Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

University of Padua

Department of Information Engineering DEI

Master degree in Control Systems Engineering

Bearing-only Formation Control of multiple UAVs with an NMPC approach: architectures and methodologies

---

Student

Ciresola Federico (2019290)

Supervising

Cenedese Angelo

Co-supervising

Fantoni Isabelle

Academic Year 2022 - 2023

"It is important to be a positive derivative in our lives . Cit. An old friend"

Abstract

One of the major fields of development and innovation in our century concerns mobile robotics, particularly the study of multi-agent system coordination. This type of technology can be employed in natural disaster scenarios or critical situations where multi-agent systems can perform search and rescue missions or environmental monitoring. In this field, autonomous ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) are used, which can cooperate with each other. Outdoor environments pose various challenges, and one of these challenges can be the lack of access to GPS network. For this reason, control laws that do not rely on GPS information are studied, such as "bearing-only formation control," where agents only perceive relative angles between each other, which can be calculated using a simple 2D camera.

The purpose of this thesis work is to explore the theory of bearing rigidity applied to formations of homogeneous agents, specifically quadcopters, using Nonlinear Model Predictive Control (NMPC) with MATMPC. The thesis aims to explore the differences between different architectures, particularly by implementing a centralized architecture and laying the groundwork for a possible development of a decentralized architecture. The work demonstrates how to create the model for NMPC with various agent model variants,different cost functions, and different formations. ROS2 is used for the implementation of the entire control ecosystem, and the PX4 platform is used, which provides an excellent low-level control architecture for managing individual quadcopters. Additionally, the thesis utilizes the PX4 Software In The Loop (SITL) system to make Gazebo simulations as close to reality as possible.

Keywords: bearing-only formation control, Nonlinear Model Predictive Control (NMPC), centralized architecture

Sommario

Uno dei maggior campi di sviluppo e d'innovazione nel nostro secolo riguarda l'ambito della robotica mobile in particolare lo studio del coordinamento di sistemi multi agenti. Questo tipo di tecnologia permette di essere impiegata in ambiti di calamità naturali o situazioni critiche dove i sistemi multi agenti posso operare missioni di *search and rescue* o monitoraggio ambientale. In campo vengono impiegati veicoli terrestri autonomi UGV (Unmanned Ground Vehicle) e veicoli aerei autonomi UAV(Unmanned Aerial Vehicle) che possono cooperare tra di loro. Gli ambienti esterni presentano diverse problematiche e una di queste può essere la mancanza di accesso alla rete GPS. Per tale motivo si studiano leggi di controllo che non impiegano questo tipo di informazioni come *bearing-only formation control* dove gli agenti rilevano tra di loro solo angoli relativi, i quali possono essere calcolati tramite una semplice camera 2D.

Questo lavoro di tesi ha lo scopo di esplorare la teoria della bearing rigidity applicata a formazioni di agenti omogenei nello specifico quadricotteri sfruttando la tecnologia NMPC con MATMPC. La tesi va ad esplorare le differenze delle diverse architetture in particolare va ad implementare una architettura centralizzata e mette giù le basi per un possibile sviluppo di un'architettura decentralizzata. Il lavoro mostra come è creare il modello per NMPC con diverse varianti del modello dell'agente (come è stato modellizzato), diversi funzionali di costo e diverse formazioni. Per l'implementazione viene usato ROS2 per lo sviluppo dell'intero ecosistema di controllo e viene usata la piattaforma PX4 che mette a disposizione un ottima architettura di controllo a basso livello per la gestione del singolo quadricottero. Inoltre la tesi usa il sistema SITL(Software In The Loop) di PX4 per rendere le simulazioni di Gazebo il più vicine alla realtà.

Keywords: bearing-only formation control, Nonlinear Model Predictive Control (NMPC), centralized architecture

Acknowledgements

This significant journey would not have been the same without my fellow classmates with whom I shared the many anxieties and joys of university life. But first and foremost, I want to wholeheartedly thank my supervisor Angelo Cenedese and co-supervisor Isabelle Fantoni for allowing me to develop this work with an Erasmus experience abroad in Professor Isabelle's LS2N laboratory in Nantes.

I also want to thank my long-time friends Angelo, Michele, and Giacomo with whom I've shared various adventures over the years and who have always been there when I needed them. Thanks also go to my girlfriend, who has supported me through challenging times and has always stood by my side.

All of this would never have been possible without the unwavering support of my family, who has always encouraged me in my academic journey, allowing me to pursue my passions and dreams.

This thesis, as well as my own self, is the result of many people and experiences that, for better or worse, have allowed me to become the person I am today. Today marks the end of both my master's degree and my university journey, a journey I embarked on six years ago that has taught me a lot, both professionally and personally.

Contents

List of Figures

List of Tables

Abbreviations

OCP	**O**ptimal **C**ontrol **P**roblem
NLP	**N**on **L**inear **P**rogramming
IP	**I**nterior **P**oint
QP	**Q**uadratic **P**rogramming
SQP	**S**equantial **Q**uadratic **P**rogramming
AS	**A**ctive **S**et
MAS	**M**ulty **A**gent **S**ystem
EKF	**E**xtended **K**alman **F**ilter
MPC	**M**odel **P**redictive **C**ontrol
NMPC	**N**onlinear **M**odel **P**redictive **C**ontrol
MOO	**M**ulti **O**bjective **O**ptimization
POF	**P**areto **O**ptimal **F**rontier
DMPC	**D**ecentralized **M**odel **P**redictive **C**ontrol
TMC	**T**rivial **M**otion **C**ontrol

Symbols

$\|\| \cdot \|\|_2$	Euclidean norm	-
$\mathbf{1}_n \in \mathbb{R}^n$	Unitary vector of N dimension	-
$A \otimes B$	Kronecker product	-
$\hat{b} = \frac{b}{\|\|b\|\|_2}$	Unit vector	-
${}^A_B\mathbf{R}$	Rotation matrix from frame A to frame B	-
$\mathbf{S} = [\cdot]_\times$	Skew matrix	-
$\mathbb{S}^d$	d-dimension manifold	-
$\mathcal{N}[A]$	Null space of matrix A	-
$Img[A]$	Image of matrix A	-
I_n	Identity matrix of n-dimension	-
$\mathcal{G}$	Graph	-
$rk[A]$	Rank of matrix A	-
$\dot{a}$	the derivative of first order of a	-
$a^{'}$	the derivative of first order of a	-

Part I

Theoretical approach

Chapter 1

Multi Agent System (MAS)

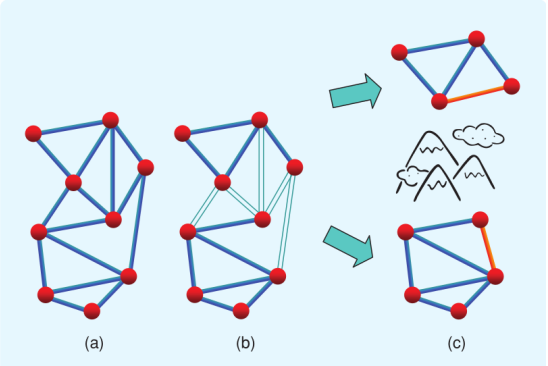
1.1 Introduction

Nature provides us with examples of collective behavior in certain animals, due to the fact that the individual is too weak to survive on his own, thus leading it to work in groups. Through group action, the individual is forced to achieve common goals such as the search for food (e.g. ants), the defense of territory (e.g. bees) or migrating from one place to another for the change of season (e.g. ants), for the change of season (e.g. birds)

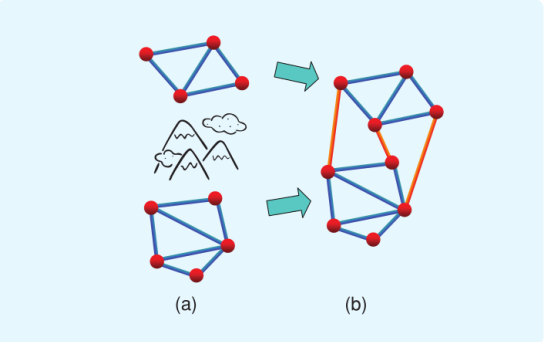
From these group behaviors, it was observed that certain types of animals or insects created specific formations to achieve their desired purpose. A well-known example is the behavior of birds to fly in flocks to reach new places safely.

Observing these interesting behaviors, over time researchers began to study and formulate theories so that they could be applied to autonomous aircraft's systems also called multi-agent autonomous systems (or multi-agent system MAS).

The study of creating and maintaining a formation brought interesting problems to be solved in order to achieve this. Firstly, an unambiguous way of representing the formation to be achieved was sought along with all the information of the various agents and the constraints between them. This led, for example, to the use of Graph Theory and Rigidity Theory, which made it possible to define a centralized/decentralized control. Once the formation has been achieved, arise problems such as motion planning, obstacle avoidance and then defining the optimal way to divide the formation and after joining pre-existing formations. In addition, there is also how to maximize the efficiency of exploration of a given area [6].



(a) Split formation to avoid obstacle



(b) Join formations after obstacle

FIGURE 1.1: Some examples of challenges with formation control. Source: [6]

A well-known behavior studied by researchers concerns animal groups moving together, known as flocking. This behavior is called emergent behavior and arises when there is an interaction of individual agents that adhering to a set of simple rules (which is called Reynolds rules).

FIGURE 1.2: Flocking of birds. Source: Wikipedia

1.2 Types of formation controller

We can categorize the formation control into four main labels:

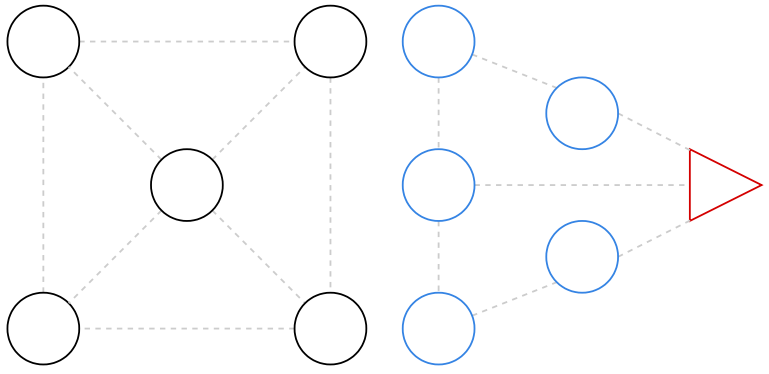
- **Position-based control**: Agents sense their own positions with respect to a global coordinate system

- **Displacement-based control**: Agents actively control the displacements of their neighboring agents. It works under the assumption that each agent is capable of sensing the relative positions of its neighboring agents in relation to the global coordinate system
- **Distance-based control**: Individual agents are assumed to have the ability to sense the relative positions of their neighboring agents in relation to their own local coordinate system
- **Bearing-based control**: Individual agents are assumed to have the capability to sense the relative bearing of their neighboring agents in relation to their own body frame

The controllers are divided into their sensing capabilities and interaction level. The Position-based control relies on strong assumptions that are challenging to realize in reality. Instead, getting off the controllers have weaker assumption than previous controllers and they are easier to apply in real-world scenarios.

The formation can be addressed through two main approaches:

- **Leader-follower**: the formation is created followed by a unique leader such as when the duck's formation flies together
- **Behavior-based** [**Leaderless**]: the formation is created when the agents reach a common idea, namely they use a consensus protocol



(a) Leaderless formation where all agents have the same weight in the decision

(b) Leader-follower formation where the red traingle is the leader instead the blue circles are followers

FIGURE 1.3: Leaderless vs leader-follow technique

1.3 Why the bearing formation control [1]

The MAS with rigidity theory has two main tasks:

- **Formation control**: the goal is to achieve a desired formation
- **Network localization**: the goal is to localize other agents based only on known information

These two tasks can be achieved by using two difference based information namely:

- **Distance-based control**: it uses the distance among agents to archive the tasks
- **Bearing-based control**: it uses the bearing information to reach the tasks

To obtain these in practice, each agent can use the GPS to obtain its absolute position. However, this method is not always viable due to GPS-denied environments, which typically include indoor or underwater locations, as well as deep space. Furthermore, GPS accuracy may not match high-accuracy requirements for applications like formation control.

For these motivations, it is more advantageous to use onboard sensors and local/relative information to do these tasks (that we cited above). Optical cameras are widely used as onboard sensors for ground and aerial vehicles to achieve various sensing tasks due to their low-cost, lightweight and low-power characteristics. The optical camera, in general, is a bearing-only sensor but it can measure distance thanks to stereo-vision. However it is more expensive than a standard camera.

Once a target has been identified in an image, its bearing relative to the camera can be calculated immediately from its pixel coordinate using the pinhole camera model.

The above-mentioned motivation has acquired many interests among researchers, as this topic has various applications in three distinct areas:

- **Bearing-based network localization**: the nodes are classified as **anchors** (the nodes know their position) and **followers** (the nodes don't know their position). They are capable of measuring the bearings of their neighboring and share this among them. The objective is to locate the follower nodes using the bearing measurements, utilizing only bearing information and the anchors' absolute positions
- **Bearing-based formation control**: the agents are able to obtain the relative positions of their neighbors and the aim is to reach a desired formation
- **Bearing-only formation control**: the agents are able to sense the relative bearing of their neighbors and the objective is the same as Bearing-based formation control

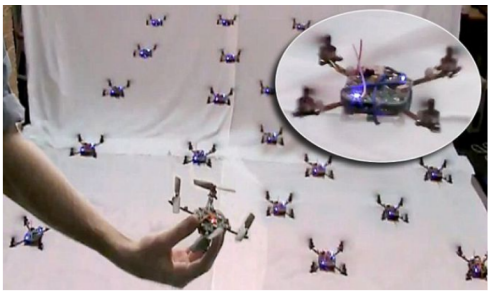


FIGURE 1.4: Formation control of a little swarm quadcopters

1.4 Goal of thesis

The aim of this thesis is to explore some aspects of the bearing rigidity theory applied to the formation task. The idea is to study this topic applied to different architectures including centralised, decentralised, and distributed and to integrate this theory with a new control technique called Model Predictive Control (MPC) or in our case Nonlinear Model Predictive Control (NMPC). We will discuss the theoretical and practical components of these architectures, utilizing this innovative control tool and considering its advantages, disadvantages, and restrictions.

In practice, we will develop an internal model system of NMPC to control a homogeneous triangular formation of three agents. In this specific case, we will work with three quadcopters to achieve a triangular formation like the desired formation. The whole system will be controlled by a unique NMPC that will work on an external power computer which will send the control input to PX4s which will manage the quadcopters (PX4 is a firmware onboard control of each agent).

After developing all the theories and models, we will test them on Simulink to understand how the NMPC with our model will behave in an ideal environment. Later on, we will move from an ideal environment to simulated environments through a Gazebo simulation where we will implement a control pipeline to handle all the experiments and it has being implemented with ROS2 framework.

1.5 Thesis structure

Briefly, in the following there is a little resume for each chapter:

- *Chapter 2*: we will report all theories that we utilized to develop our thesis. This includes an examination of modeling the quadrotor in various forms, the bearing rigidity theory, moving through MPC architectures, and until multi-objective optimization (MOO)

- *Chapter 3*: we will explain our design of the internal system model for NMPC with different aims, the trivial motion controller and finally we will discuss the evolution of cost functions
- *Chapter 4*: we will describe the tools utilized to implement the thesis and how we connected them together
- *Chapter 5*: we will present the findings that we obtained to test the different controllers (we described them in the previous chapter). In addition, the tests were been conducted before using MATLAB/Simulink and after using ROS 2/Gazebo
- *Chapter 6*: we will expose our final conclusions and future enhancements to further develop the thesis
- *Appendices*: there will be some insights into particular aspects

Chapter 2

Theory

In this chapter, we have reported the main theories that we have used to develop and use this thesis. We will start with how to model a quadrotor agent, then we will introduce some concepts of graph theory together with bearing rigidity theory and finally, we will explain the main concepts of MPC and NMPC.

