VIABILITY-BASED CONTROL FOR ROBUSTNESS IN 6G NETWORKS: ENSURING EFFICIENT COMMUNICATION UNDER EXTREME CONDITIONS

by
İDIL BENSU ÇILBIR

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements
for the degree of Master of Science

Sabancı University
July 2024

VIABILITY-BASED CONTROL FOR ROBUSTNESS IN 6G NETWORKS: ENSURING EFFICIENT COMMUNICATION UNDER EXTREME CONDITIONS

Approved by:

Prof. Dr. Özgür ERÇETİN .....
(Thesis Supervisor)

Prof. Özgür GÜRBÜZ .....

Assoc. Prof. Ertuğrul BAŞAR .....

Date of Approval: 22/07/2024

ABSTRACT

VIABILITY-BASED CONTROL FOR ROBUSTNESS IN 6G NETWORKS: ENSURING EFFICIENT COMMUNICATION UNDER EXTREME CONDITIONS

İDIL BENSU ÇILBIR

ELECTRONICS ENGINEERING M.S. THESIS, JULY 2024

Thesis Supervisor: Prof. Dr. Özgür ERÇETİN

Keywords: Viability, Control, Optimization, Dynamical Network Systems, Efficient Communication

The fifth generation of communications, 5G, is presently being deployed in commercial networks worldwide. Novel 6G technologies, which are the next beyond 5G technology, are investigated, including Key Performance Indicators such as delay, network capacity, and user terminal motion velocity. A 6G network is conceived to bring together nomadic, mobile, and stationary extreme edge devices into a single resource pool, which includes mobile and vehicular user equipments, on-board units, and automation-guided vehicles.

The network is inherently uncertain, which complicates service management and decision-making. Extreme edge load has a substantial impact on resource allocation, and heavy workloads or transmission disruptions can result in service degradation and safety concerns. In such cases, local decisions must be made by the entities of the network to guarantee back to safety while achieving the goal.

This thesis proposes a paradigm shift in the design of robust control plane systems, viewing failure as the norm rather than the exception. It is represented by introducing a concept called viability theory, which is a less common but theoretically robust framework that ensures system safety with effective communication and robustness even during severe overloads or outages. The controller aims to obtain sub-optimal local decisions and increase availability to achieve viability. The performance of our proposed method is demonstrated under different network topologies and scenarios, and its improvement in efficient communication is also shown.

ÖZET

6G AĞLARINDA DAYANIKLILIK İÇİN UYGUNLUK TEMELLİ KONTROL: AŞIRI KOŞULLAR ALTINDA VERİMLİ İLETİŞİM SAĞLAMA

İDIL BENSU ÇILBIR

ELEKTRONİK MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, TEMMUZ 2024

Tez Danışmanı: Prof. Dr. Özgür ERÇETİN

Anahtar Kelimeler: Canlılık, Kontrol, Optimizasyon, Dinamik Ağ Sistemleri, Verimli İletişim

Beşinci nesil iletişim, 5G, şu anda dünya çapındaki ticari ağlarda kullanılmaktadır. Bir sonraki 5G ve ötesi teknolojisi olan yeni 6G teknolojileri, gecikme, ağ kapasitesi ve kullanıcı terminali hareket hızı gibi Anahtar Performans Göstergeleri ile birlikte araştırılmaktadır. 6G ağı, göçebe, mobil ve sabit ekstrem uç cihazlarının, araç kullanıcı cihazları, elektronik cihazlar ve otomasyon güdümlü araçlar içeren tek bir kaynakta bir araya getirilmesi ile öngörülmektedir.

Ağ doğası gereği belirsizdir ve bu da hizmet yönetimini karmaşık hale getirir. Aşırı uç yükünün kaynak paylaştırma üzerinde önemli bir etkisi vardır ve ağır iş yükleri ya da iletim kesintileri servision bozulmasına ve güvenlik kaygılarına neden olabilir. Bu gibi durumlarda, ağ ortamının birimleri tarafından yerel kararlar alınarak amacına ulaştırılırken ağı güvenli duruma geri döndürülmesi garantilenmelidir.

Bu tez, hatayı istisnadan ziyade kural olarak gören sağlam bir kontrol sisteminde bir paradigma değişikliğini önermektedir. Bu paradigma değişikliği Uygunluk Teorisi adlı yeni bir konsepti tanıtarak temsil edilmiştir ve az yaygın olmasına rağmen teorik olarak sağlam bir yapı önermektedir. Bu yapı, sistemin güvenliğini, efektif iletişim ve sağlamlığı ile birlikte, ciddi aşırı yüklemeler veya hatalar altında bile garantileyerek sağlamaktadır. Amacı en optimale yakın yerel kararlar elde etmeye çalışarak uygunluğa ulaşmaktır. Önerilen yöntemimizin performansı farklı ağ topolojileri ve senaryolar altında gösterilmiştir ve ayrıca verimli iletişimin yükseldiği de gösterilmiştir.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor, Prof. Dr. Özgür Erçetin. His wisdom and his experience have admired me, and he never lacked his guidance and support throughout my studies at Sabancı University. It was an honor working with him and I am grateful to him for his valuable reviews and discussions which assisted me to complete my work successfully. I could not ask for a better supervisor and mentor.

I would also like to thank my thesis defense committee members Prof. Özgür Gürbüz, and Assoc. Prof. Ertuğrul Başar for taking their time to attend my defense, and I also thank them for kindly reading my thesis and giving valuable feedback.

Lastly, I would like to thank my parents for their infinite support and motivation, encouraging me to write this thesis. I am very grateful for them to let me go on my own way.

To my parents
who always believed in me

TABLE OF CONTENTS

LIST OF TABLES ..... **x**

LIST OF FIGURES ..... **xi**

1. INTRODUCTION AND BACKGROUND **1**

1.1. Introduction and Motivation 1

1.2. Literature Review 4

1.3. Background 7

1.3.1. Viability Theory 7

1.3.2. Barrier Functions 12

1.3.3. Neural Network Controller 14

1.4. Contributions 17

1.5. Thesis Outline 18

2. CONTROL OF THE DYNAMICAL COMMUNICATION SYSTEM **19**

2.1. Preliminaries 19

2.2. System Model 22

2.3. Problem Statement 24

2.4. Viability Guarantee with Control Barrier Function 27

2.5. Controller with Control Barrier Function 30

2.6. Neural Network Controller 35

2.7. Overall Architecture 37

2.7.1. The Structure of The Central Controller 38

2.7.2. Structural Overview of the Controller Approach in the Multi-Agent System 39

3. COMMUNICATION PROTOCOL **41**

3.1. The Communication of A Node and Its Corresponding Controller 41

3.1.1. Sub-Model of The System 42

3.1.2. Control Signal Transmission 42

3.2. The Communication Between Nodes 44
3.2.1. Network Topology Model 44
3.2.2. Data Transmission Between Nodes 46
3.2.3. Scheduling Policy 47
3.3. Algorithm of The Communication Protocol 48

4. EXPERIMENTAL SETUP 51
4.1. System Definition 51
4.1.1. Scenario 1: A System with Identical Nodes 51
4.1.2. Scenario 2: A System with Non-Identical Nodes 53
4.1.3. Scenario 3: A Second System with Non-Identical Nodes 55
4.2. Viability Kernel Definition 57
4.3. MPC Controller Using CBF Function 59
4.4. NN Controller and Its Cooperation with MPC 63

5. NUMERICAL RESULTS 64
5.1. Results of Output and System Safety 64
5.2. Results of Viability Kernel and CBF 70
5.3. Results with MPC and NN Controller 73
5.4. Results with Changing Viability Kernel 77
5.5. Results of Communication Cost 79

6. CONCLUSION & FUTURE WORKS 84

BIBLIOGRAPHY 87

APPENDIX A 94
A.1. Gronwall's Lemma 94

APPENDIX B 95
B.1. Runge-Kutta Methods 95
B.1.1. Runge-Kutta Order Four 96

LIST OF TABLES

Table 2.1. Notations 20

Table 4.1. Table of Functions for All Nodes in Scenario 1 52
Table 4.2. Table of Functions in Node 1 and 2 54
Table 4.3. Table of Functions in Node 3 and 4 54
Table 4.4. Table of Functions in Node 1, Node 2 and Node 3 56
Table 4.5. Table of Functions in Node 4, Node 5 and Node 6 56
Table 4.6. Boundary Functions of Each Node for Scenario 2 58
Table 4.7. Boundary Functions of Nodes 1, 2, and 3 for Scenario 3 58
Table 4.8. Boundary Functions of Nodes 4, 5, and 6 for Scenario 3 59

Table 5.1. Initial State Values of Agents in Scenario 1 65
Table 5.2. Initial States, Viability Kernel Boundary Offset Values and Constant Values in CBF and Extended Class K Functions for Scenario 2 67
Table 5.3. Initial States, Viability Kernel Boundary Offset Values and Constant Values in CBF and Extended Class K Functions for Scenario 3 69
Table 5.4. Complexity of NN+MPC and MPC methods 76
Table 5.5. Overhead of NN+MPC and MPC methods 77

LIST OF FIGURES

Figure 1.1. Parametrized Systems 9

Figure 2.1. Illustration of Viability Kernel 22
Figure 2.2. Control Construction Summarized with a Diagram. 31
Figure 2.3. The Architecture Structure 37
Figure 2.4. The Central Controller Block Diagram 38
Figure 2.5. The Overall Approach Structure 39

Figure 3.1. Sub-Model with The Node and Its Corresponding Controller .. 42
Figure 3.2. The Communication Network Model in A Real-World System 45
Figure 3.3. The Complete Transmission Process Between Nodes 47

Figure 4.1. Communication Topology of MAS with Identical Nodes 52
Figure 4.2. Communication Topology of MAS with Non-Identical Nodes (Scenario 2) 53
Figure 4.3. Communication Topology of MAS with Non-Identical Nodes (Scenario 3) 55

Figure 5.1. Scenario 1 output results with symmetric error constraints 65
Figure 5.2. Scenario 1 synchronization error results when the system has symmetric error constraints 65
Figure 5.3. Scenario 1 output results with asymmetric error constraints ... 66
Figure 5.4. Scenario 1 synchronization error results when the system has asymmetric error constraints 67
Figure 5.5. Scenario 2 output results 68
Figure 5.6. Scenario 2 synchronization error results 68
Figure 5.7. Scenario 3 output results 69
Figure 5.8. Scenario 3 synchronization error results 70
Figure 5.9. Viability Kernels for All Agents in the Second Scenario 71
Figure 5.10. Viability Kernel and CBF of All Agents in the Second Scenario 72
Figure 5.11. Synchronization error results for implicit viability kernel 73
Figure 5.12. Graph Structures and Layers of the Graph Neural Network ... 74

Figure 5.13. Train and Validation Loss Graphs of the Graph Neural Network Model 75
Figure 5.14. Underlying Graph Structures Designed with Three Types of Topologies 75
Figure 5.15. Control input values of each node in the system with NN+MPC and MPC 77
Figure 5.16. Synchronization error results in changing limit functions 78
Figure 5.17. Control input compared with MPC controller in changing limit functions 79
Figure 5.18. Output results in changing limit functions 80
Figure 5.19. Viability Kernel Boundary Offset - Total Transmitted Number of Control Packets 80
Figure 5.20. The number of transmitted bits with viability method 83
Figure 5.21. The number of transmitted bits without viability method 83

1. INTRODUCTION AND BACKGROUND

1.1 Introduction and Motivation

After more than a decade of research and development, the fifth generation of communications, abbreviated as 5G, is now deployed in commercial networks worldwide. While manufacturers and operators battle for market dominance, researchers are already looking beyond 5G to explore new technologies and use them for the next generation, called 6G. During the early stages of technology development, many presentations and articles raise expectations for the future. These expectations for the new communication environment include traditional Key Performance Indicators (KPIs) like delay and network capacity and novel ones like user terminal motion velocity, network reliability, energy consumption, and resource efficiency [1]. Using these needs, the many areas of study are motivated as components of a more extensive 6G system, providing an initial direction for the system. We foresee a 6G network integrating nomadic, mobile, and stationary extreme edge devices into a single resource pool. The device types considered range from powerful mobile and vehicular User Equipments (UEs) and on-board units (OBUs) to stationary customer premises equipment (CPEs), manufacturing floor robots, cobots, automated guided vehicles (AGVs), and unmanned aerial vehicles (UAVs).

This network must be controlled to handle unexpected failures and errors in the system. Some researchers examined to achieve this goal by virtualizing the network system. For example, [2] used a virtualized base station to analyze the performance and power consumption of those upon different scenarios and offered a non-parametric fully-adaptive learning framework for optimizing those base stations. Another paper [3] analyzed the relationship between radio dynamics and computing when the network system is virtualized and decoupled radio and computing control decisions by proposing a deep deterministic policy gradient algorithm to manage the high-

dimensional action space efficiently and to handle the nature of control actions in the system. Authors in [4] aimed to minimize long-term total network operating costs while reacting to changing traffic needs and resource availability. Although these papers provide solutions for the specific purpose of the virtualized network system, there was no deep examination of the external causes of the network's unsafety, resulting in service disruptions.

Particularly, the network is inherently uncertain, posing considerable hurdles to service management and decision-making. The load at the extreme edge level is non-stationary, unlike central aggregation nodes like routers, which have significant consequences for resource allocation. For example, high workload or transmission disruptions can cause service deterioration, resulting in substantial delays and potentially posing safety issues. Similarly, a far-edge failure in a safety-critical situation can cause service outages. In such a network environment, since it is impossible to view the instantaneous global network state from large-scale networks, local decisions must made by making a local observation. In such safety-critical scenarios, these local decisions are necessary for steering back to the safety of the system while reaching its aim. The literature contains many papers that focus on a specific goal and offer optimization methods to achieve it, which will be mentioned in the next section. However, most of these papers did not consider that these entities may act differently from each other. In the real-world environment, devices mostly behave differently, e.g., two smart vehicles move with different speeds and accelerations on a hill. Although devices can self-adapt to the system given its constraints, the target of each agent may be different from that of the other. Each entity must make decisions based on the system constraints to achieve its intended goal, which may be time-varying. In addition, these papers did not mention much about possible safety-critical scenarios that will cause leaving out from its aim. These local decisions are primarily needed for keeping the system safe, and thus, the system can achieve its intended goal with less consideration of system degradations.

While devices try to obtain an optimal path to the target, they should also consider efficient communication. However, there are very limited papers prioritizing the significance of communications decisions in a dynamic environment. Authors in [5] proposed a Bayesian decision framework in which an agent predicts the next triggering instant based on the resources available in the other agent that can communicate. The study [6] offers an approach allowing any table-based parallel reinforcement learning algorithm running reinforcement learning-based applications, keeping the communication overhead minimum in a distributed environment. Therefore, we will also consider communication efficiency while proposing the optimal control method for dynamic environments.

Examining the overall communication cost in the system is essential to analyze communication efficiency. In the literature, some areas highlight the cost for safety: *1)* tele-operated driving and *2)* high density platooning. The paper [7] studied mobility management to meet the requirements of low latency and high reliability for a tele-operated driving use case using the Software Defined Networking paradigm. Another paper [8] also uses Software Defined Networking for their proposed framework, which selects the network and modifies the routing settings to guarantee user QoS and succession of the network load. Authors in [9] a privacy-aware distributed learning framework for QoS prediction, which supports heterogeneous nodes and models, enhancing robustness and generalization capabilities while preserving data privacy by encoding raw input data. The focus of the prediction of QoS for ad-hoc communications in high density platooning is emphasized in [10] by designing a platooning system that drives through different vehicular traffic conditions, collecting position and transmission data for the analysis of packet inter-reception time to choose the model features. Authors in [11] consider vehicle-to-vehicle communication as the critical enabler in high density platooning and present a novel scheduling mechanism for high density platooning for packet reception development and reducing scheduling delay. The objective of [12] is presented in terms of quality of service prediction for high density platooning application, and authors introduce a conditional exponential distribution model for the prediction of the packet inter-reception time, meeting the requirement of information exchange with high reliability and low latency. According to the given areas with references, we will investigate the low communication cost in vehicular communication for safe driving.

As a countermeasure of the failures in a safety-critical scenario, sub-optimal local decisions may be made, leading to downgraded service accuracy, e.g., reliance on compressed data, but still ensuring robustness and improved availability even under severe overloads or failures. In this thesis, we aim to propose a method that represents a paradigm shift in designing robust control plane mechanisms, treating failure as the norm rather than the exception. It is going to be represented by a concept called viability theory. It suggests a theoretically robust framework to assure system safety, although the concept is less common. This framework guarantees system safety with the consideration of effective communication and robustness. The proposed controller intends to increase availability to make optimal local decisions, providing an optimal path to reach its goal while intending to keep the system continuously safe.

1.2 Literature Review

Addressing a system susceptible to failures and errors requires a wide investigation, and it depends on what it aims to achieve. The authors modeled the system based on any service performance of the network, referred to as an optimization problem, and provided methods to solve it. For example, in [13], it is detected that behaviors of a virtualized base station are non-linear and non-monotonic, and therefore, a novel cost model representing the virtualizing resource management in the network is proposed, and then a learning-based orchestration framework is provided such that it determines whether to maintain the previous network settings or reconfigure them by reallocating virtualized resources. In literature, virtualized networks are used to design a model addressing a solution of its services, such as a data-driven model that optimizes the allocation of computing resources [14], and a model based on the orchestration of base stations that deviates significantly from the energy consumption profile and the balance of performance and cost of it [15]. In this thesis, the network model will be represented according to its behaviors, and its sensitivity to errors and failures will be expressed by utilizing a novel concept, which will be defined later.

Such a system needs efficient and effective control to satisfy the requirements of a real-world network scenario. Most of the proposed controllers in the literature are learning-based. The paper [16] offers a new machine learning-based radio controller for the virtualized network such that it optimizes the network performance by pairing it with other components in its open exosystem while considering the computational resource limitations that are typical in cloud platforms. Some other authors in [17] introduce a controller that allows horizontal and vertical flexibility of computer resources in the 5G core of the network. A multi-edge computing orchestration problem is studied in [18] where it jointly controls the network functional splits, the allocated resources and placement locations of computing services, and the routing for each of its data flow, and then a learning-based controller is constructed to address the vast and multi-dimensional action space with the linear growth of the neural network outputs. Another paper [19] presents a primary control approach for organizing vehicle platoon movement within Connected and Automated Vehicle (CAV) systems, dramatically improving traffic efficiency and reducing energy usage, and an attention mechanism-integrated policy network is offered to enhance the performance of CAV communication and decision-making. In a heterogeneous wireless network, a power control optimization problem for each base station's utility function is introduced in [20] while assuring that the utility function of each connected

user is optimal, given each user's unique communications characteristics and type. We will propose a controller for a communication network system that provides an optimal path to achieve the target while considering possible external conditions that may lead to unsafety.

On the other hand, the authors stated the fact that the control cost is another effect of a dynamic communication network system. The paper [21] establishes a functional relationship between communication performance, estimation squared error, and control cost in multi-sensor scalar wireless networked control systems with noisy observations, proposing a sliding window approach for balance. We are going to provide a dynamic system and relate it to a communication network system by explaining the information interaction between nodes.

Our proposed controller aims to guarantee minimizing risks defined for the network, and this will be done by demonstrating the forward invariance of a defined set for a dynamical network. The forward invariance term was used with many proposed methods in the literature. Some authors analyzed the forward invariance of sets in two parts for hybrid dynamical systems. The first part [22] provides a comprehensive overview of forward invariance in hybrid dynamical systems, stating that it is universal and nonunique and that sufficient conditions must be met for it to be robustly forward invariant. The second part [23] discusses the design of controllers for hybrid systems, utilizing differential and difference inclusions to simulate continuous and discrete dynamics and constructing robust, continuous state-feedback laws. Another paper [24] utilizes computational topology principles that measure forward invariance, offering an invariance test, which generates robust forward invariant sets for n-dimensional Lipschitz continuous nonlinear systems. Authors in [25] introduce the negative barrier function, enhancing the observability of the reciprocal barrier function, thereby generating safety criterion and control barrier functions for dynamic systems.

In this thesis, the control barrier function will ensure the forward invariance, which will give its background later. Many papers in the literature have used the control barrier function to ensure the dynamic system's safety. For instance, the paper [23] combines control barrier functions with approximate Nonlinear Model Predictive Control to create an efficient method with continuous-time safety guarantees and recursive feasibility, demonstrating the importance of accounting for discretization errors. Another Model Predictive Control was proposed in [26], introducing an incremental discrete sliding model predictive control technique for NCS output tracking, enhancing robustness, responsiveness, and reducing tracking errors in networked control systems. The sliding Model Predictive Control technique is also proposed in

[27], and in there, it is expanded by introducing an adaptive discrete sliding model predictive control scheme, combining sliding mode control and model predictive control, to improve real-time control performance in networked multi-agent systems. The following letter [28] aims to expand on input-to-state safety (ISSf) and develop ISSf-CBFs for real-time safety-critical controllers in nonlinear systems. It investigates CBFs and ISSf-CBFs and develops a quadratic program-based formulation. Another paper [29] introduces a barrier function approach for high-order control barrier functions, allowing for more complex dynamics and enhancing safety set resilience, thereby reducing motor control effort. Some other authors in [30] explore quadratic program (QP)-based control synthesis for multitask situations involving nonlinear agents, combining robust control barrier function and fixed-time control Lyapunov function requirements. High-order discrete-time control barrier function (CBF) and Lyapunov function enhance constraint satisfaction in high-degree systems, validated through simulations on a three-link manipulator model in the paper [31]. Another letter [32] explores the forward invariance of safe sets using zero-order-hold controllers, defining controller and physical margins, enhancing techniques, and addressing discrete-time CBF requirements through simulations.

In this thesis, we will provide the control barrier function method, and its background will be given later. The paper [33] addresses the invariance of the desired center of mass trajectory and robustness of limited model predictive control for humanoid walking, adjusting system state, viability kernel bounds, and cost function. There are also some papers in the literature that use the viability theory concept. For instance, the paper [34] uses viability theory to design a conservative algorithm for robot navigation to handle time-varying constraints, like moving obstacles. Authors in [35] computed an inner and an outer approximation of the viability kernel with interval analysis tools. Another research [36] got help from the analytical tools of viability theory, characterizing the largest detectable set of target points, to design a dynamic coverage for nonlinear mobile sensor networks to handle arbitrary sensory range, mobility constraints, and collision avoidance, ensuring safety. Authors in [37] satisfied constraints for all future times it is enough to find by defining recursive viability and using it. The optimal control problem is solved by designing reward functions in [38], and it is done by examining the issue from the viability theory perspective. As seen, the viability theory concept was mainly used in the robotics area. In this thesis, we will introduce the idea of viability theory to the communications area.

The control barrier function will provide a model predictive controller. While we will explain the method and show its numerical results, we will also put another controller to obtain the gain of viability, and this controller will be designed with the neural

network. A neural network controller design was previously used in the literature. It contains many techniques for verification of neural network (NN) properties, which is usually motivated by defending against adversarial instances [39]. Although these methods are not intended for closed-loop systems, neural network relaxations in [40] serve as a solid foundation for neural network controllers. Safety verification is achieved with modern methods, such as Hamiltonian-Jacobi [41] for closed-loop systems. Authors in [42] inject conservatism by assuming that the neural network controller can emit extreme values at all states. Another study proposed a neural network controller [37] to focus on controlling multiple non-identical systems with varying initial conditions without individual controllers, using a single control signal to meet input and state constraints, regardless of the number of individual systems.

1.3 Background

This section will provide information about the methods and concepts used in this thesis to introduce novel techniques for use in the communications area.

1.3.1 Viability Theory

Viability theory is a field that studies the evolution of dynamical systems under state constraints, known as *viability constraints*. It aims to determine the largest initial conditions within a set that ensures at least one system trajectory satisfies these constraints for all time [43]. This theory is applied in various domains, including biological evolution, economics, environmental sciences, financial markets, control theory, robotics, and cognitive sciences, to investigate adaptation to these constraints.

The viability theory of mathematical tools aims to directly address the dynamic adaptation of uncertain evolutionary systems to settings characterized by viability constraints. Notably, the environment is described by many sorts of viability constraints, a term that encompasses polysemous ideas such as stability, confinement, homeostasis, adaptability, and so on, expressing the idea that some variables must adhere to limits that can never be breached. A trajectory that meets viability re-

quirements at all times is considered viable. The viability kernel refers to the set of beginning circumstances that allow for viable trajectories. To elaborate on the main concepts of viability theory, some terms will be defined, and these terms will help introduce the viability kernel definition.

Definition 1.1 (Set-Valued Map [43]). *A set-valued map $F : X \rightsquigarrow Y$ correlates with any $x \in X$ a subset $F(x) \subset Y$ (may be the empty set). It is a (single-valued) map $f := F : X \mapsto Y$ if for any x, $F(x) := \{y\}$ is decreased to a single element y. The symbol "$\rightsquigarrow$" represents set-valued maps, while the symbol "$\mapsto$" represents single-valued maps.*

The graph $Graph(F)$ of a set-valued map F is the set of pairs $(x, y) \in X \times Y$ satisfying $y \in F(x)$. If $f := F : X \mapsto Y$ is a single-values map, it coincides with the typical graph concept. The inverse of F is the set-valued map from Y to X identified as follows.

$$x \in F^{-1}(y) \Longleftrightarrow y \in F(x) \Longleftrightarrow (x, y) \in Graph(F) \tag{1.1}$$

Parametrized systems specify the main examples of differential inclusions. Their definitions and block diagram are given below.

Definition 1.2 (Parametrized Systems [43]). *Let $U := \mathbb{R}^c$ be a space of parameters. A parametrized system consists of two blocks:*

- *The "input-output block" associating with any evolution $u(\cdot)$ of the parameter (input) the evolution governed by the differential equation $x'(t) = f(x(t), u(t))$ starting from an initial state (open loop)*
- *The non-deterministic "output-input block," associating with any state a subset $U(x)$ of parameters (output)*

It identifies the set-valued map F related with any x the subset $F(x) := \{f(x, u)\}_{u \in U(x)}$ of velocities parametrized by $u \in U(x)$. The associated evolutionary system S maps any initial state x to the set $S(x)$ of evolutions $x(\cdot)$ starting from x ($x(0) = x$) and governed as follows.

$$\dot{x}(t) = f(x(t), u(t)) \text{ where } u(t) \in U(x(t)) \tag{1.2}$$

or, equivalently, to differential inclusion $\dot{x}(t) \in F(x(t))$

The input-output and output-input blocks of a parametrized systems are illustrated

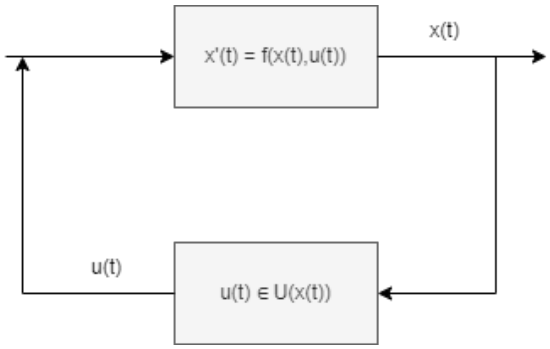


Figure 1.1 Parametrized Systems

in Figure 1.1: The controlled dynamical system at the input-output level, the "cybernetical" map imposing state-dependent constraints on the control at the output-input level.

The parameters range over a state-dependent "cybernetic" map $U : x \rightsquigarrow U(x)$, providing the system opportunities to adapt at each state to viabilities (often, as slowly as possible) and/or to regulate intertemporal optimal evolutions.

To be more specific, we are going to formally define the concepts of the viability kernel of an environment.

Definition 1.3 (Viability [43])**.** *If a subset $K \subset \mathbb{R}^d$ is considered as an environment (defined by viability constraints), an evolution $x(\cdot)$ is said to be viable in the environment $K \subset \mathbb{R}^d$ on an interval $[0, T[$ (where $T \leq +\infty$) if for every time $t \in [0, T[$, $x(t)$ belongs to K.*

The viability kernel is defined under an evolutionary system related to a nonlinear parametrized system.

Definition 1.4 (Viability Kernel [43])**.** *Let K be an environment and S an evolutionary system. The viability kernel of K under the evolutionary system S is the set $Viab_S(K)$ of initial states $x \in K$ from which starts at least one evolution $x(\cdot) \in S(x)$ viable in K for all times $t \geq 0$:*

$$Viab_S(K) := \Big\{ x_0 \in K \;\Big|\; \exists x(\cdot) \in S(x_0) \text{ such that } \forall t \geq 0,\ x(t) \in K \Big\} \tag{1.3}$$

Two extreme situations should be singled out:

- *The environment is said to be viable under S if it is equal to its viability kernel: $Viab_S(K) = K$*
- *The environment is said to be a repeller under S if its viability kernel is empty:*

$$Viab_S(K) = \emptyset$$

It can be said that all evolutions starting from a state belonging to the complement of viability kernel in K leave the environment in finite time.

Aubin provided this formal description for viability kernels and developed the key features that distinguish viability kernels. Aubin, in particular, defined necessary and sufficient requirements that characterize a set's viability kernel using set invariance principles as above derived from the Nagumo Theorem. Aubin's theory relies significantly on nonsmooth analysis. Aubin's nonsmooth analysis allowed him to apply a rather broad concept of tangency to his work. In addition to this essential study of differentiability, Aubin investigated systems characterized by differential inclusions, which resulted in a highly general theory. Despite this broadness, the high analytical formality of nonsmooth analysis allowed Aubin to be exceedingly rigorous in his work.

An illustrative example would be helpful to understand this theory. This example is taken from [44]. Consider a multi-input, single-output nonlinear system affine in the control, and a set of state constraints are given, which are also called viability constraints. These viability constraints define a *safe set* for the system. The aim is to regulate the largest subset of the safe set, called the viability kernel, and an associated control so that the (controlled) system trajectories remain inside the safe set, beginning from any initial condition in the viability kernel, using the control. To achieve this, we will ensure the set invariance, requiring all trajectories to remain inside the safe set.

