Universidade de Brasília

Faculdade de Tecnologia

Departamento de Engenharia Elétrica

Neural-network-based model predictive control for consensus of nonlinear systems

Bruno Rodolfo de Oliveira Floriano

TESE DE DOUTORADO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Controle preditivo baseado em redes neurais para consenso de sistemas não-lineares

Bruno Rodolfo de Oliveira Floriano

Tese de Doutorado submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Doutor.

Orientador: Prof. Dr. Henrique Cezar Ferreira
Coorientador: Prof. Dr. João Yoshiyuki Ishihara

Publicação PPGEE 201/23

Brasília

2023

REFERÊNCIA BIBLIOGRÁFICA

Oliveira Floriano, Bruno Rodolfo de (2023). Neural-network-based model predictive control for consensus of nonlinear systems. Tese de Doutorado (Programa de Pós-Graduação em Engenharia Elétrica), Publicação PPGEE 201/23, Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 89 p.

FICHA CATALOGRÁFICA

Oliveira Floriano, Bruno Rodolfo de.

O48n Neural-network-based model predictive control for consensus of nonlinear systems / Bruno Rodolfo de Oliveira Floriano; orientador Henrique Cezar Ferreira; coorientador João Yoshiyuki Ishihara. -- Brasília, 2023.

89 p.

Tese de Doutorado (Programa de Pós-Graduação em Engenharia Elétrica) -- Universidade de Brasília, 2023.

1. Consensus control. 2. Neural networks. 3. Model predictive control. 4. Multi-agent systems. I. Ferreira, Henrique Cezar, orient. II. Ishihara, João Yoshiyuki, coorient.

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Neural-network-based model predictive control for consensus of nonlinear systems

Bruno Rodolfo de Oliveira Floriano

Tese de Doutorado submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Doutor.

Trabalho aprovado. Brasília, 15 de dezembro de 2023:

Prof. Dr. Henrique Cezar Ferreira,
UnB/FT/ENE
Orientador

Prof. Dr. José Alfredo Ruiz Vargas
UnB/FT/ENE
Examinador interno

Prof. Dr. Atila Madureira Bueno,
USP/EP/PTC
Examinador externo

Prof. Dr. Marco Henrique Terra,
USP/EESC/SEL
Examinador externo

Brasília
2023

Este trabalho é dedicado a todos os brasileiros que acreditam na ciência e na tecnologia para melhorar a vida das pessoas

Agradecimentos

Agradeço a minha família por todo o apoio durante minha jornada acadêmica, sempre incentivando o estudo e a persistência para buscar sempre melhorar como pessoa em todos os sentidos. Agradeço todos os meus amigos por me ouvirem e apoiarem em momentos de maior a agonia e ansiedade que vêm em processos longos como um doutorado.

Agradeço meus orientadores pelo conhecimento transmitido e à Universidade de Brasília por todo apoio institucional.

O presenta trabalho só foi possível graças ao apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) que financiou esta pesquisa.

“Não há nada a temer na vida, apenas tratar de compreender.”
(Marie Curie)

“O cientista não é a pessoa que dá as respostas certas,
mas quem faz as perguntas certas.”
(Claude Levi-Strauss)

“Somos os representantes do cosmos. Somos um exemplo
do que o hidrogênio pode fazer com 15 bilhões
de anos de evolução cósmica.”
(Carl Sagan)

Abstract

This work addresses, through a model predictive control (MPC) approach, the consensus problem for discrete-time nonlinear multi-agent systems subjected to switching communication topologies. For systems following a random switching law, there is not any MPC solution that results in a reliable optimization in real-time. We propose a new neural-network-based algorithm that reduces the effects of communication deficiencies, caused by Markovian switching, by approximating and minimizing, in real-time, the MPC's cost function. The convenience of the proposed method is certified in simulations for different applications and scenarios. Finally, the future steps of the current research are detailed.

Keywords: Consensus control. Neural networks. Model predictive control. Multi-agent systems.

Resumo

Este trabalho aborda, através do método de controle preditivo, o problema de consenso para sistemas multiagentes não lineares a tempo discreto sujeitos a topologias de comunicação chaveadas. Para sistemas multiagentes não lineares que seguem uma lei de chaveamento aleatório, não há soluções baseadas em controle preditivo que resultam em uma otimização confiável em tempo real. Nós propomos um novo algoritmo baseado em redes neurais que reduz os efeitos das deficiências de comunicação, causados pelo chaveamento Markoviano, aproximando e minimizando, em tempo real, a função de custo do controle preditivo. A conveniência do método proposto é certificada em simulações para diferentes cenários e aplicações. Finalmente, os passos futuros da atual pesquisa são detalhados.

Palavras-chave: Controle de consenso. Redes neurais. Modelo de controle preditivo. Sistemas multiagentes.

List of figures

List of tables

List of abbreviations and acronyms

MSE	Mean-squared error
AUV	Autonomous underwater vehicles
BIBO	Bounded Input, Bounded Output
MAS	Multi-agent system
MPC	Model predicitve control
NN	Neural network
NNMPC	Neural-network-based model predictive control
UAV	Unmanned aerial vehicle

List of symbols

Roman letters

$\bar{A}_c$	Cumulative area covered
$\bar{A}_n$	Cumulative area interest
$\bar{L}$	Cummulative Laplacian matrix
$\bar{l}_{ij}$	(i,j)-th component of the cummulative Laplacian matrix
$\bar{V}_{cc}$	Communication relative size
$\bar{z}$	Step change in the altitude
$\boldsymbol{D}$	Diffusion tensor
$\boldsymbol{H}_i$	Observation area matrix
$\boldsymbol{q}_i$	2D coordinate of the i-th balloon
$\boldsymbol{v}$	Drift vector
$\boldsymbol{x}$	Position vector
$\lambda(t)$	Communication linear decay function
$\mathcal{A}$	Adjacency matrix
$\mathcal{E}$	Set of edges on a graph
$\mathcal{G}$	Directed graph
$\mathcal{S}$	Set of allowed topologies
$\mathcal{V}$	Set of nodes on a graph
$\mathcal{X}$	Studied region
$\mathscr{F}$	Event space
$\mathscr{F}_k$	Event space filtration over time
A	Discrete-time state matrix
a	Damping constant
A_c	Continuous-time state matrix
a_{ij}	Element (i,j) of the Adjacency matrix
A_n	Remaining interest of the area covered by all the agents
A_T	Total interest of the area
B	Discrete-time control matrix
B_c	Continuous-time control matrix
B_ω	External disturbance matrix
C_{ij}	Routing protocol distance-based cost
C_{min}	Routing protocol minimum cost
d	Number of outputs in a neural network
d	Spring constant
D_{ij}	(i,j)-th component of the diffusion tensor

d_{ij}	Distance between agents i and j
E	Expected value
e	Euler's constant
e	State error vector
E_{NN}^{t}	Error of the t-th sample estimation
e_i	Error between the i-th state and the virtual agent's state
e_i^r	r-th entry of the error vector
f	Nonlineaer function of an agent's dynamics
f_c	Continuous nonlinear function
F_p	Function vector of the MAS dynamics at mode p
f_{pi}	Function of the i-th agent's dynamics at mode p
g	Index of the hidden layer's perceptrons
g	Quadratic function for MPC optimization
h	Predictive horizon
i	Index to identify an agent
i	Neural network input index
J	MPC's predictive cost function
j	Index to identify an agent
j	Neural network output index
k	Discrete time index
K_k^*	Optimal feedback matrix at instant k
K_0	Initial feedback gain matrix
k_f	Last instant of the simulation
K_k	Feedback matrix at instant k
K_k^{ab}	Entry (a,b) of the matrix K_k
L	Laplacian matrix
L	Length of a car's wheelbase
l	Number of inputs in a neural network
l_{ij}	Element (i,j) of the Laplacian matrix
M	Number of perceptrons in a neural network
m	Number of individual control variables
M_{max}	Maximum number of perceptrons for cross-validation
MSE	Mean-squared error
MSE_i^r	Mean-squared error of the r-th element of the i-th state
N	Number of agents in the system
n	Number of individual states
P	MPC's state weight matrix
p	Mode of communication topology
Q	MPC's control weight matrix

q	Mode of communication topology
r	Index of an individual state's entry
r^t	t-th sample of the dataset output
r_i	Radial position of the i-th balloon
R_V	Radius at which V_m is attained
S	Maximum number of allowed topologies
s	Index of probability matrices used in the simulations
T	Cumulative period
T	Sampling period
t	Time value / Sample index in a dataset
t_c	Connection time
t_d	Disconnection time
T_{ac}	Area coverage period
T_{cc}	Communication period
u	Control vector
u^*	Optimal control vector
u_i	Individual control of the i-th agent
U_{in}	Largest possible magnitude of a bounded-input
V	MPC's cost function
v	Vector of neural network intermediate signal
v_i	i-th component of the drift vector
v_i	i-th node of a graph
v_i	Tangential velocity of the i-th balloon
V_m	Maximum tangential wind speed
V_{ac}	Area coverage cost energy
V_{cc}	Communication cost energy
W	Matrix of neural network weights
w	Vector of neural network weights
x	State vector
x_0	State of the virtual agent
x_i	i-th component of the position vector
x_i	Individual state of the i-th agent
x_i^r	r-th entry of the individual state vector
x_i^*	Desired distance between i-th agent and the virtual agent
x_{fi}	Position of the i-th car in the x axis
X_{out}	Largest possible magnitude of a bounded-output
y	Vector of neural network outputs
y_f	Function that is being approximated by a NN
y_{fi}	Position of the i-th car in the y axis

z	Vector of neural network inputs
z_i	Vertical position of the i-th balloon

Greek letters

α	Lipschitz constant
α	Time-averaged radial velocity's vertical gradient
β_{ac}	Weight of the area coverage component
β_{cc}	Weight of the communication component
$\boldsymbol{\mu}$	Position of the hurricane's center
χ	Hurricane parameter for maximum wind values and data in the outer circulation
δ_i	i-th car steering angle
δ_K	Step size for the feedback matrix's entries
ε_{ij}	Difference between i-th and j-th states
η	Learning factor
η_i	Radius of the i-th observation area
Γ	Interest function
γ	Weight of the routers communication
μ_{tk}	Topology mode at time t given k
Ω	Sample space
ω_i	Angular velocity of the i-th balloon
ω_i	External disturbance at the i-th agent
φ	Shape parameter that defines the proportion of pressure in hurricane
φ	Uni-dimensional position of a quadrotor
Φ_i	Coverage influence of the i-th balloon
Π	Probability matrix
Π_s	s-th probability matrix
π_{pq}	Probability of transition from mode p to q
Ψ	Matrix of neural network weights
ρ	Communication range
σ	Interest radius
σ	Neural network's activation function
τ	Time set of the last area coverage period
θ	Mode of the communication topology
θ_i	Angular position of the i-th balloon
θ_{ri}	i-th car heading
ξ	Fluctuations in the radial velocity
ξ	Learning stop condition
ζ	Rate of change of the interest function with linear decay

Contents

1 Introduction

Researchers have successfully applied cooperative control of multi-agent systems (MAS) to handle a broad range of robotics applications, see, for instance, applications in mobile robots (Han, 2018), in formation flight (Floriano *et al.*, 2021), and in networked manipulators (Liu; Li; Guo, 2021), to mention a few. One essential feature studied for cooperative control is to achieve agents' state consensus. Further, developing a protocol that leads to consensus demands a practical design that makes all agents reach an agreement on their states (see, e.g., Lewis *et al.* (2013), Wang *et al.* (2020), Ren, Beard, and Atkins (2007)).

According to Li *et al.* (2019), to reach consistent agreement and consensus, it is necessary to ensure a reliable communication system. In real-time experiments, however, certain physical features can degrade the communication topology, thus generating uncertainties in the information exchange (Savino *et al.*, 2015). In this scenario, consensus should work even if the system is subject to switching topologies (Olfati-Saber; Murray, 2004). Studies suggest using Markov chain to model that switching topology (Wang *et al.*, 2020; Gao *et al.*, 2020; Savino *et al.*, 2015; Ming *et al.*, 2016), even though it is possible to assure consensus when the switching topology is arbitrary (Valcher; Zorzan, 2017; Sakthivel *et al.*, 2019; Kaviarasan *et al.*, 2018). Wang *et al.* (2020), for instance, have studied a class of multi-agent system with Markovian switching topologies for linear dynamics. In this case, consensus is proven to be obtained in terms of linear matrix inequalities. In another work, Gao *et al.* (2020) have obtained similar achievement for systems with disturbances and nonlinearities. It was provided sufficient conditions to obtain consensus in such system by using a event-triggered mechanism. However, both works have solutions based on offline strategies, which might not reflect unknown factors during the operation.

Similar to (Gao *et al.*, 2020), other works have contributed to the study of nonlinear MAS with switching topologies once nonlinear systems are widely present in important MAS applications. For instance, (Liu; Huang, 2017) developed an adaptive control technique for the leader-following consensus problem for a class of nonlinear MAS subjected to switching topologies and external disturbances. Although it provides a great solution for higher-order nonlinearities, the communication switching is still modeled with deterministic functions. Another relevant work, (Zou *et al.*, 2019) has investigated random switching applied for MAS with nonlinear dynamics (including manipulators), successfully proving the event-triggered efficiency in tracking the leader with no Zeno behavior. Nevertheless, the switching mechanism was applied to the individual dynamics, not the communication topology.

Due to their flexibility in dealing with unknown parameters, and their significant capacity to estimate nonlinear functions, machine learning algorithms have been used for consensus control in online strategies (Li *et al.*, 2020; Zhang; Zhang; Feng, 2017; Zhao *et al.*,

2017). For instance, Zhang, Zhang, and Feng (2017) used neural network-based adaptive dynamic programming to solve the consensus problem for systems with unknown dynamics. In another study, Zhao *et al.* (2017) have created an adaptive neural network (NN) controller for finite-time consensus problems in uncertain MAS. Note that both studies have systems modeled for fixed topologies only. By contrast, Li *et al.* (2020) have contemplated a time-varying topology. In summary, all of these studies have designed their systems for deterministic topologies.

On the other front, stochastic topology has allowed researchers to expand the application of consensus in multi-agent systems. For instance, controllers based on neural networks have provided significant results with Markov jumps in single-agent systems. Zhong *et al.* (2015) have proposed an online system based on neural network optimization of an adaptive dynamic programming protocol for discrete-time nonlinear Markov jump systems (MJS). In another study, Yang, Yin, and Kaynak (2020) have applied neural networks to estimate nonlinearities in fault-tolerant MJS. It is then reasonable to extend the application of neural-network-based methods for MAS subject to the stochastic topology. This work contributes to this direction, as detailed next.

Recall that model predictive control (MPC) has been popular in industrial applications due to its predicting characteristics and capacity of working with varying operating conditions, providing performance robustness (Cheng *et al.*, 2015; Han *et al.*, 2015; Namara *et al.*, 2013). Recently, researchers have expanded the knowledge of MPC to deal with neural networks and MAS (Wang; Gao; Qiu, 2015; Chen *et al.*, 2018; Xiao; Chen, 2018). While only Xiao and Chen (2018) have studied MAS with switching topologies, the consensus problem was solved only for a particular system (wheeled robots) and the switching between topologies was modeled as deterministic and not stochastic. Indeed, to the best of the author's knowledge, there is not any work in the literature that applies MPC technique for the consensus protocol design of nonlinear MAS with Markovian switching topology. Studying the stochastic effects of Markovian models on MPC-based MAS is particularly relevant to provide practical applicability since these models are convenient tools to represent real communication imperfections (such as disconnection, noise and disturbances, preeminent adversities faced in the current consensus problem).

1.1 Contributions

The main contribution of this research is to show a novel neural-network-based model predictive control for the consensus problem of nonlinear multi-agent systems with Markovian switching topology. In this approach, the cost function of MPC considers the expected value of a certain Lyapunov function that in turn accounts for the error of the system. The function cost also weights the behavior of the communication topology. A feedforward neural network is used to adaptively approximate the cost function of MPC and,

by minimizing it, obtain the solution for the corresponding control problem. Simulations with different systems are performed in order to verify the protocol's effectiveness as well as to compare it with other existing methods such as the approaches developed by Wang *et al.* (2020) and Gao *et al.* (2020). Finally, the approach's practical relevance is illustrated in numerical evaluations by verifying the proposed method in disturbed systems as well as in a group of quadrotors.

To summarize, this work contributes in the following:

- For nonlinear MAS with MPC-based protocol design, this study expands the class of allowed topologies in the literature. Here, Markovian switching topologies are allowed, contrasting to deterministic switching topologies of (Xiao; Chen, 2018);
- The difficulty of MPC optimization caused by random processes is sorted out by a novel online neural-network-based algorithm that mitigates the effects of communication failures;
- The developed algorithm advances the studies of nonlinear multi-agent consensus with random switched topologies of (Gao *et al.*, 2020) by introducing adaptive neural networks that improve the system capacity to handle unknown external changes;
- The proposed protocol can be applied to a broad class of nonlinear systems due to the neural network estimation properties. As exemplified in simulations, the class of allowed nonlinear dynamics enables us to deal with important practical nonlinear systems such as quadrotors and robot cars and encompass those dynamics considered in the work of (Xiao; Chen, 2018), limited to wheeled robots, and of (Gao *et al.*, 2020), which is restricted to a specific class of nonlinearities.

Moreover, our research has been rigorously tested in a larger and more intricate application, exemplified by the monitoring of hurricanes using a swarm of buoyancy-driven balloons. This comprehensive case study, as presented in chapter 5, serves as a litmus test for our architecture, particularly in the context of a problem characterized by conflicting objectives: the simultaneous optimization of area coverage and communication. This dual-goal challenge enables us to showcase not only the real-time adaptability of the NN-based MPC approach but also the model's ability to contend with a highly noisy and disturbance-prone environment, replete with communication constraints. The outcomes and findings presented in this chapter also represent a significant advancement in the existing literature for several reasons, which will be further elaborated, but can be summarized as following:

- Expands the buoyancy-driven balloon control in a stratified flowfield of Meneghello, Luchini, and Bewley (2016), Meneghello, Luchini, and Bewley (2018) to multiple agents which increases area coverage and data collection;
- Builds an effective trade-off between area coverage and repositioning for connection by optimizing a weighted cost function via the neural-network-based approach of

Floriano *et al.* (2022);

- Establishes a constant restoring interest function, driven by the Fokker–Planck equation, to characterize the requirement of updating a previously covered region after some time

This document is organized as follows: chapter 2 provides the theoretical background on the main topics that are relevant to the proposed work, which includes: graph theory, neural networks, model predictive control, and the overall description of the problem to be solved; chapter 3 presents the proposed consensus protocol based on MPC and neural network; chapter 4 illustrates the results of the proposed method in simplified models; chapter 5 evaluates the performance of the proposed architecture in a more complex operation, exemplified by the hurricane monitoring application; and finally chapter 6 presents the final remarks.

2 Theoretical background

2.1 Multiagent systems

2.1.1 Graph theory

A directed graph, denoted by $\mathcal{G}(\mathcal{V},\mathcal{E},\mathcal{A})$, is composed by a set of nodes $\mathcal{V} = \{v_1,\ldots,v_N\}$, a set of edges $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$, and an adjacency matrix $\mathcal{A} = [a_{ij}]_{N\times N}$. The Laplacian matrix $L = [l_{ij}]_{N\times N}$ corresponding to $\mathcal{G}(\mathcal{V},\mathcal{E},\mathcal{A})$ is defined as $l_{ii} = \sum_{j=1,j\neq i}^{N} a_{ij}$ and $l_{ij} = -a_{ij}$, if $i \neq j$ (Ren; Beard; Atkins, 2007; Bapat, 2010).

For systems with switching topologies, the graph is denoted as $\mathcal{G}(\theta(k)) = (\mathcal{V},\mathcal{E}(\theta(k)), \mathcal{A}(\theta(k)))$ and the Laplacian matrix $L(\theta(k))$ of the switching network is built as $l_{ii}(\theta(k)) = \sum_{j=1,j\neq i}^{N} a_{ij}(\theta(k))$ and $l_{ij}(\theta(k)) = -a_{ij}(\theta(k))$, if $i \neq j$. Where the variable $\theta(k)$ represents the mode of the communication topology at step k, takes values in the finite set $\mathcal{S} = \{1,2,\ldots,S\}$, and follows a Markov chain with probability matrix

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & \ldots & \pi_{1S} \\ \pi_{21} & \pi_{22} & \ldots & \pi_{2S} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{S1} & \pi_{S2} & \ldots & \pi_{SS} \end{bmatrix}. \tag{2.1}$$

The value π_{pq} indicates the probability of transition from p to q, $p,q \in \mathcal{S}$, and we have $\sum_{q=1}^{S} \pi_{pq} = 1$ for each $p \in \mathcal{S}$.

2.1.2 Consensus

Let us consider a multi-agent system (MAS) composed of N discrete-time nonlinear subsystems:

$$x_i(k+1) = f(x_i(k),u_i(k)), \tag{2.2}$$

where $x_i(k) \in \mathbb{R}^n$ and $u_i(k) \in \mathbb{R}^m$ denote, respectively, the state and the input signal of the i-th subsystem ($i \in \{1,\ldots,N\}$) at time $k \in \mathbb{Z}$. The dynamics of the communication topology follows a Markovian process on a fixed, filtered probability space $(\Omega,\{\mathcal{F}_k\},\mathcal{F},\Pi)$. The set $\mathcal{F}_k$ denotes the filtration corresponding to the σ-field generated by $\{x(0),\theta(0),\ldots,x(k),\theta(k)\}$ (see, e.g., Costa and Fragoso (1995)). The mode of the topology at time t, conditioned to the filtration up to the step $k > 0$, is determined by the probability distribution $\mu_{t|k}(p) = \Pr(\theta(t) = p|\mathcal{F}_k)\ t > k$.

Notice that equation (2.2) supports to extent the system's deficiencies beyond the Markovian communication topology. Accordingly, it is possible to model other imperfections

in the MAS or the environment, including other types of random switching, delay or disturbances. Indeed disturbances will be considered in the numerical experiments, indicating the scope of practical possibilities. In addition, it is possible to observe in equation (2.2) that, due to the generality of the function $f(x_i(k),u_i(k))$, the range of possible nonlinear applications is wider than in previous works (such as Gao *et al.* (2020), Xiao and Chen (2018)), which will be reinforced by the numerical evaluations in chapter 4.

Assumption 2.1. The system described in equation (2.2) is BIBO (Bounded Input, Bounded Output) stable. This means that, if the input is bounded, i.e. $\exists U_{in} \in \mathbb{R}^+$, s.t. $|u_i(k)| \leq U_{in}$, $\forall k \in \mathbb{Z}$, then the output will also be bounded, i.e.

$$\exists X_{out} \in \mathbb{R}^+ \text{ s.t. } |x_i(k)| \leq X_{out}, \ \forall k \in \mathbb{Z}. \tag{2.3}$$

It is noteworthy that assumption 2.1 introduces the critical consideration of BIBO (Bounded Input, Bounded Output) stability for the individual system delineated in equation (2.2). This stability assumption enables the exploration of group dynamics behavior and facilitates the pursuit of consensus within a noisy environment characterized by dynamically switching communication topologies. The adherence to individual BIBO stability is also pivotal for the effective operation of the proposed method because it relies on predictive control and artificial intelligence employing a real-time learning mechanism. This assumption underpins the reliability of the methodology in the face of dynamic interactions and varying communication patterns.

Based on the information that each individual agent has on its neighbours (which depends on the topology at each instant), the consensus control protocol takes the form (e.g., Savino *et al.* (2015), Ming *et al.* (2016))

$$u_i(k) = -K_k \sum_{j=1}^{N} a_{ij}(\theta(k))\varepsilon_{ij}(k), \quad \forall k \geq 0, \tag{2.4}$$

where $K_k \in \mathbb{R}^{m\times n}$ represents the feedback gain matrix and $\varepsilon_{ij}(k) \equiv x_j(k) - x_i(k)$. The structure of the communication and control of each i-th agent is shown as a block diagram in figure 2.1.

