1$, after the optimized input trajectories $u_{d_1}^*(t)$ and $u_{d_2}^*(t)$ have been exchanged between the two MPCs, the optimization problem of MPC-1 becomes:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), u_{d_2}^*(t)) dt \tag{7.17a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_1}(t), u_{d_2}^*(t)) \tag{7.17b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.17c}$$

$$u_{d_1}(t) \in U_1, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.17d}$$

$$\dot{W}(x(t_k), u_{d_1}(t_k), u_{d_2}^*(t)) \tag{7.17e}$$

$$\leq \dot{W}(x(t_k), \Phi_1(x(t_k)), \Phi_2(x(t_k))),$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.17f}$$

$$W(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W(x(t_k)) \leq \rho_s \tag{7.17g}$$

$$W(\tilde{x}(t)) < W(x(t_k)), \ \forall\, t \in (t_k, t_{k+N}),$$

$$\text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.17h}$$

And the optimization problem of MPC-2 becomes:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u^*_{d_1}(t), u_{d_2}(t))dt \tag{7.18a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u^*_{d_1}(t), u_{d_2}(t)) \tag{7.18b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.18c}$$

$$u_{d_2}(t) \in U_2, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.18d}$$

$$\dot{W}(x(t_k), u^*_{d_1}(t_k), u_{d_2}(t)) \tag{7.18e}$$

$$\leq \dot{W}(x(t_k), \Phi_1(x(t_k)), \Phi_2(x(t_k))),$$

$$\text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } W(x(t_k)) > \rho_s \tag{7.18f}$$

$$W(\tilde{x}(t)) \leq \rho_s, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W(x(t_k)) \leq \rho_s \tag{7.18g}$$

$$W(\tilde{x}(t)) < W(x(t_k)), \ \forall\, t \in (t_k, t_{k+N}),$$

$$\text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{7.18h}$$

Since the two MPCs in an iterative framework can be simultaneously solved in a parallel structure using separate processors, the total computation time would equal to the maximum time of the two MPCs including all iterations taken until termination of iterations. The total number of iterations would depend on the termination criterion. Some examples of these criteria may include, the total iterations must not exceed a maximum threshold, $c \leq c_{max}$; the computation time each MPC takes must not surpass a time threshold; between two consecutive iterations, the difference in the cost function value or the computed trajectory of control actions must be sufficiently small.

Remark 7.2. *In this work, we have presented the formulations and simulations of DMPC systems in the case of two subsystems (and thus, two controllers) for simplicity of notation, but the results are conceptually straightforward and can be similarly extended to the case of N_{sys} subsystems having N_{sys} controllers in total.*

Once solving both optimization problems of MPC-1 and MPC-2 is complete, the proposed CLBF-DMPC provides the optimal control actions in the following form:

$$\begin{aligned} u_1(t) &= u^*_{d_1}(t_k), \forall t \in [t_k, t_{k+1}) \\ u_2(t) &= u^*_{d_2}(t_k), \forall t \in [t_k, t_{k+1}) \end{aligned} \tag{7.19}$$

The control actions computed by each MPC will be applied in a sample-and-hold manner to the nonlinear process of Eq. 8.1 with bounded disturbances.

We will now demonstrate that, for the nonlinear system of Eq. 8.1, stability and safety can be established under the CLBF-based DMPC system with the theorem and proof below. Note that the proof is written with respect to the sequential DMPC, but the same concept can be applied to the iterative DMPC as well.

Theorem 7.1. *Consider the system described by Eq. 8.1, and it has a constrained CLBF $W(x)$ that satisfies Eq. 8.31 with its minimum value at the origin. Given any initial condition $x_0 \in \mathscr{U}_\rho$, the CLBF-DMPC optimization problems of Eq. 7.13 – 7.14 are guaranteed to have recursive feasibility for all times, and under the sample-and-hold implementation of CLBF-DMPC control actions $[u_1 \;\; u_2] = [u_{d_1}^* \;\; u_{d_2}^*]$, $x(t)$ is bounded in $\mathscr{U}_\rho$ for all $t \geq 0$, and as $t \to \infty$, converges to $\mathscr{U}_{\rho_{min}}$.*

Proof.

Part 1: In Eq. 7.13 – 7.14, the optimization problems of the CLBF-DMPC have feasible solutions at all times, and this is because the CLBF-DMPC constraints of Eqs. 7.13d, 7.13h, and Eqs. 7.14d, 7.14h can be met respectively by the sample-and-hold implementation of control actions $u_1 = \bar{u}_1(x) \in U_1,\ u_2 = \bar{u}_2(x) \in U_2,\ \forall x \in \mathscr{B}_\delta(x_e)$ and $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2,\ \forall x \in \mathscr{U}_\rho \backslash \mathscr{B}_\delta(x_e)$. By letting $u_1(t_k) = \Phi_1(x(t_k)),\ u_2(t_k) = \Phi_2(x(t_k))$ when $x(t_k) \in \mathscr{U}_\rho \backslash (\mathscr{B}_\delta(x_e) \cup \mathscr{U}_\rho)$, Eq. 7.13f and Eq. 7.14f are also satisfied. In Proposition 7.1, we have shown that once x is driven inside $\mathscr{U}_{\rho_s}$ by the CLBF-based control laws $u_1 = \Phi_1(x) \in U_1,\ u_2 = \Phi_2(x) \in U_2$, it will not exit $\mathscr{U}_{\rho_{min}}$ within one sampling period for any $u_1 \in U_1,\ u_2 \in U_2$. Therefore, the CLBF-based control laws are able to provide a feasible solution for the input trajectories and satisfy the constraints of Eq. 7.13g and Eq. 7.14g. Lastly, as the controller $u_1 = \bar{u}_1(x) \in U_1,\ u_2 = \bar{u}_2(x) \in U_2$ are able to satisfy $W(x(t_{k+1})) < W(x(t_k))$, the control action $u_j(t) = \bar{u}_j(x(t_{k+i})) \in U_j$, for $j = 1,2,\ \forall\ t \in [t_{k+i}, t_{k+i+1})$ with $i = 0,...,N-1$ will satisfy the constraints of Eq. 7.13h and Eq. 7.14h and eventually navigate the states away from the stationary saddle points if $x(t_k) \in \mathscr{B}_\delta(x_e)$.

Part 2: We will now demonstrate simultaneous stability and safety for the nonlinear system of Eq. 8.1 can be guaranteed under the optimized solutions of Eq. 7.13 – 7.14. For any $x_0 \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_s}$, the constraints of Eq. 7.13f and Eq. 7.14f ensure that the CLBF-DMPC control actions $u_{d_1}^*$, and sequentially $u_{d_2}^*$, are optimized to decrease the value of the CLBF and will drive x towards the origin; the closed-loop state x will eventually enter $\mathscr{U}_{\rho_s}$ within finite sampling steps.

After x enters $\mathscr{U}_{\rho_s}$, the constraints of Eq. 7.13g and Eq. 7.14g ensure the boundedness of the closed-loop state in $\mathscr{U}_{\rho_{min}}$ for the remaining time considering the impact of sample-and-hold control and the presence of bounded disturbance. As the safe operating region $\mathscr{U}_\rho$ does not intersect with the unsafe region $\mathscr{D}$, x will not enter $\mathscr{D}$ for all times and will remain inside $\mathscr{U}_\rho$ for any

$x_0 \in \mathscr{U}_\rho$. With $x_0 \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_s}$, the constraints of Eq. 7.13f and Eq. 7.14f pull the state towards the origin. The constraint of Eq. 7.13h and Eq. 7.14h will be activated when x arrives at a saddle point neighborhood, i.e., $x(t_k) \in \mathscr{B}_\delta(x_e)$; x will be driven away from $\mathscr{B}_\delta(x_e)$ in the direction of $W(x)$ decreasing. After it leaves from $\mathscr{B}_\delta(x_e)$, the DMPC constraints of Eqs. 7.13f-7.13g and Eqs. 7.14f-7.14g will take over and continue to ensure closed-loop safety and stability thereafter; ultimately, the closed-loop state converges towards the origin and is bounded in $\mathscr{U}_{\rho_{min}}$. Thus, closed-loop stability and safety under the sample-and-hold implementation of CLBF-DMPC for the nonlinear system of Eq. 8.1 with sufficiently bounded disturbance in the presence of bounded unsafe sets have been shown.

7.4.3 Modified DMPC Structure in Special Cases

In many industrial processes, there are examples where the process variables of an upstream sub-process impact the dynamics of a downstream sub-process, but not vice versa. In these cases where the first subsystem is independent and the second subsystem is dependent on the first subsystem, we can design the DMPC with some special considerations to assess whether safety and stability can be simultaneously guaranteed. We use sequential DMPC as an example. Since the first subsystem is completely independent, its contractive constraint of Eq. 7.13f can be modified to only account for the CLBF of subsystem-1 $\dot{W}_j,\ j = 1$, and therefore only depends on the states and inputs of subsystem-1, $x_j(t_k),\ u_j$ where $j = 1$. In doing so, MPC-1 can guarantee the stability and safety of the upstream process, subsystem-1. The contractive constraint of Eq. 7.14f can also be similarly modified to account for the CLBF function of subsystem-2 only, $\dot{W}_j,\ j = 2$, where $\dot{W}_2(x_1, x_2, u_2)$ can be simplified to be a function of x_1 of subsystem-1, x_2 of subsystem-2, and u_2 of subsystem-2. This leads to a modified formulation of the DMPC system with simpler computation complexity. However, since the constraints to each controller are only with respect to its own subsystem, one caveat to this modification is that the state measurements of subsystem-1 will be treated as disturbances in the computation of MPC-2. Therefore, as we will demonstrate with a nonlinear example in Section 8.5, there are values of of states for subsystem-1 that may result in non-negative values of $\dot{W}_2$ for subsystem-2.

The DMPC formulation for such processes can be modified for improved computation time and algorithm simplicity. Using sequential DMPC as an example, the optimization problem of

MPC-1 is as follows:

$$\mathscr{J} = \min_{u_{d_1} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t)))dt \tag{7.20a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_1}(t), \Phi_2(\tilde{x}(t))) \tag{7.20b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.20c}$$

$$u_{d_1}(t) \in U_1, \ \forall\, t \in [t_k, t_{k+N}) \tag{7.20d}$$

$$\dot{W}_1(x_1(t_k), u_{d_1}(t_k)) \tag{7.20e}$$

$$\leq \dot{W}_1(x_1(t_k), \Phi_1(x(t_k))),$$

$$\text{if } x_1(t_k) \notin \mathscr{B}_{\delta_1}(x_{1e}) \text{ and } W_1(x_1(t_k)) > \rho_{s_1} \tag{7.20f}$$

$$W_1(\tilde{x}_1(t)) \leq \rho_{s_1}, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } W_1(x_1(t_k)) \leq \rho_{s_1} \tag{7.20g}$$

$$W_1(\tilde{x}_1(t)) < W_1(x_1(t_k)), \ \forall\, t \in (t_k, t_{k+N}),$$

$$\text{if } x_1(t_k) \in \mathscr{B}_{\delta_1}(x_{1e}) \tag{7.20h}$$

where the level sets of $W_j(x_j)$ will be respectively defined with positive real constants $\rho_{s_j},\ j = 1,2$, and the neighborhood of saddle points present in the stability and safety region of the subsystem-j will be correspondingly denoted as $\mathscr{B}_{\delta_j}(x_{je})$.

The predicted state trajectory calculation of subsystem-2 $\tilde{x}_2$ relies on the state measurements of subsystem-1 $x_1(t_k)$. The calculation of $\dot{W}_2$ will need the measurements of $x_1(t_k)$ in addition to $x_2(t_k)$ and $u_2(t_k)$. Note that the full state vector is the combination of states of subsystems 1 and 2, i.e., $x(t_k)^T = [x_1(t_k)^T, x_2(t_k)^T]$. Therefore, the formulation of the modified MPC-2 in a sequential

distributed design is as follows:

$$\mathscr{J} = \min_{u_{d_2} \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u_{d_1}^*(t), u_{d_2}(t))dt \tag{7.21a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u_{d_1}^*(t), u_{d_2}(t)) \tag{7.21b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.21c}$$

$$u_{d_2}(t) \in U_2, \ \forall \ t \in [t_k, t_{k+N}) \tag{7.21d}$$

$$\dot{W}_2(x(t_k), u_{d_2}(t_k)) \leq \dot{W}_2(x(t_k), \Phi_2(x(t_k))), \tag{7.21e}$$

$$\text{if } x_2(t_k) \notin \mathscr{B}_{\delta_2}(x_{2e}) \text{ and } W_2(x_2(t_k)) > \rho_{s_2} \tag{7.21f}$$

$$W_2(\tilde{x}_2(t)) \leq \rho_{s_2}, \ \forall \ t \in [t_k, t_{k+N}), \text{ if } W_2(x_2(t_k)) \leq \rho_{s_2} \tag{7.21g}$$

$$W_2(\tilde{x}_2(t)) < W_2(x_2(t_k)), \ \forall \ t \in (t_k, t_{k+N}), \text{ if } x_2(t_k) \in \mathscr{B}_{\delta_2}(x_{2e}) \tag{7.21h}$$

A similar modification can be applied to iterative DMPC structures. Following the MPC-1 calculation, closed-loop stability and safety can be guaranteed for subsystem-1 using the CLBF-based constraints on W_1. Since subsystem-2 is dependent on the states of subsystem-1 and since the calculation of MPC-2 is carried out with CLBF-based constraints on W_2 only, there may exist points in state space where the explicit control law $\Phi_2(x(t_k))$ can no longer guarantee a negative $\dot{W}_2(x(t_k), \Phi_2(x(t_k)))$ given the impact from the state measurements of subsystem-1 $x_1(t_k)$. When this happens, the optimizer for MPC-2 may still attempt to find a set of input trajectory $u_{d_2}^*$ that decreases W_2, i.e., $\dot{W}_2(x(t_k), u_{d_2}(t_k)) < 0$. Alternatively, MPC-2 can opt to use the discontinuous control actions $\bar{u}_2(x) \in U_2$, which ensures the existence of a solution for $u_{d_2}^*$ that will decrease $W_2(\tilde{x}_2)$ along the predicted trajectory.

Remark 7.3. *With respect to the use of a Barrier function to express safety specifications, it is important to note that the Barrier Function plays essentially the role of a safety constraint (i.e., temperature needs to be below a certain value or a nonlinear function of several state variables needs to be within a certain range) but it is implemented within the MPC scheme in a way that ensure that the closed-loop state does not enter the unsafe set (i.e., region of increased risk) in a guaranteed manner (simply, setting up a constraint in the MPC to require that the closed-loop state does not enter in a certain unsafe region cannot ensure that such an excursion of the closed-loop state to the unsafe set will not occur). Regions in state-space where the system trajectory may*

be allowed to enter, but due to increasing safety risk the state should not stay there long, can be formulated as soft constraints in MPC, and have been studied in past works [138]. Such soft constraints can be added in the DMPC framework presented in this work as they do not lead to infeasibility of the MPC optimization problem. In the case where the closed-loop state is allowed to enter the unsafe region for a particular subsystem, the amount of time a controller takes to return the closed-loop system state to a safe region is related to the speed of the closed-loop response under the CLBF-based DMPC, which can be tuned by adjusting the weights in the MPC cost function. Finally, it is important to note that if there exist model uncertainties and process disturbances that lead to process/model mismatch, the proposed DMPC provides a robustness margin to sufficiently small bounded disturbances through the negative margin in the Lyapunov function time-derivative, which is a consequence of the use of measurement feedback in MPC at each sampling time.

7.5 Application to a Nonlinear Chemical Process

We will demonstrate the proposed CLBF-DMPC method on a chemical process example, which consists of two well-mixed, non-isothermal continuous stirred tank reactors (CSTRs) in series. An irreversible second-order exothermic reaction takes place in each reactor that transforms a reactant A to a product B ($A \rightarrow B$). Each CSTR is fed with reactant material A with the inlet concentration C_{Aj0}, the inlet temperature T_{j0} and feed volumetric flow rate of the reactor F_{j0}, $j = 1,2$, where $j = 1$ denotes the first CSTR and $j = 2$ denotes the second CSTR. The reactors are equipped with heating jackets to remove/supply heat at a rate Q_j, $j = 1,2$. This system can be modelled by the following material and energy balance equations:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_1}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{7.22a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_1} \tag{7.22b}$$

$$\frac{dC_{B1}}{dt} = -\frac{F_{10}}{V_1} C_{B1} + k_0 e^{\frac{-E}{RT_1}} C_{A_1}^2 \tag{7.22c}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_2} C_{A20} + \frac{F_{10}}{V_2} C_{A1} - \frac{F_{10} + F_{20}}{V_2} C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{7.22d}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2} T_{20} + \frac{F_{10}}{V_2} T_1 - \frac{F_{10} + F_{20}}{V_2} T_2 + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_2} \tag{7.22e}$$

$$\frac{dC_{B2}}{dt} = \frac{F_{10}}{V_2} C_{B1} - \frac{F_{10} + F_{20}}{V_2} C_{B2} + k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{7.22f}$$

where Q_j, V_j, C_{Aj}, T_j, where $j = 1,2$, are the heat input rate, the volume of the reacting liquid, concentration of reactant A, and the temperature in the first and the second reactor, respectively. ΔH, E, k_0, and R represent the enthalpy of the reaction, activation energy, pre-exponential constant, and ideal gas constant, respectively. All process parameter values can be found in Table 7.1.

Table 7.1: Values and descriptions of process parameters and steady-states of state and input variables.

Parameter/Value	Description
$F_{10}, F_{20} = 5\ m^3/hr$	Feed flow rate of CSTR 1 & 2
$T_{10} = 300\ K$, $T_{20} = 300\ K$	Feed temperatures of CSTR 1 & 2
$V_1 = 1.0\ m^3$, $V_2 = 1.0\ m^3$	Volume of reacting liquid in CSTR 1 & 2
$k_0 = 8.46 \times 10^6\ h^{-1}$	Pre-exponential constant
$E = 5.0 \times 10^4\ kJ/kmol$	Activation energy
$\Delta H = -1.15 \times 10^4\ kJ/kmol$,	Enthalpy of reaction
$C_p = 0.231\ kJ/(kg\ K)$	Heat capacity
$R = 8.314\ kJ/(kmol\ K)$	Gas constant
$\rho = 1000\ kg/m^3$	Liquid solution density
$C_{A10_s} = 4kmol/m^3, C_{A20_s} = 4kmol/m^3$	Inlet concentration steady-state values
$Q_{1_s} = 0kJ/hr, Q_{2_s} = 0kJ/hr$	Heat input rate steady-state values
$C_{A1_s} = 1.9537kmol/m^3, C_{A2_s} = 1.9537kmol/m^3$	Concentration of reactant A steady-state values
$T_{1_s} = 401.9K, T_{2_s} = 401.9K$	Temperature steady-state values

The manipulated inputs for both CSTRs are the inlet concentration of species A and the heat input rate, which are in deviation variable representations $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1,2$, respectively. The manipulated inputs have their respective lower and upper bounds: $|\Delta C_{Aj0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q_j| \leq 5 \times 10^5\ kJ/hr$, $j = 1,2$. The states of the two-CSTR system are $x^T = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}\ \ C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]$, where C_{A1_s}, C_{A2_s}, T_{1_s} and T_{2_s} are the steady-state values of concentration of A and temperature in the two reactors, such that the operating steady-state and equilibrium of the nonlinear system is at the origin of the state-space. States of the CSTR-1 can be separately denoted as $[x_1, x_2] = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}]$ and the states of the CSTR-2 are denoted as $[x_3, x_4] = [C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]$. In a distributed MPC framework, both MPCs have knowledge of full-state measurements as well as the overall plant model of the two-CSTR process. Feedback measurements on $x(t)$ are received by both MPCs, where MPC-1 optimizes $[u_1, u_2] = [\Delta C_{A10}\ \ \Delta Q_1]$ and MPC-2 optimizes $[u_3, u_4] = [\Delta C_{A20}\ \ \Delta Q_2]$. The common control objective of the two MPCs is to stabilize the two-CSTR process at the unstable operating steady-state $x_s^T = [C_{A1_s}\ \ C_{A2_s}\ \ T_{1_s}\ \ T_{2_s}] = [1.9537kmol/m^3,\ \ 1.9537kmol/m^3,\ \ 401.9K,\ \ 401.9K]$. To numerically simulate the dynamic ODE model of Eq. 8.45, we use the explicit Euler method

with an integration time step of $h_c = 10^{-5}\ hr$. We demonstrate our simulations with the sequential DMPC framework. The nonlinear optimization problems of the sequential DMPC of Eq. 7.20 – Eq. 7.21 are calculated every sampling period $\Delta = 10^{-3}\ hr$ using the Python module of the IPOPT software package [113]. The objective function in the DMPC optimization problem has the form $L(x,u_1,u_2) = x^T Qx + u_1^T R_1 u_1 + u_2^T R_2 u_2$, where $Q = diag[2\times 10^3 \quad 1 \quad 2\times 10^3 \quad 1]$, $R_1 = R_2 = diag[8\times 10^{-13} \quad 0.001]$; the same objective function is used in both MPC-1 and MPC-2. Due to the special structure of the nonlinear process studied, where the first CSTR is completely independent of the second CSTR, we can adopt the modified DMPC design in Eq. 7.20 – Eq. 7.21. In this manuscript, we present the simulation results of a sequential DMPC; however, the same closed-loop performance can be similarly demonstrated with an iterative DMPC.

We first consider a bounded unsafe region $\mathscr{D}_b$, which is embedded fully in the closed-loop system stability region, and is located in the middle of the stability region, as shown in Fig. 7.1. This is so that the state will encounter this unsafe set on its trajectory as it converges towards the origin if no safety control is considered. It is challenging to handle such unsafe sets for the CLBF-DMPC as the closed-loop state needs to be driven around the unsafe set, towards the steady-state thereafter, and ultimately bounded in a neighborhood around the steady-state. In this work, we consider an ellipsoid described as $\mathscr{D}_b := \{x \in \mathbf{R}^4 \mid h_1(x) = (x_1+0.92)^2 + \frac{(x_2-42)^2}{500} < 0.06,\ h_2(x) = (x_3+0.92)^2 + \frac{(x_4-42)^2}{500} < 0.06\}$. By following the design method in [122], we can define the set $\mathscr{H}$ which encloses $\mathscr{D}$ as $\mathscr{H} := \{x \in \mathbf{R}^4 \mid h_1(x) \leq 0.07, h_2(x) \leq 0.07\}$. Then, the CBF $B_j(x),\ j = 1,2$ can be constructed as follows:

$$B_j(x) = \begin{cases} e^{\frac{h_j(x)}{h_j(x)-0.07}} - e^{-6}, & \text{if } x \in \mathscr{H} \\ -e^{-6}, & \text{if } x \notin \mathscr{H} \end{cases} \tag{7.23}$$

From Eq. 7.23, it is guaranteed that $B(x)$ is positive in the unsafe region $\mathscr{D}$. The overall CLBF is the sum of the CLBFs for the two CSTRs, i.e., $W(x) = W_1(x) + W_2(x) = V_1(x) + V_2(x) + \mu(B_1(x) + B_2(x)) + \nu$ where $V_1(x) = x_1^T P_1 x_1$ and $V_2(x) = x_2^T P_2 x_2$. $\mathscr{U}_\rho$, which is safe operating region and the set of valid initial conditions, is defined with $\rho = 0$ as per Eq. 8.31d. $W(x)$ is designed using $\nu = -340$, $\mu = 1\times 10^9$, which are selected based on the design method in [122], and the following positive definite P matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{7.24}$$

Similarly, we also study the scenario of unbounded unsafe region, which is defined as $\mathscr{D}_u := \{x \in$

$\mathbf{R}^4 \mid h_1(x) = x_1 + x_2 > 47,\ h_2(x) = x_3 + x_4 > 47\}$. The enclosing compact set $\mathscr{H}$ is defined as $\mathscr{H} := \{x \in \mathbf{R}^4 \mid h_1(x) > 45, h_2(x) > 45\}$, and the corresponding CBFs for the two subsystems $B_j(x),\ j = 1,2$ are shown as follows:

$$B_j(x) = \begin{cases} e^{h_j(x)-47} - 2 \times e^{-2}, & \text{if } x \in \mathscr{H} \\ -e^{-2}, & \text{if } x \notin \mathscr{H} \end{cases} \tag{7.25}$$

The CLBF $W(x)$ for the unbounded unsafe region is constructed with $\nu = -0.104$ and $\mu = 5000$.

Closed-loop simulations are run starting from various initial conditions of the two CSTRs inside the safety and stability regions under two scenarios: (1) in the presence of bounded, and (2) unbounded unsafe sets. The state trajectories of both CSTRs under CLBF-DMPC for cases of bounded and unbounded unsafe sets are shown in Fig. 7.1 and Fig. 7.2 respectively. These initial conditions are chosen to cover various points in state-space where the control problem becomes challenging to solve. For example, both CSTRs start at an initial condition very close to the boundary of the unsafe set, but at different positions such that the directions of state evolution may be different; one CSTR may start from the side of the unsafe set and the other CSTR may start from the side without the unsafe set, such that one MPC drives the closed-loop state of its respective subsystem around the unsafe ellipse, and the other MPC drives the closed-loop state of its subsystem towards the origin at optimal rate. It is demonstrated that the closed-loop system achieve stability while successfully avoiding the unsafe regions in both CSTRs when the simulation starts at the illustrated five initial conditions inside their respective regions $\mathscr{U}_{\rho_1}$ for CSTR-1 and $\mathscr{U}_{\rho_2}$ for CSTR-2, and eventually converges and is bounded in their respective terminal sets $\mathscr{U}_{\rho_{min_1}}$ and $\mathscr{U}_{\rho_{min_2}}$. This is shown for both scenarios of bounded and unbounded unsafe sets.

Note that we have selected initial conditions of the two CSTRs inside their respective stability and safety regions, $\mathscr{U}_{\rho_1}$ and $\mathscr{U}_{\rho_2}$, and the stability and safety region for the overall system $\mathscr{U}_{\rho}$ should be a subset of the union of the two individual sets, $\mathscr{U}_{\rho} \subseteq (\mathscr{U}_{\rho_1} \cup \mathscr{U}_{\rho_2})$. As it is difficult to have a closed-form representation of $\mathscr{U}_{\rho}$, the characterization of $\mathscr{U}_{\rho}$ can be carried out through numerical simulation to first find a region for which $\dot{W}(x, \Phi_1(x), \Phi_2(x)) < 0$, and then find the largest level set of $W(x)$ within this region.