The following example models the consumption of a renewable resource. Let x_1 represent the quantity of the renewable resource and let x_2 denote the consumption level of the resource. The model is given as shown below.

$$\begin{cases} \dot{x}_1 = (r - x_2)x_1 \\ \dot{x}_2 = u \\ y = x_1 \end{cases} \tag{1.4}$$

where $r > 0$ is the (constant) rate of production (or growth) of the resource and $u \in [-1, 1]$ is the control input. An interesting problem would be to designate that the quantity of the renewable resource is maintained above some positive level $c > 0$. In other words, the output should satisfy the inequality $y = x_1 > c$ to maintain the system viable. Apparently, one can see that there will be specific initial conditions for which solutions of (1.4) will violate the constraint, regardless of the control input

decision (e.g., suppose $x_1(0) = c$ and $x_2(0) > r$). On the other hand, some initial conditions that may threaten safety (i.e., $x_1(0) > c$ and $x_2 > r$) exist, but control also exists such that it will let the system recover before violating the constraint.

There are many approaches to solving viability theory. In this thesis, we are going to solve it using an analytical method. We will introduce a procedure to evaluate a specific formula for the viability kernel and a viability control for a particular class of control systems. A system is considered in a control affine form, with the control taking values in a compact, convex polyhedron. The safe set took the following form.

$$\left\{x \in \mathbb{R}^n \;\middle|\; h(x) \geq c\right\} \tag{1.5}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ represents a smooth function. The subset of the state space for which the system's viability was in jeopardy — that is, the subset for which $h(x) < 0$ — was subsequently identified by taking advantage of the fact that the system under consideration was a control system. In this thesis, the viability kernel will be characterized by this smooth function h, which will be called the control barrier function. The background of the control barrier function will be given later.

On the viability front, several techniques have been offered to approximate the viability kernel. For low dimensional systems with nonlinear dynamics, there are discrete methods such as the viability algorithm [45], those based on the viscosity solutions for Hamiltonian-Jacobi partial differential equations [46], or most recently, interval analysis [35]. Invariance sets have proven more efficient for systems with polynomial dynamics [47]. Lagrangian approaches are practical for higher-dimensional linear systems, as in [48]. Viability Theory has applications in several disciplines, including mobile robots. In [49], Model Predictive Control was used to handle the challenge of maintaining balance and assuring passive safety for a bipedal robot. A discrete technique based on [45] was designed in [50] to ensure safe autonomous racing, i.e., to drive as fast as possible around a predefined track. The aim in [51] was to remove unsafe states from the search area and expedite motion planners, while in [52], it was to improve the safety of systems by preventing them from entering failure zones. A learning technique can lead to misclassification, which may not be a concern in some cases but could jeopardize safety assurances in others.

In this thesis, we will provide a controller such that the system's safe set renders forward invariance. The background of the proposed controller will be given next.

1.3.2 Barrier Functions

Nagumo established the necessary and sufficient requirements for set invariance in the 1940s, paving the way for the study of safety in dynamical systems [53] ([54] gives more detailed history, and [55] gives modern proof). Remarkably, given a dynamical system $\dot{x} = f(x)$ with $x \in \mathbb{R}^n$, assuming that the safe set C is the superlevel set of the smooth function $h : \mathbb{R}^n \to \mathbb{R}$, i.e., $C = \{x \in \mathbb{R}^n \ : \ h(x) \geq 0\}$, and that $\frac{\partial h}{\partial x}(x) \neq 0$ for all x such that $h(x) = 0$, then Nagumo's Theorem gives necessary and sufficient conditions for set invariance based upon the derivative of h on the boundary of C.

$$C \text{ is invariant} \qquad \iff \qquad \dot{h}(x) \geq 0 \qquad \forall x \in \partial C \tag{1.6}$$

Barrier certificates were established as a convenient method to explicitly show the safety of nonlinear and hybrid systems [56]. These results, again, appeared to uncover Nagumo's theorem independently. The name "barrier" was chosen based on its application in optimization literature, where barrier functions are combined with cost functions to avoid undesired regions. In barrier certificates, one considers an unsafe set C_u and a set of initial conditions C_0 together with a function $B : \mathbb{R} \to \mathbb{R}$ where $B(x) \leq 0$ for all $c \in C_0$ and $B(x) > 0$ for all $x \in C_u$. Then, B is a barrier certificate if the following condition is satisfied.

$$\dot{B}(x) \leq 0 \qquad \Rightarrow \qquad C \text{ is invariant} \tag{1.7}$$

In the notation for C above, by choosing the safe set to be the complement of the unsafe set $C = C_u^c$, with $B(x) = -h(x)$ the barrier certificate conditions become: $\dot{h}(x) \geq 0$ which implies that C is invariant. Thus, these conditions are equivalent to Nagumo's theorem. Significantly, barrier certificates were explored [57] and extended to a stochastic situation [58].

Several "Lyapunov-like" procedures have been used to extend the safety assurances beyond the set's border. Lyapunov functions produce invariant level sets, and if these sets are contained in the safe set, one may ensure safety. Importantly, these requirements may be applied over the whole set rather than simply on the edge. As described in [59], a "barrier Lyapunov function" B must be positive definite. Enforcing the constraint $\dot{B} \leq 0$ on the set C assures its invariance and safety. The main constraint is that while these requirements provide safety, they also need invariance at all levels. As a result, they are mighty and conservative.

The concept of a barrier certificate was expanded to include a "control" version,

resulting in the first definition of a "control barrier function" [60], which differs from the one presented in this work. Particularly, given a control system and a safe set C defined by function h, the requirements in [60] can be effectively met.

$$\exists u \text{ s.t. } \dot{h}(x,u) \geq 0 \qquad \Rightarrow \qquad C \text{ is invariant} \tag{1.8}$$

These concepts were expanded upon to link control explicitly Lyapunov functions with barrier functions [61], and this was done concurrently with the development of the techniques described in this study, which combine Lyapunov and barrier functions via the use of optimization-based controllers. Specifically, conditions for establishing "control Lyapunov barrier functions" that jointly provide stability and safety were provided, as elaborated in [62]. However, in such cases, conditions lead to enforcing the reduction of $\dot{h}(x,u) \geq 0$. Yet, these conditions are required,, which motivates the "modern" version of control barrier functions.

All of the previously discussed techniques resulted in the creation of the most recent safety certificates, known as control barrier functions, which acknowledge the previously indicated historical developments. These were introduced firstly in [63] and then clarified in [64]. Specifically, the objective was to apply the barrier function conditions (e.g., Nagumo's discoveries) to the whole safe set. This new class of control barrier functions is defined by the following condition for a control system and a safe set C specified by a function h:

$$\exists u \text{ s.t. } \dot{h}(x,u) \geq -\alpha(h(x)) \qquad \Leftrightarrow \qquad C \text{ is invariant} \tag{1.9}$$

where $\alpha(\cdot)$ is a class K function. This requirement is both necessary and sufficient for compact sets, resulting in minimal restriction. Lastly, as these requirements hold for the entire set C, they provide a means of synthesizing safe controllers, in this case, by modifying the desired controller once again in the least intrusive way. This is done through the application of optimization-based control techniques. Consequently, this formulation offers a fundamental paradigm for safety-critical control.

This unique approach to regulating barrier functions has proven helpful in various applications since its development. Automotive systems [65], multi-robot systems [66], quadrotors [67], and robotic systems including walking robots [68] are some examples of applications the control barrier function is used. Furthermore, it enables the unification of stability (via a control Lyapunov function) and safety (via a control barrier function) in the context of an optimization-based controller; in fact, the development of this new type of barrier function was inspired by optimization-based

controllers that employed control Lyapunov functions.

When all of the above explanations focus on closed dynamical systems, i.e., systems without inputs, the viability theory will expand them to open dynamical systems, e.g., control systems provided by $\dot{x} = f(x) + g(x)u$ for $u \in \mathbb{R}^m$. This required transitioning from invariant sets to controlled invariant sets, which can be made invariant by implementing an appropriate controller. This is the main aim of this thesis, and it will be clarified in detail in the following chapters.

1.3.3 Neural Network Controller

Researchers have created a vast and diverse body of literature on neural networks. Neural network applications fall into two categories: control and signal processing and classification. Furthermore, two categories of control applications exist: closed-loop control and open-loop identification. Since they frequently employ the same algorithms, identification applications are conceptually similar to signal processing and classification applications. However, in closed-loop applications, the neural network is inside the control loop, therefore extra care needs to be taken by the developer to guarantee that the weights of the neural network stay confined throughout the control run [69]. These challenges have hindered the creation of resilient and adaptable neural network closed-loop control applications.

Neural networks exhibit several key characteristics. For control, the function approximation property should be concerned. Let $f(x)$ be a smooth function whose mapping is $\mathbb{R}^n \to \mathbb{R}^m$. As long as x is limited to a compact set $S \in \mathbb{R}^n$, for some number of hidden layer neurons N_h, there are weights and thresholds such that the following equality holds.

$$f(x) = W^T \sigma(V^T x) + \epsilon \tag{1.10}$$

where W and V are the neural network weights in matrices, $\sigma(\cdot)$ is the activation function, and ϵ is the neural network functional approximation error. This applies to many different activation functions. The above equation indicates that a neural network may estimate any smooth function on a compact set.

Finding the optimal neural network weights in matrices W and V to approximate a given nonlinear function $f(x)$ can be challenging. They might not even be unique.

For control, the reader needs to know that there are optimum approximation neural network weights for a given value of ϵ.

Neural networks have practical applications due to their key characteristics. A developer can utilize any neural network activation functions $\sigma(\cdot)$ with a bounded first derivative as long as they have a function approximation property, as shown in the above equation. In dynamical systems, its gain needs to be chosen such that it is large enough to overbound a specific "bad" nonlinear term defined in the dynamical system equation; a large positive number will do. Acceleration measurements are not necessary for the neural network controller, just like in well-designed adaptive controllers. A proof in [69] demonstrates that the signal that should be backpropagated in closed-loop neural network applications is exactly the filtered error. The neural network controller guarantees that the constraint of a system is satisfied throughout time.

The additional robustness qualities provided by the neural network controller are easily shown. The closed-loop system is robust to more unstructured uncertainties and disturbances because the tuning algorithms ensure that it satisfies a crucial state-tight passivity feature. Hidden-layer neural network signals require an additional persistence of excitation (PE) requirement to behave properly. PE conditions are ubiquitous in adaptive control and challenging to prove in multi-layer neural networks.

The neural network controller is comparable to modem adaptive control techniques regarding implementation difficulty in practical systems. Neural network control incorporates concepts from robust control in the signal. Neural network control has two distinct benefits over adaptive control, making it likely to be widely accepted in the industrial control field in the coming years. Extensive system modeling and preliminary analysis must be performed to compute a regression matrix to implement adaptive controllers. Regression matrix computation requires using a model specific to the system that the user wishes to govern and must be done independently. A new regression matrix for the system must be produced whenever the neural network controller is applied to a different system.

The universal approximation property of neural networks leads to the model-free property of neural network controllers. Even though the model-free condition typically lacks both proofs and performance guarantees, it also explains why neural network controller research has succeeded in various scenarios. In other words, neural network controllers withstand both unmodeled disturbances and subpar technical design procedures.

It is easy to understand why neural networks are superior to adaptive control. Adaptive control works under the assumption that the linear-in-the-parameters equation gives the unknown function, which refers to the nonlinearity of the dynamical system. The approximation of the neural network functional approximation, whose equation is shown in the former, is guaranteed to hold for every smooth function $f(x)$, and the activation functions $\sigma(\cdot)$ are always the same for any system. Therefore, whereas the linear-in-the-parameters equation offers a basis set solely for particular systems, neural networks provide a basis set for any smooth function $f(x)$.

Adaptive control significantly lacks another important feature in addition to the challenges associated with calculating the regression matrix. In actuality, the linear-in-the-parameters assumption severely limits the kinds of systems that adaptive control techniques can effectively regulate. It does not hold for all systems. For every system, if the error dynamics are derived using an appropriate control engineering formulation, the neural network approximation property is accurate.

Some implementation strategies of the neural network controller are designed in the literature. In addition, some practical applications, including force control, have been developed with neural network controllers. Digital control methods are mostly used, and therefore, the neural network controller is designed with discrete time. For instance, the neural network controller is simplified with Hebbian weight tuning [69]. Hebbian tuning is no longer often used in neural networks due to poor performance. The controller in the table with updated Hebbian tuning achieves excellent closed-loop performance. Nonlinear controllers typically use digital signal processors or microprocessors, making building neural network controllers with discrete-time weight updating methods crucial. Researchers have written extensively about such tuning techniques. However, these ad hoc modifications of gradient-based algorithms, like the delta rule, do not ensure stability or tracking in closed-loop control applications. The discrete-time adaptive control to nonlinear systems with neural network controller provides the no longer necessary for the following conditions: *i)* certainty equivalence assumptions, *ii)* persistence of excitation, *iii)* linearity in the parameters or *iv)* a regression matrix.

In this thesis, the neural network controller is combined with the control barrier function. The neural network controller guarantees that the dynamical system can reach the target by gaining viability. The control barrier function acts as a model predictive controller to prevent the system from becoming unsafe.

1.4 Contributions

The main contributions of this thesis are given as follows:

- We propose two different controllers for time-varying multi-agent systems, which communication network systems are modeled, and then combine these two controllers to provide a central controller for corresponding agents in the system. The objective here is to address the system susceptible to errors and failures and give this central controller to achieve an effective and efficient performance for such a system.

- The first controller is the Model Predictive Controller, and it aims to steer the agent toward the closest point to the viability region's boundaries when the agent finds itself very near to the boundaries. The proposed method for the Model Predictive Controller is the Control Barrier Function. It is defined over a safe set, which in turn achieves viability, and this will be provided with its solutions. The second controller is the Neural Network Controller, and the goal of this controller is to use the viability region to check whether the agent can reach the target region within a finite time. The importance of both controllers will be explained and shown, especially this will be given for the Model Predictive Controller.

- A communication protocol is specified for the central controller design, and we explain how the communication is optimized efficiently. Here, the communication between nodes/agents and the communication between a node/an agent and its corresponding controller is clarified, and a scheduling policy is designed to consider some limitations in the communication system.

- Numerical simulations are provided. Firstly, the Model Predictive Controller results are provided with two different network models. Secondly, one of the scenario's results is compared with another method's results by examining the communication performance. Finally, the Neural Network Controller is included with the Model Predictive Controller, and its results are compared with the only Model Predictive Controller. It will be highlighted that their collaboration improves efficient decision-making and communication of the network system.

1.5 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 expresses the communication network system as a multi-agent system model and then defines the viability kernel, along with the forward invariance term, which is used to make the system sensitive to failures and errors and guide its safety region. After that, the method of the Model Predictive Controller is proposed with given concepts, and it is mathematically proven with theorems and lemmas. The method we used here is the control barrier function. It is suggested to achieve viability by defining the function over a safe set, which in turn guarantees the system in a defined viability kernel. The Neural Network Controller is also proposed, and its goal is explained. Chapter 3 describes the communication protocol for the system. The communication is divided into two parts: the communication between nodes and the communication between the node and its corresponding controller. Then, a scheduling policy is provided to handle communication limitations, which is also explained in detail. Chapter 4 provides experimental setup by giving three topology scenarios, and then defines viability kernel, control barrier function and other needed parameters. Chapter 5 gives the numerical results of the simulation using the proposed method and communication protocol, and the results are analyzed to show the accomplishment of our aim. Chapter 6 presents a brief conclusion and explains what can be done next in the future.

2. CONTROL OF THE DYNAMICAL COMMUNICATION SYSTEM

This chapter will introduce basic definitions of mathematical terms and the fundamentals of viability kernels and control barrier functions (CBFs). The CBF is an effective method to ensure a system's safety over time. We will modify the CBF method using Viability Theory to address systems susceptible to failures and errors, making the proposed method more effective. All notations used in this chapter is given in Table 2.1.

2.1 Preliminaries

Definition 2.1. *A nonlinear autonomous control system is considered as a differential equation.*

$$\dot{x} = f(x) + g(x)u + d(t), \tag{2.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ denotes a state and control input, respectively. Assume that $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, and $d : \mathbb{R}_{>0} \to \mathbb{R}$ are continuous. The initial state of the system is represented by $x_0 = x(0)$.

The Lie derivative is a mathematical tool used to describe how a function changes along the flow of a vector field. In the context of control systems, it provides insights into the behavior of functions as they evolve over time due to the influence of system dynamics.

Definition 2.2. *Positive and non-negative real numbers are denoted as $\mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$ respectively. For a function $h : \mathbb{R}^n \to \mathbb{R}$, Lie-derivatives with respect to the mappings $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, both of which are continuous, are defined*

Notation	Meaning
$\mathbb{R}^n$	n-Dimensional Vectors of Real Numbers
$\mathbb{R}_{>0}$	Positive Real Numbers
$\mathbb{R}_{\geq 0}$	Non-Negative Real Numbers
$\mathcal{L}_f h$	Lie-derivative of h with respect to f
$\dot{h}$	Time Derivative of h
$\nu(\cdot)$	Control input
$\mathcal{X}$	Safe Set
$\mathcal{U}$	Input Constraint Set
$\mathcal{G}$	Directed Graph
$\mathcal{V}$	Node Set
$\mathcal{E}$	Edge Set
$\mathcal{A}$	Adjacency Matrix
$\mathcal{D}$	Degree Matrix
L	Laplacian Matrix
V_x	Viability Kernel
$\underline{x}$	Vector
$\delta(h)$	Higher-Order Term of h
$\mathbf{W}^j$	Weight Matrix for the j-th Layer
$\mathbf{b}^{(j)}$	Bias Vector for the j-th Layer

Table 2.1 Notations

as follows:

$$\mathcal{L}_f h(x) = \frac{\partial h}{\partial x} f(x) = \lim_{t \to 0} \frac{h(\Phi_f^t(x)) - h(x)}{t}, \tag{2.2a}$$

$$\mathcal{L}_g h(x) = \frac{\partial h}{\partial x} g(x) = \lim_{t \to 0} \frac{h(\Phi_g^t(x)) - h(x)}{t}, \tag{2.2b}$$

where $\Phi_f^t : M \to M$ and $\Phi_g^t : M \to M$, which M is the manifold, describes how points on the manifold move under the influence of f and g, respectively. The flows $\Phi_f^t(x)$ and $\Phi_g^t(x)$ represent the position at point $x \in M$ after time t, as it moves along the curve generated by f and g, respectively. Lie derivatives describe how the function $h(x)$ changes along the flows defined by $f(x)$ and $g(x)$, respectively.

Definition 2.3 (An Extended Class K Function)**.** *A continuous function* $\alpha : (-b, a) \to \mathbb{R}$ *for some* $a, b \in \mathbb{R}_{>0}$ *is an extended class K function if it is strictly increasing and* $\alpha(0) = 0$ *[70]. It should be noted that* b *and* a *may be infinity.*

Definition 2.4 (0-Superlevel Set [71])**.** *The 0-superlevel set* $\mathcal{C}$ *of a smooth function* $h : \mathbb{R}^n \to \mathbb{R}$ *is given as below.*

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\}. \tag{2.3}$$

According to the definition of the nonlinear autonomous system (2.1) and with given the definition of the 0-superlevel set (2.3), we are going to define a safe set as follows.

Definition 2.5 (Safe Set [70]). *A safe set $\mathcal{X} \subset \mathbb{R}^n$ is a strict 0-superlevel set of a continuously differentiable function $h_x : \mathbb{R}^n \to \mathbb{R}$:*

$$\mathcal{X} = \{x \in \mathbb{R}^n | h_x(x) > 0\}. \tag{2.4}$$

With this safe set, we are going to provide a theorem for the existence of a continuously differentiable function.

Theorem 2.1. *A continuously differentiable function $h_x : \mathbb{R}^n \to \mathbb{R}$ exists if the following conditions are satisfied [70].*

- *(A1) $h_x(t)$ is proper $\forall x \in \mathcal{X}$, i.e., for any $L \in \mathbb{R}_{\geq 0}$, the superlevel set $\{x | h_x(t) \geq L\}$ is compact.*
- *(A2) For any continuous mapping $u_h : \mathbb{R} \to \mathbb{R}^m$, a locally Lipschitz continuous extended class K function $\alpha(\cdot)$ exists such that the following inequality is valid:*

$$\sup_{\nu \in \mathbb{R}^m} \dot{h}_x(x, \nu) > -\alpha(h(x)); \quad \forall x \in \mathcal{X}, \tag{2.5}$$

where $\dot{h}_x$ is the time-derivative of h_x. The proof of the Theorem 2.1 is given in [72].

A convex subset $\mathcal{U} \subset \mathbb{R}^m$ stands for input constraints, defined as follows.

$$\mathcal{U} = \{\nu \in \mathbb{R}^m \mid \varrho_i(\nu(u, u_h(t))) \leq 0 \quad (i = 1, \dots r)\}, \tag{2.6}$$

where ϱ_i are continuously differentiable convex functions with respect to u, $\nu(u, u_h(t)) \in \mathcal{U}$ is the control input, defined as a function where u and $u_h(t)$ denote the controller and a supervisory control respectively.

Note that a mapping $u_h : \mathbb{R} \to \mathcal{U}_h := \{u_h \in \mathbb{R}^m | \nu \in \mathcal{U}\}$ is also continuous. Now that we have both state constraint $x(t) \in \mathcal{X}$ and input constraint $\nu(u, u_h(t)) \in \mathcal{U}$ for $t \geq 0$, we will introduce the viability kernel, which is a subset of the safe set, i.e., $V_x \subseteq \mathcal{X}$.

Definition 2.6 (Forward Invariance [54]). *The safe set $\mathcal{X} \subset \mathbb{R}^n$ is forward invariant for the system if for every state $x_0 \in \mathcal{X}$ satisfies $x(t) \in \mathcal{X}$ for $\forall t \geq 0$.*

The concept of forward invariance, first associated with viability theory in [73], guarantees the viability kernel's forward invariance under a control system.

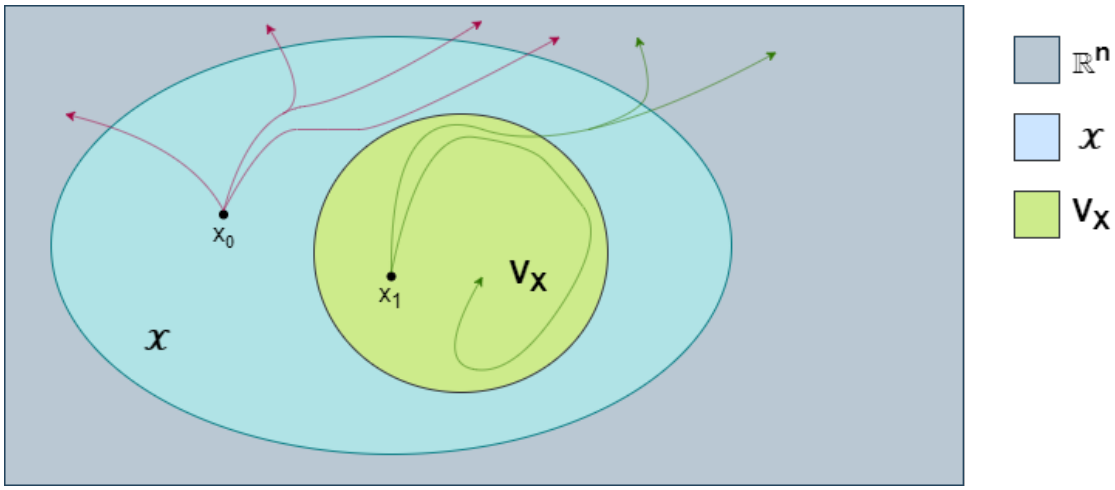


Figure 2.1 Illustration of Viability Kernel

The presence of input constraints, represented by $\nu \in \mathcal{U}$, implies the existence of an initial state $x_0 \in \mathcal{X}$ for which the violation of state constraints is unavoidable. Therefore, we utilize the viability kernel to characterize the initial states for which the system remains viable.

Definition 2.7 (Viability Kernel [43])**.** *Given a system and a safe set $\mathcal{X} \subset \mathbb{R}^n$, a subset $V_x := Viab(\mathcal{X}) \subseteq \mathcal{X}$ is referred to as the viability kernel of $\mathcal{X}$ if there exists a control input $\nu \in \mathcal{U}$ that makes the safe set forward invariant:*

$$V_x = \{x \in \mathcal{X} \mid \exists \nu \in \mathcal{U}, \text{ such that } x(t) \in \mathcal{X}, \forall t \geq 0\}. \tag{2.7}$$

The viability kernel V_x set can be illustrated in Figure 5.9. The difference between V_x and $\mathcal{X}$ is that V_x is satisfied with both state and input constraints, while $\mathcal{X}$ is satisfied with only state constraints.

2.2 System Model

The supervisory control unit acts independently of the internal controllers of each agent. It is in charge of making control decisions that influence the network's general behavior and coordination. Our method connects with the supervisory control unit to improve system robustness and communication efficiency. Specifically, our method assists the supervisory control unit by providing additional viability-based control mechanisms to guarantee agents remain within safe operational limits. This

assistance supplements the supervisory control unit's choices by offering tiered defense against system failures and optimizing communication protocols in dynamic environments.

We outline the system model with input constraints, and we define it as a control-affine system to express its dynamics linearly with respect to the control inputs, simplifying the design and analysis of control strategies for decentralized agent coordination. The definition of the control-affine system with supervisory control $u_h(t)$ is given as follows:

$$\begin{aligned} \dot{x} &= f(x) + g(x)\nu(u, u_h(t)) + d(t) \\ &:= f(x) + g(x)\left[u + u_h(t)\right] + d(t) \end{aligned}, \tag{2.8}$$

where $x \in \mathbb{R}^n$ denotes a state, and $u \in \mathbb{R}^m$ is a controller, respectively. ν is the control input, which is the summation of the controller and the supervisory control. In addition, $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth function, $g(x) : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is a function interacted with the control input, and $d(t) : \mathbb{R}_{>0} \to \mathbb{R}$ is the external disturbance.

Based on (2.8), a heterogeneous nonlinear input-constrained multi-agent system (MAS) is considered composed of N agents, in which agents are indexed from the set $\mathcal{V} = \{1, ..., N\}$. The dynamics of the ith ($i \in \mathcal{V}$) agents can be depicted as shown below:

$$\begin{aligned} \dot{x}_{i,j} &= f_{i,j}(\underline{x}_i) + g_{i,j}(\underline{x}_i)\left[u + u_h(t)\right] + d_{i,j}(t); \quad j = 1, \ldots, n, \\ y_i &= x_{i,1}, \end{aligned} \tag{2.9}$$

where $\underline{x}_i = [x_{i,1}, ..., x_{i,n}] \in \mathbb{R}^n$ and y_i are the state vector and output of the system, respectively, and n ($n \geq 2$) is the system order. Note that n may not be identical, but it is assumed that it is identical for all agents for simplicity. Furthermore, $f_{i,j}(\underline{x}_i) : \mathbb{R}^j \to \mathbb{R}^j$ is a smooth function representing the unmodeled dynamics, $g_{i,j}(\underline{x}_i) : \mathbb{R}^j \to \mathbb{R}^{i\times j}$ is a function interacted with the control input, and $d_i(t) : \mathbb{R}_{>0} \to \mathbb{R}$ is the external disturbance with an unknown bound $\overline{d}_{i,j} \in \mathbb{R}_+$ satisfying $|d_{i,j}(t)| \leq \overline{d}_{i,j}$. The control input is $\nu(u, u_h(t)) = [u + u_h(t)]$, and its constraint set definition (2.6) is updated to the following control set according to the MAS model.

$$\mathcal{U} = \{v \in \mathbb{R} : ||\nu(u, u_h(t))|| \leq a_{max}\}, \tag{2.10}$$

where $a_{max} \in \mathbb{R}_{>0}$ is a positive constant.

Graph Theory

Information transfer between agents can be described using graph theory, which also helps to relate the multi-agent system (MAS) to the communication system. Graph theory is commonly employed to represent the interactions and information exchanges among agents within MAS systems. The system is described by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $\mathcal{V} = \{1, ..., N\} \in \mathbb{R}^N$. In graph $\mathcal{G}$, each node denotes an agent, and the edge $(j, i) \in \mathcal{E}$ demonstrates that the information flows from node j to node i. For nodes $i, j \in \mathcal{V}$, the adjacency matrix $\mathcal{A} = [a_{ij}] \subseteq \mathbb{R}^{N \times N}$, which refers to the interconnection between nodes, is identified by $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$, and $a_{ij} = 0$ otherwise, since there is no allowance of self-loop, $a_{ii} = 0$. The degree of node i is denoted as $\phi_i = \sum_{j=1}^{N} a_{ij}$, and the degree matrix is defined with degree of node $\mathcal{D} = diag\{\phi_1, ..., \phi_N\} \in \mathbb{R}^{N \times N}$. Then, $L = \mathcal{D} - \mathcal{A}$ is defined as the Laplacian matrix of the graph.

2.3 Problem Statement

In many cases, the state of a nonlinear system may approach infinity in finite time, a phenomenon known as finite escape time. It means that within a limited period, the system's state can become extremely large, theoretically approaching infinity, which indicates that the system is experiencing an instability or unbounded behavior in a very short time span. The controller function must ensure the forward invariance of the viability kernel for all $t \in \mathcal{I}(x_0)$, including scenarios where $\mathcal{I}(x_0) = [0, t_1)$ and t_1 represents the finite escape time. This controller function is known as the control barrier function.