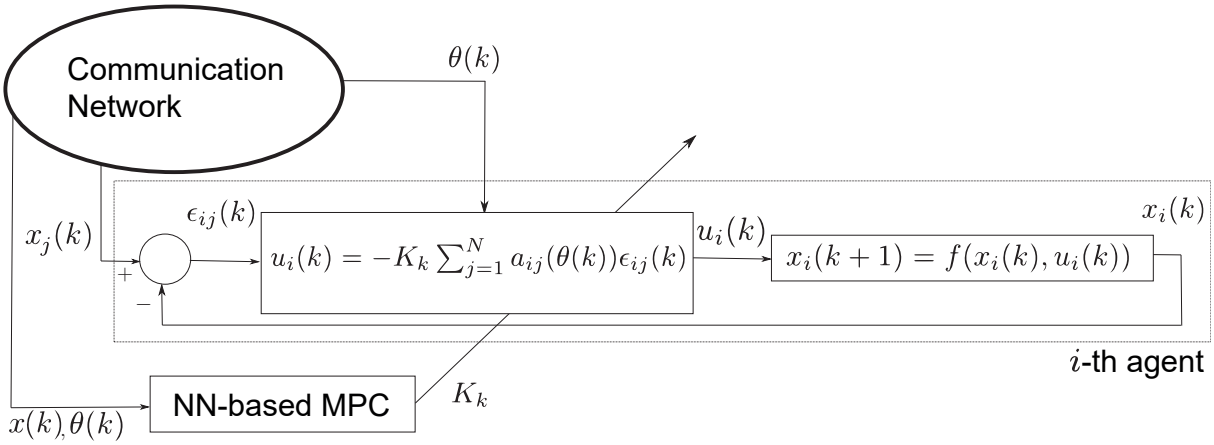
A single control vector can be obtained by stacking the individual signals, i.e. $u(k) = \begin{bmatrix} u_1^T(k) & \dots & u_N^T(k) \end{bmatrix}^T$. Similarly, one has the state $x(k) = \begin{bmatrix} x_1^T(k) & \dots & x_N^T(k) \end{bmatrix}^T$. It follows that the consensus protocol stated in equation (2.4) is represented for the overall cooperative system by

$$u(k) = (L(\theta(k)) \otimes K_k)\, x(k). \tag{2.5}$$

Equation (2.5) expresses that the control vector $u(k)$ is a function of the state $x(k)$ and the mode $\theta(k)$. Let $\theta(k) = p$, then the MAS state can be expressed as

$$x(k+1) = F_p(x(k)) = \begin{bmatrix} f_{p1}^T(x(k)) & \dots & f_{pN}^T(x(k)) \end{bmatrix}^T, \tag{2.6}$$

Figure 2.1 – Block diagram of a MAS consensus control with switching topology



where $f_{pi}(x(k))$ is the function $f(x_i(k),u_i(k))$ in equation (2.2) with the control given by equation (2.4) in mode p.

Assumption 2.2. The nonlinear function $F_p(x)$ satisfies the Lipschitz condition with scalar $\alpha > 0$, i.e.,

$$||F_p(x) - F_p(y)|| \leq \alpha||x - y||, \tag{2.7}$$

for all $x,y \in \mathbb{R}^{nN}$ and for all $p \in \mathcal{S}$.

The Lipschitz condition provides that the multiagent system dynamics $F_p(x(k))$ has a uniform continuity property (Sohrab, 2003) assuring a limited growth. This property is a reasonable assumption since any real platform has limitations for sharp changes (with infinite derivatives) and is common in MAS literature (see, e.g. Gao *et al.* (2020), Dong and Nguang (2020), Li and Duan (2017)).

Let $e_i(k) = x_i(k) - x_0(k)$ be the error between the states of the i-th agent and of a virtual agent, which is driven the following dynamics:

$$x_0(k+1) = f(x_0(k),0). \tag{2.8}$$

One can stack the errors into the vector $e(k) = [e_1^T(k) \dots e_N^T(k)]^T$ to obtain the recurrence

$$e(k+1) = F_p(x(k)) - \mathbf{1_N} \otimes f(x_0(k),0), \tag{2.9}$$

where $\mathbf{1_N}$ is the vector composed by N entries of 1.

The main problem to be solved for the MAS under analysis is consensus, which then requires a proper definition. A system is said to achieve consensus when certain information (in this case, the individual state $x_i(k)$) is the same for all the agents. In other words, the difference between any pair of individual states must reach zero. However, since the system under study has stochastic properties, the expected value of that difference (given the initial conditions) is applied. Therefore, the formal definition can be stated as follows:

Definition 2.1. We say the MAS dynamics (2.2) subject to (2.4) reaches consensus if

$$\lim_{k\to\infty} E\left[||x_j(k) - x_i(k)|| \,|\, x(0), \theta(0)\right] = 0 \tag{2.10}$$

holds true for all pairs (i,j), with $i \neq j$, and for all $x(0) \in \mathbb{R}^n$ and all $\theta(0) \in \mathcal{S}$.

Although works such as Cheng *et al.* (2014), Su, Shi, and Sun (2019) have developed MPC algorithms that directly obtain the input signal, there are still some significant modeling differences with our approach. None of those works have modeled the systems with Markovian switching topologies (or any random switching whatsoever). Another relevant contrast is that our framework is suited for nonlinear functions. Therefore, the solutions mentioned above do not apply to the designed system. Given those differences, it was proposed that a neural network generates the feedback gain to solve the consensus problem.

Note that there may be infinite sequences of gains $\{K_k\}_{k=0}^{\infty}$ that lead to consensus. However, finding even one such sequence is not an easy task. In the next section, we propose to obtain one optimal sequence of gains that leads to consensus through the MPC framework based on a neural network.

2.2 Neural Networks

2.2.1 Introduction

Neural networks (NN) are machine learning models inspired by brain behavior to approximate a given function and ultimately perform a classification or regression task (Alpaydin, 2020).

The NN's most basic processing element, the perceptron, was designed to mimic the behavior of the nerve cell, which gets electrically excitable if a linear combination of inputs reaches a particular value (Russel; Norvig, 2012). By using multiple perceptrons in parallel is possible to obtain the classical structure of a feedforward neural network. In addition, other NN structures can be built depending on the application, among them there is the multilayer NN, recurrent NN (RNN), radial basis function (RBF) NN, convulition NN (CNN), Long Short Term Memory (LSTM) and others (LeCun; Bengio; Hinton, 2015).

In the last decade, NN's use has been spreading to wide variety of applications. Its most recurrent implementation has been for classification, recommendation algorithms and computer vision (see Lee and Shin (2020), Abiodun *et al.* (2018)). Popular frameworks such as PyTorch and Tensorflow are among the most used in the current software development industry.

The success of the NNs is due to their universal approximation property. This property states that if there is a sufficient number of parameters, then it can approximate any function

(Alpaydin, 2020). What makes it specially interesting is that such accuracy is possible even though there is no prior knowledge on the function structure, i.e. NN is a nonparametric machine learning algorithm (e.g. it is not necessary to know if the function is linear, sinusoidal, exponential and etc).

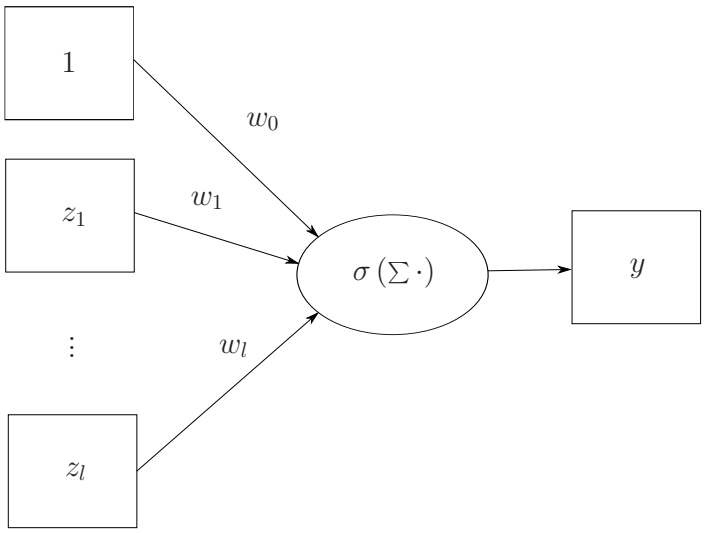
In the robotics context, specially for MAS applications, it has been use for multiple objectives such as estimate states and outputs, learn unknown nonlinearities or for optimization purposes (see Choi and Cha (2019)).

2.2.2 Neural network structure

2.2.2.1 The perceptron

The perceptron (or neuron) is the basic element that builds up a neural network. It has a set of $l + 1$ inputs (that may be associated with the dataset or can be the output of other neurons), activation function σ and an output y. The graphical representation of a perceptron can be seen in figure 2.2.

Figure 2.2 – Representation of a perceptron



Consider the vector $z \in \mathbb{R}^{l+1}$ as follows

$$z = \begin{bmatrix} 1 & z_1 & \dots & z_l \end{bmatrix}^T. \tag{2.11}$$

The inclusion of a 1 in the vector is to include a term for bias, which makes the model more general (see Alpaydin (2020)). There is a weight w_i for each input z_i, as well a bias weight w_0. Therefore, the weighted sum, y', is given as

$$y' = \sum_{i=1}^{l} w_i z_i + w_0 = wz, \tag{2.12}$$

considering $w = \begin{bmatrix} w_0 & \dots & w_l \end{bmatrix}$.

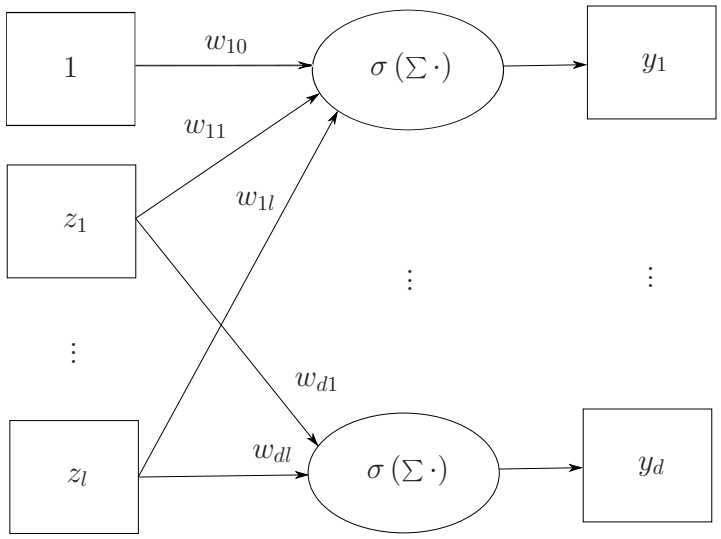
Finally, the output of the perceptron requires an activation function, i.e. a function that defines if the neuron is excitable, and if so, the activation extent. Choosing such function depends on the application, e.g. sigmoid and softmax are used for classification, while linear and Rectified Linear Unit (ReLU) are used for regression tasks (Alpaydin, 2020). For now, we will leave the activation function as $\sigma(\cdot)$ in order to be chosen later as the application demands. Therefore, the output y of the perceptron in given as

$$y = \sigma(y') = \sigma\left(wz\right). \tag{2.13}$$

2.2.2.2 Parallel perceptrons

If the application requires multiple outputs, it is possible to add several perceptrons in parallel as seen in figure 2.3.

Figure 2.3 – Representation of perceptrons in parallel



$$y_j = \sigma\left(\sum_{i=1}^{l} w_{ji} z_i + w_{j0}\right). \tag{2.14}$$

In this case, there is a weight w_{ji} for each conection between the j-th perceptron and i-th input. Each perceptron can result in an output if the application requires (generally for classification applications).

Building a matrix $W = [w_{ji}]$, $j \in \{1, \ldots, d\}$, $i \in \{0, \ldots, l\}$ then

$$y = \sigma(Wz), \tag{2.15}$$

where $y = \begin{bmatrix} y_1 & \ldots & y_d \end{bmatrix}^T$.

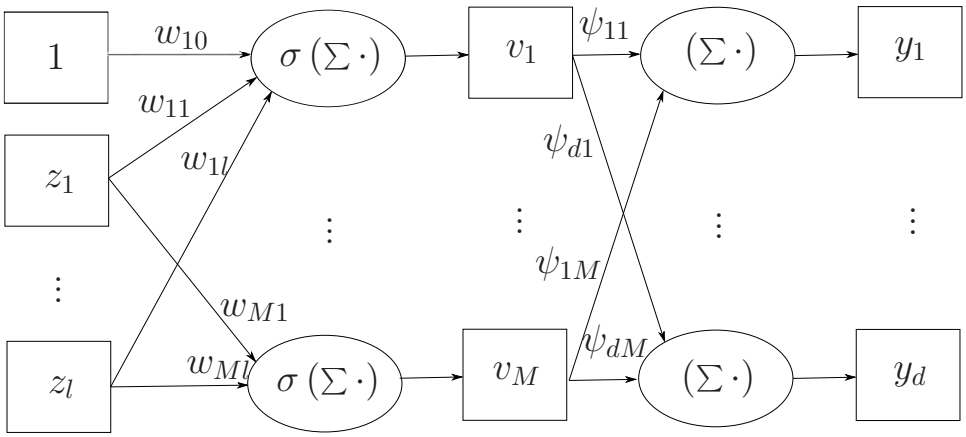
2.2.2.3 Multilayer neural networks

The perceptron structures seen so far has only a single layer of weights, which are not able to perform nonlinear classification or regression (Alpaydin, 2020). To solve this

problem, a common solution is to add an intermediate layer between the input and the output, the so called hidden layer, due to its position in the structure.

In addition, it is possible to perform another weighted sum, to generate a single output. This structure is the most basic form of neural network, called Shallow Neural Network. It is said that this structured is constructed with 3 layers: the input layer, the hidden layer (the perceptrons), and the output layer (figure 2.4).

Figure 2.4 – Representation of a shallow neural network



Consider the intermediate signal $v \in \mathbb{R}^g$, in which $g \in \{1, \ldots, M\}$, given by

$$v = \sigma(Wz), \tag{2.16}$$

where $W \in \mathbb{R}^{M\times(l+1)}$ is the neurons weight matrix and M is the number of hidden perceptrons. It is common practice that the the output layer is only the linear combination of the the last layer (without any activation function), since the nonlinearity part of the NN is already present in the hidden layer. Therefore, the output vector of the multylayer NN is given by

$$y = \Psi v = \Psi\sigma(Wz), \tag{2.17}$$

with $\Psi \in \mathbb{R}^{d\times M}$ is the output weight matrix and d the number of outputs.

More hidden layers can be included between the input and output layer, which makes the NN's structure more complex, but capable of approximating highly nonlinear and discontinuous functions to fit a desired functionality (see Russel and Norvig (2012)). However, considering the scope of this work, a single layer is sufficient for the dynamic systems and online controller to be presented in the chapter 3.

2.2.2.4 The universal approximation property

Feedforward neural networks as shown so far hold the universal approximation property. This theorem states that any arbitrary function can be approximated with a multilayer neural network, provided that a sufficient number of perceptrons are available (see Hornik, Stinchcombe, and White (1989), Alpaydin (2020)).

Assumption 2.3. The function $y_f : \mathbb{R}^l \rightarrow \mathbb{R}^d$, which is being approximated by a neural network in a compact input set, is continuous across that set.

The condition specified in assumption 2.3 asserts that the function y_f, which a neural network endeavors to approximate in compact input set, is presumed to exhibit continuity across that set. This assumption is consistent with the criteria established in the literature (Hornik, Stinchcombe, and White (1989), Hornik (1991), Alpaydin (2020)) for the universal approximation property to be valid, particularly in the context of a shallow neural network with a single hidden layer. With this continuity condition met, the neural network is capable of approximating y_f with an arbitrary number of neurons, illustrating the network's ability to effectively capture and represent the underlying continuous function. This adherence to continuity aligns with established principles for achieving universal approximation capabilities within the specified neural network architecture.

This property has great significance in machine learning applications, because it provides high approximation accuracy without a previous knowledge on the function characteristics (since NNs are nonparametric algorithms).

Therefore, NNs are great tools that have been used in many scientific and engineering fields, including robotics, as they can estimate states, outputs, nonlinearities, disturbances, uncertainties and other functions while modeling a system or designing a controller.

Neural networks have the universal approximation property, which means they are well suited to approximate functions (Zhang; Zhang; Feng, 2017). For that reason, the NNs are used in this work to approximate the relation of the function $J(e(k))$ with the sequence of feedback gain matrices $\{K_k, \ldots, K_{k+h}\}$.

2.2.3 Learning with backpropagation

Learning, in the context of neural networks, is to derive the values for the weights that better fit the dataset. In the shallow NN, it means to find the matrices W and Ψ.

The error function used to verify the learning process differs with the application. For classification purposes, cross-entropy is preferable. However, for regression problems (as it is our case), the mean squared error is used more often. Therefore, in a single perceptron the error of the t-th sample of the dataset is stated as

$$E_{NN}^t(w|z^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}(r^t - (wz^t))^2, \tag{2.18}$$

where r^t is the dataset output, y^t is the predicted output and z^t is the dataset input.

The online update law that adjusts the weights is given as

$$\Delta w_i^t = \eta(r^t - y^t)z_i^t, \tag{2.19}$$

where η is the learning factor, which generally decreases during the process for a better convergence.

The learning process to find the weights that guarantees the minimum error is called *stochastic gradient descent* (Alpaydin, 2020). In this method, the gradient of the weights, shown in equation (2.19), are computed so that the weights are adjusted in the "direction" of decreasing the error. The process continues until a point of local minimum is reached.

In the case of multilayer NN, on top of the gradient descent, the learning process uses the backpropagation algorithm, name given due to its process of propagating the error from the output y back to to the input, passing through the hidden layer. The error is given by

$$E_{NN}(W,\Psi|z,r) = \frac{1}{2}\sum_t\sum_j (r_j^t - y_j^t)^2. \tag{2.20}$$

The output weights are trained the same way a single perceptron, as

$$\Delta\psi_{jg} = \eta\sum_t (r_j^t - y_j^t)v_g^t, \tag{2.21}$$

while the weights of the hidden layer are adjusted with

$$\Delta w_{gi}^t = \eta\sum_t \left[(r_j^t - y_j^t)\psi_{jg}\right] v_g^t(1-v_g^t)z_i^t. \tag{2.22}$$

2.3 Model predictive control

Model predictive control (MPC) is a control strategy, based on predicting the future behavior of a dynamic system, subjected to a set of constraints, and computing a control signal that optimizes a cost function within a given horizon (Kouvaritakis; Cannon, 2016).

Consider the following dynamic:

$$x(k+1) = f(x(k),u(k)), \tag{2.23}$$

with the constraints $x(k) \in \mathcal{X}$, $u(k) \in \mathcal{U}$.

The MPC problem is to optimize a cost function J, such as

$$J = \sum_{k=0}^{h} g(x(k),u(k)). \tag{2.24}$$

in which h is the horizon. Ideally, the horizon should go to infinity, or as large as possible, however due to processing limitation it is often set with a value sufficiently large for the application. In general, g is chosen as a quadratic function to guarantee the convexity of the problem, and consequently, its convergence. Hence, g usually is written as

$$g(x(k),u(k)) = x^T(k)Px(k) + u^T(k)Qu(k), \tag{2.25}$$

with P and Q being respectively the state and control weight matrices.

Therefore, solving an MPC is to solve an optimization problem of the form

$$\text{minimize } J = \sum_{k=0}^{h} g(x(k),u(k)), \tag{2.26}$$

$$\text{subject to } x(k) \in \mathcal{X}, u(k) \in \mathcal{U}. \tag{2.27}$$

The result of that optimization is a sequence of control inputs $u^*(0), u^*(1), \ldots, u^*(h)$, that if applied in the system will result in the minimum cost J.

When the algorithm runs online, a different optimization is performed at each time step. So at time $k = t$, the minimization can be stated as

$$\text{minimize } J = \sum_{k=t}^{t+h} g(x(k),u(k)), \tag{2.28}$$

$$\text{subject to } x(k) \in \mathcal{X}, u(k) \in \mathcal{U}, \tag{2.29}$$

which results in the optimal control $u^*(t)$.

3 Neural-network-based model predictive control for consensus of nonlinear multi-agent systems

3.1 Introduction

Model predictive control (MPC) is one of the most accepted control strategies, and it has been historically used in several industrial applications (Kouvaritakis; Cannon, 2016). One of the reasons for its great acceptance is the compromise between optimality and computation speed, i.e., it is not necessary for too powerful hardware to compute the control signal that will lead to the application's objective. In addition, MPC holds the convenience of providing some degree of robustness to uncertainties in the model as well as to system constraints (Su; Shi; Sun, 2019). Those characteristics are highly valuable for multiagent applications, such as unmanned aerial vehicles (UAVs) or autonomous underwater vehicles (AUVs), since they are inserted in noisy environments with limited processing or communication capacity while still requiring a reliable level of precision. Accordingly, as one of the most fundamental issues in MAS, the consensus problem has been greatly developed with MPC approaches (as seen in works of Ferrari-Trecate *et al.* (2009), Cheng *et al.* (2014), Su, Shi, and Sun (2019)). Therefore, it would be straightforward to use this control scheme for solving the consensus problem described in chapter 2.

However, while MPC-based protocols include a variety of advantages, there are still some drawback to their use. For instance, MPC requires a knowledge of the model dynamics to provide the prediction up to a given horizon, which jeopardizes applications where the model is uncertain or changes with time (e.g., switching systems). Moreover, if the system has nonlinear characteristics, the MPC's computational burden increases, discarding the optimization trade-off advantage of the protocol (Venkatesan; Kamaraj; Vishnupriya, 2020).

In that context, neural networks (NN) are valuable solutions for those MPC drawbacks. For starters, NNs hold the universal approximation property, which assigns them the possibility to estimate any function. Therefore, NNs can increment MPC by estimating the model functions and other uncertainties, removing the immediate necessity of the knowledge of the system dynamics. Furthermore, that estimation can be extended to nonlinearities, maintaining the optimization with an acceptable computational burden. In addition, NNs are suited for learning in real-time, which can adapt the model for systems that can change with time, such as switching systems. Algorithms with NN-based MPC have already been tested in the literature in works such as Wang, Gao, and Qiu (2015), Wu, Rincon, and

Christofides (2020), showing relevant results.

While significant results have been achieved for single-agent applications, the only contribution of an NN-based MPC protocol in the context of multiagent systems has been developed by Xiao and Chen (2018). However, some issues can be stated in such work: the communication topology and the class of systems. By modeling the communication topology with deterministic switching instead of stochastic, the work restricts the application to specific scenarios in which the topology switching is already established. Modeling random switching allows a better representation of real environmental characteristics. In addition, the protocol was developed for a specific class of systems (wheeled robots), which reduces the range of applications.

In this context, our method solves the consensus problem of a generic MAS (the dynamics can be generalized to a broad class of robots, even with nonlinear characteristics) with Markovian switching topology (the communication between agents changes randomly). The solution is an algorithm built with an MPC framework powered by an NN-based optimizer. The NN is applied to estimate the cost function of the expected states within a given horizon. This approximation allows the system to search for the minimized cost even though the system is subjected to random processes.

In this chapter, a model predictive control (MPC) algorithm based on a neural-network (NN) is presented in order to obtain an optimal feedback gain matrix K_k that makes the MAS reach a consensus in the sense of definition 2.1. First, we present conditions on the nonlinear system that will ensure the feasibility of the MPC consensus problem. Later on, the protocol for achieving consensus will be presented by applying a NN-based optimization procedure inside an MPC framework.

3.2 Conditions for consensus

In this section we will derive the conditions for achieving consensus of the multiagent system subjected to Markovian switching communication described in the previous chapter.

Lemma 3.1. Assume that there exist a gain sequence $\{K_k\}_{k=0}^{\infty}$ and a constant $\alpha < 1$ such that assumption 2.2 holds for all k. Then the multi-agent system with individual dynamics (2.2) subject to (2.4) reaches consensus.

Proof. Note that $F_p(\mathbf{1_N} \otimes x_0(k)) = \mathbf{1_N} \otimes f(x_0(k),0)$. Then, from (2.7) with the identifications $x \leftarrow x(k)$ and $y \leftarrow \mathbf{1_N} \otimes f(x_0(k),0)$, we have, for all k, that

$$||F_p(x(k)) - \mathbf{1_N} \otimes f(x_0(k),0)|| \leq \alpha||x(k) - \mathbf{1_N} \otimes x_0(k)||, \quad \forall p \in \mathcal{S}. \tag{3.1}$$

Considering both the definition of the error $e(k)$ and (2.9), we have from (3.1) that

$$||e(k+1)|| \leq \alpha ||e(k)||. \tag{3.2}$$

Using the assumption that there exits K such that $\alpha = \alpha(K) < 1$, we can write

$$||e(k)|| \leq \alpha^k ||e(0)|| \ \rightarrow 0 \quad \text{as} \quad k \rightarrow \infty. \tag{3.3}$$

The result then follows because

$$x_j(k) - x_i(k) = e_j(k) - e_i(k), \quad \forall i,j = 1,\ldots,N,$$

and (3.3) assures that each error component $e_i(k)$ tends to zero as k tends to infinity. □

Remark 3.1. Lemma 3.1 allows us to obtain consensus if one can restrict the dynamic system (2.2) to find a convenient sequence of gains such that assumption 2.2 is satisfied. However, one can note that finding a convenient sequence of gains is still a challenging task even within this restricted class of dynamic functions. Therefore, in the following section, we propose to search for the best candidate for the sequence of gains through an MPC framework.

3.3 Model predictive control

Let us define the model predictive control (MPC) problem. For this purpose, consider the cost function $V(e(k),\theta(k))$ as

$$V(e(k),\theta(k) = p) = e^T(k) P_p e(k), \tag{3.4}$$

where P_p, $p \in \mathcal{S}$, denotes a positive-definite matrix of dimension n (to be defined later). The cost (3.4) weights the error of $e(k)$ while the topology mode is at $\theta(k) = p \in \mathcal{S}$.