2.1 Quadrotor model (Agent)

There are several ways to model a quadrotor agent. In our case, we started by using an agent model such as a point without mass because the idea was to test the algorithm with a simplified dynamics. We then derived the equations to describe the full dynamics of a quadrotor and we incorporated them into the final algorithm.

2.1.1 Quadrotor designed like a point without mass

We denoted the agent $\mathcal{A}$ and designed its dynamics as a point without mass and a single integrator, resulting in the following simplified kinematics [9]:

$$\begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \mathbf{R_z} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ w \end{bmatrix} \tag{2.1}$$

where:

- $\mathbf{p} \in \mathbb{R}^3$ position of $\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$ (World Frame)
- $\psi \in \mathbb{S}^1$ yaw rate of $\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$ (World Frame)
- $\mathbf{R_z}$ rotation frame on z-axis $\mathbf{R_z}(\psi)$ of $\mathcal{A}$
- $\mathbf{v} \in \mathbb{R}^3$ linear velocity in $\mathcal{F}_\mathcal{B}$ (Body frame)

- $w \in \mathbb{R}$ angular velocity in $\mathcal{F}_{\mathcal{B}}$ (Body frame)

Then its state is $\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \psi \end{bmatrix} \in \mathbb{R}^3 \times \mathbb{S}^1$. Lets notice that the model works in the $\mathbb{R}^3 \times \mathbb{S}^1$ space.

2.1.2 Quadrotor designed with the whole dynamic in SE(3)

We then designed the above model and proceeded to develop the quadrotor model with complete dynamics and kinematics, but we neglect **gyroscopic effect** and **inertial effect** due to the rotors and the flapping:

$$\textbf{Kinematics}: \begin{cases} \dot{\mathbf{p}}^{\mathcal{W}} &= \mathbf{v}^{\mathcal{W}} \\ \dot{\boldsymbol{\alpha}}^{\mathcal{W}} &= {}^{\mathcal{W}}_{\mathcal{B}}\mathbf{R}[\boldsymbol{\omega}^{\mathcal{B}}]_{\times} \end{cases} \tag{2.2}$$

$$\textbf{Dynamics}: \begin{cases} m\ddot{\mathbf{p}}^{\mathcal{W}} &= -mg\mathbf{e}_3 + {}^{\mathcal{W}}_{\mathcal{B}}\mathbf{R}\mathbf{F}\mathbf{u}^{\mathcal{B}} \\ \mathbf{J}\dot{\boldsymbol{\omega}}^{\mathcal{B}} &= -\boldsymbol{\omega}^{\mathcal{B}} \times \mathbf{J}\boldsymbol{\omega}^{\mathcal{B}} + \mathbf{M}\mathbf{u}^{\mathcal{B}} \end{cases} \tag{2.3}$$

where:

- m = quadrotor's mass
- $\mathbf{p}$ = quadrotor's position in $\mathcal{F}_{\mathcal{W}}$
- $\dot{\mathbf{p}}, \mathbf{v}$ = quadrotor's velocity in $\mathcal{F}_{\mathcal{W}}$
- $\ddot{\mathbf{p}}$ = quadrotor's acceleration in $\mathcal{F}_{\mathcal{W}}$
- $\boldsymbol{\omega}$ = quadrotor's angular velocity in $\mathcal{F}_{\mathcal{B}}$
- ${}^{\mathcal{W}}_{\mathcal{B}}\mathbf{R}$ = rotation matrix from Body frame to World frame
- $\mathbf{F}, \mathbf{M}$ = physic matrices of quadrotor
- $\mathbf{J}$ = inertial matrix
- $\mathbf{u}$ = input vector of spinning rotors in $\mathcal{F}_{\mathcal{B}}$
- $\boldsymbol{\alpha} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$ = the vector of Roll (ϕ),Pitch(θ) and Yaw(ψ) is expressed in $\mathcal{F}_{\mathcal{W}}$

Now, let's decide to make the following assumptions:

- The quadrotor works with little angles
- The model has a diagonal inertial matrix, namely $\mathbf{J} = diag([J_x, J_y, J_z])$ and a similar value on the diagonal

and we considered the following input:

- $\mathbf{f}_c = \mathbf{F}\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$ in $\mathcal{F}_\mathcal{B}$
- $\boldsymbol{\tau}_c = \mathbf{M}\mathbf{u} = \begin{bmatrix} \tau_\phi \\ \tau_\omega \\ \tau_\psi \end{bmatrix}$ in $\mathcal{F}_\mathcal{B}$

Where T (in the vector force applied to the barycenter $\mathbf{f}_c$) represents the thrust force applied only to the z-axis and $\boldsymbol{\tau}_c$ is the torque vector applied to all axes.

So, based on the above assumptions, we can simplify the model presented in Equation 2.2, Equation 2.3:

$$\textbf{Kinematics:} \begin{cases} \dot{\mathbf{p}}^\mathcal{W} &= \mathbf{v}^\mathcal{W} \\ \dot{\boldsymbol{\alpha}}^\mathcal{W} &= \boldsymbol{\omega}^\mathcal{B} \end{cases} \tag{2.4}$$

$$\textbf{Dynamics:} \begin{cases} m\ddot{\mathbf{p}}^\mathcal{W} &= -mg\mathbf{e}_3 + {}^\mathcal{W}_\mathcal{B}\mathbf{R} \begin{bmatrix} 0 \\ 0 \\ T^\mathcal{B} \end{bmatrix} \\ \dot{\boldsymbol{\omega}}^\mathcal{B} &= \mathbf{J}^{-1} \begin{bmatrix} \tau_\phi^\mathcal{B} \\ \tau_\theta^\mathcal{B} \\ \tau_\psi^\mathcal{B} \end{bmatrix} \end{cases} \tag{2.5}$$

In the figure below, we can see a representation of a world frame and a body frame with its rotation matrix from world to body frame:

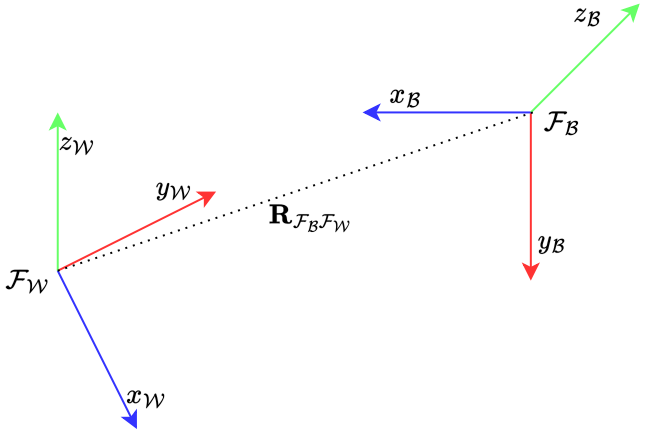


FIGURE 2.1: representation of world and body frame with its rotation matrix

Transformation from SE(3) to $\mathbb{R}^3 \times \mathbb{S}^1$ [10]

The quadrotor model evolves in SE(3). However, when working in $\mathbb{R}^3 \times \mathbb{S}^1$ a transformation from SE(3) to $\mathbb{R}^3 \times \mathbb{S}^1$ is required. Therefore, the following maps are introduced to reproject SE(3) manifold to $\mathbb{R}^3 \times \mathbb{S}^1$ using the ZYX Euler angle convention:

$$T_{SE} : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3 \times \mathbb{S}^1 \tag{2.6}$$

$$(\mathbf{p}^{SE(3)}, \boldsymbol{\alpha}) \mapsto \begin{bmatrix} \mathbf{p}^{\mathbb{R}^3 \times \mathbb{S}^1}(\mathbf{p}^{SE(3)}, \boldsymbol{\alpha}) \\ \psi^{\mathbb{R}^3 \times \mathbb{S}^1}(\boldsymbol{\alpha}) \end{bmatrix} \tag{2.7}$$

with the map is defined so:

$$\begin{array}{ll} \mathbf{p}^{\mathbb{R}^3 \times \mathbb{S}^1}(\mathbf{p}^{SE(3)}, \boldsymbol{\alpha}) : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3 \times \mathbb{S}, & \mathbf{p}^{\mathbb{R}^3 \times \mathbb{S}^1} = \mathbf{R}_y(\psi)\mathbf{R}_x(\theta)\mathbf{p}^{SE(3)} \\ \psi^{\mathbb{R}^3 \times \mathbb{S}^1}(\boldsymbol{\alpha}) : \mathbb{S}^2 \mapsto \mathbb{S} & \psi^{\mathbb{R}^3 \times \mathbb{S}^1} = \psi^{SE(3)} \end{array} \tag{2.8}$$

where

- $\mathbf{p}^{SE(3)}$ the position state in SE(3)
- $\mathbf{p}^{\mathbb{R}^3 \times \mathbb{S}^1}$ the position state in $\mathbb{R}^3 \times \mathbb{S}^1$
- $\boldsymbol{\alpha}$ contains ψ and θ which are Pitch and Roll using the ZYX Euler angle convention
- $\mathbf{R}_x$ or $\mathbf{R}_y$ are rotation matrix for specific axis

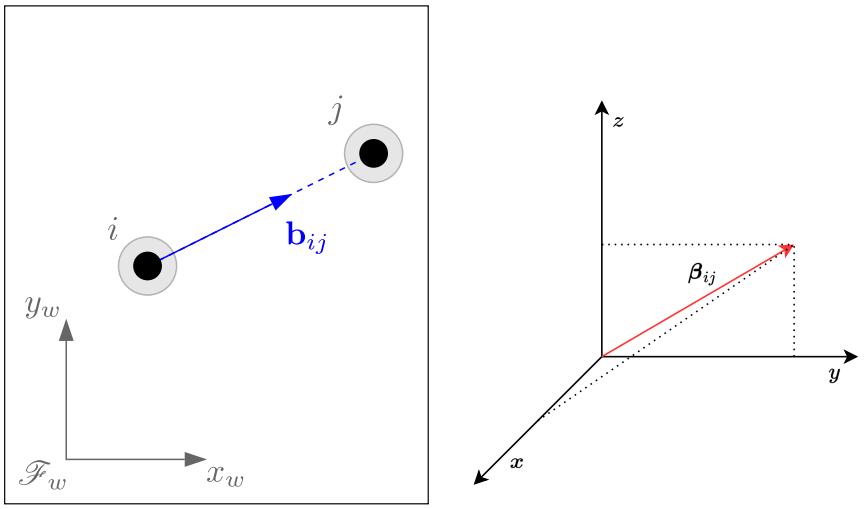
So, we can use these maps to move from the SE(3) to the $\mathbb{R}^3 \times \mathbb{S}^1$ manifold when required.

2.1.3 Bearing and quadrotor

In our case, we have decided that the agents can only sense one measure between them, which is the bearing among us (defined through edges of the formation graph). The *agent relative bearing* (from agent i to agent j and expressed in the body frame of agent i) is defined as the 3D unit vector:

$$\boldsymbol{\beta}_{ij} = {}^{\mathcal{W}}_{i}\mathbf{R} \frac{\mathbf{p}_j - \mathbf{p}_i}{||\mathbf{p}_j - \mathbf{p}_i||} \in \mathbb{S}^2 \tag{2.9}$$

where ${}^{\mathcal{W}}_{i}\mathbf{R}$ represents the rotation matrix from the world frame to the agent frame i (or the body frame) and $e_{ij} = \mathbf{p}_j - \mathbf{p}_i$ is the edge between two agents. The bearing vector represents a rotation in a sphere and its entries are the values of $cos(\cdot)$ on each axis.



(a) Representation of bearing information between two agents

(b) Representation of bearing information in 3D

FIGURE 2.2: Bearing representation

2.2 Graph theory

In mathematics, graph theory is the study of graphs which are mathematical structures that are employed to model the relations among objects. In this context, a graph consists of vertices representing the entities and the edges representing the connections or communications between entities.
The graph can be undirected where the edge doesn't have a direction and vice versa for directed graphs.

2.2.1 Basic concepts [2]

Definition 2.1 (Graph)**.** An *(undirected) graph* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of the nodes endowed with a state $\mathbf{x}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, i.e. connections between vertices.

Definition 2.2 (Oriented graph)**.** An *oriented graph* (also called directed graph or digraph) is a type of graph in which the edges have a tail and a head node, representing the starting and ending points, respectively.

Definition 2.3 (Node neighbors)**.** v_i, v_i are said to be *neighbors* if and only if $e_{ij} \in \mathcal{E}$ hence the set of neighbors of vertex i is denoted as $\mathcal{N}_i = j \in \mathcal{V} : (i, j) \in \mathcal{E}$.

Definition 2.4 (Node degree)**.** The number of neighbors of the node i is called the *degree of the node i.*

On the graph, we can find some properties which can be described using the matrix notation:

Definition 2.5 (Node degree matrix)**.** We define the *node degree matrix* of a (non-oriented) graph $\mathcal{G}$ as the diagonal matrix $\mathbf{\Delta}_{\mathcal{G}} \in \mathbb{R}^{n\times n}$ with entries the degree of the nodes. For oriented graphs, we define the diagonal matrices $\Delta_{\mathcal{G}_{IN}} \in \mathbb{R}^{n\times n}$ and $\mathbf{\Delta}_{\mathcal{G}_{OUT}} \in \mathbb{R}^{n\times n}$ whose elements are, respectively, the in-degrees and out-degrees of the nodes.

Definition 2.6 (Adjacency matrix)**.** The *adjacency matrix* $\mathbf{A}_{\mathcal{G}} \in \mathbb{R}^{n\times n}$ of an undirected graph without self-loops is defined as:

$$\mathbf{A}_{\mathcal{G}}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

whereas for a directed graph without self-loops, it is:

$$\mathbf{A}_{\mathcal{G}}(i,j) = \begin{cases} 1 & \text{if } (i \to j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Definition 2.7 (Incidence Matrix)**.** We define the *incidence matrix* $\mathbf{D}_{\mathcal{G}} \in \mathbb{R}^{n\times m}$ of an undirected graph as follows:

$$\mathbf{D}_{\mathcal{G}}(i,k) = \begin{cases} 1 & \text{if } v_i \in e_k \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

and, for directed graphs:

$$\mathbf{D}_{\mathcal{G}}(i,k) = \begin{cases} +1 & \text{if } v_i \text{ tail of} e_k \\ -1 & \text{if } v_i \text{ head of} e_k \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

2.2.2 Graph and Formation

Definition 2.8 (Network or formation)**.** A network, denoted as $(\mathcal{G}, \mathbf{x})$, is $\mathcal{G}$ with its vertex $i \in \mathcal{V}$ mapped to $\mathbf{x}_i$ (state of $\mathcal{A}_i$) namely $\mathbf{x}_i : \mathcal{V} \mapsto \mathcal{X}$. A network may be called like *formation* in the context of either formation control or network localization.

Let us consider an orientation of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the communication among agents is defined through this graph which has a vertexes set $\mathcal{V} = \{1, ..., N\}$ and an edges set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$.

The vertex $\mathcal{V}_i$ is an agent instead the edge $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ is the communication link between $\mathcal{A}_i$ and $\mathcal{A}_j$ where the $\mathcal{A}_i$ needs to measure the relative bearing $\boldsymbol{\beta}_{ij} = \frac{\mathbf{e}_{ij}}{||\mathbf{e}_{ij}||}$ to $\mathcal{A}_j$. Here, $|\mathcal{V}| = N$ denotes the number of agents and $|\mathcal{E}| = m$ denotes the number of edges.