The stability and safety region $\mathscr{U}_{\rho_1}$ for CSTR-1 can be characterized through numerical simulation by assessing $\dot{W}_1(x_1(t_k), \Phi_1(x_1(t_k)))$ and using information on the states of CSTR-1 itself. To rigorously characterize the stability and safety region $\mathscr{U}_{\rho_2}$ for CSTR-2, discretized points in state-space for which $\dot{W}_2(x(t_k), \Phi_2(x(t_k))) < 0$ need to be assessed first. However, since $x(t_k)$ also includes the $x_1(t_k)$, the characterization of $\mathscr{U}_{\rho_2}$ cannot be done without considering the process state

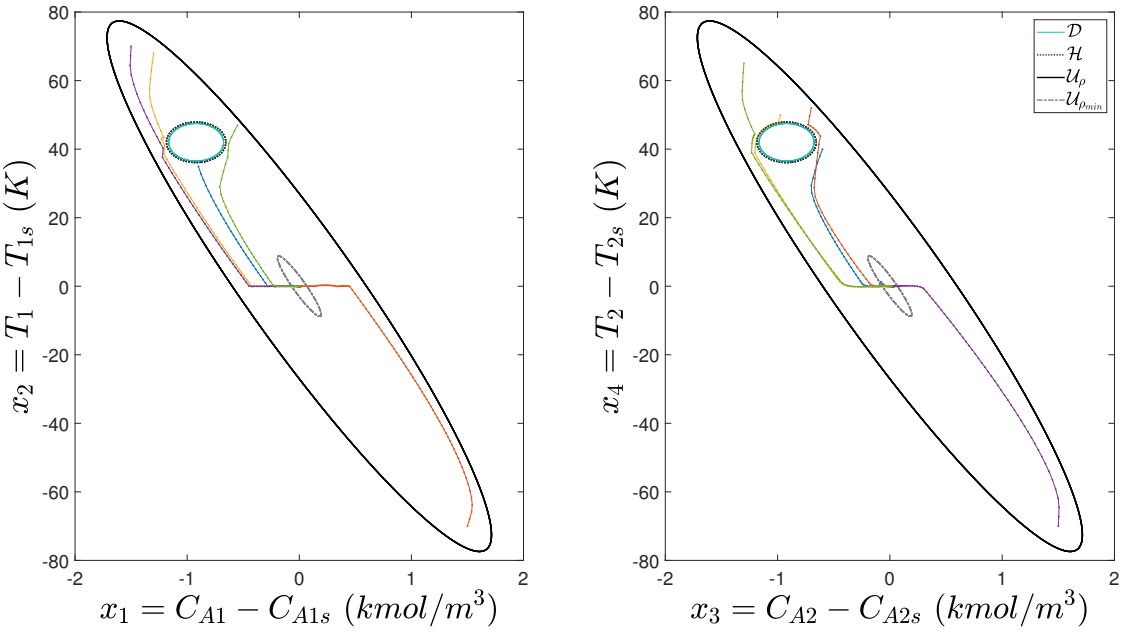


Figure 7.1: Closed-loop trajectories of CSTR-1 and CSTR-2 under the sequential CLBF-DMPC in the presence of a bounded unsafe set.

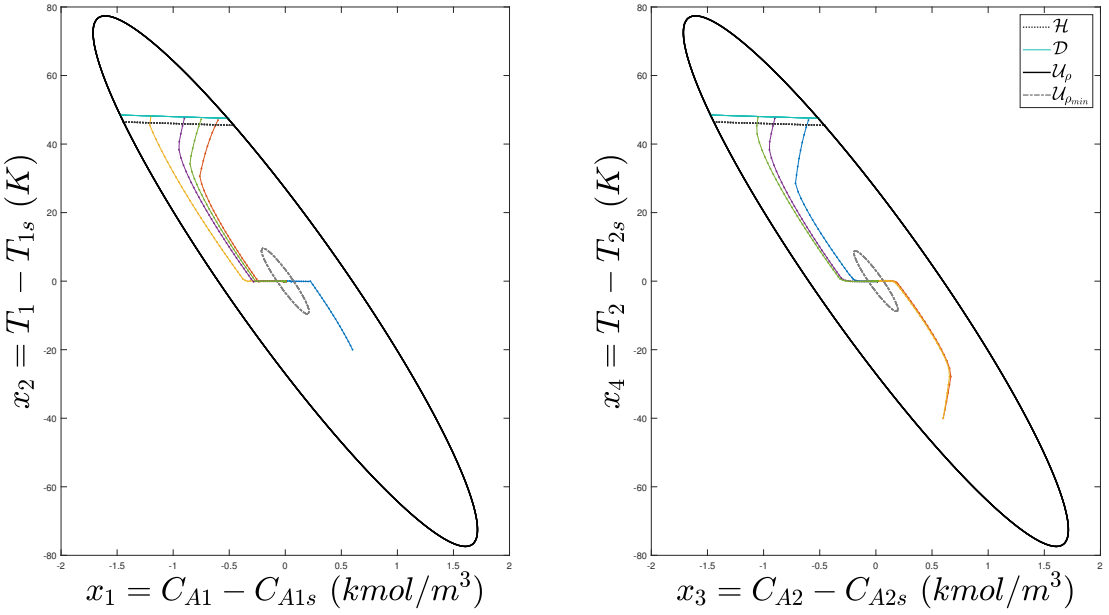


Figure 7.2: Closed-loop trajectories of CSTR-1 and CSTR-2 under the sequential CLBF-DMPC in the presence of an unbounded unsafe set.

of CSTR-1; this region can be also found via state-space discretization and extensive numerical simulations, but is difficult to visualize since it involves a 4-D state vector. Thus, considering

bounded unsafe sets, Fig. 7.3 and Fig. 7.4 show some CSTR-2 points in state-space where $\dot{W}_2$ is rendered negative under the CLBF-based Sontag control law $\Phi_2(x)$ plotted with respect to x_1 of CSTR-1 and x_2 of CSTR-1 separately. The x_1 and x_2 points of CSTR-1 are generated by discretizing $\mathscr{U}_{\rho_1}$, and the x_3 and x_4 points of CSTR-2 are generated by discretizing only the region in between the unsafe set $\mathscr{D}_2$ and the compact set $\mathscr{H}_2$ which encloses $\mathscr{D}_2$. We only assess discretized points in this critical region of safety to see which points may contribute to jeopardized safety when the states of CSTR-2 are near the boundary of the unsafe set. We can see that there exists combinations of (x_1, x_2) values that result in $\dot{W}_2 \geq 0$ under the CLBF-based Sontag control law $\Phi_2(x(t_k))$. In these situations, the CLBF-DMPC can still optimize for solutions of $u_{d_2}(x(t_k))$ that will yield decreasing W_2 along the predicted trajectory; for example, the constraint of Eq. 7.21h can be activated and the set of discontinuous control actions $\bar{u}_2(x) \in U_2$ that exist to address the cases of saddle points can be used. In situations where $\dot{W}_2(x, \Phi_2(x)) = 0$, the existence of $\bar{u}_2(x) \in U_2$ ensure the feasibility of DMPC-2 in guaranteeing stability and safety. However, in situations where $\dot{W}_2(x, \Phi_2(x)) > 0$, DMPC-2 may run into points of in-feasibility during optimization and this is demonstrated in Fig. 7.6.

In this study, we only consider the set of initial conditions in the respective regions $\mathscr{U}_{\rho_1}$ and $\mathscr{U}_{\rho_2}$ for the closed-loop simulations of CSTR-1 and CSTR-2. In our simulations, $\mathscr{U}_{\rho_2}$ mirrors $\mathscr{U}_{\rho_1}$ for simplistic visualization and to provide a preliminary set of initial conditions for which we can consider to perform closed-loop control using the CLBF-DMPC. As such, we can demonstrate that there are certain values of states of CSTR-1 that may jeopardize closed-loop safety for the same valued CSTR-2 states under the explicit CLBF-based Sontag control law. Furthermore, even though the discontinuous control actions $\bar{u}_2(x) \in U_2$ ensure feasibility of the CLBF-DMPC and provide a set of solutions that decrease W_2 along the prediction trajectory in the neighborhood of saddle points where $\dot{W}_2 = 0$, there may be situations where $\dot{W}_2 > 0$ and DMPC-2 is unable to reach a feasible solution that decreases W_2 at that particular point in state-space. Fig. 7.5 demonstrates that starting from five different initial conditions of CSTR-1 within $\mathscr{U}_{\rho_1}$ and the same initial condition of CSTR-2 within $\mathscr{U}_{\rho_2}$, simultaneous stability and safety can be achieved for both CSTRs where the closed-loop states for the overall system do not enter the unsafe region and converges to the terminal sets. This demonstrates the efficacy of the CLBF-DMPC in handling the impact of the states of CSTR-1 on the closed-loop evolution of CSTR-2. We may also examine the efficacy of CLBF-DMPC when the state of CSTR-2 is on the verge of critical safety. Starting from the same initial condition of CSTR-2 $(x_3,\ x_4) = (-1.135 kmol/m^3,\ 45.2K)$ that is within the enclosing compact set $\mathscr{H}_2$ but outside the unsafe set $\mathscr{D}_2$, we can see in Fig. 7.6 that some initial

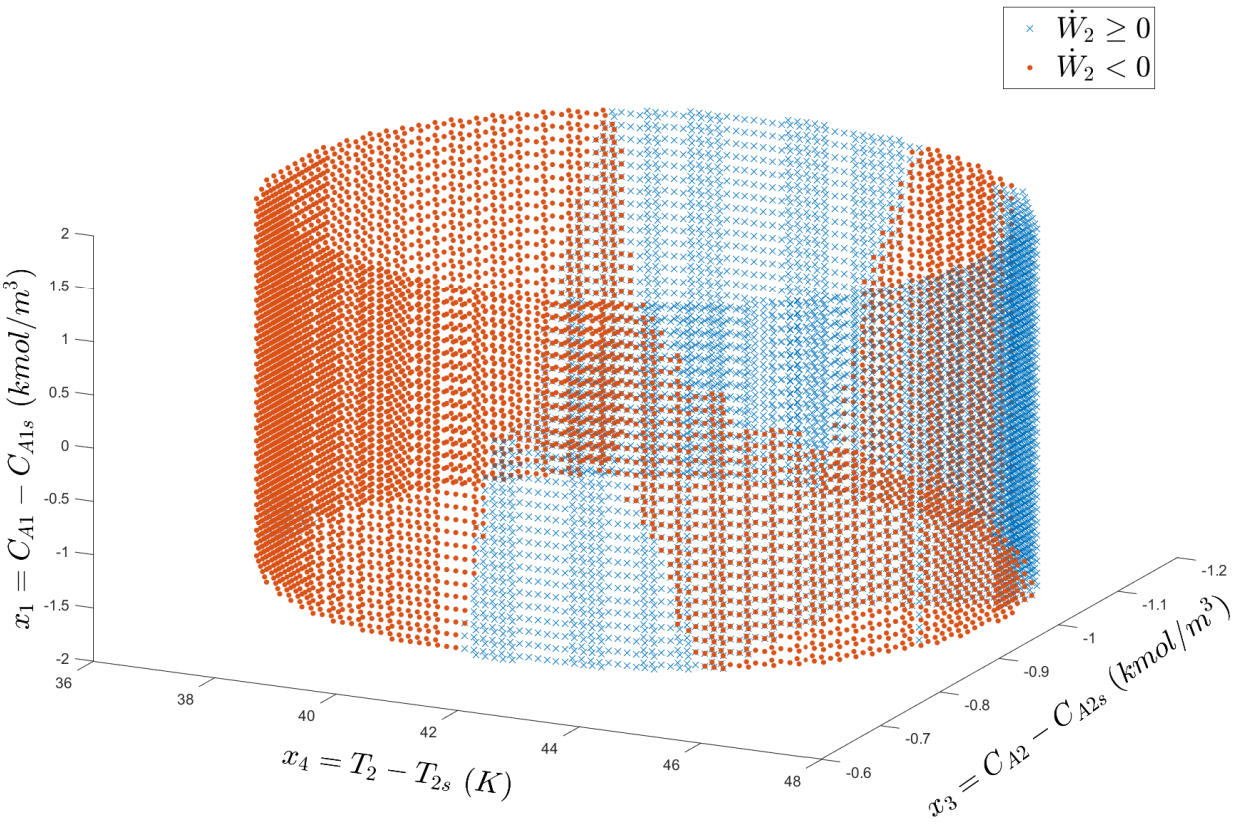


Figure 7.3: Discretized points (x_3, x_4) near CSTR-2's unsafe region $\mathscr{D}_2$ in state-space showing the negativity and non-negativity of $\dot{W}_2$ under the CLBF-based Sontag control law with respect to different values of x_1 discretized from CSTR-1's safe operating region $\mathscr{U}_{\rho_1}$.

conditions of CSTR-1 (orange, $(x_1,\ x_2) = (-1.08kmol/m^3,\ 64K)$) may result in safe closed-loop operation where the closed-loop state successfully avoids the unsafe sets $\mathscr{D}_1$ and $\mathscr{D}_2$, but some (blue, $(x_1,\ x_2) = (-0.6kmol/m^3,\ 36K)$) may result in the closed-loop state of CSTR-2 entering the unsafe set $\mathscr{D}_2$. Note that both sets of initial conditions of CSTR-1 shown in Fig. 7.6 have been evaluated to have $\dot{W}_2(x, \Phi_2(x)) > 0$. However, it is shown that starting from $(x_1,\ x_2,\ x_3,\ x_4) = (-1.08kmol/m^3,\ 64K,\ -1.135kmol/m^3,\ 45.2K)$ (orange), the CLBF-DMPC is able to provide feasible solutions that yield $\dot{W}_2(x(t_k), u^*_{d_2}(t_k)) < 0$ and drive the closed-loop states away and around the unsafe set $\mathscr{D}_2$. On the other hand, starting from $(x_1,\ x_2,\ x_3,\ x_4) = (-0.6kmol/m^3,\ 36K,\ -1.135kmol/m^3,\ 45.2K)$ (blue), the CLBF-DMPC fails to provide a set of feasible solutions with $\dot{W}_2(x(t_k), u^*_{d_2}(t_k)) < 0$, therefore resulting in the closed-loop state of CSTR-2 entering the unsafe set $\mathscr{D}_2$ within the first sampling period.

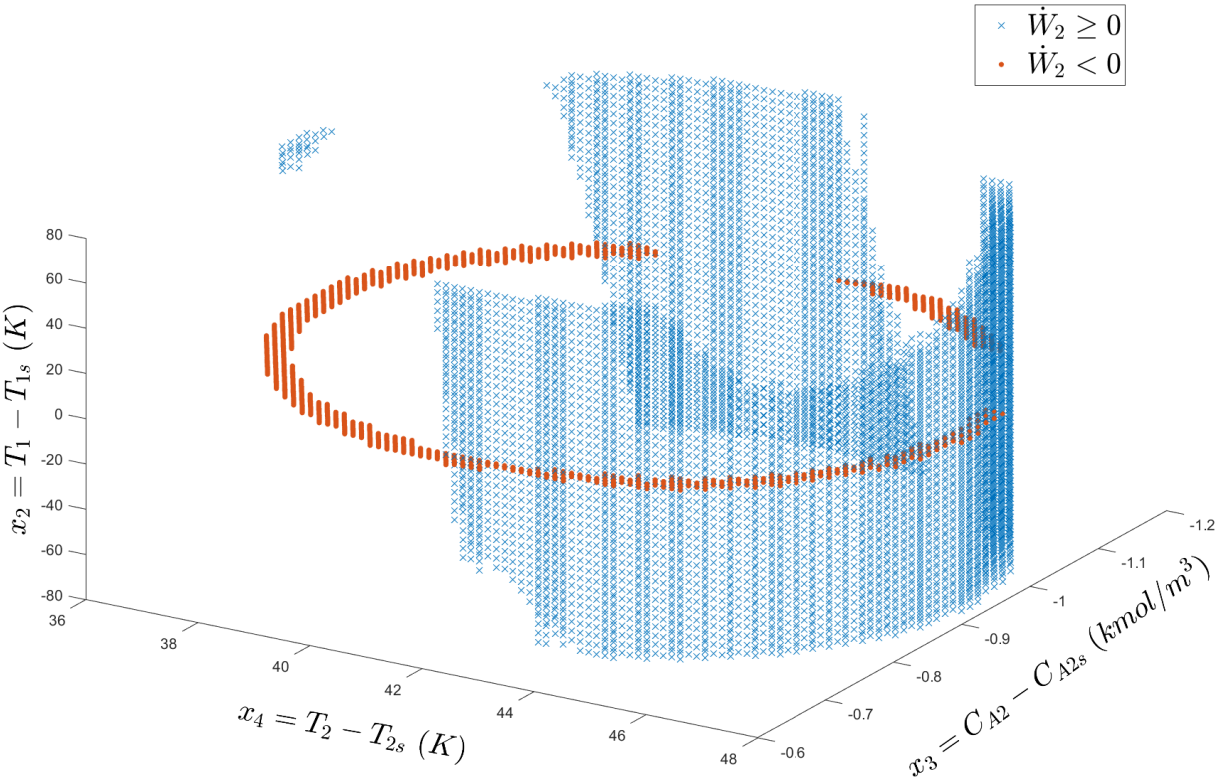


Figure 7.4: Discretized points (x_3, x_4) near CSTR-2's unsafe region $\mathscr{D}_2$ in state-space showing the negativity and non-negativity of $\dot{W}_2$ under the CLBF-based Sontag control law with respect to different values of x_2 discretized from CSTR-1's safe operating region $\mathscr{U}_{\rho_1}$.

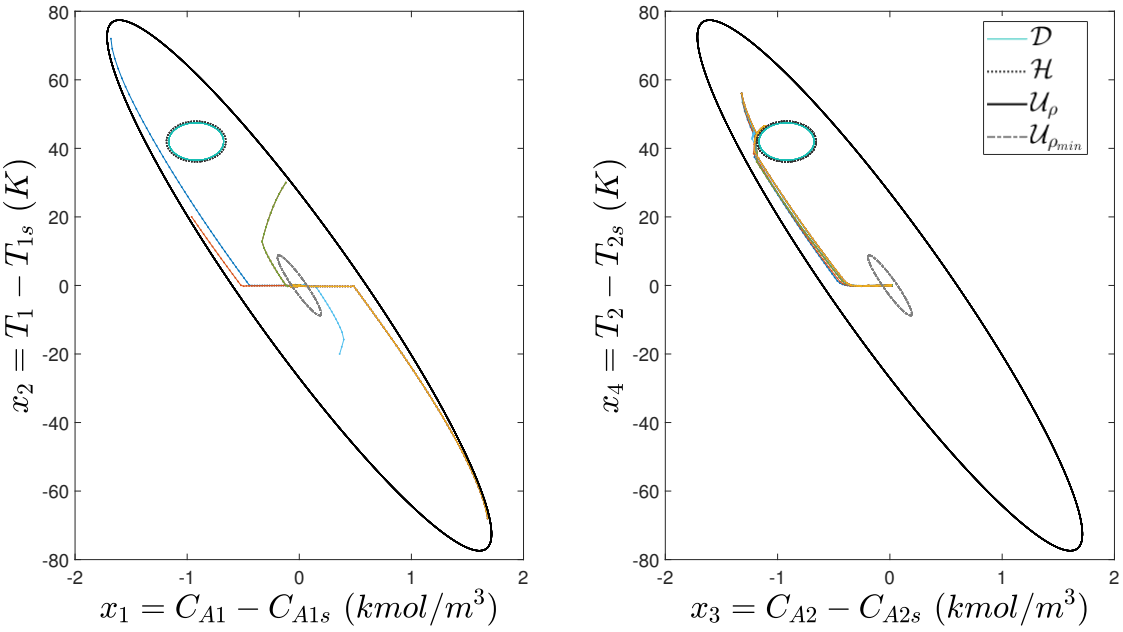


Figure 7.5: Closed-loop trajectories starting from different initial conditions of CSTR-1 and the same initial condition of CSTR-2 under the sequential CLBF-DMPC in the presence of a bounded unsafe set showing safe and stable performance.

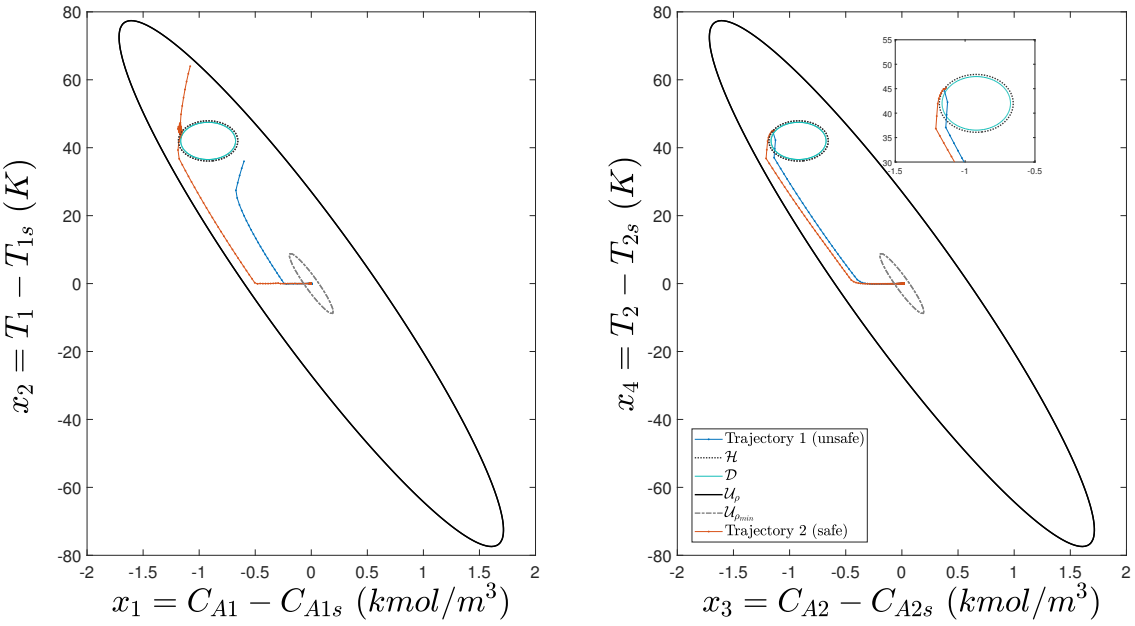


Figure 7.6: Closed-loop trajectories starting from two different initial conditions of CSTR-1 and the same initial condition of CSTR-2 under the sequential CLBF-DMPC in the presence of a bounded unsafe set showing one safe (orange) and one unsafe (blue) trajectory.

Chapter 8

Statistical Machine-Learning-based Predictive Control Using Barrier Functions for Process Operational Safety

In this work, we present statistical model predictive control with Control Lyapunov-Barrier Functions (CLBF) built using machine learning approaches, and analyze closed-loop stability and safety properties in probability using statistical machine learning theory. A feedforward neural network (FNN) is used to construct the Control Barrier Function, and a generalization error bound can be obtained for this FNN via the Rademacher complexity method. The FNN Control Barrier Function is incorporated in a CLBF-based model predictive controller (MPC), which is used to control a nonlinear process subject to input constraints. The stability and safety properties of the closed-loop system under the sample-and-hold implementation of FNN-CLBF-MPC are evaluated in a statistical sense. We use a chemical process example to demonstrate the relation between various factors of building an FNN model and the generalization error, as well as the probabilities of closed-loop safety and stability for both bounded and unbounded unsafe sets.

We provide statistical analysis on the CBF construction method proposed in our previous work in [25], and model the CBF using a feed-forward neural network, which will be used to design a CLBF-based model predictive control system. We first develop the generalization error bound on the FNN-CBF, and derive probabilistic safety and stability guarantees for the control law designed using a CLBF with FNN-CBF under sufficient conditions. The sampling, modeling, and verification procedures of the FNN are discussed. Then, we extend the probabilistic stability and safety properties to the FNN-CLBF-MPC, and demonstrate that with high probability, the

FNN-CLBF-MPC is able to maintain the closed-loop state of a nonlinear process within a safe set and ultimately keep it bounded within a terminal set around the origin.

The rest of the chapter is organized as follows. Preliminaries on the nonlinear system and definitions of Lyapunov Function and Barrier Function are given in Section 2. The construction of barrier functions using neural networks, including assumptions, design, data generation and model verification, are presented in Section 3. Section 4 develops the generalization error bounds on the FNN-CBF and explain their implications. In Section 5, the design of the FNN-CLBF control law and the FNN-CLBF-based MPC are provided, and the probabilistic stability and safety properties of the control system are provided. Lastly, the proposed control method and the associated generalization error and closed-loop performance are shown via a nonlinear chemical process example in Section 6.

8.1 Preliminaries

8.1.1 Notation

The Euclidean norm is denoted by the operator $|\cdot|$. The notation $\|W\|_{1,\infty} = \max_j(\sum_i |W_{i,j}|)$ denotes the infinity norm of the 1-norms of the columns of matrix W. We use "\" to represent set subtraction, i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. $\mathbf{x}^T$ denotes the transpose of matrix $\mathbf{x}$. $L_f V(\mathbf{x}) := \frac{\partial V(\mathbf{x})}{\partial x} f(\mathbf{x})$ represents the Lie derivative of V with respect to f. A function $f($ is class C^1 of the first derivative of f exists and is continuous. A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is said to be L-Lipschitz continuous, if there exists $L \geq 0$ such that for all $a, b \in \mathbf{R}^n$, $|f(a) - f(b)| \leq L|a - b|$. A continuous function $r : [0, a) \to [0, \infty)$ belongs to a class $\mathscr{K}$ function if $r(0) = 0$, and is strictly increasing. Lastly, $\mathbb{P}(A)$ represents the probability of the occurrence of an event A, and $\mathbb{E}[X]$ denotes the expected value of a random variable X.

8.1.2 Class of Systems

In this study, we consider a general class of continuous-time nonlinear systems, which can be represented by the following state-space model:

$$\dot{x} = F(x, u) := f(x) + g(x)u,\ x(t_0) = x_0 \tag{8.1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^k$ denotes the manipulated input vector bounded by $u \in U$, where $U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^k$. It is assumed that the vector and matrix functions $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth with $f(0) = 0$, and thus the origin is a steady-state of the nonlinear system. Lastly, the initial time is assumed to be at 0, i.e., $t_0 = 0$.