Definition 2.8 (Control Barrier Function [71])**.** *Given a system model as defined in (2.8) and the safe set $\mathcal{X}$ (2.4), a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ is a control barrier function (CBF) if there exists an extended class K function $\alpha(\cdot)$ such that the following inequality holds.*

$$\sup_{u \in \mathbb{R}^m} \dot{h}(x) > -\alpha(h(x)) \quad \forall x \in \mathbb{R}^n, \tag{2.11}$$

CBF is used to ensure safety within a specified safe set by keeping the system viable over time. The goal is to maintain all node states within the viability kernel.

Controllers with input constraints (2.10) cannot guarantee the forward invariance of the safe set $\mathcal{X}$ outside the viability kernel. If the system state lies outside the viability kernel V_x, the closed-loop system will unavoidably exit the safe set $\mathcal{X}$ for any $\nu(u, u_h(t)) \in \mathcal{U}$. In that case, there is no control input $u \in \mathcal{U}$ that can prevent the system from eventually leaving the safe set $\mathcal{X}$. The purpose of a CBF is to guarantee that the system remains in $\mathcal{X}$ by enforcing safety at the boundary of $\mathcal{X}$. If the system starts outside V_x, any control action will eventually fail for safety because the system is guaranteed to leave $\mathcal{X}$ at some point. As a result, no CBF function can enforce safety in such conditions. The viability kernel is thus a necessary component when defining an effective CBF, as it encompasses the set of states from which safety can actually be enforced.

The definition of the CBF function will be revised and extended to ensure that the system is viable, and this will be done by formulating CBF with respect to the safe set. This approach guarantees that the system remains within the viability kernel V_x.

Definition 2.9 (Control Barrier Function for Viability Kernel [73])**.** *The system* (2.9) *and the viability kernel* V_x *is given. Then, a continuously differentiable function* $h : \mathbb{R}^n \to \mathbb{R}$ *is a control barrier function for viability kernel if the following conditions hold.*

- (B1) Assume that there exists h such that the viability kernel can be defined as

$$V_x = \{x \in \mathbb{R}^n \mid h(x) > 0\}. \tag{2.12}$$

  The system remains safe as long as $h(x)$ is positive. Therefore, the condition in the viability kernel is referred as the positivity condition of the CBF.

- (B2) $h(x)$ is proper for $\forall x \in V_x$; for any $L \in \mathbb{R}_{\geq 0}$, the superlevel set $\{x | h(x) \geq L\}$ is compact.

- (B3) For any continuous mapping $u_h : \mathbb{R} \to \mathcal{U}_h$, i.e., there exists a locally Lipschitz continuous extended class K functions $\alpha(\cdot)$ such that the following inequality holds:

$$\sup_{\nu \in \mathcal{U}} \dot{h}(x, \nu(u, u_h(t))) > -\alpha(h(x)); \quad \forall x \in V_x. \tag{2.13}$$

The viability kernel V_x is determined based on state and input constraints and can

Algorithm 1 Utilizing Viability Kernel V_x to Obtain CBF $h(x)$

Description: Let V_x is a convex subset of $\mathbb{R}^n$. V_x is convex if for $x, y \in V_x$ and $\lambda \in [0,1]$, $(1-\lambda)x + \lambda y \in V_x$
Input: $V_x \in \mathbb{R}^n$, $x \in \mathbb{R}^n$
Define Viability Kernel V_x:
Determine constraints and conditions for V_x
Construct $h(x)$:
if V_x is defined by explicit constraints **then**
  Set $h(x) = -s(x)$ if $V_x = \{x \mid s(x) \leq 0\}$
else
  Use grid sampling to obtain viable states
  Approximate the boundary of V_x with a convex hull on the viable states
  Apply SVM to find a hyperplane separating viable from non-viable states
  Define decision function $h(x)$ as the signed distance from the SVM hyperplane
  Use $h(x)$ to determine whether a point lies within V_x (i.e., $h(x) > 0$ for points in V_x, $h(x) \leq 0$ otherwise)
end if
Verify $h(x)$:
Ensure $h(x)$ satisfies:
- *Positivity:* $h(x) > 0$ if $x \in V_x$
- *Non-positivity:* $h(x) \leq 0$ if $x \notin V_x$

Update $h(x)$:
Modify $h(x)$ to meet conditions if necessary
Output: CBF $h(x)$ distinguishing points inisde V_x from outside

be explicitly described by boundaries that may be constant or time-varying. An equation identifies V_x as a closed form with these boundaries. We modify this equation to meet the conditions of the CBF function mentioned earlier. Once all three conditions are fulfilled, we can derive the $h(x)$. On the other hand, if the viability kernel V_x is defined as an open form, i.e., there is no explicit formula or algebraic expression for V_x, then we first have to approximate the set of feasible initial conditions. In other words, a simulation has to be run to obtain trajectories which satisfy system constraints. These trajectories will provide us the viable region and we use them to obtain CBF $h(x)$ with numerical approaches (e.g. machine learning methods; Support Vector Machine (SVM) or regression). The pseudo algorithm of these whole process is given in Algorithm 1.

Based on Definition 2.3, the main objective is to design a controller that ensures the forward invariance of the viability kernel with minimal intervention while effectively handling unexpected errors as they arise. In other words, the goal is to maintain the system's safety over time by satisfying both the state constraint $x(t) \in \mathcal{X}$ and the input constraint $\nu(u, u_h(t)) \in \mathcal{U}$ for all $t \geq 0$, while making local decisions in response to sudden failures or disturbances.

2.4 Viability Guarantee with Control Barrier Function

This section will demonstrate that the proposed control function treats errors coming from any agent in the system as a rule when any of the required conditions are not met. The discussion will begin with a theorem related to Lipschitz continuity.

Lemma 2.1. *Consider the system described in* (2.9) *and the safe set* $\mathcal{X}$ *defined by* (2.4). *Recall that* $\alpha(\cdot) : \mathbb{R} \to \mathbb{R}$ *is a locally Lipschitz continuous extended class K function. If* $h : \mathbb{R}^n \to \mathbb{R}$ *is a CBF [70], h is proper for* $x \in \mathcal{X}$ *and* $\dot{h} > -\alpha(h(x))$, *then there exists a positive constant* $\eta \in \mathbb{R}_{>0}$ *such that*

$$\eta h(x) \geq \alpha(h(x)); \quad \forall x \in \mathcal{X}, \tag{2.14}$$

Proof: Note that $x \in \mathcal{X}$ implies $h(x) > 0$. Given function $\alpha(\cdot)$ is locally Lipschitz continuous, it can be represented as follows:

$$\alpha(h(x)) = \eta_1 h(x) + \delta(h(x)), \tag{2.15}$$

where $\eta_1 \in \mathbb{R}_{>0}$ is a positive constant and $\delta : \mathbb{R} \to \mathbb{R}$ is a higher-order term of $h(x)$. To analyze how $\alpha(h(x))$ behaves near $h(x) = 0$, the following limit

$$\lim_{h \to 0} \frac{1}{h} \left(\frac{1}{2} \eta_1 h - \delta(h) \right) = \frac{1}{2} \eta_1 > 0 \tag{2.16}$$

is considered. This means that for small values of $h(x)$, $\delta(h(x))$ is dominated by the linear term $\eta_1 h(x)$. The result of the limit is crucial because it shows that for small values of $h(x)$, the higher-order term $\delta(h(x))$ is sufficiently small relative to $\eta_1 h(x)$. Thus, a positive constant $C \in \mathbb{R}_{>0}$ exists such that for all $x \in \mathcal{B} = \{x | h(x) < C\} \subset \mathcal{X}$ the following equation holds

$$\frac{1}{2} \eta_1 h(x) \geq \delta(h(x)). \tag{2.17}$$

By substituting equations (2.17) into (2.15), it implies that a positive constant η $(> \eta_1) \in \mathbb{R}_{>0}$ exists such that the following equation holds

$$\eta \geq \frac{\alpha(h(x))}{h(x)}; \quad \forall x \in \mathcal{B} \subset \mathcal{X}. \tag{2.18}$$

The case $h(x) \geq C$, i.e., $x \in \mathcal{X}\backslash\mathcal{B}$, is now going to be considered. Because $h(x)$ is a proper function for all $x \in \mathcal{X}$, the superlevel set $\mathcal{X}\backslash\mathcal{B} = \{x|h(x) \geq C\}$ is also compact. Therefore, there exists a positive constant $\eta \in \mathbb{R}_{>0}$ such that the following equation holds.

$$\eta \geq \frac{\alpha(h(x))}{h(x)}; \quad \forall x \in \mathcal{X}\backslash\mathcal{B} \subset \mathcal{X}, \tag{2.19}$$

due to the extreme value theorem, which is expressed and its prove is given below.

Theorem 2.2 (Extreme Value Theorem [74])**.** *Let z be a function defined on the closed interval $I = v : a \leq x \leq b$, and assume z is continuous on Y. Then, there exist points v_0 and v_1 within Y such that $z(v_0) \leq z(v) \leq z(v_1)$ for all $z \in Y$. In other words, z attains both its maximum and minimum values on Y.*

Proof: According to Theorem 2.2, the range of z is bounded. Let us define:

$$R = \sup z(v) \text{ for } z \in Y, \qquad r = \inf z(v) \text{ for } z \in Y.$$

We aim to show there exist points v_0 and v_1 in Y such that $z(v_0) = r$ and $f(x_1) = R$. We will first prove the existence of v_1, with the proof for v_0 being similar. Assume that R is not in the range of z, and we will arrive at a contradiction. Consider the function Z on Y defined by:

$$Z : v \to \frac{1}{R - z(v)}.$$

This function is continuous on Y and, by Theorem 2.3, has a bounded range. Let $\overline{R} = \sup Z(v)$ for $v \in Y$. Since $\overline{R} > 0$, we have:

$$\frac{1}{R - z(v)} \leq \overline{R} \text{ or } z(v) \leq R - \frac{1}{R} \text{ for } v \in Y.$$

This inequality contradicts the assumption that $R = \sup z(v)$ for $v \in Y$. Therefore, R must be within the range of z, meaning there exists $v_1 \in Y$ such that $z(v_1) = R$.

■

The compactness of the set $\mathcal{X}\backslash\mathcal{B}$ implies that $h(x)$ attains its maximum and minimum values on this set due to the extreme value theorem. By this theorem, since $h(x)$ is continuous on the compact set $\mathcal{X}\backslash\mathcal{B}$, it must attain its maximum and minimum on the set.

Let us denote the maximum of $h(x)$ on $\mathcal{X}\backslash\mathcal{B}$ as $h(x_b)$, where $x_b \in \mathcal{X}\backslash\mathcal{B}$ is the point where maximum occurs. The extreme value theorem ensures that for all $x_b \in \mathcal{X}\backslash\mathcal{B}$, the ratio $\frac{\alpha(h(x_b))}{h(x_b)}$ is bounded. Therefore, there exists a positive constant $\eta \in \mathbb{R}_{>0}$ such that:

$$\eta \geq \frac{\alpha(h(x_b))}{h(x_b)}; \quad \forall x_b \in \mathcal{X}\backslash\mathcal{B} \subset \mathcal{X},$$

This constant η comes directly from the boundedness of the ratio $\frac{\alpha(h(x_b))}{h(x_b)}$, which is guaranteed by the compactness of the set $\mathcal{X}\backslash\mathcal{B}$ and the continuity of the function $h(x)$. The inequality in (2.19) is thus derived using the fact that on a compact set, a continuous function attains both its maximum and minimum, ensuring that the ratio is bounded and a constant η can be found to satisfy the inequality. ■

By using Lemma 2.1, we will provide the following theorem.

Theorem 2.3. *Consider the system defined in* (2.9)*, with the viability kernel* V_x *as defined in* (2.12) *and a CBF* $h : \mathbb{R}^n \to \mathbb{R}$*. Any continuous controller* u *that satisfies* $\dot{h}(x, \nu(u, u_h)) \geq -\alpha(h(x))$ *ensures that the viability kernel* V_x *remains forward invariant, thereby guaranteeing that* $x \in \mathcal{X}$ *for all* $t \geq 0$*.*

Proof: If η exists for the safe set $\mathcal{X}$ such that the former equations hold, then there also exists a positive constant $\varpi \in \mathbb{R}_{>0}$ for viability kernel V_x, since $V_x \subset \mathcal{X}$.

Specifically, if $x_0 \in V_x$ then $h(x_0) > 0$. Under this condition, the time derivative of the CBF function satisfies $\dot{h} > -\alpha(h(x))$, as assumed. The extension of Lemma 2.1 demonstrates that there exists a positive constant $\varpi \in \mathbb{R}_{>0}$ such that the following equation holds.

$$\varpi h(x) \geq \alpha(h(x)); \quad \forall x \in V_x, \tag{2.20}$$

and thus, by replacing the assumption $\dot{h} > -\alpha(h(x))$ for the viability kernel V_x and substituting it to (2.20), the result becomes as follows,

$$\dot{h} \geq -\alpha(h(x)) \geq -\varpi h(x); \quad \forall x \in V_x, \tag{2.21}$$

Hence, the following inequality holds according to Gronwall's lemma (see Appendix A.1):

$$h(t) \geq h(x_0)\exp(-\varpi t) > 0; \quad \forall t \in [0, t_1], \tag{2.22}$$

where $t_1 \in \mathbb{R}_{>0}$ can be arbitrarily extended. Since $h(x)$ is a proper function for all $x \in V_x$, the level set $\{x|h(x) \geq h(x_0)\exp(-\varpi_1 t_1)\} \subset V_x$ is compact; therefore, any continuous controller u such that $\dot{h} \geq -\alpha(h(x))$ renders the viability kernel V_x forward invariant, and the inclusion demonstrates that $x(t) \in \mathcal{X}$ for $\forall t \geq 0$.

This discussion completes the proof of Theorem 2.3. ■

Theorem 2.3 establishes the global Lipschitz condition for a CBF, which is a significant result that guarantees the solution $x(t)$ to the system (2.9) can continue to evolve over time without interruptions or encountering singularities.

A key property derived from condition (B2) is that the solution $x(t)$ to the system (2.9) does not experience finite escape time. Additionally, the invariance of CBF properties under coordinate transformations will be maintained if condition (B2) is satisfied.

2.5 Controller with Control Barrier Function

Building on the conditions of the CBF to achieve viability, we propose a controller for the system (2.9) as a key result of this thesis. Its summary is given in Figure 2.2.

Theorem 2.4. *Consider the system* (2.9)*, the safe set* $\mathcal{X}$ *defined by* (2.4)*, and the CBF* $h : \mathbb{R}^n \to \mathbb{R}$ *that satisfies the given conditions. For any continuous mapping* $u_h : \mathbb{R} \to \mathbb{R}^m$*, the following controller* $u = k(x,t)$ *ensures the forward invariance of* $\mathcal{X}$*.*

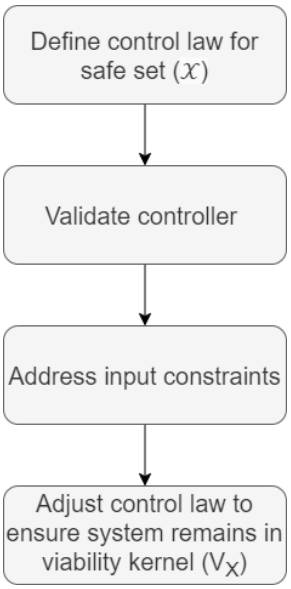


Figure 2.2 Control Construction Summarized with a Diagram.

$$u = k(x,t) = \begin{cases} -\frac{I(x,u_h(t))-J(x)}{||\mathcal{L}_g h(x)||^2}\left(\mathcal{L}_g h(x)\right)^T, & \text{if} \quad I(x,u_h(t)) < J(x) \\ 0, & \text{if} \quad I(x,u_h(t)) \geq J(x) \end{cases}, \tag{2.23}$$

where functions $I : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $J : \mathbb{R}^n \to \mathbb{R}$ are defined as follows.

$$I(x,u_h(t)) = \mathcal{L}_f h(x) + \mathcal{L}_g h(x) \cdot u_h(t), \tag{2.24a}$$

$$J(x) = -\alpha(h(x)), \tag{2.24b}$$

Proof: The system (2.8) can be rewritten with the proposed controller.

$$\begin{aligned} \dot{x}_{i,j} &= f_{c_{i,j}}(\overline{x}_i, k_i(\overline{x}_i,t), u_h(t)) + d_{i,j}(t), \qquad\qquad j = 1,\ldots,n, \\ &= f_{i,j}(\overline{x}_i) + g_{i,j}(\overline{x}_i)\left[k(x,t) + u_h(t)\right] + d_{i,j}(t), \end{aligned} \tag{2.25}$$

First of all, we will prove the continuity of the proposed controller (2.23) to guarantee that a local solution $x(t)$ exists for (2.25).

Note that $\mathcal{L}_f h(x)$, $\mathcal{L}_g h(x)$ and $u_h(t)$ are all continuous functions in t, so functions $I(x,u_h(t))$ and $J(x)$ are also continuous. Furthermore, $u \to 0$ as $I \to J$ uniformly when $\mathcal{L}_g h(x) \neq 0$. Therefore, the proposed controller (2.23) is continuous when $\mathcal{L}_g h(x) \neq 0$. Hence, we only need to prove the continuity of (2.23) at $\mathcal{L}_g h(x) = 0$. According to the Lipschitz continuity condition for the CBF function, for all $t_0 \geq 0$ such that $L_g h(t_0) = 0$, the following inequality holds:

$$\dot{h} = \mathcal{L}_f h(t_0) > -\alpha(h(t_0)), \tag{2.26}$$

Therefore, this shows that a positive constant exists $\epsilon \in \mathbb{R}_{>0}$ such that the following equation holds.

$$\dot{h} = \mathcal{L}_f h(t_0) - \epsilon \geq -\alpha(h(t_0)), \tag{2.27}$$

Since $\mathcal{L}_g h(t_0) \cdot u_h(t_0)$ is a continuous mapping, a neighborhood $\mathcal{H} \subset \mathbb{R}^n \times \mathbb{R}$ of $x(t_0)$ exists such that the following inequality holds.

$$||\mathcal{L}_g h(t_0) \cdot u_h(t_0)|| \leq \epsilon; \quad \forall x(t_0) \in \mathcal{H}, \tag{2.28}$$

Hence, by replacing ϵ with (2.28), the inequality holds for any $x \in \mathcal{H}$ as shown below.

$$\mathcal{L}_f h(t_0) + \mathcal{L}_g h(t_0) \cdot u_h(t_0) \geq -\alpha(h(t_0)), \tag{2.29}$$

Accordingly, this shows that $I(x, u_h(t)) \geq J(x)$. Thus $u = k(x,t) = 0$ for any $x \in \mathcal{H}$, and this shows that a neighborhood $\mathcal{H}$ of $\mathcal{L}_g h(x) = 0$ such that $u = k(x,t) = 0$ for $\forall x \in \mathcal{H}$. Hence, $u = k(x,t)$ is continuous for $\forall x \in \mathbb{R}^n$. Thus, for any continuous mapping $u_h(t)$, the mapping $f_{c_{i,j}}$ is continuous in x and thus a local solution $x(t)$ to the system (2.25)

Secondly, we will prove that the proposed controller (2.23) ensures the forward invariance of $\mathcal{X}$.

The time derivative of the CBF function is calculated as shown below:

$$\begin{aligned} \dot{h} &= \frac{dh}{dx}\frac{dx}{dt} \\ &= \mathcal{L}_f h + \mathcal{L}_g h \cdot k(x,t) + \mathcal{L}_g h \cdot u_h(t), \end{aligned} \tag{2.30}$$

Now, the Lipschitz continuity condition will be examined with cases defined in the controller function (2.23).

Case 1: $I \geq J$

In this case, there is no control update, according to (2.23). Since $u = 0$, the following inequality holds:

$$\dot{h} = I(x, u_h(t)) = \mathcal{L}_f h + \mathcal{L}_g h \cdot u_h(t) \geq -\alpha(h(x)), \tag{2.31}$$

Case 2: $I < J$

For this case, substituting (2.23) into (2.30) yields to the following equation

$$\dot{h} = J(x) = -\alpha(h(x)), \tag{2.32}$$

As a result, the following inequality holds in both cases 1 and 2:

$$\dot{h} \geq -\alpha(h(x)), \tag{2.33}$$

The right-hand side of (2.33) is globally Lipschitz continuous in h for all $x \in \mathcal{X}$. Consequently, there exists a unique bounded function $\underline{h}(t)$ over the interval $[0, t_1]$, where $t_1 \in \mathbb{R}_{>0}$ can be extended arbitrarily. In this context, the validity of the inequality, as established by Gronwall's lemma, is applicable. This was previously demonstrated in (2.22).

According to the compactness of $h(x)$ function with the level set $\{x | h(x) \geq h(x_0)\exp(-\varpi_1 t_1)\}$, there exists a positive constant $s \in \mathbb{R}_{>0}$ such that $\{x \mid ||x - x_0|| \leq s\}$ is supset of the level set. Consider the rectangle $\mathcal{S}$ defined as shown below.

$$\mathcal{S} = \{0 \leq t \leq t_1, \quad ||x - x_0|| \leq s\}, \tag{2.34}$$

where s is sufficiently large. Since the mapping $f_{c_{i,j}}$ is continuous in x on the rectangle $\mathcal{S}$, at least one solution $x(t)$ to (2.25) exists and it is defined on $[0, t_1]$, and $t_1 \in \mathbb{R}_{>0}$ can be extended indefinitely by (2.22); for any constant $t_1 \in \mathbb{R}_{>0}$, a solution exists $x : [0, t_1] \to \mathcal{X}$ for any initial state $x \in \mathcal{X}$. Thus, the proposed controller (2.23) guarantees the forward invariance of $\mathcal{X}$, i.e., $x(t) \in \mathcal{X}$ for $\forall t \geq 0$.

This completes the proof of the forward invariance of $\mathcal{X}$ with the proposed controller, which was initially derived without considering input constraints. ■

To establish that the viability kernel also ensures forward invariance, input constraints will now be incorporated. We will make minimal modifications to the proposed controller to ensure viability. This modified controller is a specialized version of Theorem 2.4 adapted for the viability kernel.

Theorem 2.5. *Consider the system (2.9) with the input $\nu(u, u_h(t)) \in \mathbb{R}$, the input constraint (2.10), and a CBF function $h : \mathbb{R}^n \to \mathbb{R}$. For any continuous mapping $u_h : \mathbb{R} \to [-a_{max}, a_{max}]$, the following defined controller function $u = k(x,t)$ always satisfies the input constraint.*

$$k(x,t) = \begin{cases} -\frac{1}{\mathcal{L}_g h(x)} \left(I(x, u_h(t)) + J(x) \right), & \text{if} \quad I(x, u_h(t)) < J(x) \\ 0, & \text{if} \quad I(x, u_h(t)) \geq J(x) \end{cases}, \tag{2.35}$$

where $I(x, u_h(t))$ and $J(x)$ were already presented at (2.24).

Proof: If the controller defined in (2.35) is applied to the time derivative of the CBF function h, it results in $\dot{h} > -\alpha(h(x))$ for both cases: $I(x, u_h(t)) < J(x)$ and $I(x, u_h(t)) \geq J(x)$. These cases will be analyzed in the context of the viability kernel.

Case 1: $I(x, u_h(t)) \geq J(x)$

In Case 1, where the controller function is zero ($k(x,t) = 0$), the input constraint $u_h \in [-a_{max}, a_{max}]$ is clearly satisfied. Consequently, we consider the other case: $I(x, u_h(t)) < J(x)$. If $L_g h \geq 0$, then the control input $\nu_P(u, u_h(t)) = u_P + u_h(t) = a_{max}$ maximizes $\dot{h}$ and satisfies the following conditions:

$$\max_{\nu \in \mathcal{U}} \dot{h}(x, \nu) = \mathcal{L}_f h(x) + \mathcal{L}_g h(x) \cdot a_{max} > -\alpha(h(x)), \tag{2.36}$$

coming from the Lipschitz continuity condition for a CBF function. Thus, the following equation is valid.

$$-\frac{1}{\mathcal{L}_g h(x)} \left(\mathcal{L}_f h(x) + \alpha(h(x)) \right) < a_{max}, \tag{2.37}$$

Case 2: $I(x, u_h(t)) < J(x)$

In Case 2, the controller function (2.35) is rewritten and updated to the following function.

$$k(x,t) + u_h(t) = -\frac{1}{\mathcal{L}_g h(x)} \left(\mathcal{L}_f h(x) + \alpha(h(x)) \right), \tag{2.38}$$

Correspondingly, the inequality $k(x,t) + u_h(t) < a_{max}$ is satisfied. If $\mathcal{L}_g h(x) < 0$, the control input $\nu_p(u, u_h(t)) = u_p + u_h(t) = -a_{max}$ maximizes $\dot{h}$ and the inequality $k(x,t) + u_h(t) > -a_{max}$ is satisfied similarly with the case of $\mathcal{L}_g h(x) \geq 0$. This shows that the proposed controller (2.35) satisfies the input constraints (2.10). ■

From now on, we will refer to the proposed controller (2.35) as the Model Predictive Controller (MPC). This MPC is designed to continuously ensure the safety of the system under the given constraints. We incorporated Viability Theory into the MPC definition to address system failures or errors as they arise, rather than applying the controller at all times. The controller leverages the learned viability kernel to maintain safety.

We have designated the controller (2.35) as an MPC because its control input generation is based on the behavior of agents within the system. Each agent may exhibit different behaviors, and this non-linearity can potentially lead to unsafe conditions. Therefore, the MPC controller, in conjunction with the CBF, generates control inputs tailored to the dynamics of each agent. This approach ensures that all agents remain in a safe state, thereby ensuring the overall safety of the system. The MPC controller effectively manages the challenges posed by nonlinearity, which is a key factor contributing to system failures or errors.

2.6 Neural Network Controller

The neural network controller leverages gains provided by the viability region. When integrated with the viability region, it can assess whether an agent can reach the target within a finite time. The MPC controller is activated when the agent deviates due to external effects or nonlinearity, placing it near the boundaries of the region. It then guides the agent towards a point closer to the center of the viability region.

According to the system definition given in (2.9), we assume an output-feedback controller $u_t^{NN} = \pi(y_i)$, where $\pi(\cdot)$ represents the control policy of the neural network controller. This controller is parametrized by an n-layer neural network model and is subject to input constraints $u_t^{NN} \in \mathcal{U}$, as defined in (2.10). For the n-layer neural network, the number of neurons in each layer is denoted by n_j, for all $j \in [n]$, where $[n]$ denotes the set $\{1, 2, \ldots, n\}$. The weight matrix for the j-th layer is denoted as $\mathbf{W}^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}$, and the bias vector for the j-th layer is $\mathbf{b}^{(j)} \in \mathbb{R}^{n_j}$. An operator that maps the network input, which measures the output y_i, to the j-th layer is denoted as $\mathbf{F}_j = \mathbb{R}^{n_x} \to \mathbb{R}^{n_j}$. This operator is defined as follows:

$$\mathbf{F}_j(y_i) = \sigma(\mathbf{W}^{(j)}\mathbf{F}_{j-1}(y_i) + \mathbf{b}^{(j)}), \qquad \forall j \in [n-1], \tag{2.39}$$

where $\sigma(\cdot)$ is the coordinate-wise activation function. General activation functions, including ReLU, $\sigma(x) = \max(0, x)$, are applied. The network input produces the control input as follows.

$$u_t^{NN} = \pi(y_i) = \mathbf{F}_n(y_i) = \mathbf{W}^{(n)}\mathbf{F}_{m-1}(y_i) + \mathbf{b}^{(n)}, \tag{2.40}$$

The key step in computing an optimal path to the target is to relax the nonlinear constraints induced by the nonlinear activation functions of the neural network. The relaxation transforms nonlinearities into linear upper and lower bounds that apply to the activation's known input ranges.

The ϵ-ball is going to be identified to denote the range of inputs to the activation functions. The ϵ-ball under the ℓ_p norm is represented, centered at x, with scalar radius ϵ as follows

$$B_p(x, \epsilon) = \left\{ x \;\middle|\; ||x - x||_p \leq \epsilon \right\}, \tag{2.41}$$

The ϵ-ball to the vector case $\overline{\epsilon} \in \mathbb{R}^n_{\geq 0}$, identified as shown below.

$$B_p(x, \epsilon) = \left\{ x \;\middle|\; \lim_{\overline{\epsilon} \to \overline{\epsilon}^+} ||(x - x) \oslash \overline{\epsilon}||_p \leq 1 \right\}, \tag{2.42}$$

where $\oslash$ represents element-wise division. The convex relaxation of the neural network is given as a theorem below.

Theorem 2.6 (Convex Relaxation of NN)**.** *An n-layer neural network control policy is given $\pi : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$. Two explicit functions exist $\pi_l^L : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ and $\pi_l^U : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$ such that $\forall l \in [n_j]$, $\forall y \in B_p(y, \epsilon)$, the inequality $\pi_l^L(y) \leq \pi_l(y) \leq \pi_l^U(y)$ holds true, where $\pi_l^L(y)$ and $\pi_l^U(y)$ are given as shown below.*

$$\pi_l^U(y) = \Omega_l : y + \mu_l, \tag{2.43}$$

$$\pi_l^L(y) = \Psi_l : y + \zeta_l, \tag{2.44}$$

where $\Phi, \Psi \in \mathbb{R}^{n_u \times n_y}$ and $\mu, \zeta \in \mathbb{R}^{n_u}$ are defined recursively using NN weights, biases, and activations. The proof of this theorem is given in [40].