2.3 Bearing rigidity theory

Bearing rigidity theory studies the conditions where the geometric pattern of a network can be uniquely determined if the bearing of each edge in the network is fixed. We can equivalently state that the bearing rigidity studies the conditions where two networks have the same geometric pattern if they have the same bearing [1]. It is shown in the below figure:

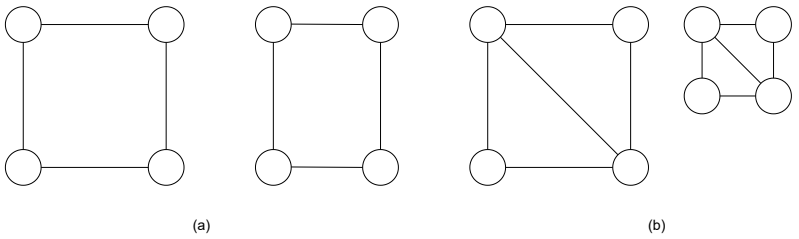


FIGURE 2.3: (a) **without** the rigidity property (b) **with** the rigidity property

In the rigidity theory, the structure may exhibit rigidity property or not. A structure is considered rigid if it can not change its shape with external forces or other perturbation. The bearing rigidity theory adds the requirement that the structure needs to have the same bearing information and the property of rigidity at the same moment.
In general, if a structure possesses the rigidity property then preserve the bearing information instead and vice versa not (this is depicted in above Figure 2.3).

There are three notions of bearing rigidity:

- **Bearing**
- **Global bearing**
- **Infinitesimal bearing**

that we will see their definition in the next section. However, the third is the most important and we will exploit this property to achieve the goal of creating a stable formation. Unlike of other two, they cannot ensure unique geometric patterns of the network.

2.3.1 Basics definitions [1]

Definition 2.9 (orthogonal projection matrix)**.** For any nonzero vector $\mathbf{x} \in \mathbb{R}^d$ $(d \geq 2)$, define an *orthogonal projection matrix* as

$$\mathbf{P}(\mathbf{x}) = \mathbf{I}_d - \frac{\mathbf{x}\mathbf{x}^T}{||\mathbf{x}||^2} \tag{2.14}$$

to denote the matrix we can use $\mathbf{P}(\mathbf{x}) = \mathbf{P}_\mathbf{x}$.

Definition 2.10 (Bearing equivalency)**.** Two networks $(\mathcal{G}, \mathcal{X})$ and $(\mathcal{G}, \mathcal{X}')$ are *bearing equivalent* if $\mathbf{P}_{(x_i - x_j)} \cdot (x'_i - x'_j) = 0$ for all $i, j \in \mathcal{V}$.

Definition 2.11 (Bearing congruency)**.** Two networks $(\mathcal{G}, \mathcal{X})$ and $(\mathcal{G}, \mathcal{X}')$ are *bearing congruent* if $\mathbf{P}_{(x_i - x_j)} \cdot (x'_i - x'_j) = 0$ for all $i, j \in \mathcal{V}$.

Definition 2.12 (Bearing rigidity)**.** A network $(\mathcal{G}, \mathcal{X})$ is *bearing rigid* if there exists a constant $\epsilon > 0$ such that any network $(\mathcal{G}, \mathcal{X}')$ that is bearing equivalent to $(\mathcal{G}, \mathcal{X})$ and satisfies $||\mathcal{X}' - \mathcal{X}|| < \epsilon$ is also bearing congruent to $(\mathcal{G}, \mathcal{X})$.

Definition 2.13 (Global bearing rigidity)**.** A network $(\mathcal{G}, \mathcal{X})$ is *globally bearing rigid* if an arbitrary network that is bearing equivalent to $(\mathcal{G}, \mathcal{X})$ is also bearing congruent to $(\mathcal{G}, \mathcal{X})$.

Let us consider an oriented graph where the intern neighbor bearings can be expressed by $\{\boldsymbol{\beta}_{e_i}\}_{i=1}^{m}$ and we then define the bearing function:

Definition 2.14 (Bearing function)**.** The bearing function is defined so

$$\boldsymbol{\beta}_{\mathcal{G}} : \mathcal{X} \to \mathcal{R}^{dm} \tag{2.15}$$

$$\boldsymbol{\beta}_{\mathcal{G}}(\mathcal{X}) = \begin{bmatrix} \boldsymbol{\beta}_{e_i} \\ ... \\ \boldsymbol{\beta}_{e_i} \end{bmatrix} \in \mathbb{R}^{dm} \tag{2.16}$$

so the bearing rigidity matrix can be defined as:

Definition 2.15 (Bearing rigidity matrix)**.** The *bearing rigidity matrix* is defined as the Jacobian of the bearing function

$$\mathcal{B}_{\mathcal{G}} = \frac{\partial \boldsymbol{\beta}_{\mathcal{G}}(\mathcal{X})}{\partial \mathcal{X}} \tag{2.17}$$

In the end, we introduce $\delta\mathcal{X}$ which is a variation of the configuration $\mathcal{X}$. If $\boldsymbol{\beta}_{\mathcal{G}}(\mathcal{X})\delta\mathcal{X} = 0$, then $\delta\mathcal{X}$ is an *infinitesimal bearing motion* of $(\mathcal{G}, \mathcal{X})$. An infinitesimal bearing motion is considered *trivial* if it corresponds only to a translation and a scaling of the entire network.

Definition 2.16 (Infinitesimally bearing rigid)**.** A network is *infinitesimally bearing rigid* if all the infinitesimally bearing motions are trivial

2.3.2 Bearing rigidity in $\mathbb{R}^3 \times \mathbb{S}^1$

Now, we take all the above definitions and rewrite them for the bearing formation in an $\mathbb{R}^3 \times \mathbb{S}^1$ environment.
We start to define the formation state $\mathbf{q} = (\mathbf{p}, \boldsymbol{\psi}) \in (\mathbb{R}^3 \times \mathbb{S}^1)^N$ that represents the configuration of N agents (Equation 2.1) with $\mathbf{q}_i = (\mathbf{p}_i, \psi_i) \in \mathbb{R}^3 \times \mathbb{S}^1$ being the configuration of the i-th agent in the group.

Definition 2.17 (Framework $\mathbb{R}^3 \times \mathbb{S}^1$ [9])**.** A framework (or formation) is the triple $(\mathcal{G}, \mathbf{p}, \boldsymbol{\psi})$ (or the pair $(\mathcal{G},\mathbf{q})$) where $\mathbf{p}_i : \mathcal{V} \to \mathbb{R}^3$ and $\psi_i : \mathcal{V} \to S^1$ map each vertex in $\mathcal{V}$ to a point $(\mathbf{p}_i, \psi_i) \in \mathbb{R}^3 \times \mathbb{S}^1$.

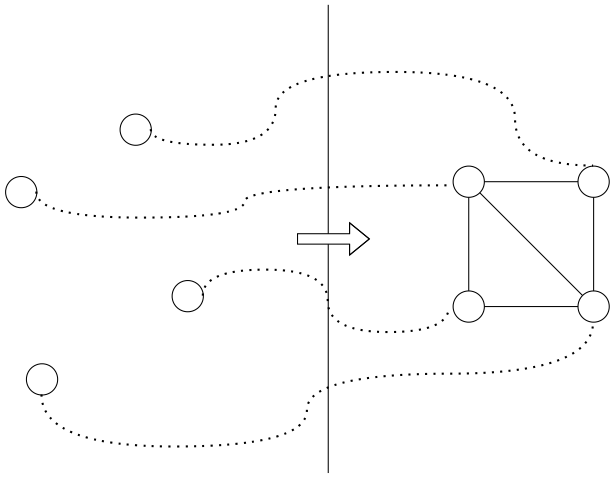


FIGURE 2.4: Evolution of a formation from random initial position

After defining the concept of the framework, we introduce two important elements to develop the bearing formation field as we have already seen in the previous section:

- Directed bearing rigidity function
- Directed bearing rigidity matrix

Definition 2.18 (Directed bearing rigidity function [9])**.** The *directed bearing function* associated to a framework $(\mathcal{G},\mathbf{q})$ is the map $\boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q}) : (\mathbb{R}^3 \times \mathbb{S}^1)^N \to (\mathbb{S}^2)^m$:

$$\boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q}) = [\boldsymbol{\beta}_{\mathbf{e}_1} \dots \boldsymbol{\beta}_{\mathbf{e}_m}] \tag{2.18}$$

where the notation $\mathbf{e}_i \in \mathcal{E}$ is used to represent a directed edge in the graph $\mathcal{G}$ according to any chosen to label.

Before defining the final tool, it is essential to recall the notions of rigidity and global rigidity in $\mathbb{R}^3 \times \mathbb{S}^1$:

Definition 2.19 (Rigidity and roto-flexible in $\mathbb{R}^3 \times \mathbb{S}^1$ [9])**.** The $\mathbb{R}^3 \times \mathbb{S}^1$ framework $(\mathcal{G},\mathbf{q})$ is *rigid* if there exists a neighborhood S of $\mathbf{q}$ such that

$$\boldsymbol{\beta}_{\mathcal{K}_{|\mathcal{V}|}}(\mathbf{q})^{-1} \cap S = \boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q})^{-1} \cap S \tag{2.19}$$

where $\boldsymbol{\beta}_{\mathcal{K}_{|\mathcal{V}|}}(\mathbf{q})^{-1} \subset \mathbb{R}^3 \times \mathbb{S}^1$ denotes the pre-image of the point $\boldsymbol{\beta}_{\mathcal{K}_{|\mathcal{V}|}}(\mathbf{q})$ under the directed bearing rigidity map.

The $\mathbb{R}^3 \times \mathbb{S}^1$ framework $(\mathcal{G}, \mathbf{q})$ is *roto-flexible* in $\mathbb{R}^3 \times \mathbb{S}^1$ if there exists an analytic path $\eta : [0, 1] \to (\mathbb{R}^3 \times \mathbb{S}^1)^N$ such that $\eta(0) = \mathbf{q}$ and

$$\eta(t) \in \boldsymbol{\beta}_{\mathcal{K}_{|\mathcal{V}|}}(\mathbf{q})^{-1} - \boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q})^{-1} \forall t \in (0, 1]. \tag{2.20}$$

Definition 2.20 (Bearing equivalent and Bearing congruent $\mathbb{R}^3 \times \mathbb{S}^1$ frameworks [9])**.** The Frameworks $(\mathcal{G}, \mathbf{p}, \boldsymbol{\psi})$ and $(\mathcal{G}, \mathbf{l}, \boldsymbol{\mu})$ are *bearing equivalent* if

$$\mathbf{R}_z(\psi_i)^T \mathbf{p}_{ij} = \mathbf{R}_z(\mu_i)^T \mathbf{l}_{ij} \tag{2.21}$$

for $\forall (i,j) \in \mathcal{E}$ and are *bearing congruent* if $\mathbf{R}_z(\psi_i)^T \mathbf{p}_{ij} = \mathbf{R}_z(\mu_i)^T \mathbf{l}_{ij}$ and $\mathbf{R}_z(\psi_j)^T \mathbf{p}_{ji} = \mathbf{R}_z(\mu_j)^T \mathbf{l}_{ji}$ for $\forall (i,j) \in \mathcal{V}$ with $i \neq j$.

Thanks to the above definitions, we can define the concept of global rigidity concept:

Definition 2.21 (Global Rigidity of $\mathbb{R}^3 \times \mathbb{S}^1$ Framework [9])**.** A framework $(\mathcal{G}, \mathbf{q})$ is **global rigid** in $\mathbb{R}^3 \times \mathbb{S}^1$ if every framework which is bearing equivalent to $(\mathcal{G}, \mathbf{q})$ is also bearing congruent to $(\mathcal{G}, \mathbf{q})$

The concept of infinitesimal rigidity is characterized by the kernel of the Jacobian matrix of the directed bearing rigidity function. Therefore, we present a new tool:

Definition 2.22 (Directed bearing rigidity matrix [9])**.** The world frame (directed) bearing rigidity matrix is the Jacobian of the bearing function w.r.t the agent configuration $\mathbf{q}$, that is, the matrix:

$${}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}} = \frac{\partial \boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{3m \times 4N} \tag{2.22}$$

where its k-th row block associated to edge $e_k = (i,j)$ is expressed so:

$$\begin{bmatrix} -\mathbf{0}- & \underbrace{-\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}}_{i} & -\mathbf{0}- & \underbrace{\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}}_{j} & -\mathbf{0}- & -\mathbf{S}\boldsymbol{\beta}_{ij} & -\mathbf{0}- \end{bmatrix} \tag{2.23}$$

with

- $d_{ij} = ||\mathbf{p}_i - \mathbf{p}_j||$ is the distance between two agents
- $\mathbf{P}_{ij} = \mathbf{I}_3 - \boldsymbol{\beta}_{ij}\boldsymbol{\beta}_{ij}^T$ is the orthogonal projector matrix onto the orthogonal complement of $\boldsymbol{\beta}_{ij}$
- $\mathbf{S} = [[0 \quad 0 \quad 1]^T]_\times$ skew symmetric matrix
- ${}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}$ from $\mathcal{F}_{\mathcal{W}}$ to $\mathcal{F}_{\mathcal{B}}$

Therefore, by defining the directed bearing rigidity matrix, we can define the infinitesimal bearing rigidity and its property through bearing rigidity matrix:

Definition 2.23 (Infinitesimal bearing rigidity in $\mathbb{R}^3 \times \mathbb{S}^1$ [9])**.** A $\mathbb{R}^3 \times \mathbb{S}^1$ framework $(\mathcal{G}, \mathbf{q})$ is *infinitesimally bearing rigid* if

$$Null[{}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}] = Null[{}^{\mathcal{W}}\mathcal{B}_{\mathcal{K}_N}]$$

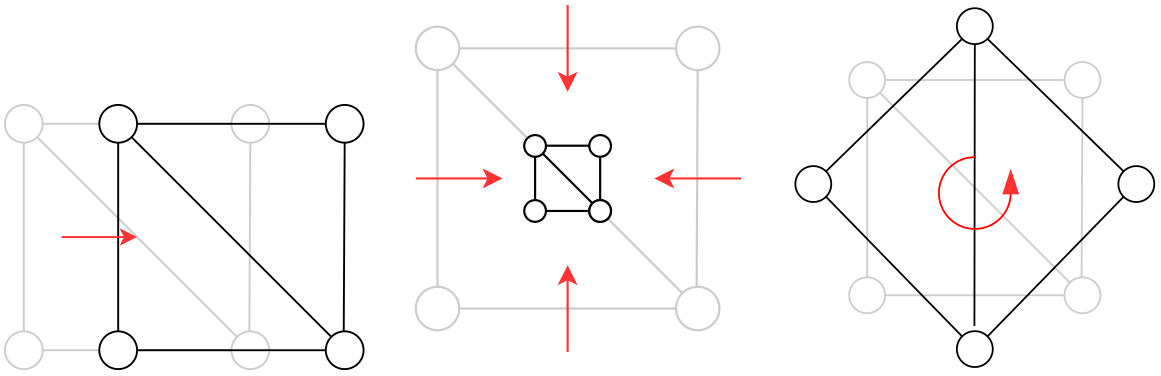
with $\mathcal{K}_N$ being the complete directed graph.

Theorem 2.24 (Property of bearing rigidity matrix [9]). *A $\mathbb{R}^3 \times \mathbb{S}^1$ framework is infinitesimally bearing rigid if and only if $rk[{}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}] = 3|\mathcal{V}| - 4$.*

Thanks for this definition, for an infinitesimally rigid framework in $\mathbb{R}^3 \times \mathbb{S}^1$, the 5-dimensional null space of the bearing rigidity matrix includes three actions corresponding to:

- ${}^{\mathcal{W}}\mathbf{v}_{\mathcal{F}}$ - a vector of three rigid-body translations along each axis
- $\dot{s}_{\mathcal{F}}$ - a velocity of dilatation relative to a reference point
- $\dot{\psi}_{\mathcal{F}}$ - an angular rotation around a vertical axis that passing through a reference point

These three actions allow us to manage the formation without breaking it during the flight. For this reason, they are referred to as **trivial motions** and its set is denoted with the following symbol $\boldsymbol{\mathcal{M}} = \{{}^{\mathcal{W}}\mathbf{v}_{\mathcal{F}}, \dot{s}_{\mathcal{F}}, \dot{\psi}_{\mathcal{F}}\}$.