8.1.3 Stabilizability via Lyapunov-based Control

For the nonlinear system of Eq. 8.1, it is assumed that a stabilizing feedback control law $u = \Phi(x) \in U$ exists such that there exists a positive definite and proper Control Lyapunov Function (CLF), denoted as $V(x)$, that satisfies the following inequalities as well as the small control property:

$$c_1 |x|^2 \leq V(x) \leq c_2 |x|^2 \tag{8.2a}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq r_V(|x|) \tag{8.2b}$$

$$L_f V(x) < 0, \forall x \in \{z \in \mathbf{R}^n \backslash \{0\} \mid L_g V(z) = 0\} \tag{8.2c}$$

where r_V is a function that belongs to class $\mathscr{K}$, and c_1, c_2 are positive constants. $V(x)$ also meets the small control property, which states that, for every $\varepsilon > 0$, $\exists\ \delta > 0$, s.t. $\forall\ x \in \mathscr{B}_\delta(0)$, there exists an input u satisfying $|u| < \varepsilon$ and $L_f V(x) + L_g V(x) \cdot u < 0$ [98]. The existence of such CLF implies that the origin of the nonlinear system of Eq. 8.1 is rendered asymptotically stable under $u = \Phi(x) \in U$ for all x in a neighborhood around the origin. This region where the time derivative of $V(x)$ can be rendered negative under $u = \Phi(x) \in U$ is defined as $\phi_u = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V(x) + L_g V(x) \cdot u < 0, u = \Phi(x) \in U\} \cup \{0\}$. Furthermore, we define a level set of $V(x)$ within ϕ_u as $U_b := \{x \in \phi_u \mid V(x) \leq b,\ b > 0\}$, which is a forward invariant set in a sense that for any initial condition $x_0 \in U_b$, the closed-loop trajectory $x(t)$, $t \geq 0$ of the nonlinear system of Eq. 8.1 remains in U_b under $u = \Phi(x) \in U$.

8.1.4 Control Barrier Function

Consider that an open set $\mathscr{D}$ exists in state space, forming an unsafe region that should be avoided at all times for reasons such as violation of safety protocols. In contrast, a set of safe states can also be characterized as $\mathscr{X}_0 := \{x \in \mathbf{R}^n \backslash \mathscr{D}\}$ where $\{0\} \in \mathscr{X}_0$ and $\mathscr{X}_0 \cap \mathscr{D} = \emptyset$. The safe set $\mathscr{X}_0$ represents the set of initial conditions that will be considered. In this work, we consider process operational safety as follows:

Definition 8.1. *For any initial state $x(t_0) = x_0 \in \mathscr{X}_0$, if there exists a constrained control law $u = \Phi(x) \in U$ that renders the origin of the closed-loop system of Eq. 8.1 asymptotically stable, and the closed-loop state trajectories do not enter the unsafe set D at all times, i.e., $x(t) \in \mathscr{X}_0$, $x(t) \notin \mathscr{D}$, $\forall\, t \geq 0$, then the control law $u = \Phi(x) \in U$ maintains the closed-loop state within the safe region $\mathscr{X}_0$ for all times.*

Subsequently, we present the properties of a Control Barrier Function (CBF) in the following definition [116]:

Definition 8.2. *Consider $\mathscr{D}$ which is a set of unsafe state values in state space, a $\mathscr{C}^1$ function $B(x) : \mathbf{R}^n \to \mathbf{R}$ is a Control Barrier Function (CBF) if the following conditions are met:*

$$B(x) > 0, \quad \forall\, x \in \mathscr{D} \tag{8.3a}$$

$$L_f B(x) \leq 0,\ \forall\, x \in \{z \in \mathbf{R}^n \backslash \mathscr{D} \mid L_g B(z) = 0\} \tag{8.3b}$$

$$\mathscr{X}_B := \{x \in \mathbf{R}^n \mid B(x) \leq 0\} \neq \emptyset \tag{8.3c}$$

8.2 Barrier Function Construction using Feed-forward Neural Networks

8.2.1 Model Structure and Training

The control barrier function is developed from operating data in the state space that are labelled based on their safety status. This barrier function will then be synthesized using a feed-forward neural network (FNN), which typically consists of an input layer, some hidden layers, and an output layer. Each layer contains neurons undergoing nonlinear transformations, with activation functions of the weighted sum of neurons in the previous layer plus a bias term. In this study, the inputs to the FNN are the state vector $x \in \mathbf{R}^n$ of the nonlinear system of Eq. 8.1, and the output of the FNN predicts the barrier function value $\hat{B}(x) \in \mathbf{R}^n$. Training data points are collected from both the unsafe and the safe operating regions, where the target output values of $B(x)$ will satisfy the CBF conditions of Eq. 8.3a and Eq. 8.3c for the unsafe and the safe regions, respectively. More specifically, safe data points are labeled with a target output value of $B(x) = -1$, and unsafe data points are labeled with a target output value of $B(x) = +1$.

A general FNN model is considered, where m number of data samples are used to develop this model. The data samples are generated independently as per the data distribution over $X \times Y \in$

$\mathbf{R}^{d_x} \times \mathbf{R}^{d_y}$, where d_x and d_y denote the dimension of the FNN input and output vectors respectively; in this application, $d_x = n$, which is the dimension of the state vector of the nonlinear system of Eq. 8.1, and $d_y = 1$, which is the dimension of the barrier function output $B(x)$. The general structure of FNN model with inputs denoted as $\mathbf{x} \in \mathbf{R}^{d_x}$ and predicted output denoted as $\hat{\mathbf{y}} \in \mathbf{R}^{d_y}$ in terms of scalar or vector-valued functions and weight matrices for d total number of layers can be formulated as follows:

$$\hat{\mathbf{y}} = \sigma_d(W_d\sigma_{d-1}(W_{d-1}\sigma_{d-2}(...\sigma_1(W_1\mathbf{x})))) \tag{8.4}$$

where each W_l for $l = 1, ..., d$ layers represents the weight parameter matrix, and each σ_l represents the activation function in each layer. The number of layers d represents the depth of the network, and the width of the network h_{max} can be defined as the maximum number of neurons in a hidden layer (maximal column or row dimension of W_l), i.e., $h_{max} = \max_{l=1,...,d}\{h_l\}$, where h_l denotes the number of neurons in the l-th layer.

In this study, due to the unique dichotomous nature of $B(x)$, we choose a hyperbolic tangent sigmoid function $\sigma(z) = tanh(z) = \frac{2}{1+e^{-2z}} - 1$ as the activation function to polarize the output of the network and in turn, improve the prediction accuracy. This is because of the property of the $tanh(z)$ function approaching $+1$ as z approaches $+\infty$, and -1 as z approaches $-\infty$, thus polarizing the outputs of each layer and enforces the output of the FNN to approximate constant positive values (+1 for safe points), or constant negative values (-1 for unsafe points). To clarify notations used in this paper, when discussing the general properties of FNN, the input and output of the FNN model are denoted by the bold face $\mathbf{x} \in \mathbf{R}^{d_x}$ and $\mathbf{y} \in \mathbf{R}^{d_y}$ respectively. For this particular application, $\mathbf{x}$ is the state vector of Eq. 8.1 ($x \in \mathbf{R}^n$), and $\mathbf{y}$ is the barrier function value ($B(x) \in \mathbf{R}^1$).

Before proceeding with developing the generalization error bound, there are some standard assumptions presented as follows:

Assumption 8.1. *The FNN inputs are bounded, i.e., $|\mathbf{x}_i| \leq B_X$, for all $i = 1, ..., m$ samples.*

Assumption 8.2. *The maximal 1-norm (l_1/l_∞) of the rows of weight matrices in the output and in the hidden layers are bounded as follows:*

$$\|W\|_{1,\infty} \leq B_W \tag{8.5}$$

Assumption 8.3. *All the datasets (i.e., training and testing) are drawn from the same underlying distribution.*

Assumption 8.4. *σ_l (where l denotes any hidden layers) is a 1-Lipschitz continuous activation function, and satisfies $\sigma_l(0) = 0$.*

Remark 8.1. *Assumption 8.1 specifies the upper bound on the FNN inputs, which is consistent with the way we sample the FNN inputs (i.e., the state vector) as we only consider a bounded set around the steady-state of the nonlinear system of Eq. 8.1. Assumption 8.2 assumes the boundedness of the FNN weight matrices; this can be ensured during FNN training, as only a finite class of hypothesis functions are searched to find the optimal set of FNN parameters. Assumption 8.3 is required as the model trained from the training dataset will be evaluated on the testing dataset, and training and testing model accuracy metrics are compared against each other. The evaluation of the model on the testing data (including closed-loop simulations) as well as the comparison of accuracy metrics are only valid if the two datasets have the same underlying target distribution. Assumption 8.4 is an assumption on the activation functions of the FNN, which is satisfied by many common activation functions, and can be used to derive the upper bound for the Rademacher complexity of the FNN hypothesis class. An example of a 1-Lipschitz continuous activation function is $tanh(\cdot)$.*

We sample points from the operating region of the system (i.e., $x \in \mathscr{X} \subset \mathbf{R}^n$ where $\mathscr{X}$ is a compact set) to use as training and testing data for the FNN. Since the conditions of Eq. 8.3 imposed on the resulting $\hat{B}(x)$ must be satisfied in a continuous sense, the regions from which discrete data points are sampled from must be compact and connected. This is done by first characterizing a compact and connected set $\mathscr{H}$, which is a superset of the open set $\mathscr{D}$ (as indicated in Eq. 8.32 in Section 8.4), then designing a larger compact and connected set $\mathscr{H}'$, which is a superset of $\mathscr{H}$ and encloses $\mathscr{H}$ with sufficient margin. This region $\mathscr{H}'$ is used to generate unsafe data points from, such that the unsafe set the FNN model predicts will remain as a superset of $\mathscr{H}$, given bounded modeling and numerical error of the FNN model. This means that the FNN model may classify safe points as unsafe, but will not classify unsafe points as safe; the latter is not tolerated and should be avoided. Readers who are interested may refer to [25] for more details on how to characterize the unsafe region for data collection purposes when building a FNN-CLBF-MPC that uses both first-principles and RNN models. We collect samples from the safe region $\mathscr{X} \setminus \mathscr{H}'$ and the unsafe region $\mathscr{H}'$ by discretizing the regions by a grid size of $(\delta x)_{\mathscr{H}'}$ and $(\delta x)_{\mathscr{X} \setminus \mathscr{H}'}$ respectively. The datasets consisting of finite samples are denoted as $S_{\mathscr{J}}$ and $S_{\mathscr{H}'}$ for safe and unsafe regions, respectively. Together, $S_{\mathscr{J}}$ and $S_{\mathscr{H}'}$ form the overall sample set S_s.

The FNN parameters (weights and biases) are optimized by minimizing the loss function shown in Eq. 8.6 using the *Adam* solver as a part of the Tensorflow Keras software package. Specifically,

the loss function consists of two parts. The first part L_1 uses mean squared error to calculate the difference between the target $B(x)$ and the prediction $\hat{B}(x)$, and in minimizing this error, aims to satisfy the conditions of Eq. 8.3a and Eq. 8.3c. The second part L_2 penalizes sample points that do not comply with the conditions of Eq. 8.3b by using the $ReLu(\cdot)$ function and adding a small positive constant τ_I as seen in Eq. 8.6c.

$$L(\hat{B}, B) = \alpha L_1 + \beta L_2 \tag{8.6a}$$

$$L_1 = \frac{1}{m}\sum_{k=1}^{m}(\hat{B}(x_k) - B_k)^2 \tag{8.6b}$$

$$L_2 = \frac{1}{N_{I_f}}\sum_{j=1}^{N_{I_f}} ReLu(L_f\hat{B}(x_j) + \tau_I) \tag{8.6c}$$

where L_1 tracks the mean squared error (MSE) between the target B and the predicted barrier function $\hat{B}$ for all discretized data points $x_k, k = 1, ..., m$, in the entire operating region that we sample from, and L_2 is the loss function term that aims to satisfy $L_f\hat{B} \leq 0$ for all $x \in \{S_{\mathscr{I}} | L_g\hat{B}(x) = 0\}$, where N_{I_f} is the number of discretized data points that satisfies this condition in the safe region. Since $ReLu$ takes the maximum between its argument and 0, i.e., $ReLu(z) = \max\{0, z\}$, L_2 penalizes any samples that produce $L_f\hat{B}_j + \tau_I > 0$, therefore forcing $L_f\hat{B}_j \leq 0$ to hold for the applicable points in the safe region. $\alpha > 0$ and $\beta > 0$ are hyperparameters that adjust the weighting of L_1 and L_2 in the cost function. When L_2 has reached 0 during training, then the weights and biases have been optimized in a way that the predicted barrier function $\hat{B}(x)$ satisfies the condition Eq. 8.3b. To make sure that all conditions of Eq. 8.3 are satisfied at the end of training, L_1 and L_2 are evaluated and monitored separately during training, and both L_1 and L_2 are required to be below a respective threshold value such that the modeling error for $\hat{B}(x)$ is bounded and the negative semi-definiteness of $L_f\hat{B}(x)$ for all x in the safe region with $L_g\hat{B}(x) = 0$ can be shown.

8.2.2 Verification of FNN-based CBF

Upon arriving at an FNN-CBF from the discretized data samples, it is important to demonstrate that the conditions of Eq. 8.3 in the Definition of CBF are satisfied and that FNN-CBF can be used to design control laws for the continuous nonlinear system of Eq. 8.1.

8.2.2.1 Continuity and Differentiability

The CBF is continuously differentiable (i.e., a C^1 function) by Definition 8.2, mandating that $\hat{B}(x)$ and $\dot{\hat{B}}(x)$ must be proven to be continuous. As per the universal approximation theorem [99], with sufficient model complexity, FNNs are capable of modeling any continuous nonlinear functions on a compact set of the state space. In addition, $\hat{B}(x)$ is the output of an FNN that consists of a chain of nonlinear activation functions, i.e., $tanh(\cdot)$, which is a Lipschitz continuous and continuously differentiable function in the compact subset we sample from. Thus, $\hat{B}(x)$ is also Lipschitz continuous and continuously differentiable on the sampled compact subset. In terms of FNN notations, we have shown that the overall hypothesis function class $h(\mathbf{x})$ that maps the FNN inputs $\mathbf{x}$ to the FNN output $\mathbf{y}$ in the form of barrier function value is also a C^1 function. It is assumed that the barrier function satisfies the following inequality:

$$\left|\frac{\partial B}{\partial x}\right| \leq r_B(|x|) \tag{8.7}$$

where r_B is a class $\mathscr{K}$ function similar to r_V in Eq. 8.2b.

8.2.2.2 Verification

Training an FNN that minimizes the loss function of Eq. 8.6 aims to meet the conditions of Eq. 8.3 in Definition 8.2 for all discretized points sampled from the compact subsets that we consider, but does not guarantee that the conditions are met for all points in the respective compact subsets. Therefore, the conditions must be verified to hold over the compact subsets in a continuous sense. Similar to the approaches implemented in [13, 58, 88], we use a Lipschitz method to verify that the decrease condition holds for a candidate function on a finite sample of a bounded set. The following theorem presents the necessary criteria to use this verification technique:

Theorem 8.1. *Consider a compact set $S \subset \mathbf{R}^n$ and let S_s be a finite set sampled from S s.t. $\forall\, x \in S$, there exists at least a pair $(x_s, \delta_{x_s}) \in S_s \times \mathbf{R}_+$ such that $|x - x_s| \leq \delta x_s$. If $F(x_s) \leq -L_F \cdot \delta x_s$ (or respectively $F(x_s) < -L_F \cdot \delta x_s$) holds for all $x_s \in S_s$, where the Lipschitz constant for the function F is denoted by $L_F > 0$, then $F(x) \leq 0$ (respectively $F(x) < 0$) holds for all $x \in S$. [13]*

Therefore, by checking the tightened inequality $L_f\hat{B}(x) \leq -L' \cdot \delta x_{\mathscr{X}\setminus\mathscr{H}'},\ \forall\, x \in S_{\mathscr{I}}$, it will be verified that $L_f\hat{B}(x) \leq 0,\ \forall\, x \in \mathscr{X}\setminus\mathscr{H}'$, where $L' > 0$ is the Lipschitz constant for $L_f\hat{B}(x)$, the finite set $S_{\mathscr{I}}$ is sampled from the compact set $\mathscr{X}\setminus\mathscr{H}'$, and $\delta x_{\mathscr{X}\setminus\mathscr{H}'} > 0$ is the discretization grid size (distance between two discretized x points) of the safe set $\mathscr{X}\setminus\mathscr{H}'$. On a similar note,

$\hat{B}(x) \leq 0,\ \forall\, x \in \mathscr{X}\backslash\mathscr{H}'$ can be shown to hold by verifying that $\hat{B}(x) \leq -L'' \cdot \delta x_{\mathscr{X}\backslash\mathscr{H}'}\ \forall\, x \in S_{\mathscr{J}}$, where the Lipschitz constant for $\hat{B}$ is denoted by L''. Lastly, we show that Eq. 8.3a is satisfied by checking $-\hat{B}(x) < -L'' \cdot \delta x_{\mathscr{H}'},\ \forall\, x \in S_{\mathscr{H}'}$, which is sufficient to verify that $-\hat{B}(x) < 0\ \forall\, x \in \mathscr{H}'$, thus equivalent to $\hat{B}(x) > 0\ \forall\, x \in \mathscr{H}'$. These conditions will be checked for all sample points in the respective discretized sets after an FNN model is obtained. More details on the sampling, design, training, and verification of the FNN-CBF can be found in our previous work in [25].

8.3 FNN Generalization Error

When we train an FNN model, the model is obtained by minimizing the loss function calculated based on training data samples only. Therefore, there is no information given on the error or performance of the model on new testing data. The generalization error measures the model's ability of making an accurate prediction for new data from the same underlying distribution that has not been seen or studied by the neural network. Using statistical theory commonly used in machine learning, we present an upper bound for the generalization error of the FNN model in predicting the value of the barrier function output.

We first introduce some important preliminary concepts that will be referenced in the development of FNN generalization error bound. Without loss of generality, we let $\mathscr{K}$ be the hypothesis class of FNN functions $h(\cdot)$ that map a d_x-dimensional input $\mathbf{x} \in \mathbf{R}^{d_x}$ to a d_y-dimensional output $\hat{\mathbf{y}} \in \mathbf{R}^{d_y}$. We use $\hat{\mathbf{y}} = h(\mathbf{x})$ to denote the predicted output of the FNN model and $L(\hat{\mathbf{y}}, \mathbf{y})$ to represent the loss function. Here, the loss function can be of many forms; for example, in our case of constructing a barrier function FNN, the loss function is the sum of two loss functions as shown in Eq. 8.6, where one loss function (L_1) assesses the mean squared error between the predicted and the true barrier function output values, and the other loss function (L_2) ensures that the Lie derivative properties of the resulting FNN barrier function are met. Nevertheless, in supervised learning where the true output values are known and used during training, the loss function will involve calculating the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$. The following error definitions are presented for FNN model training.

Definition 8.3. *[81] Given a function h that predicts* $\boldsymbol{y}$ *(output) using* $\boldsymbol{x}$ *(input), the generalization error or expected loss / error over an underlying data distribution is* D_d *is*

$$L_D(h) \triangleq \mathbb{E}[L(h(\boldsymbol{x}), \boldsymbol{y})] = \int_{X \times Y} L(h(\boldsymbol{x}), \boldsymbol{y}) \rho(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \tag{8.8}$$

where $\rho(\boldsymbol{x},\boldsymbol{y})$ is the joint probability distribution for $\boldsymbol{x}$ and $\boldsymbol{y}$, X and Y respectively denote the vector space for all possible inputs and outputs.

In most cases, the joint probability distribution ρ is not known. Therefore, we approximate the expected error by using the empirical error presented as follows:

Definition 8.4. *[81] Consider a dataset $S_s = (s_1, ..., s_m)$, $s_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$, with m number of data samples collected from the underlying data distribution D_d, the **empirical risk** or **error** is*

$$\hat{\mathbb{E}}_{S_s}[L(h(\boldsymbol{x}),\boldsymbol{y})] = \frac{1}{m}\sum_{i=1}^{m} L(h(\boldsymbol{x}_i),\boldsymbol{y}_i) \tag{8.9}$$

In addition, we also need to demonstrate the loss function $L(\hat{\mathbf{y}},\mathbf{y})$ is locally Lipschitz continuous. In this particular study, the true FNN output is the true barrier function value $B \in \mathbf{R}^1$ that takes the values of either -1 or $+1$, thus $|\mathbf{y}| \leq 1$. Since the FNN uses hyperbolic tangent sigmoid $\sigma(z) = tanh(z) = \frac{2}{1+e^{-2z}} - 1$ as the activation function, the predicted FNN output $\hat{B}$ is also bounded by $|\hat{\mathbf{y}}| \leq 1$. Furthermore, the training of FNN is designed such that it will only stop after L_2 in Eq. 8.6 reaches below a threshold (i.e., $L_f\hat{B}(x) \leq 0 \; \forall \; x \in \{S_{\mathscr{I}} | L_g\hat{B}(x) = 0\}$ is satisfied only when $L_2 \leq \tau_I$, where $\tau_I > 0$ is a small positive constant). Therefore, L_2 is also upper bounded. With these considerations, both L_1 and L_2 loss functions are locally Lipschitz continuous, and the overall loss function L is also locally Lipschitz continuous with the following inequality satisfied for any two predictions:

$$|L(\mathbf{y},\hat{\mathbf{y}}_2) - L(\mathbf{y},\hat{\mathbf{y}}_1)| \leq L_r\,|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1| \tag{8.10}$$

where L_r denotes the local Lipschitz constant for the loss function L.

8.3.1 Rademacher Complexity

We use empirical Rademacher complexity to bound the generalization error as it is commonly used in machine learning theory to quantify the richness of a class of functions. The Rademacher complexity is defined as follows:

Definition 8.5. *[81] Given a dataset of m samples $S_s = \{s_1, ..., s_m\}$, and a hypothesis class $\mathscr{F}$ of scalar-valued functions, the empirical Rademacher complexity of $\mathscr{F}$ is defined as:*

$$\mathscr{R}_{S_s}(\mathscr{F}) = \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\sup_{f\in\mathscr{F}} \frac{1}{m}\sum_{i=1}^{m} \varepsilon_i f(s_i)\right] \tag{8.11}$$

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

For the hypothesis class $\mathscr{H}_h$ of vector-valued functions $h \in \mathbf{R}^{d_y}$, it also satisfies the inequality shown in the following lemma:

Lemma 8.1 (c.f. Corollary 4 in [77])**.** *Given a hypothesis class* $\mathscr{H}_h$ *of vector-valued functions* $h \in \mathbf{R}^{d_y}$*, and a dataset of m samples* $S_s = \{s_1, ..., s_m\}$*. Consider the loss function* $L(\cdot)$ *which is a* L_r*-Lipschitz function mapping* $h \in \mathbf{R}^{d_y}$ *to* $\mathbf{R}$*, then we have*

$$\mathbb{E}_{\boldsymbol{\varepsilon}} \left[\sup_{h \in \mathscr{H}_h} \sum_{i=1}^{m} \varepsilon_i L(h(\boldsymbol{x}_i), \boldsymbol{y}_i) \right] \leq \sqrt{2} L_r \mathbb{E}_{\boldsymbol{\varepsilon}} \left[\sup_{h \in \mathscr{H}_h} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \varepsilon_{ik} h_k(\boldsymbol{x}_i) \right] \tag{8.12}$$

where ε_{ik} *is a* $m \times d_y$ *matrix consisting of independent Rademacher variables, and* $h_k(\cdot)$ *denotes the k-th component of the vector-valued function* $h(\cdot)$*. For simplicity, the subscript* $\boldsymbol{\varepsilon}$ *on the expectation will be omitted for the remainder of the manuscript.*

The following bound ([77]) can be derived to simplify the bound in terms of vector-valued functions to one in terms of scalar-value functions:

$$\mathbb{E} \left[\sup_{h \in \mathscr{H}_h} \sum_{i=1}^{m} \sum_{k=1}^{d_y} \varepsilon_{ik} h_k(\mathbf{x}_i) \right] \leq \sum_{k=1}^{d_y} \mathbb{E} \left[\sup_{h \in \mathscr{H}_{h,k}} \sum_{i=1}^{m} \varepsilon_i h(\mathbf{x}_i) \right] \tag{8.13}$$

where $\mathscr{H}_{h,k}$, $k = 1, ..., d_y$ represent scalar-valued function classes for the components of the vector-valued function class $\mathscr{H}_h$ for a network of d layers. We derive the bound for empirical Rademacher complexity in terms of scalar-valued function class first, then use Eq. 8.13 to develop the bound for vector-valued functions.

8.3.2 Generalization Error Bound of FNN

Consider the class of loss functions associated with the function class $\mathscr{H}_h$:

$$\mathscr{G} = \{g : (\mathbf{x}, \mathbf{y}) \to L(h(\mathbf{x}), \mathbf{y}), h \in \mathscr{H}_h\} \tag{8.14}$$

where $\mathbf{y}$ is the true FNN output vector, $\mathbf{x}$ is the FNN input vector, and $h(\mathbf{x})$ represents the predicted FNN output vector. We have the following lemma to upper bound the generalization error using the Rademacher complexity of the family of loss functions $\mathscr{R}_{S_s}(\mathscr{G})$.

Lemma 8.2 (c.f. Theorem 3.3 in [81]). *Given a data set of m number of i.i.d samples, the following inequality holds for all $g \in \mathscr{G}$ over the sample space $S_s = (s_i)$, $s_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$ with probability of at least $1-\delta$:*

$$\mathbb{E}[g(\boldsymbol{x},\boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g(\boldsymbol{x}_i,\boldsymbol{y}_i) + 2\mathscr{R}_{S_s}(\mathscr{G}) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{8.15}$$

Interested readers may refer to [126] and [81] for the full proof of this lemma. The RHS of this inequality includes three terms, the sum of which specifies the upper bound for the FNN generalization error. These three terms represent the empirical loss based on the sample dataset S_s, the Rademacher complexity, and an error term that depends on the sample size and confidence δ. We further bound the Rademacher complexity such that the upper bound of the generalization error can be quantified by known specific values such as the sample size m, confidence δ, neural network depth d, input dimension d_x, and upper bounds on the input vector B_X and on the weight matrices B_W.

We first consider the hypothesis class $\mathscr{H}_{h,k}$ of scalar-valued functions, where k represents components of the vector-valued function class $\mathscr{H}_h$. For the scalar-valued function class $\mathscr{H}_{h,k}$, the following lemma is presented to upper-bound the scaled empirical Rademacher complexity. We will later use this lemma to derive the upper bound for the empirical Rademacher complexity for the vector-valued hypothesis function class $\mathscr{H}_h$.