Let

$$J(e(k)) = \sum_{t=k+1}^{k+h+1} E_{x_k,\mu_{t|k}} \left[V(e(t),\theta(t)) \right], \tag{3.5}$$

where $E_{x_k,\mu_{t|k}}[.] \equiv E[.|x(k) = x_k, \theta(t) \sim \mu_{t|k}]$.

The objective of the MPC is to find K such that the predicted cost $J(e(k))$ is minimized. The optimal feedback gain matrix, say K_k^*, satisfies

$$K_k^* = \arg_1 \min_{K_k,\ldots,K_{k+h}} J(e(k)), \tag{3.6}$$

where $\arg_1$ denotes the first argument of the sequence of the gains $\{K_k,\ldots,K_{k+h}\}$.

Several algorithms are possible to solve the optimization problem (3.6). However, due to the nonlinear nature of the involved functions, an online optimization is usually

prohibitive for most optimization algorithms. In the next subsection, we propose a simplified (yet online) solution for the "best" gain K_k based on neural network.

Notice that the cost function is subjected to random processes due to the switching communication topologies. The current MPC solutions for MAS face difficulties during optimization due to such stochastic effects. The proposed solution for this issue is to perform a neural-network-based cost function estimation to reduce the fallout of communication deficiencies. The protocol is detailed next.

3.4 Neural network optimization

Neural networks have the universal approximation property, which means they are well suited to approximate functions (Zhang; Zhang; Feng, 2017). For that reason, the NNs are used in this work to approximate the relation of the function $J(e(k))$ with the sequence of feedback gain matrices $\{K_k, \ldots, K_{k+h}\}$.

To do the optimization stated in equation (3.6), a shallow neural network is used. Shallow NNs are composed of at least three layers: one input layer, at least one hidden layer, and one output layer. In this work, only the value of one gain (of the gain K_k) is sought (by setting $K_{k+h} = \ldots = K_{k+1} = K_k$), and we use only one hidden layer. According to the optimization purpose, the $l = mn$ entries of $K_k = [K_k^{ab}]$, $a \in \{1, \ldots, m\}$, $b \in \{1, \ldots, n\}$ form the input layer.

The output layer is formed by the value of $J(e(k))$. The hidden layer is composed of M neurons to be defined now. Let

$$z = \begin{bmatrix} 1 & K_k^{11} & \ldots & K_k^{1n} & \ldots & K_k^{m1} & \ldots & K_k^{mn} \end{bmatrix}^T \in \mathbb{R}^{l+1}. \tag{3.7}$$

The hidden layer results in the signal $v \in \mathbb{R}^M$ given by

$$v = \sigma(Wz), \tag{3.8}$$

where $W \in \mathbb{R}^{M \times (l+1)}$ is the neurons weight matrix and σ is the sigmoid function. The output layer approximation $\tilde{J}$ for the cost function, given the parameters of K_k, is

$$\tilde{J} = \Psi v, \tag{3.9}$$

with $\Psi \in \mathbb{R}^{1 \times M}$ is the output weight matrix. The learning process of the NN is based on optimizing the weights of W and Ψ that better fit the data.

3.5 NN-based MPC protocol for consensus

Provided the equations for the MPC prediction cost and the NN approximation of J, the control algorithm can be stated. Algorithm 3.1 provides the pseudocode of the NN-based MPC control protocol. All codes were written in Matlab [1].

[1] Available at https://github.com/brunofloriano/NN_MPC

Algorithm 3.1 NN-based MPC pseudocode for consensus control

System Initialization at $x(0), \theta(0)$
$\hat{J} = J_0 = V(e(0), \theta(0))$
$K_0^{ab} = 0 \; \forall (a,b)$
$dataset \leftarrow [K_0, J_0]$
for $k = 0 : k_{max}$ **do** ▷ Simulation loop
  while $\hat{J} \geq \xi$ **do**
    $(W, \Psi) \leftarrow train(dataset)$
    $\tilde{J} = \Psi\sigma(Wz) \; \forall K_k^{ab} \pm \delta_K$
    $\tilde{K}_k^* = \arg\min_{K_k}(\tilde{J})$
    $u(k) = \left(L(\theta(k)) \otimes \tilde{K}_k^*\right) x(k)$
    $\hat{x}(k) = x(k); \hat{\theta}(k) = \theta(k)$
    for $t = k : k + h + 1$ **do** ▷ Prediction loop
      $p = \hat{\theta}(t)$
      $\hat{u}(t) = \left(L(p) \otimes \tilde{K}_k^*\right) \hat{x}(t)$
      $\hat{x}(t+1) = F_p(\hat{x}(t))$
      $x_0(t+1) = f(x_0(t), 0)$
      $\hat{e}(t) = F_p(\hat{x}(t)) - \mathbf{1_N} \otimes f(x_0(t), 0)$
      $\hat{V}_t = \hat{e}^T(t) P_p \hat{e}(t)$
      $\hat{\theta}(t+1) = randMarkov(\hat{\theta}(t), \Pi)$
    end for
    $\hat{J} = \sum_{t=k+1}^{k+h+1} \hat{V}_t$
    $dataset \leftarrow [\tilde{K}_k^*, \hat{J}]$
  end while
  $K_k = \tilde{K}_k^*$
  $x(k+1) = F_p(x(k))$
  $x_0(k+1) = f(x_0(k), 0)$
end for

In addition, figure 3.1 shows the algorithm as a flow diagram that provides a more general perception on its functionality.

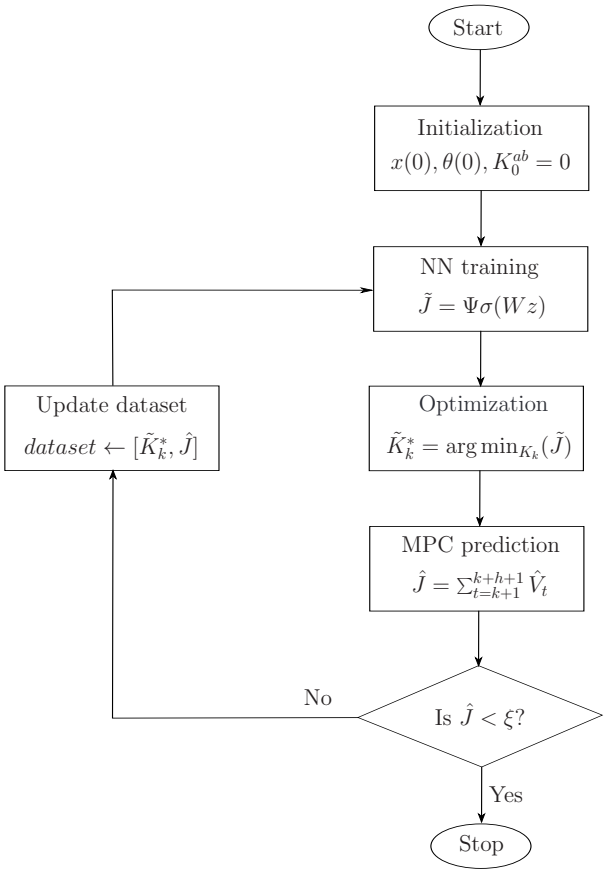
3.5.1 Initialization

Firstly, the system is initialized at the initial condition $x(0)$ and initial mode $\theta(0)$ resulting in the cost function J_0. The gain K_0 is started with null value, i.e. $K_0^{ab} = 0 \; \forall a \in \{1, \ldots, m\}, b \in \{1, \ldots, n\}$. Then, K_0 and J_0 initialize a dataset that will be used to train the NN. A further discussion on the parameters choice is presented in chapter 4.

3.5.2 Online learning

At each iteration, the NN is trained and provides a prediction, $\tilde{J}$, of the cost function for every different combination of the previous K_k with a slight variation of $\pm\delta_K$ in each entry K_k^{ab}. The one with the smallest cost is chosen as the optimized approximation gain $\tilde{K}_k^*$, which will be used in the control protocol, $u(k)$.

Figure 3.1 – Flow diagram of the NN-based MPC algorithm



This training process is repeated until the cost reaches a minimum condition ξ, when the optimization objective is considered achieved and the training can be finalized.

3.5.3 Prediction with MPC

To compute a proper approximation of $\hat{V}_t$ for the h future instants, a prediction loop is used. Firstly, the algorithm generates a random Markovian mode $\hat{\theta}(t)$ based on the previous instants and the matrix Π. Then, a virtual control signal, $\hat{u}(t)$ is computed as well as the systems predicted states, $\hat{x}(t)$. Furthermore, an approximation $\hat{V}_t$ is made that can be used to estimate the cost. At the end of each iteration, an approximation, $\hat{J}$ of the cost function is computed by summing up all $\hat{V}_t$ within the horizon. Finally, $\hat{J}$ is stored in the dataset together with K_k^*.

The NN-based MPC algorithm for nonlinear MAS under Markovian switching topologies, presented in this chapter, as well as its results (to be shown in the next chapter), was published in the journal Engineering Applications of Artificial Intelligence as Floriano *et al.* (2022). The journal was ranked as A1 by the CAPES' Qualis classification system in the quadrennium 2017-2020.

4 Performance Evaluation of Simplified Models

In order to test the proposed NN-based MPC for consensus control in switching topologies, different scenarios were proposed. Distinct system dynamics were chosen from the literature for a comparison with the current state-of-the-art models of MAS controls. In section 4.1, a quadrotor fleet based on the work of Wang *et al.* (2020) is tested. Later on, in section 4.2 the proposed control protocol is investigated in a system with nonlinear components and external disturbances based on the work of Gao *et al.* (2020). Section 4.3 investigates the proposed algorithm in a nonlinear system, based on the work of Zhong *et al.* (2015). Finally, section 4.4 investigates the proposed algorithm in a nonlinear system that represents a fleet of robot cars. For each system, the number of neurons in the hidden layer was chosen based on the cross-validation procedure.

Some of the systems are presented in continuous-time form, so in order to convert for the model stated in chapter 2, a zero-order-hold discretization process took place. The sampling period T was chosen based on the time constant of each system. Appealing to the results of the original works Wang *et al.* (2020), Gao *et al.* (2020), it is possible to identify the order of magnitude of the systems time constant performing around 4 to 6 s. It is then feasible to adopt a sampling period one or two orders of magnitude smaller, depending on the available bandwidth. In this work, $T = 0.01$ s.

In this work, the search method of looking for a local minimum point is based on gradient descent since it has a good trade-off between performance and simplicity in general optimization problems (Chong; Zak, 2004). In addition, this approach was adequate for the applications based on the neural network treated here. The parameter K_0 can be set as zero without jeopardizing the system's response (see Soltero, Schwager, and Rus (2014)). However, the established techniques of optimization literature for choosing the step size δ_K in the gradient descent algorithm do not work in neural-network-based control problems, and only ad hoc solutions can be pursued (see Qiu *et al.* (2021), Huynh, Wu, and Kuo (2019)). In the example, by searching for a value that achieves convergence relatively fast, one is led to $\delta_K = 0.25$. The learning stop condition parameter was chosen as a value that allows for a break in learning while still ensuring a satisfactory approximation of the MPC cost (that does not compromise consensus). The value reached was $\xi = 0.01$.

The horizon parameter was established as $h = 100$, but will be further analysed in more detail in section 4.5. Throughout the results section, several simulations are performed with different values for the transition matrix Π. It is set as one of the distinct matrices Π_s

($s \in \{1,2,3,4\}$), given as follows:

$$\Pi_1 = \begin{bmatrix} 0.95 & 0.05 \\ 0.02 & 0.98 \end{bmatrix}, \quad \Pi_2 = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}, \quad \Pi_3 = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \quad \Pi_4 = \begin{bmatrix} 0.05 & 0.95 \\ 0.98 & 0.02 \end{bmatrix}. \tag{4.1}$$

To weight the communication topology effects, the matrix P_p, as shown in (3.4), was chosen for all simulations as follows:

$$P_p = \frac{1}{2}\left(L(p) + L'(p)\right). \tag{4.2}$$

The results data can also be found in the online repository[1].

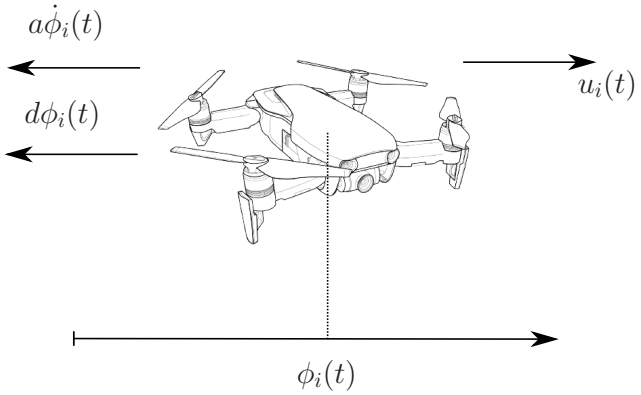
4.1 Quadrotor fleet

The first system to be tested represents a group of three networked quadrotors based on the work of Wang *et al.* (2020). Considering $\varphi_i(t)$ as the position of the i-th quadrotor with respect to one of its coordinate axis (Savino *et al.*, 2015), then its continuous-time dynamics has the form:

$$\ddot{\varphi}_i(t) + a\dot{\varphi}_i(t) + d\varphi_i(t) = u_i(t), \quad i = 1,2,3, \tag{4.3}$$

where a and d are the damping and spring constants and $\dot{\varphi}_i(t)$ and $\ddot{\varphi}_i(t)$ are the i-th agent speed and acceleration, respectively. Figure 4.1 shows a diagram of the dynamics which the quadrotor is subjected to.

Figure 4.1 – Representation of the i-th quadrotor dynamics



Equation (4.3) can be rewritten as

$$\dot{x}_i(t) = A_c x_i(t) + B_c u_i(t), \quad i = 1,2,3, \tag{4.4}$$

with

$$A_c = \begin{bmatrix} 0 & 1 \\ -d & -a \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad x_i(t) = \begin{bmatrix} \varphi_i(t) \\ \dot{\varphi}_i(t) \end{bmatrix}. \tag{4.5}$$

[1] Available at https://github.com/brunofloriano/NN_MPC

In the linear case, the zero-order-hold discretization process provides the function

$$f(x_i(k),u_i(k)) = Ax_i(k) + Bu_i(k), \tag{4.6}$$

in which, $A = e^{A_c T}$ and $B = \int_0^T \mathrm{e}^{A_c \tau} B \, \mathrm{d}\tau$. The damping and spring parameters are set to $a = 0.3$ and $d = 0.8$ and the initial conditions are set to $x_0(0) = x_1(0) = \begin{bmatrix} 0 & 3 \end{bmatrix}^T$, $x_2(0) = \begin{bmatrix} 4 & 2 \end{bmatrix}^T$ and $x_3(0) = \begin{bmatrix} 7 & -1 \end{bmatrix}^T$, and the Laplacian matrices to:

$$L(1) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L(2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}. \tag{4.7}$$

4.1.1 Cross-validation

This section aims to determine the value of M, i.e. the number of neurons at the hidden layer of the NN. In order to do that, a cross-validation is performed. This procedure consists in sweeping M from $M = 1$ until $M = M_{max}$ and verifying the final error for each simulation. The optimum value, called elbow point, is the one in which the error stops decreasing significantly, in such way that the further increase in the number of neurons would not result in a relevant error reduction and would raise the simulation complexity unnecessarily (Alpaydin, 2020).

For the quadrotor system, a value of $M_{max} = 50$ was chosen and let us consider the transition matrix set as $\Pi = \Pi_1$. The error parameter used for cross-validation was the mean-squared error (MSE) of the state, i.e.

$$MSE = \frac{1}{k_f} \sum_{k=0}^{k_f} ||e(k)||^2, \tag{4.8}$$

where k_f is the last instant of the simulation.

The results for the cross-validation procedure can be seen in figure 4.2. It is possible to observe that values between 1 and 10 perform well in comparison to larger values such as $M = 30$. Therefore, it is required to verify the details of this range which can be seen in the zoomed figure. This picture provides the information that the value of $M = 2$ is well fitted for this particular objective. Further increase of M is not necessary, characterizing the elbow point. Therefore, $M = 2$ was chosen.

4.1.2 Consensus result for the quadrotor fleet

Considering the optimized value of neurons, figure 4.3 shows the quadrotor fleet system result for one realization of the chain $\{\theta(k)\}$ with the NN-based MPC compared with Wang *et al.* (2020), in which $x_i^r(k)$ is the r-th entry of the individual state vector $x_i(k)$.

Figure 4.2 – Cross-validation of the linear system

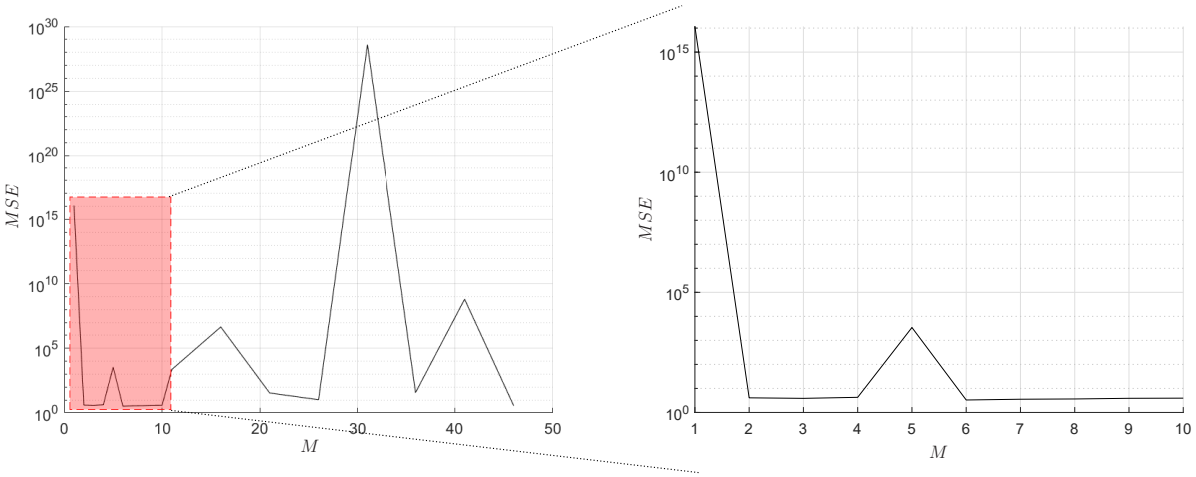
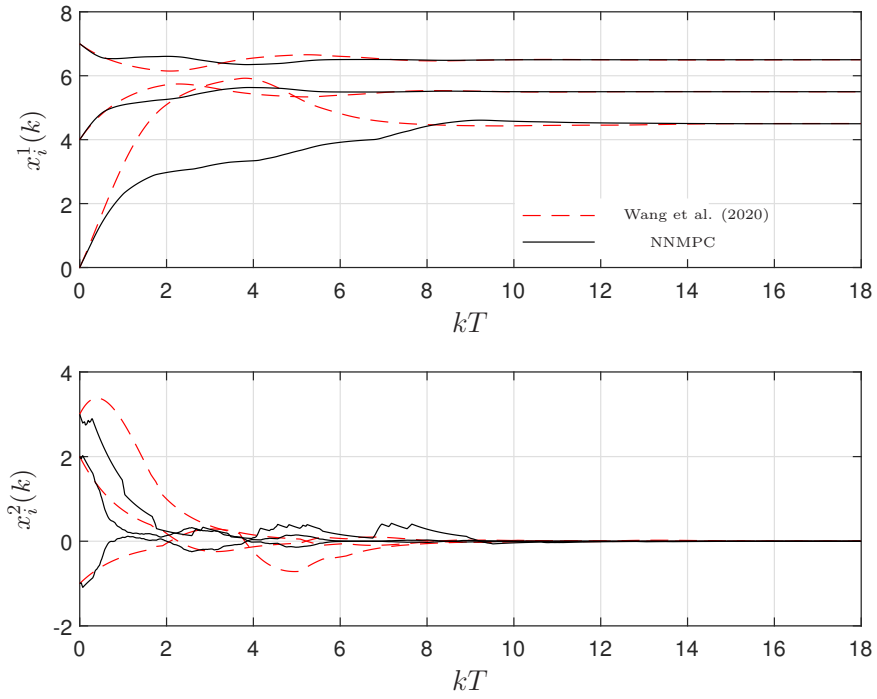


Figure 4.3 – States' progression in linear system for $\Pi = \Pi_1$ with NN-based MPC and the method from Wang *et al.* (2020)



Graphically, it is possible to see that the MAS reaches a consensus relatively fast, at around 8 seconds.

To get a more detailed view of the improvements of the proposed method when compared with the original work of Wang *et al.* (2020), table 4.1 specifies the settling time (defined with the ±2% criteria) and mean-squared error of the r-th element of the i-th state (MSE_i^r) from both control protocols. The later is given by

$$MSE_i^r = \frac{1}{k_f} \sum_{k=0}^{k_f} |e_i^r(k)|^2, \tag{4.9}$$

where e_i^r is the r-th entry of the error vector $e_i(k)$.

Table 4.1 – Performance comparison of quadrotor fleet system for i-th agent and r-th state

Parameter	(i,r)	Wang *et al.* (2020)	NN MPC	Decrease (%)
Settling Time (s)	(1,1)	6.82	4.59	32.6
	(1,2)	7.80	5.16	33.8
	(2,1)	8.66	2.70	68.9
	(2,2)	7.72	3.18	58.8
	(3,1)	9.45	4.41	53.4
	(3,2)	9.82	3.97	59.6
MSE_i^r	(1,1)	0.7831	1.0223	-30.5
	(1,2)	0.8542	0.3409	60.1
	(2,1)	0.0463	0.0424	8.3
	(2,2)	0.1141	0.1284	-12.5
	(3,1)	0.0147	0.0124	15.6
	(3,2)	0.0340	0.0467	-37.4

It is possible to observe that indeed the NN-based MPC is faster, providing a response which can be up to 68.9% quicker than Wang *et al.* (2020), depending on the state. However, the same can not be said about MSE_i^r, once there are some states that perceive an increase (instead of a reduction) in the error.

4.1.3 Quadrotor fleet results with different switching matrices

In order to verify the protocol's effectiveness in different arrangements, the system also can be tested with all four values for the switching matrix $\Pi = \Pi_s$. To measure the system's performance in each case, the parameters settling time and MSE_i^r were used.

The results can be seen in table 4.2. It is possible to observe that in all four cases, consensus is achieved, even though the four scenarios are built with very distinct topology transitions. This difference is indeed observed in the results of settling time and MSE as they present some fluctuation depending on the transition matrix Π_s . Naturally, as the switching probabilities change, so as the control protocol (equation (2.4)) which depends on the communication at each time. However, it is particularly meaningful that for all of those dynamics, consensus is always achieved.

4.2 System with disturbances

Real platforms are subjected to many noises and disturbances. As much as possible, it is necessary to show the practical significance of the proposed protocol. The first experiment, for example, was performed as the simulation of a group of quadrotors. Therefore, it would be plausible to perform a test with believable difficulties such as systems with nonlinearities and disturbances. It can be accomplished by performing simulations with the system based

Table 4.2 – Performance comparison of quadrotor fleet system with different switching for i-th agent and r-th state

Parameter	(i,r)	Π_1	Π_2	Π_3	Π_4
Settling Time (s)	(1,1)	4.59	16.34	4.96	9.97
	(1,2)	5.16	17.11	5.78	11.74
	(2,1)	2.70	7.88	4.05	8.74
	(2,2)	3.18	8.21	4.20	9.87
	(3,1)	4.41	8.83	4.16	9.64
	(3,2)	3.97	9.40	4.32	10.66
MSE_i^r	(1,1)	1.0223	3.4833	1.1100	1.5400
	(1,2)	0.3409	0.4762	0.3324	0.9136
	(2,1)	0.0424	0.0917	0.0401	0.0458
	(2,2)	0.1284	0.0800	0.1048	0.0932
	(3,1)	0.0124	0.0364	0.0025	0.0046
	(3,2)	0.0467	0.0245	0.0305	0.0226

on Gao *et al.* (2020), not only to compare the proposed method with the original work but also to perform a robustness analysis that could verify the practical feasibility of the protocol.