(a) Translation (b) Scaling (c) Rotation

FIGURE 2.5: Trivial motion. Source: [6]

Furthermore, the bearing rigidity matrix relates the changes of the bearing function $\boldsymbol{\beta}_{\mathcal{G}}$(expressed in $\mathcal{F}_{\mathcal{B}}$) to velocities in the world frame $\dot{\mathbf{q}} = (\dot{\mathbf{p}}, \dot{\boldsymbol{\psi}})$:

$$\dot{\boldsymbol{\beta}}_{\mathcal{G}} = {}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\boldsymbol{\psi}} \end{bmatrix} \tag{2.24}$$

2.4 Model Predictive Control (MPC) and Nonlinear Model Predictive Control (NMPC)

In this section, we will explain the main concept of MPC, LQR, NMPC, and the architecture of MPC.

2.4.1 Model Predictive Control (MPC)

Model predictive control (MPC) is an advanced method of process control that is used to control a process while satisfying a set of constraints. It relies on dynamic models of the process that are used to predict future behavior of that model in a time slot called *prediction horizon* and the idea is to optimize the control input to minimize a cost function with an optimal control input. The MPC has the advantage that predicting the future step of the process by the model that it has and it finds a set of optimal control inputs (it is as long as the number of prediction horizons).

MPC is based on iterative, finite-horizon optimization of a plant model. At the time t the current plant state is sampled and a cost-minimizing control strategy is computed, through numerical algorithms, for a short time horizon in the future $[t, t+T]$ called prediction horizon. Only the first step of the control strategy is implemented, then the plan state is sampled again and the calculations are repeated starting from the new current state, yielding a new control and a new predicted state path (this is depicted in the below figure).

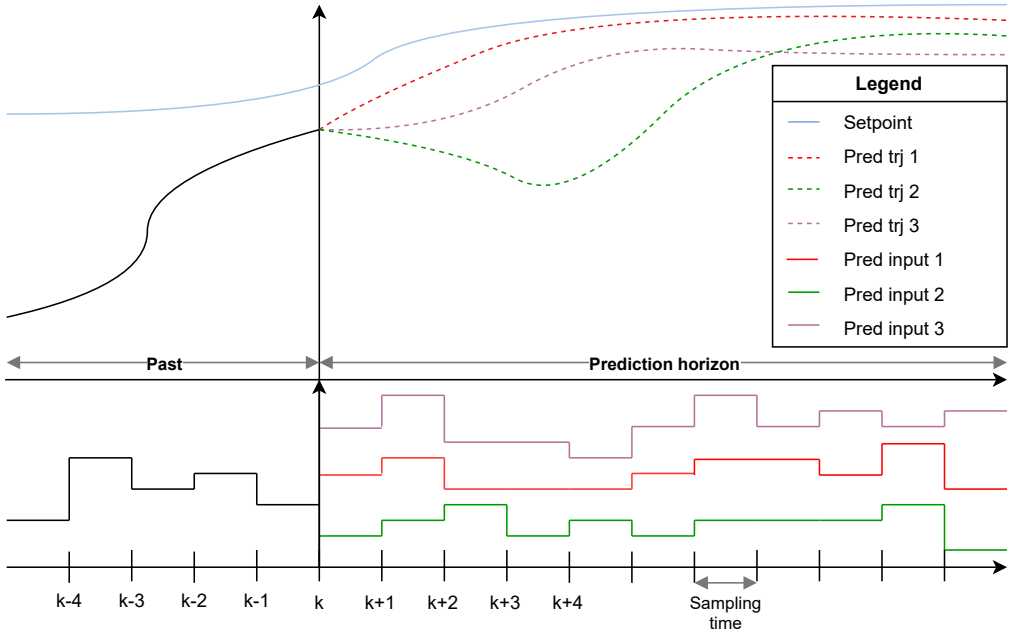


FIGURE 2.6: MPC prediction horizon

The main ingredients for MPC are:

- Internal system model
- Cost function
- Initial conditions
- Set of constraints on input and system state

The more basic cost function is defined in the following way:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}^d, \mathbf{u}, \Delta t) = \frac{1}{2}\mathbf{e_x}(\mathbf{x}(k + N_p\Delta t))^T\mathbf{S}\mathbf{e_x}(\mathbf{x}(k + N_p\Delta t)) + \tag{2.25}$$

$$\frac{1}{2}\sum_{l=1}^{N_p-1}[\mathbf{e_x}(\mathbf{x}(k + l\Delta t))^T\mathbf{Q}\mathbf{e_x}(\mathbf{x}(k + l\Delta t)) + \tag{2.26}$$

$$\mathbf{u}(k + l\Delta t)^T\mathbf{R}\mathbf{u}(k + l\Delta t)] \tag{2.27}$$

where

- $\mathbf{Q} \geq \mathbf{0}$ is the weight matrix on state error
- $\mathbf{R} > \mathbf{0}$ is the weight matrix on control input
- $\mathbf{S} \geq \mathbf{0}$ is the weight matrix on the final term
- k is the discrete-time
- Δt is the sampling time
- N_p is the length of the prediction horizon
- $\mathbf{x}(k\Delta t), \mathbf{u}(k\Delta t)$ state and input at k-th time
- $\mathbf{x}^d$ is the desired plant state

and the error is defined so:

$$\mathbf{e_x}(\mathbf{x}(k + l\Delta t)) = \mathbf{x}(k + l\Delta t) - \mathbf{x}^d$$

2.4.2 MPC vs Linear Quadratic Regulator (LQR)

The Linear Quadratic Regulator (LQR) is the most famous optimal controller. Its goal is to minimize the cost function through the algebraic Riccati equation that it takes into account the plant model. The LQR has two approaches:

- Infinite horizon
- Finite horizon

The MPC works with fixed length prediction, often weighted sets of error functions. The LQR looks at all linear system inputs and provides the transfer function that will reduce the total error across the frequency spectrum, trading off state error against input frequency.

For this reason, LQR has better global stability properties, but MPC often has more locally optimal and complex performance.

The main differences between MPC and LQR are:

- **LQR**
  - it optimizes the control input across the entire time window and it uses this optimal control input in the whole time window
  - is not able to handle constraints on system state and control input
- **MPC**
  - is an implementation of the Receding Horizon (RH) principle where it optimizes the control input for a finite control input but it applies only to the first optimal control input and repeats the process for the next step
  - can handle hard constraints on state and control input

2.4.3 Nonlinear Model Predictive Control (NMPC)

The NMPC (Nonlinear MPC) is the extension of MPC that is able to handle to solve an optimization problem with a nonlinear system. The NMPC is divided into 5 main phases [7]:

- Define the problem in continuous time, namely Optimal Control Problem (OCP)
- Discretization of optimization problem by integration
- Formalize the nonlinear problem with Karush-Kuhn-Tucker (KKT) conditions and Newton methods
- Compute derivative through sensitivity
- Solve the Quadratic Programming (QP) with one of several methods

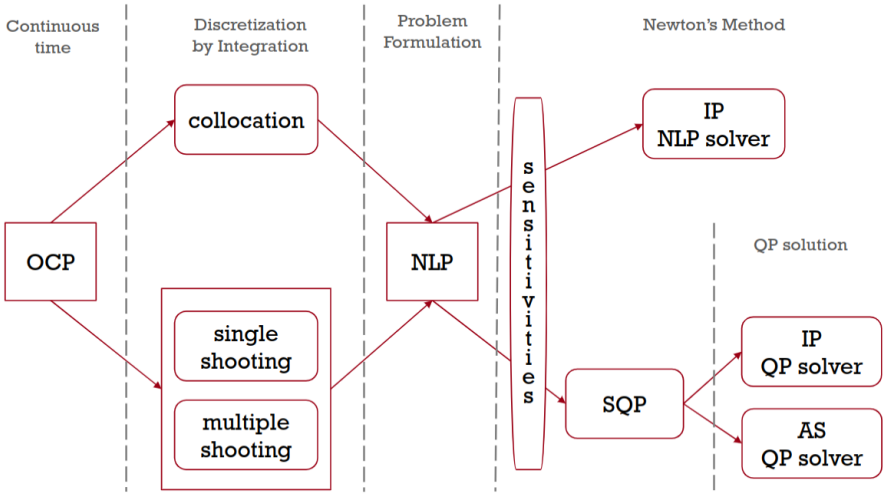


FIGURE 2.7: NMPC algorithm. Source: Slide course AeMPC [7]

The idea is to resolve the following problem:

$$u^* = \arg\min_{u(\cdot)} L(x(t), u(t))$$
$$\text{s.t} \begin{cases} x(t_0) = \hat{x}_0 \\ \dot{x}(t) = F(x(t), u(t)) \quad t \in [t_0, t_f] \\ H(x(t), u(t)) \leq 0 \quad t \in [t_0, t_f] \end{cases} \tag{2.28}$$

where the system $F(\cdot, \cdot)$ is a nonlinear system (NMPC internal system model) and the problem can have a constraint set $H(\cdot, \cdot)$ with non-linear constraints. Which ones are defined for a finite horizon that starts from t_0 (initial time) to t_f (final time).

2.4.4 Architectures with the MAS

In this section, we present the three architectures of NMPC when it is used with the MAS. We will introduce the main abstract concept for each one and in the end, we will compare them with advantages and disadvantages.
We will see the following architectures:

- **Centralized**
- **Distributed**
- **Decentralized**

2.4.4.1 Centralized architecture

The first architecture that we see is the centralized. The main concept of this architecture is that there is a single MPC that computes the set of optimal control inputs for all the agents and once it has computed them, it sends each control input to the specific agent. The state of the internal system model is composed of each agent state, for example, if we have a MAS of 5 agents and each agent state is d dimensions then the state of the internal system is $5d$ dimensions.
In most cases, the matrices that describe the MAS, in this type of architecture, are very large and the NMPC needs a lot of power to optimize them.

In this architecture, it is important to synchronize the communication between the agents and the MPC, so that the MPC is able to predict the best control input at the right time. In most cases, in this architecture, the MPC works on an external powerful computer that is able to receive the information, compute the optimizations, and send the optimal control input in real time.
The following figure shows the centralized architecture:

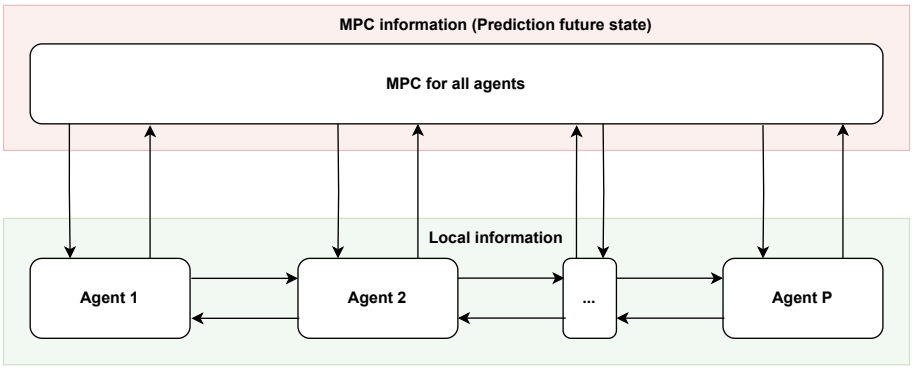


FIGURE 2.8: Centralized MPC technique

2.4.4.2 Distributed architecture

In a distributed architecture, the centralised MPC problem is divided into sub-problems, where each agent only needs to compute its sub-problem in parallel. Each MPC solves a sub-problem by using local information and the information obtained from the interactions between the systems, where the controller can communicate its optimal control input and the predicted states to other controllers (it is shown in Figure 2.9).
In this way, it is no longer necessary to have a dedicated PC with a large computing power to achieve a feasible solution as with a centralised problem.

Therefore, each agent (its MPC) shares the following information with other agents

- **Output**:
  - its local state information
  - its prediction states
- **Input**:
  - all local state information of other agents (only those linked to it)
  - all predicted future states of other agents (only those linked to it)

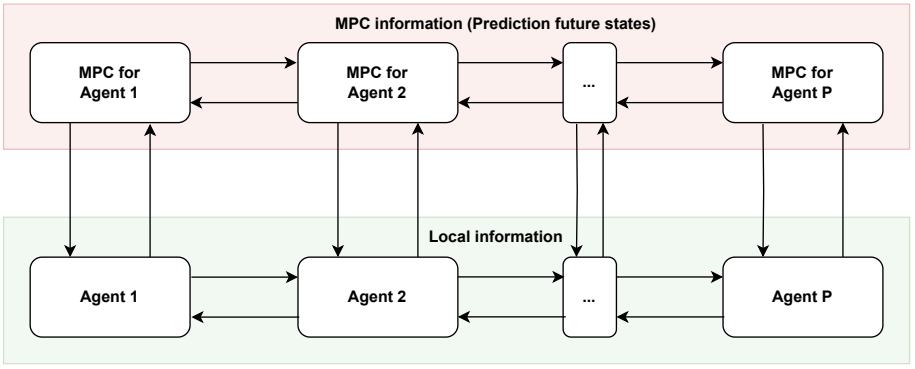


FIGURE 2.9: Distributed MPC technique [8]

Thanks to this exchange of the prediction horizon between the MPCs, the distributed architecture doesn't need to compute a centralized feasible solution for the first computation step.
For the above reason, the MPC is placed onboard of the quadrotor hardware and it doesn't need an external PC.

The paper [8] proposes a possible implementation of this strategy.

2.4.4.3 Decentralized technique

The idea of decentralised MPC for the formation of P agents is to divide the MPC problem into sub-problems, but unlike the distributed technique, there is no exchange of future states between the agents. It is illustrated in the Figure 2.10:

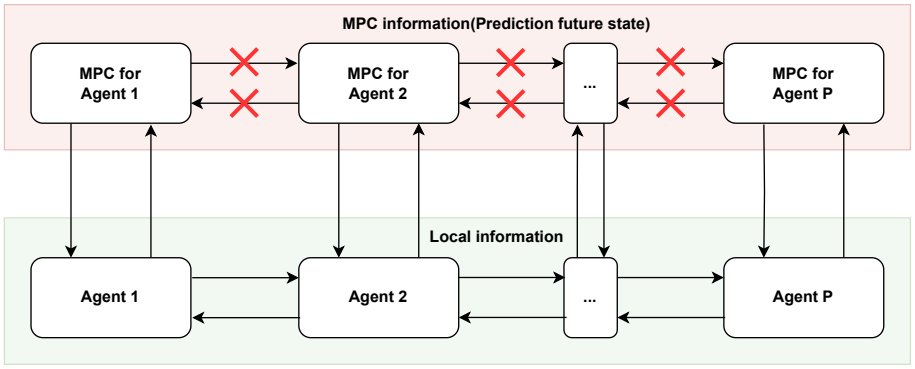


FIGURE 2.10: Decentralized MPC technique [8]

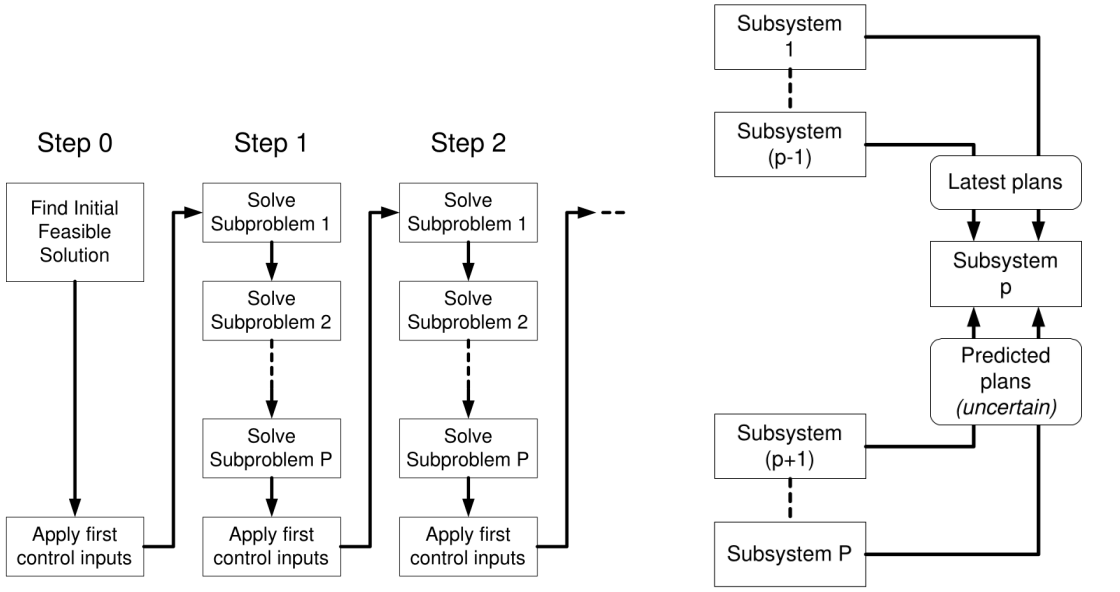
Therefore, each agent (its MPC) shares the following information with other agents

- **Output**:
  - its local state information
- **Input**:
  - all local state information from other agents (only those linked to it)

The key feature of the DMPC (Decentralized MPC) algorithm is that each agent solves only one sub-problem for its own plan, and each of these sub-problems is solved only once per time step, without iteration among the MPCs (this is shown in Figure 2.11(a)).