Lemma 8.3 (c.f. Lemma 4 in [126]). *With $\lambda > 0$, the scaled empirical Rademacher complexity $m\mathscr{R}_{S_s}(\mathscr{H}_{h,k}) = \mathbb{E}[\sup_{h\in\mathscr{H}_{h,k}} \sum_{i=1}^{m} \varepsilon_i h(\boldsymbol{x}_i)]$ satisfies the following inequality:*

$$\begin{aligned} m\mathscr{R}_{S_s}(\mathscr{H}_{h,k}) &= \mathbb{E}[\sup_{h\in\mathscr{H}_{h,k}} \sum_{i=1}^{m} \varepsilon_i h(\boldsymbol{x}_i)] \\ &= \frac{1}{\lambda}\log\exp\left(\lambda\,\mathbb{E}\left[\sup_{h\in\mathscr{H}_{h,k}} \sum_{i=1}^{m} \varepsilon_i h(\boldsymbol{x}_i)\right]\right) \\ &\leq \frac{1}{\lambda}\log\left(\mathbb{E}\left[\sup_{h\in\mathscr{H}_{h,k}} \exp(\lambda \sum_{i=1}^{m} \varepsilon_i h(\boldsymbol{x}_i))\right]\right) \end{aligned} \tag{8.16}$$

We further specify the upper bound of the Rademacher complexity by breaking down the function $h(\mathbf{x}_i)$; this is done through a "peeling" approach to "peel" off the weights and activation functions of the FNN model layer by layer. Here, due to the unique application of the FNN model we construct, all the activation functions are $tanh(\cdot)$ in order to polarize the results to $+1$ and -1 values. We present the following lemma, which is modified from Lemma 2 in [43], to

demonstrate this peeling step inside a convex, monotonically increasing function (such as $exp(\cdot)$) for a 1-Lipschitz activation function $\sigma(\cdot)$ that satisfies $\sigma(0) = 0$ (such as $tanh(\cdot)$).

Lemma 8.4 (c.f. Lemma 2 in [43])**.** *Given any vector-valued function class $\mathscr{N}$ with a 1-Lipschitz continuous activation function $\sigma(\cdot)$ that satisfies $\sigma(0) = 0$ applied element-wise, and a convex and monotonically increasing function $p : \mathbf{R} \to \mathbf{R}_+$, the following inequality holds:*

$$\mathbb{E}\left[\sup_{||W||_{1,\infty} \leq B_W, \nu \in \mathscr{N}} p\left(||\sum_{i=1}^{m} \varepsilon_i \sigma(W\nu(\boldsymbol{x}_i))||_\infty\right)\right] \leq 2\mathbb{E}\left[\sup_{\nu \in \mathscr{N}} p\left(B_W||\sum_{i=1}^{m} \varepsilon_i \nu(\boldsymbol{x}_i)||_\infty\right)\right] \tag{8.17}$$

Lemma 8.4 holds for the vector-valued function class $\nu \in \mathscr{N}$ (or equivalently $h \in \mathscr{H}_h$), and therefore also holds for the scalar-valued function class $\nu \in \mathscr{N}_k$ (or equivalently $h \in \mathscr{H}_{h,k}$), where k represents the k-th component of the vector-valued function class. Following Lemma 8.4, we now reference Theorem 2 in [43] to derive a bound on the Rademacher complexity for the scalar-valued FNN function class $\mathscr{H}_{h,k}$, as presented in Lemma 8.5. The full proof of Lemma 8.5 can be found in [43]. First, Eq. 8.16 is used as a starting point to provide an inequality involving the scaled Rademacher complexity for the scalar-valued function class $\mathscr{H}_{h,k}$ and the scalar-valued hypothesis function $h(\mathbf{x}) \in \mathscr{H}_{h,k}$, which provides the predicted output in the output layer. Since the function $exp(\cdot)$ in Eq. 8.16 qualifies as a convex, monotonically increasing function, we can apply Lemma 8.4 repetitively to Eq. 8.16 by "peeling" off the neural network layer by layer, starting from $h(\mathbf{x})$ in the output layer. The function $p(\cdot)$ in Eq. 8.17 refers to $exp(\cdot)$, and the scalar-valued functions $\nu \in \mathscr{N}_k$ refer to subnetworks of the FNN from the input layer up to the layer being "peeled". The resulting upper bound on the Rademacher complexity for the scalar-valued function class $\mathscr{H}_{h,k}$ is presented in Lemma 8.5 and can be represented in terms of FNN input bound, weight matrix bounds, FNN depth, sample size, and FNN input dimension.

Lemma 8.5 (c.f. Theorem 2 in [43])**.** *Given neural networks with depth d and a class of scalar-valued functions $\mathscr{H}_{h,k}$ where $||W_l||_{1,\infty} \leq B_W$ for all $l = 1, ..., d$, and Assumptions 8.1 - 8.4 satisfied, the following inequality holds:*

$$\mathscr{R}_{S_s}(\mathscr{H}_{h,k}) \leq \frac{2B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}} \tag{8.18}$$

Interested readers may refer to Section 7 of [43] for the proof of this theorem.

The above lemma presents the Rademacher complexity upper bound for the scalar-valued functions $\mathscr{H}_{h,k}$, $k = 1, ..., d_y$, for the k-th component of the vector-valued function class $\mathscr{H}_h$.

Now we will derive the generalization error bound for the loss function class associated with the vector-valued hypothesis FNN function class $\mathscr{H}_h$. We use Eqs. 8.12-8.13 to derive the following theorem:

Theorem 8.2 (c.f. Theorem 1 in [126]). *Consider the dataset S_s consisting of m i.i.d. data samples and the class of loss functions associated with the vector-valued FNN hypothesis class $\mathscr{H}_h$ satisfying Assumptions 8.1 - 8.4. With probability of at least $1-\delta$, we have the following inequality:*

$$\mathbb{E}[g(\boldsymbol{x},\boldsymbol{y})] \leq \mathscr{O}\left(L_r d_y \frac{B_X (B_W)^d \sqrt{d+1+log(d_x)}}{\sqrt{m}}\right) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \frac{1}{m}\sum_{i=1}^{m} g(\boldsymbol{x}_i,\boldsymbol{y}_i) \tag{8.19}$$

where B_X is the upper bound on FNN inputs defined in Eq. 8.1, B_W is the upper bound on FNN weight matrices as stated in Eq. 8.2, L_r is the local Lipschitz constant for the loss function $L(\cdot)$ as defined in Eq. 8.10, d_x is the FNN input dimension, d_y is the FNN output dimension.

Proof. Using Eqs. 8.12 - 8.13, we can derive the following upper bound for the loss function $L(h(\mathbf{x}_i),\mathbf{y}_i)$ with $h(\mathbf{x}_i)$ being vector-valued functions:

$$\begin{aligned}\mathscr{R}_{S_s}(\mathscr{G}) = \mathbb{E}\left[\sup_{h\in\mathscr{H}_h}\frac{1}{m}\sum_{i=1}^{m}\varepsilon_i L(h(\mathbf{x}_i),\mathbf{y}_i)\right] &\leq \sqrt{2}L_r\mathbb{E}\left[\sup_{h\in\mathscr{H}_h}\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{d_y}\varepsilon_{ik}h_k(\mathbf{x}_i)\right] \\ &\leq \sqrt{2}L_r\frac{1}{m}\sum_{k=1}^{d_y}\mathbb{E}\left[\sup_{h\in\mathscr{H}_{h,k}}\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right]\end{aligned} \tag{8.20}$$

Using the definition of Rademacher complexity for the scalar-valued function class $\mathscr{H}_{h,k}$, we have the following:

$$\begin{aligned}\sqrt{2}L_r\frac{1}{m}\sum_{k=1}^{d_y}\mathbb{E}\left[\sup_{h\in\mathscr{H}_{h,k}}\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right] &= \sqrt{2}L_r\frac{1}{m}\sum_{k=1}^{d_y} m\mathscr{R}_{S_s}(\mathscr{H}_{h,k}) \\ &= \sqrt{2}L_r\sum_{k=1}^{d_y}\mathscr{R}_{S_s}(\mathscr{H}_{h,k})\end{aligned} \tag{8.21}$$

Therefore, using Eq. 8.18, we can derive the bound on the Rademacher complexity of the loss

function as follows:

$$\begin{aligned}\mathscr{R}_{S_s}(\mathscr{G}) &\leq \sqrt{2}L_r \sum_{k=1}^{d_y} \mathscr{R}_{S_s}(\mathscr{H}_{h,k}) \\ &\leq \sqrt{2}L_r \sum_{k=1}^{d_y} \frac{2B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}} \\ &\leq 2\sqrt{2}L_r d_y \frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\end{aligned} \tag{8.22}$$

Lastly, we can substitute Eq. 8.22 into Eq. 8.15, and obtain the generalization error bound as seen in Eq. 8.19. □

8.3.3 Implications of Generalization Error Bound for Different Loss Functions

As seen in Eq. 8.6, there are two parts to the loss function of the FNN, and each part is being monitored separately during training. As explained in Section 8.3, both loss functions L_1 and L_2 are locally Lipschitz continuous functions satisfying the following inequalities:

$$|L_1(\mathbf{y},\hat{\mathbf{y}}_2) - L_1(\mathbf{y},\hat{\mathbf{y}}_1)| \leq L_{r1}\,|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1| \tag{8.23a}$$

$$|L_2(\hat{\mathbf{y}}_2) - L_2(\hat{\mathbf{y}}_1)| \leq L_{r2}\,|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1| \tag{8.23b}$$

where L_{r1} and L_{r2} denote the local Lipschitz constant for loss functions L_1 and L_2 respectively. Note that L_1 is a function assessing the MSE between the true output $\mathbf{y}$ and the predicted output $\hat{\mathbf{y}}$, and L_2 is a function of the predicted output $\hat{\mathbf{y}}$ only (the explicit form of $\frac{\partial B}{\partial x}$, hence $L_f B(x)$, are not known ahead of time).

Therefore, we can develop the generalization error bound with respect to each loss function, and explain their respective implications. Here, we replace the general notations of FNN inputs $\mathbf{x}$ and output $\mathbf{y}$ with the specific variables under consideration in our case, which include states of the nonlinear system of Eq. 8.1 x as the inputs, barrier function value B as the true output, and $\hat{B}(x)$ as the predicted output. The expected loss of L_1 is upper bounded by the following inequality with

probability of at least $1-\delta$:

$$\mathbb{E}[L_1(\hat{B}(x),B)] \leq \mathscr{O}\left(L_{r1}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}+\frac{1}{m}\sum_{i=1}^{m}L_1(\hat{B}(x_i),B_i) \tag{8.24}$$

Since L_1 evaluates error between true FNN output (i.e., B) and predicted FNN output (i.e., $\hat{B}(x)$) in terms of MSE, the upper bound on $|\hat{B}-B|$ is:

$$|\hat{B}-B| \leq \sqrt{\mathscr{O}\left(L_{r1}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}+\frac{1}{m}\sum_{i=1}^{m}L_1(\hat{B}(x_i),B_i)} \tag{8.25}$$

We can further develop a bound on the value of $\hat{B}(x)$, which holds with probability of at least $1-\delta$ as follows:

$$\begin{aligned}
|\hat{B}| &= |\hat{B}+B-B| \\
&\leq |B|+|\hat{B}-B| \\
&\leq |B|+\sqrt{\mathscr{O}\left(L_{r1}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}+\frac{1}{m}\sum_{i=1}^{m}L_1(\hat{B}(x_i),B_i)}
\end{aligned} \tag{8.26}$$

Given the conditions of Eq. 8.3a and Eq. 8.3c, the true barrier function B take values of $+1$ for unsafe x, and -1 for safe x; therefore, $|B| \leq 1$ for all x in the operating region. In order to ensure that $\hat{B}$ satisfies $\hat{B} \leq 0$ for all safe x, and $\hat{B} > 0$ for all unsafe x, the upper bound on the modeling error of the barrier function output must be less than 1, thus,

$$\sqrt{\mathscr{O}\left(L_{r1}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}+\frac{1}{m}\sum_{i=1}^{m}L_1(\hat{B}(x_i),B_i)} \leq 1 \tag{8.27}$$

The FNN model must be trained and built by selecting the appropriate number of samples m, the depth of the network d, the bound on the weight matrices B_W such that this bound on the modeling error is satisfied.

Moreover, the generalization error bound of L_2 represents the upper bound of the expected value of L_2 when applied on testing data that has not been studied by the FNN. The generalization

error bound of L_2 can be written as follows:

$$\mathbb{E}[L_2(\hat{B}(x)] \leq \mathscr{O}\left(L_{r2}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}+\frac{1}{m}\sum_{i=1}^{m}L_2(\hat{B}(x_i)) \tag{8.28}$$

where the term $\frac{1}{m}\sum_{i=1}^{m}L_2(\hat{B}(x_i))$ represents the empirical loss of L_2 resulting from m data samples from the training dataset. As described in Section 8.2.1, we monitor L_2 during training and only stop training when L_2 reaches 0 for all training data samples. Therefore, $\frac{1}{m}\sum_{i=1}^{m}L_2(\hat{B}(x_i)) = 0$. Furthermore, by the law of large numbers, with sufficiently large number of data sample size, the sample mean can sufficiently approximate the real expected value. In this case, we can use the testing dataset empirical loss to approximate the expectation of L_2, which assesses $ReLu(L_f\hat{B}(x)+\tau_I)$ for x values that have not been studied by the FNN. We can further simplify Eq. 8.28 to the following form by utilizing the fact that the empirical loss of L_2 on the training dataset is 0:

$$\mathbb{E}[\frac{1}{N_{I_f}^{test}}\sum_{i=1}^{N_{I_f}^{test}}ReLu(L_f\hat{B}(x_i)+\tau_I)] \leq \mathscr{O}\left(L_{r2}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{8.29a}$$

$$\mathbb{E}[ReLu(L_f\hat{B}(x)+\tau_I)] \leq \mathscr{O}\left(L_{r2}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{8.29b}$$

where x_i for $i = 1,...,N_{I_f}^{test}$ represents safe states in the testing dataset at which $L_g\hat{B}(x_i) = 0$. In order to meet the condition of Eq. 8.3b for testing data points that have not been previously studied by the FNN, the following inequality must hold:

$$\mathscr{O}\left(L_{r2}d_y\frac{B_X(B_W)^d\sqrt{d+1+log(d_x)}}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}-\tau_I \leq 0 \tag{8.30}$$

By carefully choosing the number of layers to the FNN (depth d), the number of training sample size m, the upper bounds on weight matrices B_W, as well as the upper bound on the input vector B_X by selecting the range of states considered in the compact set in state space appropriately, we build a FNN that satisfies Eq. 8.30, and in turn, ensures that $L_f\hat{B}(x) \leq 0$ in the operating region for which we consider the states are constrained within with probability $1-\delta$.

8.4 Probabilistic Stabilization and Safety via Control Lyapunov-Barrier Function

The Control Lyapunov-Barrier Function (CLBF) in the form of a weighted average of CLF and CBF was proposed in [90], and it shows that when a CLBF exists for the system of Eq. 8.1, there exists a controller $u = \Phi(x)$ that keeps the closed-loop state bounded within a level set of the CLBF and outside of the unsafe set $\mathscr{D}$ for all times for any initial condition $x_0 \in \mathscr{X}_0$. This work is further extended in [119, 122] to account for input constraints in the system and the constrained CLBF was presented. Furthermore, a constrained CLBF-MPC where the prediction model inside the MPC was developed using an ensemble of Recurrent Neural Network (RNN) models was proposed in [123]. Based on this work, we proposed a machine-learning-based CLBF-MPC in [25] where the CBF is built using an FNN model to characterize the safety status of the states inside the operating region, and the MPC uses an RNN model for its predictions. In this work, we provide statistical analysis on the probability of stabilization and safety of a CLBF-based controller where the CBF is built using an FNN, first under the control law $u = \Phi(x) \in U$ for the nonlinear system of Eq. 8.1, then under the CLBF-MPC where MPC uses the first-principles model in the form of ODE as described by Eq. 8.1 to predict future states. The FNN-CBF $\hat{B}$ can be shown to meet the conditions outlined in Eq. 8.3 in probability with proper model construction, parameter selection, and post-training verification. Therefore, it can be readily used as a valid CBF in the design of CLBF. The constrained CLBF built using the FNN-CBF $\hat{B}$ is defined as follows:

Definition 8.6. *Given a set of unsafe points in state-space $\mathscr{D}$, a proper, lower-bounded and $\mathscr{C}^1$ function $\hat{W}(x) : \mathbf{R}^n \to \mathbf{R}$ is a constrained CLBF if $\hat{W}(x)$ has a minimum at the origin and also satisfies the following properties:*

$$\hat{W}(x) > \rho, \quad \forall\, x \in \mathscr{D} \subset \phi_{uc} \tag{8.31a}$$

$$\left| \frac{\partial \hat{W}(x)}{\partial x} \right| \leq r_W(|x|) \tag{8.31b}$$

$$L_f \hat{W}(x) < 0,$$

$$\forall\, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathscr{X}_e) \mid L_g \hat{W}(z) = 0\} \tag{8.31c}$$

$$\mathscr{U}_\rho := \{x \in \phi_{uc} \mid \hat{W}(x) \leq \rho\} \neq \emptyset \tag{8.31d}$$

$$\overline{\phi_{uc} \backslash (\mathscr{D} \cup \mathscr{U}_\rho)} \cap \overline{\mathscr{D}} = \emptyset \tag{8.31e}$$

where f and g are from the nonlinear model in Eq. 8.1, $\rho \in \mathbf{R}$ is a constant, r_W is a class $\mathscr{K}$

function, $\mathscr{X}_e := \{x \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\}) \mid \partial \hat{W}(x)/\partial x = 0\}$ is a set of states for the nonlinear model of Eq. 8.1 where $L_f \hat{W}(x) = 0$ (for $x \neq 0$) due to $\partial \hat{W}(x)/\partial x = 0$. If $\hat{W}(x)$ exists for the nonlinear system of Eq. 8.1 as defined in Eq. 8.6, then there exists a control law $u = \Phi(x) \in U$ such that the origin of the system is rendered asymptotically stable within a region ϕ_{uc}, which is defined as the union of the origin, and the set $\mathscr{X}_e$, and the set for which the time-derivative of $\hat{W}(x)$ is negative with constrained inputs: $\phi_{uc} = \{x \in \mathbf{R}^n \mid \{0\} \cup \mathscr{X}_e \cup \dot{\hat{W}}(x(t), \Phi(x)) = L_f \hat{W} + L_g \hat{W} \cdot u < -\alpha_W |\hat{W}(x) - \hat{W}(0)|, u = \Phi(x) \in U\}$, and $\alpha_W > 0$ is a real constant used to characterize the set ϕ_{uc}. An example of such control law $\Phi(x)$ takes the form of the Lyapunov-based universal Sontag law [67] with the Lyapunov function $V(x)$ replaced by the CLBF $\hat{W}(x)$; details can be found in [119, 122, 124].

8.4.1 Design of Constrained CLBF

The design of CLBF can be carried out following the practical design guidelines in [119], by first designing valid CLF and CBF that meet their conditions outlined in Eq. 8.2 and Eq. 8.3 respectively. This design method is further expanded and proven in [25] in the case of FNN-based CBF and RNN-based process model, and it was shown that through a FNN-CBF $\hat{B}(x)$ that meets all its required conditions, the resulting machine-learning based $\hat{W}(x)$ has a global minimum at the origin and is able to meet all its requirements of Eq. 8.31. The proof for the following proposition can be found in [122] and [25] and will be omitted here. In this work, we have introduced the statistical analysis on the generalization error of the FNN-CBF $\hat{B}$. Accounting for the general expected error of $\hat{B}$, the FNN-CBF $\hat{B}$ is shown to meet all the requirements of Eq. 8.3 with probability of $1 - \delta$ if the two conditions on the modeling error bound shown in Eq. 8.27 and Eq. 8.30 are met. Therefore, the properties of the resulting CLBF $\hat{W}$ as well as the associated safety and stabilizability properties of the CLBF-based controller will also hold with probability $1 - \delta$.

Proposition 8.1. *Consider the $\mathscr{C}^1$ FNN-CBF $\hat{B}(x) : \mathbf{R}^n \to \mathbf{R}$, trained using the dataset S_s consisting m i.i.d data samples satisfying Assumptions 8.1-8.4, and has a resulting loss function errors constrained by Eq. 8.27 and Eq. 8.30. Given an open set $\mathscr{D}$ of unsafe states for the system of Eq. 8.1, assume that there exists a $\mathscr{C}^1$ CLF $V : \mathbf{R}^n \to \mathbf{R}_+$, such that the following conditions hold:*

$$\mathscr{D} \subset \mathscr{H} \subset \mathscr{H}' \subset \phi_{uc},\ 0 \notin \mathscr{H},\ 0 \notin \mathscr{H}' \tag{8.32}$$

$$\hat{B}(x) = -\eta < 0,\ \forall x \in \mathbf{R}^n \backslash \mathscr{H}';\ \hat{B}(x) > 0,\ \forall x \in \mathscr{H}' \tag{8.33}$$

where $\mathscr{H}$ and $\mathscr{H}'$ are both compact and connected sets within ϕ_{uc}, and $\mathscr{H}'$ encloses $\mathscr{H}$ with sufficient margin accounting for modeling errors in $\hat{B}(x)$. Consider $\hat{W}(x)$ designed as $\hat{W}(x) := V(x) + \mu\hat{B}(x) + \nu$, and satisfies:

$$\left|\frac{\partial \hat{W}(x)}{\partial x}\right| \leq r_W(|x|) \tag{8.34}$$

$$\begin{aligned} L_f\hat{W}(x) < 0, \\ \forall\, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathscr{X}_e) \mid L_g\hat{W}(z) = 0\} \end{aligned} \tag{8.35}$$

$$\mu > \frac{c_2c_3 - c_1c_4}{\eta}, \tag{8.36a}$$

$$\nu = \rho - c_1c_4, \tag{8.36b}$$

$$c_3 := \max_{x \in \partial\mathscr{H}'} |x|^2, \tag{8.36c}$$

$$c_4 := \min_{x \in \partial\mathscr{D}} |x|^2 \tag{8.36d}$$

then, with probability of at least $1-\delta$, the control law $\Phi(x) \in U$ (Lyapunov-based Sontag control law with $V(x)$ replaced by $\hat{W}(x)$) guarantees that, for any initial state $x_0 \in \phi_{uc} \backslash \mathscr{D}_{\mathscr{H}'}$, where $\mathscr{D}_{\mathscr{H}'} := \{x \in \mathscr{H}' \mid \hat{W}(x) > \rho\}$, the state is bounded in $\phi_{uc} \backslash \mathscr{H}$ and does not enter the unsafe region $\mathscr{H}$ for all $t > 0$.

Proof. Through the selection of parameters μ and ν, the conditions of Eq. 8.31a and Eq. 8.31e can be met. The proofs for these two conditions are shown in [119] and will be omitted here. We will focus on how the conditions of Eq. 8.31b and Eq. 8.31c can be met. Given that Eq. 8.27 and Eq. 8.30 are met, $\hat{B}(x)$ satisfies the CBF properties presented in Eq. 8.3 with probability at least $1-\delta$. From Eq. 8.2b and Eq. 8.7, as well as the way the CLBF is constructed $\hat{W}(x) := V(x) + \mu\hat{B}(x) + \nu$, we have the following:

$$\begin{aligned} \left|\frac{\partial \hat{W}(x)}{\partial x}\right| &= \left|\frac{\partial V}{\partial x} + \mu\frac{\partial \hat{B}}{\partial x}\right| \\ &\leq r_V(|x|) + \mu r_B(|x|) \\ &\leq r_W(|x|) \end{aligned} \tag{8.37}$$

where r_W, as the weighted sum of two class $\mathscr{K}$ functions r_V and r_B, is also a class $\mathscr{K}$ function. Thus, it is shown that Eq. 8.31b is satisfied. Similarly, for all $x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup$

$\mathscr{X}_e) \mid L_g\hat{W}(z) = 0\}$, Eq. 8.31c can be also shown to hold with the following derivation:

$$L_f\hat{W}(x) = L_fV(x) + \mu L_f\hat{B}(x) < 0 \tag{8.38}$$

Thus, Eq. 8.31b and Eq. 8.31c are both satisfied. In addition, the global minimum of $V(x)$ is at the origin, i.e., $V(0) = 0$, and $V(x) > 0$ for all $x \in \mathbf{R}^n \backslash \{0\}$. With a sufficiently small modeling error as characterized by its generalization error bound, $\hat{B}(x) = -1$ for all $x \in \phi_{uc} \backslash \mathscr{H}'$, where $\{0\} \in \phi_{uc} \backslash \mathscr{H}'$, and $\hat{B}(x) = +1$ for all $x \in \mathscr{H}'$ in probability. Hence, $\hat{B}(x)$ also has a global minimum at the origin in probability. Since $\hat{W}(x)$ is a weighted average of $V(x)$ and $\hat{B}(x)$, the global minimum of $\hat{W}(x)$ is at the origin. Therefore, it has been demonstrated that a CLBF $\hat{W}(x)$ and the control law $u = \Phi(x) \in U$ exist that satisfy all conditions of Eq. 8.31 with probability $1 - \delta$, and guarantee the safety and asymptotic stability of the states for all $x_0 \in \phi_{uc} \backslash \mathscr{D}_{\mathscr{H}'}$. □

We specify the set of initial conditions considered in our study as $\mathscr{U}_\rho$, which is a level set of $\hat{W}(x)$ as described in Eq. 8.31d. Since $\dot{\hat{W}}(x) = 0$ for $x = 0$ and $x = x_e \in \mathscr{X}_e$, and $\dot{\hat{W}}(x) < 0$ within the set $\phi_{uc} \backslash (\mathscr{X}_e \cup 0)$ under the control law $u = \Phi(x) \in U$, it holds that $\dot{\hat{W}}(x) \leq 0$ for all $x \in \mathscr{U}_\rho$. We know that $\hat{W}(x)$ is a proper function, therefore the level set of $\hat{W}(x)$, $\mathscr{U}_\rho$, is a compact, forward invariant set. For any initial condition $x_0 \in \mathscr{U}_\rho$, the closed-loop state $x(t)$ is bounded in $\mathscr{U}_\rho$ under the continuous control law $u = \Phi(x) \in U$. Furthermore, since the set $\mathscr{U}_\rho$ has no intersection with the set $\mathscr{D}_{\mathscr{H}'}$, the closed-loop state will not enter the unsafe set $\mathscr{D}_{\mathscr{H}'}$ characterized by Proposition 8.1.

For bounded unsafe sets (e.g., the entire unsafe region occurs as an obstacle in the middle of the operating region), there are stationary points in state space (in addition to the origin), denoted as $x_e \in \mathscr{X}_e$ where $\dot{\hat{W}} = 0$, which can be considered as saddle points. When states reach these stationary points, the continuous controller $u = \Phi(x) \in U$ is incapable of steering the states away from these points and they will remain there and become trapped. Thus, we design discontinuous control actions $u = \bar{u}(x) \in U$ that can drive the states away from x_e in a path of decreasing $\hat{W}(x)$. Once the states leave x_e under $\bar{u}(x)$, then the controller $u = \Phi(x) \in U$ is able to continue driving the state towards the origin asymptotically since $\dot{\hat{W}}(x) < 0$ for all $x \in \mathscr{U}_\rho \backslash (\mathscr{X}_e \cup 0)$. In the case of unbounded unsafe sets, the origin will be the only stationary point in state-space, therefore the CLBF-based control law $u = \Phi(x) \in U$ is able to ensure asymptotic stability and safety.