Theorem 2.3 provides constraints on the control values for a specific measurement of y in a closed-loop system. It establishes limits for y and u based on the known

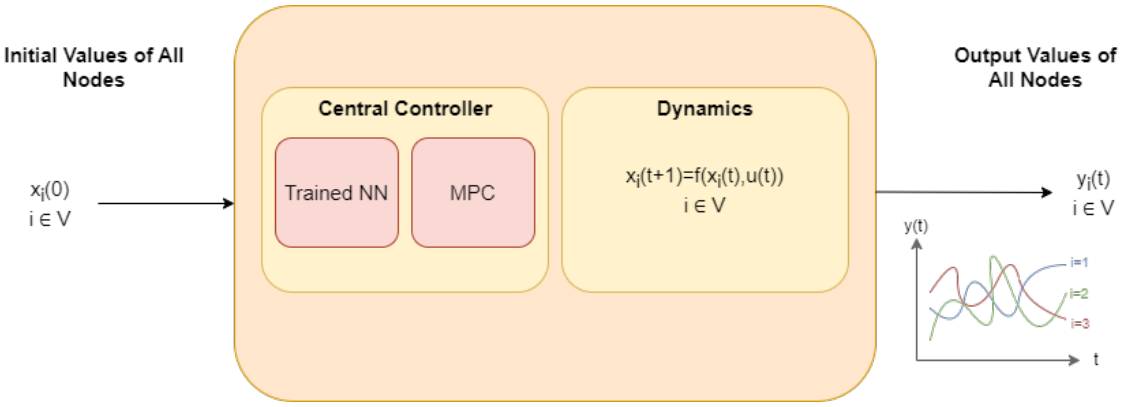


Figure 2.3 The Architecture Structure

set of potential y values, facilitating the efficient determination of neural network output boundaries.

In this context, the input to the neural network (NN) model is the output value of the node y_i. However, to ensure that the agent can reach the target within a finite time, we need to incorporate the boundaries of the viability region. The viability region is defined by the viability kernel in (2.12), where the control barrier function $h(x)$ may include time-varying functions that define the boundaries of the viability region. The neural network controller must utilize these boundaries to meet the requirement of achieving the intended goal in a finite time. Thus, these performance functions should be included as inputs, along with the output values of the nodes.

2.7 Overall Architecture

This section details the architecture using the provided definitions and explanations of the CBF and the NN controller.

The system architecture is illustrated in block diagrams, as shown in Figure 2.3. This figure depicts the architecture for each node individually. While the neural network controller is trained for all nodes in the multi-agent system, the requirements of the MPC may differ from node to node. Therefore, each node can be considered to have its own dedicated controller within the system. The communication between nodes will be discussed in the next chapter, and the structure of the central controller will be detailed in the following subsection.

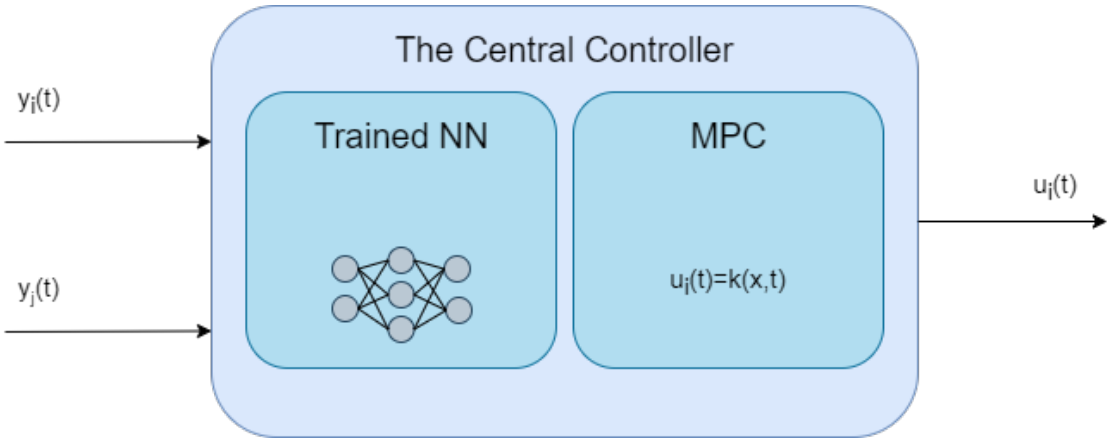


Figure 2.4 The Central Controller Block Diagram

In this architecture, the initial values of each node $x_i(0)$, for $i \in \mathcal{V}$, serve as the input. The *Dynamics* block represents the behavioral evaluation of an agent, with its input being the state value $x_i(t)$. The output from the *Dynamics* block is forwarded to the corresponding central controller (illustrated as the *Central Controller* block in the figure) for control decision evaluation. The control decision is then sent back to the *Dynamics* block to determine the next state of the agent. This setup constitutes a feedback system.

2.7.1 The Structure of The Central Controller

The central controller comprises two distinct controllers that work in tandem. As previously mentioned, NN controller ensures that the agent reaches the target within a finite time. Meanwhile, the MPC is responsible for guiding the agent back to the viability region boundaries if it is displaced by external factors.

The central controller is illustrated in Figure 2.4. Its inputs include the output values of the node and its neighboring nodes. The central controller first assesses whether the agent is within the viability region. If the agent is outside this region, the MPC takes control to return the agent to the safe zone. If the agent is within the viability region, the NN controller activates to determine if the agent can achieve the system's objectives and make appropriate adjustments. In the figure, the output value of the node is denoted as $y_i(t)$, the output value of the neighboring node is $y_j(t)$, and the control input value of the node is represented as $u_i(t)$.

The proposed controller integrates NN and MPC to ensure both safety and efficient operation. The NN controller handles control inputs when the system state is far

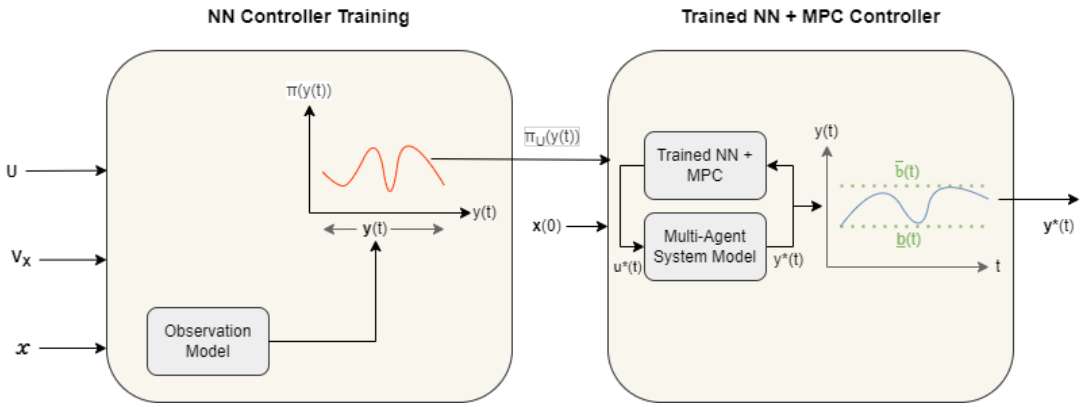


Figure 2.5 The Overall Approach Structure

from safety boundaries, using its computational speed for effective control. MPC takes over near the boundaries, ensuring the system stays within safe limits by satisfying constraints. This dynamic switching minimizes computational overhead and optimizes control actions.

By employing reachability analysis based on the viability region, the NN controller generates control inputs only when necessary, reducing interventions and conserving resources. This balance between robustness and efficiency ensures safe and effective system operation.

In summary, the controller combines the robustness of MPC with the efficiency of the NN controller, maintaining safety near critical boundaries while reducing computational load for routine tasks. The controller adapts to dynamic environments through role switching and minimizes unnecessary actions via reachability analysis, ensuring robust, efficient, and adaptable performance across varying conditions.

2.7.2 Structural Overview of the Controller Approach in the Multi-Agent System

This subsection details the integration of our approach into the multi-agent system. The overall structure of the approach is depicted in block diagrams shown in Figure 2.5.

To utilize the neural network controller within the multi-agent system, it must first be trained. Training requires input values that include the safe set of the system $\mathcal{X}$, the viability kernel V_x (which defines the viability region and is specified in (2.12)), and the control input set $\mathcal{U}$, which represents the input constraints of the system.

The neural network, tailored to the multi-agent system and its objectives, is trained with these inputs to develop the control policy for the neural network controller, denoted as $\pi_U(\mathbf{y}(t))$. Here, $\mathbf{y}(t) = [y_1(t), \ldots, y_N(t)]$ represents the output values obtained from the observation model, which aligns with the multi-agent system model. The multi-agent system is assumed to have N nodes.

Once the policy is trained, the neural network controller is ready to be deployed alongside the MPC within the central controller. The system operates with initial values for each agent, $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]$, and generates output value $y^*(t)$ based on control inputs $u^*(t)$, which are determined by the central controller. This setup allows us to compute the optimal path for each node, ensuring it remains within the boundary functions ($\underline{b}(t)$ as the minimum boundary function and $\overline{b}(t)$ as the maximum boundary function).

3. COMMUNICATION PROTOCOL

This chapter will explain how communication is handled in a described communication network system. According to the model of the system given in the previous chapter, the communication is divided into two parts: *1)* The sporadic communication between the node and its corresponding controller, and *2)* the communication between nodes. Both of these communications help nodes to adapt to the system. However, both communications can bring a cost into the system, which brings inefficiency, even though the system is controlled perfectly. Therefore, some limitations will be included in the communication between nodes. The communication between the node and the corresponding controller will be operated according to the control signals, which will be clarified deeply in this chapter. Furthermore, the way communication is operated between nodes will be clarified with its requirements and limitations. The definition of limitations will lead to designing a protocol policy, which will also be mentioned in this chapter. After all of these explanations, these communications will be expressed as algorithms.

3.1 The Communication of A Node and Its Corresponding Controller

Before explaining the communication between the node and the corresponding controller, a structure including the dynamics of the node, controller, and other physical devices will be illustrated as a model, and this model will be clarified. After the model is explained, the communication between the node and its corresponding controller.

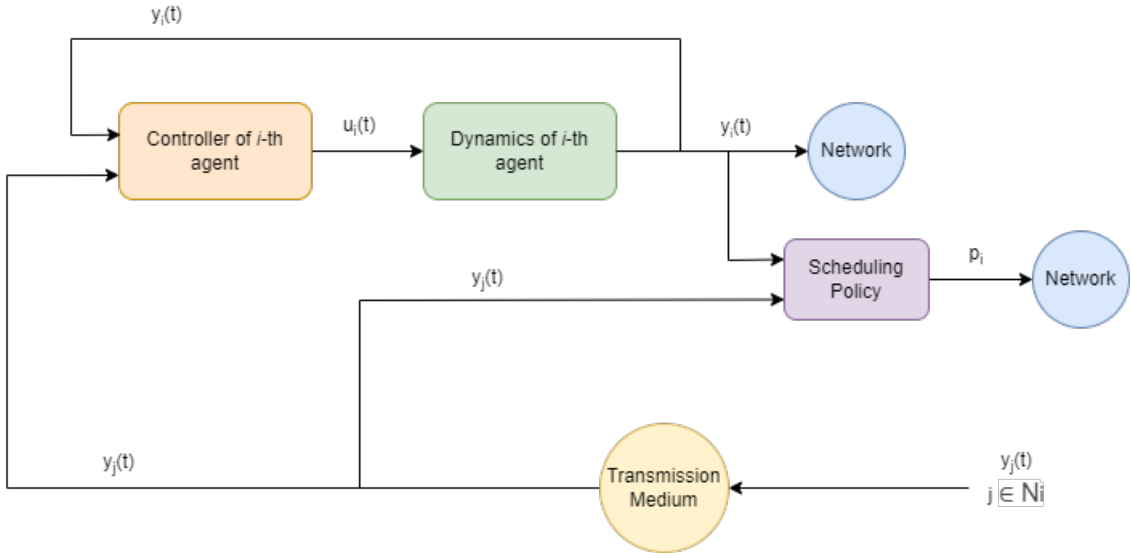


Figure 3.1 Sub-Model with The Node and Its Corresponding Controller

3.1.1 Sub-Model of The System

This model is considered as a sub-model because it illustrates what happens when the input enters the node, and the output of this sub-model is the output of the node. This sub-model is shown in Figure 3.1. In this figure, the input comes from the neighbor of the node, and it enters into the controller of the corresponding agent along with the output of the dynamics of the agent. The control input is then entered into the dynamics for evaluation. The output of dynamics is the output of the node, and it is also sent to the corresponding scheduling policy when the priority decision time has come. The scheduling policy will be explained later.

The "Transmission Medium" block included in Figure 3.1 means that the data coming from the neighbor of the node is passing from the transmission medium. This means that data can be affected by the transmission medium, e.g., transmission rate, interference, distance between nodes. The "Network" block(s) is included in the model to display that the output of the node is sent to the communication network model, i.e., its neighbor(s) connected with the transmission link.

3.1.2 Control Signal Transmission

The essential part of the sub-model is communication between the controller of the corresponding node (agent) and the node (agent) itself. The controller decides

whether communication is required and checks whether the corresponding node needs to be controlled before making a decision.

As a recall from Chapter 2, the controller was constructed with the control barrier function, and it was defined in (2.23). In the function, the controller decides the control signal value by checking the Lipschitz continuity of the control barrier function for the viability kernel. Assuming that the control barrier function satisfies all other conditions, given in Chapter 2, if the Lipschitz continuity is also satisfied, the control signal becomes zero. This means that the corresponding agent does not need help from the controller at the time. Therefore, the controller decides not to send a signal to the node, and the node continues evaluating its dynamic system. If, on the other hand, the Lipschitz continuity is not satisfied, the controller calculates the control signal to help the corresponding node satisfy the Lipschitz continuity condition. After that, the controller sends the control signal value to the corresponding node at the time, so the node continues to evaluate the dynamical system with the help of the control signal at the next time step. When the agent is in the viability region, a trained neural network controller uses the region to check whether the agent can reach the target in a finite time. If it can't, the neural network controller sends the control input provided with the input of the controller to the agent. Otherwise, the neural network controller will not send anything, and the agent will continue evaluating its dynamic system without the assistance of the controller.

The system starts by transmitting the initial control signal values to the corresponding node. The transmission of the control signal to the node takes the same time as the data transmission between nodes, i.e., its transmission period is represented as $\tau_{t_c} = \tau_t$. The node starts to evaluate its dynamical system when it gets the initial control signal. While the node sends its output to its neighbor nodes, it also sends it to the controller. If the controller decides to send the control signal to the corresponding node, it sends the new decision in the same transmission period. The node waits τ_{t_c} seconds for the control input value. If it does not receive control input, it continues evaluating its dynamical system for the next time period. If it receives the control input value, then the node begins evaluation with the new control update.

In the following chapters, the results of the communication between the controller and its corresponding node will be shown and then analyzed. The aim is to decrease the controller requirement of the node, and the proposed controller will achieve a lower communication cost in this part. It will be explained in detail in the following chapter.

3.2 The Communication Between Nodes

The second part will explain the communication protocol between nodes. It was assumed that all nodes can communicate with only one neighbor node. This limitation was included to decrease the communication cost in this part. Nodes transmit their outputs, i.e., the evaluated state of the dynamical system, to the prior neighbor node. A scheduling policy decides the priority. The network topology model, the transmission, and its physical layer requirements will be explained in this part. In addition, the design of the scheduling policy will be clarified.

3.2.1 Network Topology Model

Before explaining the protocol, the first thing to do is to define the network topology model. In Chapter 2, the network model for each node was described as a differential equation, and the communication interaction was briefly explained. There are some communication limits defined in the network model. Here, these limitations are going to be clarified.

As a reminder, a multi-agent system composed of N agents was considered. The same multi-agent system is going to be considered here too. Graph theory was used to describe the information interaction $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and this directed graph is utilized for the topology model. In this directed graph, a node set was defined as $\mathcal{V}$, and an edge set $\mathcal{E}$, defining the data flow existence between nodes. With these set definitions, a new set definition is identified here. The neighbor set of each node is denoted as $\mathcal{N}_i$, $i \in \mathcal{V}$, and it is defined as the number of nodes connected to node i. The neighbor set is identified as a mathematical set in the following.

$$\mathcal{N}_i = \{j \in \mathcal{V} \mid (j,i) \in \mathcal{E}\}; \quad i \in \mathcal{V} \tag{3.1}$$

The network topology model can be illustrated as Figure 3.2, showing nodes and edges of a communication network model included in a real-world system. Figure 3.2 is a general example of a network topology model. The model could be in any shape. We here want to show that our proposed controller is suitable for all types of network topology models. The model is given here to display and make the protocol understandable. This suitability will be proven in the following chapter by providing

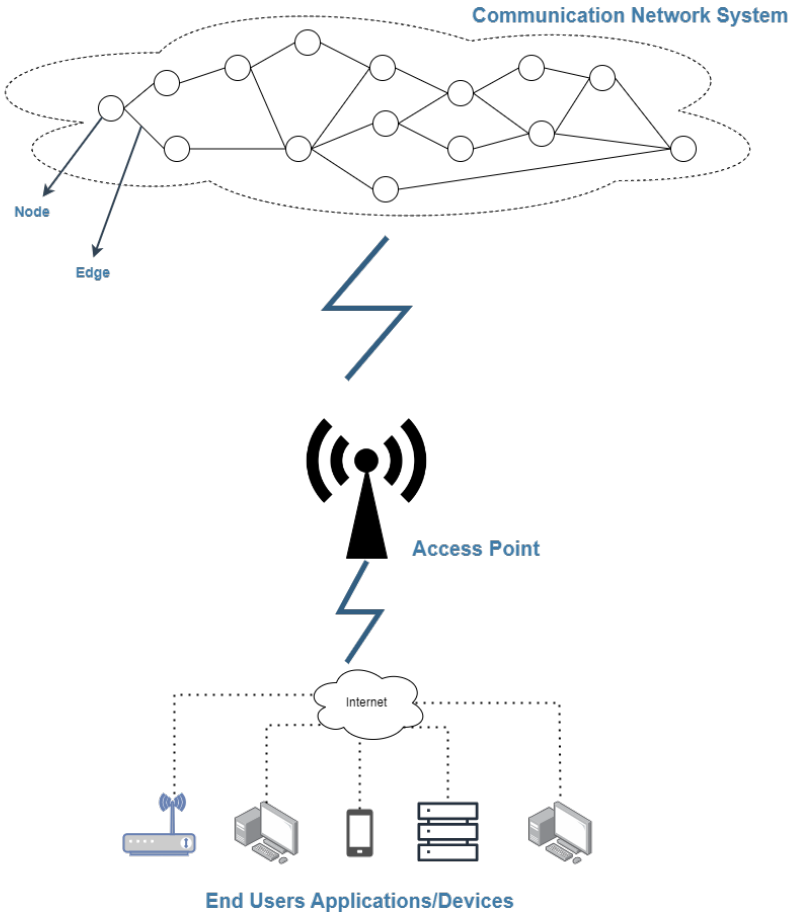


Figure 3.2 The Communication Network Model in A Real-World System

two different scenarios with different topology models.

In Figure 3.2, the communication network system receives data from the access point, which also collects data from devices. The network system here aims to adapt all nodes, which may all behave differently or the same, according to the upcoming data coming from devices. Some examples from the real world could be as follows. *1)* Intelligent transportation systems (ITS) enable vehicles to communicate for collision avoidance, traffic management, and navigation, adapting transmission power and channels and prioritizing messages based on real-time traffic conditions and safety requirements. *2)* Wireless sensor networks are low-power sensors used in environmental monitoring, industrial automation, and healthcare, requiring dynamic adjustments to conserve energy, overcome interference, and maintain connectivity in changing environmental conditions. *3)* Dynamic adaptation is crucial for optimal performance under changing link conditions, using techniques like adaptive modulation and coding in satellites, which offer communication services for broadcasting, telecommunication, and remote sensing.

The adaptation of all nodes to the environment of the network could have been done

perfectly if all nodes sent data to all of their neighbor nodes. However, this can bring network congestion or interference, causing packet loss, delays, and reduced data rates. To avoid these costs, we will limit the allowance of data transmission between nodes. For that, a scheduling policy is designed, and this design will be mentioned later. Before this, the way how data transmission happens between nodes and the physical layer requirements for a described transmission rate will be explained.

3.2.2 Data Transmission Between Nodes

At $t = 0$, all nodes start to transmit their initial output y_i, $i \in \mathcal{V}$ to their neighbor nodes. Since nodes are allowed to receive data from only one neighbor node, an initial priority of the neighbor node is selected randomly. In addition, the initial controller is initialized to solve the differential equation.

While the transmission is happening, all nodes begin calculating the output of their dynamics for the next period of transmission and also check whether the state of the node is safe or not. In other words, all nodes try to ensure that they are viable and render to keep viable by their control update. If they are already viable, control will not be updated (it will be zero). The way the controller is updated was already mentioned in Chapter 2; it was defined as a function. It is important to note that one of the dependants of the control update is the output of the neighbor node. The CBF function $h : \mathbb{R}^n \to \mathbb{R}$ is designed according to the output of the neighbor node and the node itself.

After the control update is done, nodes send their new output to their neighbor node. With the latest transmission of nodes, the next period is started. We can define the data transmission period as τ_t. It is assumed that data transmission and the solution of dynamical systems of nodes are done at the same time within the period τ_t. Every period τ_t, the system is trying to keep viable under limited communications throughout time.

To explain this in detail, let's consider that the system tries to keep viable between the interval $t \in [0, T]$, where T is the finite time. Then, the whole process is illustrated in Figure 3.3. In this figure, the beginning of the period τ_t means that output is received from the neighbor node, and the evaluation of differential equations of each node is started. The end of the period τ_t means that the control decision is made, and the new output of the node is ready to be sent.

Since there are communication limits, nodes are allowed to receive from only one

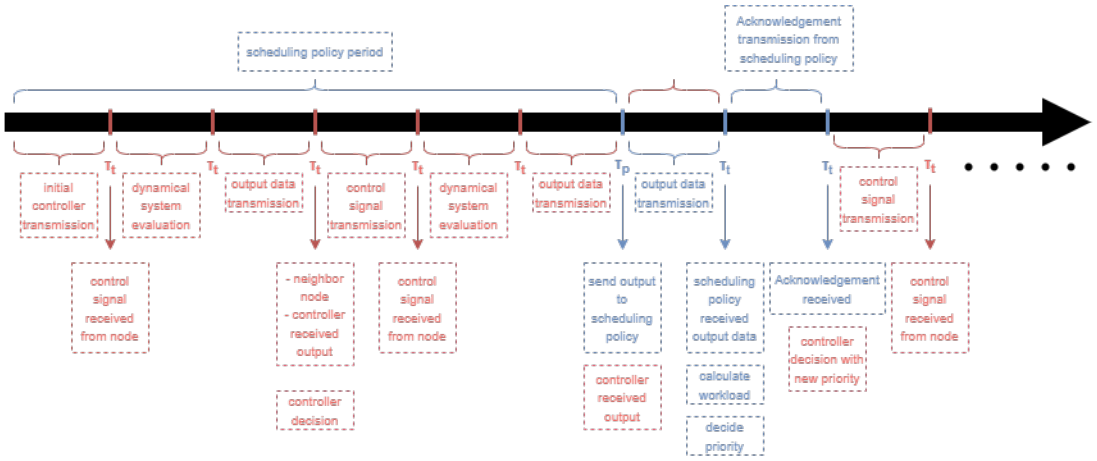


Figure 3.3 The Complete Transmission Process Between Nodes

of the neighbor nodes. Although this decreases channel congestion, there are some requirements for deciding the priority between neighbor nodes. The scheduling policy, which will be mentioned in the next section, decides the priority between nodes.

3.2.3 Scheduling Policy

For this protocol, the scheduling policy is designed for the priority decision between neighbor nodes $\mathcal{N}_i$. If the number of neighbor nodes is one, then the priority decision is not necessary. In other words, this scheduling policy is designed for nodes that have more than one neighbor node.

The priority decision is made periodically, and the period is denoted as τ_p. It was illustrated in the transmission process diagram in Figure 3.3. It is assumed that τ_p hits after an output value is transmitted. Although $\tau_p > \tau_t$, τ_p can be set after any process or during the transmission of output data. But right now, the former assumption will be made to describe how the scheduling policy works.

The scheduling policy requires data from all neighbor nodes, with each having its own scheduling policy. Data transmission takes τ_t seconds, and the scheduling policy calculates the communication workload by calculating the difference between the output and input values. The mathematical representation of the communication workload is provided.

$$\mathcal{W}_{j\to i} = y_i - y_j; \qquad i \in \mathcal{V} \tag{3.2}$$

The scheduling policy calculates communication workloads for neighbor nodes and chooses the one with the minimum overload, making it the priority node, accepting data from the prior neighbor node, ρ_i.

$$\rho_i = \min_{j \in \mathcal{N}_i} |\mathcal{W}_{j \to i}| = \min_{j \in \mathcal{N}_i} |y_i - y_j|; \qquad i \in \mathcal{V} \tag{3.3}$$

Neighbor nodes must determine data acceptance or decline, as continuous transmission without prior knowledge can lead to system non-viability if all declined packages are dropped. To avoid non-viability, the node sends acknowledgments to neighbor nodes to avoid priority decision delays. Neighbor nodes stop and wait for acknowledgment at the priority decision period τ_p. When the scheduling policy is active, the dynamical system and its controller is passive until the acknowledgement is received.

3.3 Algorithm of The Communication Protocol

Algorithm 2 explains the communication between nodes. Parameters needed for the communication between nodes are the number of nodes N, the finite time T, the transmission period τ_t, the scheduling policy period τ_p, and the operator input $u_h(t)$. The algorithm requires the node set $\mathcal{V}$, the edge set $\mathcal{E}$, the control input set $\mathcal{U}$, and the neighbor node set $\mathcal{N}_i$. The algorithm initializes with the initial output value $y_i(0)$ of nodes n_i and neighbor nodes n_j, and the initial control input $v(u, u_h)$ is transmitted at time $t = 0$. The prior neighbor node ρ_i is also set for initialization. After the evaluation of the dynamical system f_c, the output data is transmitted to the neighbor node. This process is repeated at every transmission time τ_t. When the time hits the scheduling policy period τ_p, all neighbor nodes n_j transmit output data to obtain the priority decision. After ρ_i is set, n_i sends an acknowledgment to neighbor nodes n_j. The parameter d is there for updating the value of scheduling policy period τ_p. It is initialized as 1 before entering the transmission period for loop, and it is increased every time the process enters the if scheduling policy period case.

Algorithm 2 Algorithm of Communication Between Nodes

Parameters: N, T, V_x τ_t, τ_p, $u_h(t)$
Required: $\mathcal{V}$, $\mathcal{E}$, $\mathcal{U}$ $\mathcal{N}_i$ $(i \in \mathcal{V})$
$y_i(0) \leftarrow V_x$, $n_i \leftarrow \mathcal{V}$, $n_j \leftarrow \mathcal{N}_i$, $\rho_i \leftarrow \mathcal{N}_i$ $(i \in \mathcal{V})$
$d = 1$
$v_i(u, u_h)$ transmission
for $t = \{0, \tau_t, 2\tau_t, ..., T/\tau_t\}$ **do**
  Evaluate $f_c(t, \overline{x}_i, u, u_h(t))$ and get $y_i(t + \tau_t)$
  n_i send $y_i(t + \tau_t)$
  if t==τ_p **then**
    All n_j sends y_j to n_i
    $t \leftarrow t + \tau_t$
    n_i receives y_j
    $\mathcal{W}_{j \to i} \leftarrow$ Eqn. (3.2)
    $\rho_i \leftarrow$ Eqn. (3.3)
    Send ACK to all n_j
    $t \leftarrow t + \tau_t$
    $d \leftarrow d + 1$
    $\tau_p \leftarrow d\tau_p$
  end if
end for

Algorithm 3 gives the process of communication between the node and its corresponding controller. The parameters of this algorithm are the number of nodes N, the finite time T, the transmission period τ_t, and the human operator input function $u_h(t)$. The requirements for this algorithm are the node set $\mathcal{V}$ and the control input set $\mathcal{U}$. Similar to Algorithm 2, Algorithm 3 begins with initial control input $v_i(u, u_h)$, and the initial output values $y_i(0)$ of each n_i. As soon as n_i receives the initial control input, n_i evaluates the dynamical system f_c to obtain the next output value $y_i(t + \tau_t)$. The output data is transmitted to the controller to calculate the control input $v(u, u_h)$. To do that, n_i first calculates the control barrier function $h(x)$, the lie-derivative functions $\mathcal{L}_f h$, $\mathcal{L}_g h$, and the class K function $\alpha(\cdot)$, which were all defined in Chapter 2. If the control input is zero, the controller does not transmit any value to the dynamical system of n_i. Otherwise, the control input transmission happens to assist n_i in reaching and staying viable. The next period starts with checking whether the control input is received or not. If the control input is received, then the dynamical system evaluation starts with the control input value. If, otherwise, no control input is received, the node begins the evaluation, assuming that the control input is zero.

Both these communications are explained in this chapter and summarized by giving algorithms. This communication protocol will be used to get numerical results, analyze them, and demonstrate that this protocol assists the designed controller in

Algorithm 3 Algorithm of Communication Between Node and Its Corresponding Controller

Parameters: N, T, V_x τ_t, $u_h(t)$
Required: $\mathcal{V}$, $\mathcal{U}$
$v_i(u,u_h) = u_i(0) + u_h(0) \leftarrow \mathcal{U}$ $(i \in \mathcal{V})$
$y_i(0) \leftarrow V_x$, $n_i \leftarrow \mathcal{V}$
for $t = \{0, \tau_t, 2\tau_t, ..., T/\tau_t\}$ **do**
  if $v_i(u,u_h)$ received **then**
    Start $f_c(t, \overline{x}_i, u, u_h(t))$ with $v_i(u,u_h)$
  else
    Start $f_c(t, \overline{x}_i, u, u_h(t))$ with $v_i(u,u_h) = 0$
  end if
  Evaluate $f_c(t, \overline{x}_i, u, u_h(t))$ and get $y_i(t+\tau_t)$
  n_j receives $y_i(t+\tau_t)$
  Calculate $h(x), \mathcal{L}_f h, \mathcal{L}_g h, \alpha(h(x))$
  if $(\mathcal{L}_f h + \mathcal{L}_g h \cdot u_h) < -\alpha(h(x))$ **then**
    $v_i(u,u_h) = k(x,t) \leftarrow$ Eqn. (2.23)
  else
    $v_i(u,u_h) = \pi_i(y_i(t))$
  end if
  if $v_i(u,u_h) == 0$ **then**
    Don't transmit $v_i(u,u_h)$
  else
    $v_i(u,u_h)$ transmission
    $t \leftarrow t + \tau_t$
  end if
end for

obtaining efficient communication. All of these will be clarified in the next chapter.