The Figure 2.11(b) shows the required information for subproblems, and we notice that in the first part, it gets the latest plans of the previous subsystems in order. Instead, in the second part, it uses the predicted plan solved in the previous step (we can imagine that it is working in the k-step, so it will use the predicted plane in the k-1-step).

The figures below illustrate the concept of breaking down the problem into subproblems, as explained earlier:



(a) DMPC procedure flow

(b) DMPC Information flow of a single step

FIGURE 2.11: DMPC algorithm

In the following, we want to present the concept of an algorithm that is proposed in the paper [11] to implement a decentralized architecture for MPC

Algorithm 1 Decentralized MPC

1: Find a solution to the initial *centralized* problem.
2: **if** Solution is not feasible **then**
3: stop (problem is infeasible)
4: **end if**
5: Set $k = 0$
6: Apply control $\mathbf{u}_p^*(k|k)$ of each subsystem p and k++
7: **for** $\forall$ subsystem p $\in p = 1, .., P$ **do**
8: Compile the plan data from other subsystems $\forall q \neq p$
9: Solve the sub-problem p
10: **end for**

The paper [11] says an important thing about the first step of the Algorithm 1 namely it is **not necessary** that the solution must be **an optimal solution**. Therefore, we can use also other methods to find a feasible solution.

One possible solution is to start with an initial condition (input and state) that produces a feasible solution for the first step of the decentralised algorithm. This way we can avoid the 0-step where it has to compute a feasible solution for the **centralised** problem.

The paper [12] proposes a decentralised and parallelized algorithm to compute a near-optimal solution to a specific class of non-linear, non-convex problems under the assumption of no delay or loss of communication.

2.4.4.4 Comparison between the three

Centralized MPC:

- *Advantages*:
  - Design a huge matrix without making new assumptions about the agents
- *Disadvantages*:
  - It needs many resources to compute the optimization

Distributed MPC:

- *Advantages*:
  - The algorithm doesn't have to compute a feasible solution for a centralized model
  - It uses medium/small matrices to describe the internal system model
- *Disadvantages*:
  - Need to exchange large quantities of data from agents (in the paper [8] choose to work with **one step ahead prediction**)
  - Likely to find a problem with delay in data exchange (high usage of network resources)
  - Share current states and prediction states
  - Store prediction states (implementation side)

Decentralized MPC:

- *Advantages*:
  - The agents exchange a small quantity of data with each other
  - Likely to find a problem with delay in data exchange (low usage of network resources)
  - Share only current states
  - It uses medium/little matrices to describe the internal system model
- *Disadvantages*:
  - the algorithm proposed by [11] has to compute a feasible solution for the centralized model (it can be avoided with special assumptions)

2.5 Multi-Objective Optimization (MOO)

The multi-objective optimization is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multi-objective is a type of vector optimization where optimal decisions need to be taken in the presence of trade-offs between two o more conflicting objectives. For example, we think about a car where we want to minimize the production cost while maximizing comfort and at the same time maximizing performance while minimizing fuel consumption. It is a hard challenge to satisfy all these objectives and we need to find a balance among them.

We can define the MOO problem using $\{f_i(x)\}_{i=1}^{n}$ optimization functions [4]:

$$\min / \max f_1(x), \ldots, f_n(x) \tag{2.29}$$

$$\text{subject to: } g_j(x),\ j = 1, \ldots, m\ x \in U \tag{2.30}$$

where x is the decision variable, x^* is a solution (also called *Pareto optimal*), g_j is a constraint of the problem and U is the feasible set of the MOO problem.

We can represent the MOO problem with all feasible solutions using through the Pareto method which we will explain in the next section.

2.5.1 Pareto frontier [3]

The *Pareto frontier* allows us to restrict attention to the set of efficient choices, and to make tradeoffs within this set, rather than considering the full range of every parameter. The MOO problem described in the Equation 2.29, its set of solutions is called *Pareto front* which shows the best trade-offs relations between all objectives.

The main objective of MOO is to find a collection of acceptable optimal solutions known as the *Pareto optimal set* or also *Pareto efficiency* and selects the best answer from the obtained *Pareto set*

The Pareto efficient refers to a situation where there is no action that makes one objective better off without making another worse off. There are three Pareto situations that are closely related:

- **Pareto improvement** is a new situation where some objective will gain and no objective will lose
- **Pareto-dominated** is a situation where there exists a possible Pareto Improvement or when one objective function cannot increase without reducing the other objective functions
- **Pareto-efficient/Non dominated** is a situation where there isn't change that could lead to improved satisfaction for some objective without some other objective losing. It is equivalent that there isn't scope for further Pareto improvement.

There are two other important concepts in the Pareto method which are:

- **Anchor Point** can be obtained through the best of an objective function
- **Utopia Point** is obtained through the intersection of the maximum/minimum value of an objective function and the maximum/minimum value of another objective function.

There is an example with two cost functions $f_1(x)$ and $f_2(x)$, in the figure below :

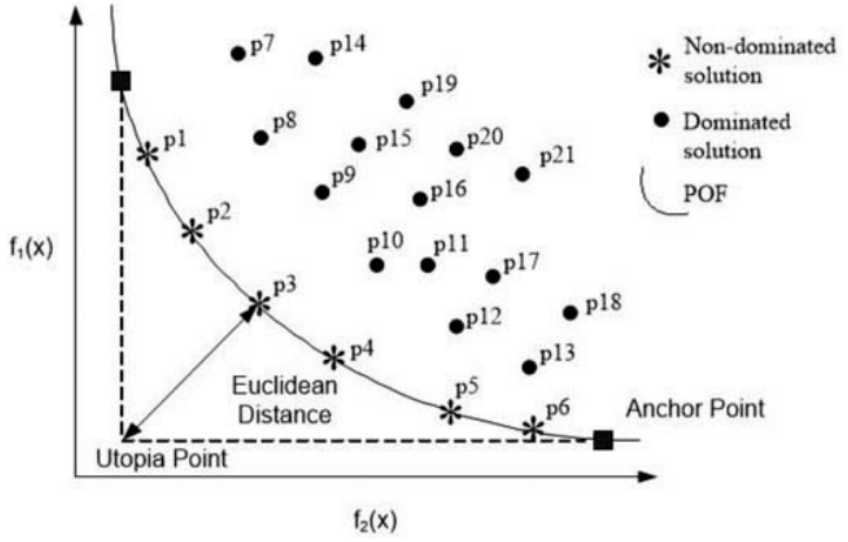


FIGURE 2.12: POF of minimization for two objective functions. Source: [3]

Therefore, once the solution points and the utopia point have been found, the POF's the optimal value can be obtained by calculating the shortest Euclidean distance from the Utopia point.

There are several ways to solve multi-objective optimization problem. In general, the classical methods can be categorized into three approaches:

- Weighted method
- Goal Programming
- ϵ-Constraint method

2.5.2 Weighted method or scalarization method [4]

The weighted method is based on all objective functions being combined together and converting them into a single objective function using different coefficients or each function

as a weight factor. The weights are determined before the optimization process and indicate which task is more important than another.

Through this method, we can rewrite the problem (Equation 2.29) as follows:

$$F(x) = w_1 f_1(x) + \cdots + w_n f_n(x) = \sum_{k=1}^{n} w_k f_k(x) \tag{2.31}$$

$$\text{with } w_k \in (0,1) \sum_{k=1}^{n} w_k = 1 \tag{2.32}$$

the weights w_k of the objective function determine the solution of the fitness function $F(x)$ and how the performance priority. The most difficult part of this method is finding the appropriate weights for all objectives based on the relative importance of each objective when there is insufficient information about the desired problem. The problem can be solved repeatedly with various values for weights coefficients then the decision makers select the proper solution regarding the importance of each objective function.

The advantages of this method are:

- Efficiency
- Simplicity to apply
- Finding the optimal solutions on the entire Pareto set.

Instead, the disadvantages are:

- Decision-makers need to determine the weights by their intuition/experience or judgment
- Changing the weights may cause big or small changes in the objectives
- The method can not find the appropriate solutions on the non-convex part of the Pareto set
- There is the possibility of having a number of minimum solutions for a particular set of weights in which generates various solutions in the Pareto set

In our case, we are lucky because we already know which task has priority over others.

2.5.3 ϵ-Constraint method [4]

The method is based on transforming a multi objective problem into a traditional single objective problem by choosing one objective as the main objective function and considering the remaining objective as constraints in the desired optimization problem.

We can rewrite the problem in the Equation 2.29 using this method:

$$\begin{aligned} &\min f_i(X); \quad X = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} \\ &\text{Subject to: } \left\{ \begin{array}{c} g_j(X) \leq 0 \quad j = 1, 2, ..., m \\ f_k(X) \leq \epsilon_k \quad k = 1, ..., n; \quad k \neq i \end{array} \right\} \\ &\text{where } \epsilon = \epsilon_1, \epsilon_2, ..., \epsilon_{i-1}, \epsilon_{i+1}, ..., \epsilon_n \end{aligned} \tag{2.33}$$

Therefore, the Pareto set is generated by solving the optimization problem repeatedly for different values of ϵ.

Chapter 3

Controllers

In this chapter, we will show the controllers that we used during the thesis. We will present the classic controller for bearing-only formation, which utilizes the gradient descent method. Then, we will present various internal system models for NMPC that implement distinct quadrotor dynamics, control inputs, and objectives.

3.1 Classical bearing-only control for multi-agent formation

As stated in the Section 2.3.2, the bearing rigidity matrix ${}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}(\mathbf{q})$ relates the changes of the bearing function $\boldsymbol{\beta}_{\mathcal{G}}$ (expressed in $\mathcal{F}_{\mathcal{B}}$) induced by the world-frame velocities $\dot{\mathbf{q}} = (\dot{\mathbf{p}}, \dot{\psi})$ in the framework (or formation):

$$\dot{\boldsymbol{\beta}}_{\mathcal{G}} = {}^{\mathcal{W}}\mathcal{B}_{\mathcal{G}}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\psi} \end{bmatrix} \tag{3.1}$$

Our goal is that a MAS can achieve the desired formation. Therefore, the desired formation can be expressed through a desired bearing information $\boldsymbol{\beta}^{d}_{\mathcal{G}}$. We utilize the bearing rigidity properties for our formation which already discussed in the Section 2.3.2 and the idea is to bring the bearing error the $\mathbf{e}(\mathbf{q}) = \boldsymbol{\beta}^{d}_{\mathcal{G}} - \boldsymbol{\beta}_{\mathcal{G}}(\mathbf{q})$ to zero. To accomplish this, we can formulate the problem as an optimization problem with a quadratic cost function $\mathcal{L}(\mathbf{q})$ regarding the bearing error:

$$\arg\min_{\mathbf{q}} \mathcal{L}(\mathbf{q}) = \arg\min_{\mathbf{q}} \frac{1}{2}||\mathbf{e}(\mathbf{q})||^2 \tag{3.2}$$

Therefore, we employ the gradient descent method to achieve this objective:

$$\nabla_{\mathbf{q}}(L(\mathbf{q})) = \nabla_{\mathbf{q}}(\frac{1}{2}||\mathbf{e}(\mathbf{q})||^2) \tag{3.3}$$

$$= \nabla_{\mathbf{q}}(\frac{1}{2}\mathbf{e}(\mathbf{q})^T\mathbf{e}(\mathbf{q})) \tag{3.4}$$

$$= \frac{1}{2}\dot{\mathbf{e}}(\mathbf{q})^T\mathbf{e}(\mathbf{q}) + \frac{1}{2}\mathbf{e}(\mathbf{q})^T\dot{\mathbf{e}}(\mathbf{q}) \tag{3.5}$$

$$= \dot{\mathbf{e}}(\mathbf{q})^T\mathbf{e}(\mathbf{q}) \tag{3.6}$$

$$= \begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{\psi}}\end{bmatrix}^T \mathcal{W}\mathcal{B}_{\mathcal{G}}(\mathbf{q})^T\mathbf{e}(\mathbf{q}) \tag{3.7}$$

$$= -\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{\psi}}\end{bmatrix}^T \mathcal{W}\mathcal{B}_{\mathcal{G}}(\mathbf{q})^T\boldsymbol{\beta}_{\mathcal{G}}^d \tag{3.8}$$

After obtaining this result, we replace the term $\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{\psi}}\end{bmatrix}$ with the Equation 2.1, so:

$$\nabla_{\mathbf{q}}(L(\mathbf{q})) = -\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{\psi}}\end{bmatrix}^T \mathcal{W}\mathcal{B}_{\mathcal{G}}(\mathbf{q})^T\boldsymbol{\beta}_{\mathcal{G}}^d \tag{3.9}$$

$$= -\mathbf{u}^T \begin{bmatrix} diag(\mathbf{R}_i^T) & \\ & \mathbf{I}_N \end{bmatrix} \mathcal{W}\mathcal{B}_{\mathcal{G}}(\mathbf{q})^T\boldsymbol{\beta}_{\mathcal{G}}^d \tag{3.10}$$

then we obtained the following control law [9]:

$$\mathbf{u} = \begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix} = k_c \begin{bmatrix} diag(d_{ij}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_N \end{bmatrix} \mathcal{B}_{\mathcal{G}}(\mathbf{q})^T\boldsymbol{\beta}_{\mathcal{G}}^d \tag{3.11}$$

We notice that we can tune the convergence speed though a proportional k_c. We will present the experiments of this controller in Chapter 5. In the next section, we will formulate the same problem but with an NMPC tool.