8.4.2 Sample-and-hold Implementation of CLBF-based Controller

We have shown that if there exists a constrained CLBF $\hat{W}$ built from FNN-CBF $\hat{B}$ that meets the conditions of Eq. 8.31 and a set of control law $u = \Phi(x) \in U$ that is continuously implemented, the closed-loop state can be maintained within the safe region for all times. This CLBF-based control law $u = \Phi(x) \in U$ is used to design CLBF-based constraints in MPC. As the MPC is executed every sampling period Δ, the control law will be implemented in a sample-and-hold manner. Therefore, we will now discuss the impact of sample-and-hold application of control actions on the probabilistic stability and safety of the nonlinear system of Eq. 8.1.

We consider the region $\mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_s} \cup \mathscr{B}_\delta(x_e))$, where $\rho_s < \rho_{min} < \rho$, and prove that for all $x(t_k)$ in this region, $\dot{W}(x(t), u(t)) < -\varepsilon$ where $u(t)$ is applied in a sample-and-hold manner $u(t) = u(t_k) = \Phi(x(t_k))$, $\forall\, t \in [t_k, t_k + \Delta')$. Since this region is a bounded region within ϕ_{uc} and the functions $f(\cdot)$ and $g(\cdot)$ are continuous, we have the following inequalities:

$$\dot{\hat{W}}(x(t_k)) < -\alpha_W |\hat{W}(x) - \hat{W}(0)| < -\alpha_W \rho_0 \tag{8.39a}$$

$$|x(t) - x(t_k)| \leq k_1 \Delta',\ \forall\, t \in [t_k, t_k + \Delta') \tag{8.39b}$$

where k_1 is a positive real number and $\Delta' > 0$ represents a sampling period, where the sampling period of the CLBF-based controller and CLBF-MPC Δ will be taken from the range $\Delta \in (0, \Delta^*]$. Eq. 8.39a comes from the definition of the region ϕ_{uc}, and $\rho_0 := \min_{x \in \mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_s} \cup \mathscr{B}_\delta(x_e))} |\hat{W}(x) - \hat{W}(0)|$, and $\hat{W}(0)$ is the minimum of $\hat{W}(x)$ which is found at the origin. Furthermore, since $\hat{W}(x)$ is a $\mathscr{C}^1$ function that meets the property of Eq. 8.31b, and considering the fact that $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth functions, we have the following inequalities:

$$|L_f \hat{W}(x(t)) - L_f \hat{W}(x(t_k))| \leq k_2 |x(t) - x(t_k)| \tag{8.40a}$$

$$|(L_g \hat{W}(x(t)) - L_g \hat{W}(x(t_k)))u(t)| \leq k_3 |x(t) - x(t_k)| \tag{8.40b}$$

where k_2 and k_3 are positive real numbers. With these inequalities established, the following proposition is presented to show that with sufficient conditions, the controller $u = \Phi(x) \in U$ designed based on the FNN-based CLBF $\hat{W}(x)$ and the discontinuous control law $u = \bar{u}(x) \in U$ are able to guarantee closed-loop stability and safety for the nonlinear system in Eq. 8.1.

Proposition 8.2. *Consider the nonlinear system of Eq. 8.1 with a FNN-based CLBF $\hat{W}(x)$ designed based on a valid CLF $V(x)$ and a valid FNN-CBF $\hat{B}(x)$ that satisfies Eq. 8.3 with probability of at*

least $1-\delta$. *There exists* $\varepsilon > 0, \Delta' > 0, \Delta'' > 0, \rho > \rho_{min} > \rho_s$ *that satisfy:*

$$\Delta' < \frac{\alpha_W \rho_0 - \varepsilon}{k_1(k_2+k_3)},\ 0 \leq \varepsilon < \alpha_W \rho_0 \tag{8.41a}$$

$$\rho_{min} := \max_{\Delta t \in [0,\Delta'')} \{\hat{W}(x(t_k+\Delta t)) \mid x(t_k) \in \mathscr{U}_{\rho_s},\ u \in U\} \tag{8.41b}$$

$$\Delta^* = \min\{\Delta', \Delta''\} \tag{8.41c}$$

such that, for any $x(t_k) \in \mathscr{U}_\rho$, *under the sample-and-hold application of either* $u(t) = \Phi(x(t_k))\ \forall t \in [t_k, t_{k+1})$ *where* $t_{k+1} = t_k + \Delta$ *and* $\Delta \in (0, \Delta^*]$, *or* $u(t) = \bar{u}(x(t_k)) \in U$ *when* $x(t_k) \in \mathscr{B}_\delta(x_e)$, $\hat{W}(x)$ *is guaranteed to decrease over one sampling period with probability of at least* $1-\delta$, *and* $x(t)$ *is bounded in* $\mathscr{U}_\rho$ *for all times and ultimately converges to* $\mathscr{U}_{\rho_{min}}$.

Proof. We first consider the case of bounded unsafe sets in state space. We will first prove that the closed-loop state trajectory $x(t)$ will be bounded in $\mathscr{U}_\rho$ and will enter $\mathscr{U}_{\rho_s}$ in finite steps under the sample-and-hold implementation of control actions $u = \Phi(x) \in U$ or $u = \bar{u}(x) \in U$ if $x \in \mathscr{B}_\delta(x_e)$. Then we will prove that once the state enters $\mathscr{U}_{\rho_s}$, i.e., $x(t_k) \in \mathscr{U}_{\rho_s}$, $x(t)$ will stay in $\mathscr{U}_{\rho_{min}}$ for $t = [t_k, t_k + \Delta'')$.

Under the sample-and-hold implementation of $u(t)$, for $x(t_k) \in \mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_s} \cup \mathscr{B}_\delta(x_e))$, we can write $\dot{\hat{W}}(x)$ as follows:

$$\begin{aligned}
\dot{\hat{W}}(x(t),u(t)) =& \dot{\hat{W}}(x(t_k),u(t_k)) + (\dot{\hat{W}}(x(t),u(t)) \\
&- \dot{\hat{W}}(x(t_k),u(t_k))) \\
=& L_f\hat{W}(x(t_k)) + L_g\hat{W}(x(t_k))u(t_k) \\
&+ (L_f\hat{W}(x(t)) - L_f\hat{W}(x(t_k))) \\
&+ (L_g\hat{W}(x(t)) - L_g\hat{W}(x(t_k)))u(t)
\end{aligned} \tag{8.42}$$

Substituting Eq. 8.39a, Eq. 8.40 and Eq. 8.39b, we derive the following inequality:

$$\begin{aligned}
\dot{\hat{W}}(x(t),u(t)) &< -\alpha_W \rho_0 + k_1(k_2+k_3)\Delta' \\
&< -\varepsilon
\end{aligned} \tag{8.43}$$

which sufficiently shows that under sample-and-hold implementation of control actions $u(t)$, $\dot{\hat{W}}(x)$ can be rendered negative for any $x(t_k) \in \mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_s} \cup \mathscr{B}_\delta(x_e))$, and $\hat{W}(x(t)) < \hat{W}(x(t_k)) \leq \rho$, therefore bounded within $\mathscr{U}_\rho\ \forall t > t_k$. Within finite steps, $x(t)$ will eventually enter $\mathscr{U}_{\rho_s}$.

For bounded unsafe sets where stationary points in state-space exist, consider $x(t_k) \in \mathscr{B}_\delta(x_e)$.

$x(t_{k+1})$ can be driven to a smaller level set of $\hat{W}(x)$ under the discontinuous control law $u = \bar{u}(x) \in U$ which decreases $\hat{W}(x)$ over one sampling period; i.e., $\hat{W}(x(t_{k+1})) < \hat{W}(x(t_k))$. Within finite sampling periods, the closed-loop state will eventually leave $\mathscr{B}_\delta(x_e)$, and will never return since the control law $u = \Phi(x) \in U$ will take over and ensure that $\hat{W}(x(t)) < \hat{W}(x(t_k))$ for all $t > t_k$.

Once the state enters the set $\mathscr{U}_{\rho_s}$, $x(t_k) \in \mathscr{U}_{\rho_s}$, the definition of $\mathscr{U}_{\rho_{min}}$ in Eq. 8.41b shows that the trajectory $x(t)$ will stay in $\mathscr{U}_{\rho_{min}}$ for $t \in [t_k, t_k + \Delta'')$. We choose a maximal sampling period Δ^* which is the minimum of Δ' and Δ'' as described by Eq. 8.41c, and choose a sampling period $\Delta \in (0, \Delta^*]$. Within $t \in [t_k, t_k + \Delta)$, under the sample-and-hold implementation of $u = \Phi(x) \in U$ or $u = \bar{u} \in U$, we are able to show that, with probability at least $1 - \delta$, for $x(t_k) \in \mathscr{U}_\rho \backslash \mathscr{U}_{\rho_s}$, $x(t)$ moves towards the origin into smaller level sets of $\hat{W}$ and eventually into the level set $\mathscr{U}_{\rho_s}$, and for $x(t_k) \in \mathscr{U}_{\rho_s}$, $x(t)$ remains in $\mathscr{U}_{\rho_{min}}$. Since the CLBF properties on $\hat{W}(x)$ are satisfied with a probability of at least $1 - \delta$, the closed-loop stability and safety of the system under the sample-and-hold implementation of CLBF-based control laws also follow the same probability.

In the case of unbounded unsafe sets, stationary points other than the origin $\mathscr{B}_\delta(x_e)$ do not exist, therefore, the sample-and-hold control actions $u = \Phi(x) \in U$ are able to drive closed-loop state towards smaller level sets of $\hat{W}(x)$ since $\dot{\hat{W}}(x, \Phi(x)) < 0$ holds, and similarly, will be bounded within $\mathscr{U}_{\rho_{min}}$ eventually.

□

8.4.3 FNN-CLBF-based MPC

Given the probabilistic stability and safety analysis provided by the sample-and-hold implementation of FNN-CLBF-based control laws $u = \Phi(x) \in U$, the FNN-CLBF-based MPC

is formulated as follows:

$$\mathscr{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) dt \tag{8.44a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = f(\tilde{x}(t)) + g(u(t)) \tag{8.44b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{8.44c}$$

$$u(t) \in U, \ \forall \, t \in [t_k, t_{k+N}) \tag{8.44d}$$

$$\dot{\hat{W}}(x(t_k), u(t_k)) \leq \dot{\hat{W}}(x(t_k), \Phi(x(t_k))) \quad \text{if } x(t_k) \notin \mathscr{B}_\delta(x_e) \text{ and } \hat{W}(x(t_k)) > \rho_{min} \tag{8.44e}$$

$$\hat{W}(\tilde{x}(t)) \leq \rho_{min}, \ \forall \, t \in [t_k, t_{k+N}), \text{ if } \hat{W}(x(t_k)) \leq \rho_{min} \tag{8.44f}$$

$$\hat{W}(\tilde{x}(t)) < \hat{W}(x(t_k)), \ \forall \, t \in (t_k, t_{k+N}), \quad \text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{8.44g}$$

where the state trajectory predicted by the ODE model of Eq. 8.1 is represented by $\tilde{x}(t)$, the number of sampling periods in the prediction horizon is denoted by N, and $S(\Delta)$ is a piece-wise constant function with a sampling time Δ. This optimization problem of Eq. 8.44 is solved by the MPC every time a new measurement is received (every Δ), and the optimization problem has an objective function Eq. 8.44a that is in the form of the integral of $L(\tilde{x}(t), u(t)) = \tilde{x}^T Q \tilde{x} + u^T R u$ over the prediction horizon. Here, Q, R are positive definite weight matrices. The objective function is formulated this way such that it has a minimum at the origin. Eq. 8.44d describes the constraints imposed on the input vector along the predicted trajectory. It is assumed that state measurements are received at every sampling period. As seen in Eq. 8.44c, the initial condition of the predicted state trajectory in Eq. 8.44b are obtained from the feedback state measurements at $t = t_k$. The constraints of Eqs. 8.44e-8.44g are used to ensure closed-loop stability and safety. When $x(t_k) \notin \mathscr{B}_\delta(x_e)$ and $\hat{W}(x(t_k)) > \rho_{min}$, the constraint in Eq. 8.44e decreases $\hat{W}(\tilde{x})$ at a rate at least of the rate achieved by the CLBF-based controller $u = \Phi(x) \in U$. When $\hat{W}(x(t_k)) \leq \rho_{min}$, Eq. 8.44f maintains the closed-loop state trajectory over the prediction horizon inside the level set $\mathscr{U}_{\rho_{min}}$. If $x(t_k) \in \mathscr{B}_\delta(x_e)$, Eq. 8.44g is activated to decrease $\hat{W}(x)$ over the next sampling period so that the state will escape the saddle point within finite steps. The first control action $u^*(t_k)$ of the optimized input trajectory $u^*(t)$ will be applied in a sample-and-hold manner for the next sampling period. After that, the horizon will move forward one sampling period, and the above optimization problem is solved again.

The CLBF used in the CLBF-MPC of Eq. 8.44 is one constructed using an FNN-based CBF

$\hat{B}(x)$, which is well-trained and designed to satisfy modeling error constraints in Eq. 8.27 and Eq. 8.30. Subsequently, with probability at least $1-\delta$, $\hat{B}(x)$ meets the conditions of Eq. 8.3, CLBF meets the conditions of Eq. 8.31 via the design method presented in Proposition 8.1, and therefore, probabilistic safety and stability under the CLBF-based control laws are provided. The following theorem will demonstrate that probabilistic stability and safety can be established under the CLBF-MPC of Eq. 8.44.

Theorem 8.3. *Consider the nonlinear system of Eq. 8.1 with a constrained CLBF $\hat{W}(x)$ built following Proposition 8.1 using a FNN-CBF $\hat{B}(x)$ that satisfies Eq. 8.27 and Eq. 8.30 and meets the conditions of Eq. 8.3 with probability of at least $1-\delta$. Let $\Delta > 0$, $\varepsilon > 0, \rho > \rho_{min} > \rho_s$ satisfy the requirements in Proposition 8.2. Given $x_0 \in \mathscr{U}_\rho$, with probability of at least $1-\delta$, recursive feasibility can be guaranteed for the optimization problem of q. 8.44, and the closed-loop state is bounded in $\mathscr{U}_\rho$, $\forall\, t \geq 0$, and converges to $\mathscr{U}_{\rho_{min}}$ as $t \to \infty$.*

Proof. There always exists a feasible solution for the CLBF-MPC optimization problem since sample-and-hold implementation of the CLBF-based control law $u = \Phi(x) \in U$ (when $x(t_k) \in \mathscr{U}_\rho \backslash \mathscr{B}_\delta(x_e)$) and the discontinuous control law $u = \bar{u} \in U$ (when $x(t_k) \in \mathscr{B}_\delta(x_e)$ in the case of bounded unsafe sets) provide one such solution that satisfy the constraints of Eqs. 8.44d-8.44g for all $x(t_k) \in \mathscr{U}_\rho$. This has been proven in Proposition 8.2. The properties Eq. 8.33 ensure that the CBF $\hat{B}$ is able to discern the unsafe region from the safe region accurately with a probability of at least $1-\delta$. Furthermore, it has been shown in Proposition 8.1 that $\dot{\hat{W}}(x) \leq 0$ is held with probability at least $1-\delta$ in the region $\mathscr{U}_\rho$.

For unbounded unsafe sets, there are no stationary points in the operating region other than the origin. For any $x_0 \in \mathscr{U}_\rho \backslash (\mathscr{B}_\delta(x_e) \cup \mathscr{U}_{\rho_{min}})$, Eq. 8.44e forces the optimal control action calculated by the FNN-CLBF-based MPC $u^*(t_k)$ to decrease $\hat{W}(x)$ at a rate at least as fast as that achieved by the control law $\Phi(x(t_k))$. Therefore, $u^*(t_k)$ will drive the closed-loop state towards the origin and into $\mathscr{U}_{\rho_{min}}$ within finite steps. After that, Eq. 8.44f ensures that the closed-loop state remains inside $\mathscr{U}_{\rho_{min}}$. We can conclude that the closed-loop state under the CLBF-MPC will be bounded in $\mathscr{U}_\rho$ for $t > 0$ and eventually be bounded in $\mathscr{U}_{\rho_{min}}$, thus will not enter the unsafe set $\mathscr{D}$ for all times since the safe set $\mathscr{U}_\rho$ has no intersection with the unsafe set $\mathscr{D}$.

In the case of bounded unsafe sets, when the closed-loop state reaches a stationary point, $x(t_k) \in \mathscr{B}_\delta(x_e)$, Eq. 8.44g is activated to ensure that the optimal solution of the MPC drives the state away from the stationary point in a direction of decreasing $\hat{W}$. After the state escapes the neighborhood around the saddle point, Eqs. 8.44e-8.44f will continue to ensure that $x(t)$ is bounded in $\mathscr{U}_\rho$ and eventually converges to $\mathscr{U}_{\rho_{min}}$ without entering the bounded unsafe set.

□

When Eq. 8.44e is activated, the FNN-CBF is used to predict the corresponding barrier function value $\hat{B}$ based on $x(t_k)$. This $\hat{B}(x(t_k))$ prediction is shown to satisfy the CBF properties of Eq. 8.3 with probability of at least $1-\delta$, and therefore stability and safety properties enforced by Eq. 8.44e are achieved with a probability of at least $1-\delta$. When Eq. 8.44f or Eq. 8.44g are activated, FNN predictions of the barrier function are carried out for the entire trajectory $\hat{B}(\tilde{x}(t))$ for $t \in [t_k, t_{k+N}]$. Each of the FNN inputs, $\tilde{x}(t)$, are calculated based on the ODE model of Eq. 8.1, which are accurate assuming there are no modeling mismatches. The predictions $\hat{B}(\tilde{x}(t))$ based on $\tilde{x}(t)$ are therefore independent predictions and do not affect one another. At each time step of the trajectory in the MPC prediction horizon, the probability of the actual closed-loop state being maintained inside $\mathscr{U}_{\rho_{min}}$ (in the case of Eq. 8.44f), or the actual closed-loop state being driven around the unsafe set in the direction of decreasing CLBF (in the case of Eq. 8.44f) is at least $1-\delta$. However, to ensure that the entire trajectory satisfies its safety and stability properties, the probability will be reduced (specifically, $(1-\delta)^N$ for N time steps in the prediction horizon). Although the overall probability of stability and safety for this predicted trajectory is reduced, the stability and safety properties of the system under the first control action $u^*(t_k)$ of the FNN-CLBF-MPC for the current time step $t = t_k$ is guaranteed with probability $1-\delta$. When a new feedback measurement is received, the MPC is executed again and computes a new control action to be applied that ensures stability and safety with probability $1-\delta$ over the next sampling step.

Remark 8.2. *In this study, we study the generalization error bound of the FNN-CBF and the probabilistic closed-loop stability and safety properties of the FNN-CLBF-MPC where the MPC uses the first-principles model for prediction. In our previous work in [25], we have also developed FNN-CLBF-MPC systems where the MPC can use a prediction model of the nonlinear process built using recurrent neural networks (RNN). Similar to the FNN used in this study, with a neural-network-based model, there exists an expected error in the predicted output $\hat{x}$ of the nonlinear system that can be upper-bounded following machine learning theory; this has been developed in [126]. In our previous work [25], we have discussed design methods with data generation and unsafe region characterization to account for both modeling error in the FNN-CBF and in the RNN process model, as well as with numerical approximations of the predicted vector and matrix functions $\hat{f}$ and $\hat{g}$. We have demonstrated through theoretical development as well as closed-loop simulations that with adequate design and verification of the FNN-CBF as well as sufficient boundedness of the modeling and numerical errors, closed-loop stability and safety can be achieved for FNN-CLBF-MPC using both first-principles and RNN models. In this work,*

we have only conducted closed-loop studies on FNN-CLBF-MPC using a first-principles model as the focus of this manuscript is on the generalization error upper bound of the FNN model. We can easily extend the statistical stability and safety analysis to FNN-CLBF-MPC using RNN models by following the work in [126], where we can further specify the upper bound on the modeling error of the RNN process model as it depends on a number of factors such as sample size, weight matrix bounds, input length, and network complexity, and in turn construct the RNN to meet Lyapunov-based stability properties in probability.

8.5 Application to a Chemical Process Example

8.5.1 Preliminaries

A chemical process example is simulated to demonstrate the effectiveness of the FNN-based CLBF in ensuring the closed-loop stability and safety of a nonlinear process, and to demonstrate how various aspects of FNN design and training may impact the outcome of the FNN model. The system we consider is a continuously stirred tank reactor (CSTR) which is non-isothermal and assumed to be well-mixed, undergoing a second-order, exothermic, irreversible reaction converting reactant A into product B. There is a heating jacket equipped to remove and supply heat. The process dynamics can be modelled by material and energy balances as shown below:

$$\frac{dC_A}{dt} = \frac{F}{V_L}(C_{A0} - C_A) - k_0 e^{-E/RT} C_A^2 \tag{8.45a}$$

$$\frac{dT}{dt} = \frac{F}{V_L}(T_0 - T) - \frac{\Delta H k_0}{\rho_L C_p} e^{-E/RT} C_A^2 + \frac{Q}{\rho_L C_p V_L} \tag{8.45b}$$

where the two states of the system, C_A and T, are the concentration of A in the tank and the temperature inside the tank, respectively. V_L, F, T_0 represent the volume of the reacting fluid in the reactor, volumetric flow rate of the feed, and temperature of the feed, respectively. Q denotes the heat jacket input rate, and C_{A0} denotes the feed concentration of reactant A. It is assumed that the reacting liquid has a constant heat capacity C_p and density ρ_L. Other constants such as the pre-exponential constant, ideal gas constant, enthalpy and activation energy of the reaction are denoted by k_0, R, ΔH, and E, respectively. The values of these process parameters are given in [128].

The CSTR process is stabilized at its unstable equilibrium point $(C_{As},\ T_s) =$

$(1.954\ kmol/m^3,\ 401.9\ K)$ by the CLBF-based MPC, which manipulates the inputs C_{A0} and Q with corresponding steady-state values $(C_{A0_s}\ Q_s) = (4\ kmol/m^3,\ 0\ kJ/hr)$. The manipulated inputs have the following bounds: $|\Delta C_{A0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q| \leq 5 \times 10^5\ kJ/hr$, which originate from physical constraints. The states and the inputs of the system are represented in deviation variable for the subsequent analyses such that the equilibrium point of Eq. 8.45 is at the origin, i.e., $[\Delta C_A = C_A - C_{As},\ \Delta T = T - T_s]$, and $[\Delta C_{A0} = C_{A0} - C_{A0_s},\ \Delta Q = Q - Q_s]$. For simplicity of notation, the state and input vectors are represented in the following forms: $x^T = [\Delta C_A\ \Delta T]$ and $u^T = [\Delta C_{A0}\ \Delta Q]$. The CLBF-MPC is executed every sampling period where $\Delta = 10^{-3}\ hr$, where the nonlinear optimization problem of Eq. 8.44 is solved using the python module PyIpopt. To simulate the CSTR process and predict the state trajectory inside the MPC, the system of ODE of Eq. 8.45 is solved using the explicit Euler method with an integration time step of $h_c = 10^{-5}\ hr$.

We use the following positive definite P matrix to build a CLF $V(x)$ in the form of $V(x) = x^T Px$:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{8.46}$$

where the values of the P matrix are determined via extensive closed-loop simulations of the process. The unsafe region $\mathscr{D}$ can be either bounded or unbounded, and is a set within the stability region such that the state may enter the unsafe region on its path while it is driven towards the origin under a control law that does not consider safety constraints. The CLBF-MPC accounts for these unsafe regions in state-space and is capable of navigating the states around the unsafe set and towards the equilibrium point thereafter.

8.5.2 Development of the FNN Model for Barrier Function

The control barrier function within the CLBF is built using an FNN model, which takes x as inputs and computes the value $\hat{B}(x)$. In this study, we consider the cases of both bounded and unbounded unsafe regions. First, the bounded unsafe set is considered, where the unsafe region is defined as $\mathscr{D}_b := \{x \in \mathbf{R}^2 \mid F_b(x) = \frac{(x_1+0.92)^2}{1} + \frac{(x_2-42)^2}{500} < 0.06\}$. $\mathscr{H}_b$ is defined as $\mathscr{H}_b := \{x \in \mathbf{R}^2 \mid F_b(x) < 0.07\}$ such that it satisfies $\mathscr{D}_b \subset \mathscr{H}_b \subset \phi_{uc}$ in Proposition 8.1. The unsafe region is an ellipse embedded in the operating region to demonstrate the challenging case of a bounded unsafe set obstructing the trajectory of the closed-loop state. Practically, the unsafe sets may not be easily represented in a closed form function. However, based on engineering knowledge or simulations, one may collect sufficiently dense data in the operating region with corresponding

labels indicating whether the data point is safe or unsafe. Following this, we can obtain respective sets of data samples that are labelled as unsafe and safe, and can be subsequently used for model training. In our study, after specifying the region of unsafe operation, we generate training data for the FNN model. This is done by first specifying a region which the system is likely operated within, in this case, we specify $V(x) \leq 368$, which is a level set of CLF characterized as the stability region in the absence of unsafe sets under the use of Lyapunov-based control laws. Then, we specify $\mathscr{H}'_b := \{x \in \mathbf{R}^2 \mid F_b(x) < 0.0952\}$, which is a larger compact set that encloses $\mathscr{H}$ with enough contingency accounting for modeling and numerical error. Similarly, we also consider the case of unbounded unsafe sets, which have the unsafe region defined as follows: $\mathscr{D}_u := \{x \in \mathbf{R}^2 \mid F_u(x) = x_1 + x_2 > 47\}$. Since both the unsafe and the safe sets from which we sample must be compact, we first approximate this unbounded region with a sufficiently large compact set within the operating region $\mathscr{D}_{u'} := \{x \in \mathbf{R}^2 \mid F_u(x) \geq 46 \text{ and } V(x) \leq 368\}$. We then characterize $\mathscr{H}'_u \supset \mathscr{D}_{u'}$ as $\mathscr{H}'_u := \{x \in \mathbf{R}^2 \mid F_u(x) > 45 \text{ and } V(x) \leq 368\}$.

Data points that fall in the set $\mathscr{H}'$ are labeled as "unsafe", while data points outside of this set are labeled as "safe". Both the safe and the unsafe regions are discretized into the same number of data samples, where the samples are labeled with a target output of $B(x) = +1$ if x belongs to the unsafe set, and $B(x) = -1$ if x belongs to the safe set. The inputs to the FNN model are the vector of state measurements x, and the FNN model produces $\hat{B}(x)$ values that classify x as being safe or unsafe.