Gao *et al.* (2020) considered the continuous-time dynamics of the form:

$$\dot{x}_i(t) = A_c x_i(t) + B_c u_i(t) + f_c(t,x_i(t)) + B_\omega \omega_i(t),\ i = 1,2,3,4. \tag{4.10}$$

in which, $\omega_i(t)$ is the external disturbance for the i-th agent. Let $f_c(t,x_i(t)) = 0.01\sin(x_i(t))$,

$$A_c = \begin{bmatrix} 0 & 1 \\ 0 & -0.5 \end{bmatrix}, B_c = \begin{bmatrix} 0.8 \\ 1.2 \end{bmatrix}, B_\omega = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{4.11}$$

$$\omega_i(t) = \begin{bmatrix} 0.02\sin(t) \\ 0.02\cos(t) \end{bmatrix}, \tag{4.12}$$

with initial conditions of $x_0(0) = x_1(0) = \begin{bmatrix} 3 & -1 \end{bmatrix}^T$, $x_2(0) = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$, $x_3(0) = \begin{bmatrix} 1 & -3 \end{bmatrix}^T$ and $x_4(0) = \begin{bmatrix} 3 & -3 \end{bmatrix}^T$ and Laplacian matrices as

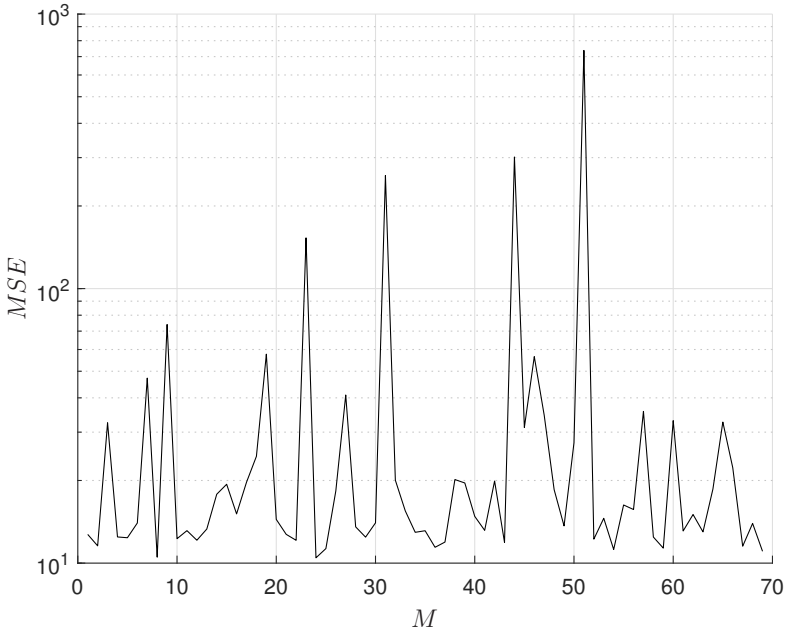
$$L(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad L(2) = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{4.13}$$

The discretization was performed with forward Euler's method with the same time period $T = 0.01s$ (Biswas *et al.*, 2013).

4.2.1 Cross-validation

The cross-validation was performed in the disturbed system with $M_{max} = 69$ and $\Pi = \Pi_1$. The error metric was set as the mean squared error (MSE). The result can be seen in figure 4.4. It is possible to observe that the optimum point occurs around $M = 10$, after which any increase in the number of neurons does not result in a significant reduction on the MSE, instead the are points which show an increase in the error. Therefore, the number of neurons to be used is $M = 10$.

Figure 4.4 – Cross-validation of the system with disturbances



4.2.2 Consensus result for system with disturbances

Figure 4.5 shows the results of the disturbed system with NN-based MPC control protocol and the optimum value of $M = 10$. It is possible to see that the system's response was still capable of reaching consensus in a relatively fast way. Another noteworthy observation pertains to the slight susceptibility of the NNMPC system to noise in the initial time steps, as compared to the findings in Gao *et al.* (2020). This susceptibility is attributed to the early stages of the algorithm, where the system is in the initial learning phase. During this stage, the try-and-error nature of the learning curve results in increased variations in the states. It is important to emphasize, however, that this learning curve is characterized by its rapid pace, with the system swiftly converging to consensus even faster than the approach presented in Gao *et al.* (2020).

The detailed parameters of the MAS response and its comparison with the results of Gao *et al.* (2020) can be examined in table 4.3. Once again the majority of the states displayed a faster response with the NN-based MPC, with a reduction of up to 87.9%. A few states showed a smaller decrease (e.g. 1.4%). However, the mean-squared error, just as in the linear

Figure 4.5 – States' progression in system with disturbances for $\Pi = \Pi_1$ with NN-based MPC and the method from Gao *et al.* (2020)

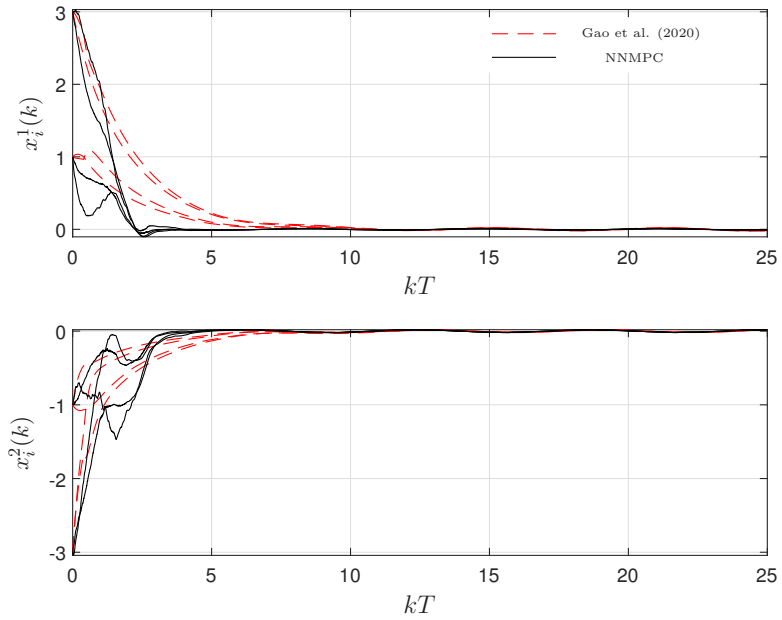


Table 4.3 – Performance comparison of the system with disturbances for i-th agent and r-th state

Parameter	(i,r)	Gao *et al.* (2020)	NN MPC	Decrease (%)
Settling Time (s)	(1,1)	8.91	2.21	75.3
	(1,2)	23.20	21.86	5.8
	(2,1)	22.90	2.76	87.9
	(2,2)	23.28	22.95	1.4
	(3,1)	22.63	3.16	86.0
	(3,2)	5.42	3.15	41.9
	(4,1)	9.39	2.72	71.0
	(4,2)	6.44	3.62	43.8
MSE_i^r	(1,1)	0.3395	0.3169	6.6
	(1,2)	0.0781	0.1119	-43.4
	(2,1)	0.0588	0.0314	46.6
	(2,2)	0.0194	0.0287	-47.5
	(3,1)	0.0824	0.0136	83.5
	(3,2)	0.1029	0.1602	-55.8
	(4,1)	0.4142	0.2133	48.5
	(4,2)	0.1823	0.2497	-37.0

case, remained variable with some states obtaining a decreased amount, but the others performing with an increased error.

4.2.3 System with disturbances with different switching matrices

Similar to the linear case, the disturbed system is tested for distinct scenarios in order to test the NN-MPC on a broader range of possibilities. Therefore, testing for $\Pi = \Pi_s$, for all $s = 1,\ldots,4$, results in table 4.4. Once again, consensus is achieved in all four scenarios, with some variations on the parameters of settling time and MSE. Such outcome allows to conclude that the NN-MPC approach has significant performance while dealing with distinct scenarios of topology switching.

Table 4.4 – Performance comparison of the system with disturbances with different switching for i-th agent and r-th state

Parameter	(i,r)	Π_1	Π_2	Π_3	Π_4
Settling Time (s)	(1,1)	2.21	7.82	3.41	3.00
	(1,2)	21.86	10.42	4.15	21.90
	(2,1)	2.76	22.70	15.47	3.55
	(2,2)	22.95	22.78	23.10	22.90
	(3,1)	3.16	5.89	21.23	3.75
	(3,2)	3.15	5.18	4.54	3.56
	(4,1)	2.72	8.91	3.89	3.41
	(4,2)	3.62	7.14	4.74	4.02
MSE_i^r	(1,1)	0.3169	1.7580	0.1466	0.2472
	(1,2)	0.1119	0.1797	0.2218	0.1520
	(2,1)	0.0314	0.0512	0.0287	0.0227
	(2,2)	0.0287	0.0292	0.0499	0.0342
	(3,1)	0.0136	0.1009	0.0292	0.0099
	(3,2)	0.1602	0.3491	0.2122	0.2124
	(4,1)	0.2133	0.6871	0.1613	0.1482
	(4,2)	0.2497	0.1714	0.4702	0.3340

4.3 Nonlinear system

While the previous section showed results for a disturbed system, it still had a established pattern consisting of a linear component, a nonlinear function and a disturbance signal. For this reason, the previous system was not as generic as the problem had been stated in chapter 2.

Accordingly, it is relevant to test the proposed method in a more general nonlinear system. Therefore, a system was built motivated by the work of Zhong *et al.* (2015), based on the following function:

$$f(x_i(k),u_i(k)) = \begin{bmatrix} -x_i^1(k) - x_i^2(k) + \sin(x_i^1(k) + u(k)) \\ 1 - \cos(x_i^2(k) + u(k)) \end{bmatrix}, \quad i = 1,2,3. \tag{4.14}$$

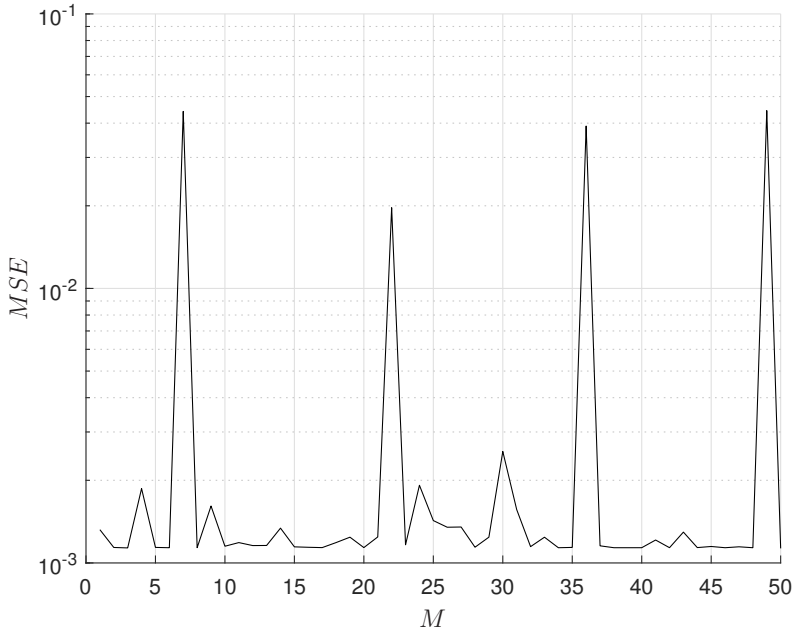
The parameters were set to: initial conditions of $x_0(0) = x_1(0) = \begin{bmatrix} -0.6 & 0.6 \end{bmatrix}^T$, $x_2(0) = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$ and $x_3(0) = \begin{bmatrix} 2.5 & 0.5 \end{bmatrix}^T$, and Laplacian matrices:

$$L(1) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L(2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}. \tag{4.15}$$

4.3.1 Cross-validation

As in the previous sections, the first step consists in a cross-validation procedure to determine the number of neurons M. By sweeping M from 1 to 50, with $\Pi = \Pi_1$ and verifying the MSE, one obtains the graph shown in figure 4.6. Once again, the elbow point occurs at $M = 10$ and therefore this is the number of neurons to be used for the nonlinear system.

Figure 4.6 – Cross-validation of the nonlinear system



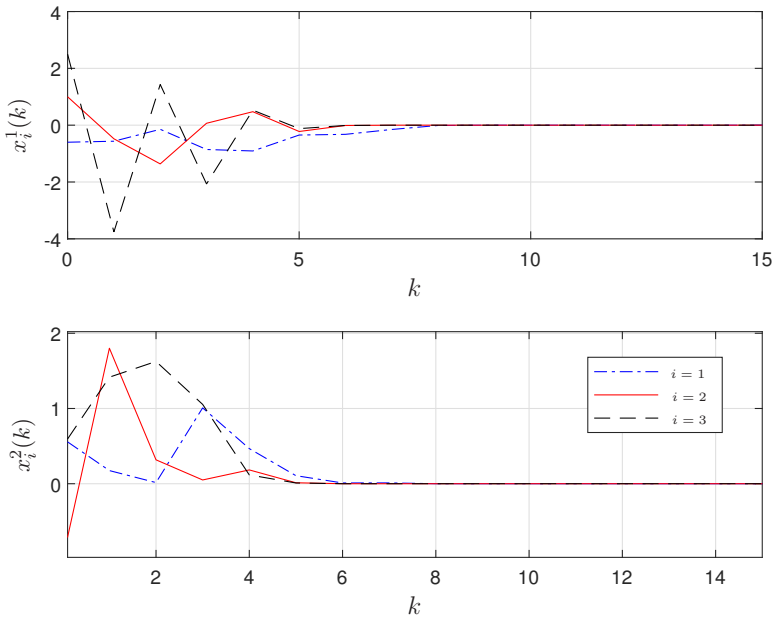
4.3.2 Consensus result for nonlinear system

The states development for the nonlinear system with $M = 10$ and $\Pi = \Pi_1$ can be seen in figure 4.7. One can observe that the system rapidly achieves consensus, in about 8 time steps.

4.3.3 Nonlinear system with different switching matrices

To compare the performance of the proposed method in different scenarios, one can test it with all the values of $\Pi = \Pi_r$. Measuring the settling time and MSE_i^r in all of the four

Figure 4.7 – States' progression in nonlinear system for $\Pi = \Pi_1$ with NN-based MPC



cases results in table 4.5. It is possible to observe that in all cases the consensus is achieved. Some variation takes place, especially for the settling time. However, they remain withing the same order of magnitude, without any large oscillation. This shows that the proposed method is valid to be used in systems with strong nonlinearities and subjected to different communication variations.

Table 4.5 – Performance comparison of nonlinear system with different switching for i-th agent and r-th state

Parameter	(i,r)	Π_1	Π_2	Π_3	Π_4
Settling Time (s)	(1,1)	7.95	10.86	13.67	9.69
	(1,2)	5.9	10.73	10.58	4.96
	(2,1)	5.92	7.95	10.16	7.7
	(2,2)	4.86	6.94	3.29	3.47
	(3,1)	5.41	7.76	9.76	7.45
	(3,2)	4.79	5.9	5.79	6.04
MSE_i^r	(1,1)	0.06	0.1	0.08	0.06
	(1,2)	0.04	0.12	0.08	0.04
	(2,1)	0.07	0.18	0.07	0.07
	(2,2)	0.1	0.17	0.1	0.1
	(3,1)	0.6	1.02	0.6	0.61
	(3,2)	0.13	0.2	0.14	0.15

4.4 Robot car fleet

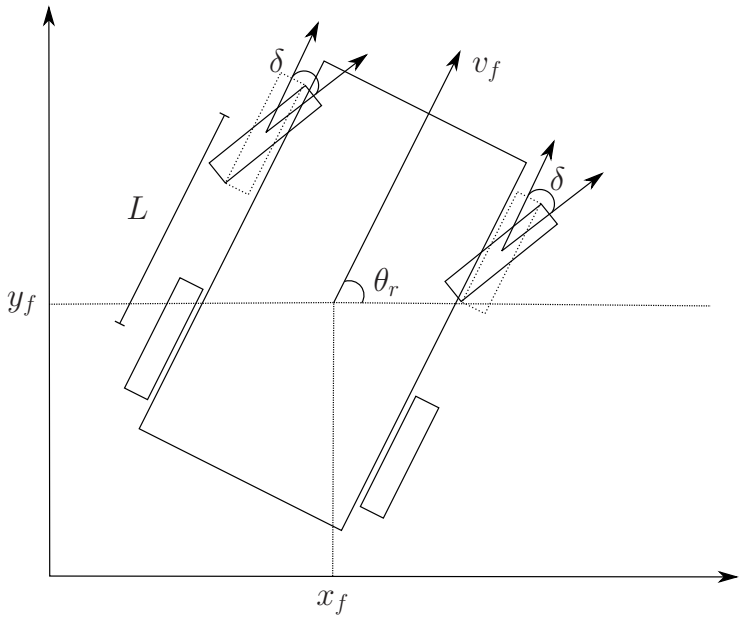
The previous sections showed the proposed approach applied for quadrotor fleets, a disturbed system and a nonlinear system. It is relevant to evaluate the performance of the proposed algorithm in other systems with practical significance, specifically for nonlinear applications, to expand the range of possibilities.

Therefore, a system was built motivated by the application in mobile robot cars, based on the kinematics bicycle model with individual nonlinear dynamics, for the i-th agent, described as follows (see Rajamani (2011), Matute *et al.* (2019)):

$$\dot{x}_i = \begin{bmatrix} \dot{x}_{fi} \\ \dot{y}_{fi} \\ \dot{\theta}_{ri} \\ \dot{\delta}_i \end{bmatrix} = \begin{bmatrix} v_f \cos(\theta_{ri} + \delta_i) \\ v_f \sin(\theta_{ri} + \delta_i) \\ v_f \sin(\delta_i)/L \\ u_i \end{bmatrix}, \tag{4.16}$$

where x_{fi} and y_{fi} are the x- and y-position of the front axle, θ_{ri} is the robot heading, δ_i is the steering angle and $\dot{\delta}_i$ the steering rate, L is the length of the wheelbase and u_i is the control input. The representation of the corresponding robot car model is shown in figure 4.8.

Figure 4.8 – Representation i-th robot car dynamics



Since the current model seeks to simulate a system of multiple cars, it is desired that each agent keeps a fixed distance with respect to the virtual agent. Therefore, considering that to maintain a given formation, the position of the i-th agent has to be distant by the vector x_i^* from $x_0(k)$, than the error, previously stated as $e_i(k) = x_i(k) - x_0(k)$, will be

restated as

$$e_i(k) = x_i(k) - (x_0(k) + x_i^*). \tag{4.17}$$

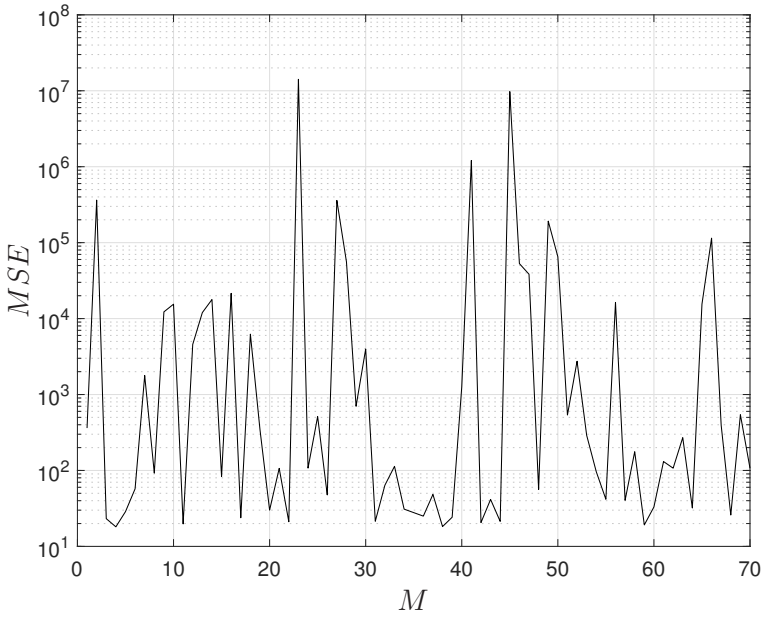
The discretization was performed with forward Euler's method (Biswas *et al.*, 2013). The parameters were set to: initial conditions of $x_0(0) = x_1(0) = \begin{bmatrix} -3 & 3 & \pi/2 & 0 \end{bmatrix}^T$, $x_2(0) = \begin{bmatrix} 5 & -5 & 0 & 0 \end{bmatrix}^T$ and $x_3(0) = \begin{bmatrix} 12.5 & 2.5 & -\pi/2 & 0 \end{bmatrix}^T$, desired formation of $x_1^* = \begin{bmatrix} -17 & 16 & 0 & 0 \end{bmatrix}^T$, $x_2^* = \begin{bmatrix} 9 & -18 & 0 & 0 \end{bmatrix}^T$ and $x_3^* = \begin{bmatrix} 25 & -2 & 0 & 0 \end{bmatrix}^T$ $v_f = 2$ m/s, $L = 0.3$ and Laplacian matrices:

$$L(1) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L(2) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}. \tag{4.18}$$

4.4.1 Cross-validation

As in the previous sections, the first step consists in a cross-validation procedure to determine the number of neurons M. By sweeping M from 1 to 70, with $\Pi = \Pi_1$ and verifying the MSE, one obtains the graph shown in figure 4.9. The elbow point occurs at $M = 4$ and therefore this is the number of neurons to be used for the nonlinear system.

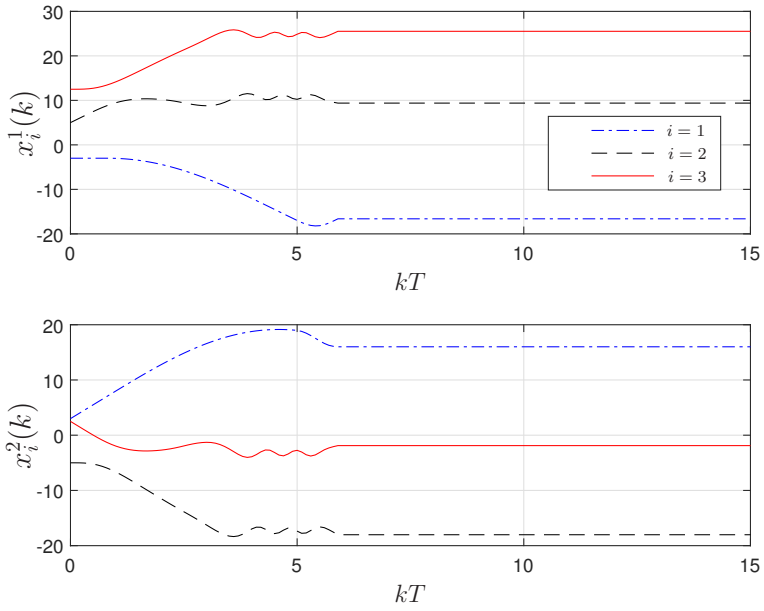
Figure 4.9 – Cross-validation of the car fleet system



4.4.2 Consensus result for car fleet system

The states development for the nonlinear system with $M = 4$ and $\Pi = \Pi_1$ can be seen in figure 4.10. One can observe that the system rapidly achieves consensus, in about 6s. All agents successfully reach their desired point in the formation and remains there for the rest of the time.

Figure 4.10 – States' progression in car fleet system for $\Pi = \Pi_1$ with NN-based MPC



4.4.3 Car fleet system with different switching matrices

We compared the performance of the proposed method in different scenarios by testing it with all the values of $\Pi = \Pi_r$. Measuring the settling time and MSE_i^r in all of the four cases results in table 4.6. It is possible to observe that in all cases the consensus is achieved. Some variation takes place, especially for the settling time. However, they remain withing the same order of magnitude, without any large oscillation. This shows that the proposed method is valid to be used in nonlinear systems with practical significance, as the robot-car fleet.

Table 4.6 – Performance comparison of car fleet system with different switching for i-th agent and r-th state

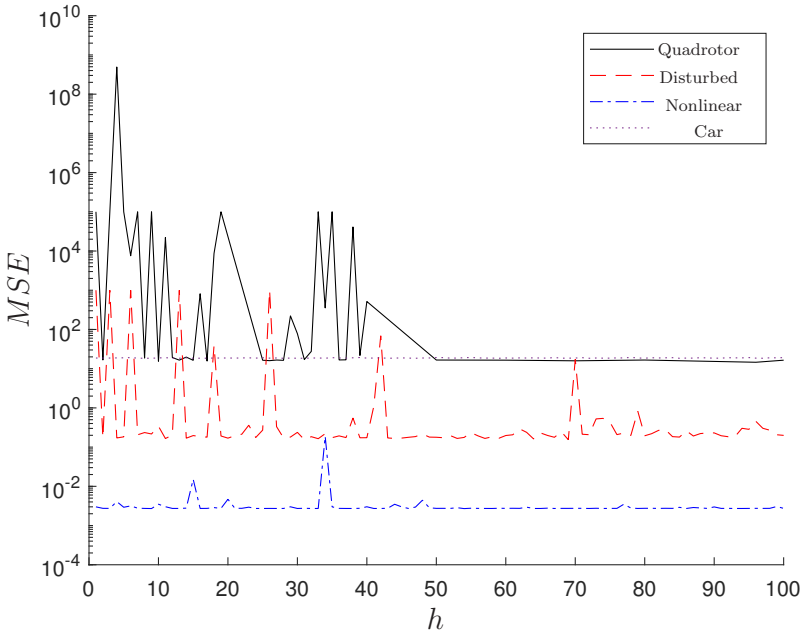
Parameter	(i,r)	Π_1	Π_2	Π_3	Π_4
Settling Time (s)	(1,1)	5.9	7.91	5.93	3.91
	(1,2)	5.87	7.85	5.25	3.85
	(2,1)	0.98	0.98	0.98	0.98
	(2,2)	0.98	0.98	0.98	0.98
	(3,1)	0.98	0.98	0.98	0.98
	(3,2)	0.98	0.98	0.98	0.98
MSE_i^r	(1,1)	1.25	1.32	1.33	1.21
	(1,2)	0.87	0.86	0.85	0.82
	(2,1)	1.8	1.8	1.8	1.8
	(2,2)	2.22	2.22	2.22	2.22
	(3,1)	12.8	12.8	12.8	12.8
	(3,2)	0.56	0.56	0.56	0.56

4.5 Effect of the horizon on the consensus

Another important issue to be considered while deliberating on the method's efficiency is to verify the effect of the horizon h on the consensus. Generally in the MPC literature, it is expected that a smaller value of h might remove the prediction capability of the method while a larger value can neglect the effect of an unexpected disturbance or noise.