3.2 Model for centralized NMPC

In this section, two main internal system models for NMPC are presented:

- model with quadrotor like a singular integrator in a centralized architecture
- model with quadrotor with full complex dynamics

The second internal system model has four different designs with different objectives:

- centralized MAS model with 2 fixed agents and 1 unfixed agent

- centralized MAS with thrust and torque control (3 unfixed agents)
- centralized MAS with thrust and angular velocities control (3 unfixed agents)
- decentralized MAS with thrust and angular velocities control (3 unfixed agents)

For the sake of simplicity, we have opted to control a **triangular formation** with:

- 3 agents
- **Full connected graph**
- **Leaderless**
- Architecture:
  - Centralized
  - Decentralized
- the graph of sensing and communication is the same

Below the left side there is the representation of the triangular formation, instead the right side, there is an incidence matrix $\mathbf{D}_{\mathcal{G}}$ that described the relation among edges and vertices:

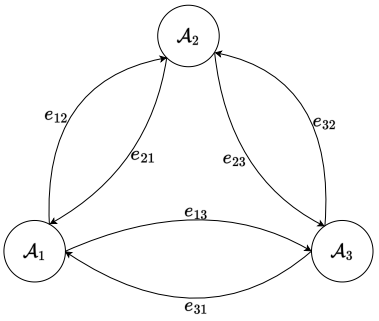


FIGURE 3.1: Triangle formation with full connected graph

$$\mathbf{D}_{\mathcal{G}} = \begin{bmatrix} 1 & 1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

For the sake of simplicity, all vectors in this study are dependent on time t but this is only explicitly indicated when necessary, for example, to highlight a time-variant matrix. In addition, we will note that all models will be non-linear and time-variant due to the inclusion of the bearing rigidity matrix and the quadrotor model.

3.2.1 Centralized approach with quadrotor modeled as a singular integrator

We start by defining the state and control input for this model. The objective of this model is to control a MAS comprising three agents as a single integrator with the bearing rigidity matrix.

The state of the NMPC internal system model needs the bearing information for all edges and the states of each agent. The control input that the NMPC has to optimize is composed of linear and angular velocity around the z-axis for each agent as outlined in our description in the Equation 2.1:

System state variables: System inputs:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \\ \mathbf{p}_1 \\ \psi_1 \\ \mathbf{p}_2 \\ \psi_2 \\ \mathbf{p}_3 \\ \psi_3 \end{bmatrix} \qquad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\beta}}_{12} \\ \dot{\boldsymbol{\beta}}_{13} \\ \dot{\boldsymbol{\beta}}_{21} \\ \dot{\boldsymbol{\beta}}_{23} \\ \dot{\boldsymbol{\beta}}_{31} \\ \dot{\boldsymbol{\beta}}_{32} \\ \dot{\mathbf{p}}_1 \\ \dot{\psi}_1 \\ \dot{\mathbf{p}}_2 \\ \dot{\psi}_2 \\ \dot{\mathbf{p}}_3 \\ \dot{\psi}_3 \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Then, we formulate the subsequent state equation for the entire system:

$$\dot{\mathbf{x}} = \mathbf{B}(\mathbf{x}, t)\mathbf{u} + \mathbf{F}\widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}}(t) = \begin{bmatrix} \mathbf{B}_1(\mathbf{x}, t) \\ \mathbf{B}_2(\mathbf{x}, t) \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}}(t) \tag{3.12}$$

where $\circ$ is the Hadamard product or element-wise product.

We start to describe the matrices $\mathbf{B}_1(\mathbf{x}, t)$ and $\mathbf{B}_2(\mathbf{x}, t)$. The first contains a part of **bearing rigidity matrix** information (Equation 2.23):

$$\mathbf{B}_1(\mathbf{x}, t) = \begin{bmatrix} -\frac{\mathbf{P}_{12}}{d_{12}} & \frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_2^1 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ -\frac{\mathbf{P}_{13}}{d_{13}} & \mathbf{0}_{3\times3} & \frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_3^1 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_1^2 & -\frac{\mathbf{P}_{21}}{d_{21}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & -\frac{\mathbf{P}_{23}}{d_{23}} & \frac{\mathbf{P}_{23}}{d_{23}}\mathbf{R}_3^2 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_1^3 & \mathbf{0}_{3\times3} & -\frac{\mathbf{P}_{31}}{d_{31}} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \frac{\mathbf{P}_{32}}{d_{32}}\mathbf{R}_2^3 & -\frac{\mathbf{P}_{32}}{d_{32}} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.13}$$

where the matrix $\mathbf{B}_1(\mathbf{x}, t) \in \mathbb{R}^{3m\times(3+1)N} = \mathbb{R}^{18\times12}$ and the terms are:

- $\mathbf{P}_{ij}(t) = \mathbf{I_3} - \boldsymbol{\beta}_{ij}(t)\boldsymbol{\beta}_{ij}(t)^T = \mathbf{P}_{ij}$ is the projection matrix, that is a time variant matrix
- $\mathbf{R}^i_j(t) = \mathbf{R}_z(\psi_i(t) - \psi_j(t)) = \mathbf{R}^i_j$ is the rotation matrix that is a time variant matrix

instead, the second matrix contains the **dynamics of agents** (Equation 2.1):

$$\mathbf{B}_2(\mathbf{x}, t) = \begin{bmatrix} \mathbf{R}_1 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 1 & 0 & 0 \\ \mathbf{0}_{3\times3} & \mathbf{R}_2 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 1 & 0 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{R}_3 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

where the matrix $\mathbf{B}_2(\mathbf{x}, t) \in \mathbb{R}^{3m\times(3+1)N} = \mathbb{R}^{12\times12}$. For the reasons above, the NMPC internal system model, that we design is time-variant and nonlinear.

Now, we will describe the second part of the Equation 3.12. The objective is to describe the $-S\boldsymbol{\beta}_{ij}w$ present in the bearing rigidity matrix (Equation 2.23) in matrix form. Therefore, we design the subsequent matrices $\mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix}$, $\widetilde{\boldsymbol{\beta}}$, $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{w}}$ to achieve this aim. We start from $\mathbf{F} \in \mathbb{R}^{30\times18}$ that contains $\mathbf{S} = [[0 \quad 0 \quad 1]^T]_\times$ and it is designed as follows:

$$\mathbf{F_1} = \begin{bmatrix} -\mathbf{S} & & & & & \\ & -\mathbf{S} & & & & \\ & & -\mathbf{S} & & & \\ & & & -\mathbf{S} & & \\ & & & & -\mathbf{S} & \\ & & & & & -\mathbf{S} \end{bmatrix} \in \mathbb{R}^{3m\times3m} = \mathbb{R}^{18\times18} \qquad \mathbf{F}_2 = \mathbf{0}_{12\times18} \tag{3.15}$$

The $\widetilde{\boldsymbol{\beta}}$ includes all relative bearing information for the MAS and the $\widetilde{\mathbf{w}}$ includes all angular velocities for the MAS:

$$\widetilde{\boldsymbol{\beta}} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \end{bmatrix} \in \mathbb{R}^{18\times1} \qquad \widetilde{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \in \mathbb{R}^{3\times1} \tag{3.16}$$

and we design the matrix $\widetilde{\mathbf{W}}$ so:

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0}_{12\times 3} \end{bmatrix} \in \mathbb{R}^{30\times 3} \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{1}_{3\times 1} & & \\ \mathbf{1}_{3\times 1} & & \\ & \mathbf{1}_{3\times 1} & \\ & \mathbf{1}_{3\times 1} & \\ & & \mathbf{1}_{3\times 1} \\ & & \mathbf{1}_{3\times 1} \end{bmatrix} \in \mathbb{R}^{18\times 3} \tag{3.17}$$

In that way, we are able to incorporate the latter half part of the bearing rigidity matrix (Equation 2.23) into our system:

$$\dot{\mathbf{x}} = \mathbf{B}(\mathbf{x}, t)\mathbf{u} + \mathbf{F}\widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}}(t) = \begin{bmatrix} \mathbf{B}_1(\mathbf{x}, t) \\ \mathbf{B}_2(\mathbf{x}, t) \end{bmatrix} \mathbf{u} + \underbrace{\begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}}(t)}_{-S\boldsymbol{\beta}_{\mathcal{G}}\mathbf{w}} \tag{3.18}$$

At this point, we include all elements in the NMPC internal system model which now can simulate a centralized formation with bearing rigidity. The results of simulations of this model are shown in the Chapter 5.

3.2.2 Model with quadrotor with full complex dynamics

In this section, we will use the complex quadrotor dynamics in our internal system model rather than an agent described as a point with mass because this allows us to have a better simulation of our quadcopter.

3.2.2.1 Centralized MAS model with 2 fixed agents and 1 unfixed agent

We start to define a state and control input for this model. Our objective for this model is to control a single agent that must establish a formation with two other agents (anchors) that are fixed in their position. The unfixed agent knows all information about the other two agents, including their position, velocities, and so on.

We begin by defining the state of our centralized MAS where the NMPC must only have the state of the unfixed agent because it can control only that. In addition, in the state, we add the bearing information of two edges with bidirectional orientation. This is necessary when implementing NMPC to achieve a triangular formation with a single unfixed agent and no distance constraints. In this case, a constraint on three angles must be imposed:

- Bearing information of agent 1: $\boldsymbol{\beta}_{12}$ and $\boldsymbol{\beta}_{13}$
- Bearing information of agent 2: $\boldsymbol{\beta}_{21}$

- Bearing information of agent 3: $\boldsymbol{\beta}_{31}$

Therefore, the state obtained by us is defined as follows:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{31} \\ \mathbf{p}_1 \\ \boldsymbol{\alpha}_1 \\ \mathbf{v}_1 \end{bmatrix} \in \mathbb{R}^{21\times 1} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\beta}}_{12} \\ \dot{\boldsymbol{\beta}}_{13} \\ \dot{\boldsymbol{\beta}}_{21} \\ \dot{\boldsymbol{\beta}}_{31} \\ \dot{\mathbf{p}}_1 \\ \dot{\boldsymbol{\alpha}}_1 \\ \dot{\mathbf{v}}_1 \end{bmatrix} \in \mathbb{R}^{21\times 1} \tag{3.19}$$

where the meanings of symbols are:

- $\boldsymbol{\beta}_{ij}$ is expressed in $\mathcal{F}_{\mathcal{B}}$
- $\mathbf{p}_i, \boldsymbol{\alpha}_i, \mathbf{v}_i$ are expressed in $\mathcal{F}_{\mathcal{W}}$

We assume that the position of the other two agents is **known** (with little variations) and constant for the duration of the prediction horizon. As a result, the control input is:

$$\mathbf{u} = \begin{bmatrix} T_1 \\ \boldsymbol{\omega}_1 \end{bmatrix} \in \mathbb{R}^{4\times 1} \text{ where } T_i \in \mathbb{R} \text{ and } \boldsymbol{\omega}_i \in \mathbb{R}^3 \text{ both in } \mathcal{F}_{\mathcal{B}} \tag{3.20}$$

and we define the known variables (specifically the linear and angular velocities of anchors) as follows:

$$\mathbf{l}_v = \begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} \in \mathbb{R}^{6\times 1} \tag{3.21}$$

The NMPC internal system model is defined as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x},t)\mathbf{x} + \mathbf{L}_v(\mathbf{x},t)\mathbf{l}_v + \mathbf{B}(\mathbf{x},t)\mathbf{u} + \widetilde{\mathbf{g}} + \mathbf{F}\widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \tag{3.22}$$

$$= \begin{bmatrix} \mathbf{A}_1(\mathbf{x},t) \\ \mathbf{A}_2 \end{bmatrix} \mathbf{x} + \mathbf{L}_v(\mathbf{x},t)\mathbf{l}_v + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2(\mathbf{x},t) \end{bmatrix} \begin{bmatrix} T_1 \\ \boldsymbol{\omega}_1 \end{bmatrix} + \widetilde{\mathbf{g}} + \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \boldsymbol{\beta} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \tag{3.23}$$

where $\circ$ is the Hadamard product or element-wise product. We distinguish between the number of unfixed drones and fixed drones as follows:

- N is the number of unfixed agents
- $\widetilde{N}$ is the number of fixed agents

we build the matrices $\mathbf{A}(\mathbf{x},t)$ and $\mathbf{L}_v(\mathbf{x},t)$ incorporating the bearing rigidity matrix Equation 2.23 ($m=4$):

$$\mathbf{A}_1(\mathbf{x},t) = \begin{bmatrix} \mathbf{0}_{3\times 18} & -\frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} \\ \mathbf{0}_{3\times 18} & -\frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} \\ \mathbf{0}_{3\times 18} & \frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} \\ \mathbf{0}_{3\times 18} & \frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} \end{bmatrix} \in \mathbb{R}^{3m\times(3m+9N)} = \mathbb{R}^{12\times 21} \tag{3.24}$$

$$\mathbf{L}_v(\mathbf{x},t) = \begin{bmatrix} \frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} \\ -\frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & -\frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} \\ \mathbf{0}_{9\times 3} & \mathbf{0}_{9\times 3} \end{bmatrix} \in \mathbb{R}^{(3m+9N)\times(3\widetilde{N})} = \mathbb{R}^{21\times 6} \tag{3.25}$$

with

- $\mathbf{P}_{ij}(t) = \mathbf{I_3} - \boldsymbol{\beta}_{ij}(t)\boldsymbol{\beta}_{ij}(t)^T = \mathbf{P}_{ij}$ is the projection matrix, that is a time variant matrix
- $\mathbf{R}_i(t) = \mathbf{R}_z(\psi_i(t)) = \mathbf{R}_i$ is the rotation matrix that is a time variant matrix

Instead, the quadrotor model is contained $\mathbf{A}_2$ and $\mathbf{B}_1$ (Equation 2.4-Equation 2.5):

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{0}_{3\times 12} & \mathbf{0}_{3\times 6} & \mathbf{I}_{3\times 3} \\ \mathbf{0}_{3\times 12} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 12} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \end{bmatrix} \in \mathbb{R}^{9N\times(3m+9N)} = \mathbb{R}^{9\times 21} \tag{3.26}$$

$$\mathbf{B}_1 = \mathbf{0}_{12\times 4} \qquad \mathbf{B}_2(\mathbf{x},t) = \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 1} & \mathbf{I}_{3\times 3} \\ \mathbf{R}_\mathbf{1}^{\mathbf{RPY}}\mathbf{e}_3\frac{1}{m} & \mathbf{0}_{3\times 3} \end{bmatrix} \in \mathbb{R}^{9\times 4} \tag{3.27}$$

Now we describe the latter half part of the Equation 2.23 with $\mathbf{F} \in \mathbb{R}^{21\times 12}$:

$$\mathbf{F}_2 = \mathbf{0}_{9\times 12} \qquad \mathbf{F_1} = \begin{bmatrix} -\mathbf{S} & & & \\ & -\mathbf{S} & & \\ & & -\mathbf{S} & \\ & & & -\mathbf{S} \end{bmatrix} \in \mathbb{R}^{3m\times 3m} = \mathbb{R}^{12\times 12} \tag{3.28}$$

we create a vector that includes all bearings information:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{31} \end{bmatrix} \in \mathbb{R}^{6\times 1} \qquad \widetilde{\mathbf{w}} = \begin{bmatrix} \omega_1^z \\ \omega_2^z \\ \omega_3^z \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{3.29}$$

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0}_{9\times 3} \end{bmatrix} \in \mathbb{R}^{21\times 3} \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{1}_{3\times 1} & & \\ \mathbf{1}_{3\times 1} & & \\ & \mathbf{1}_{3\times 1} & \\ & & \mathbf{1}_{3\times 1} \end{bmatrix} \in \mathbb{R}^{12\times 3} \tag{3.30}$$

In the end, the vector with gravitational effects on dynamics is defined as follows:

$$\widetilde{\mathbf{g}} = \begin{bmatrix} \mathbf{0}_{12\times 1} \\ \mathbf{0}_{6\times 1} \\ -g\mathbf{e}_3 \end{bmatrix} \tag{3.31}$$

The experiments done using this model are shown in the appendices.