4. EXPERIMENTAL SETUP

This section will illustrate two scenarios of a communication network system. The aim of both of them is to synchronize with a desired signal or the behavior of a node. These two scenarios' network topology models are chosen to be different from each other because it is aimed to demonstrate that the proposed method is successful in any system. Particularly, the main failure of these scenarios can be defined as the synchronization failure of the nodes, and our proposed controller will treat this failure as a rule, making local decisions accordingly. Therefore, these two scenarios are suitable for addressing the issues. The system definitions will be done and the calculations will be applied accordingly. After that, the simulation results will be shown, and it will be analyzed. Finally, this method will be compared with another control design method, and the efficiency of communication with given limitations will be discussed.

4.1 System Definition

4.1.1 Scenario 1: A System with Identical Nodes

The first scenario is inspired by [75]. This scenario is convenient to the given problems in previous chapters because this scenario is susceptible to the synchronization failure of any node, and our proposed controller is aimed to handle this failure as a norm instead of considering it as an abnormal situation. Consider a multi-agent system equation which is defined as follows:

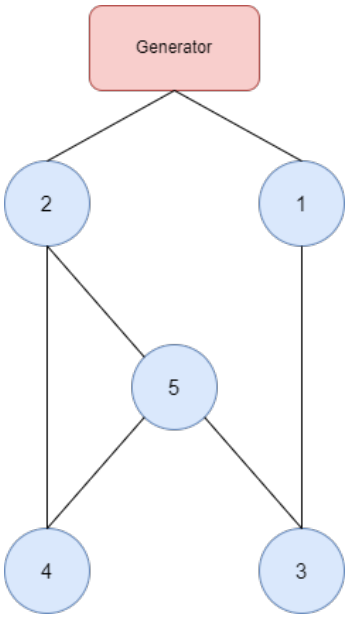


Figure 4.1 Communication Topology of MAS with Identical Nodes

Table 4.1 Table of Functions for All Nodes in Scenario 1

Functions	Node i $(i = 1, ..., 5)$
$f_i(x_i)$	$[x_{i,2} + x_{i,3} \quad x_{i,2} \quad 2x_{i,1}x_{i,2} - 2x_{i,3}\|x_{i,3}\|]$
$g_i(x_i)$	$[0 \quad 0 \quad 5]$
$d_i(t)$	$[0 \quad 0 \quad 0]$

$$\begin{cases} \dot{x}_{i,1} = \dot{x}_{i,2} + \dot{x}_{i,3} \\ \dot{x}_{i,2} = \dot{x}_{i,3} \\ \dot{x}_{i,3} = 2x_{i,1}x_{i,2} - 2x_{i,3}|x_{i,3}| + 5(u_i + u_h(t)) \\ y_i = x_{i,1} \end{cases}, \tag{4.1}$$

where $i = 1, ..., 5$, $x_i = [x_{i,1}, x_{i,2}, x_{i,3}]$ is a three-dimensional state vector of node i, y_i is the output of the node i, which is sent to the connected neighbor node, u_i is the controller of node i, and $u_h(t) = 1 - 0.5cos(\pi t/5)$ is the operator input. The communication topology of the muti-agent system is described in Figure 4.1.

Functions are distributed according to the defined multi-agent system in this scenario, and it is shown in Table 4.1. As a recall, we defined the heterogeneous nonlinear input-constrained MAS in (2.9). In there, we defined the unmodeled dynamics $f_i(x_i)$, the function interacted with the control input $g_i(x_i)$, and the external disturbance $d_i(t)$. According to (2.9), we extracted the former functions from (4.1), and Table 4.1 shows the function vectors since the dynamics of each node are three-dimensional.

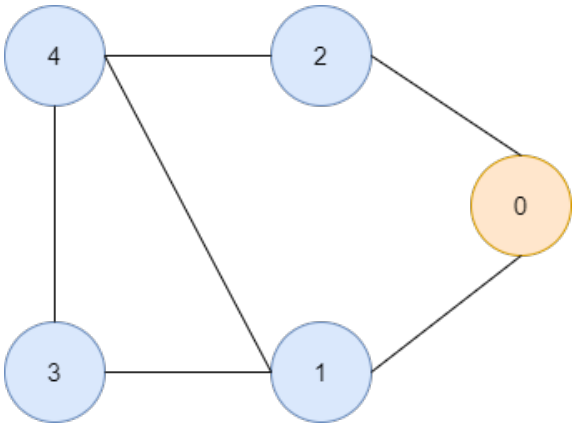


Figure 4.2 Communication Topology of MAS with Non-Identical Nodes (Scenario 2)

In this scenario, the unit step signal is used as the desired signal, and all nodes are aimed to track the desired signal. For the first scenario, two viability kernels will be defined. Its boundaries 1) will stay constant and 2) will change in time.

4.1.2 Scenario 2: A System with Non-Identical Nodes

The second scenario is inspired by [76]. This scenario is applicable to the use of our proposed controller because some effects, like noise or the act of priority mechanism, may lead the system to failure, and our proposed controller will handle these errors caused by apparent changes. In this scenario, all nodes are behaving differently from each other, and, there exists an external disturbance for each of them. The communication topology of this scenario is shown in Figure 4.2. The multi-agent system for this scenario is shown below.

$$\begin{cases} \dot{x}_{1,1} = x_{1,2} + \cos(x_{1,1}) + 0.1\cos(0.2t) \\ \dot{x}_{1,2} = (u_1 + u_h(t)) - \tanh(x_{1,1})x_{1,2} + 0.2\cos(0.2t)\ , \\ \ \ y_1 = x_{1,1} \end{cases} \tag{4.2a}$$

$$\begin{cases} \dot{x}_{2,1} = x_{2,2} + 0.5\sin(x_{2,1}) + 0.1\cos(0.2t) \\ \dot{x}_{2,2} = (u_2 + u_h(t)) - \tanh(x_{2,1})|x_{2,2}|x_{2,2} + 0.15\cos(0.1t)\ , \\ \ \ y_2 = x_{2,1} \end{cases} \tag{4.2b}$$

Table 4.2 Table of Functions in Node 1 and 2

Functions	Node 1	Node 2
$f_i(x_i)$	$[x_{1,2}+cos(x_{1,1}) \quad -tanh(x_{1,1})x_{1,2}]$	$[x_{2,2}+0.5sin(x_{2,1}) \quad -tanh(x_{2,1})\|x_{2,2}\|x_{2,2}]$
$g_i(x_i)$	1	1
$d_i(t)$	$[0.1cos(0.2t) \;\; 0.2cos(0.2t)]$	$[0.1cos(0.2t) \;\; 0.15cos(0.1t)]$

Table 4.3 Table of Functions in Node 3 and 4

Functions	Node 3	Node 4
$f_i(x_i)$	$[x_{3,2}+sin(x_{3,1})cos(x_{3,1})$ $x_{3,1}cos(x_{3,2})\|x_{3,2}\|]$	$[x_{4,2}+x_{4,1}cos(x_{4,1})$ $-x_{4,1}sin(x_{4,2})cos(x_{4,2})]$
$g_i(x_i)$	1	1
$d_i(t)$	$[0.1cos(0.2t) \;\; 0.1cos(0.1t)]$	$[0.1cos(0.2t) \;\; 0.1cos(0.1t)]$

$$\begin{cases} \dot{x}_{3,1} = x_{3,2} + \sin(x_{3,1})\cos(x_{3,1}) + 0.1\cos(0.2t) \\ \dot{x}_{3,2} = (u_3 + u_h(t)) + x_{3,1}\cos(x_{3,2})|x_{3,2}| + 0.1\cos(0.1t) \\ y_3 = x_{3,1} \end{cases}, \tag{4.2c}$$

$$\begin{cases} \dot{x}_{4,1} = x_{4,2} + x_{4,1}\cos(x_{4,1}) + 0.1\cos(0.2t) \\ \dot{x}_{4,2} = (u_4 + u_h(t)) - x_{4,1}\sin(x_{4,2})\cos(x_{4,2}) + 0.1\cos(0.1t) \\ y_4 = x_{4,1} \end{cases}, \tag{4.2d}$$

where $x_i = [x_{i,1}, x_{i,2}]$, y_i is a two-dimensional state vector and output of node $i = 1, \ldots, 4$ respectively, u_i is the controller of node i, and $u_h(t) = 1 - 0.5cos(\pi t/20)$ is the supervisory control. According to the system definition given in the previous chapter, the components of each equation can be denoted in Tables 4.2 and 4.3. The unmodeled dynamics $f_i(x_i)$, the function interacted with the control input $g_i(t)$ and the external disturbance $d_i(t)$ is extracted from (4.2) according to MAS definition (2.9).

These functions written in the table will be used for obtaining the controller function. They are needed for calculating parameters included in the controller function, as the controller (2.23), lie-derivatives (2.2), and functions (2.24) were defined in Chapter 2. The system topology for this scenario is shown in Figure 4.2.

For this scenario, these nodes are aimed to track node 0, which can be considered as a leader in the system, and its differential equation is defined as follows.

$$\begin{cases} \dot{x}_{0,1} = x_{0,2} + 0.5sin(x_{0,1}) \\ \dot{x}_{0,2} = cos(x_{0,2})x_{0,1} + u_0(t) \end{cases}, \tag{4.3}$$

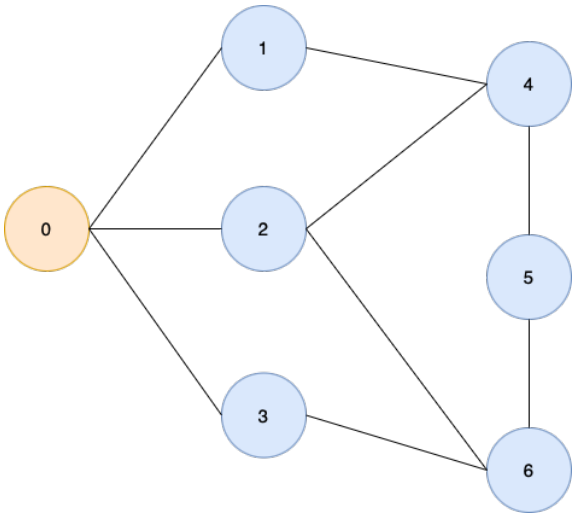


Figure 4.3 Communication Topology of MAS with Non-Identical Nodes (Scenario 3)

Where $u_0(t) = -10\,(x_{0,2} + 5\,(x_{0,1} - 2sin(\pi t/6)))$ is a time-varying dynamic input of the leader, which is unknown to all other nodes.

4.1.3 Scenario 3: A Second System with Non-Identical Nodes

The third scenario is inspired by [77]. It is similar to the second scenario, but the topology structure and dynamics of each agent are different. The topology is given in Figure 4.3, and the multi-agent system for this scenario is given below.

$$\begin{cases} \dot{x}_{1,1} = x_{1,2} + \cos(x_{1,1}) + 0.5\cos(0.3t) \\ \dot{x}_{1,2} = (u_1 + u_h(t)) + x_{1,1}\sin(x_{1,2}) + 0.1\cos(0.15t)\,, \\ y_1 = x_{1,1} \end{cases} \tag{4.4a}$$

$$\begin{cases} \dot{x}_{2,1} = x_{2,2} + 0.1x_{2,1} - 0.5 + 0.1\cos(0.4t) \\ \dot{x}_{2,2} = (u_2 + u_h(t)) + 0.6x_{2,2} - 0.1\cos(x_{2,1}) + 0.15\cos(0.2t)\,, \\ y_2 = x_{2,1} \end{cases} \tag{4.4b}$$

$$\begin{cases} \dot{x}_{3,1} = x_{3,2} + 0.5 - x_{3,1}^3 + 0.3\cos(0.25t) \\ \dot{x}_{3,2} = (u_3 + u_h(t)) + 1 - 0.37\cos(x_{3,1}) + x_{3,2}^2 + 0.1\cos(0.1t)\,, \\ y_3 = x_{3,1} \end{cases} \tag{4.4c}$$

Table 4.4 Table of Functions in Node 1, Node 2 and Node 3

Functions	Node 1	Node 2	Node 3
$f_i(x_i)$	$[x_{1,2}+cos(x_{1,1})$ $x_{1,1}sin(x_{1,2})]$	$[x_{2,2}+0.1x_{2,1}-0.5$ $0.6x_{2,2}-0.1cos(x_{2,1})]$	$\left[x_{3,2}+0.5-x_{3,1}^3\right.$ $\left.1-0.37cos(x_{3,1})+x_{3,2}^2\right]$
$g_i(x_i)$	1	1	1
$d_i(t)$	$[0.5cos(0.3t)$ $0.1cos(0.15t)]$	$[0.1cos(0.4t)$ $0.15cos(0.2t)]$	$[0.3cos(0.25t)$ $0.1cos(0.1t)]$

Table 4.5 Table of Functions in Node 4, Node 5 and Node 6

Functions	Node 4	Node 5	Node 6
$f_i(x_i)$	$\left[x_{4,2}+1-x_{4,1}^2\right.$ $\left.3.15-0.1x_{4,1}e^{-0.0001x_{4,1}}\right]$	$[x_{5,2}+0.2cos(x_{5,1})$ $0.3cos(x_{5,1})+0.15sin(x_{5,2})]$	$[x_{6,2}+x_{6,1}+0.3cos(x_{6,1})$ $1-0.5sin(x_{6,2})cos(x_{6,1})]$
$g_i(x_i)$	1	1	1
$d_i(t)$	$[0.1cos(0.15t)$ $0.6cos(0.45t)]$	$[0.1cos(0.3t)$ $0.5cos(0.15t)]$	$[0.3cos(0.2t)$ $0.1cos(0.1t)]$

$$\begin{cases} \dot{x}_{4,1} = x_{4,2}+1-x_{4,1}^2+0.1\cos(0.15t) \\ \dot{x}_{4,2} = (u_4+u_h(t))+3.15-0.1x_{4,1}e^{-0.0001x_{4,2}}+0.6\cos(0.45t)\,, \\ y_4 = x_{4,1} \end{cases} \tag{4.4d}$$

$$\begin{cases} \dot{x}_{5,1} = x_{5,2}+0.2\cos(x_{5,1})+0.1\cos(0.3t) \\ \dot{x}_{5,2} = (u_5+u_h(t))+0.3\cos(x_{5,1})+0.15\sin(x_{5,2})+0.5\cos(0.15t)\,, \\ y_5 = x_{5,1} \end{cases} \tag{4.4e}$$

$$\begin{cases} \dot{x}_{6,1} = x_{6,2}+x_{6,1}+0.3\cos(x_{6,1})+0.3\cos(0.2t) \\ \dot{x}_{6,2} = (u_6+u_h(t))+1-0.5\sin(x_{6,2})\cos(x_{6,1})+0.1\cos(0.1t)\,, \\ y_6 = x_{6,1} \end{cases} \tag{4.4f}$$

where $x_i = [x_{i,1}, x_{i,2}]$, y_i is a two-dimensional state vector and output of node $i = 1, \cdot, 6$ respectively, u_i is the controller of node i, and $u_h(t) = 1-0.5cos(\pi t/20)$ is the supervisory control. The functions for this scenario can be denoted in Tables 4.4 and 4.5. The unmodeled dynamics $f_i(t)$, the function interacted with control input $g_i(t)$ and the external disturbance $d_i(t)$ is extracted from (4.4) according to MAS definition (2.9).

This scenario also aims to track down node 0, as it is the leader of the system, and its dynamics is defined as a time-varying equation, as shown below.

$$x_0(t) = -e^{-0.1t}(sin(t) - cos(t)), \tag{4.5}$$

Viability kernel and CBF definitions will be done for these three scenarios accordingly in the following sections.

4.2 Viability Kernel Definition

All nodes in the system aim to track down a desired signal or a node (leader) included in the system. Accordingly, a tracking error is going to be defined. Since not all nodes have a direct connection to the desired signal or the leader node, the tracking error cannot be handled directly. Therefore, the tracking error becomes local and its calculation is done by using only its neighbor nodes.

$$z_i = \sum_{j \in \mathcal{N}_i} a_{ij}(y_i - y_j) + b_i \widetilde{y}_i; \qquad i = 1, ..., 4 \tag{4.6}$$

where a_{ij} comes from the adjacency matrix, which is defined in Section 2.2.1, $\widetilde{y}_i = y_i - y_0$ is the tracking error between leader node/desired signal and node i, and b_i is the relationship between node i and leader node/desired signal. However, in calculating tracking error, all nodes can transmit their output to their neighbor nodes. Limitations will be considered in this system to analyze the communication cost.

Assume that nodes can receive data from only one node. In such case, the tracking error is updated as follows.

$$z_i = a_{ij}(y_i - y_j) + b_i \widetilde{y}_i; \qquad j \in \mathcal{N}_i, \quad i = 1, ..., 4 \tag{4.7}$$

To achieve the synchronization, error constraints will be identified. Two types of error constraints will be identified: 1) symmetric and 2) asymmetric error constraints.

The symmetric error constraints in (4.8a) do not change in time, and the ranges are assumed to be the same throughout time. On the other hand, asymmetric error constraints in (4.8b) are time-varying, i.e., ranges change throughout time:

Table 4.6 Boundary Functions of Each Node for Scenario 2

Upper & Lower Bounds	Node 1	Node 2	Node 3	Node 4
$\underline{p}_i(t)$	$5e^{-t}+\underline{c}_1$	$5e^{-t}+\underline{c}_2$	$5e^{-t}+\underline{c}_3$	$5e^{-t}+\underline{c}_4$
$\overline{p}_i(t)$	$4e^{-1.2t}+\overline{c}_1$	$4e^{-1.2t}+\overline{c}_2$	$4e^{-t}+\overline{c}_3$	$4e^{-t}+\overline{c}_4$

Table 4.7 Boundary Functions of Nodes 1, 2, and 3 for Scenario 3

Upper & Lower Bounds	Node 1	Node 2	Node 3
$\underline{p}_i(t)$	$1.2e^{-0.6t}+\underline{c}_1$	$1.5e^{-0.7t}+\underline{c}_2$	$1.7e^{-0.8t}+\underline{c}_3$
$\overline{p}_i(t)$	$3.1e^{-1.2t}+\overline{c}_1$	$2.7e^{-1.2t}+\overline{c}_2$	$2.4e^{-t}+\overline{c}_3$

$$-p_i < z_i(t) < p_i, \tag{4.8a}$$

$$-\underline{p}_i(t) < z_i(t) < \overline{p}_i(t), \tag{4.8b}$$

where p_i is the bound value of node i, $\underline{p}_i(t)$ and $\overline{p}_i(t)$ are time-varying minimum and maximum boundary functions of node i respectively.

Dynamical behaviors of each node are the same in the first scenario, and both error constraints are going to be considered separately. For symmetric constraints, the bound values for all nodes are going to be the same, and $p_i = 0.9$ is chosen as an example range value to demonstrate that our method is successful in a system with symmetric constraints. For asymmetric error constraints, minimum and maximum functions are also going to be the same, and they are designed for the first scenario as follows.

$$\begin{cases} \underline{p}_i(t) = 1.195e^{-0.6t} + 0.005 \\ \overline{p}_i(t) = 0.995e^{-t} + 0.005 \end{cases}, \tag{4.9}$$

In the second scenario, all nodes behave differently. Error constraints for each node are different, where $\underline{p}_i(t)$ and $\overline{p}_i(t)$ are the lower bound and upper bound respectively:

In the third scenario, all nodes are behaving differently as well. Their error constraints are defined for each node. They are aimed to be adapted to their own, just like in the second scenario.

Now that error constraints are defined, the viability kernel is going to be defined accordingly. With given symmetric and asymmetric error constraints, viability kernels V_x can be formally derived as follows:

Table 4.8 Boundary Functions of Nodes 4, 5, and 6 for Scenario 3

Upper & Lower Bounds	Node 4	Node 5	Node 6
$\underline{p_i}(t)$	$2.1e^{-0.9t}+\underline{c}_4$	$2.4e^{-t}+\underline{c}_5$	$2.6e^{-t}+\underline{c}_6$
$\overline{p}_i(t)$	$2e^{-t}+\overline{c}_4$	$1.6e^{-t}+\overline{c}_5$	$1.2e^{-t}+\overline{c}_6$

$$V_x = \left\{ x \in \mathbb{R}^3 | \left(p_i + z_i(t)\right)\left(p_i - z_i(t)\right) > 0 \right\}, \tag{4.10a}$$

$$V_x = \left\{ x \in \mathbb{R}^2 | \left(\underline{p_i}(t) + z_i(t)\right)\left(\overline{p}_i(t) - z_i(t)\right) > 0 \right\}, \tag{4.10b}$$

For the viability kernel, the control barrier function should satisfy three conditions. The positivity condition is satisfied as long as the tracking error is between boundaries, as defined in (4.8). To satisfy the compactness condition, the viability kernel is modified, yielding to the control barrier function $h : \mathbb{R}^N \to \mathbb{R}$; $N = 2, 3$. In (4.11), there are two separate CBF functions. (4.11a) is designed with symmetric error constraints, and (4.11b) is designed with asymmetric error constraints.

$$h(x) = \left(p_i + z_i(t)\right)\left(p_i - z_i(t)\right)\frac{1}{1 + K x_{i,N}^2}, \tag{4.11a}$$

$$h(x) = \left(\underline{p_i}(t) + z_i(t)\right)\left(\overline{p}_i(t) - z_i(t)\right)\frac{1}{1 + K x_{i,N}^2}, \tag{4.11b}$$

where $i = 1, ..., N$, $K \in \mathbb{R}_{>0}$ is a positive constant.

The Lipschitz continuity condition is satisfied with the controller function.

The process of obtaining $h(x)$ from V_x is described with psudo-algorithm, as shown in Algorithm 4.

4.3 MPC Controller Using CBF Function

Recall that Lipschitz continuity is satisfied when a continuous extended class K function exists such that the inequality is satisfied.

To obtain the time derivative of the control barrier function, the lie-derivatives of the system should be calculated. Since there are two different control barrier functions, the lie-derivatives are going to be calculated separately.

Algorithm 4 Deriving CBF $h(x)$ from Viability Kernel V_x for Scenarios

Input: V_x, $y_i(t)$, $y_j(t)$, $y_0(t)$, p_i, $\underline{p}_i(t)$, $\overline{p}_i(t)$, K $(i \in \mathcal{V})$, $(j \in \mathcal{N}_i)$
$a_{ij}(y_i - y_j) + b_i(y_i - y_0) \leftarrow z_i$
Define Viability Kernel V_x:
Symmetric: $V_x = \{x \mid (p_i + z_i)(p_i - z_i) > 0\}$
Asymmetric: $V_x = \{x \mid (\underline{p}_i + z_i)(\overline{p}_i - z_i) > 0\}$
Verify conditions of CBF $h(x)$:
Symmetric: Take $h(x)$ as $(p_i + z_i)(p_i - z_i)$
Asymmetric: Take $h(x)$ as $\underline{p}_i + z_i)(\overline{p}_i - z_i)$
Verify conditions of CBF holds for $h(x)$.
Positivity: $h(x) > 0$ is true for error constraints.
Compactness: Modify $h(x)$ by adding $\frac{1}{1+Kx_{i,N}^2}$
Lipschitz Continuity: $\dot{h}(x) > -\alpha(x)$ satisfied with controller function.
Latest update of $h(x)$:
Symmetric: $h(x) = (p_i + z_i)(p_i - z_i)\frac{1}{1+Kx_{i,N}^2}$
Asymmetric: $h(x) = \underline{p}_i + z_i)(\overline{p}_i - z_i)\frac{1}{1+Kx_{i,N}^2}$

Lie-derivatives for control barrier function (4.11a) are:

$$\begin{aligned}\mathcal{L}_f h = &\left(\frac{(-a_{ij} - b_i)\,(p_i + z_i(t))}{1 + Kx_{i,N}^2} + \frac{(a_{ij} + b_i)\,(p_i - z_i(t))}{1 + Kx_{i,N}^2}\right) \cdot f_{i,1}(x_i) \\ &+ \frac{-2K\,(p_i - z_i(t))\,(p_i + z_i(t))\,x_{i,N}^2}{\left(1 + Kx_{i,N}^2\right)^2} \cdot f_{i,N}(x_i),\end{aligned} \tag{4.12a}$$

$$\mathcal{L}_g h = \frac{-2K\,(p_i - z_i(t))\,(p_i + z_i(t))\,x_{i,N}^2}{\left(1 + Kx_{i,N}^2\right)^2} \cdot g_{i,N}(x_i), \tag{4.12b}$$

Lie-derivatives for control barrier function (4.11b) are:

$$\begin{aligned}\mathcal{L}_f h = &\left(\frac{(-a_{ij} - b_i)\left(\underline{p}_i(t) + z_i(t)\right)}{1 + Kx_{i,N}^2} + \frac{(a_{ij} + b_i)\,(\overline{p}_i(t) - z_i(t))}{1 + Kx_{i,N}^2}\right) \cdot f_{i,1}(x_i) \\ &+ \frac{-2K\,(\overline{p}_i(t) - z_i(t))\left(\underline{p}_i(t) + z_i(t)\right)x_{i,N}^2}{\left(1 + Kx_{i,N}^2\right)^2} \cdot f_{i,N}(x_i),\end{aligned} \tag{4.13a}$$

$$\mathcal{L}_g h = \frac{-2K\,(\overline{p}_i(t) - z_i(t))\left(\underline{p}_i(t) + z_i(t)\right)x_{i,N}^2}{\left(1 + Kx_{i,N}^2\right)^2} \cdot g_{i,N}(x_i), \tag{4.13b}$$

These lie-derivative equations are replaced for the time-derivative of control bar-

rier functions. The time derivative of CBF is calculated for both symmetric and asymmetric error constraints, respectively.

$$\begin{aligned}\dot{h} = &\left(\frac{(-a_{ij}-b_i)\,(p_i+z_i(t))}{1+Kx_{i,N}^2}+\frac{(a_{ij}+b_i)\,(p_i-z_i(t))}{1+Kx_{i,N}^2}\right)\cdot f_{i,1}(x_i)\\ &+\left(\frac{-2K\,(p_i-z_i(t))\,(p_i+z_i(t))\,x_{i,N}^2}{\left(1+Kx_{i,N}^2\right)^2}\cdot g_{i,N}(x_i)\right)\cdot u_h(t),\end{aligned} \tag{4.14a}$$

$$\begin{aligned}\dot{h} = &\left(\frac{(-a_{ij}-b_i)\left(\underline{p}_i(t)+z_i(t)\right)}{1+Kx_{i,N}^2}+\frac{(a_{ij}+b_i)\,(\overline{p}_i(t)-z_i(t))}{1+Kx_{i,N}^2}\right)\cdot f_{i,1}(x_i)\\ &+\left(\frac{-2K\,(\overline{p}_i(t)-z_i(t))\left(\underline{p}_i(t)+z_i(t)\right)x_{i,N}^2}{\left(1+Kx_{i,N}^2\right)^2}\cdot g_{i,N}(x_i)\right)\cdot u_h(t),\end{aligned} \tag{4.14b}$$

Accordingly, we can obtain the controller for both scenarios (4.1) and (4.2) by the controller function defined in Chapter 2. Systems are going to be considered with a single input for all nodes, and the input constraint is given as the following norm constraint, and this is called the control set.

$$\mathcal{U} = \{v \in \mathbb{R} \mid ||v(u, u_h(t))|| \leq a_{max}\}, \tag{4.15}$$

Including this input constraint, the controller is expected to satisfy the input constraint, along with the error constraints.

$$k(x,t) = \begin{cases} -\dfrac{1}{\mathcal{L}_g h}\left(I(x,u_h(t))+\alpha(h(x))\right) & \text{if } I(x,u_h(t)) < -\alpha(h(x)) \text{ and } h(x) > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4.16}$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ is a locally Lipschitz continuous extended class K function and a function $I : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}$ is defined with lie-derivative functions and supervisory control as follows:

$$I(x,u_h(t)) = \left(\frac{(-a_{ij}-b_i)\,(p_i+z_i(t))}{1+Kx_{i,N}^2} + \frac{(a_{ij}+b_i)\,(p_i-z_i(t))}{1+Kx_{i,N}^2} \right) \cdot f_{i,1}(x_i) + \left(\frac{-2K\,(p_i-z_i(t))\,(p_i+z_i(t))\,x_{i,N}^2}{\left(1+Kx_{i,N}^2\right)^2} \cdot g_{i,N}(x_i) \right) \cdot u_h(t), \tag{4.17a}$$

$$I(x,u_h(t)) = \left(\frac{(-a_{ij}-b_i)\left(\underline{p}_i(t)+z_i(t)\right)}{1+Kx_{i,N}^2} + \frac{(a_{ij}+b_i)\,(\overline{p}_i(t)-z_i(t))}{1+Kx_{i,N}^2} \right) \cdot f_{i,1}(x_i) + \left(\frac{-2K\,(\overline{p}_i(t)-z_i(t))\left(\underline{p}_i(t)+z_i(t)\right)x_{i,N}^2}{\left(1+Kx_{i,N}^2\right)^2} \cdot g_{i,N}(x_i) \right) \cdot u_h(t), \tag{4.17b}$$

For $\alpha(\cdot)$, it is designed separately for two scenarios as follows: for non-negative constants $\varpi_1, \varpi_2, \varpi_3 \in \mathbb{R}_{\geq 0}$

$$\alpha(h(x)) = \varpi_1 h(x) + \varpi_2 h^2(x), \tag{4.18a}$$

$$\alpha(h(x)) = \varpi_3 h(x), \tag{4.18b}$$

These continuous functions are mapped as $(a,b) \to \mathbb{R}$, stating that it is continuous. In addition, when taking the derivative of functions, it can be said that these functions are strictly increasing because it is shown that $h(x)$ is always positive under given boundaries, and this tells us that the derivative of $\alpha(\cdot)$ function, which is given below, is positive when constants $a, b \in \mathbb{R}$ of the mapping is assumed to be set according to $h(x)$ conditions.

$$\alpha'(h(x)) = \varpi_1 \dot{h}(x) + \varpi_2 2h(x)\dot{h}(x) > 0, \tag{4.19a}$$

$$\alpha'(h(x)) = \varpi_3 \dot{h}(x) > 0, \tag{4.19b}$$

Lipschitz's continuity condition could not be shown analytically. We demonstrate that it is satisfied by numerical experiments.