To see if the same goes in the proposed method, one can test the system for different values of h. In that context, the three previous scenarios (linear, with disturbances and nonlinear system) were used to test the NN-based MPC with horizons ranging between $h = 1$ and $h = 100$. The MSE is the parameter chosen to observe this effect. The results can be seen in figure 4.11.

Figure 4.11 – Mean squared error with different horizons for each system



It is possible to see that, in general, the error for smaller values of h are significantly higher.For sufficiently high values, as h is increased, the error decreases and around $h = 50$ the MSE settles with some minor variations. Therefore, it can be concluded that the effect of the horizon on the consensus is highly nonlinear but follows the overall expected behaviour in which smaller values result in larger errors.

4.6 Discussion on scalability

As shown in the previous sections, the NN-based MPC approach was tested with $N = 3$ and $N = 4$. However, it is noteworthy to mention that it was designed to be used with any natural number of N agents. Accordingly, some discussion may arise as for the scalability of the proposed method. In other words, it is relevant to question how is the method fit for a larger group of agents.

The scalability problem becomes more intricate as one notes that the NN-based MPC algorithm is mainly running in a centralized manner (see figure 2.1). As such, the increase in the number of agents N particularly affects the length of the state $x(k)$ and all the operations that are performed with it while computing the MPC cost. On the other hand, the neural network learning process is not affected by the number of agents, since it only depends on m, n and M, respectively, the length of the individual control vector, the length of the individual state vector and the number of neurons. Certainly, the neural network's inputs being the entries of K_k (uniform across all agents) and its output representing the cost function (a one-dimensional output) implies that the number of agents does not impact the learning process of the neural network. The consistency in the input structure and the one-dimensional nature of the output ensures that the neural network's learning dynamics remain unaffected by variations in the number of agents. This characteristic contributes to the scalability and generalization capabilities of the neural network across different MAS configurations.

Finally, while the centralization characteristics of the method reduce its scalability, the neural network optimization is not affected by it. Therefore, the method is particularly appealing for an increased number of agents. Further studies on such characteristics, especially in real platforms are significant for a better understanding on its performance.

5 Hurricane Monitoring with Balloon-Based Systems

This chapter is focused on the validation of the NN-based MPC (NNMPC) control strategy within a complex system, particularly in contexts characterized by extreme conditions such as substantial noise, disturbances, and communication constraints. To exemplify these challenging scenarios, we have selected the application of a buoyancy-controlled balloon fleet for hurricane monitoring. This choice holds dual significance—it not only addresses a real-world problem but also encapsulates an environment rife with disturbances, encompassing issues such as power management, restricted communication, noise interference, and conflicting objectives. Consequently, substantial effort has been invested in the application of the NNMPC strategy to develop a novel architecture tailored to hurricane monitoring using these specialized vehicles.

5.1 Introduction

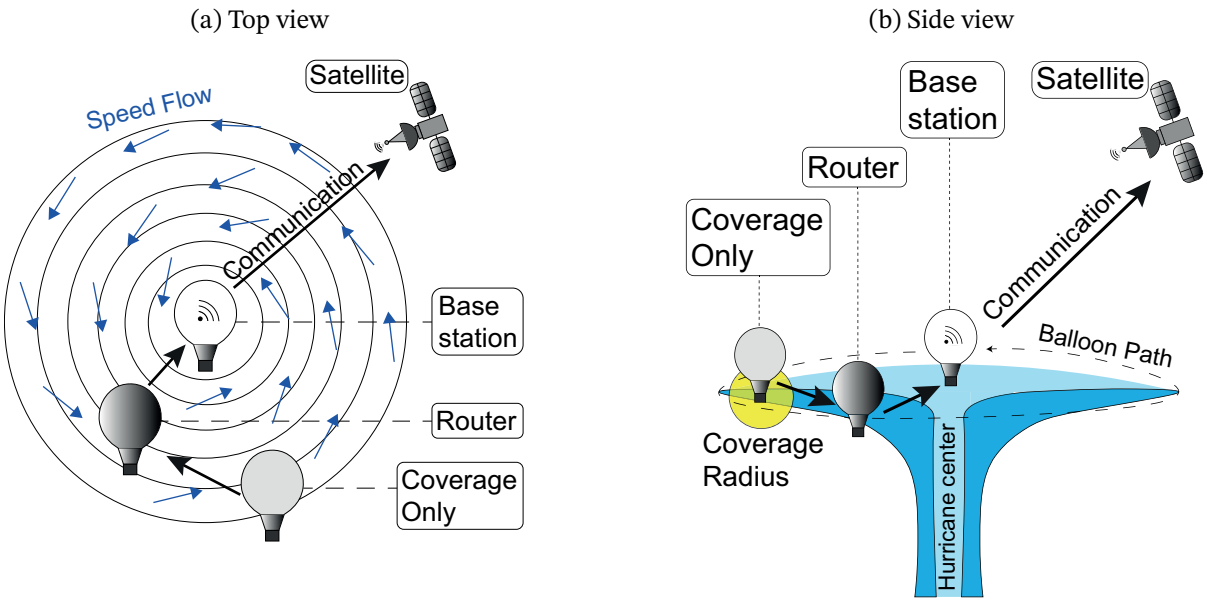
On-site monitoring of extreme natural phenomena, particularly hurricanes, plays a crucial role in acquiring essential data for an accurate forecast (Meneghello; Luchini; Bewley, 2016). The correct anticipation of such weather conditions can conduct preventive responses that might effectively safeguard lives (Bewley; Meneghello, 2016). Efficient on-site monitoring can be accomplished by deploying multiple Unmanned Aerial Vehicles (UAVs) at the site of the occurrence (Cione *et al.*, 2016; Stampa *et al.*, 2021; Mohsan *et al.*, 2023). Utilizing Multi-Agent Systems (MAS) enhances the volume of accessible information, including speed profiles, temperature, pressure, and other sensor data, due to their expanded coverage capacity (Floriano *et al.*, 2021). For instance, numerous studies have explored the use of UAV swarms for monitoring wildfires or floods in recent years (Lin; Liu, 2018; Hu *et al.*, 2022; Tzoumas *et al.*, 2023; Viseras; Meissner; Marchal, 2021; Afghah *et al.*, 2019; Seraj; Silva; Gombolay, 2022; Baldazo; Parras; Zazo, 2019).

When it comes to monitoring hurricane-affected areas, recent studies have presented UAV solutions utilizing quadrotors, commonly known as drones. For instance, the impacts of the 2017 Harvey and Irma hurricanes in the United States were accompanied by numerous studies investigating their effects on roads, buildings, and cities (Greenwood; Nelson; Greenough, 2020; Yeom *et al.*, 2019; Rojas; Khan; Shahtakhtinskiy, 2022; Congress *et al.*, 2019). Similar investigations were conducted for other recent incidents, such as the Willa hurricane (Vizcaya-Martínez *et al.*, 2022) and the Maria hurricane (Schaefer *et al.*, 2020), employing these types of vehicles. However, as outlined in the survey conducted by Mohsan *et al.* (2023), operating quadrotors in adverse weather conditions like hurricanes remains a

significant challenge due to operational complexities and the difficulty in obtaining precise data. Furthermore, both quadrotors and fixed-wing aircraft often face limitations in terms of flight duration due to their energy requirements (Meneghello; Luchini; Bewley, 2016).

Our team, with a rich background in developing innovative monitoring systems for hurricanes, has proposed a swarm of balloons as an *in situ* long-lasting solution (Bewley; Meneghello, 2016). This coordinated system consists of multiple, widely distributed, buoyancy-driven balloons equipped with sensors, enabling prolonged monitoring for several days. The concept behind this approach is to employ low-cost balloons, as highlighted in our previous work (Meneghello *et al.*, 2017). Furthermore, our team has also investigated various control schemes for buoyancy-driven balloons, with particular emphasis on the three-level control (TLC) method (Meneghello; Luchini; Bewley, 2018). However, it is worth noting that the previous TLC approach was limited to a single balloon. Hence, this current work represents an extension of TLC to address the challenges associated with MAS.

Figure 5.1 – Description of the balloon swarm in a hurricane



Building upon our previous contribution, figure 5.1 illustrates the innovative representation of the proposed heterogeneous balloon swarm within a hurricane flowfield. Positioned at its core is the base station, a single agent embedded with privileged hardware that connects directly to a satellite. Surrounding the hurricane's periphery are agents specialized in coverage, solely focused on collecting sensor data within their designated areas. Bridging these agents and enabling efficient data transmission are the routers, balloons endowed with both coverage and routing capabilities. These routers facilitate the transfer of information from other balloons to the base station, leveraging a LoRa (long-range) network (Ghazali; Teoh; Rahiman, 2021), a technology that has already been investigated for UAV-based disaster responses (Saraereh *et al.*, 2020). As the system evolves, the balloons' roles

as router or coverage units can change over time based on their positions. This structure allows the use of a relatively reduced number of multiple cost-effective vehicles that can share valuable information with the satellite while maximizing area coverage.

Still, the difference between the hurricane's area (around 200 km radius) (Holland; Belanger; Fritz, 2010) and the LoRa range (up to 10 km) (Sanchez-Iborra; Cano, 2016) are significant and might imperil the balloons' constant connectivity. Several works have studied multi-agent monitoring systems with intermittent communication (Hu; Liu; Feng, 2018; Shi *et al.*, 2023; Wen *et al.*, 2014; Xiao; Dong, 2021; Zhang *et al.*, 2021). However, to the best of the author's knowledge, none have established a control architecture to deal with the trade-off between coverage and communication. To circumvent this problem, our main objective in this work is to establish a reasonable concession between area coverage (i.e. maximize the area in which the balloons will cover to collect data from the hurricane) and strategically repositioning the balloons to establish the communication required for all data to reach the base station. This is achieved by creating a weighted set of cost function components that are optimized in real time by an intelligent updating process in a periodic setting, where both components constantly relax to their steady state.

In recent years, research in the field of artificial intelligence (AI), has yielded significant contributions to coverage, monitoring, and communication challenges in the context of multiple UAVs (Manoharan; Sujit, 2022; Eshaghi; Nejat; Benhabib, 2023; Day; Salmon, 2021; Puente-Castro *et al.*, 2022). These works have explored various aspects of UAV operations, including cooperative target defense and coverage (Manoharan; Sujit, 2022), concurrent planning with different objectives (Eshaghi; Nejat; Benhabib, 2023), stochastic target search (Day; Salmon, 2021), and path planning (Puente-Castro *et al.*, 2022). Additionally, swarm intelligence algorithms have been recognized for their role in facilitating collaborations among multiple UAVs (Tang; Duan; Lao, 2023). The use of AI-based approaches, such as multi-objective evolutionary optimization (Ramirez-Atencia; Camacho, 2019) has shown promise in addressing conflicting objectives and enhancing mission planning. Furthermore, AI algorithms have been applied to MAS problems, effectively managing UAV trajectories, optimizing load distribution, and enabling targeted communication (Queralta *et al.*, 2020; Wang *et al.*, 2020; Das *et al.*, 2019). However, none of these solutions have simultaneously solved intermittent communication while maximizing area coverage.

In terms of AI approaches, Neural Network Model Predictive Control (NNMPC) has been introduced as an online multi-agent single-objective solution (Floriano *et al.*, 2022). In our work, NNMPC will be adapted to solve the conflicting objectives, providing a real-time approach that can adaptively define the best control output. This is particularly convenient since both problems are constantly being updated. The cost function components for both communication and area coverage are modeled to relax to a steady state value. The area interest function is time-marched via the Fokker-Planck equation, an evolution that represents the recency of an area's information, while the communication component is

modeled with a linear decay.

To summarize, this chapter contributes in the following:

- Expands the buoyancy-driven balloon control in a stratified flowfield of Meneghello, Luchini, and Bewley (2016), Meneghello, Luchini, and Bewley (2018) to multiple agents which increases area coverage and data collection;
- Builds an effective trade-off between area coverage and repositioning for connection by optimizing a weighted cost function via the neural-network-based approach of Floriano *et al.* (2022);
- Establishes a constant restoring interest function, driven by the Fokker–Planck equation, to characterize the requirement of updating a previously covered region after some time

This chapter is organized as follows: Section 5.2 states the system dynamics and the problem's objective; Section 5.3 describes the control architecture based on the NNMPC optimization; Section 5.4 provides the simulation results and corresponding analyzes; and Section 5.5 presents the conclusion remarks.

5.2 Problem Statement

Let us consider a multi-agent system (MAS) composed of N buoyancy-driven balloons. The position of the i-th balloon ($i \in \{1,\dots,N\}$, where $i = 1$ represents the base station) is described by r_i and z_i denoting radial and vertical positions, respectively. The continuous-time dynamics of each balloon, as described in Meneghello, Luchini, and Bewley (2018), are given by the equations:

$$\dot{r}_i = \alpha z_i + \xi_i, \tag{5.1a}$$

$$\dot{z}_i = u_i(t), \tag{5.1b}$$

where the time-averaged radial velocity's vertical gradient can be represented by the variable α. The white Gaussian noise ξ_i represents the fluctuations in the radial velocity caused by the turbulence of hurricanes and has zero mean and c^2 variance. The function $u_i(t)$ determines the feedback control law which acts directly on the vertical position of the balloon.

The angular position θ_i of each balloon in the hurricane depends on its angular velocity ω_i which is a function of its tangential velocity v_i and its radial position r_i. The kinematic equation governing the angular velocity of each balloon is given by:

$$\frac{d\theta_i}{dt} = \omega_i = \frac{v_i}{r_i}. \tag{5.2}$$

The 2D coordinate, $\boldsymbol{q}_i(t) \in \mathbb{R}^2$, of the i-th balloon at time t is given by

$$\boldsymbol{q}_i(t) = \begin{bmatrix} r_i(t)\cos(\theta_i(t)) \\ r_i(t)\sin(\theta_i(t)) \end{bmatrix}. \tag{5.3}$$

The vector of all balloons' positions is defined as $\boldsymbol{q}(t) = \begin{bmatrix} \boldsymbol{q}_1(t) & \dots & \boldsymbol{q}_N(t) \end{bmatrix}^T$.

The tangential velocity v_i is a function of the balloon's position in the hurricane. It is determined by the hurricane model, and for this work, we use the H10 model described in Holland, Belanger, and Fritz (2010). This model is known for its low sensitivity to observing system errors. The tangential velocity v_i is given by:

$$v_i = V_m \left(\left(\frac{R_V}{r_i} \right)^{\varphi} \exp \left[1 - \left(\frac{R_V}{r_i} \right)^{\varphi} \right] \right)^{\chi}, \tag{5.4}$$

where V_m is the maximum tangential wind speed, R_V is the radius at which V_m is attained, φ is a shape parameter that defines the proportion of pressure near the maximum wind radius, and the exponent variable χ is used to account for both the maximum wind values and the data in the outer circulation.

The objective of the system is to cover a significant portion of the hurricane-affected areas (to be defined later), collect sensor data such as wind speed, pressure, and temperature, and transmit it to a satellite.

Due to the limited hardware capacity of small balloons, it is impractical to equip them with all the necessary gear for satellite communication. Therefore, our communication goal is for all balloons to transmit their data to a central agent, which we will refer to as the base station. The base station is a robust agent that collects information from the balloons and sends it to the satellite.

The communication technology chosen for the system is LoRa (long range) due to its low power consumption and wide range capacity (Sanchez-Iborra; Cano, 2016). Low power consumption is crucial for the balloons to remain cost-effective and operate for multiple hours, enabling them to cover a larger area of the hurricane. The wide range of LoRa is required to accommodate the magnitude of a hurricane, which can span up to 200 km (Holland; Belanger; Fritz, 2010).

Although LoRa was the best choice of communication technology for our system it is still limited to a maximum range of 10 km (Petajajarvi *et al.*, 2015). To address this communication challenge, two potential solutions are available: significantly increase the number of balloons to allow for easy neighbor connectivity or design a control strategy that causes the balloons to periodically move closer to each other to enable data routing until it reaches the base station. The latter approach was chosen since it requires fewer balloons, resulting in lower costs.

Finally, we can state the objective of the control protocol as follows:

- Cover a significant portion of the hurricane-affected areas;
- Periodically reposition the balloons to ensure they are within communication range of each other, thereby enabling reliable data routing.
- Route the information of the balloons until it reaches the base station

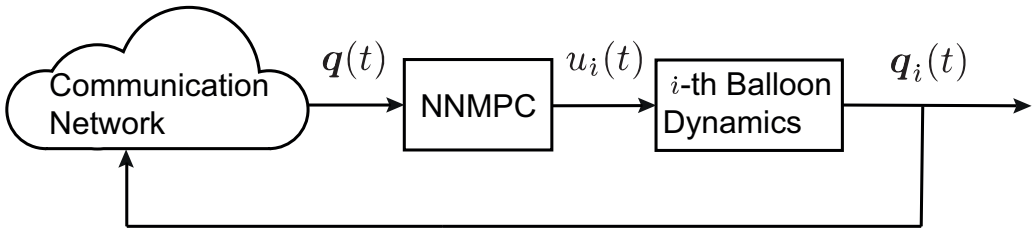
5.3 Control Architecture

Having established the general problem in the previous section, which included formalizing the equations governing the hurricane and balloons, as well as discussing the communication limitations and coverage goals, this section is dedicated to presenting our proposed architecture. The purpose of this architecture is to address the system's objectives previously outlined, namely, to control a cooperative team of balloons capable of simultaneously covering the hurricane, repositioning themselves for effective communication, and ensuring the reliable transmission of data to the base station.

To achieve these objectives, we will first develop cost functions that accurately represent the coverage and communication goals. Subsequently, we will provide the mathematical formulations necessary to optimize both objectives using a neural-network-based model predictive control (NNMPC), as introduced in Floriano *et al.* (2022). By employing this approach, we aim to effectively coordinate the actions of the balloon team, leveraging advanced control strategies to enhance area coverage and enable efficient communication within the network.

The code for the implemented control protocol is available online[1]. A block diagram of the control architecture is shown in figure 5.2.

Figure 5.2 – Block diagram of the control architecture of the i-th balloon



5.3.1 Area Coverage

The first objective of the system is area coverage, i.e. to ensure the balloons cooperatively cover the most area affected by the hurricane to gather as much data as possible.

To do so, an interest function must be defined, i.e. a function of the geographical coordinates that establishes the most relevant area to be covered at time t.

Let $\Gamma(\boldsymbol{q}(t), *, t) : \mathbb{R}^2 \to \mathbb{R}$ be the interest function of the hurricane at time t and with the balloons' positions at $\boldsymbol{q}(t)$. The interest function's initial distribution along the two-dimensional plane, represented by the position vector $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2$, is given by:

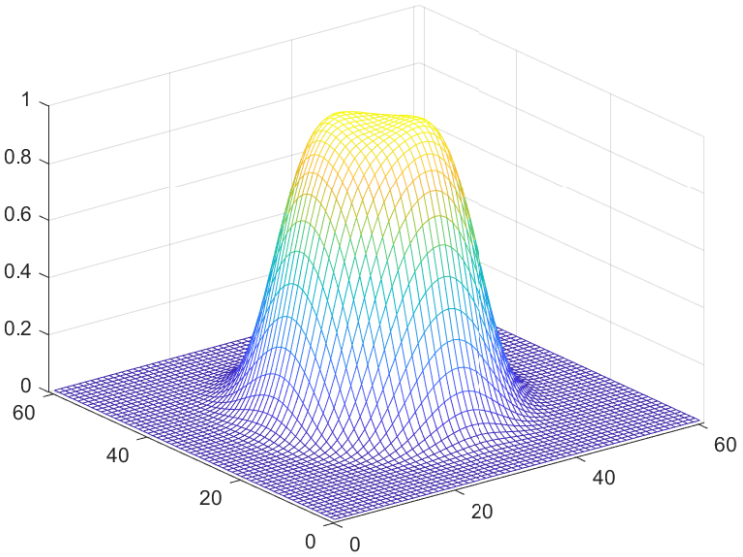
$$\Gamma(\boldsymbol{q}(0), \boldsymbol{x}, 0) = \frac{1}{(2\pi|\boldsymbol{\Sigma}|)^{1/2}} \exp\left[-\frac{1}{2}\left(\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right)^2\right], \tag{5.5}$$

[1] Available at https://github.com/brunofloriano/NN_MPC_application

where $\boldsymbol{\mu}$ is the position of the hurricane's center, therefore $\boldsymbol{\mu} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $\boldsymbol{\Sigma} = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, where σ is the interest radius, set according to the radial distance at which the hurricane provides most relevant data.

The shape of the interest function at time $t = 0$, described in equation (5.5), is visually depicted in figure 5.3. It is noteworthy that the summit of the function exhibits a flatter profile compared to a conventional Gaussian distribution. The selection of a 4th power Gaussian was deliberate, driven by the need to capture this structure, enabling the characterization of a region of interest with a specific radius in accordance to the hurricane's most relevant area.

Figure 5.3 – Interest function shape at $t = 0$



The interest function undergoes two steps of updating. The first step involves updating $\Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)$ based on the detection area of the balloons. The second step is a process through which the interest function is restored when and where it is no longer influenced by any nearby balloon.

5.3.1.1 Coverage update

As each balloon passes through an area, the value of the interest function in the vicinity recedes based on the balloon's coverage influence, $\Phi_i(\boldsymbol{q}_i(t), \boldsymbol{x})$, described by:

$$\Phi_i(\boldsymbol{q}_i(t), \boldsymbol{x}) = \frac{1}{(2\pi|\boldsymbol{H}_i|)^{1/2}} \exp\left[-\frac{1}{2}\left(\left(\boldsymbol{x} - \boldsymbol{q}_i(t)\right)^T \boldsymbol{H}_i^{-1}\left(\boldsymbol{x} - \boldsymbol{q}_i(t)\right)\right)\right], \tag{5.6}$$

where $\boldsymbol{H}_i = \eta_i^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, and η_i is the radius of each observation area, i.e. the area which each balloon can detect and collect data.

Therefore, the interest function can be updated as

$$\frac{\partial \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)}{\partial t} = -\sum_{i=1}^{N} \Phi_i(\boldsymbol{q}_i(t),\boldsymbol{x}). \tag{5.7}$$

In addition, the interest function should always be limited above by 1 and below by 0.

5.3.1.2 Restoring update

The second update step is dedicated to restoring the interest function to its initial value, as described by equation (5.5), when and where the balloons' influence is no longer present. The time-evolution of $\Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)$ is governed by the two-dimensional Fokker-Planck equation:

$$\frac{\partial \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)}{\partial t} = -\sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left[v_i(\boldsymbol{x}) \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t) \right] + \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial^2}{\partial x_i \partial x_j} \left[D_{ij}(\boldsymbol{x}) \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t) \right], \tag{5.8}$$

where $v_i(\boldsymbol{x})$ is the i-th component of the drift vector $\boldsymbol{v}(\boldsymbol{x})$, and $D_{ij}(\boldsymbol{x})$ is the (i,j)-th component of the diffusion tensor $\boldsymbol{D}(\boldsymbol{x})$.

$$v_1(\boldsymbol{x}) = -\frac{D_{11}}{\sigma^4} ||\boldsymbol{x}||^3 \cos\varphi, \tag{5.9a}$$

$$v_2(\boldsymbol{x}) = -\frac{D_{22}}{\sigma^4} ||\boldsymbol{x}||^3 \sin\varphi, \tag{5.9b}$$

where $\varphi = \tan^{-1}(\frac{x_2}{x_1})$. Defining $v_1(\boldsymbol{x})$ and $v_2(\boldsymbol{x})$ as shown in equation (5.9) ensures that as $t \to \infty$, $\Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)$ approaches its steady-state, balloon-free value.

5.3.2 Communication

5.3.2.1 Routing Protocol

The routing protocol is an algorithm to define which communication will be established among the agents of the system. While LoRa allows multiple communications, the power consumption limitations impose the necessity of optimizing the number of connections in the communication graph.

The most suitable protocol for that requirement was the tree-based routing (TRP) designed by Gong and Jiang (2011). The adaptation of that protocol can be seen in algorithm 5.1.