3.2.2.2 Centralized MAS model with thrust and torque control

In this model, we aim to have complete control over all agents. As a result, we include all bearing information and the state of each agent within the state:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \\ \mathbf{p}_1 \\ \boldsymbol{\alpha}_1 \\ \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{p}_2 \\ \boldsymbol{\alpha}_2 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \\ \mathbf{p}_3 \\ \boldsymbol{\alpha}_3 \\ \mathbf{v}_3 \\ \boldsymbol{\omega}_3 \end{bmatrix} \in \mathbb{R}^{54\times 1} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\beta}}_{12} \\ \dot{\boldsymbol{\beta}}_{13} \\ \dot{\boldsymbol{\beta}}_{21} \\ \dot{\boldsymbol{\beta}}_{23} \\ \dot{\boldsymbol{\beta}}_{31} \\ \dot{\boldsymbol{\beta}}_{32} \\ \dot{\mathbf{p}}_1 \\ \dot{\boldsymbol{\alpha}}_1 \\ \dot{\mathbf{v}}_1 \\ \dot{\boldsymbol{\omega}}_1 \\ \dot{\mathbf{p}}_2 \\ \dot{\boldsymbol{\alpha}}_2 \\ \dot{\mathbf{v}}_2 \\ \dot{\boldsymbol{\omega}}_2 \\ \dot{\mathbf{p}}_3 \\ \dot{\boldsymbol{\alpha}}_3 \\ \dot{\mathbf{v}}_3 \\ \dot{\boldsymbol{\omega}}_3 \end{bmatrix} \in \mathbb{R}^{54\times 1} \tag{3.32}$$

where:

- $\boldsymbol{\beta}_{ij}$ is expressed in $\mathcal{F}_{\mathcal{B}}$
- $\mathbf{p}_i, \boldsymbol{\alpha}_i, \mathbf{v}_i$ are expressed in $\mathcal{F}_{\mathcal{W}}$
- $\boldsymbol{\omega}_i$ is expressed in $\mathcal{F}_{\mathcal{B}}$

In this case, it is necessary for the NMPC to optimize the thrust and torque of any agents; therefore, the control input is defined as follows:

$$\mathbf{u} = \begin{bmatrix} T_1 \\ \boldsymbol{\tau}_1 \\ T_2 \\ \boldsymbol{\tau}_2 \\ T_3 \\ \boldsymbol{\tau}_3 \end{bmatrix} \in \mathbb{R}^{12\times 1} \text{ where } T_i \in \mathbb{R} \text{ and } \boldsymbol{\tau}_i \in \mathbb{R}^3 \text{ both in } \mathcal{F}_{\mathcal{B}} \tag{3.33}$$

then the update equation is defined as follows:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}(\mathbf{x},t)\mathbf{x} + \mathbf{B}(\mathbf{x},t)\mathbf{u} + \widetilde{\mathbf{g}} + \mathbf{F}\widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \\ &= \begin{bmatrix} \mathbf{A}_1(\mathbf{x},t) \\ \mathbf{A}_2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2(\mathbf{x},t) \end{bmatrix} \begin{bmatrix} T_1 \\ \boldsymbol{\tau}_1 \\ T_2 \\ \boldsymbol{\tau}_2 \\ T_3 \\ \boldsymbol{\tau}_3 \end{bmatrix} + \widetilde{\mathbf{g}} + \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \boldsymbol{\beta} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \end{aligned} \tag{3.34}$$

The first matrix that we design is the $\mathbf{A}(\mathbf{x},t)$ which is then split into two separate matrices. The matrix $\mathbf{A}_1(\mathbf{x},t)$ contains the first portion of the Equation 2.23:

$$\mathbf{A}_1(\mathbf{x},t) = \begin{bmatrix} \mathbf{0}_{3\times 24} & -\frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 9} & \frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & -\frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 9} & \frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & \frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 9} & -\frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 9} & -\frac{\mathbf{P}_{23}}{d_{23}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 9} & \frac{\mathbf{P}_{23}}{d_{23}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & \frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 9} & -\frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 9} & \frac{\mathbf{P}_{32}}{d_{32}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 9} & -\frac{\mathbf{P}_{32}}{d_{32}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{3.35}$$

where $\mathbf{A}_1(\mathbf{x},t) \in \mathbb{R}^{3m\times(3m+12N)} = \mathbb{R}^{18\times 54}$ with

- $\mathbf{P}_{ij}(t) = \mathbf{I_3} - \boldsymbol{\beta}_{ij}(t)\boldsymbol{\beta}_{ij}(t)^T = \mathbf{P}_{ij}$ is the projection matrix, that is a time variant matrix
- $\mathbf{R}_i^z(t) = \mathbf{R}_z(\psi_i(t)) = \mathbf{R}_i^z$ is the rotation matrix that is a time variant matrix

Instead, the matrix $\mathbf{A}_2$ contains a part of quadrotor kinematics that we previously defined in Equation 2.4 (the blank spaces represent either zero scalars or matrices. The reason for this is to highlight the structure of matrices):

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{0}_{3\times18} & \mathbf{0}_{3\times6} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & & & & & \\ \mathbf{0}_{3\times18} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & & & & & \\ \mathbf{0}_{6\times18} & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & & & & & \\ \mathbf{0}_{3\times18} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & & & & & \\ \mathbf{0}_{3\times18} & & & & \mathbf{0}_{3\times6} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & & \\ \mathbf{0}_{3\times18} & & & & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & & \\ \mathbf{0}_{6\times18} & & & & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & & \\ \mathbf{0}_{3\times18} & & & & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & & \\ \mathbf{0}_{3\times18} & & & & & & & \mathbf{0}_{3\times6} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times18} & & & & & & & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{6\times18} & & & & & & & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} \\ \mathbf{0}_{3\times18} & & & & & & & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{3.36}$$

where the matrix $\mathbf{A}_2$ has dimension $\mathbb{R}^{12N\times(3m+12N)} = \mathbb{R}^{36\times54}$.

Now, it is time to design the matrix $\mathbf{B}(\mathbf{x}, t)$, which we will divide into two parts. The first part, $\mathbf{B}_1$ is empty because the control input doesn't update the bearing information control:

$$\mathbf{B}_1 = \mathbf{0}_{18\times12} \tag{3.37}$$

instead, in the matrix $\mathbf{B}_2(\mathbf{x}, t)$ contains the dynamics of all agents as expressed in Equation 2.5:

$$\mathbf{B}_2(\mathbf{x}, t) = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & & & \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & & & \\ \mathbf{R}_1^{\mathbf{RPY}}\mathbf{e}_3\frac{1}{m} & \mathbf{0}_{3\times3} & & & \\ \mathbf{0}_{3\times1} & \mathbf{J}^{-1} & & & \\ & & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \\ & & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \\ & & \mathbf{R}_2^{\mathbf{RPY}}\mathbf{e}_3\frac{1}{m} & \mathbf{0}_{3\times3} & \\ & & \mathbf{0}_{3\times1} & \mathbf{J}^{-1} & \\ & & & & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ & & & & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ & & & & \mathbf{R}_3^{\mathbf{RPY}}\mathbf{e}_3\frac{1}{m} & \mathbf{0}_{3\times3} \\ & & & & \mathbf{0}_{3\times1} & \mathbf{J}^{-1} \end{bmatrix} \tag{3.38}$$

which has dimension $\mathbb{R}^{36\times12}$.

Now, we will describe the second part of the Equation 3.34 using $\mathbf{F} \in \mathbb{R}^{54\times18}$. It is divided into two parts. The first part $\mathbf{F}_1$ contains the latter half part of the bearing rigidity matrix

in the Equation 2.23:

$$\mathbf{F_1} = \begin{bmatrix} \boxed{-\mathbf{S}} & & & & & \\ & \boxed{-\mathbf{S}} & & & & \\ & & \boxed{-\mathbf{S}} & & & \\ & & & \boxed{-\mathbf{S}} & & \\ & & & & \boxed{-\mathbf{S}} & \\ & & & & & \boxed{-\mathbf{S}} \end{bmatrix} \in \mathbb{R}^{3m \times 3m} = \mathbb{R}^{18 \times 18} \tag{3.39}$$

instead $\mathbf{F}_2$ is empty:

$$\mathbf{F}_2 = \mathbf{0}_{36 \times 18} \tag{3.40}$$

Now, we define two vectors: the first contains all bearings information, and the second contains the angular velocity around the z-axis:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \end{bmatrix} \in \mathbb{R}^{18 \times 1} \qquad \widetilde{\mathbf{w}} = \begin{bmatrix} \omega_1^z \\ \omega_2^z \\ \omega_3^z \end{bmatrix} \in \mathbb{R}^{3 \times 1} \tag{3.41}$$

and we design the following matrix in the following manner:

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0}_{36 \times 3} \end{bmatrix} \in \mathbb{R}^{54 \times 3} \text{ with } \mathbf{W} = \begin{bmatrix} \boxed{\mathbf{1}_{3\times1}} & & \\ \boxed{\mathbf{1}_{3\times1}} & & \\ & \boxed{\mathbf{1}_{3\times1}} & \\ & \boxed{\mathbf{1}_{3\times1}} & \\ & & \boxed{\mathbf{1}_{3\times1}} \\ & & \boxed{\mathbf{1}_{3\times1}} \end{bmatrix} \in \mathbb{R}^{18 \times 3} \tag{3.42}$$

As outlined in sections Chapter 3.18 the matrix design was chosen to allow the calculation of the equation $-S\boldsymbol{\beta}_{ij}w$ which forms a component of the Equation 2.23 and the Equation 2.24.

In the end, we define a vector with gravitational effects on agents' dynamics:

$$\widetilde{\mathbf{g}} = \begin{bmatrix} \mathbf{0}_{18\times1} \\ \mathbf{0}_{6\times1} \\ -g\mathbf{e}_3 \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{6\times1} \\ -g\mathbf{e}_3 \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{6\times1} \\ -g\mathbf{e}_3 \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.43}$$

3.2.2.3 Centralized MAS model with thrust and angular velocities control

After simulation and with further explanation provided in the Chapter 5, it is necessary to adjust the control input for the MAS. The main incentive for this modification is that the current controller is not sufficiently fast to calculate and transmit the control input to the flight control PX4. Therefore, we utilize the previously designed structure and modify the control input.

We begin by defining the state of our centralized MAS and removing the angular velocity of the agents:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \\ \mathbf{p}_1 \\ \boldsymbol{\alpha}_1 \\ \mathbf{v}_1 \\ \mathbf{p}_2 \\ \boldsymbol{\alpha}_2 \\ \mathbf{v}_2 \\ \mathbf{p}_3 \\ \boldsymbol{\alpha}_3 \\ \mathbf{v}_3 \end{bmatrix} \in \mathbb{R}^{45\times1} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\beta}}_{12} \\ \dot{\boldsymbol{\beta}}_{13} \\ \dot{\boldsymbol{\beta}}_{21} \\ \dot{\boldsymbol{\beta}}_{23} \\ \dot{\boldsymbol{\beta}}_{31} \\ \dot{\boldsymbol{\beta}}_{32} \\ \dot{\mathbf{p}}_1 \\ \dot{\boldsymbol{\alpha}}_1 \\ \dot{\mathbf{v}}_1 \\ \dot{\mathbf{p}}_2 \\ \dot{\boldsymbol{\alpha}}_2 \\ \dot{\mathbf{v}}_2 \\ \dot{\mathbf{p}}_3 \\ \dot{\boldsymbol{\alpha}}_3 \\ \dot{\mathbf{v}}_3 \end{bmatrix} \in \mathbb{R}^{45\times1} \tag{3.44}$$

where:

- $\boldsymbol{\beta}_{ij}$ is expressed in $\mathcal{F}_{\mathcal{B}}$
- $\mathbf{p}_i, \boldsymbol{\alpha}_i, \mathbf{v}_i$ are expressed in $\mathcal{F}_{\mathcal{W}}$

We introduce the new control input by substituting torque with angular velocity for each body axis. Furthermore, we notice that the size of the vector remains unchanged:

$$\mathbf{u} = \begin{bmatrix} T_1 \\ \boldsymbol{\omega}_1 \\ T_2 \\ \boldsymbol{\omega}_2 \\ T_3 \\ \boldsymbol{\omega}_3 \end{bmatrix} \in \mathbb{R}^{12\times 1} \text{ where } T_i \in \mathbb{R} \text{ and } \boldsymbol{\omega}_i \in \mathbb{R}^3 \text{ both in } \mathcal{F}_{\mathcal{B}} \tag{3.45}$$

then we rewrite the update equation (Equation 3.34) with the new control input:

$$\begin{aligned} \dot{\mathbf{x}} =& \mathbf{A}(\mathbf{x},t)\mathbf{x} + \mathbf{B}(\mathbf{x},t)\mathbf{u} + \widetilde{\mathbf{g}} + \mathbf{F}\widetilde{\boldsymbol{\beta}} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \\ =& \begin{bmatrix} \mathbf{A}_1(\mathbf{x},t) \\ \mathbf{A}_2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2(\mathbf{x},t) \end{bmatrix} \begin{bmatrix} T_1 \\ \boldsymbol{\omega}_1 \\ T_2 \\ \boldsymbol{\omega}_2 \\ T_3 \\ \boldsymbol{\omega}_3 \end{bmatrix} + \widetilde{\mathbf{g}} + \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \boldsymbol{\beta} \circ \widetilde{\mathbf{W}}\widetilde{\mathbf{w}} \end{aligned} \tag{3.46}$$

Now, the matrix $\mathbf{A}(\mathbf{x},t)$ in partitioned into two matrices with the first matrix $\mathbf{A}_1(\mathbf{x},t)$ constructed as follows:

$$\mathbf{A}_1(\mathbf{x},t) = \begin{bmatrix} \mathbf{0}_{3\times 24} & -\frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 6} & \frac{\mathbf{P}_{12}}{d_{12}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & -\frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6} & \frac{\mathbf{P}_{13}}{d_{13}}\mathbf{R}_1^{z^T} \\ \mathbf{0}_{3\times 24} & \frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 6} & -\frac{\mathbf{P}_{21}}{d_{21}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 24} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6} & -\frac{\mathbf{P}_{23}}{d_{23}}\mathbf{R}_2^{z^T} & \mathbf{0}_{3\times 6} & \frac{\mathbf{P}_{23}}{d_{23}}\mathbf{R}_2^{z^T} \\ \mathbf{0}_{3\times 24} & \frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6} & -\frac{\mathbf{P}_{31}}{d_{31}}\mathbf{R}_3^{z^T} \\ \mathbf{0}_{3\times 24} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6} & \frac{\mathbf{P}_{32}}{d_{32}}\mathbf{R}_3^{z^T} & \mathbf{0}_{3\times 6} & -\frac{\mathbf{P}_{32}}{d_{32}}\mathbf{R}_3^{z^T} \end{bmatrix} \tag{3.47}$$

were te matrix $\mathbf{A}_1(\mathbf{x},t)$ has dimension $\mathbb{R}^{3m\times(3m+12N)} = \mathbb{R}^{18\times 45}$. Instead, the matrix $\mathbf{A}_2$ is designed as follows:

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{0}_{3\times 18} & \mathbf{0}_{3\times 6} & \mathbf{I}_{3\times 3} & & & & & \\ \mathbf{0}_{3\times 18} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & & & & & \\ \mathbf{0}_{3\times 18} & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & & & & & \\ \mathbf{0}_{3\times 18} & & & \mathbf{0}_{3\times 6} & \mathbf{I}_{3\times 3} & & & \\ \mathbf{0}_{3\times 18} & & & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & & & \\ \mathbf{0}_{3\times 18} & & & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} & & & \\ \mathbf{0}_{3\times 18} & & & & & \mathbf{0}_{3\times 6} & \mathbf{I}_{3\times 3} \\ \mathbf{0}_{3\times 18} & & & & & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 18} & & & & & \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{3.48}$$

where $\mathbf{A}_2$ has dimension $\mathbb{R}^{9N\times(3m+12N)} = \mathbb{R}^{27\times 45}$