4.4 NN Controller and Its Cooperation with MPC

Recall from Section 2.6 that the gains offered by the viability region are used with a neural network controller. The neural network controller with a viability region will be able to determine whether it can reach the target within a finite time.

The input of the neural network model is the state values of each node, which can be considered as the safe set $\mathcal{X}$ of this system. With the state set, the time-varying minimum and maximum boundary functions are also given as input because these boundary functions guarantee safety continuously throughout the time, i.e., it is a representation of viability kernel V_x. Considering the input constraints U, the output of this model is the control input values of each node. The hidden layers consist of Dense layers with ReLU activation function, and the shapes of hidden layers are proportionally designed according to the given network model.

A trained neural network controller cooperates with the MPC controller, as described below. The MPC controller takes over when the state of a node is near the boundaries, and this is decided with the given controller equation. The MPC controller steers the state of the node away from the boundaries. Otherwise, when the safety of the state of the node is not in danger, then it is left to the trained neural network controller. The NN controller checks whether the node can reach the target in a given finite seconds by using the viability region. If the reachability is not successful, then the control input decision is made from the policy of the neural network controller. Otherwise, i.e., if the reachability is successful, then no control input will be generated from the NN controller policy. Whether the control input value is zero or not zero, the dynamics evaluation of the node is made anyway.

5. NUMERICAL RESULTS

Both scenarios are simulated with given calculations. The dynamical system is solved with the high-order Runge-Kutta method (see Appendix B), and other calculations are included with the solutions of the dynamical system.

5.1 Results of Output and System Safety

All of the three scenarios are sensitive to leader synchronization failure, and the controller treats this failure as a norm. The aim of the Model Predictive Controller is that when the synchronization fails, it will steer it to the safe zone to keep all nodes synchronized with the leader. The aim of the Neural Network Controller is to track the aim of the system, which is to follow the leader and achieve continuous synchronization.

The first scenario is simulated with symmetric and asymmetric error constraints, and all constraints are the same for all nodes. For symmetric error constraints, the range is chosen as $p_i = 0.9$, and for asymmetric error constraints, the limit functions are given in (4.9). Constant values are set as $K = 0.1$, $\varpi_3 = 10$ for $\alpha(\cdot)$ function (4.18b).

For the first scenario, the initial states of five agents are given in Table 5.1. Results with symmetric error constraints are shown in Figure 5.1 and Figure 5.2. In Figure 5.1, y_0 is defined as a command generator for the first scenario, and it generates a unit step signal. Although all other nodes cannot reach the values of the command generator, they are all in the viability kernel, as shown in Figure 5.2, so it means that the system can stay viable throughout time.

Results with asymmetric error constraints are shown in Figure 5.3 and Figure 5.4. In Figure 5.1, y_0 generates a unit step signal. It can be seen that all nodes can perfectly

Table 5.1 Initial State Values of Agents in Scenario 1

Initial Values	Node 1	Node 2	Node 3	Node 4	Node 5
$x_{i,1}$	0	0.1	0.18	0.25	0.3
$x_{i,2}$	0	0	0	0	0
$x_{i,3}$	0	0	0	0	0

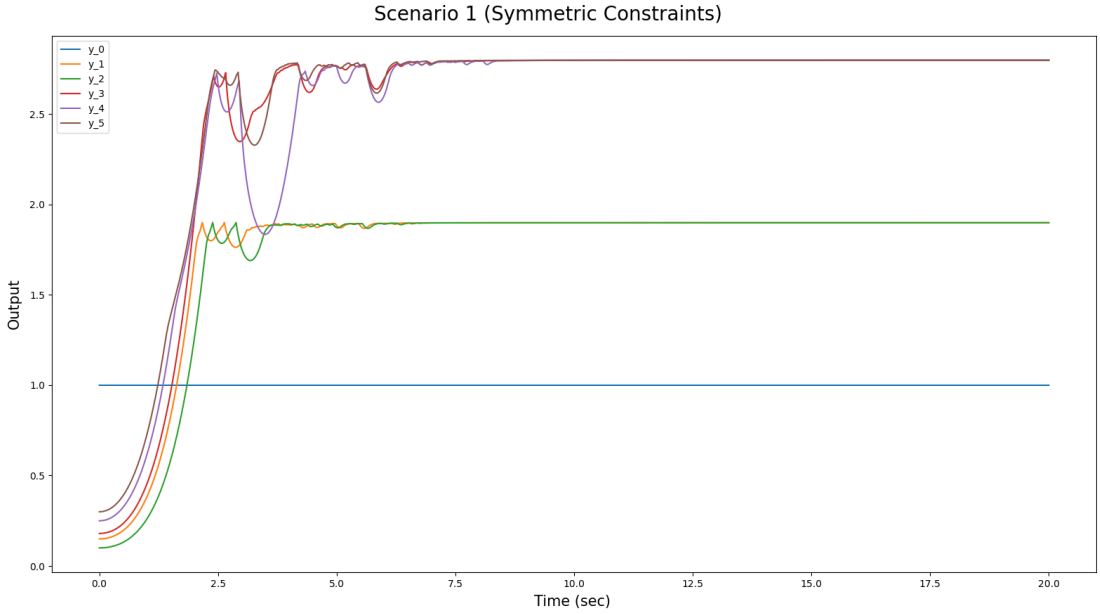


Figure 5.1 Scenario 1 output results with symmetric error constraints

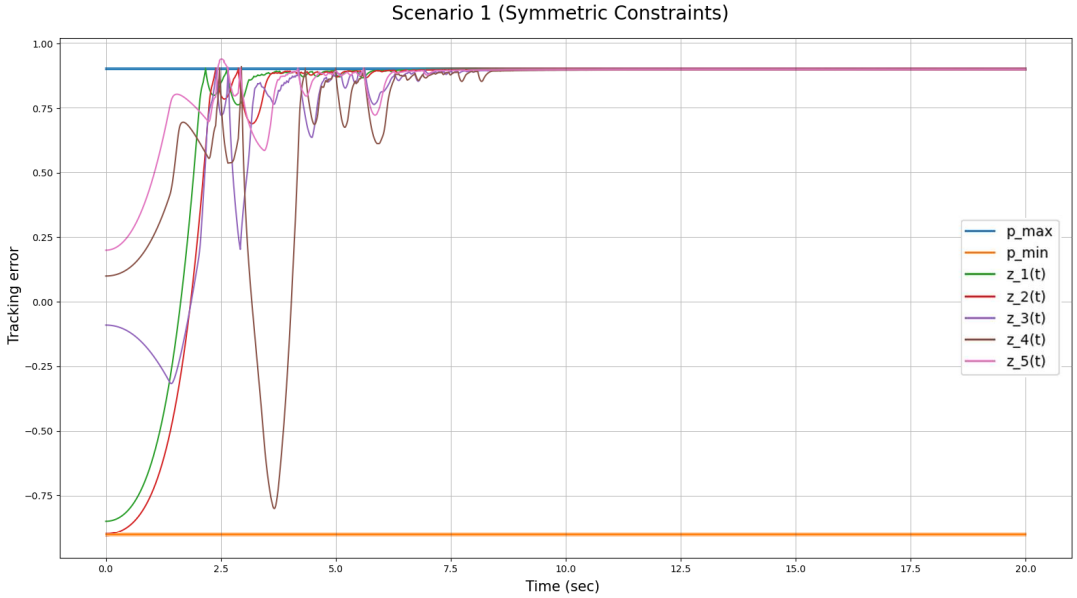


Figure 5.2 Scenario 1 synchronization error results when the system has symmetric error constraints

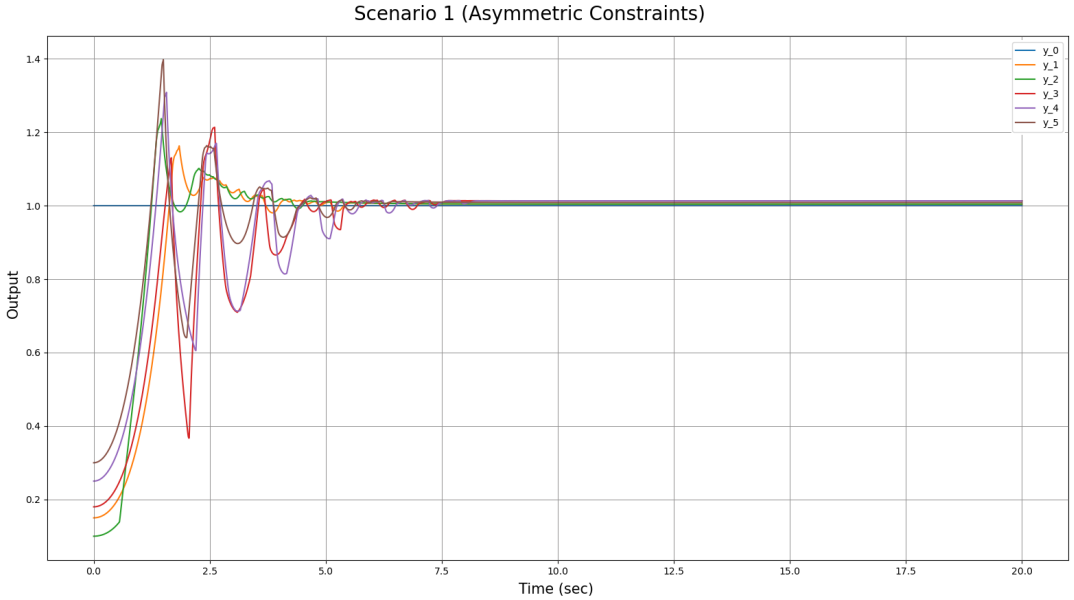


Figure 5.3 Scenario 1 output results with asymmetric error constraints

synchronize the desired signal, and since all nodes can stay in the viability kernel, the system stays viable throughout the time when error constraints are asymmetric, too. According to these results, it can be said that this method gives a successful result when viability kernel boundaries are constant and time-varying.

Scenario 2 is introduced here to demonstrate that this method is also successful when all nodes are behaving differently and an external disturbance, i.e., noise, exists for nodes.

The second scenario is simulated with asymmetric error constraints, and each node has different constraints. Error constraints for four nodes were given in Table 4.6. (4.18a) is selected for extended Lipschitz continuous class K function for this scenario. Constant values, the initial states of four agents are given in Table 5.2. For Table 4.6, boundary offset values are also given in Table 5.2. Results for the second scenario are shown in Figure 5.5 and 5.6. In Figure 5.5, the output value of the node 0 y_0 is selected as a leader of the network system. The other four nodes aim to synchronize the behavior of node 0. It can be seen that the proposed method is also successful when other nodes have unique error constraints and even an external disturbance exists for nodes.

The third scenario is also simulated with asymmetric error constraints, and each node also has different constraints. Error constraints for six nodes were given in Tables 4.7 and 4.8. (4.18a) is selected for extended class K function for the third scenario. Constant values for 4.18a is the same constants as it was set in the second scenario. The initial values for six agents are given in Table 5.3. For Tables 4.7 and

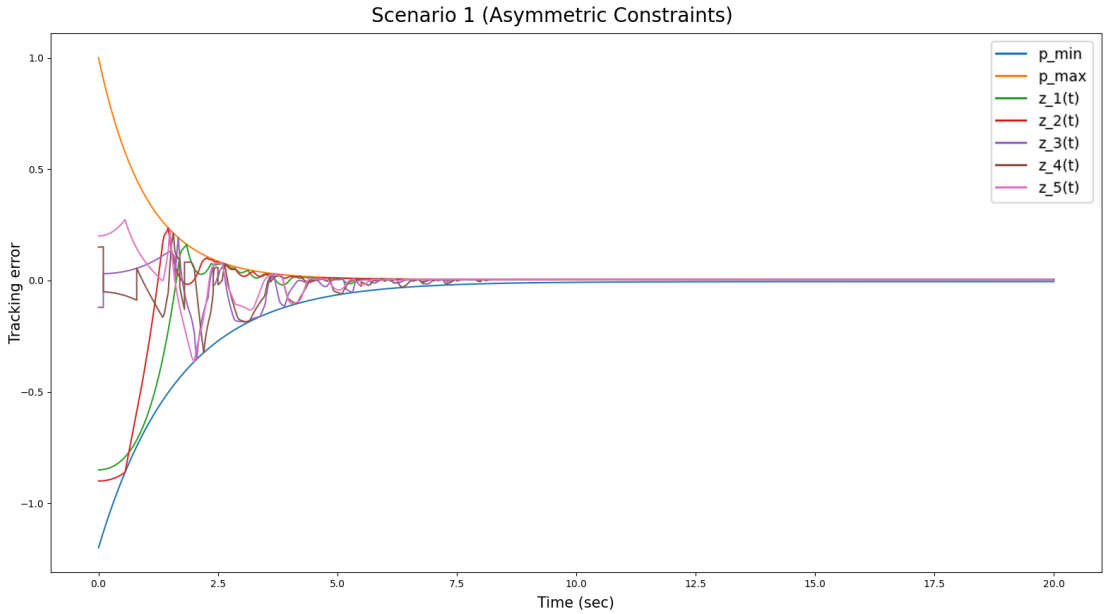


Figure 5.4 Scenario 1 synchronization error results when the system has asymmetric error constraints

Table 5.2 Initial States, Viability Kernel Boundary Offset Values and Constant Values in CBF and Extended Class K Functions for Scenario 2

Initial & Constant Values	Node 0	Node 1	Node 2	Node 3	Node 4
$x_{i,1}$	0	-0.2	0.6	-0.1	0.3
$x_{i,2}$	0	0	0	0	0
$\underline{c}_i$	–	-0.2	-0.3	-0.1	-0.4
$\overline{c}_i$	–	0.2	0.3	0.1	0.4
ϖ_1	–	1000	1000	1000	1000
ϖ_2	–	1	1	1	1
K	–	0.1	0.1	0.1	0.1

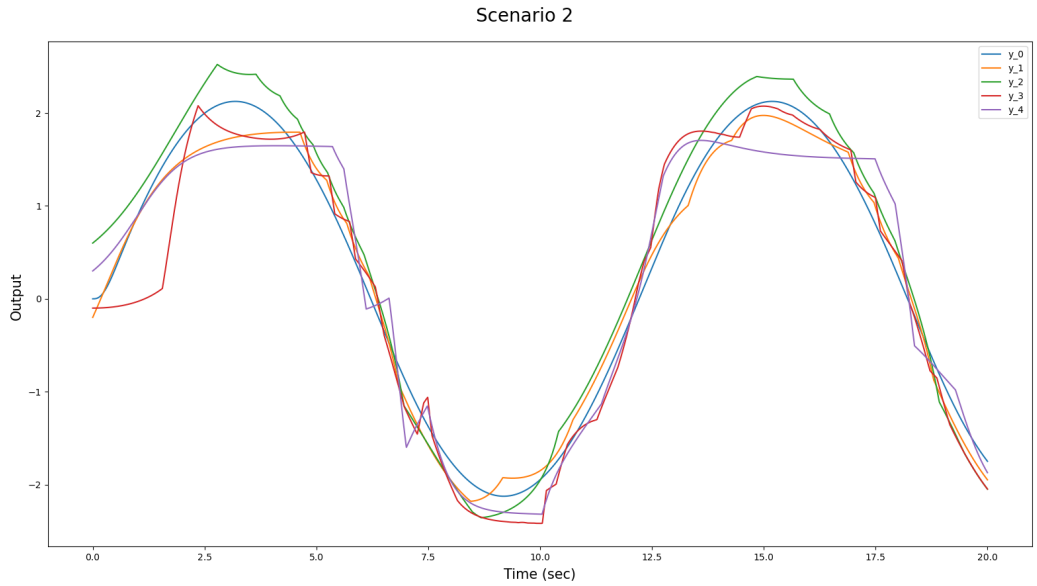


Figure 5.5 Scenario 2 output results

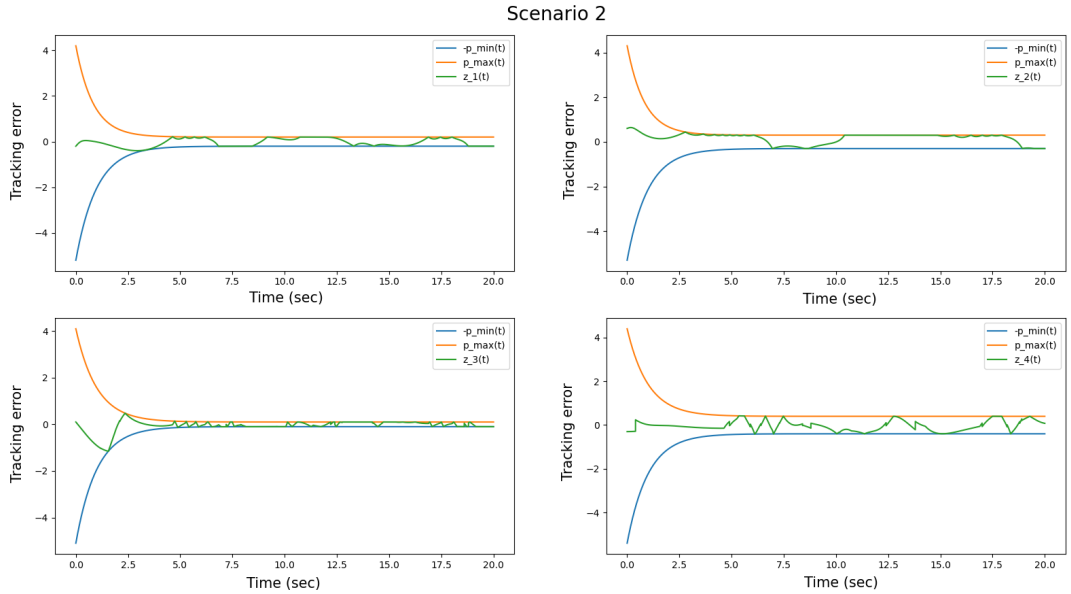


Figure 5.6 Scenario 2 synchronization error results

Table 5.3 Initial States, Viability Kernel Boundary Offset Values and Constant Values in CBF and Extended Class K Functions for Scenario 3

Initial & Constant Values	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
$x_{i,1}$	0	1.6	0.7	0.9	0.15	0.3	0.2
$x_{i,2}$	0	0	0	0	0	0	0
$\underline{c}_i$	–	-0.6	-0.5	-0.7	-0.2	-0.5	-0.1
$\bar{c}_i$	–	0.6	0.5	0.7	0.2	0.5	0.1
ϖ_1	–	1000	1000	1000	1000	1000	1000
ϖ_2	–	1	1	1	1	1	1
K	–	0.1	0.1	0.1	0.1	0.1	0.1

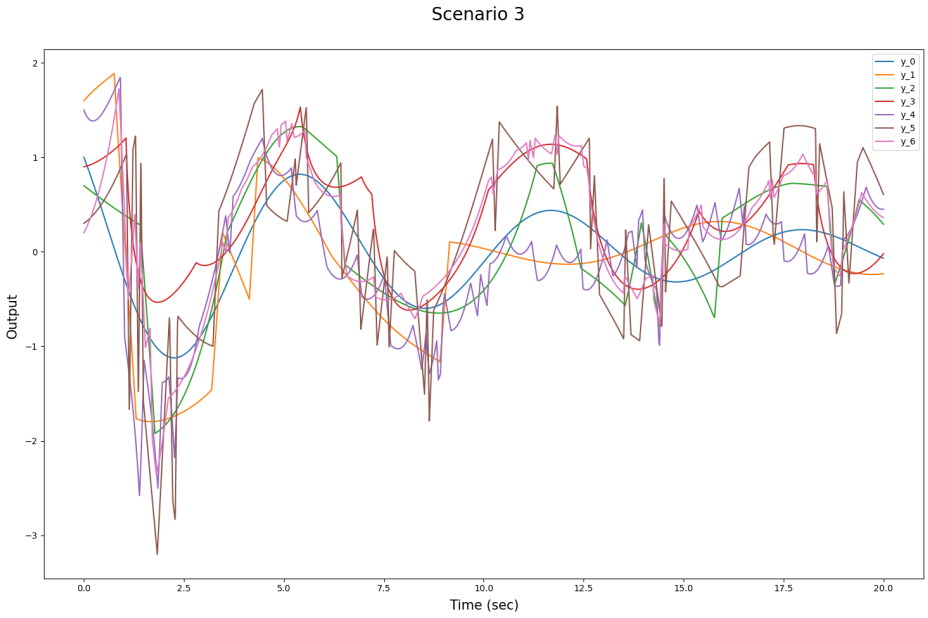


Figure 5.7 Scenario 3 output results

4.8, constants $\underline{c}_i$ and $\bar{c}_i$ are also given in Table 5.3. Just like in the second scenario, the output value of the node 0 y_0 is selected as leader, as shown in Figure 4.3 in this network system. Similar to the second scenario, other six nodes aim to synchronize the behaviour of node 0. It is observed that our method is also successful in this scenario.

You can observe some fluctuations in Figure 5.7, and that's because nodes do not communicate with the controller when they are in safe boundaries. Nodes can be leaded to unsafety because of the disturbance or their behaviors, and in that case, the MPC controller takes over to take them back to safety. With this third scenario, it is shown that our method can be implemented into different systems with different dynamics.

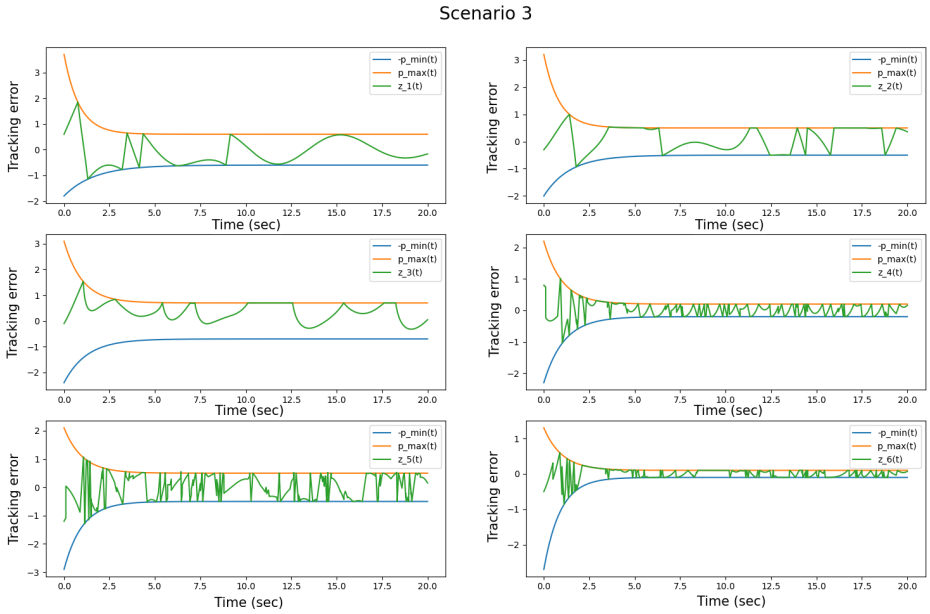


Figure 5.8 Scenario 3 synchronization error results

In all scenarios, nodes transmit their output to their neighbor nodes at 0.0001-second period. Since the system can stay viable in various scenarios, the next step is to compare the proposed method with another method used for adapting the system in the literature. The comparison will be made by obtaining the system's communication cost.

5.2 Results of Viability Kernel and CBF

Viability kernels for all nodes in the second scenario are drawn as a heatmap. It is calculated by checking the control existence of the given initial state of the nodes to their dynamical system. Since nodes are behaving differently, their learned viability kernel is also expected to be different. According to the heatmap graphs, as given in Figure 5.9, the color bar from 0 to 1 refers to the viability score. This score offers a quantified measure of the viability of a state. For the range of viability score, 0 means that the agent is never viable throughout time, and 1 means that the system is always viable throughout time. For each state, it is determined by using the following formula.

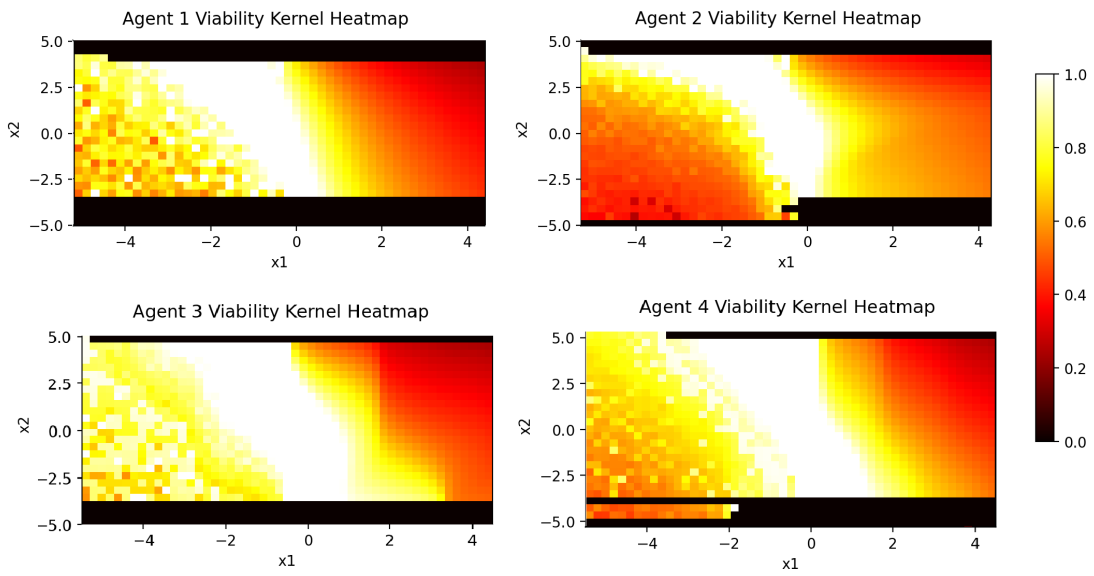


Figure 5.9 Viability Kernels for All Agents in the Second Scenario

$$V_{score}(x_i(t_0)) = \frac{\sum_{t=0}^{T} \mathbb{1}_{V_x}}{S} \qquad i \in \mathcal{V}, \tag{5.1}$$

which S represents the total number of states, equivalent to the cardinality of the set $x_{i,1} \times x_{i,2}$, $x_{i,1} \in \left[-\underline{p_i}(t), \overline{p}_i(t)\right]$, $x_{i,2} \in [-5,5]$. In addition, the indicator function $\mathbb{1}_{V_x}$ defined whether a state lies within the defined viability kernel V_x.

With Figure 5.9, it is proven that all nodes have their own viability kernel to learn to ensure safety. In addition, there are common regions where all nodes are viable throughout time. For example, $x_{i,1}$ values between 0 and 1 are always viable, and this means that the system is successful in safety. As the color goes darker in the heatmap, the probability of an agent staying viable throughout time decreases. For example, according to agent 2 viability kernel heatmap, when $x_{2,1} = 3$ and $x_{2,2} = 3$, the agent is only successful in staying viable in nearly half of the finite time.

To further explain how the CBF $h(x)$ is derived from the viability kernel V_x, we will now define V_x implicitly. Specifically, V_x is represented as a set of points $(x_1, x_2) \in \mathbb{R}^2$, which includes feasible states. There are also infeasible states represented as points to make a classification while obtaining $h(x)$. The feasible states, also can be called as viable states, are obtained with grid sampling method. Since V_x is implicit, we need to approximate boundaries of V_x, and we did that with convex hull approximation on the set of viable states. To approximate $h(x)$, we used a Support Vector Machine (SVM), a machine learning method that analyzes and classifies points to determine the boundary between classifications. In this case, the

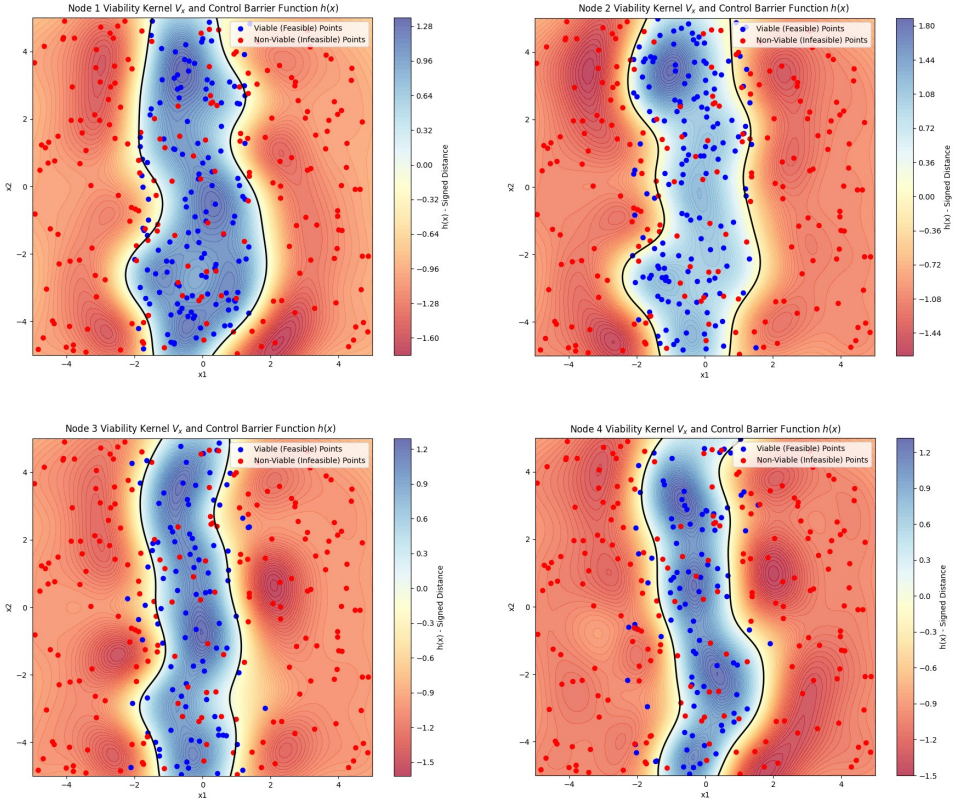


Figure 5.10 Viability Kernel and CBF of All Agents in the Second Scenario

classifications are based on the feasibility of points according to the definition of V_x. The hyperplane of SVM refers to the definition of $h(x)$, which is the signed distance from the hyperplane. Finally, we used $h(x)$ to reflect whether a point lies within the boundary of V_x (i.e., $h(x) > 0$ for points in V_x, $h(x) \leq 0$ otherwise).