For every j-th balloon which is not the base station (i.e. $j \neq 1$), the first part is to determine if it is within the communication range ρ of the base station. The distance between each pair of balloons is defined as

$$d_{ij}(t) = ||\boldsymbol{q}_i(t) - \boldsymbol{q}_j(t)||. \tag{5.10}$$

If $d_{ij}(t) < \rho$ then both balloons are considered connected.

Algorithm 5.1 Routing protocol

for $j = 2, \ldots, N$ **do**
  if $d_{1j} < \rho$ **then**
    $l_{1j} \Leftarrow -1$
  else
    $C_{min} = d_{1j}$
    $p = 0$
    for $i = 2, \ldots, N, i \neq j$ **do**
      if $d_{ij} < \rho$ **then**
        $C_{ij} \Leftarrow d_{ij}$
        if $(d_{i1} < d_{j1})\&(C_{ij} < C_{min})$ **then**
          $C_{min} \Leftarrow C$
          $p \Leftarrow i$
        end if
      end if
    end for
    if $p \neq 0$ **then**
      $l_{pj} \Leftarrow -1$
    end if
  end if
end for
$l_{ii} \Leftarrow -\sum_{j=1}^{N} l_{ij} \forall i \in \{1, \ldots, N\}$

If the balloon is not able to connect directly to the base station, then it will look for its neighbors (the ones within its connectivity range) and decide which one to send data to. Another requirement is that a balloon can only send data to an agent nearer to the base station than itself; this way the graph establishes a tree directed to the base station at the center. For this work, it was sufficient to establish that data will be sent to the nearest balloon; in other words, the algorithm chooses i that minimizes the cost $C_{ij} = d_{ij}$. Once a connection is established (from j to p), the element l_{pj} of the instant Laplacian matrix $L(\boldsymbol{q}(t))$ should be updated as -1. At the end of the algorithm, the diagonal of $L(\boldsymbol{q}(t))$ should be corrected as the sum of each row.

5.3.2.2 Data Storage for Message Forwarding

When a balloon establishes a connection with a neighboring balloon, it gains the ability to both receive and transmit data. To fulfill one of the objectives outlined in section 5.2, which is to ensure maximum data accessibility for the base station, it is necessary to differentiate the roles of each balloon. To achieve this, we propose the concept of routing balloons, which comprise a distinct group of agents capable of receiving and storing data from other balloons. Subsequently, these routing balloons can efficiently forward the accumulated data to the base station. By implementing this approach, we can guarantee that the information reliably reaches its intended destination without compromising the coverage task.

Since we have the advantage of working with a predictive control (to be explained in

detail in section 5.3.3), the system can plan ahead which balloon will work as the router, i.e. the agent to transmit from one neighbor to another (or to the final destination, which is the base station).

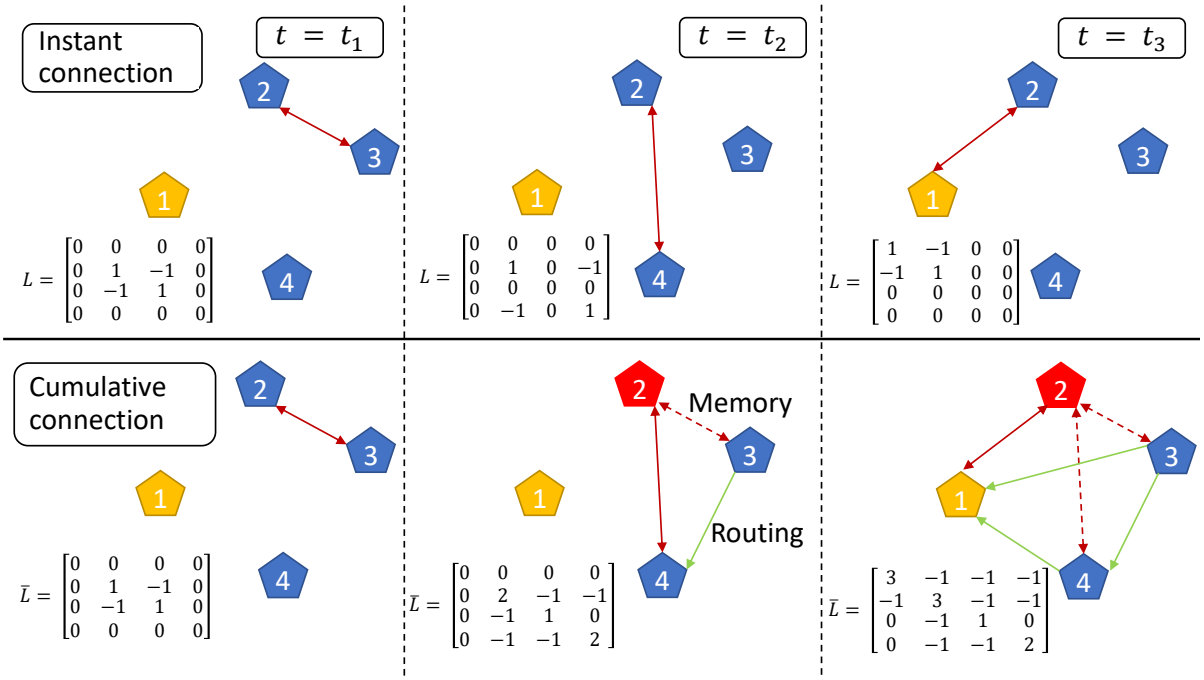
Therefore, a mathematical variable is necessary to characterize the cumulative behavior of the data that are transmitted and stored from one balloon to the other.

Let $\bar{L}(\boldsymbol{q}(t), t) \in \mathbb{R}^N$ be a cumulative Laplacian matrix. It differs from $L(\boldsymbol{q}(t))$, which only represents the graph of instant communication. Instead, $\bar{L}(\boldsymbol{q}(t),t)$ characterizes a cumulative graph, which represents both past and present communication. In that sense, after a disconnection between i and j, its entry $\bar{l}_{ij}$ keeps a non-null value (denoting that there was a connection in the past), while l_{ij} drops to zero immediately.

As an illustrative example, figure 5.4 shows a comparison between instantaneous and cumulative connectivity, characterized by the Laplacian matrices L and $\bar{L}$, respectively. For each time instant ($t_1 < t_2 < t_3$), agent $i = 2$ connects with one of the 3 other members of an $N = 4$ MAS. It is notable that at $t = t_2$, agent $i = 2$ retains a memory of its prior connection with $j = 3$, despite the current absence of a direct connection. This retention is evident in $\bar{L}$ as well since the terms relating the two vehicles are non-zero (in contrast with the instantaneous matrix L). Furthermore, agent $i = 1$ exhibits the capability to function as a router by transmitting previously memorized information to $j = 4$. Finally, at $t = t_3$ a similar process unfolds as agent $i = 2$ connects with the base station, $i = 1$, transmitting data from all other members of the MAS, in accordance with our specified requirements.

As the system is designed to operate for extended periods of time, a key consideration arises regarding the potentially outdated nature of data stored in the routers. By the time

Figure 5.4 – A Comparative Illustration of Instantaneous and Cumulative Connectivity



the stored data reaches the base station, it may no longer be relevant or representative of the current conditions. To address this challenge and ensure periodic updating, we propose the incorporation of a decay factor into the Laplacian matrix. This decay factor indicates the gradual degradation of stored data over time, signifying the diminishing relevance of outdated information. Consequently, we can establish the evolution of $\bar{l}_{ij}$ as a means of capturing the temporal dynamics and mitigating the impact of outdated data on the overall system performance, as follows

$$\bar{l}_{ij}(\boldsymbol{q}(t),t) = \begin{cases} l_{ij}(\boldsymbol{q}(t_c)), & \text{if } t_c \leq t < t_d \\ l_{ij}(\boldsymbol{q}(t_c))\lambda(t-t_d), & \text{if } t \geq t_d \end{cases} \tag{5.11}$$

where $\lambda(t)$ is a decay function, and t_c and t_d are the connection and disconnection time, respectively. Since the communication decay is slow so that the information can be stored for hours, $\lambda(t)$ can be set as a linear decay with period T_{cc}, i.e.

$$\lambda(t) = \begin{cases} 1 - \frac{t}{T_{cc}}, & \text{if } 0 \leq t < T_{cc} \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

Finally, if at the moment of connection between i (recipient) and j (sender), the latest has the information of another agent k ($k \neq i,j$), i.e. $\bar{l}_{jk} \neq 0$, then it can forward its data to i. In other words, $\bar{l}_{ik} = \bar{l}_{jk}$.

5.3.3 Neural-Network-Based MPC (NNMPC)

The control protocol chosen to achieve the area coverage and communication stated in section 5.2 is the neural-network-based model predictive control (NNMPC) designed by Floriano *et al.* (2022). This method has the advantage of working in random processes and with unknown external disturbances due to its online learning feature. In addition, it is designed to work in applications with limited communication capacities.

5.3.3.1 Cost Function

Design of the cost function is of fundamental importance as it will state the systems goals in the form of a function that must be minimized. Since the objectives are anchored to the area coverage and the communication requirement, two components must be created.

The cost energy, $V_{ac}(\boldsymbol{q}(t),t)$, associated with the area coverage requirement can be stated as:

$$V_{ac}(\boldsymbol{q}(t),t) = \beta_{ac}\left(\frac{A_n(\boldsymbol{q}(t),t)}{A_T}\right), \tag{5.13}$$

where $A_n(\boldsymbol{q}(t),t) = \int_{\boldsymbol{x}\in\mathcal{X}} \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)\mathrm{d}\boldsymbol{x}$ is the remaining interest of the area covered by all the agents and $A_T = \int_{\boldsymbol{x}\in\mathcal{X}} \Gamma(\boldsymbol{q}(t),\boldsymbol{x},0)\mathrm{d}\boldsymbol{x}$ is the total interest of the area, where $\mathcal{X} \subset \mathbb{R}^2$ is the studied region. The parameter β_{ac} is the weight of the area coverage component in the cost function.

The communication component, defined by $V_{cc}(\boldsymbol{q}(t),t)$, should reflect the communication requirement, i.e. that the base station has the data of the maximum number of balloons. To mathematically state that requirement, we can use the cumulative communication matrix $\bar{L}(\boldsymbol{q}(t),t)$ which stores all the past interactions between the balloons. The communication component is stated as:

$$V_{cc}(\boldsymbol{q}(t),t) = -\beta_{cc}\left(\bar{l}_{11}(\boldsymbol{q}(t),t) + \gamma \sum_{i\in\mathcal{M}} \bar{l}_{ii}(\boldsymbol{q}(t),t)\right). \tag{5.14}$$

where β_{cc} is the weight of $V_{cc}(\boldsymbol{q}(t),t)$ in the cost function. It is a function of time to address the necessity of having an increasing urgency for the system not being connected. $\mathcal{M} \subset \{1,\ldots,N\}$ is the set of balloons designated as routers and γ is the weight of the routers communication.

The inclusion of the negative signal in equation (5.14) serves to characterize the direction of the communication component $V_{cc}(\boldsymbol{q}(t),t)$ in the optimization process. This choice is motivated by the fact that the parameters intended for optimization ($\bar{l}_{ii},\ \forall i \in \{1,\ldots,N\}$) should be maximized. To ensure consistency in the optimization direction, the negative signal is introduced. Consequently, maximizing $\bar{l}_{ii}$ aligns with the minimization of $V_{cc}(\boldsymbol{q}(t),t)$. This deliberate choice maintains a unified optimization objective while accommodating the dual nature of the parameters involved.

Finally, both requirements can be stacked in a single energy vector whose norm is the main numerical energy that should be minimized, which can be stated as:

$$V(\boldsymbol{q}(t),t) = \left\| \begin{bmatrix} V_{cc}(\boldsymbol{q}(t),t) \\ V_{ac}(\boldsymbol{q}(t),t) \end{bmatrix} \right\|. \tag{5.15}$$

By taking its expected value, $E\left[V(\boldsymbol{q}(t),t)\right]$, and summing it up in discrete time t_k, where $k \in \mathbb{Z}, k \geq 0$, until the horizon step h in the MPC framework results in the final cost function:

$$J(t_0) = \sum_{k=0}^{h} E\left[V(\boldsymbol{q}(t_k),t_k)\right], \tag{5.16}$$

which must be minimized from time to time by the neural network approach, i.e. to find a $u(t_0)$, at time $t = t_0$, that reaches a minimum cost $J(t_0)$, as in

$$u^*(t_0) = arg\min_{u(t_0)} J(t_0). \tag{5.17}$$

5.3.3.2 Three Level Control (TLC)

The control of each balloon is based on the three level control (TLC) as described in Meneghello, Luchini, and Bewley (2016), Meneghello, Luchini, and Bewley (2018). This means that the balloon can control its altitude in three steps, i.e. $Z_i \in \{-\bar{z}, 0, \bar{z}\}$, where $\bar{z}$

is a step change in the altitude. By changing its altitude, the balloon can also control its radial position due to its dynamics, as described in equations (5.1a) and (5.1b). Therefore the control output of the neural network should be within the three steps.

In order to combine the NNMPC framework with TLC, we have to modify the parameter δ_K used in Floriano *et al.* (2022). This parameter is the variation on $K(t)$ to verify which one results in the smallest cost function. Since the control input u_i is limited to be in the set $\{-\bar{z}, 0, \bar{z}\}$. Therefore $\delta_K = \bar{z}$.

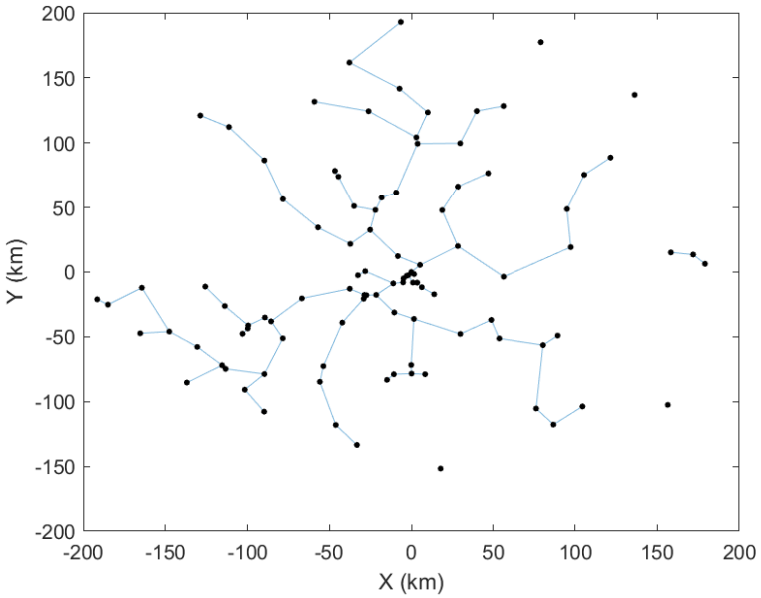
5.4 Results

5.4.1 Simulation-Based Validation of the Routing Protocol

To verify the effectiveness of the tree-based routing protocol we have developed, we have conducted a simulation involving $N = 100$ balloons placed within a hurricane environment, where no external control is applied. In this simulation, the balloons will rely solely on the movement induced by the hurricane's wind speed, as defined in equation (5.4). Since there is no external control mechanism, according to equation (2.5), the radius of each balloon will remain constant throughout the simulation, except for some random small oscillations. To facilitate the evaluation of the routing protocol's performance, we have intentionally exaggerated the communication range by setting it to $\rho = 50$ km.

The simulation was configured to run for a duration of $t = 2$ hours, during which the initial state of the system provides valuable insights into the routing tree established by our protocol. As illustrated in figure 5.5, this figure depicts the communication tree among

Figure 5.5 – Communication tree established by the routing protocol with $N = 100$ agents. Notice how several branches are formed to send data to the base station at the center.



the agents at the initial stage of the simulation. It is noteworthy that several branches have formed, with the primary objective of efficiently transmitting data to the base station located at the center of the hurricane.

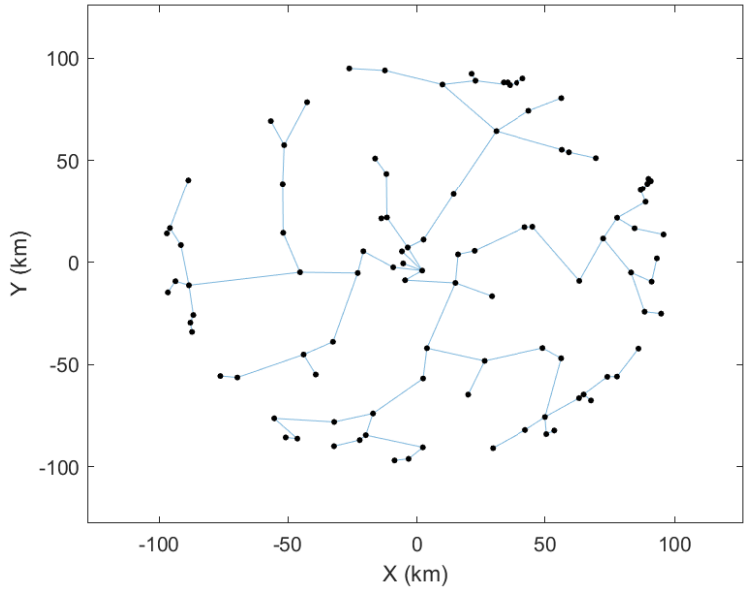
Our simulation results align with the original concept of the routing protocol, where the system exhibits a preference for establishing trees that optimize information flow toward the central base station. This strategic routing approach effectively reduces the communication effort required by the balloons in the network.

5.4.2 Performance Evaluation of Simplified Altitude Control Strategy

Following the successful validation of our routing protocol, our focus shifted towards evaluating the performance of the control mechanism, as defined in equation (2.5), in a system consisting of $N = 100$ balloons. To assess the control's effectiveness, we implemented a straightforward control strategy referred to as the "Simplified Altitude Control Strategy". This control strategy is based on the distance of each balloon from the center of the hurricane. Specifically, if a balloon is situated in a region where its distance from the center exceeds 100 km, the 3-level-control system will activate to lower its altitude, thereby moving it closer to the center. Conversely, if a balloon is within the circle of a 100 km radius, centered on the base station, the control remains at zero.

The simulation was conducted with the same communication range of $\rho = 50$ km and was executed for a total duration of $t = 24$ hours. The final state of the system at the conclusion of the simulation is depicted in figure 5.6. Notably, the figure illustrates how the balloons maintain a uniform distance from the center, staying within a radius of 100

Figure 5.6 – Communication tree with a Simplified Altitude Control Strategy. Notice how the balloons keep a distance to the center up to 100 km.



km. This outcome serves as compelling evidence that the 3-level-control protocol effectively regulates the balloons' positions within the desired range.

With the verification of our control mechanism, we are now well-equipped to proceed with addressing the area coverage and communication trade-off problem.

5.4.3 Area Coverage Validation

5.4.3.1 Binary Interest Function without Restoring

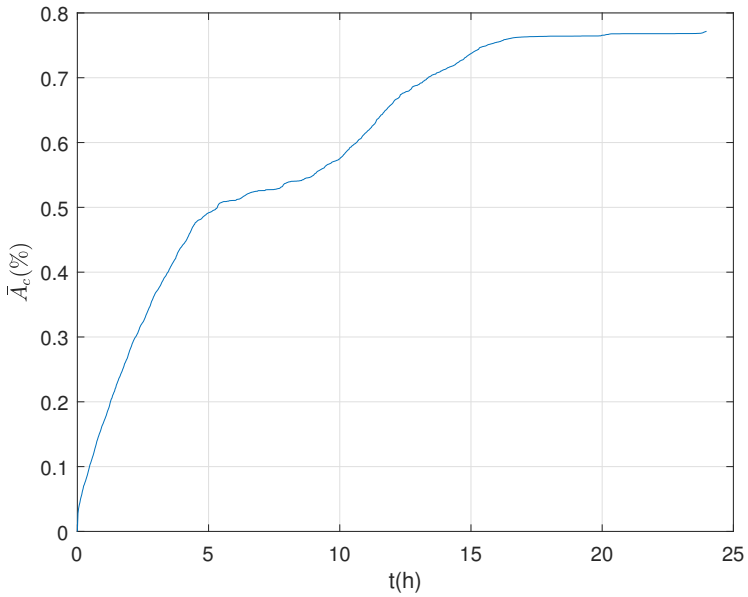
We started our study of area coverage by adopting a progressive approach. Our initial exploration involved employing straightforward models and gradually augmenting the complexity of the interest function to align with the system's evolving requirements outlined in section 5.2.

In this preliminary model, our interest function, denoted as $\Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)$, operated in a binary fashion. Specifically, at any given time t, it assumed a value of either 0 if any balloon had traversed a predefined area near point $\boldsymbol{x}$ in a manner that encompassed the point $\boldsymbol{x}$ within its observation range, or 1 otherwise.

Crucially, this initial model excluded any temporal restoring component; once the interest function was decreased to 0 at certain points, it remained unchanged indefinitely. Consequently, we did not yet address the system's inherent requirement for revisiting and re-monitoring specific areas within the hurricane. In essence, it adhered to a "one-time monitored, forever monitored" principle.

Figure 5.7 presents a time evolution of the area covered percentage. Notably, at the culmination of the simulation, which spans 24 hours of hurricane development, we observe

Figure 5.7 – Time evolution of the area covered percentage



that approximately 77% of the area is considered monitored. This outcome underscores the commendable performance of our control protocol in terms of effectively distributing the balloons around the hurricane's perimeter, minimizing overlap, and optimizing area coverage. Furthermore, the validation of the routing protocol is evident, as all balloons successfully transmit their data to the base station by the simulation's end.

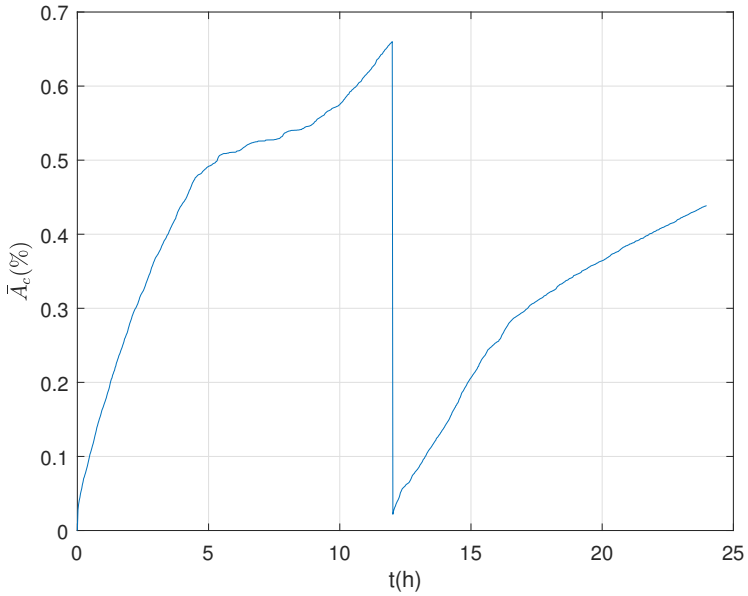
5.4.3.2 Binary Interest Function with Forced Reboot

In line with our progressive approach to modeling the interest function, our next step involves the introduction of a restoring feature. The current objective of this feature is to implement a forced reboot mechanism once a substantial portion of the hurricane has been covered. This allows us to assess the adaptability of the established control protocol in response to changes in the interest function, particularly in scenarios where already covered areas, marked as 0, have transitioned back to 1.

In this particular configuration, our interest function remains binary, lacking a quantitative measure for monitoring other than the binary set. The temporal restoring aspect of the interest function remains somewhat constrained, as the sole trigger for restoration is a forced reduction of the entire domain to 1. Consequently, the system is prompted to revisit and re-monitor the hurricane's regions following a single change that occurs midway through the simulation.

As illustrated in figure 5.8, during the initial half of the simulation, the system behaves in a manner similar to the previous section. The abrupt drop in the interest function occurs after X hours, causing the entire domain to reset to 1. Importantly, the system successfully resumes its monitoring efforts for the areas marked for restoration, effectively minimizing

Figure 5.8 – Time evolution of the area covered percentage with forced reboot



the associated coverage cost. By the simulation's conclusion, the total covered area stands at 44%. It's important to note that this percentage is lower than the previous section, primarily due to the fact that the system had only half the time available for coverage compared to the previous result.

This outcome confirms the capability of our control approach to handle restoring features, even in response to sudden changes in the interest function. While the final covered area percentage is lower, the system demonstrates its adaptability and effectiveness in responding to shifts in monitoring requirements.

5.4.3.3 Continuous Interest Function with Linear Decay

With the addition of a linear decay to the interest function, we have transitioned from a binary representation to a continuous measure, where $\Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)$ now quantifies the level of interest in monitoring a specific region within the hurricane. The value assigned to each region provides a quantitative measure of its priority for observation, with higher values signifying greater necessity.