The following three case studies are examined: varying the number of neurons in the FNN, varying the number of layers in the FNN, and varying the number of training sample size in the FNN. We construct numerous FNN models under each scenario to study the impact of the structure and training of FNN on the generalization error of the resulting model. In all models we construct, the activation functions used in all hidden layers are $tanh(\cdot)$, and the cost functions of Eq. 8.6 are used, where both loss functions are monitored separately during training. The FNN undergoes 500 epochs of training. $L_2 = 0$ and L_1 no longer decreasing for 100 consecutive epochs are the two criteria to trigger early-stopping of training.

Once a FNN-CBF is built, it must be verified that the conditions of Eq. 8.3 must hold for all x in their respective compact sets, by examining whether the strict inequalities of $Eq.$ 8.3 hold for a tightened bound as described in Theorem 8.1. For example, it has been shown that for a 3-hidden-layer FNN with 10 neurons in each layer, $\hat{B}(x) \geq 5.751 \times 10^{-1}$ for all discretized $x \in \hat{\mathscr{H}}$, where $\hat{\mathscr{H}}$ is the unsafe region characterized by the predictions of the FNN model, and $\hat{B}(x) \leq -3.3 \times 10^{-3}$ for all discretized $x \in \mathscr{U}_\rho \backslash \hat{\mathscr{H}}$. It is shown that $\hat{\mathscr{H}}$ is a superset of $\mathscr{D}$, since there

are safe points outside the boundary of $\mathscr{D}$ being misclassified as unsafe by the FNN, but there are no unsafe points being misclassified as safe. Therefore, the conditions of Eq. 8.3a and Eq. 8.3c are proven to hold in a continuous sense. To prove that the condition of Eq. 8.3b also holds, we examine $L_f\hat{B}(x)$ values for all discretized x in the safe set for which $L_g\hat{B}(x) = 0$. This can be seen from the error metrics in Fig. 8.6, where for a 3-hidden-layer model, the errors from $ReLu(L_f\hat{B}(x) + 0.01)$ for all safe x's at which $L_g\hat{B}(x) = 0$ in both training and testing datasets are below 1.18×10^{-6}, which means that $L_f\hat{B} \leq -0.01 - 1.18 \times 10^{-6}$. Thus, the condition of Eq. 8.3b is proven to be true in a continuous sense.

Once the control barrier function is verified, the CLBF $\hat{W}(x)$ is characterized with the following parameters: $c_1 = 0.001$, $c_2 = 100$, $c_3 = 49.38$, $c_4 = 35.21$, $\mu = 5000$, $\rho = 0$, and $\nu = -0.0352$ following the guidelines in Proposition 8.1. The stability and safety region $\mathscr{U}_\rho$ is therefore defined according to Eq. 8.31d.

8.5.3 Analysis on Generalization Performance and Closed-loop Stability and Safety

The generalization performance is assessed via three metrics: the misclassification rate calculated as the ratio of misclassified samples over the total number of samples in the training and testing data sets, the MSE between the predicted and true barrier function output, and loss function calculated from $ReLu(L_f\hat{B}(x) + \tau_I$ for all safe x's in each data set, where $\tau_I = 0.01$ is a positive constant used to ensure the negative definiteness of $L_f\hat{B}(x)$.

Within each case study of FNN models trained using different width, depth, and sample size, both bounded and unbounded unsafe sets are studied. In addition to studying the generalization performance of these FNN models, closed-loop simulations are also ran and compared, and the probability of stability and safety has been investigated. We also run closed-loop simulations with these FNN models and assess its probability of unsafe, unstable, or non-convergent behavior. Unsafe behavior is defined as the closed-loop state entering the unsafe region $\mathscr{D}$ any time during its trajectory from the initial condition to the final state. Unstable behavior is when the closed-loop state exits the stability operating region any time during the simulation period. Non-convergence occurs when the final state at the end of the simulation period is not within the terminal set $\mathscr{U}_{\rho_{min}}$, or when the state exits the terminal set after entering it for the first time. We discretize the operating region evenly to generate a set of x_0 used as initial conditions for closed-loop studied. We run closed-loop simulations starting from 83 different initial conditions in the operating region

$\mathscr{U}_\rho \backslash (\mathscr{U}_{\rho_{min}} \cup \mathscr{B}_\delta(x_e))$ for the case of bounded unsafe sets, and from 74 different initial conditions in the operating region $\mathscr{U}_\rho \backslash \mathscr{U}_{\rho_{min}}$ for the case of unbounded unsafe sets. The probability of each of these three undesirable behaviors is calculated by tabulating the number of occurrences out of the total number of initial conditions ran.

8.5.3.1 Varying number of neurons

In this case study, the FNN is trained with different number of neurons within 1 hidden layer, where the number of neurons (or the width of the FNN) varies from $n_w = 1$ to $n_w = 300$. In the case of the bounded unsafe set, by discretizing the boxes around both the safe and the unsafe regions along each dimension of the state vector by a mesh grid size of 350 by 350, we obtain 20472 unsafe samples and 25695 safe samples. In the case of the unbounded unsafe set, since it is a simpler case where the boundary of safety is linear, we discretize the safe and unsafe regions by 150 by 150, resulting in a dataset of 3021 unsafe and 4198 safe samples respectively. 70% of these samples are used for training, and 30% are used for testing. The generalization performance for FNN models with various number of neurons to address the presence of bounded and unbounded unsafe sets are shown in Fig. 8.1 and Fig. 8.2 respectively.

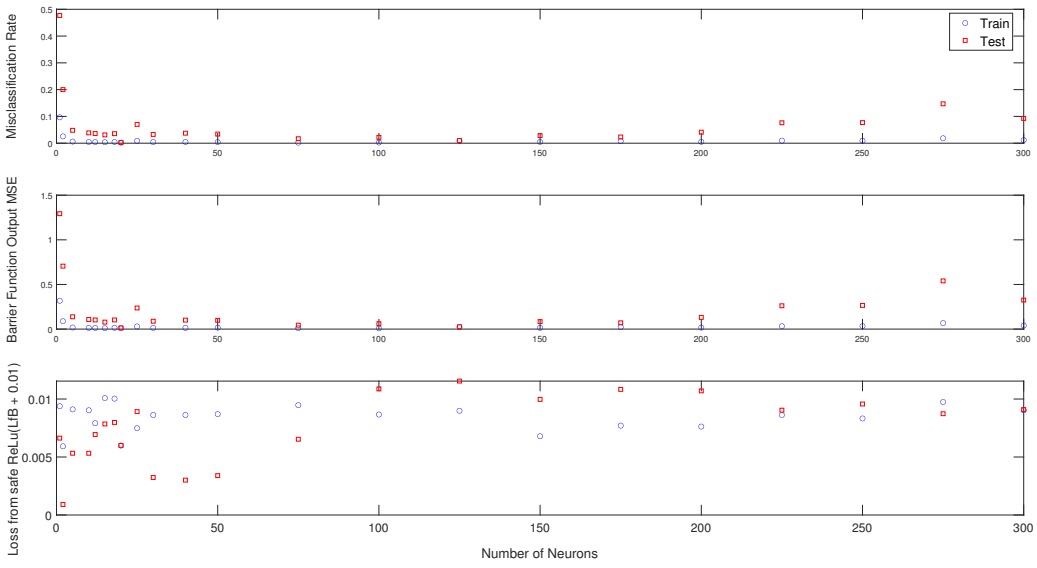


Figure 8.1: Generalization performance for the FNN models for characterizing bounded unsafe regions utilizing various neurons.

In the case of bounded unsafe regions, the drop in misclassification rate and barrier function output MSE are prominently shown as the number of neurons increases from $n_w = 1$ to $n_w = 10$

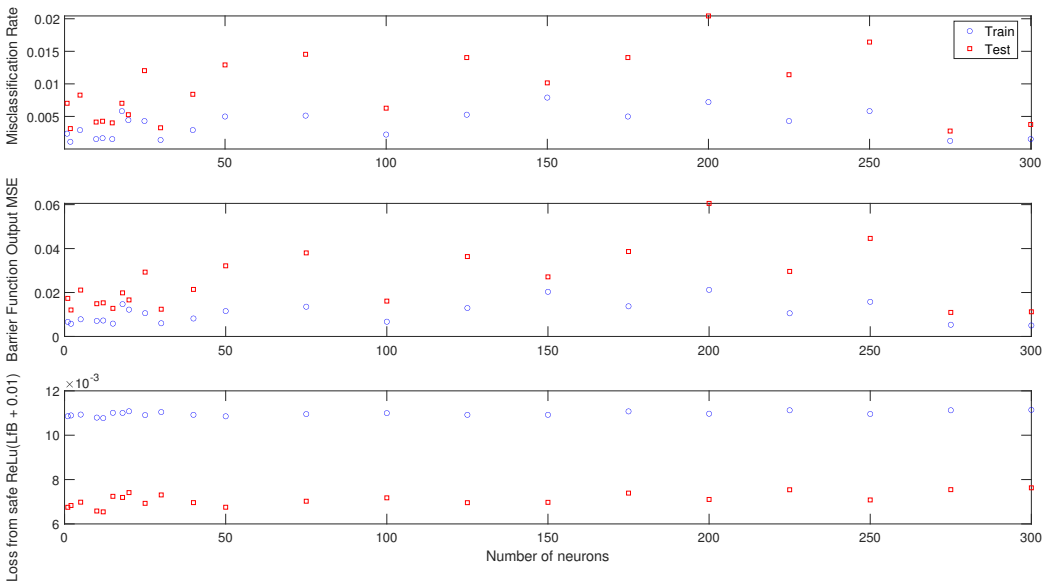


Figure 8.2: Generalization performance for the FNN models for characterizing unbounded unsafe region utilizing various neurons.

for both training and testing datasets. The misclassification rate and output MSE for the training dataset stay consistently low for $n_w \geq 10$, where its misclassification rate is maintained below 0.018 and MSE output is maintained below 0.067 (this high point occurs at $n_w = 300$). Misclassification rate and output MSE in the testing dataset are consistently higher than the training dataset for all variations of n_w, which is expected as there exists a gap between the expected error and the empirical error as shown in the generalization error analysis in this work. For the testing dataset, misclassification rate stays below 0.04 and output MSE stays below 0.11 for $n_w \in [10, 200]$ except for the one-off case at $n_w = 25$, which has a testing data misclassification rate of 7×10^{-2} and an output MSE of 2.3×10^{-1}. Sometimes one-off cases of FNN models occur where their resulting errors are higher than other FNN models of similar structure due to the stochastic nature of FNN training and prediction. For $n_w > 200$, it is seen that the testing errors in misclassification rate and output MSE increase as n_w increases while the training errors for these two metrics stay consistently low. This is expected as the FNN model is essentially over-parametrized by too many number of neurons, and while this improves the model's ability to learn and fit existing data, it becomes overfitted and therefore producing increasingly larger errors when applied to other data samples that do not exist in the training set. The third error metric is the loss calculated from $ReLu(L_f\hat{B}(x) + 0.01)$ for all safe x that satisfy $L_g\hat{B}(x) = 0$ in both training and testing datasets. Although there are no obvious trends in the relation between this error and the number of neurons,

it is observed that the error in the training set stays below 1.008×10^{-2}, while the highest error in the testing set is at 1.154×10^{-2}.

In the case of unbounded unsafe sets, all three error metrics achieve relatively low values compared to the case of bounded unsafe sets due to the less challenging nature of unbounded unsafe sets similar to a linear boundary. There are no obvious trends of errors increasing or decreasing as the number of neurons increase because the errors are already maintained at a low level (misclassification rate is kept under 2×10^{-2}, output MSE is kept under 6×10^{-2}, and $L_f \hat{B}(x)$ is kept under 1.107×10^{-2}, accounting for both training and testing errors). However, we do observe that the gap between training and testing error generally increases as n_w increases beyond 50. This may be due to model overfitting where the model is again parametrized with too many neurons.

The probabilities of unsafe, unstable, and non-convergent closed-loop behavior under the control of FNN-based CLBF-MPC built using FNN models with varying number of neurons in the case of bounded unsafe set are shown in Fig. 8.3. In addition, the figure also shows the probability of any of these three behaviors occurring. It is demonstrated that the probability decreases drastically for $n_w > 2$ and reaches its minimum at $n_w = 15$. It is also noted that the instances of non-convergence also increases for $n_w \geq 250$, which is consistent with the trend of testing error and generalization error gap increasing for overfitted models with $n_w > 200$.

To better illustrate how FNN models trained with insufficient number of neurons may impact the closed-loop performance of the FNN-CLBF-MPC, Fig. 8.4 compares two state trajectories starting from the same initial condition, one under an FNN barrier function trained with 15 neurons (blue), and one under an FNN barrier function trained with 2 neurons (red). The FNN barrier function trained with 2 neurons, which has much higher errors and probabilities of instability and violation of safety, falsely identifies all states within this time-series trajectory as "unsafe" (labeled by diamond markers), including the initial condition x_0. Therefore, it is shown to produce a closed-loop trajectory that fails to navigate the state around the unsafe region. The closed-loop state enters the unsafe region and struggles to leave within the simulation period. This shows an instance of unsafe and non-convergent behavior amongst the 83 runs of closed-loop simulations starting from different initial conditions. On the other hand, with an FNN barrier function trained with 15 neurons, starting from the same initial condition, the closed-loop state of the CSTR process is able to converge to the terminal set within the simulation period and avoid entering the unsafe region $\mathscr{H}'$. All states within this trajectory are correctly classified as "safe", which is labeled by the circle markers.

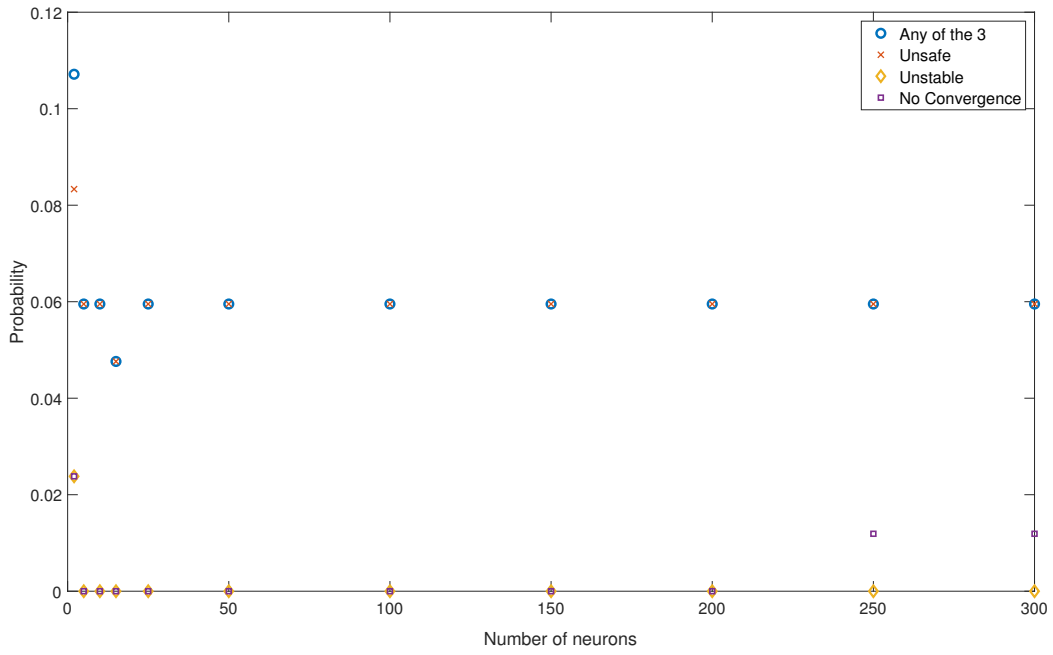


Figure 8.3: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying neurons in the case of bounded unsafe region.

The probabilities of unsafe, unstable, and non-convergent closed-loop behavior in the presence of unbounded unsafe regions are shown in Fig. 8.5. Since all models have low misclassification rate and low barrier function output MSE, the number of occurrences of such unsafe, unstable, or non-convergent trajectories is zero for various FNN models trained with different number of neurons. In other words, all of the closed-loop simulation runs starting from 74 different initial conditions are able to converge to the terminal set within the simulation period while not entering the unsafe region and not exiting the stability region.

8.5.4 Varying number of layers

The relation between model depths and generalization performance are also studied, where FNN models with various number of layers from $n_l = 1$ to $n_l = 20$ are constructed with 10 neurons within each hidden layer. The same data generation and sampling method is used as in the case study of varying number of neurons. The generalization performance for FNN models with various number of layers for both cases of bounded and unbounded unsafe sets are shown in Fig. 8.6 and Fig. 8.7 respectively.

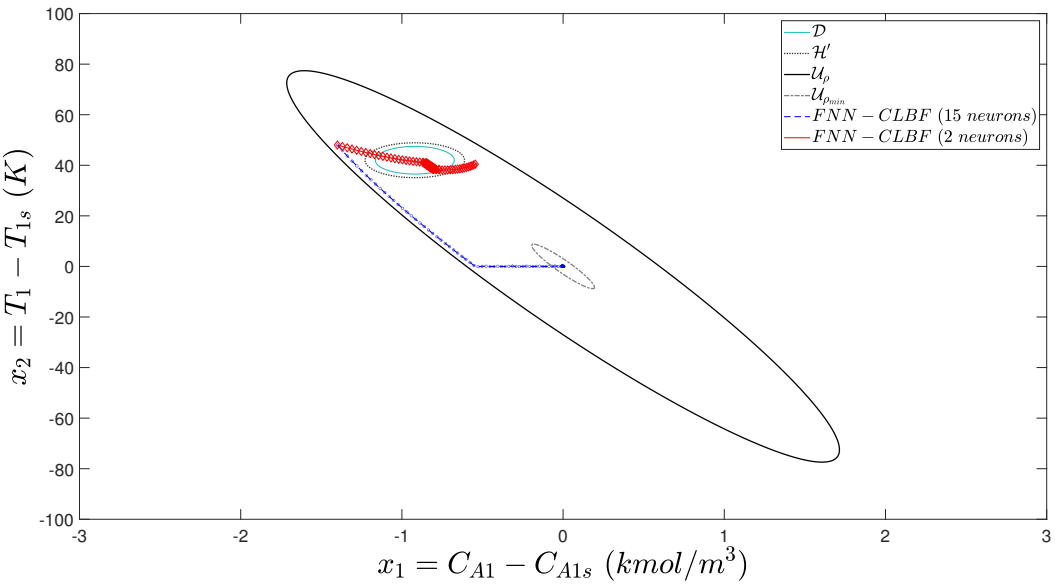


Figure 8.4: Closed-loop state trajectories under CLBF-MPC with FNN-based barrier function trained with 15 neurons (blue) vs. 2 neurons (red), where states classified as safe by each FNN model are labelled by circle markers, and states classified as unsafe by each FNN model are labelled by diamond markers.

In the case of bounded unsafe sets, the misclassification rate and the output MSE for the testing dataset are maintained below 5.6×10^{-2} and 1.79×10^{-1} respectively for layers $n_l = 1$ to $n_l = 12$, and the testing errors are shown to be higher than the training errors for all layers. For layers $n_l \geq 15$, the generalization error gap between testing and training error drastically increases, which can be attributed to the model being overfitted, thus unable to generalize to new data as effectively. It is also observed that both training and testing error increase as the number of layer increases for $n_l \geq 15$. This is a common phenomenon that has been seen in neural networks with increasing depth; some possible explanations include: the network may be not able to find an appropriate mapping between two consecutive layers and becomes hard to optimize, or higher-level layers may lose access to important lower-level layer features. However, this remains a topic that is continuously studied by researchers. For the third error metric which assesses the negative definiteness of $L_f\hat{B}$, all models produced an average $ReLu(L_f\hat{B}(x) + 0.01)$ of less than 0.01 in both training and testing datasets, and this error is maintained under 2.8×10^{-4} in both training and testing sets for models with $n_l \geq 2$. This shows that an FNN of at least 2 layers is needed.

In the case of unbounded unsafe sets, the resulting training and testing misclassification rate and output MSE are again sporadic because their values are already low for all layers. The highest misclassification rate and output MSE are 1.37×10^{-2} and 3.65×10^{-2} respectively, which are

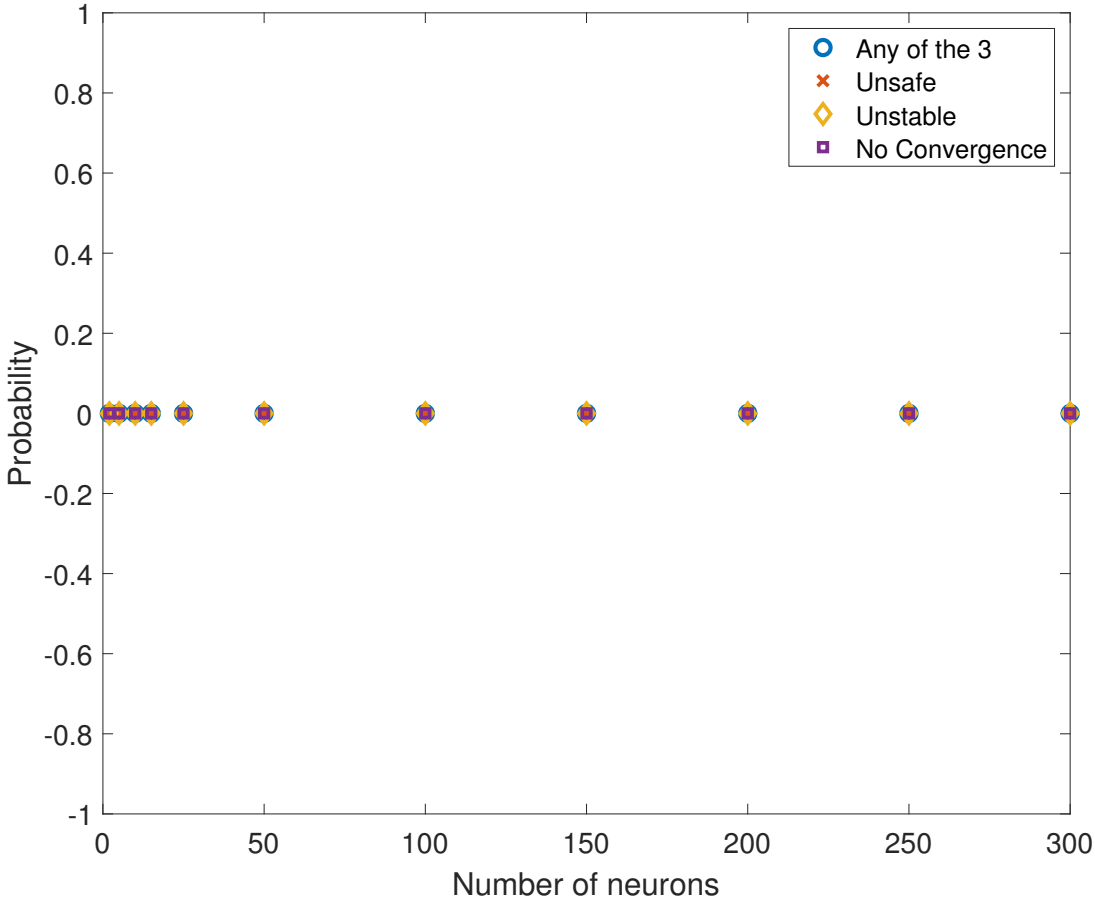


Figure 8.5: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying neurons in the case of unbounded unsafe regions.

obtained at $n_l = 5$. For the first two error metrics, the testing dataset consistently yields a higher error than the training dataset, which agrees with the theoretical development of Section 8.3. The training loss from $ReLu(L_f\hat{B} + 0.01)$ is oddly higher than the testing losses for $n_l = 1,2,3$. The highest loss of this error metric is 0.0109 for training and 6.8×10^{-3} for testing at $n_l = 2$. For $n_l \geq 3$, $L_f\hat{B}(x) < 0, x \in \{\mathscr{U}_\rho | L_g\hat{B}(x) = 0\}$ holds for both training and testing datasets. For this particular study, Fig. 8.7 shows that it is best to choose an FNN built with 4 layers.

Closed-loop probability studies are also conducted for both bounded and unbounded unsafe sets. For bounded unsafe sets, the probability of non-convergent behavior starts increasing for $n_l \geq 12$, and the probability of unsafe, unstable behavior starts increasing for $n_l \geq 15$. The probabilities are plotted against varying FNN depth in Fig. 8.8. This is consistent with the generalization error

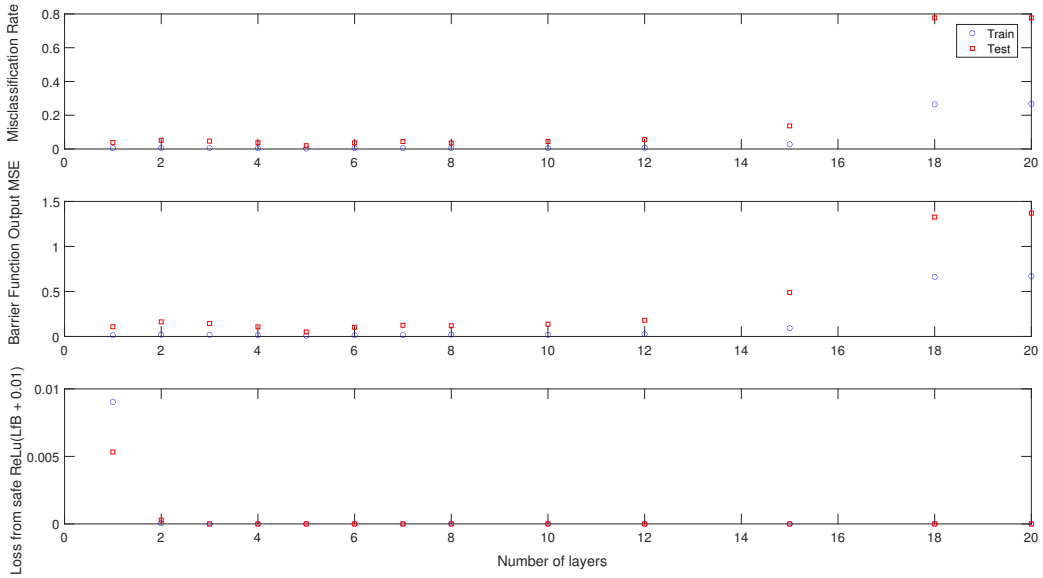


Figure 8.6: Generalization performance for the FNN models for characterizing bounded unsafe regions utilizing various layers.