In Scenario 2, the feasibility of states for each node is assumed to differ. As a result, we designed $h(x)$ for each individual node. Figure 5.10 illustrates the feasible points representing V_x, a colormap showing the values of $h(x)$ after training the SVC with feasible and infeasible states, and a black line representing the boundary of V_x, derived from $h(x)$. From these figures, it is clear that most feasible states fall within $h(x) > 0$ region (blue), while most infeasible points lie in the $h(x) < 0$ region (red). However, with the method we used for approximation, we still could not obtain the perfect approximation of V_x, because some infeasible states lies within $h(x) > 0$ (blue). Some improvements to obtain an accurate V_x will be necessary.

We also observed the performance of viability with the CBF $h(x)$ by using the SVM method in Scenario 2. We evaluated dynamics of nodes and we used these trajectories with the CBF $h(x)$ designed from SVM. According to the result of $h(x)$, conditions of CBF is checked and the control input is generated accordingly

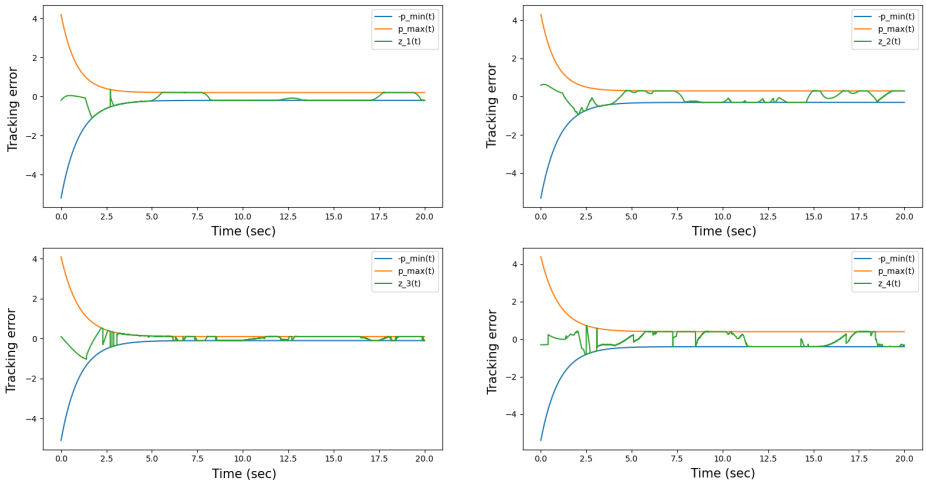


Figure 5.11 Synchronization error results for implicit viability kernel

to guarantee viability. The synchronization error results when the viability kernel is defined implicitly is given in Figure 5.11. Note that since the viability kernel is time-varying, $h(x)$ design is updated according to the viability kernel's time-varying boundaries. According to tracking error results, we can still guarantee viability with the implicit viability kernel, but since $h(x)$ is designed periodically, the overall performance is smoothly degraded.

5.3 Results with MPC and NN Controller

So far, all results were shown with only a Model Predictive (MPC) controller, whose proposed method was the control barrier function. Now, we enhance the controller by including a new controller, which is a trained neural network controller. As explained before, the neural network controller is used for reaching the target region in finite seconds.

This central controller is utilized in the second scenario with the same differential equations and the same aim. The target for this scenario is that all nodes synchronize with the behaviors of node 0 throughout the time. Therefore, the goal of the neural network controller for this scenario is to keep all nodes on the optimal road to reach the target in a given finite seconds.

Accordingly, the model is designed with a graph neural network, and its underlying

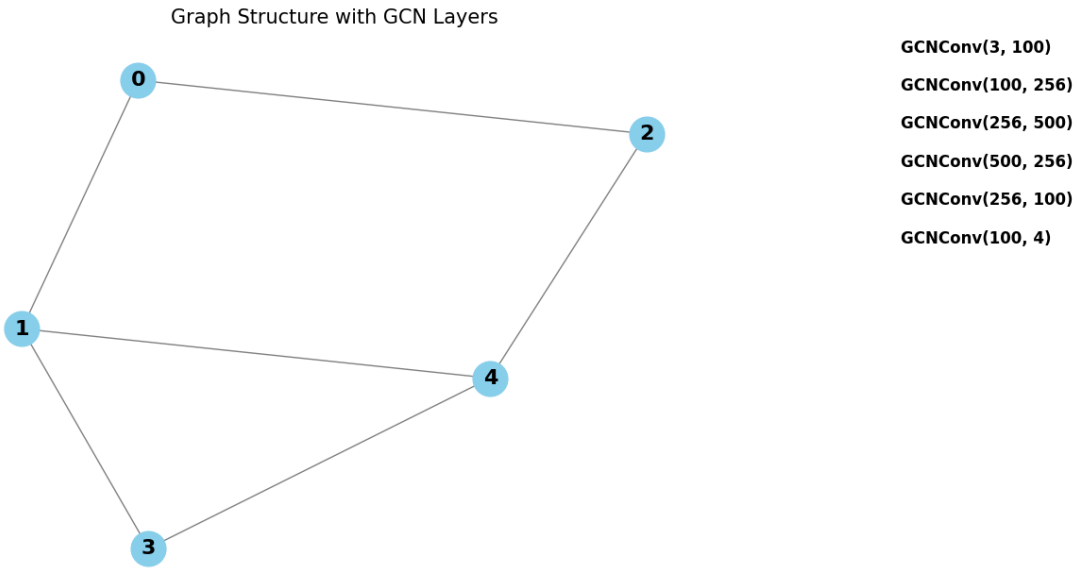


Figure 5.12 Graph Structures and Layers of the Graph Neural Network

graph structure is given in Figure 5.12. The graph structure is designed based on the network topology model given in Figure 4.1 for the second scenario. The graph neural network layers are listed on the upper right side of the figure. This neural network's input is three-dimensional and takes 4 inputs, as the number of followers is 4 according to scenario 2. Since the target here is the synchronization of node 0, each input contains values of performance functions $\underline{p}_i(t)$, $\overline{p}_i(t)$, and the local tracking error $z_i(t)$. Equations of these were given in Table 4.6 and (4.7) respectively. All hidden layers are dense layers with different and coherent units, and ReLU is chosen as an activation function. The output function has 4 distinct values, representing control input values for each node.

The neural network controller is trained with 800000 samples of $\underline{p}_i(t)$, $\overline{p}_i(t)$ and $z_i(t)$ per each node. These samples were obtained with only the model predictive controller when the system was run. For the compilation of the neural network, gradient descent (SGD) is chosen as the optimizer, and mean absolute error is chosen as the loss function. The trained samples were split into train and validation samples with the 90-10 ratio. Early stopping is included for the callback, and monitoring the loss function to prevent overfitting. Moreover, hidden layers are L2-regularized to avoid overfitting with a value of 0.001. The neural network is trained with a batch size of 32 and 50 epochs. Figure 5.13 shows the train and validation loss graphs.

The trained neural network is used as a controller in the system along with the model predictive controller. Test samples are obtained for predictions after evaluating dynamical systems in a finite time, which is chosen as 20. Constant values $\underline{c}_i$, $\overline{c}_i$, in Table 4.6 and initial values for all nodes are chosen randomly to see the performance

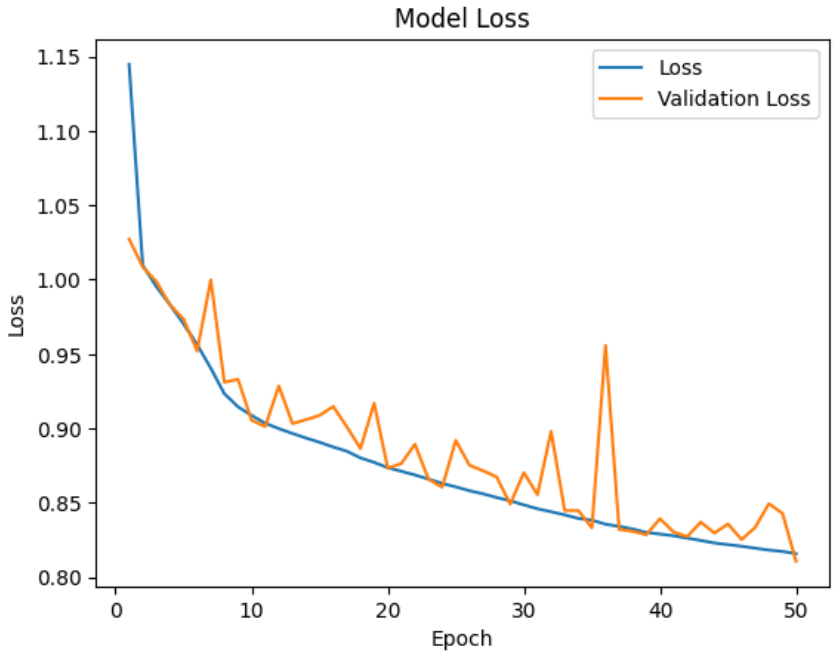
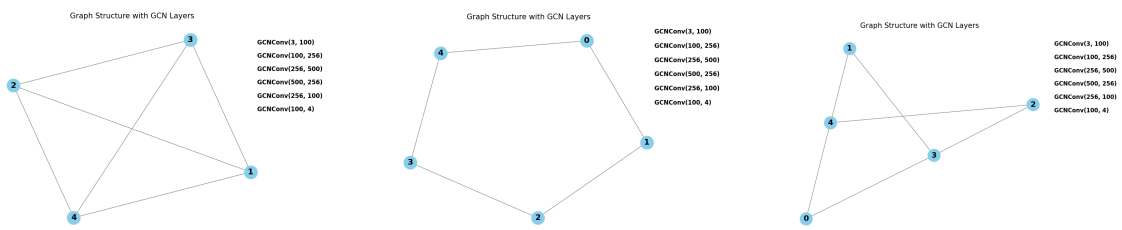


Figure 5.13 Train and Validation Loss Graphs of the Graph Neural Network Model

of the cooperation of neural network and model predictive controller.

Figure 5.13 is obtained from the graph structure given in 5.12. It is observed that the system can stay viable continuously with efficient communication. Moreover, throughout the training procedure, the training and validation loss values are fairly similar. This indicates that the model generalizes well to the validation set and is not overfitted. The synchronization error results were similar to the synchronization error graphics of the system using the only model predictive controller. What we want to demonstrate is that when the neural network controller and the model predictive controller work together, all agents can act with their own learned kernel with less help from other agents. Figure 5.15 shows the control input values of each agent in the system with the NN+MPC controller and the MPC controller. In NN+MPC controller, the NN controller is trained from the graph structure given in Figure 5.12. As it was explained in previous chapters, the control input value of 0



(a) Fully Connected (b) Ring (c) Bipartite

Figure 5.14 Underlying Graph Structures Designed with Three Types of Topologies

Table 5.4 Complexity of NN+MPC and MPC methods

Methods	**Average & Total Time (sec)**	**Computational Complexity**
NN+MPC	0.001920 384.099861	MPC: $O(n^3)$, NN:$O(LR_i^2)$
MPC	0.001279 255.869676	$O(n^3)$

indicates that the node does not need assistance from its corresponding controller. Figure 5.15 is put in here to observe the communication between the node and its corresponding controller. It is shown that nodes can behave with their learned viability kernels with less help from their controllers.

The GNN is also trained with different topologies to observe that our proposed method can cooperate with MPC in any type of topologies described as an underlying graph structure for GNN. These topology structures given in Figure 5.14 are designed based on the common topology types. Figure 5.14a is a fully-connected topology, Figure 5.14b is a ring topology, and Figure 5.14c is a bipartite topology. Nodes in the underlying graph structure represent features of input values. Therefore, we set the number of nodes as 4, and we referred these nodes to boundary functions and initial states of each agent in the MAS system. If the number of nodes increases, a mismatch happens. This means that 4 inputs have to match with 4 nodes and other nodes have to be filled with methods like interpolation or replication, and this may deprecate the performance of the model. Therefore, we set the number of nodes in the underlying graph structure equal to the number of input values of the model to obtain the best performance of GNN. The efficient communication between the node and its corresponding controller was successful with different topologies. Therefore, it can be evaluated that the GNN model is compatible with different topology structures to use viability gains and the controller makes sure that the system can achieve its aim in finite time.

MPC and NN cooperation can bring more complexity, and we quantified the complexity of the cooperation by measuring the time takes for the control decision, and computational complexity. The computational complexity for MPC controller can be formulated as $O(n^3)$, where n is the number of components in a differential equation of an agent, and the computational complexity for GNN, which it provides the complexity of forward pass can be formulated as $O(LR_i^2)$, where L is the number of layers and R_i is the number of neurons of layer i. In addition, we quantified the overhead of this cooperation by measuring the average memory usage, power consumption and CPU usage. The quantified results are written down in Tables 5.4 and 5.5. It should be noted that the measurements are done when methods are in operation, so values are obtained in every iteration of time.

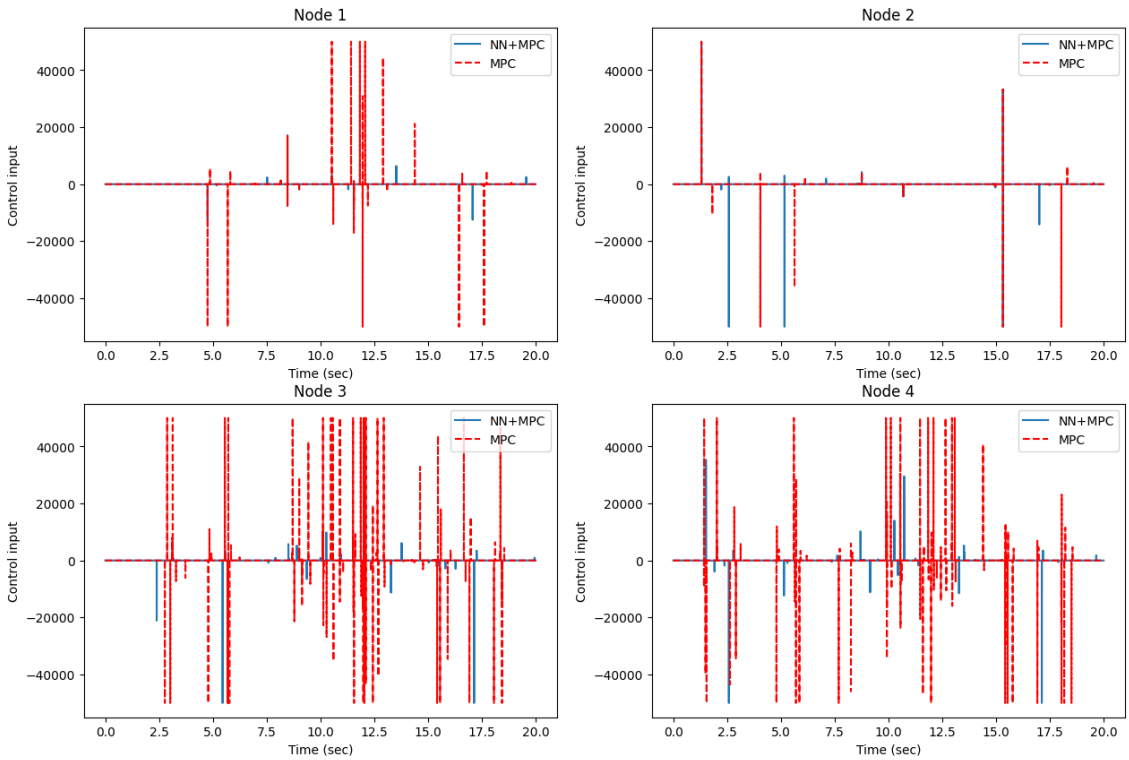


Figure 5.15 Control input values of each node in the system with NN+MPC and MPC

Table 5.5 Overhead of NN+MPC and MPC methods

Methods	**Memory Usage (MB)**	**CPU Utilization (%)**	**Power Consumption (W)**
NN+MPC	1280.58	10.33	12.63
MPC	1071.56	8.87	10.47

5.4 Results with Changing Viability Kernel

We also observed the case with the central controller when the viability kernel is shifted during the finite time. The viability kernel is a time-varying set defined in (2.12). Here, we changed the minimum and maximum limit functions while the simulation ran in a finite time. The system model chosen to observe this situation is the second scenario. The differential equations of nodes were already defined in (4.2).

The neural network is trained according to this case, and with the trained neural network controller, the simulation begins executing with defined maximum and minimum limit functions in Table 4.6, with constant values, which are chosen randomly $\underline{c}_1 = -0.1$, $\overline{c_1} = 0.1$, $\underline{c}_2 = -0.2$, $\overline{c}_2 = 0.2$, $\underline{c}_3 = -0.3$, $\overline{c}_3 = 0.3$, $\underline{c}_4 = -0.5$, $\overline{c}_4 = 0.5$.

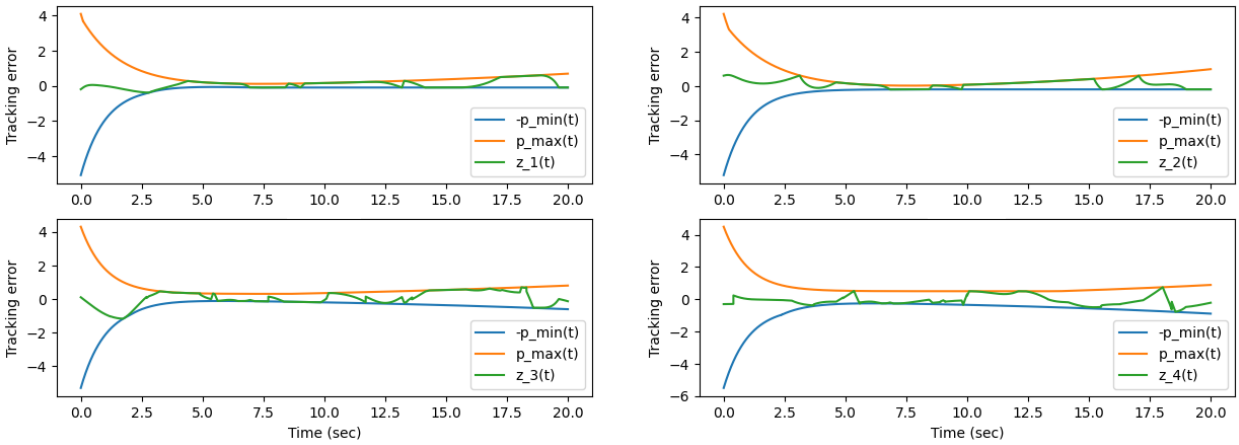


Figure 5.16 Synchronization error results in changing limit functions

While the simulation is running with these limit functions, they are changed at nearly half of the finite time to the following equations.

$$\begin{cases} \underline{p}_1(t) = 7e^{-t} + 0.3e^{0.07t} - 0.4 \\ \overline{p}_1(t) = 4e^{-0.6t} + 0.2e^{0.08t} - 0.3 \end{cases}, \tag{5.2a}$$

$$\begin{cases} \underline{p}_2(t) = 7e^{-0.3t} + 0.3e^{0.06t} - 0.5 \\ \overline{p}_2(t) = 4e^{-0.5t} + 0.2e^{0.1t} - 0.5 \end{cases}, \tag{5.2b}$$

$$\begin{cases} \underline{p}_3(t) = 7e^{-t} + 0.35e^{0.07t} - 0.2 \\ \overline{p}_3(t) = 4e^{-t} + 0.2e^{0.06t} - 0.2 \end{cases}, \tag{5.2c}$$

$$\begin{cases} \underline{p}_4(t) = 7e^{-0.9t} + 0.2e^{0.08t} - 0.1 \\ \overline{p}_4(t) = 4e^{-0.9t} + 0.2e^{0.08t} - 0.1 \end{cases}, \tag{5.2d}$$

The simulation continues with the new minimum and maximum limit functions (5.2) for a while. Then, near to the end of the simulation, the limit functions are again changed to the old ones given in Table 4.6, and the simulation ends with the old minimum and maximum limit functions.

The synchronization errors of all nodes and the output results per node for this case are shown in Figure 5.16 and Figure 5.18 respectively. Because we wanted to apply the change of min and max limit functions fluently, the shifting happened when the values of old and new limit functions were equal to each other at the instant time t. Compared with Figure 5.6, the limit function lines are curved in the current figure. This shows that the limit function is changed during the simulation.

For this case, the control input results are shown in Figure 5.17. Just like Figure 5.15, this result also compares with the scenario that runs with only the MPC

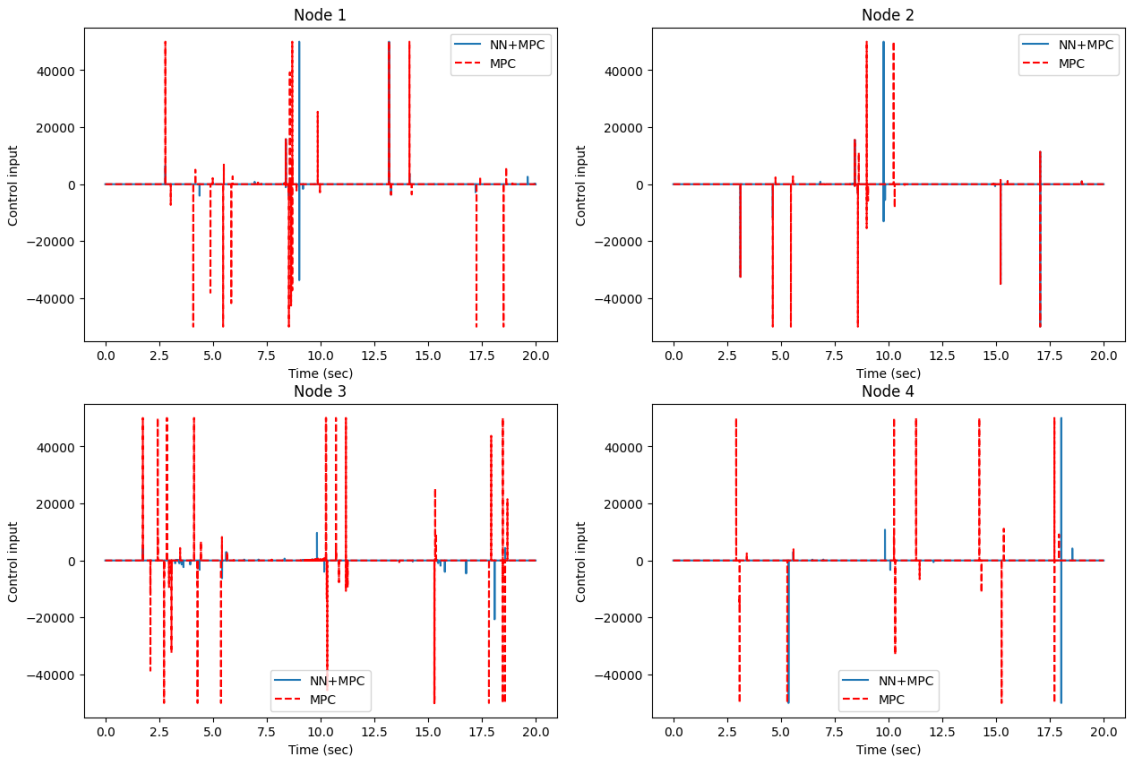


Figure 5.17 Control input compared with MPC controller in changing limit functions

controller.

5.5 Results of Communication Cost

The communication cost results are made by comparing with another method from [76], which does not use viability. In [76], a distributed control scheme is improved by offering a universal error transformation and employing a neural network in the backstepping framework. The hysteresis quantizer's sector characteristic is combined with the minimal learning parameter concept, and this is used in the neural network to decrease the computational complexity. From now on, we are going to call this method as the control law method.

For the communication cost, the analysis is divided into two parts. The first part gives the result of the packets transmitted from controllers to corresponding nodes. The control signal transmission is done according to the control function. In other words, instead of checking the value of the control transmitted to the node, it is

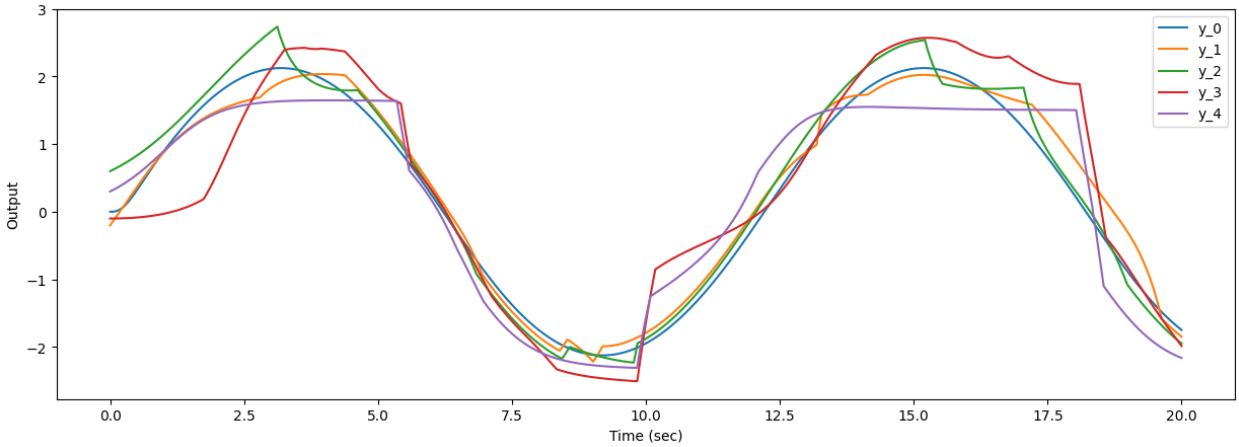


Figure 5.18 Output results in changing limit functions

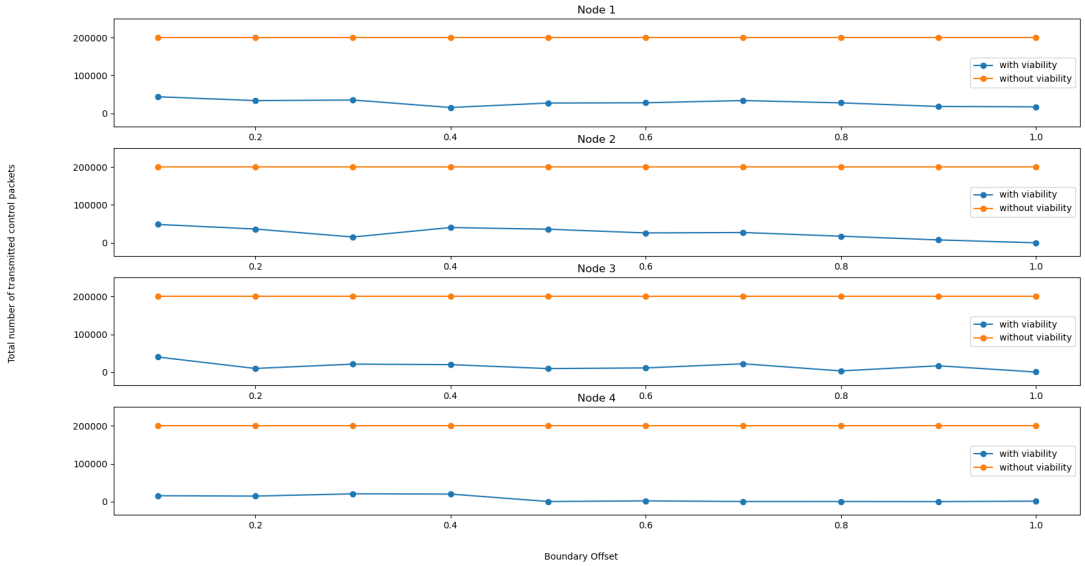


Figure 5.19 Viability Kernel Boundary Offset - Total Transmitted Number of Control Packets

checked whether the control value is zero or not. If the control value is zero, it means that the controller does not have to transmit the control signal to the corresponding node. Otherwise, the node needs to be restored with the help of the control signal. The communication between the controller and the corresponding node is examined by counting the total number of control signals transmitted as packets, just like the transmission of output data between nodes. The transmission time for the control signal transmission is chosen as τ_{t_c}=0.0001 seconds. This communication is also analyzed with the boundary offset of the performance functions defined in Table 4.6. In Table 4.6, the boundary offset definition refers to the constant value summed with the exponential component. The boundary offset values are picked from 0.1 to 1.0 with the increasing order of 0.1, and the total number of transmitted control packets is counted. The results are shown in Figure 5.19.