Moreover, the restoration mechanism has evolved from a sudden change to a continuous decay over time, following a linear function as described by:

$$\frac{\partial \Gamma(\boldsymbol{q}(t),\boldsymbol{x},t)}{\partial t} = \zeta \tag{5.18}$$

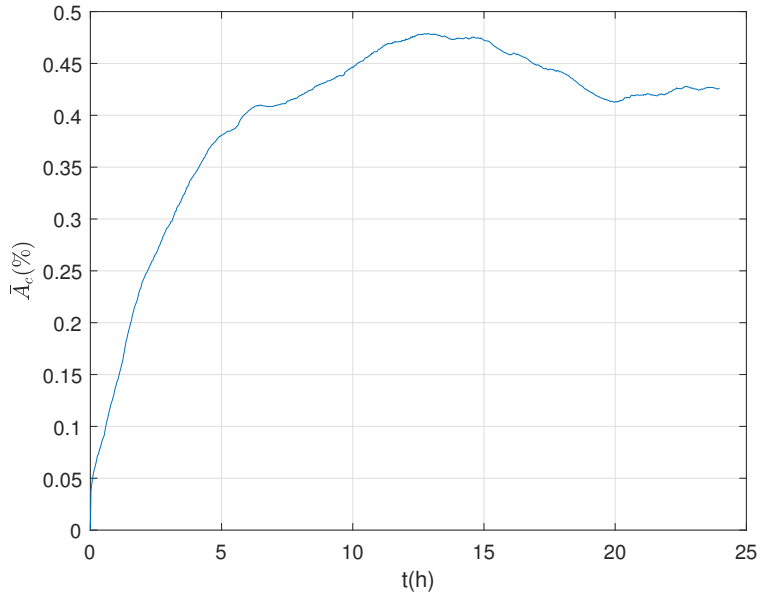
where ζ is the rate of change in which the interest function should increase back to 1 after a balloon has passed by. In our model, we established $\zeta = 1.1574 * 10^{-5}\ \mathrm{s}^{-1}$ to accommodate a slow time decay into the time span of 24 hours.

The simulation results for this scenario are presented in figure 5.9, where you can observe a noisy curve that has a growing tendency with some decrease after some time. This behavior indicates the continuous time-restoring feature. In this advanced configuration, we have introduced a more realistic attribute that aligns with the requirement outlined in section 5.2, where the hurricane area exhibits a time-restoring characteristic that prompts the system to remap specific regions while assigning quantitative priorities.

As evident from figure 5.9, the control protocol adeptly adjusts balloon positions in real-time to accommodate the varying values assigned to each region. Simultaneously, it manages the communication cost, which will be analyzed in the forthcoming sections.

In the subsequent section, we further enhance the restoring feature of the interest function by updating it with a differential equation based on the Fokker-Planck equation. This advancement allows the function to relax in a smoother manner, with spatial dependency considerations as well.

Figure 5.9 – Time evolution of the area covered percentage with continuous interest function and linear decay



5.4.4 Full Model Validation

The codes and results used in this work are available online[2].

A number of $N = 9$ balloons was used (including the base station). The simulation was run for $t_{max} = 48$ h. We used a total of 501 time samples and 61 space steps per dimension in a grid with 200 km range.

The communication range of the balloons was set as $\rho = 10$ km in accordance to the LoRa capacities as described in section 5.2.

In terms of area coverage, we set $\sigma = 100$ km and $\eta_i = 10$ km $\forall i \in \{1, \ldots, N\}$. In the Fokker-Planck equation, $D_{11} = D_{22} = 1$.

The hurricane parameters, used in equation (5.4), were set as $\varphi = 1.8$, $\chi = 0.5$, $V_m = 60$ m/s and $R_V = 20$ km based on a baseline profile described in Holland, Belanger, and Fritz (2010).

The control parameters were set as level height $\bar{z} = 558.3$ m to be consistent with the three-level control described in Meneghello, Luchini, and Bewley (2016), Meneghello, Luchini, and Bewley (2018). Also in accordance with those works, $\alpha = 10^{-3}$ s^{-1}, $c^2 = 1500$ m^2/s, $\beta_{ac} = 1$, and $\beta_{cc} = 1$. The number of neural network neurons were set as $M = 6$ and the predictive horizon as $h = 10$.

The first simulation was performed with $T_{cc} = T = 6$ h. Figure 5.10 shows 6 instances of the simulation separated by 100 time steps. Each subfigure shows the heatmap of the

[2] Available at https://github.com/brunofloriano/NN_MPC_application

interest function at that particular time. Each point represents one of the i-th balloons. Observe the contrasting patterns in the figures, some depicting the balloons in clusters and others showing them spread across the hurricane's area. This distinction directly reflects the coverage/communication trade-off, whereby the balloons periodically and interchangeably gather to establish communication links, and then disperse to extend the covered area. This dynamic adaptation enables the system to address each objective according to the specific needs of the moment.

In addition to figure 5.10, table 5.1 complements the information presented. For each time step indicated in the figure, we provide a communication relative size, denoted by a numerical value. This parameter, referred to as the communication relative size, represents the proportion of the communication energy $V_{cc}(\boldsymbol{q}(t),t)$ relative to the total energy vector $V(\boldsymbol{q}(t),t)$, and it is calculated as follows:

$$\bar{V}_{cc}(\boldsymbol{q}(t),t) = \frac{V_{cc}(\boldsymbol{q}(t),t)}{V(\boldsymbol{q}(t),t)}. \tag{5.19}$$

A smaller value indicates that the balloons are positioned closer together, prioritizing the establishment of communication links. Conversely, higher values correspond to situations where the balloons are spread out, emphasizing area coverage. In this chapter, we have defined a threshold of $\bar{V}_{cc}(\boldsymbol{q}(t),t) = -0.25$ to categorize the scenarios: above this value, the system is considered to be in a scattering situation (agents are spread out), while below this value, it is considered to be clustered. The exception occurs at $k = 0$ (the initial condition), when the system is in its early stage and coverage and communication may have similar priorities, given their low achievement levels at this time point.

In order to perform a correct analysis on the area coverage cycles, let the cumulative interest be given by

$$\bar{A}_n(\boldsymbol{q}(t),t) = \min_{\tau} A_n(\boldsymbol{q}(\tau),\tau), \tag{5.20}$$

where $\tau \in \begin{bmatrix} t - T_{ac} & t \end{bmatrix}$ and T_{ac} is the area coverage period. Then, the cumulative area covered can be defined as

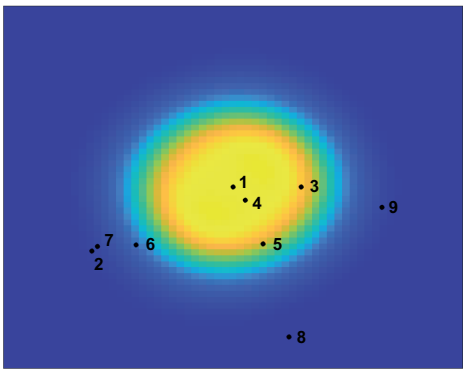
$$\bar{A}_c(\boldsymbol{q}(t),t) = 1 - \frac{\bar{A}_n(\boldsymbol{q}(t),t)}{A_T}. \tag{5.21}$$

Table 5.1 – Communication relative size, $\bar{V}_{cc}(\boldsymbol{q}(t_k),t_k)$, at each time step k. The maximum value occurs when the balloons are spread

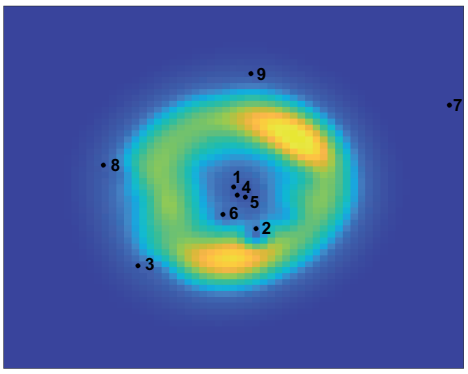
Time step k	$\bar{V}_{cc}(\boldsymbol{q}(t_k),t_k)$	Situation
0	-0.0330	Initial condition
100	**-0.1991**	**Spread**
200	**-0.2126**	**Spread**
300	**-0.1481**	**Spread**
400	-0.3053	Clustered
500	-0.4051	Clustered

Figure 5.10 – Interest function at different time steps. Compare the differences when the ballons are: clustered (indicating the communication priority) or spread (area coverage priority).

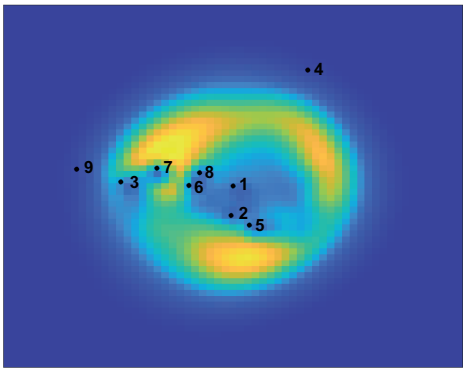
(a) $k = 0$, initial condition



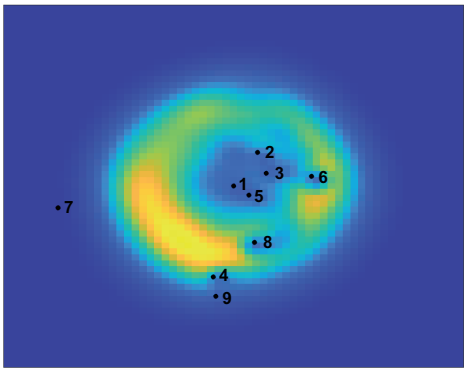
(b) $k = 100$, **spread**



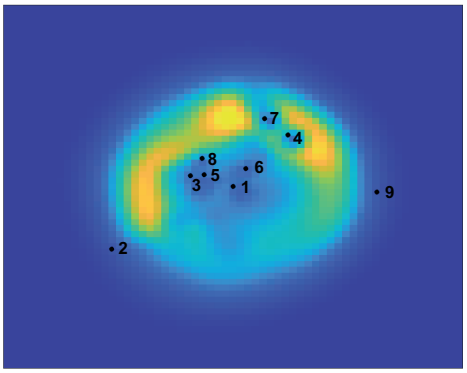
(c) $k = 200$, **spread**



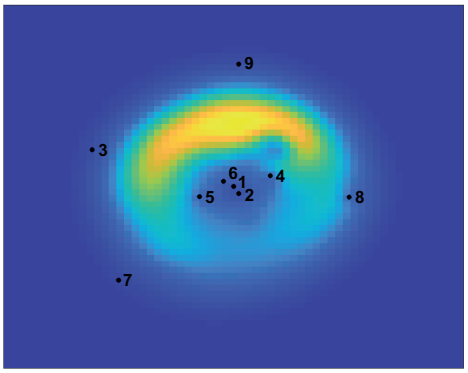
(d) $k = 300$, **spread**



(e) $k = 400$, clustered

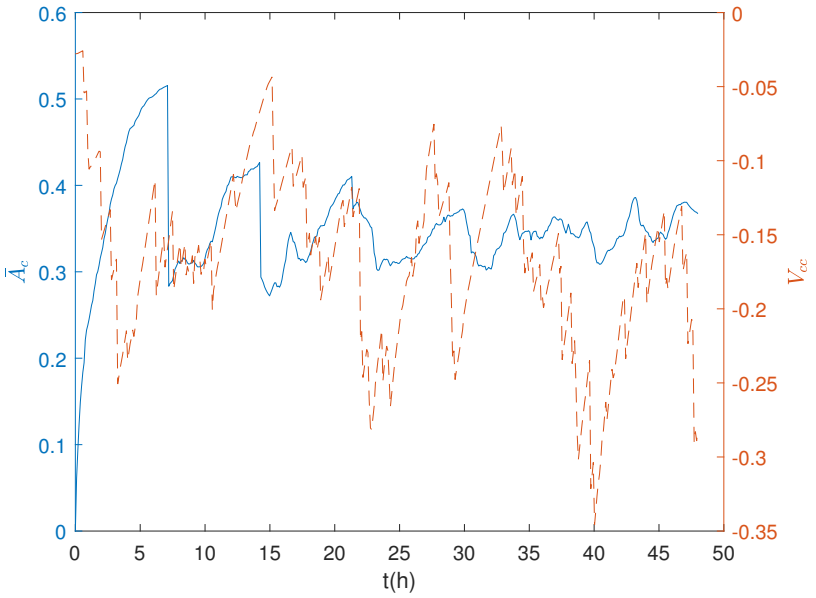


(f) $k = 500$, final state, clustered



Therefore, our analysis will focus on the cycles of the same duration, i.e. $T_{ac} = T_{cc} = T = 6$ h. Figure 5.11 shows the cumulative area covered, $\bar{A}_c$ and the communication energy, V_{cc}, with $T_{ac} = T_{cc} = T = 6$ h by the system over time.

Figure 5.11 – Cummulative area covered by the system, $\bar{A}_c$ (solid line), and communication energy, V_{cc} (dashed line), over time with $T = 6$ h. Notice the periodicity in both curves, indicating the compromise between area coverage and communication.



Notice the oscillating tendency of both curves. This is due to the effect of the restoring cost function, powered by the linear decay in the communication component of equation (5.14), and the dynamics of interest function governed by the Fokker-Planck equation described in section 5.3.1.2, which makes the interest function relax to its initial state.

As expected, both curves tend to follow the same pattern. As the balloons cover the interest area, $\bar{A}_c(\boldsymbol{q}(t),t)$ increases. However, it also increases the communication energy, due to the fact that past connections are now greatly degraded. Such energy is decreased when the agents reposition for connection, at the expense of losing previously covered areas.

As an illustration, notice figure 5.12. It shows the exact point in time where the communication energy is minimum (step 418). It is possible to recognize how the balloons are positioned in clusters (agents 1, 4, 5, 6, and 8 comprise one, while 3, 7, and 9 comprise another, with 2 being isolated). This step supports the system's arrangement that seeks to augment the communication to the detriment of area coverage. After that moment, the system is rearranged once more, and the balloons disperse to start covering the area again.

Figure 5.13 shows the cumulative area covered, $\bar{A}_c$ and the communication energy, V_{cc}, with $T_{ac} = T_{cc} = T = 12$ h by the system over time. The longer time period improves the visualization of the synchronization between the two objectives.

For the next portion, we will reduce the number of agents in order to more clearly

Figure 5.12 – Interest function when communication energy is at a minimum. Notice how the balloons are gathered in clusters, indicating they are prioritizing communication.

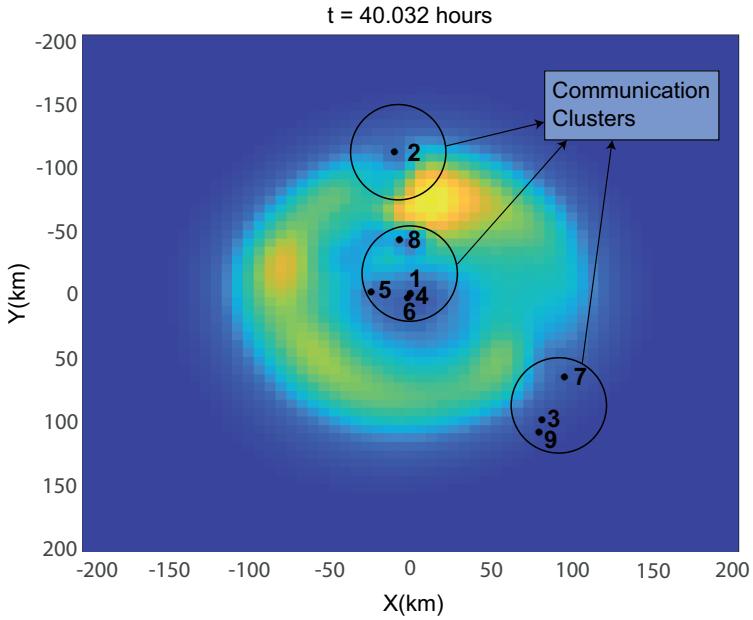
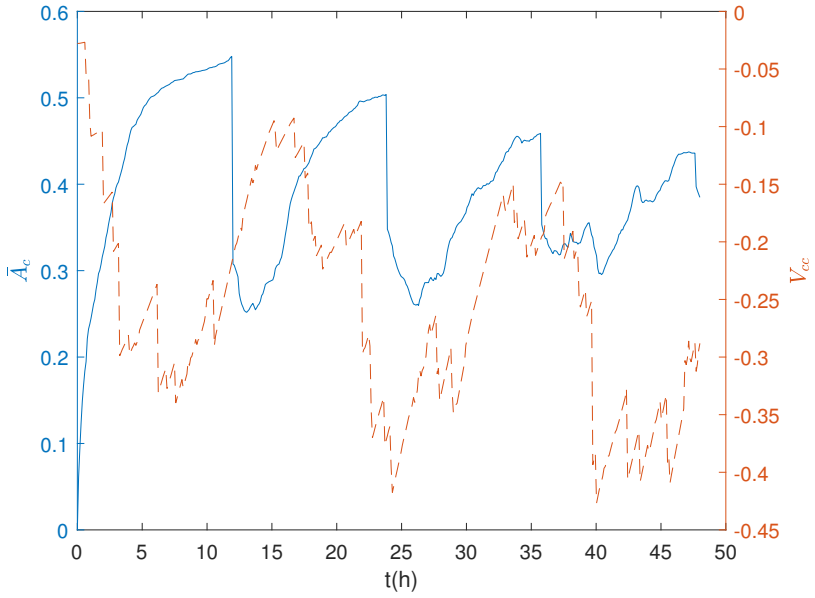


Figure 5.13 – Cumulative area covered by the system, $\bar{A}_c$ (solid line), and communication energy, V_{cc} (dashed line), over time with $T = 12$ h. Notice the synchronization between the two objectives.

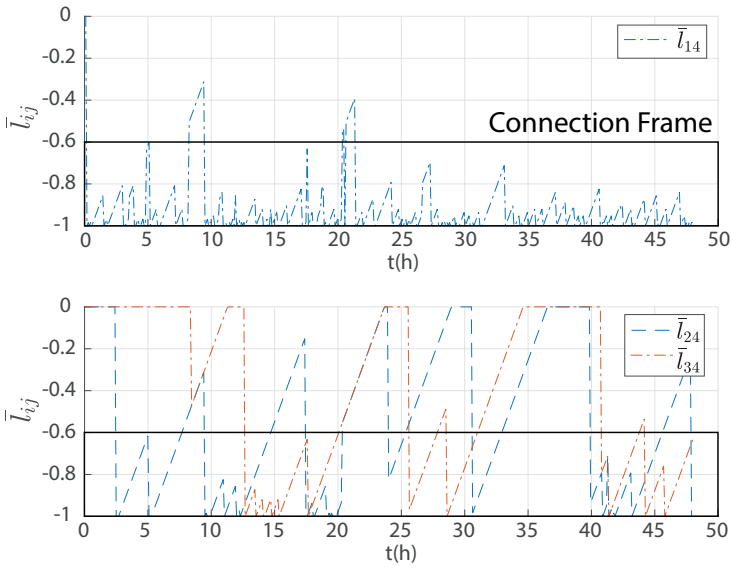


illustrate the message routing process. This will allow us to better visualize how the agents redirect messages when necessary. With fewer agents, this is easier to demonstrate and is more illustrative. Let us use the same parameters as before, except for $N = 4$ and $T = 6$ h. Here we introduce the concept of “connection frame", which is the window in which a connection still has more than 60% of its strength; in other words, $\bar{l}_{ij}$ remains between -0.6 and -1. This interval serves as an indicator of the connection’s steadiness between agents i

and j and provides insight into the consistency of their communication link.

Figure 5.14 shows the entries $\bar{l}_{ij}$ of the cumulative Laplacian matrix $\bar{L}(\boldsymbol{q}(t),t)$. In this example, agent $j = 4$ acts as the routing agent, which is why its connection to the base station (represented by $\bar{l}_{14}$) is almost constant, as seen in the top graph. During most of the simulation, $\bar{l}_{14}$ is inside the connection frame. In addition, notice in the bottom graph, that agents 2 and 3, which are coverage-only balloons, have intermittent connections with agent 4 to periodically send their data so they can ultimately reach the base station. Therefore, it was possible to clearly observe the message being forwarded between the balloons in order to send the data to the base station and guarantee the agents' observations are ultimately sent to a satellite.

Figure 5.14 – Entries $\bar{l}_{ij}$ of the cumulative Laplacian matrix $\bar{L}(\boldsymbol{q}(t),t)$. Notice how the router $j = 4$ is mostly connected to the base station ($i = 1$) and have intermittent connections with other agents.



5.5 Remarks

This chapter introduced a multi-agent control architecture for a swarm of heterogeneous low-cost buoyancy-driven balloons to monitor and collect weather data of a hurricane. The proposed method is composed of two constantly restoring cost function components, one to represent the communication energy and another for the area coverage energy. The control scheme builds upon the neural network-based MPC and TLC to generate an adaptive multi-agent framework that optimizes the cost function.

The proposed method was tested in a simulated environment with two distinct periods, for a time of 48h of coverage each. In both cases, it was possible to clearly distinguish the cycles and the expected synchronization between area coverage and communication. Therefore the concession between those conflicting objectives was fulfilled as the MAS

adaptively switched between them to better accommodate the problem.

The hurricane monitoring system, presented in this chapter, as well as its results, has been submitted for publication in the journal Engineering Applications of Artificial Intelligence. The journal was ranked as A1 by the CAPES' Qualis classification system in the quadrennium 2017-2020.

6 Conclusion

This work proposed a novel neural-network-based model predictive control algorithm so that a nonlinear multi-agent system reaches consensus with stochastic switching topologies. This approach seeks to find a feedback control matrix at each instant k such that a model predictive quadratic cost function is minimized, allowing the agents to reach stability. It stands out from the current MAS literature by adopting an adaptive strategy, which allows the system to readjust to real-time changes in the environment as well as to random communication switching.

The MPC approach is based on predicting future states up to a given horizon, acknowledging the uncertainties caused by the communication limitations. This way, a quadratic cost function is built based on the expected value of the predicted state and can be minimized as a function of the feedback control matrix.

Achieving such optimization was possible thanks to the universal approximation property of the neural network, used in this work to perform a regression of the cost as a function of the matrix. A cross-validation procedure was adopted in order to choose the number of neurons that would achieve a minimum mean-squared error.

The results were extensively tested in both linear and nonlinear systems, with different communication scenarios and compared to the literature. As seen in chapter 4, the NN-based MPC resulted in a faster response, with a decreased settling time. However, the trade-off lies in the mean-squared error which presents larger values for some states. The simulations also included systems with relevant practical value such as quadrotors and robot cars, showing the approach's applicability for real situations.

It is important to notice that the proposed method was built for stochastic switching and nonlinear dynamics. Therefore, particular cases of those characteristics, for example deterministic switching or linear dynamics with or without disturbances, are also included and can achieve consensus with the NN-based MPC algorithm.

Furthermore, the robustness and efficacy of the proposed architecture were rigorously assessed in a larger and more intricate scenario, as detailed in chapter 5. In this comprehensive test case, we employed the NN-based MPC protocol to orchestrate the monitoring of a hurricane using a swarm of buoyancy-driven balloons.

The hurricane, as a naturally occurring extreme environmental phenomenon, presented a challenge for our model. The vehicles within this swarm faced significant noise, and disturbances, and operated within an expansive environment spanning approximately 100 km in radius. These were challenges that had not been comprehensively addressed in prior literature.

As evidenced by the results, our protocol demonstrated exceptional performance in fulfilling the dual requirements of area coverage and communication, despite the inherent conflict between these objectives. This achievement underscores the capability of the NN-MPC approach to effectively coordinate complex operations within multi-agent systems operating under communication constraints.

In summary, our research has illustrated that the NNMPC approach is well-suited for managing the intricate dynamics and communication demands of multi-agent systems, particularly in challenging and real-world scenarios like hurricane monitoring.

Future research directions include validating the proposed protocol on real platforms to assess its performance in actual vehicles, especially focusing on adapting scenarios to address environmental protection issues of national relevance like wildfires and deforestation in the Amazon rainforest. The need for deep theoretical validation of the algorithm's conditions and performance is emphasized. Additionally, suggestions for enhancing the current system involve integrating a robust adaptation component into the neural network learning process to improve adaptability, optimizing communication protocols and refining algorithmic parameters. Furthermore, exploring the application of deep learning and other artificial intelligence techniques is recommended to broaden the range of solutions for MAS problems. These research endeavors collectively aim to strengthen the reliability, adaptability, and practical applicability of the proposed protocol in diverse and dynamic contexts.

References

ABIODUN, O. I.; JANTAN, A.; OMOLARA, A. E.; DADA, K. V.; MOHAMED, N. A.; ARSHAD, H. State-of-the-art in artificial neural network applications: A survey. **Heliyon**, Elsevier, v. 4, n. 11, p. e00938, 2018. Cit. on p. 26.

AFGHAH, F.; RAZI, A.; CHAKARESKI, J.; ASHDOWN, J. Wildfire Monitoring in Remote Areas using Autonomous Unmanned Aerial Vehicles. *In*: IEEE INFOCOM 2019 - IEEE Conference ON Computer Communications Workshops (INFOCOM WKSHPS). Paris, France: IEEE, 2019. p. 835–840. ISBN 978-1-72811-878-9. Cit. on p. 55.