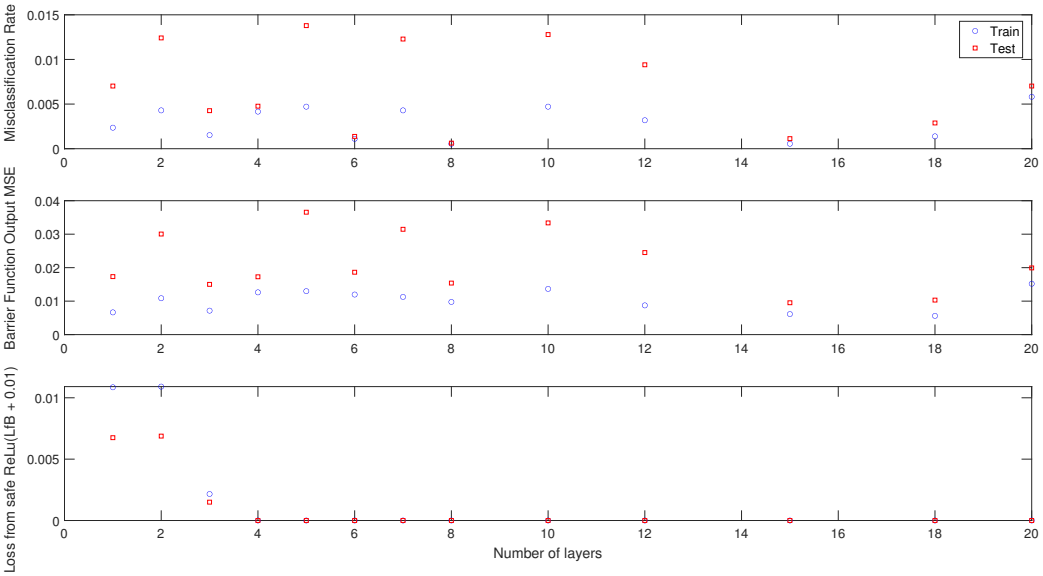


Figure 8.7: Generalization performance for the FNN models for characterizing unbounded unsafe region utilizing various layers.

performance, where the model becomes overfitted as the number of layer increases beyond $n_l \geq 15$, and the training error, the testing error, as well as the generalization error gap all increase. The larger the generalization error gap, the less likely that closed-loop stability and safety can be guaranteed, thus the occurrences of unstable, unsafe, and non-convergent behavior increase.

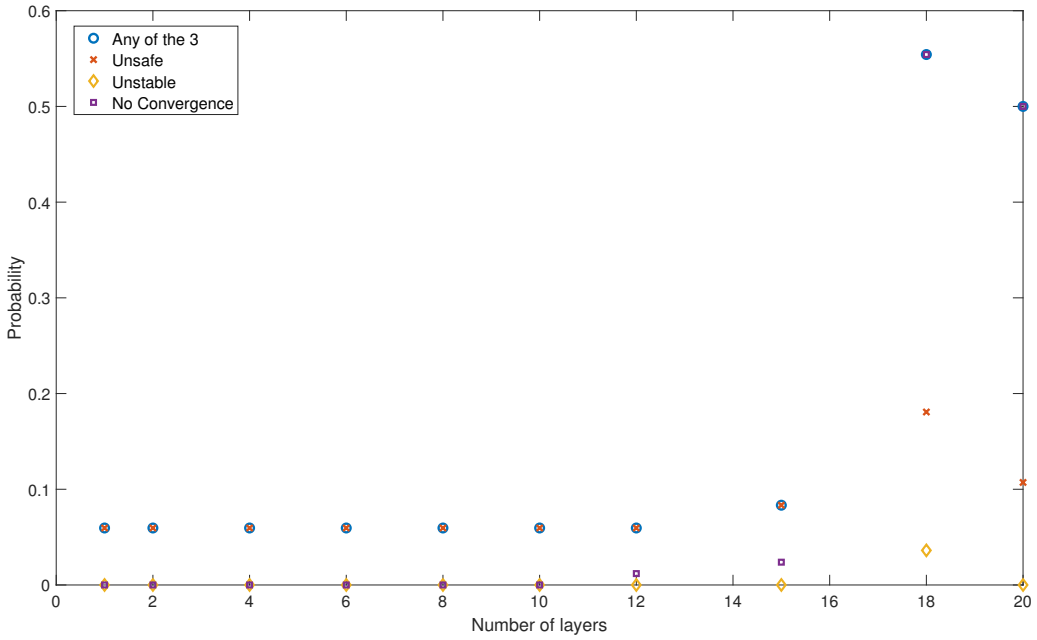


Figure 8.8: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying layers in the case of bounded unsafe region.

We further demonstrate the difference in closed-loop performance between two models trained with different number of layers for systems with bounded unsafe sets in state space. Fig. 8.9 shows two state profiles under the FNN-CLBF-MPC, one of them has an FNN barrier function trained with 2 layers (blue), and the other one has an FNN barrier function trained with 18 layers (red). Along the red-colored state trajectory, all state values are falsely identified as "unsafe" by the 18-layer FNN barrier function, causing the closed-loop state to move very slowly, eventually into the unsafe region and unable to escape. The blue-colored trajectory starts from the same initial conditon, and is driven inside the terminal set while avoiding the unsafe set successfully. Along this trajectory, only one state at $x^T = [-1.2537, 41.3475]$, which is outside the unsafe region, is being falsely identified as "unsafe". This is because the predicted unsafe region $\hat{\mathscr{H}}$ characterized by the FNN barrier function $\hat{B}(x)$ constructed using 2 layers turns out to be a superset of $\mathscr{H}'$, which allows the MPC to act preemptively before the state actually enters $\mathscr{H}'$.

Similarly, the probability of unsafe, unstable, and non-convergent behaviors for the case of unbounded unsafe regions are shown in Fig. 8.10. Due to the consistently low modeling error, the probability of such behavior is zero across all variations of FNN depth.

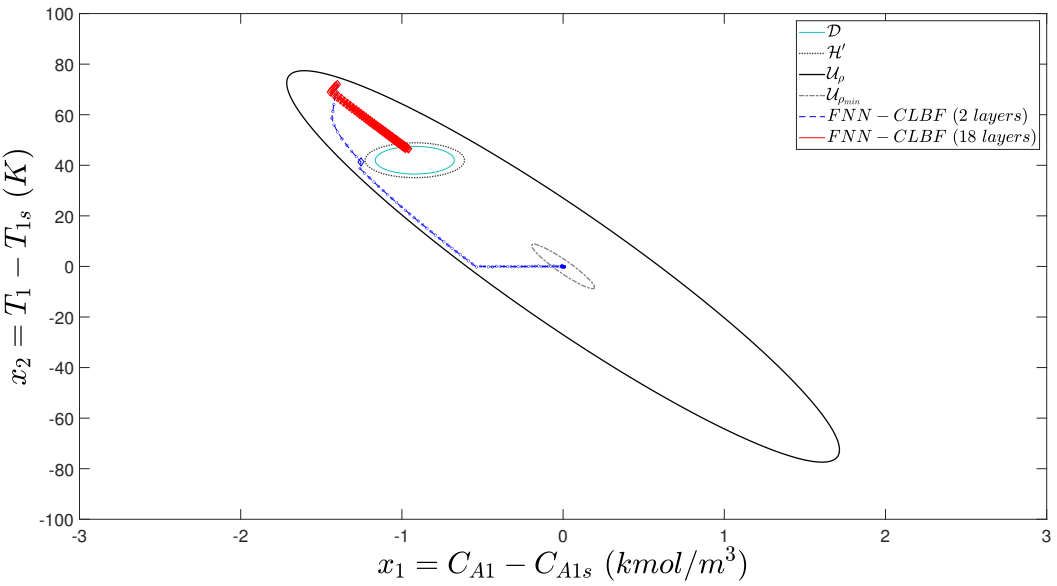


Figure 8.9: Closed-loop state trajectories under CLBF-MPC with FNN-based barrier function trained with 2 layers (blue) vs. 18 layers (red), where states classified as safe by each FNN model are labelled by circle markers, and states classified as unsafe by each FNN model are labelled by diamond markers.

8.5.5 Varying training data sample size

Lastly, the number of training data sample size is varied to examine its impact on the generalization error and closed-loop performance. The same data generation method is applied with the exception of varying the discretization grid size along each dimension of x from $n_d = 10, 20, 30, ..., 450, 500$. The resulting datasets range from having a total sample size of $m = 38$ to $m = 94251$. The same neural network structure is used, consisting of 2 hidden layers of 10 neurons each. Fig. 8.11-Fig. 8.12 illustrate the three error metrics in training and testing datasets for the bounded and unbounded unsafe sets respectively.

In the case of bounded unsafe regions, it is seen that for training data sample size between $m = 38$ to $m = 3775$, both training and testing sets produce high misclassification rate and the output MSE. The testing errors are higher than the testing errors with a generalization error gap, and the magnitude of these errors as well as the generalization error gap between training and testing sets decrease as the sample size increases. This is aligned with theoretical derivations as larger sample size results in improved model accuracy and reduced generalization error. The generalization gap, which captures the difference between expected error (testing dataset) and empirical error (training dataset), is roughly proportional to $\frac{1}{\sqrt{m}}$ as indicated by Eq. 8.19. This

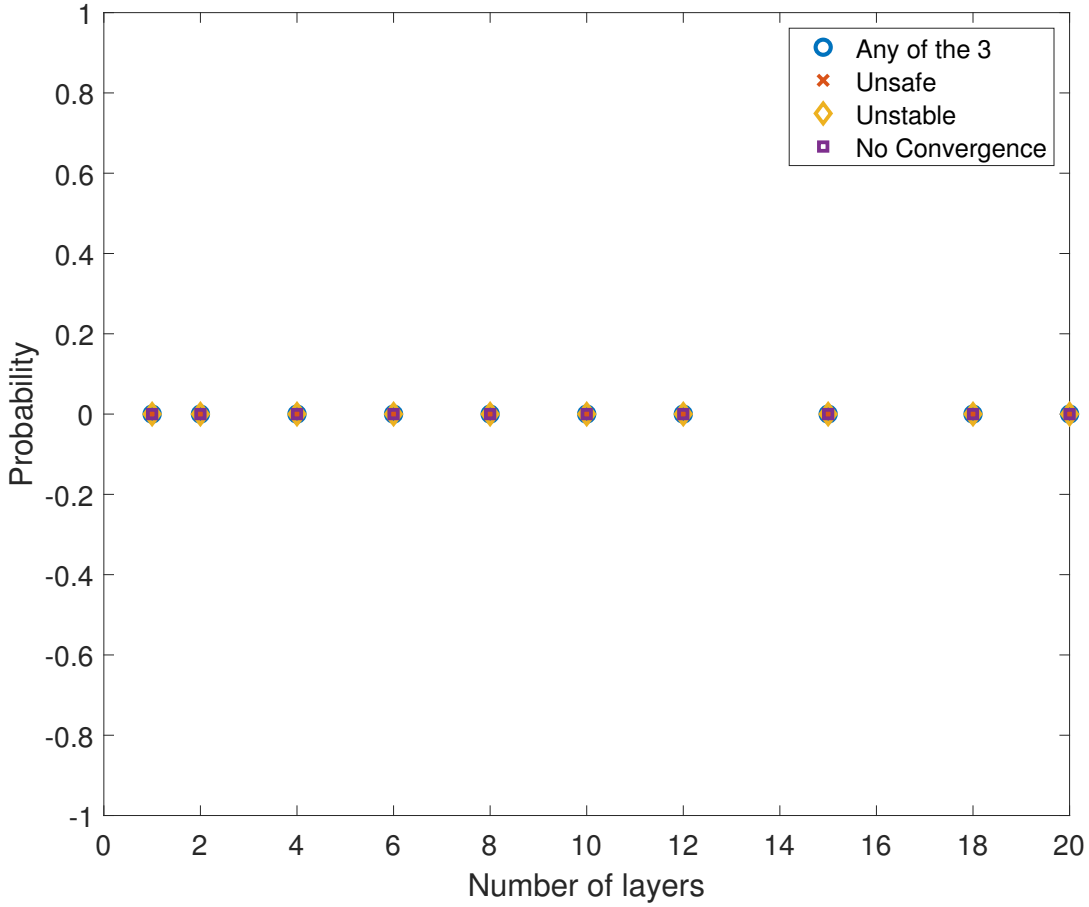


Figure 8.10: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying layers in the case of unbounded unsafe regions.

is consistent with the trend observed here where the decrease is drastic when m is small, and reaches a plateau as m increases to larger values. For data sample size $m \geq 8499$, both training and testing errors in the first two metrics stay consistently low below 1.8×10^{-2} and 5.6×10^{-2} for misclassification and MSE respectively, and no significant improvement is seen beyond $m \geq 8499$. For the losses calculated from $ReLu(L_f\hat{B}(x) + 0.01)$, it is observed that both training and testing errors are able to achieve extremely low values for $m = 38$ to $m = 3775$ where the misclassification rate and MSE are high. This may be because the data samples are too few for the FNN to learn the underlying relation between input and output, and therefore it fails to minimize L_1 and only stresses on satisfying L_2. The maximum $ReLu(L_f\hat{B}(x) + 0.01)$ for all models is 7.79×10^{-4} and therefore the expected $L_f\hat{B}(x)$ stays below 0 for all models.

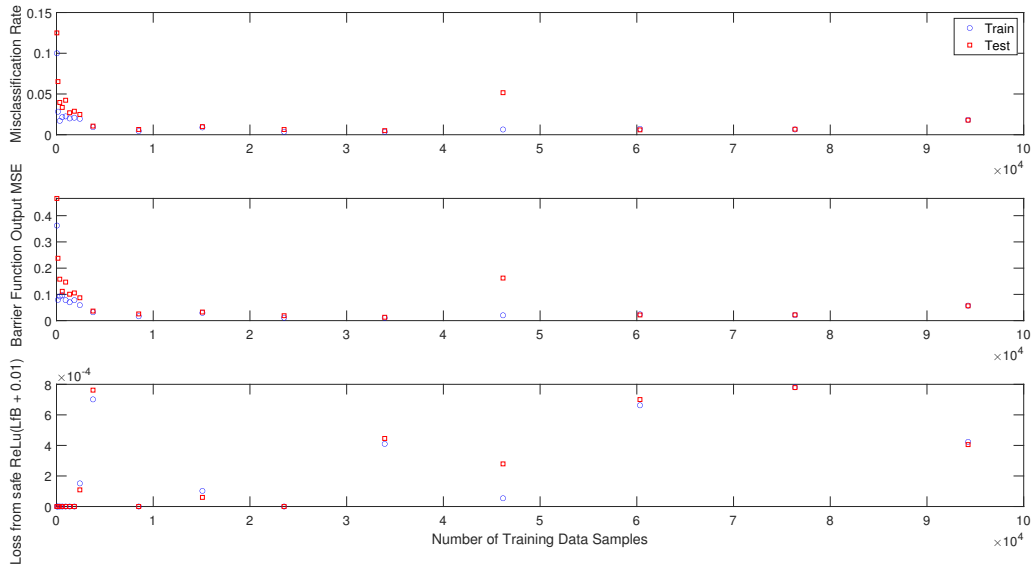


Figure 8.11: Generalization performance for the FNN models for characterizing bounded unsafe regions utilizing various data sample size.

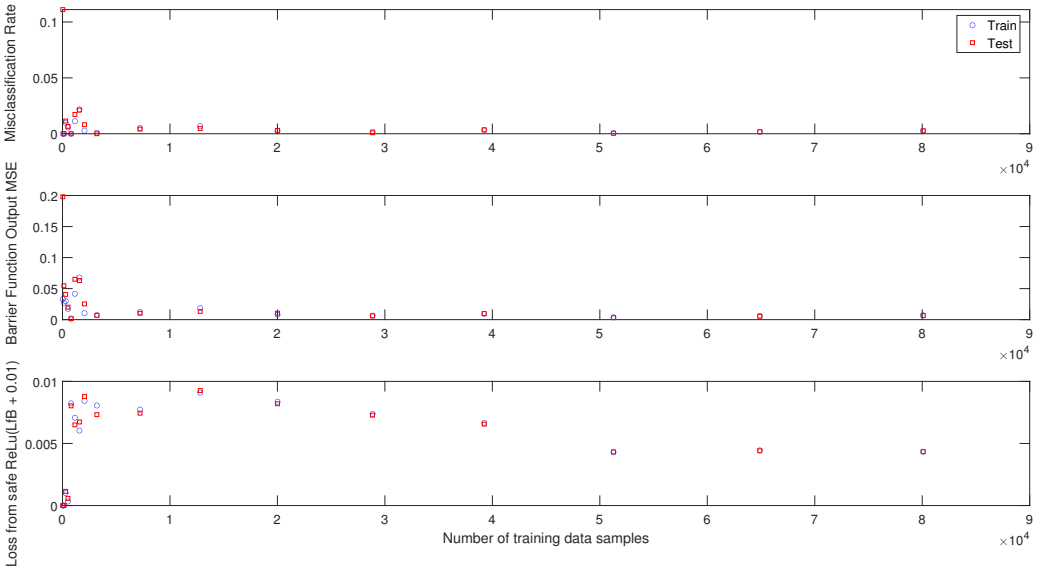


Figure 8.12: Generalization performance for the FNN models for characterizing unbounded unsafe region utilizing various data sample size.

In the case of unbounded unsafe regions, it is similarly seen in Fig. 8.12 that the testing error and generalization error gap for misclassification rate and output MSE at the smallest sample size $m = 38$ is drastically higher than the other FNN models trained with larger training sample size, and they reach a low, stable level after $m \geq 3199$. For the loss term of $ReLu(L_f\hat{B}(x) + 0.01)$, all

losses stay below 9.27×10^{-3}, which means that the expected $L_f\hat{B}(x) < 0$ for all models.

We also simulate closed-loop runs starting from various initial conditions within the operating region to assess probabilities of unstable, unsafe, and non-convergent behaviors. It is shown in Fig. 8.13 that the probabilities of unsafe and non-convergent instances drop to 0 for $m \geq 3775$, and the probability of unsafe instances is also in general lower when the sample size is larger. This is better demonstrated in the comparison of two closed-loop trajectories shown in Fig. 8.14 where the closed-loop state under an FNN model trained with 152 samples (red) and under an FNN model trained with 8499 samples (blue) are plotted together. The FNN model trained with 152 samples incorrectly classifies the initial condition as well as many states around the unsafe region as "unsafe" (diamond markers), and the closed-loop state under this FNN-CLBF-MPC enters the unsafe region and eventually traverses across the unsafe region, exiting on the other side. The closed-loop state continues to migrate towards terminal set, where eventually the FNN model correctly identifies the state as being "safe" (labeled by circle markers), and the closed-loop state ultimately is driven inside the terminal set. The closed-loop state along the trajectory controlled by the FNN-CLBF-MPC trained with 8499 samples are all correctly classified as "safe", and the MPC is able to quickly drive and maintain the state inside the terminal set in a stable and safe manner. Closed-loop simulations are also conducted starting from various initial conditions inside the operating region for the unbounded unsafe sets, and the probabilities of any of these undesirable instances occurring are zero for all models, as shown in Fig. 8.15.

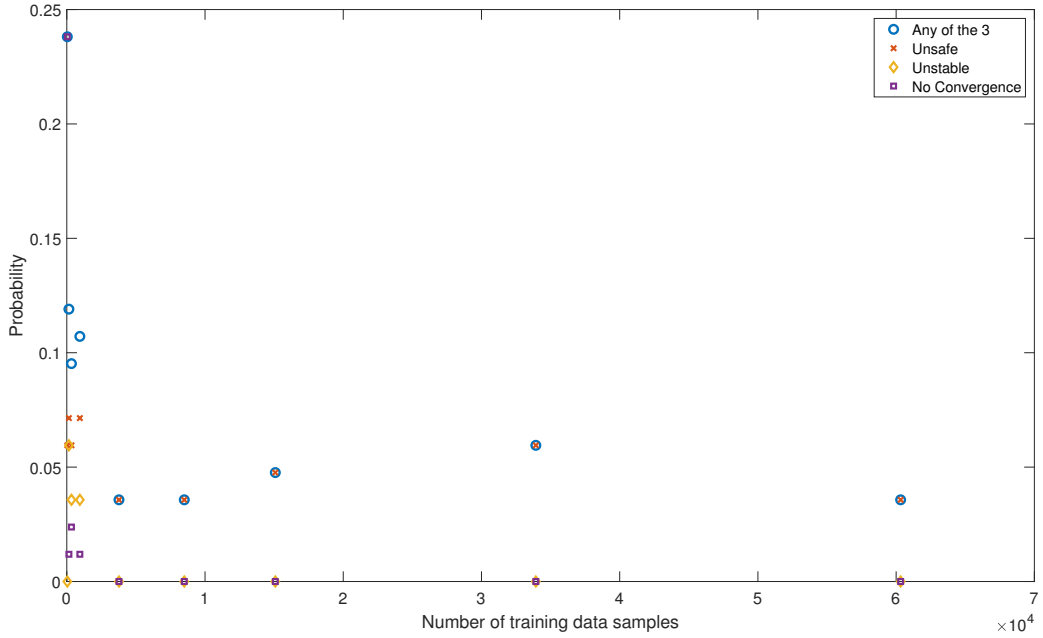


Figure 8.13: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying data sample size in the case of bounded unsafe region.

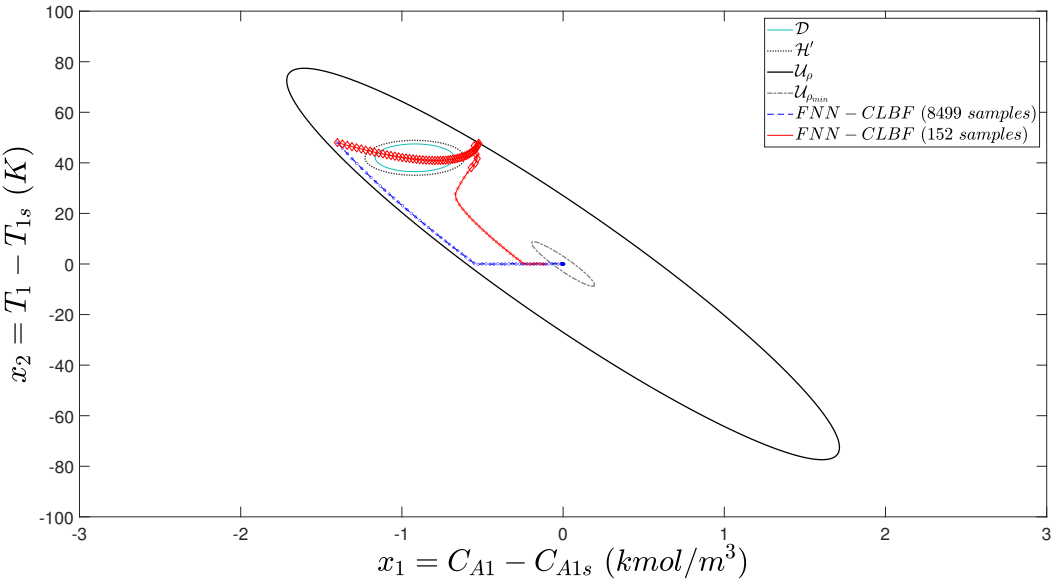


Figure 8.14: Closed-loop state trajectories under CLBF-MPC with FNN-based barrier function trained with 8499 data samples (blue) vs. 152 data samples (red), where states classified as safe by each FNN model are labelled by circle markers, and states classified as unsafe by each FNN model are labelled by diamond markers.

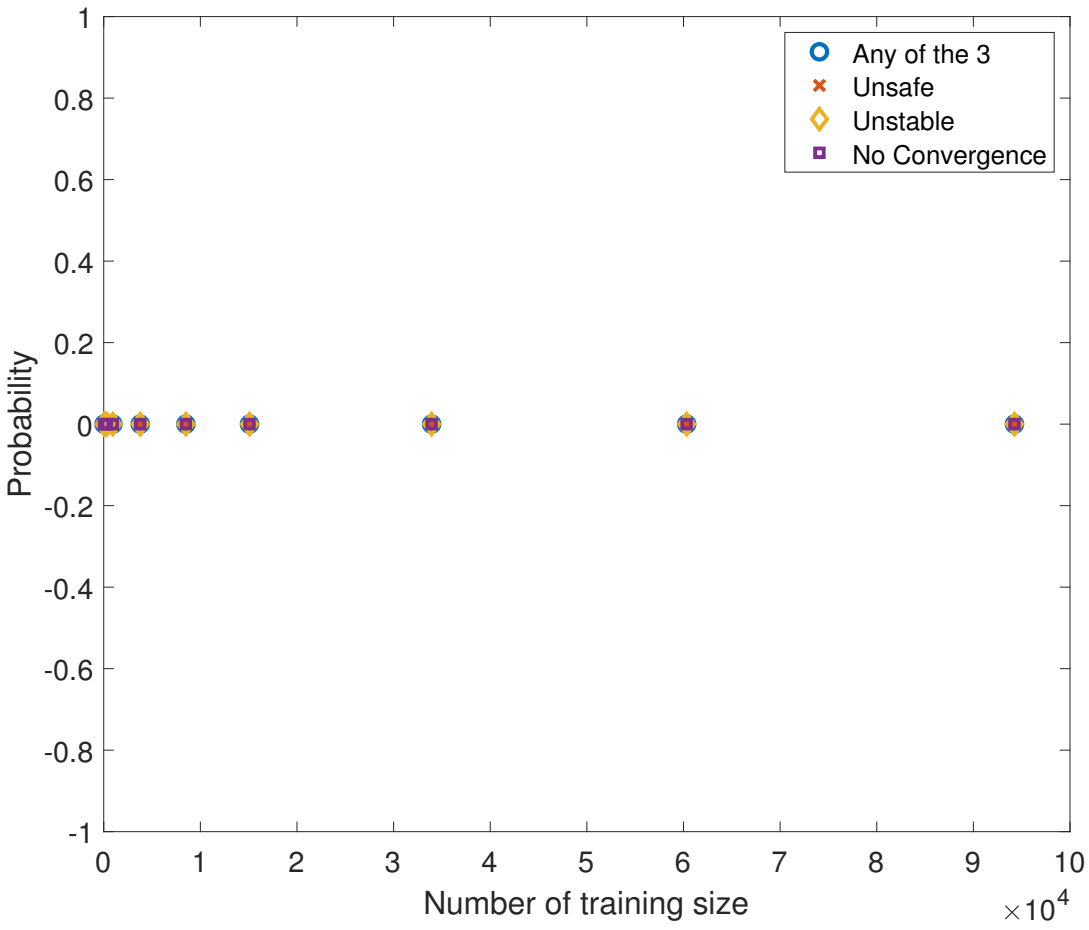


Figure 8.15: Probabilities of unsafe, unstable, and non-convergent behavior under closed-loop control of the FNN-based CLBF-MPC for FNN models trained with varying data sample size in the case of unbounded unsafe regions.