The second part of the communication analysis lies in the communication between nodes. According to the communication topology, two nodes are directly connected to more than one nodes: node 3 and node 4. The error constraints were designed assuming nodes can receive only one of the node's data. Therefore, the communication cost graphs are drawn according to the neighbors of node 3 and node 4. According to Figure 4.2, nodes 3 and 4 are connected to more than one node, and all other nodes are connected to only one node. If a node has one neighbor node, then the neighbor node priority decision is not needed, and therefore, the neighbor node transmits data continuously. In Figure 4.2, nodes 3 and 4 are able to receive from more than one neighbor node, but because of the limitation, the priority is scheduled periodically. In addition, node 3 does not transmit data, so the number of transmitted bits is zero.

Both methods are simulated with the data transmission period as $\tau_t = 0.0001$, and the priority scheduling period is chosen as $\tau_p = 0.1$. The selection of the minimum difference between the output of the node and the neighbor of the node does the design of the priority scheduling.

Figures 5.20 and 5.21 show the transmitted number of bits in 0.25 second-period for some nodes in the system. For calculating the transmitted bits, the bit depth is taken as 32 because of the encoding and decoding of output data. The encoding is done by transforming the output data into byte symbols and then transforming that into binary values. The decoding is the inverse of the encoding operation. Figure 5.20 shows the transmitted bits of nodes when the system is adapted with our proposed method: the viability method. Figure 5.21 shows the transmitted bit of nodes with control law method.

According to Figure 5.19, in the control law method, the boundary change does not affect the communication between the controller and the node. That's because the control law method looks at the difference between the tracking error and boundary values and makes the control decision accordingly. In our proposed method, the controller can decide that the node does not need help because it evaluates that the node is already safe. This evaluation happens by knowing the viability kernel and therefore, this highlights the safety and operational integrity property. Particularly, knowing the viability kernel assists in guaranteeing that each node operates safely without violating constraints, and this property allowed us to achieve these results.

To sum up this chapter, our proposed method ensures system viability by reaching all nodes in finite time with efficient controller and neighbor assistance. It achieves good performance with fewer penalties and low communication costs. Knowing the viability kernel allows agents to adapt to changes with efficient communication, even

when the kernel changes.

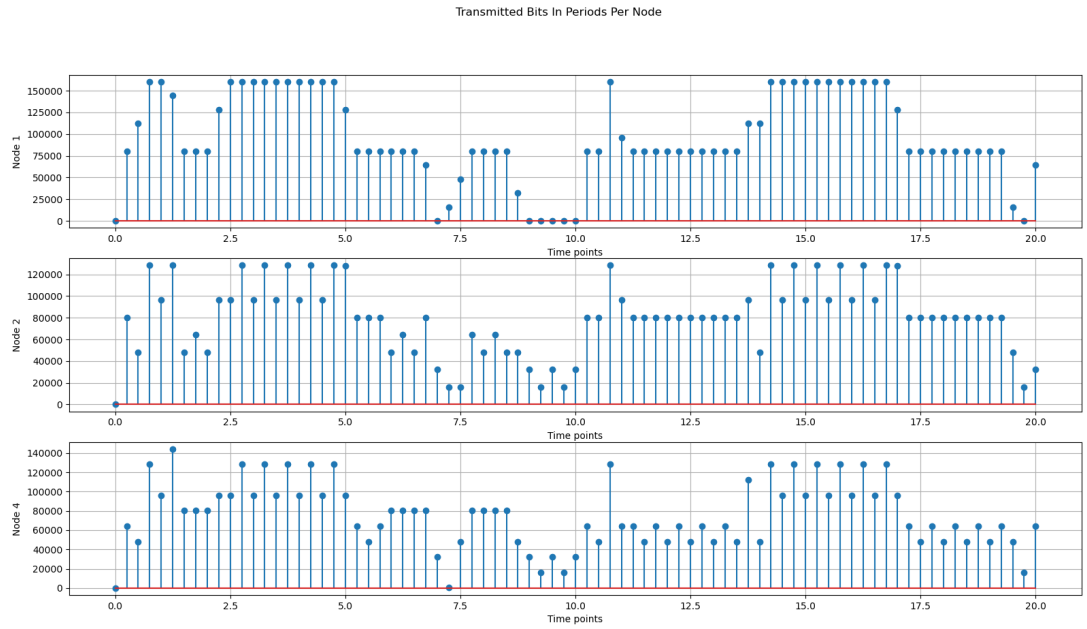


Figure 5.20 The number of transmitted bits with viability method

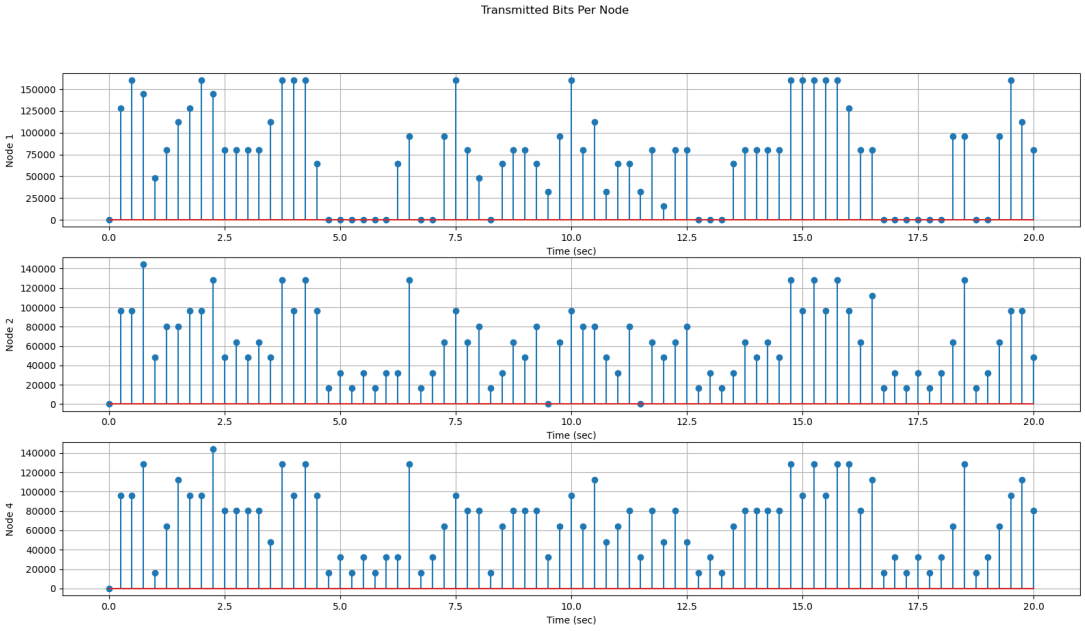


Figure 5.21 The number of transmitted bits without viability method

6. CONCLUSION & FUTURE WORKS

In this thesis, we provided an effective and efficient tracking control system for a system that is addressed as susceptible to errors and failures. This proposed controller offers an optimal path by treating these failures as a norm instead of an exception to reach the target. With the viability theory concept, we could make sub-optimal decisions for each node comprised in the network, guaranteeing robustness and developed availability even under severe failures. This control mechanism ensures that the network system can handle any incoming errors and achieve its goal. It is designed such that the communication is kept as low as possible so that nodes can behave with their learned kernels. In addition, the time-varying kernels are also considered as tracking them down and notifying agents about them.

In Chapter 2, we proposed a model predictive controller with the control barrier function method. It was related to the viability kernel to obtain the viability region of each agent, and we explained the solution step by step. In addition, we defined the neural network controller for the multi-agent system to utilize the gains of the viability region in the system. The model predictive controller steers the agent to a point closer to the boundaries of the viability region when the agent finds itself near its boundaries because of the external conditions, and the neural network controller utilizes the viability region to verify whether it can reach the target in a finite time. The cooperation between them was explained in this chapter. Chapter 3 gives the communication protocol to analyze the communication cost of the multi-agent system later. The communication was divided into two sections: the communication between a node and its corresponding controller and the communication between nodes. The algorithms of both of these communications were given at the end of the chapter. In Chapter 4, we introduced three distinct network scenarios, each with a different topology but sharing the same objective. We addressed system constraints in both symmetrical and asymmetrical forms, meaning constraints with either constant or time-varying boundaries. Based on these constraints, we defined the viability kernel and subsequently derived the control barrier function. The model predictive controller, to be applied later, is formulated based on the control

barrier function, incorporating the given state and input constraints. In Chapter 5, we provided the numerical results of our proposed method. We applied the model predictive controller in three scenarios. From the results of the synchronization error and output, it was seen that the model predictive controller can restore agents to inside of the viability region when their safety is in danger. Viability kernels were visualized, demonstrating that each node learns its own kernel for safety, with shared regions of viability across all nodes. Further investigation involved defining the viability kernel implicitly and designing the control barrier function using a support vector machine. While both the viability kernel and the approximated control barrier function were illustrated, the approximation results were not very accurate. As a next step, we added the neural network controller, designing a central controller per each node to see that this design gives more efficient results than the only model predictive controller. It is designed with graph neural network model with its underlying graph structure, which is changed later to show that we can achieve our aim with different graph structures. We achieved this by observing the control input sent to the corresponding nodes, and it can be seen that with our proposed method, the communication between the node and its corresponding controller is less than the system with only the model predictive controller. In other words, the given scenario can reach the intended goal with more minimized risk and less communication, ending in obtaining the safe and optimal path for all nodes. We also examined a case where the viability kernel changes throughout the simulation and demonstrated that our controller approach effectively manages the shifting viability kernel, ensuring the system remains safe. The communication cost is more deeply analyzed by providing more graphs, compared with another method. This other method was proposed in the literature, and its proposed controller generates more control input values to reach the target. These results show that we achieved less communication cost in the system while nodes self-adapted with their learned viability kernels.

The results presented in Chapter 5 show that our proposed controller method generally achieved its objectives, though some results were less successful than expected. For example, in the implicit viability kernel case, we were unable to accurately approximate the control barrier function. In the future, we aim to improve this using nonlinear kernel-based classification algorithms.

Looking ahead, this proposed method could be implemented in an operational environment. It is well-suited for systems with a central control unit that monitors the viability kernels of individual agents within the system and communicates updates to them. For instance, our approach could be applied to a system involving unmanned aerial vehicles (UAVs) communicating with a control center. The UAVs

would receive control instructions from the center and transmit their current states back. Additionally, this method could be used in a more general multiplex setup, such as one consisting of access points, base stations, and mobile devices, which can be considered as nodes. Since these nodes exhibit different behaviors, their respective controllers can monitor their viability kernels and provide updates, whether to address uncertainty or to optimize network performance.

Bibliography

[1] M.-I. Corici, F. Eichhorn, R. Bless, M. Gundall, D. Lindenschmitt, B. Bloessl, M. Petrova, L. Wimmer, R. Kreuch, T. Magedanz, and H. D. Schotten, "Organic 6G Networks: Vision, Requirements, and Research Approaches," *IEEE Access*, vol. 11, pp. 70 698–70 715, Jan. 2023.

[2] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Bayesian Online Learning for Energy-Aware Resource Orchestration in Virtualized RANs," Proceedings of *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10, 2021.

[3] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Pérez, A. Banchs, and J. J. Alcaraz, "vrAIn: Deep Learning Based Orchestration for Computing and Radio Resources in vRANs," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2652–2670, 2022.

[4] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-aho, "Deep Reinforcement Learning for Orchestrating Cost-Aware Reconfigurations of vRANs," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 200–216, 2024.

[5] S. Trimpe and D. Baumann, "Resource-Aware IoT Control: Saving Communication Through Predictive Triggering," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5013–5028, 2019.

[6] M. Camelo, M. Claeys, and S. Latré, "Parallel Reinforcement Learning With Minimal Communication Overhead for IoT Environments," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1387–1400, 2020.

[7] D. Hetzer, M. Muehleisen, A. Kousaridas, and J. Alonso-Zarate, "5G Connected and Automated Driving: Use Cases and Technologies in Cross-border Environments," Proceedings of *2019 European Conference on Networks and Communications (EuCNC)*, pp. 78–82, 2019.

[8] N. Mouawad, R. Naja, and S. Tohme, "SDN Based Handover Management For a Tele-Operated Driving Use Case," Proceedings of *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 47–54, 2019.

[9] N. Koursioumpas, L. Magoula, I. Stavrakakis, N. Alonistioti, M. A. Gutierrez-Estevez, and R. Khalili, "DISTINQT: A Distributed Privacy Aware Learning Framework for QoS Prediction for Future Mobile and Wireless Networks," *ArXiv*, vol. abs/2401.10158, 2024.

[10] G. Jornod, A. E. Assaad, A. Kwoczek, and T. Kürner, "Packet Inter-Reception Time Modeling for High-Density Platooning in Varying Surrounding Traffic Density," Proceedings of *2019 European Conference on Networks and Communications (EuCNC)*, pp. 187–192, 2019.

[11] S. Hegde, O. Blume, R. Shrivastava, and H. Bakker, "Enhanced Resource Scheduling for Platooning in 5G V2X Systems," Proceedings of *2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 108–113, 2019.

[12] G. Jornod, A. E. Assaad, A. Kwoczek, and T. Kürner, "Prediction of Packet Inter-Reception Time for Platooning using Conditional Exponential Distribution," Proceedings of *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 265–270, 2019.

[13] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-Aho, "Learning-Based Orchestration for Dynamic Functional Split and Resource Allocation in vRANs," Proceedings of *2022 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, pp. 243–248, 2022.

[14] J. X. S. Lozano, A. Garcia-Saavedra, X. Li, and X. C. Perez, "AIRIC: Orchestration of Virtualized Radio Access Networks With Noisy Neighbours," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 432–445, 2024.

[15] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Orchestrating Energy-Efficient vRANs: Bayesian Learning and Experimental Results," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2910–2924, 2023.

[16] N. Apostolakis, M. Gramaglia, L. E. Chatzieleftheriou, T. Subramanya, A. Banchs, and H. Sanneck, "ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263–279, 2024.

[17] F. H. Grings, L. B. D. Silveira, K. V. Cardoso, S. Correa, L. R. Prade, and C. B. Both, "Full Dynamic Orchestration in 5G Core Network Slicing Over A Cloud-Native Platform," Proceedings of *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 2885–2890, 2022.

[18] F. W. Murti, S. Ali, and M. Latva-aho, "A Bayesian Framework of Deep Reinforcement Learning for Joint O-RAN/MEC Orchestration," *ArXiv*, vol. abs/2312.16142, 2023.

[19] J. Liu, Z. Wang, P. Hang, and J. Sun, "Delay-Aware Multi-Agent Reinforcement Learning for Cooperative Adaptive Cruise Control with Model-based Stability Enhancement," *ArXiv*, vol. abs/2404.15696, 2024.

[20] M. Diamanti, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Unified User Association and Contract-Theoretic Resource Orchestration in NOMA Heterogeneous Wireless Networks," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1485–1502, 2020.

[21] H. Ma and S. Zhou, "Noisy Sensor Scheduling in Wireless Networked Control Systems: Freshness or Precision," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 1107–1111, 2022.

[22] J. Chai and R. G. Sanfelice, "Forward Invariance of Sets for Hybrid Dynamical Systems (Part I)," *IEEE Transactions on Automatic Control*, vol. 64, no. 6, pp. 2426–2441, 2019.

[23] J. Chai and R. Sanfelice, "Forward Invariance of Sets for Hybrid Dynamical Systems (Part II)," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 89–104, 2021.

[24] T. Ameen, S. Mukhopadhyay, and N. Qaddoumi, "Computing Robust Forward Invariant Sets of Multidimensional Nonlinear Systems via Geometric Deformation of Polytopes," *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 8293–8300, 2023.

[25] Z. Zhu, Y. Chai, Z. Song, and P. Huang, "A New Safety Criterion for Dynamics Systems by Barrier Function based on Forward Invariant Set," Proceedings of *2021 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS)*, pp. 1–6, 2021.

[26] C. Chi, G.-P. Liu, and C.-M. Ionescu, "Discrete Sliding Mode Predictive Control of Networked Control Systems," Proceedings of *2023 42nd Chinese Control Conference (CCC)*, pp. 5332–5337, 2023.

[27] C. Chi, G. Liu, and C. Ionescu, "Adaptive Discrete Sliding-mode Predictive Control of Networked Multi-agent Control System Based on the Incremental Model," Proceedings of *2023 2nd Conference on Fully Actuated System Theory and Applications (CFASTA)*, pp. 349–354, Jul. 2023.

[28] S. Kolathaya and A. D. Ames, "Input-to-State Safety With Control Barrier Functions," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 108–113, 2019.

[29] H. M. Sweatland, A. Isaly, and W. E. Dixon, "High-Order Control Barrier Function for Constraining Position in Motorized Rehabilitative Cycling," Proceedings of *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 951–956, 2022.

[30] K. Garg and D. Panagou, "Robust Control Barrier and Control Lyapunov Functions with Fixed-Time Convergence Guarantees," Proceedings of *2021 American Control Conference (ACC)*, pp. 2292–2297, 2021.

[31] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-Time Control Barrier Function: High-Order Case and Adaptive Case," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3231–3239, 2023.

[32] J. Breeden, K. Garg, and D. Panagou, "Control Barrier Functions in Sampled-Data Systems," *IEEE Control Systems Letters*, vol. 6, pp. 367–372, 2022.

[33] M. H. Yeganegi, M. Khadiv, A. D. Prete, S. A. A. Moosavian, and L. Righetti, "Robust Walking Based on MPC With Viability Guarantees," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2389–2404, 2022.

[34] M. A. Bouguerra, T. Fraichard, and M. Fezari, "Viability-Based Guaranteed Safe Robot Navigation," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 459–471, Aug. 2019.

[35] D. Monnet, J. Ninin, and L. Jaulin, "Computing an Inner and an Outer Approximation of the Viability Kernel," *Reliable Computing*, vol. 22, Aug. 2016.

[36] B. Peng, Z. Zhang, and Y. Song, "Dynamic Coverage Control for Constrained Mobile Sensor Networks," *IEEE Transactions on Automatic Control*, vol. 69, no. 4, pp. 2131–2142, 2024.

[37] T. Aschenbruck, F. Petzke, P. Rumschinski, and S. Streif, "On Consistency, Viability, and Admissibility in Constrained Ensemble and Hierarchical Control Systems," *IEEE Transactions on Automatic Control*, vol. 68, no. 8, pp. 4990–4997, 2023.

[38] P.-F. Massiani, S. Heim, F. Solowjow, and S. Trimpe, "Safe Value Functions," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2743–2757, 2023.

[39] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," *ArXiv*, vol. abs/1312.6199, 2013.

[40] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient Neural Network Robustness Certification with General Activation Functions," Proceedings of *32nd International Conference on Neural Information Processing Systems*, p. 4944–4953, 2018.

[41] C. Tomlin, J. Lygeros, and S. Shankar Sastry, "A Game Theoretic Approach to Controller Design for Hybrid Systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[42] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable Set Estimation for Neural Network Control Systems: A Simulation-Guided Approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2021.

[43] J.-P. Aubin, A. Bayen, and P. Saint-Pierre, *Viability Theory: New Directions*, Jan. 2011.

[44] J. Turriff, "Viability Kernels for Nonlinear Systems," Master's thesis, University of Toronto, Toronto, ON, 2008.

[45] P. Saint-Pierre, "Approximation of the viability kernel," *Applied Mathematics and Optimization*, vol. 29, no. 2, pp. 187–209, Mar. 1994.

[46] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[47] Z. She and B. Xue, "Brief Paper: Computing an invariance kernel with target by computing Lyapunov-like functions," *Control Theory Applications, IET*, vol. 7, pp. 1932–1940, Oct. 2013.

[48] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. Oishi, and G. A. Dumont, "Lagrangian methods for approximating the viability kernel in high-dimensional systems," *Automatica*, vol. 49, no. 7, pp. 2017–2029, 2013.

[49] N. Bohorquez, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Safe navigation strategies for a biped robot walking in a crowd," Proceedings of *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 379–386, 2016.

[50] A. Liniger and J. Lygeros, "Real-Time Control for Autonomous Racing Based on Viability Theory," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, p. 464–478, Mar. 2019.

[51] M. Kalisiak and M. van de Panne, "Faster Motion Planning Using Learned Local Viability Models," Proceedings of *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 2700–2705, 2007.

[52] M. Kalisiak, M. van de Panne, "Approximate Safety Enforcement Using Computed Viability Envelopes," Proceedings of *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2004*, vol. 5, pp. 4289–4294, 2004.

[53] M. Nagumo, "Uuml;ber die Lage der Integralkurven gewouml;hnlicher Differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

[54] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[55] R. H. Abraham, J. E. Marsden, T. S. Ratiu, and C. DeWitt-Morette, *Manifolds, Tensor Analysis, and Applications*. Springer New York, NY, Dec. 1983, vol. 75.

[56] S. Prajna and A. Jadbabaie, "Safety Verification of Hybrid Systems Using Barrier Certificates," Proceedings of *Hybrid Systems: Computation and Control*, pp. 477–492, 2004.

[57] S. Prajna and A. Rantzer, "On the Necessity of Barrier Certificates," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 526–531, 2005.

[58] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[59] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier Lyapunov Functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[60] P. Wieland and F. Allgöwer, "Constructive Safety Using Control Barrier Functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[61] M. Z. Romdlony and B. Jayawardhana, "Uniting Control Lyapunov and Control Barrier Functions," Proceedings of *53rd IEEE Conference on Decision and Control*, pp. 2293–2298, 2014.

[62] B. J. Muhammad Zakiyullah Romdlony, "Stabilization with Guaranteed Safety Using Control Lyapunov–Barrier Function," *Automatica*, vol. 66, pp. 39–47, 2016.

[63] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," Proceedings of *53rd IEEE Conference on Decision and Control*, pp. 6271–6278, 2014.

[64] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[65] X. Xu, T. Waters, D. Pickem, P. Glotfelter, M. Egerstedt, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Realizing simultaneous lane keeping and adaptive speed regulation on accessible mobile robot testbeds," Proceedings of *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1769–1775, 2017.

[66] L. Wang, A. D. Ames, and M. Egerstedt, "Safety Barrier Certificates for Collisions-Free Multirobot Systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[67] L. Wang and E. M. Ames, Aaron D., "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," Proceedings of *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3293–3298, 2017.

[68] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3D dynamic walking on stepping stones with control barrier functions," Proceedings of *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 827–834, 2016.

[69] F. Lewis, "Neural network control of robot manipulators," *IEEE Expert*, vol. 11, no. 3, pp. 64–75, 1996.

[70] I. Tezuka and H. Nakamura, "Strict Zeroing Control Barrier Function for Continuous Safety Assist Control," *IEEE Control Systems Letters*, vol. 6, pp. 2108–2113, 2022.

[71] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," Proceedings of *2019 18th European control conference (ECC)*, pp. 3420–3431, 2019.

[72] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of Control Barrier Functions for Safety Critical Control," *IFAC-PapersOnLine*, vol. 48, no. 27, p. 54–61, 2015.

[73] I. Tezuka, H. Nakamura, T. Hatano, K. Kamijo, and S. Sato, "Input-constrained Human Assist Control via Control Barrier Function for Viability Kernel," Proceedings of *2023 American Control Conference (ACC)*, pp. 3921–3926, 2023.

[74] M. Protter and C. Morrey, *A First Course in Real Analysis*, ser. Undergraduate Texts in Mathematics. Springer New York, 2012.

[75] Y. Jiang, Z. Liu, Z. Chen, and F. Duan, "Error-constrained Coordinated Tracking Control for High-order Multiagent Systems Based on Barrier Lyapunov Function," *International Journal of Control, Automation and Systems*, vol. 20, no. 4, pp. 1238–1249, Apr. 2022.

[76] Y. Jiang, Z. Liu, and Z. Chen, "Output Synchronization of Heterogeneous Nonlinear Multiagent Systems With Input Quantization: A Universal Performance Guaranteed Control Scheme," *IEEE Transactions on Control of Network Systems*, vol. 10, no. 3, pp. 1590–1602, 2023.

[77] H. Zhang, X. Zhao, G. Zong, and N. Xu, "Fully Distributed Consensus of Switched Heterogeneous Nonlinear Multi-Agent Systems With Bouc-Wen Hysteresis Input," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 4198–4208, 2022.

APPENDIX A

A.1 Gronwall's Lemma

Here, the following Gronwall's Lemma will be adopted instead of directly using the comparison Lemma to guarantee the forward invariance of a safe set.

Let $\lambda \in \mathbb{R}$ be a constant, and $k : [t_0, t_1] \to \mathbb{R}_{\geq 0}$ be a non-negative continuous function. If an absolute continuous non-negative function $q : [t_0, t_1] \to \mathbb{R}_{\geq 0}$ satisfies the following equation

$$q(t) \leq \lambda + \int_{t_0}^{t} k(s)q(s)ds \tag{A.1}$$

for $\forall t \in [t_0, t_1]$, then on the same interval, the following inequality holds:

$$q(t) \leq \lambda \exp\left(\int_{t_0}^{t} k(s)ds\right) \tag{A.2}$$

Under the assumption that $q(t)$ is a C^1 continuously differentiable function, we obtain the following result.

Let $k : [t_0, t_1] \to \mathbb{R}_{\geq 0}$ be a non-negative continuous function. If a non-negative C^1 continuously differentiable function $q : [t_0, t_1] \to \mathbb{R}_{\geq 0}$ satisfies the following inequality

$$\dot{q}(t) \geq -k(t)g(t) \tag{A.3}$$

for $\forall t \in [t_0, t_1]$, then on the same interval, the following inequality holds:

$$q(t) \geq q(t_0) \exp\left(-\int_{t_0}^{t} k(s)ds\right) \tag{A.4}$$

In particular, if $k(t) = \eta \in \mathbb{R}_{\geq 0}$ is a non-negative constant, then the following inequality holds for $\forall t \in [t_0, t_1]$:

$$q(t) \geq q(t_0) \exp\left(-\eta\,(t - t_0)\right) \tag{A.5}$$

APPENDIX B

B.1 Runge-Kutta Methods

Runge-Kutta methods have the same high-order local truncation error as Taylor methods; the difference here is that there is no need to compute and evaluate the derivatives of $f(t,y)$. Before presenting the details of the method, let's look at Taylor's theorem in two variables. The proof of this theorem exists in any standard of advanced calculus.

Theorem B.1. *$f(t,y)$ and all of its partial derivatives of the order less than or equal to $n+1$ are supposed to be continuous on $D=\{(t,y)\mid a\le t\le b,\ c\le y\le d\}$, and let $(t_0,y_0)\in D$. For every $(t,y)\in D$, a constant ξ exists between t and t_0 and μ exists between y and y_0 with the following equation*

$$f(t,y)=P_n(t,y)+R_n(t,y) \tag{B.1}$$

where the functions P_n and R_n are defined mathematically as follows.

$$\begin{aligned}P_n(t,y)=f(t_0,y_0)&+\left[(t-t_0)\frac{\partial f}{\partial t}(t_0,y_0)+(y-y_0)\frac{\partial f}{\partial y}(t_0,y_0)\right]\\&+\left[\frac{(t-t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0,y_0)+(t-t_0)(y-y_0)\frac{\partial^2 f}{\partial t\partial y}(t_0,y_0)\right.\\&\left.+\frac{(y-y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0,y_0)\right]+\dots\\&+\left[\frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}(t-t_0)^{n-j}(y-y_0)^j\frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0)\right]\end{aligned}$$

$$R_n(t,y)=\frac{1}{(n+1)!}\sum_{j=0}^{n+1}\binom{n+1}{j}(t-t_0)^{n+1-j}(y-y_0)^j\frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi,\mu)$$

The function $P_n(t,y)$ is called the nth Taylor polynomial in two variables for the function f about (t_0,y_0) and $R_n(t,y)$ is the remainder term associated with $P_n(t,y)$.

The first step in deriving a Runge-Kutta method is to determine values of α_1, δ_1, α_2, and δ_2 with the property of the approximation of a form which its expression is given as follows.

$$f(t+\alpha_1, y+\delta_1 f(t+\alpha_2, y+\delta_2 f(t,y))) \tag{B.2}$$

This approximation is made with error $O(h^3)$, and the most common $O(h^3)$ is Heun's method, which is given as shown below.

$$\begin{aligned} w_0 &= \alpha \\ w_{i+1} &= w_i + \frac{h}{4}\left(f(t_i,w_i) + 3f\left(t_i+\frac{2h}{3}, w_i+\frac{2h}{3}f\left(t_i+\frac{h}{3}, w_i+\frac{h}{3}f(t_i,w_i)\right)\right)\right) \\ &\quad \text{for } i = 0,1,...,N-1 \end{aligned}$$

This Runge-Kutta method is used for higher orders. However, Runge-Kutta Methods for order three are not generally used. The most common Runge-Kutta method in use is of order four in the difference-equation form, which is given by the following subsection.

B.1.1 Runge-Kutta Order Four

The following order of equations uses the order four of Runge-Kutta.

$$w_0 = \alpha \tag{B.3a}$$

$$k_1 = hf(t_i, w_i) \tag{B.3b}$$

$$k_2 = hf\left(t_i+\frac{h}{2}, w_i+\frac{1}{2}k_1\right) \tag{B.3c}$$

$$k_3 = hf\left(t_i+\frac{h}{2}, w_i+\frac{1}{2}k_2\right) \tag{B.3d}$$

$$k_4 = hf(t_{i+1}, w_i+k_3) \tag{B.3e}$$

$$w_{i+1} = w_i + \frac{1}{6}(k_1+2k_2+2k_3+k_4) \tag{B.3f}$$

for each $i = 0, 1, \ldots, N-1$. This method has a local truncation error of $O(h^4)$, provided the solution $y(t)$ has five continuous derivatives. The notations k_1, k_2, k_3, k_4 are introduced into the method to eliminate the need for successive nesting in the second variable of $f(t, y)$.