ALPAYDIN, E. **Introduction to machine learning**. MIT press, 2020. Cit. on pp. 26, 27, 28, 29, 30, 31, and 41.

BALDAZO, D.; PARRAS, J.; ZAZO, S. Decentralized Multi-Agent Deep Reinforcement Learning in Swarms of Drones for Flood Monitoring. *In*: 2019 27TH European Signal Processing Conference (EUSIPCO). A Coruna, Spain: IEEE, 2019. p. 1–5. ISBN 978-90-827970-3-9. Cit. on p. 55.

BAPAT, R. B. **Graphs and matrices**. Springer, 2010. v. 27. Cit. on p. 23.

BEWLEY, T.; MENEGHELLO, G. Efficient coordination of swarms of sensor-laden balloons for persistent, in situ, real-time measurement of hurricane development. **Physical Review Fluids**, APS, v. 1, n. 6, p. 060507, 2016. Cit. on pp. 55 and 56.

BISWAS, B.; CHATTERJEE, S.; MUKHERJEE, S.; PAL, S. A discussion on euler method: A review. **Electronic Journal of Mathematical Analysis and Applications**, v. 1, n. 2, p. 2090–2792, 2013. Cit. on pp. 44 and 51.

CHEN, S.; SAULNIER, K.; ATANASOV, N.; LEE, D. D.; KUMAR, V.; PAPPAS, G. J.; MORARI, M. Approximating explicit model predictive control using constrained neural networks. *In*: 2018 IEEE ANNUAL AMERICAN CONTROL CONFERENCE (ACC). 2018. p. 1520–1527. Cit. on p. 20.

CHENG, L.; LIU, W.; HOU, Z.-G.; YU, J.; TAN, M. Neural-network-based nonlinear model predictive control for piezoelectric actuators. **IEEE Transactions on Industrial Electronics**, IEEE, v. 62, n. 12, p. 7717–7727, 2015. Cit. on p. 20.

CHENG, Z.; ZHANG, H.-T.; FAN, M.-C.; CHEN, G. Distributed consensus of multi-agent systems with input constraints: A model predictive control approach. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEE, v. 62, n. 3, p. 825–834, 2014. Cit. on pp. 26 and 33.

CHOI, S. Y.; CHA, D. Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art. **Advanced Robotics**, Taylor & Francis, v. 33, n. 6, p. 265–277, 2019. Cit. on p. 27.

CHONG, E. K.; ZAK, S. H. **An introduction to optimization**. John Wiley & Sons, 2004. Cit. on p. 39.

CIONE, J. J.; KALINA, E.; UHLHORN, E.; FARBER, A.; DAMIANO, B. Coyote unmanned aircraft system observations in hurricane edouard (2014). **Earth and Space Science**, Wiley Online Library, v. 3, n. 9, p. 370–380, 2016. Cit. on p. 55.

CONGRESS, S.; PUPPALA, A. J.; BANERJEE, A.; JAFARI, N. H.; PATIL, U. D. Use of unmanned aerial photogrammetry for monitoring low-volume roads after hurricane harvey. *In*: 12TH INTERNATIONAL CONFERENCE ON LOW-VOLUME ROADS. 2019. v. 530. Cit. on p. 55.

COSTA, O. L.; FRAGOSO, M. D. Discrete-time lq-optimal control problems for infinite Markov jump parameter systems. **IEEE Transactions on Automatic Control**, IEEE, v. 40, n. 12, p. 2076–2088, 1995. Cit. on p. 23.

DAS, A.; GERVET, T.; ROMOFF, J.; BATRA, D.; PARIKH, D.; RABBAT, M.; PINEAU, J. Tarmac: Targeted multi-agent communication. *In*: INTERNATIONAL CONFERENCE ON MACHINE LEARNING. 2019. p. 1538–1546. Cit. on p. 57.

DAY, R.; SALMON, J. A framework for multi-uav persistent search and retrieval with stochastic target appearance in a continuous space. **Journal of Intelligent & Robotic Systems**, Springer, v. 103, n. 4, p. 65, 2021. Cit. on p. 57.

DONG, L.; NGUANG, S. K. **Consensus Tracking of Multi-agent Systems with Switching Topologies**. Academic Press, 2020. Cit. on p. 25.

ESHAGHI, K.; NEJAT, G.; BENHABIB, B. A concurrent mission-planning methodology for robotic swarms using collaborative motion-control strategies. **Journal of Intelligent & Robotic Systems**, Springer, v. 108, n. 2, p. 15, 2023. Cit. on p. 57.

FERRARI-TRECATE, G.; GALBUSERA, L.; MARCIANDI, M. P. E.; SCATTOLINI, R. Model predictive control schemes for consensus in multi-agent systems with single-and double-integrator dynamics. **IEEE Transactions on Automatic Control**, IEEE, v. 54, n. 11, p. 2560–2572, 2009. Cit. on p. 33.

FLORIANO, B. R.; VARGAS, A. N.; ISHIHARA, J. Y.; FERREIRA, H. C. Neural-network-based model predictive control for consensus of nonlinear systems. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 116, p. 105327, 2022. Cit. on pp. 22, 38, 57, 58, 60, 65, and 67.

FLORIANO, B. R. O.; BORGES, G. A.; FERREIRA, H. C.; ISHIHARA, J. Y. Hybrid Dec-POMDP/PID guidance system for formation flight of multiple UAVs. **Journal of**

Intelligent & Robotic Systems, Springer, v. 101, n. 3, p. 1–20, 2021. Cit. on pp. 19 and 55.

GAO, J.; LI, J.; PAN, H.; WU, Z.; YIN, X.; WANG, H. Consensus via event-triggered strategy of nonlinear multi-agent systems with Markovian switching topologies. **ISA transactions**, Elsevier, v. 104, p. 122–129, 2020. Cit. on pp. 9, 19, 21, 24, 25, 39, 44, 45, and 46.

GHAZALI, M. H. M.; TEOH, K.; RAHIMAN, W. A Systematic Review of Real-Time Deployments of UAV-Based LoRa Communication Network. **IEEE Access**, v. 9, p. 124817–124830, 2021. ISSN 2169-3536. Cit. on p. 56.

GONG, B.; JIANG, T. A tree-based routing protocol in wireless sensor networks. *In*: 2011 IEEE INTERNATIONAL CONFERENCE ON ELECTRICAL AND CONTROL ENGINEERING. 2011. p. 5729–5732. Cit. on p. 62.

GREENWOOD, F.; NELSON, E. L.; GREENOUGH, P. G. Flying into the hurricane: A case study of uav use in damage assessment during the 2017 hurricanes in texas and florida. **PLoS one**, Public Library of Science San Francisco, CA USA, v. 15, n. 2, p. e0227808, 2020. Cit. on p. 55.

HAN, H.-G.; ZHANG, L.; HOU, Y.; QIAO, J.-F. Nonlinear model predictive control based on a self-organizing recurrent neural network. **IEEE transactions on neural networks and learning systems**, IEEE, v. 27, n. 2, p. 402–415, 2015. Cit. on p. 20.

HAN, S. I. Prescribed consensus and formation error constrained finite-time sliding mode control for multi-agent mobile robot systems. **IET Control Theory & Applications**, v. 12, n. 2, p. 282–290, 2018. Cit. on p. 19.

HOLLAND, G. J.; BELANGER, J. I.; FRITZ, A. A revised model for radial profiles of hurricane winds. **Monthly weather review**, v. 138, n. 12, p. 4393–4401, 2010. Cit. on pp. 57, 59, and 72.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural networks**, Elsevier, v. 4, n. 2, p. 251–257, 1991. Cit. on p. 30.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural networks**, Elsevier, v. 2, n. 5, p. 359–366, 1989. Cit. on pp. 29 and 30.

HU, J.; NIU, H.; CARRASCO, J.; LENNOX, B.; ARVIN, F. Fault-tolerant cooperative navigation of networked UAV swarms for forest fire monitoring. **Aerospace Science and Technology**, v. 123, p. 107494, Apr. 2022. ISSN 12709638. Cit. on p. 55.

HU, W.; LIU, L.; FENG, G. Cooperative Output Regulation of Linear Multi-Agent Systems by Intermittent Communication: A Unified Framework of Time- and Event-Triggering Strategies. **IEEE Transactions on Automatic Control**, v. 63, n. 2, p. 548–555, Feb. 2018. ISSN 0018-9286, 1558-2523. Cit. on p. 57.

HUYNH, B.-P.; WU, C.-W.; KUO, Y.-L. Force/position hybrid control for a hexa robot using gradient descent iterative learning control algorithm. **IEEE Access**, IEEE, v. 7, p. 72329–72342, 2019. Cit. on p. 39.

KAVIARASAN, B.; SAKTHIVEL, R.; WANG, C.; ALZAHRANI, F. Resilient control design for consensus of nonlinear multi-agent systems with switching topology and randomly varying communication delays. **Neurocomputing**, Elsevier, v. 311, p. 155–163, 2018. Cit. on p. 19.

KOUVARITAKIS, B.; CANNON, M. Model predictive control. **Switzerland: Springer International Publishing**, Springer, v. 38, 2016. Cit. on pp. 31 and 33.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Cit. on p. 26.

LEE, I.; SHIN, Y. J. Machine learning for enterprises: Applications, algorithm selection, and challenges. **Business Horizons**, Elsevier, v. 63, n. 2, p. 157–170, 2020. Cit. on p. 26.

LEWIS, F. L.; ZHANG, H.; HENGSTER-MOVRIC, K.; DAS, A. **Cooperative control of multi-agent systems: optimal and adaptive design approaches**. New York, NY, USA: Springer Science & Business Media, 2013. Cit. on p. 19.

LI, H.; ZHANG, J.; JING, L.; YING, W. Neural-network-based adaptive quasi-consensus of nonlinear multi-agent systems with communication constrains and switching topologies. **Nonlinear Analysis: Hybrid Systems**, Elsevier, v. 35, p. 100833, 2020. Cit. on pp. 19 and 20.

LI, X.-M.; ZHOU, Q.; LI, P.; LI, H.; LU, R. Event-triggered consensus control for multi-agent systems against false data-injection attacks. **IEEE Transactions on Cybernetics**, IEEE, v. 50, n. 5, p. 1856–1866, 2019. Cit. on p. 19.

LI, Z.; DUAN, Z. **Cooperative control of multi-agent systems: a consensus region approach**. CRC press, 2017. Cit. on p. 25.

LIN, Z.; LIU, H. H. Topology-based distributed optimization for multi-UAV cooperative wildfire monitoring. **Optimal Control Applications and Methods**, v. 39, n. 4, p. 1530–1548, Jul. 2018. ISSN 01432087. Cit. on p. 55.

LIU, L.; LI, B.; GUO, R. Consensus control for networked manipulators with switched parameters and topologies. **IEEE Access**, IEEE, v. 9, p. 9209–9217, 2021. Cit. on p. 19.

LIU, W.; HUANG, J. Adaptive leader-following consensus for a class of higher-order nonlinear multi-agent systems with directed switching networks. **Automatica**, Elsevier, v. 79, p. 84–92, 2017. Cit. on p. 19.

MANOHARAN, A.; SUJIT, P. Nonlinear model predictive control framework for cooperative three-agent target defense game. **arXiv preprint arXiv:2207.09136**, 2022. Cit. on p. 57.

MATUTE, J. A.; MARCANO, M.; DIAZ, S.; PEREZ, J. Experimental validation of a kinematic bicycle model predictive control with lateral acceleration consideration. **IFAC-PapersOnLine**, Elsevier, v. 52, n. 8, p. 289–294, 2019. Cit. on p. 50.

MENEGHELLO, G.; BEWLEY, T.; JONG, M. de; BRIGGS, C. A coordinated balloon observation system for sustained in-situ measurements of hurricanes. *In*: 2017 IEEE AEROSPACE CONFERENCE. 2017. p. 1–6. Cit. on p. 56.

MENEGHELLO, G.; LUCHINI, P.; BEWLEY, T. On the control of buoyancy-driven devices in stratified, uncertain flowfields. *In*: INTERNATIONAL SYMPOSIUM ON STRATIFIED FLOWS. 2016. v. 1, n. 1. Cit. on pp. 21, 55, 56, 58, 66, and 72.

MENEGHELLO, G.; LUCHINI, P.; BEWLEY, T. A probabilistic framework for the control of systems with discrete states and stochastic excitation. **Automatica**, Elsevier, v. 88, p. 113–116, 2018. Cit. on pp. 21, 56, 58, 66, and 72.

MING, P.; LIU, J.; TAN, S.; LI, S.; SHANG, L.; YU, X. Consensus stabilization in stochastic multi-agent systems with Markovian switching topology, noises and delay. **Neurocomputing**, Elsevier, v. 200, p. 1–10, 2016. Cit. on pp. 19 and 24.

MOHSAN, S. A. H.; OTHMAN, N. Q. H.; LI, Y.; ALSHARIF, M. H.; KHAN, M. A. Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends. **Intelligent Service Robotics**, Springer, p. 1–29, 2023. Cit. on p. 55.

NAMARA, P. M.; NEGENBORN, R. R.; SCHUTTER, B. D.; LIGHTBODY, G. Weight optimisation for iterative distributed model predictive control applied to power networks. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 26, n. 1, p. 532–543, 2013. Cit. on p. 20.

OLFATI-SABER, R.; MURRAY, R. M. Consensus problems in networks of agents with switching topology and time-delays. **IEEE Transactions on automatic control**, IEEE, v. 49, n. 9, p. 1520–1533, 2004. Cit. on p. 19.

PETAJAJARVI, J.; MIKHAYLOV, K.; ROIVAINEN, A.; HANNINEN, T.; PETTISSALO, M. On the coverage of lpwans: range evaluation and channel attenuation model for lora technology. *In*: 2015 IEEE 14TH INTERNATIONAL CONFERENCE ON ITS TELECOMMUNICATIONS (ITST). 2015. p. 55–59. Cit. on p. 59.

Puente-Castro, A.; RIVERO, D.; PAZOS, A.; Fernandez-Blanco, E. A review of artificial intelligence applied to path planning in UAV swarms. **Neural Computing and Applications**, v. 34, n. 1, p. 153–170, Jan. 2022. ISSN 0941-0643, 1433-3058. Cit. on p. 57.

QIU, J.; MA, M.; WANG, T.; GAO, H. Gradient descent-based adaptive learning control for autonomous underwater vehicles with unknown uncertainties. **IEEE Transactions**

on Neural Networks and Learning Systems, IEEE, v. 32, n. 12, p. 5266–5273, 2021. Cit. on p. 39.

QUERALTA, J. P.; TAIPALMAA, J.; PULLINEN, B. C.; SARKER, V. K.; GIA, T. N.; TENHUNEN, H.; GABBOUJ, M.; RAITOHARJU, J.; WESTERLUND, T. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. **IEEE Access**, IEEE, v. 8, p. 191617–191643, 2020. Cit. on p. 57.

RAJAMANI, R. **Vehicle dynamics and control**. Springer Science & Business Media, 2011. Cit. on p. 50.

Ramirez-Atencia, C.; CAMACHO, D. Constrained multi-objective optimization for multi-UAV planning. **Journal of Ambient Intelligence and Humanized Computing**, v. 10, n. 6, p. 2467–2484, Jun. 2019. ISSN 1868-5137, 1868-5145. Cit. on p. 57.

REN, W.; BEARD, R. W.; ATKINS, E. M. Information consensus in multivehicle cooperative control. **IEEE Control systems magazine**, IEEE, v. 27, n. 2, p. 71–82, 2007. Cit. on pp. 19 and 23.

ROJAS, S. S.; KHAN, S. D.; SHAHTAKHTINSKIY, A. Impact of hurricane harvey on the upper texas coast: Using airborne lidar data sets with uav-derived topographic data to monitor change and track recovery. **Remote Sensing**, MDPI, v. 14, n. 21, p. 5357, 2022. Cit. on p. 55.

RUSSEL, S.; NORVIG, P. **Artificial intelligence—a modern approach 3rd Edition**. Prentice Hall Press, 2012. Cit. on pp. 26 and 29.

SAKTHIVEL, R.; KANAKALAKSHMI, S.; KAVIARASAN, B.; MA, Y.-K.; LEELAMANI, A. Finite-time consensus of input delayed multi-agent systems via non-fragile controller subject to switching topology. **Neurocomputing**, Elsevier, v. 325, p. 225–233, 2019. Cit. on p. 19.

SANCHEZ-IBORRA, R.; CANO, M.-D. State of the art in lp-wan solutions for industrial iot services. **Sensors**, MDPI, v. 16, n. 5, p. 708, 2016. Cit. on pp. 57 and 59.

SARAEREH, O. A.; ALSARAIRA, A.; KHAN, I.; UTHANSAKUL, P. Performance Evaluation of UAV-Enabled LoRa Networks for Disaster Management Applications. **Sensors**, v. 20, n. 8, p. 2396, Apr. 2020. ISSN 1424-8220. Cit. on p. 56.

SAVINO, H. J.; SANTOS, C. R. dos; SOUZA, F. O.; PIMENTA, L. C.; OLIVEIRA, M. de; PALHARES, R. M. Conditions for consensus of multi-agent systems with time-delays and uncertain switching topology. **IEEE Transactions on Industrial Electronics**, IEEE, v. 63, n. 2, p. 1258–1267, 2015. Cit. on pp. 19, 24, and 40.

SCHAEFER, M.; TEEUW, R.; DAY, S.; ZEKKOS, D.; WEBER, P.; MEREDITH, T.; WESTEN, C. J. V. Low-cost uav surveys of hurricane damage in dominica: automated processing with co-registration of pre-hurricane imagery for change analysis. **Natural hazards**, Springer, v. 101, n. 3, p. 755–784, 2020. Cit. on p. 55.

SERAJ, E.; SILVA, A.; GOMBOLAY, M. Multi-UAV planning for cooperative wildfire coverage and tracking with quality-of-service guarantees. **Autonomous Agents and Multi-Agent Systems**, v. 36, n. 2, p. 39, Oct. 2022. ISSN 1387-2532, 1573-7454. Cit. on p. 55.

SHI, Y.; HU, J.; WU, Y.; GHOSH, B. K. Intermittent output tracking control of heterogeneous multi-agent systems over wide-area clustered communication networks. **Nonlinear Analysis: Hybrid Systems**, v. 50, p. 101387, Nov. 2023. ISSN 1751570X. Cit. on p. 57.

SOHRAB, H. H. **Basic real analysis**. Springer, 2003. v. 231. Cit. on p. 25.

SOLTERO, D. E.; SCHWAGER, M.; RUS, D. Decentralized path planning for coverage tasks using gradient descent adaptive control. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 33, n. 3, p. 401–425, 2014. Cit. on p. 39.

STAMPA, M.; SUTORMA, A.; JAHN, U.; THIEM, J.; WOLFF, C.; RÖHRIG, C. Maturity levels of public safety applications using unmanned aerial systems: a review. **Journal of Intelligent & Robotic Systems**, Springer, v. 103, p. 1–15, 2021. Cit. on p. 55.

SU, Y.; SHI, Y.; SUN, C. Distributed model predictive control for tracking consensus of linear multiagent systems with additive disturbances and time-varying communication delays. **IEEE transactions on cybernetics**, IEEE, v. 51, n. 7, p. 3813–3823, 2019. Cit. on pp. 26 and 33.

TANG, J.; DUAN, H.; LAO, S. Swarm intelligence algorithms for multiple unmanned aerial vehicles collaboration: A comprehensive review. **Artificial Intelligence Review**, v. 56, n. 5, p. 4295–4327, May 2023. ISSN 0269-2821, 1573-7462. Cit. on p. 57.

TZOUMAS, G.; PITONAKOVA, L.; SALINAS, L.; SCALES, C.; RICHARDSON, T.; HAUERT, S. Wildfire detection in large-scale environments using force-based control for swarms of UAVs. **Swarm Intelligence**, v. 17, n. 1-2, p. 89–115, Jun. 2023. ISSN 1935-3812, 1935-3820. Cit. on p. 55.

VALCHER, M. E.; ZORZAN, I. On the consensus of homogeneous multi-agent systems with arbitrarily switching topology. **Automatica**, Elsevier, v. 84, p. 79–85, 2017. Cit. on p. 19.

VENKATESAN, S.; KAMARAJ, P.; VISHNUPRIYA, M. Speed control of permanent magnet synchronous motor using neural network model predictive control. **Journal of Energy Systems**, v. 4, n. 2, p. 71–87, 2020. Cit. on p. 33.

VISERAS, A.; MEISSNER, M.; MARCHAL, J. Wildfire Front Monitoring with Multiple UAVs using Deep Q-Learning. **IEEE Access**, p. 1–1, 2021. ISSN 2169-3536. Cit. on p. 55.

VIZCAYA-MARTÍNEZ, D. A.; SANTIAGO, F. Flores-de; VALDERRAMA-LANDEROS, L.; SERRANO, D.; RODRÍGUEZ-SOBREYRA, R.; ÁLVAREZ-SÁNCHEZ, L. F.; FLORES-VERDUGO, F. Monitoring detailed mangrove hurricane damage and early recovery

using multisource remote sensing data. **Journal of Environmental Management**, Elsevier, v. 320, p. 115830, 2022. Cit. on p. 55.

WANG, L.; WANG, K.; PAN, C.; XU, W.; ASLAM, N.; HANZO, L. Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing. **IEEE Transactions on Cognitive Communications and Networking**, IEEE, v. 7, n. 1, p. 73–84, 2020. Cit. on p. 57.

WANG, T.; GAO, H.; QIU, J. A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 27, n. 2, p. 416–425, 2015. Cit. on pp. 20, 33, and 34.

WANG, X.; WANG, H.; LI, C.; HUANG, T.; KURTHS, J. Consensus seeking in multiagent systems with Markovian switching topology under aperiodic sampled data. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, v. 50, n. 12, p. 5189–5200, 2020. Cit. on pp. 9, 19, 21, 39, 40, 41, 42, and 43.

WEN, G.; DUAN, Z.; REN, W.; CHEN, G. Distributed consensus of multi-agent systems with general linear node dynamics and intermittent communications: CONSENSUS OF LINEAR MULTI-AGENT SYSTEMS. **International Journal of Robust and Nonlinear Control**, v. 24, n. 16, p. 2438–2457, Nov. 2014. ISSN 10498923. Cit. on p. 57.

WU, Z.; RINCON, D.; CHRISTOFIDES, P. D. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. **Journal of Process Control**, Elsevier, v. 89, p. 74–84, 2020. Cit. on pp. 33 and 34.

XIAO, H.; CHEN, C. P. Incremental updating multirobot formation using nonlinear model predictive control method with general projection neural network. **IEEE Transactions on Industrial Electronics**, IEEE, v. 66, n. 6, p. 4502–4512, 2018. Cit. on pp. 20, 21, 24, and 34.

XIAO, S.; DONG, J. Distributed fault-tolerant tracking control for heterogeneous nonlinear multi-agent systems under sampled intermittent communications. **Journal of the Franklin Institute**, v. 358, n. 17, p. 9221–9242, Nov. 2021. ISSN 00160032. Cit. on p. 57.

YANG, H.; YIN, S.; KAYNAK, O. Neural network-based adaptive fault-tolerant control for Markovian jump systems with nonlinearity and actuator faults. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, 2020. Cit. on p. 20.

YEOM, J.; HAN, Y.; CHANG, A.; JUNG, J. Hurricane building damage assessment using post-disaster uav data. *In*: . 2019. p. 9867–9870. Cit. on p. 55.

ZHANG, J.; ZHANG, H.; FENG, T. Distributed optimal consensus control for nonlinear multiagent system with unknown dynamic. **IEEE transactions on neural networks**

and learning systems, IEEE, v. 29, n. 8, p. 3339–3348, 2017. Cit. on pp. 19, 20, 30, and 36.

ZHANG, P.; XUE, H.; GAO, S.; ZHANG, J. Distributed Adaptive Consensus Tracking Control for Multi-Agent System With Communication Constraints. **IEEE Transactions on Parallel and Distributed Systems**, v. 32, n. 6, p. 1293–1306, Jun. 2021. ISSN 1045-9219, 1558-2183, 2161-9883. Cit. on p. 57.

ZHAO, L.; YU, J.; LIN, C.; MA, Y. Adaptive neural consensus tracking for nonlinear multiagent systems using finite-time command filtered backstepping. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, v. 48, n. 11, p. 2003–2012, 2017. Cit. on pp. 19 and 20.

ZHONG, X.; HE, H.; ZHANG, H.; WANG, Z. A neural network based online learning and control approach for Markov jump systems. **Neurocomputing**, Elsevier, v. 149, p. 116–123, 2015. Cit. on pp. 20, 39, and 47.

ZOU, W.; SHI, P.; XIANG, Z.; SHI, Y. Consensus tracking control of switched stochastic nonlinear multiagent systems via event-triggered strategy. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 31, n. 3, p. 1036–1045, 2019. Cit. on p. 19.