Remark 8.3. *As shown in Theorem 8.2, the generalization performance of the FNN model depends on a number of factors, including the sample size m, the network weight matrix bounds B_W, the bound on the possible values of state vector as inputs to the FNN B_X, the depth of the neural network d, the output dimension d_y and the input dimension d_x. In this study, we have demonstrated case study results on the impact of neural network hypothesis class complexity (depth and width) and the training sample size m on the overall generalization and closed-loop performance. As an extension to this study, one may also study the impact of B_W, B_X, which have been investigated in [126], or d_x, by adjusting the number of input features if possible.*

Chapter 9

Conclusion

This dissertation discusses a number of different designs of machine-learning-based MPC systems to improve cybersecurity and safe operation of nonlinear chemical processes. Firstly, machine-learning-based detection schemes are proposed and integrated with various control systems such as a two-tier LMPC system and EMPC system with attack-resilient operation strategies. The impact of different types of cyber-attacks on distributed and decentralized MPC systems are also explored, and a machine-learning-based detector is implemented to identify the presence or location of the attack. Next, data-driven models are developed using machine learning methods for nonlinear dynamic processes consisting of multiple sub-processes, and they are incorporated in distributed and decentralized MPC systems with closed-loop stability analyses. A Control Lyapunov-Barrier Function (CLBF) is also utilized in the design of a distributed MPC system to demonstrate guaranteed closed-loop stability and operational safety properties for all sub-systems in the nonlinear process. Furthermore, machine learning techniques are used to characterize a CLBF with verified conditions, and control laws designed based on this CLBF are able to ensure closed-loop stability and safety for a nonlinear process. Lastly, the generalization error of this machine-learning-based CLBF is assessed using statistical learning theories, and the probabilistic closed-loop performance of the CLBF-MPC is evaluated.

In Chapter 2, a cyber-secure control architecture for nonlinear chemical processes incorporating secure lower-tier explicit feedback controllers and an upper-tier model predictive controller was proposed. On top of the stabilizing lower-tier controllers, the upper-tier LMPC contributed to better closed-loop performance by using networked sensor measurements which may be vulnerable to cyber-attacks. A neural-network-based detector is integrated with the two-tier control architecture such that the control system can be reconfigured to stabilize the process at the

original steady-state upon detection of a cyber-attack. Neural-network-based detection algorithms were developed and have proven their success during online implementation in detecting the presence of cyber-attacks when various types of cyber-attacks were applied on multiple sensors. Four feed-forward neural networks were trained and tested under nominal and noisy operating conditions, all of which achieved a detection accuracy of over 91%. Through the application of the proposed detection and mitigation methods on a multivariable process, this work demonstrated the effectiveness of machine-learning-based methods in developing algorithms used for cyber-attack diagnosis and cyber-defense, as well as the robustness of the proposed two-tier control architecture in maintaining cyber-security.

In Chapter 3, the secure operation of nonlinear chemical processes under economic model predictive control was presented via the design of a secure operating region, resilient control strategies, and a neural-network-based cyber-attack detector. Considering a general class of nonlinear systems, a resilient Lyapunov-based Economic Model Predictive Controller with combined closed-loop and open-loop control action implementation was developed at the cost of reduced total economic gain. Through simulating a continuously stirred tank reactor process, it was demonstrated that the proposed control strategy was effective in maintaining process stability against particular types of malicious cyber-attacks, namely min-max and surge attacks, while achieving comparable economic performance compared to nominal operation under no attacks. Two neural-network-based cyber-attack detectors were constructed to detect the presence or distinguish the type of a cyber-attack, and the time-varying trajectory of a nonlinear function of sensor measurements were used as the input variables for the detection algorithm. The detector was able to provide a diagnosis at the end of each LEMPC operation period, and simulation results demonstrated that min-max, surge, and geometric attacks could be successfully detected.

As cyber-attacks pose critical risks to large-scale industrial plants, the effects of some standard types of sensor cyber-attacks on the operation of nonlinear processes under centralized, decentralized and distributed model predictive control systems were analyzed in Chapter 4. Decentralized control systems have been demonstrated to be more robust than distributed and centralized control systems against sensor cyber-attacks. Using sensor data, a feed-forward neural network model was trained with adequate accuracy to detect the presence of cyber-attacks and was implemented online with a moving horizon detection window. Simulation studies were carried out on a nonlinear chemical process example to demonstrate the effectiveness of the combined architecture of the decentralized control system and the machine-learning-based detector in ensuring the closed-loop stability and the cyber-security of the overall closed-loop process.

In Chapter 5.1, we first presented the design of distributed model predictive control systems for nonlinear processes using a machine-learning-based model, which was used as the prediction model to capture nonlinear dynamic behavior. Closed-loop stability and performance properties were analyzed for the sequential and iterative distributed model predictive control systems. Extensive open-loop data within the stability region of the nonlinear process characterized by Lyapunov-based control methods were collected to train LSTM network models with a sufficiently small modeling error with respect to the actual nonlinear process model. It was shown that both sequential distributed LMPC system and iterative distributed LMPC system using the LSTM network model were able to achieve efficient real-time computation, and ensure closed-loop state boundedness and convergence to the origin. Working with a nonlinear chemical process network example, the distributed LMPC systems using the LSTM network model were able to obtain similar closed-loop performance as the distributed LMPC systems using the first-principles model, as well as reduced computation time when compared to the closed-loop results of the centralized LMPC system using the LSTM network model. Following this, in Chapter 5.2 we also presented the design of decentralized model predictive control systems for nonlinear processes using machine-learning models, which were developed separately to capture the nonlinear dynamic behavior of independent subsystems. The nonlinear process was divided into multiple subsystems, and the closed-loop stability and performance properties of the proposed decentralized framework with respect to the overall process were analyzed. Using Lypuanov-based control methods to characterize the stability region of the nonlinear process, extensive open-loop data were collected to train one LSTM network model for each subsystem with a sufficiently small modeling error. It was shown that the decentralized LMPCs using LSTM models were able to ensure closed-loop state boundedness and convergence to a small neighborhood around the origin for all process states inside the stability region while achieving efficient real-time computation. Using a nonlinear chemical process example, the decentralized LMPCs using LSTM models were able to obtain similar closed-loop performance as the same decentralized LMPCs using first-principles models. Furthermore, they were able to significantly reduce the computation time while ensuring similar closed-loop results when compared to the case of the centralized LMPC using an LSTM model for the overall process.

In Chapter 6, we have demonstrated that nonlinear systems subject to input constraints could be stabilized by a CLBF-MPC while not entering unsafe regions where the barrier function was constructed using an FNN model and the predictive model within MPC was obtained using an RNN model. A Control Barrier Function was first characterized by building an FNN model with

unique structures and properties, and was then trained and validated using discretized data collected from a conservative rendition of unsafe and safe regions. Given sufficiently small bounded modeling errors with the two NN models, the proposed CLBF-MPC was able to meet its control objective of ensuring simultaneous stability and safety for all initial conditions within a subset of the stability region under sample-and-hold control action implementation. The effectiveness of the machine-learning-based CLBF-MPC was demonstrated using a nonlinear chemical process example with a bounded unsafe set.

We have shown theoretical analysis that nonlinear systems with input constraints and consisting of multiple subsystems can be stabilized by a CLBF-DMPC while not crossing the boundary of unsafe regions in Chapter 7. A constrained CLBF is designed to characterize a stability region that has no intersection with the unsafe regions, and subsequently used to design CLBF-based explicit control laws for each subsystem. A CLBF-DMPC, which can be calculated either sequentially or iteratively, is presented and proven to have recursive feasibility as well as stability and safety properties with considerations of sample-and-hold control action implementation and presence of bounded disturbances. A modified DMPC structure is also studied and simulated for particular considerations of nonlinear subsystems. Lastly, the effectiveness of the proposed CLBF-DMPC system is demonstrated on a two-CSTR process with both bounded and unbounded unsafe sets.

Lastly, in Chapter 8, a machine-learning-based Control Lyapunov-Barrier Function is used to design model predictive controllers for nonlinear systems with the presence of bounded and unbounded unsafe sets. Specifically, an FNN model is used to construct the Control Barrier Function, for which the generalization error bound is analyzed using the Rademacher complexity method from statistical machine learning theory. Subsequently, probabilistic stability and safety is established for CLBF-based control laws designed using this FNN-CBF, which is then extended to the sample-and-hold implementation of an FNN-CLBF-MPC. We demonstrate the impact that structural complexity and sample size of the FNN model have on the generalization performance, as well as the probabilities of closed-loop stability and safety in the cases of bounded and unbounded unsafe sets through a chemical reactor example.

Bibliography

[1] S. Agrawal and J. Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713, 2015.

[2] A. Ahlén, J. Akerberg, M. Eriksson, A. J. Isaksson, T. Iwaki, K. H. Johansson, S. Knorn, T. Lindh, and H. Sandberg. Toward wireless control in industrial process automation: A case study at a paper mill. *IEEE Control Systems Magazine*, 39:36–57, 2019.

[3] R. K. Al Seyab and Y. Cao. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *Journal of Process Control*, 18:568–581, 2008.

[4] F. Albalawi, H. Durand, and P. D. Christofides. Process operational safety using model predictive control based on a process safeness index. *Computers & Chemical Engineering*, 104:76–88, 2017.

[5] A. Alessio, D. Barcelli, and A. Bemporad. Decentralized model predictive control of dynamically coupled linear systems. *Journal of Process Control*, 21:705–714, 2011.

[6] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen. Cyber security of water scada systems—part i: Analysis and experimentation of stealthy deception attacks. *IEEE Transactions on Control Systems Technology*, 21:1963–1970, 2012.

[7] R. Amrit, J. B. Rawlings, and D. Angeli. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control*, 35:178–186, 2011.

[8] M. R. Asghar, Q. Hu, and S. Zeadally. Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks*, 165:106946, 2019.

[9] Y. Ashibani and Q. H. Mahmoud. Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, 68:81–97, 2017.

[10] L. Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32:87–98, 2008.

[11] A. Bemporad, M. Heemels, and M. Johansson. *Networked control systems*, volume 406. Springer, 2010.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[13] R. Bobiti and M. Lazar. A sampling approach to finding Lyapunov functions for nonlinear discrete-time systems. In *In Proceedings of the 2016 European Control Conference (ECC)*, pages 561–566, Aalborg, Denmark, 2016.

[14] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18:1153–1176, 2015.

[15] F. Burden and D. Winkler. Bayesian regularization of neural networks. In *Artificial Neural Networks*, pages 23–42. Springer, New York, NY, 2008.

[16] A. A. Cárdenas, S. Amin, Z. S. Lin, Y. L. Huang, C. Y. Huang, and S. Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366, Hong Kong, China, 2011.

[17] B. Carter, S. Adams, G. Bakirtzis, T. Sherburne, P. Beling, B. Horowitz, and C. Fleming. A preliminary design-phase security methodology for cyber–physical systems. *Systems*, 7:21, 2019.

[18] M. Chamanbaz, F. Dabbene, and R. Bouffanais. A physics-based attack detection technique in cyber-physical systems: A model predictive control co-design approach. In *Proceedings of the Australian & New Zealand Control Conference (ANZCC)*, pages 18–23, Auckland, New Zealand, 2019.

[19] Y. C. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. *arXiv preprint arXiv:2005.00611*, 2020.

[20] J. Chen and F. Pan. Dynamic modeling of biotechnical process based on online support vector machine. *Journal of Computers*, 4:251, 2009.

[21] S. Chen, Z. Wu, and P. D. Christofides. Cyber-attack detection and resilient operation of nonlinear processes under economic model predictive control. *Computers & Chemical Engineering*, 136:106806, 2020.

[22] S. Chen, Z. Wu, and P. D. Christofides. A cyber-secure control-detector architecture for nonlinear processes. *AIChE Journal*, 66:e16907, 2020.

[23] S. Chen, Z. Wu, and P. D. Christofides. Decentralized machine-learning-based predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 162:45–60, 2020.

[24] S. Chen, Z. Wu, and P. D. Christofides. Cyber-security of centralized, decentralized, and distributed control-detector architectures for nonlinear processes. *Chemical Engineering Research and Design*, 165:25–39, 2021.

[25] S. Chen, Z. Wu, and P. D. Christofides. Machine-learning-based construction of barrier functions and models for safe model predictive control. *AIChE Journal*, e17456, 2021.

[26] P. D. Christofides, J. F. Davis, N. H. El-Farra, D. Clark, K. R. D. Harris, and J. N. Gipson. Smart plant operations: Vision, progress and challenges. *AIChE Journal*, 53:2734–2741, 2007.

[27] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the NIPS Workshop on Deep Learning and Representation Learning*, Montreal, Canada, 2014.

[29] A. Clark. Control barrier functions for complete and incomplete information stochastic systems. In *Proceedings of the 2019 American Control Conference*, pages 2928–2935, Philadelphia, PA, 2019.

[30] D. A. Crowl and J. F. Louvar. Chemical process safety-fundamentals with applications. *Process Safety Progress*, 30:408–409, 2011.

[31] P. Daoutidis, W. Tang, and A. Allman. Decomposition of control and optimization problems by network structure: Concepts, methods, and inspirations from biology. *AIChE Journal*, 65:e16708, 2019.

[32] R. Davies. Industry 4.0: Digitalisation for productivity and growth. 2015.

[33] R. Dey and F. M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *Proceedings of the 60th IEEE International Midwest Symposium on Circuits and Systems*, pages 1597–1600, Medford, MA, USA, 2017.

[34] A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15:61–66, 1995.

[35] H. Durand. A nonlinear systems framework for cyberattack prevention for chemical process control systems. *Mathematics*, 6:169, 2018.

[36] M. Ellis, H. Durand, and P. D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24:1156–1178, 2014.

[37] E. Eryarsoy, G. J. Koehler, and H. Aytug. Using domain-specific knowledge in generalization error bounds for support vector machine learning. *Decision Support Systems*, 46:481–491, 2009.

[38] P. Falugi and D. Q. Mayne. Getting robustness against unstructured uncertainty: a tube-based mpc approach. *IEEE Transactions on Automatic Control*, 59:1290–1295, 2013.

[39] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC Papers OnLine*, 48:531–538, 2015.

[40] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[41] H. P. Gavin. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems, 2019.

[42] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly detection in cyber-physical systems using recurrent neural networks. In *Proceedings of the 18th IEEE International Symposium on High Assurance Systems Engineering*, pages 140–145, Singapore, 2017.

[43] N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. In *Proceedings of the Conference On Learning Theory*, pages 297–299, Stockholm, Sweden, 2018.

[44] K. Gurney. *An introduction to neural networks*. CRC press, London, 2014.

[45] C. Hassan, M. S. Khan, and M. A. Shah. Comparison of machine learning algorithms in data classification. In *Proceedings of the 24th International Conference on Automation and Computing (ICAC)*, pages 1–6, Newcastle upon Tyne, United Kingdom, 2018.

[46] M. Heidarinejad, J. Liu, and P. D. Christofides. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE Journal*, 58:855–870, 2012.

[47] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[48] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan. Machine learning for power system disturbance and cyber-attack discrimination. In *Proceedings of the 7th International Symposium on Resilient Control Systems*, pages 1–8, Denver, CO, 2014. IEEE.

[49] D. Hioe, N. Hudon, and J. Bao. Decentralized nonlinear control of process networks based on dissipativity—a hamilton–jacobi equation approach. *Journal of Process Control*, 24:172–187, 2014.

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

[51] A. Hoehn and P. Zhang. Detection of replay attacks in cyber-physical systems. In *Proceedings of the 2016 American Control Conference (ACC)*, pages 290–295, Boston, MA, 2016.

[52] L. Huang, X. Nguyen, M. N. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *Proceedings of the 2007 IEEE INFOCOM*, volume 7, pages 134–142, Anchorage, Alaska, 2007.

[53] N. Hudon and J. Bao. Dissipativity-based decentralized control of interconnected nonlinear chemical processes. *Computers & Chemical Engineering*, 45:84–101, 2012.

[54] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-physical systems security—a survey. *IEEE Internet of Things Journal*, 4:1802–1831, 2017.

[55] D. Jakubovitz, R. Giryes, and M. Rodrigues. Generalization error in deep learning. In *Compressed sensing and its applications*, pages 153–193. Springer, 2019.

[56] M. Jankovic. Combining control Lyapunov and barrier functions for constrained stabilization of nonlinear systems. In *Proceedings of the American Control Conference*, pages 1916–1922, Seattle, Washington, 2017.

[57] L. Jin, P. N. Nikiforuk, and M. M. Gupta. Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks. *IEEE Transactions on Automatic Control*, 40:1266–1270, 1995.

[58] W. Jin, Z. Wang, Z. Yang, and S. Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

[59] K. N. Junejo and J. Goh. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 34–43, Xi'an, China, 2016.

[60] M. J. Khojasteh, V. Dhiman, M. Franceschetti, and N. Atanasov. Probabilistic safety constraints for learned high relative degree system dynamics. In *Learning for Dynamics and Control*, pages 781–792. PMLR, 2020.

[61] F. Khorrami, P. Krishnamurthy, and R. Karri. Cybersecurity for control systems: A process-aware perspective. *IEEE Design & Test*, 33:75–83, 2016.

[62] J. Kim, J. Kim, H. L. T. Thu, and H. Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *Proceedings of the International Conference on Platform Technology and Service*, pages 1–5, Jeju, Korea, 2016.

[63] D. Knittel, D. Gigan, and E. Laroche. Robust decentralized overlapping control of large scale winding systems. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1805–1810, Anchorage, AK, 2002.

[64] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6:422–431, 1995.

[65] P. Lee, A. Clark, L. Bushnell, and R. Poovendran. A passivity framework for modeling and mitigating wormhole attacks on networked control systems. *IEEE Transactions on Automatic Control*, 59:3224–3237, 2014.

[66] N. G. Leveson and G. Stephanopoulos. A system-theoretic, control-inspired view and approach to process safety. *AIChE Journal*, 60:2–14, 2014.

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

[68] L. Lindemann, H. Hu, A. Robey, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning hybrid control barrier functions from data. *arXiv preprint arXiv:2011.04112*, 2020.

[69] F. Liu and Y. Yao. Modeling and analysis of networked control systems using hidden markov models. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 2, pages 928–931, 2005.

[70] J. Liu, X. Chen, D. Muñoz de la Peña, and P. D. Christofides. Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. part I: Theory. In *Proceedings of the 2010 American Control Conference*, pages 3148–3155, Baltimore, MD, 2010. IEEE.

[71] J. Liu, D. Muñoz de la Peña, and P. D. Christofides. Distributed model predictive control of nonlinear process systems. *AIChE Journal*, 55:1171–1184, 2009.

[72] S. Liu, A. R. Kumar, J.F. Fisac, R. P. Adams, and P.J. Ramadge. Probf: Learning probabilistic safety certificates with barrier functions. *arXiv preprint arXiv:2112.12210*, 2021.

[73] S. Liu, J. Liu, Y. Feng, and G. Rong. Performance assessment of decentralized control systems: an iterative approach. *Control Engineering Practice*, 22:252–263, 2014.

[74] W. Luo, W. Sun, and A. Kapoor. Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates. *Advances in Neural Information Processing Systems*, 33:372–383, 2020.

[75] L. Magni and R. Scattolini. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42:1231–1236, 2006.

[76] Z. Marvi and B. Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *International Journal of Robust and Nonlinear Control*, 31:1923–1940, 2021.

[77] A. Maurer. A vector-contraction inequality for rademacher complexities. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 3–17, Bari, Italy, 2016.

[78] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk, and P. Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21:1341–1353, 2011.

[79] P. Mhaskar, N. H. El-Farra, and P. D. Christofides. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Systems & Control Letters*, 55:650–659, 2006.

[80] S. R. Mohanty, A. K. Pradhan, and A. Routray. A cumulative sum-based fault detector for power system relaying application. *IEEE Transactions on Power Delivery*, 23:79–86, 2007.

[81] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[82] D. Muñoz de la Peña and P. D. Christofides. Lyapunov-based model predictive control of nonlinear systems subject to data losses. *IEEE Transactions on Automatic Control*, 53:2076–2089, 2008.

[83] B. Niu and J. Zhao. Barrier Lyapunov functions for the output tracking control of constrained nonlinear switched systems. *Systems & Control Letters*, 62:963–971, 2013.

[84] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans. Nonlinear systems identification using deep dynamic neural networks. *arXiv Preprint arXiv:1610.01439*, 2016.

[85] S. Omar, A. Ngadi, and H. H. Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79:33–41, 2013.

[86] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor. Machine learning methods for attack detection in the smart grid. *IEEE Transactions on Neural Networks and Learning Systems*, 27:1773–1786, 2015.

[87] D. B. Pourkargar, A. Almansoori, and P. Daoutidis. Comprehensive study of decomposition effects on distributed output tracking of an integrated process over a wide operating range. *Chemical Engineering Research and Design*, 134:553–563, 2018.

[88] S. M. Richards, F. Berkenkamp, and A. Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of the Conference on Robot Learning*, pages 466–476, Zurich, Switzerland, 2018.

[89] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *Proceedings of the 59th IEEE Conference on Decision and Control*, pages 3717–3724, Jeju Island, South Korea, 2020.

[90] M. Z. Romdlony and B. Jayawardhana. Stabilization with guaranteed safety using control Lyapunov–barrier function. *Automatica*, 66:39–47, 2016.

[91] B. Samanta and K. R. Al-Balushi. Artificial neural network based fault diagnostics of rolling element bearings using time-domain features. *Mechanical Systems and Signal Processing*, 17:317–328, 2003.

[92] R. Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, 19:723–731, 2009.

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

[94] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, 1997.

[95] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177:3799–3821, 2007.

[96] P. Sibi, S. A. Jones, and P. Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, 47:1264–1268, 2013.

[97] J. Singh and M. J. Nene. A survey on machine learning techniques for intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, 2:4349–4355, 2013.

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

[99] E. D. Sontag. Neural nets as systems models and controllers. In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pages 73–79, Yale University, New Haven, CT, 1992.

[100] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7139–7145, Las Vegas, NV, 2020.

[101] B. T. Stewart, A.N. Venkat, J.B. Rawlings, S.J. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59:460–469, 2010.

[102] H. T. Su, T. J. McAvoy, and P. Werbos. Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial & Engineering Chemistry Research*, 31:1338–1352, 1992.

[103] Y. Sun and N. H. El-Farra. Quasi-decentralized model-based networked control of process systems. *Computers & Chemical Engineering*, 32:2016–2029, 2008.

[104] Y. C. Sun and G. H. Yang. Robust event-triggered model predictive control for cyber-physical systems under denial-of-service attacks. *International Journal of Robust and Nonlinear Control*, 29:4797–4811, 2019.

[105] W. Tang and P. Daoutidis. Fast and stable nonconvex constrained distributed optimization: the ELLADA algorithm. *Optimization and Engineering*, 23:259–301, 2022.

[106] K. P. Tee, S. S. Ge, and E. H. Tay. Barrier Lyapunov functions for the control of output-constrained nonlinear systems. *Automatica*, 45:918–927, 2009.

[107] T. T. Tran, O. S. Shin, and J. H. Lee. Detection of replay attacks in smart grid systems. In *Proceedings of the 2013 International Conference on Computing, Management and Telecommunications*, pages 298–302, Ho Chi Min, Vietnam, 2013.

[108] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36:11994–12000, 2009.

[109] J. V. Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49:1225–1231, 1996.

[110] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[111] B. Vastag. Exabytes: Documenting the 'digital age'and huge growth in computing capacity. *Washington Post*, 2011.

[112] R. Vinayakumar, K. P. Soman, and P. Poornachandran. Applying convolutional neural network for network intrusion detection. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 1222–1228, Udupi, India, 2017.

[113] 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.

[114] R. Wang and J. Bao. Distributed plantwide control based on differential dissipativity. *International Journal of Robust and Nonlinear Control*, 27:2253–2274, 2017.

[115] A. Widodo and B. Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21:2560–2574, 2007.

[116] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40:462–467, 2007.

[117] 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.

[118] Z. Wu, F. Albalawi, J. Zhang, Z. Zhang, H. Durand, and P. D. Christofides. Detecting and handling cyber-attacks in model predictive control of chemical processes. *Mathematics*, 6:173, 2018.

[119] Z. Wu, F. Albalawi, Z. Zhang, J. Zhang, H. Durand, and P. D. Christofides. Control Lyapunov-barrier function-based model predictive control of nonlinear systems. *Automatica*, 109:108508, 2019.

[120] Z. Wu, S. Chen, D. Rincon, and P. D. Christofides. Post cyber-attack state reconstruction for nonlinear processes using machine learning. *Chemical Engineering Research and Design*, 159:248 – 261, 2020.

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

[122] Z. Wu and P. D. Christofides. Handling bounded and unbounded unsafe sets in control Lyapunov-barrier function-based model predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 143:140–149, 2019.

[123] Z. Wu and P. D. Christofides. Control Lyapunov-barrier function-based predictive control of nonlinear processes using machine learning modeling. *Computers & Chemical Engineering*, 134:106706, 2020.

[124] Z. Wu, H. Durand, and P. D. Christofides. Safe economic model predictive control of nonlinear systems. *Systems & Control Letters*, 118:69–76, 2018.

[125] Z. Wu, D. Rincon, and P. D. Christofides. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control*, 89:74–84, 2020.

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

[127] 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.

[128] 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.

[129] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48:54–61, 2015.

[130] S. Yaghoubi, G. Fainekos, and S. Sankaranarayanan. Training neural network controllers using control barrier functions in the presence of disturbances. In *Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Rhodes, Greece, 2020.

[131] Y. Yan, R. Wang, J. Bao, and C. Zheng. Robust distributed control of plantwide processes based on dissipativity. *Journal of Process Control*, 77:48–60, 2019.

[132] Z. Yan and J. Wang. Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks. *IEEE Transactions on Industrial Informatics*, 8:746–756, 2012.

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

[134] X. Yin and J. Liu. Subsystem decomposition of process networks for simultaneous distributed state estimation and control. *AIChE Journal*, 65:904–914, 2019.

[135] X. Yin, J. Zeng, and J. Liu. Forming distributed state estimation network from decentralized estimators. *IEEE Transactions on Control Systems Technology*, 27:2430–2443, 2018.

[136] J. Zeng, B. Zhang, and K. Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *Proceedings of the American Control Conference*, pages 3882–3889, New Orleans, LA, 2021.

[137] J. Zhang and J. Liu. Distributed moving horizon state estimation for nonlinear systems with bounded uncertainties. *Journal of Process Control*, 23:1281–1295, 2013.

[138] Z. Zhang, Z. Wu, D. Rincon, C. Garcia, and P. D. Christofides. Operational safety of chemical processes via safeness-index based MPC: Two large-scale case studies. *Computers & Chemical Engineering*, 125:204–215, 2019.

[139] H. Zhao, X. Zeng, T. Chen, and Z. Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–11, Sydney, Australia, 2020.

[140] J. Zhao, J. Wang, and L. Yin. Detection and control against replay attacks in smart grid. In *Proceedings of the 12th International Conference on Computational Intelligence and Security*, pages 624–627, Wuxi, China, 2016.