After that, we build the matrix $\mathbf{B}(\mathbf{x}, t)$ which is composed of two matrices:

$$\mathbf{B}_1 = \mathbf{0}_{18\times 12} \tag{3.49}$$

which one is empty because the control input doesn't affect the bearing information evolution. Instead, $\mathbf{B}_2(\mathbf{x}, t)$ contains a part of the Equation 2.4 and the Equation 2.5:

$$\mathbf{B}_2(\mathbf{x}, t) = \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} & & & \\ \mathbf{0}_{3\times 1} & \mathbf{I}_{3\times 3} & & & \\ \mathbf{R_1^{RPY}} \mathbf{e}_3 \frac{1}{m} & \mathbf{0}_{3\times 3} & & & \\ & & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} & \\ & & \mathbf{0}_{3\times 1} & \mathbf{I}_{3\times 3} & \\ & & \mathbf{R_2^{RPY}} \mathbf{e}_3 \frac{1}{m} & \mathbf{0}_{3\times 3} & \\ & & & & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} \\ & & & & \mathbf{0}_{3\times 1} & \mathbf{I}_{3\times 3} \\ & & & & \mathbf{R_3^{RPY}} \mathbf{e}_3 \frac{1}{m} & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{3.50}$$

which one has dimension $\mathbb{R}^{27\times 12}$. Now, we will describe the second part of the Equation 3.46 with $\mathbf{F} \in \mathbb{R}^{54\times 18}$. We divide the matrix into two parts, the first $\mathbf{F}_1$ contains the last part of the bearing rigidity matrix in the Equation 2.23:

$$\mathbf{F_1} = \begin{bmatrix} -\mathbf{S} & & & & & \\ & -\mathbf{S} & & & & \\ & & -\mathbf{S} & & & \\ & & & -\mathbf{S} & & \\ & & & & -\mathbf{S} & \\ & & & & & -\mathbf{S} \end{bmatrix} \in \mathbb{R}^{3m\times 3m} = \mathbb{R}^{18\times 18} \qquad \mathbf{F}_2 = \mathbf{0}_{27\times 18} \tag{3.51}$$

In the end, we define the matrices that we need to create the last part of the Equation 3.34. Initially, we define two vectors, which the first contains all information bearings and the second contains the angular velocity around the z-axis:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_{12} \\ \boldsymbol{\beta}_{13} \\ \boldsymbol{\beta}_{21} \\ \boldsymbol{\beta}_{23} \\ \boldsymbol{\beta}_{31} \\ \boldsymbol{\beta}_{32} \end{bmatrix} \in \mathbb{R}^{18\times 1} \qquad \widetilde{\mathbf{w}} = \begin{bmatrix} \omega_1^z \\ \omega_2^z \\ \omega_3^z \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{3.52}$$

and the matrix $\widetilde{\mathbf{W}}$ is defined as follows:

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0}_{27\times 3} \end{bmatrix} \in \mathbb{R}^{45\times 3} \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{1}_{3\times 1} & & \\ \mathbf{1}_{3\times 1} & & \\ & \mathbf{1}_{3\times 1} & \\ & \mathbf{1}_{3\times 1} & \\ & & \mathbf{1}_{3\times 1} \\ & & \mathbf{1}_{3\times 1} \end{bmatrix} \in \mathbb{R}^{18\times 3} \tag{3.53}$$

In conclusion, we define a vector with the gravitational effects on dynamics

$$\widetilde{\mathbf{g}} = \begin{bmatrix} \mathbf{0}_{18\times 1} \\ \mathbf{0}_{6\times 1} \\ -g\mathbf{e}_3 \\ \mathbf{0}_{6\times 1} \\ -g\mathbf{e}_3 \\ \mathbf{0}_{6\times 1} \\ -g\mathbf{e}_3 \end{bmatrix} \tag{3.54}$$

3.3 Model for decentralized NMPC

In this section, we will take some ideas from the paper [10] and Julian's Ph.D. thesis [13] with more our clarifications to implement this architecture.

Our control input is $\mathbf{u}_i = [f_i, \boldsymbol{\omega}_i]$ for the agent $\mathcal{A}_i$ and its predicted state can be expressed as:

$$\mathfrak{q}_i(t) = \mathfrak{q}_i(0) + \int_0^t \dot{\mathfrak{q}}_i(\tau) d\tau \tag{3.55}$$

$$\mathbf{v}_i(t) = \mathbf{v}_i(0) + \int_0^t \dot{\mathbf{v}}_i(\tau) d\tau \tag{3.56}$$

$$\mathbf{p}_i(t) = \mathbf{p}_i(0) + \int_0^t \dot{\mathbf{p}}_i(\tau) d\tau \tag{3.57}$$

for $t \in [0, T_p]$ and $\mathfrak{q}_i(t)$ is the agent quaternion. We notice that:

- $\mathfrak{q}_i(0)$ is evaluated from the know Roll and Pitch of $\mathcal{A}_i$ and its estimated yaw relative to the common reference frame available through a **decentralized bearing consensus localization algorithm** [9]
- $\mathbf{v}_i(0)$ is equally measured in $\mathcal{F}_i$ using optical flow and inertial sensor fusion

- $\mathbf{p}_i(0)$ is not available, like an absolute measurement or even estimation, in a frame common to all agents. But it is not very important because we work in inter-agent bearings information.

We must find the prediction of bearings information:

$$\boldsymbol{\beta}_i(t) = [\boldsymbol{\beta}_{i1}(t), \ldots, \boldsymbol{\beta}_{in}(t)] \tag{3.58}$$

which can be computed with the following equation:

$$\boldsymbol{\beta}_{ij}(t) = R_i^T(\psi_i(t))\frac{\mathbf{p}_j(t) - \mathbf{p}_i(t)}{\hat{d}_{ij}(t)} \tag{3.59}$$

To accomplish the prediction, we have to make two important assumptions:
1° ASSUMPTION: At every prediction step, the origin of the predicted position is the flat frame aligned with $\mathcal{F}_i$. Due to we can estimate the initial position of $\mathcal{A}_j$ as

$$\mathbf{p}_j(0) = \boldsymbol{\beta}_{ij}(0)\hat{d}_{ij} \text{ where } \hat{d}_{ij} \text{ is an inter-agent distance estimate} \tag{3.60}$$

The position $\mathbf{p}_i(t)$ and yaw $\psi_i(t)$ of $\mathcal{A}_i$ is determined relative to its current position **as a function of the optimization variables** (because the NMPC system incorporates the dynamics of the drone).
The $\mathbf{p}_j(t)$ can reasonably be estimated one of two ways:

1. $\mathcal{A}_j$ could communicate its predicted path from its previous RTI predictions step to $\mathcal{A}_i$
2. $\mathcal{A}_i$ could **assume** that $\mathcal{A}_j$ is moving as if it were reacting to the desired steering command $\mathcal{M}^d = [\mathbf{v}_{\mathcal{F}}^d, \dot{\psi}_{\mathcal{F}}^d, \dot{s}_{\mathcal{F}}^d]$

We decided to use the second manner because the first one is too computationally expensive.

2° ASSUMPTION: the observed quadrotors' motions over the prediction time are purely **based on the velocity resulting from the desired trivial motion**.
If we imagine the operation of a formation, likely, the different components of $\mathcal{M}^d$ will have different purposes:

- the linear velocity $\mathbf{v}_{\mathcal{F}}^d$ is used for moving the formation through space. It may **be constant for large periods**
- The yaw rate ($\dot{\psi}_{\mathcal{F}}^d$) and scale rate ($\dot{s}_{\mathcal{F}}^d$) of the formation however are likely to use for marginally corrections the heading and scale of the formation to a desired value.

If we imagine the tasks of a formation:

- the linear velocity $\mathbf{v}_{\mathcal{F}}^d$ is used for moving the formation through space and may be constant for large periods.
- the yaw and scale rate for the formation however are likely used to marginally correct the heading and scale of the formation to a desired value

Another important observation is that the large value of $\dot{\psi}_{\mathcal{F}}^d$ and $\dot{s}_{\mathcal{F}}^d$ at t=0 will not necessarily imply that large values at the end of the prediction horizon, thus we use a **damped estimate** for the predicted future steering commands:

$$\mathbf{v}_{\mathcal{F}}^d(t) = \mathbf{v}_{\mathcal{F}}^d(0) \tag{3.61}$$
$$\dot{\psi}_{\mathcal{F}}^d(t) = \dot{\psi}_{\mathcal{F}}^d(0)e^{\frac{-t}{\tau}} \tag{3.62}$$
$$\dot{s}_{\mathcal{F}}^d(t) = \dot{s}_{\mathcal{F}}^d(0)e^{\frac{-t}{\tau}} \tag{3.63}$$

The predicted position of $\mathcal{A}_j$ relative to the initial position of $\mathcal{A}_i$ can therefore be estimated by $\mathcal{A}_i$ to be (we recall the Equation 3.57)

$$\begin{aligned}\hat{\mathbf{p}}_j(t) = &\underbrace{\boldsymbol{\beta}_{ij}\hat{d}_{ij}}_{\hat{\mathbf{p}}_j(0)} + \\ &\int_0^t \underbrace{\mathbf{R}_i^T(\psi_i(\epsilon))\overbrace{\mathbf{v}_{\mathcal{F}}^d(\epsilon) + \dot{\psi}_{\mathcal{F}}^d(\epsilon)[\mathbf{z}_0]_\times(\mathbf{c}_{\mathcal{F}}(\epsilon) - \mathbf{p}_j(\epsilon)) + \dot{s}_{\mathcal{F}}^d(\epsilon)(\mathbf{c}_{\mathcal{F}}(\epsilon) - \mathbf{p}_j(\epsilon))}^{\mathbf{v}_{\mathcal{F}}}}_{\mathbf{v}_j} d\epsilon \\ &= \hat{\mathbf{p}}_j(0) + \int_0^t \mathbf{R}_i^T(\psi_i(\epsilon))[\mathbf{v}_{\mathcal{F}}^d(0) + \\ &\quad \dot{\psi}_{\mathcal{F}}^d(0)e^{\frac{-\epsilon}{\tau}}[\mathbf{z}_0]_\times(\mathbf{c}_{\mathcal{F}}(\epsilon) - \mathbf{p}_j(\epsilon)) + \dot{s}_{\mathcal{F}}^d(0)e^{\frac{-\epsilon}{\tau}}(\mathbf{c}_{\mathcal{F}}(\epsilon) - \mathbf{p}_j(\epsilon))]d\epsilon\end{aligned} \tag{3.64}$$

with

- $t \in [0, T_p]$
- $\mathbf{z}_0 = \begin{bmatrix}0 & 0 & z_0\end{bmatrix}^T$ is the altitude at the initial time of prediction ($t = 0$ is equal to t_0)
- $\mathbf{v}_{\mathcal{F}}^d(t) = \mathbf{v}_{\mathcal{F}}^d(0) = const$ for each prediction step
- guess for $\hat{d}_{ij}$

where the formation centroid can be estimated at t=0 and evolves as

$$\dot{\mathbf{c}}_{\mathcal{F}}(t) = \mathbf{R}_z(\psi_{\mathcal{F}}(t))\mathbf{v}_{\mathcal{F}}^d(t) \tag{3.65}$$

concerning the formation centroid at t=0. Given an initial bearing measurement, state estimate, and future control input, $\mathcal{A}_i$ is thus able to estimate $\mathbf{p}_i(t)$,$\mathbf{p}_j(t)$ and $\psi_i(t)$ for all

$0 \leq t \leq T_p$ relative to $\mathcal{F}_i$. So, we can use the Equation 3.59 where every agent can predict the evolution of the formation.

Therefore, in the NMPC system model, we embedded the following equation:

$$\hat{\mathbf{p}}_j^{'}(t) = \mathbf{R}_i^T(\psi_{\mathcal{F}}(t))\mathbf{v}_{\mathcal{F}}(t) \tag{3.66}$$

where the NMPC integrates it with a numerical method (for the sake of simplicity, we have used a forward Euler to give an idea), hence we have:

$$\hat{\mathbf{p}}_j(t+1) = \hat{\mathbf{p}}_j(t) + \mathbf{R}_i^T(\psi_{\mathcal{F}}(t))\mathbf{v}_{\mathcal{F}}(t)T_s \tag{3.67}$$

where T_s is the sampling time of the NMPC.

In the end, we can resume the procedure for the prediction:

1. we have $\mathbf{v}_{\mathcal{F}}^d, \dot{\psi}_{\mathcal{F}}^d, \dot{s}_{\mathcal{F}}^d$, initial $\boldsymbol{\beta}_i(t)$, position and velocities of other agents, and a guess on $\hat{d}_{ij}$,
2. **our goal** is to compute the bearing information with the Equation 3.59
3. compute the initial position of agent j with the Equation 3.60
4. compute the initial centroid's position with the Equation 3.65
5. compute the Equation 3.64 with iteration on integral
6. now, we have $\hat{\mathbf{p}}_j$ and we can compute the bearing information a new step thought the Equation 3.59
7. uses the decentralized consensus localization and we find the estimated state of agent j
8. to find the angular velocity applied on the drone, we use $w_j(t) = \mathbf{R}_j^T\dot{\psi}_{\mathcal{F}}^d$
9. repeat step 5

In the case of decentralized NMPC in a triangular formation, a likely state of $\mathcal{A}_1$ can be:

$$\mathbf{x}_1 = \begin{bmatrix} \boldsymbol{\beta}_{12} & \boldsymbol{\beta}_{13} & \mathbf{p}_1 & \boldsymbol{\alpha}_1 & \mathbf{v}_1 & \mathbf{w}_1 & \hat{\mathbf{p}}_2 & \hat{\mathbf{p}}_3 & \mathbf{c}_{\mathcal{F}} \end{bmatrix}^T \tag{3.68}$$

3.4 Trivial motion controller for formation control

The most straightforward trivial control that we can implement is a PID controller. In our case, we implemented three distinct PI controllers, one for any trivial motion. We formalize it as follows:

$$\mathbf{v}_{\mathcal{F}}(k) = \mathbf{P}_{\mathbf{v}}\mathbf{e}_{\mathbf{c}_{\mathcal{F}}}(k) + \mathbf{I}_{\mathbf{v}} \sum_{i=0}^{k} \mathbf{e}_{\mathbf{c}_{\mathcal{F}}}(i)\Delta T_s \qquad \mathbf{e}_{\mathbf{c}_{\mathcal{F}}}(k) = \mathbf{c}_F^d - \mathbf{c}_F(k) \in \mathbb{R}^3$$

$$\omega_{z_{\mathcal{F}}}(k) = P_{\omega}e_{\psi_{\mathcal{F}}}(k) + I_{\omega} \sum_{i=0}^{k} e_{\psi_{\mathcal{F}}}(i)\Delta T_s \qquad e_{\psi_{\mathcal{F}}}(k) = \psi_{\mathcal{F}}^d - \psi_{\mathcal{F}}(k) \in \mathbb{R}$$

$$\dot{s}_{\mathcal{F}}(k) = P_{\dot{s}}e_{s_{\mathcal{F}}}(k) + I_{\dot{s}} \sum_{i=0}^{k} e_{s_{\mathcal{F}}}(i)\Delta T_s \qquad e_{s_{\mathcal{F}}}(k) = s_{\mathcal{F}}^d - s_{\mathcal{F}}(k) \in \mathbb{R}$$

with $\mathbf{P}_{\mathbf{v}}, \mathbf{I}_{\mathbf{v}} \in \mathbb{R}^{3\times3}$ instead $P_{\omega}, P_{\dot{s}}, I_{\omega}, I_{\dot{s}} \in \mathbb{R}$. We opted to use a PI controller since at present, we only need to track step signals. Furthermore, introducing a derivative term in real experiments may result in a amplified and perturbed response to the noisy signal and consequently require the implementation of a filter.

The trivial motion controller needs to have the formation state like input and it generates the trivial motion like output. To achieve this, we need to define two maps:

- Map 1 ($\mathbf{M}_1$): convert the agents state $\mathbf{x}$ to formation state $\mathbf{q}_{\mathcal{F}}$
- Map 2 ($\mathbf{M}_2$): map from trivial motion $\mathcal{M}$ to linear and angular velocities for each agent $[\mathbf{v}_i, \omega_{z,i}]_{i=1}^N$

A scheme of the trivial motion controller (TMC) with these two maps is depicted in the figure below:

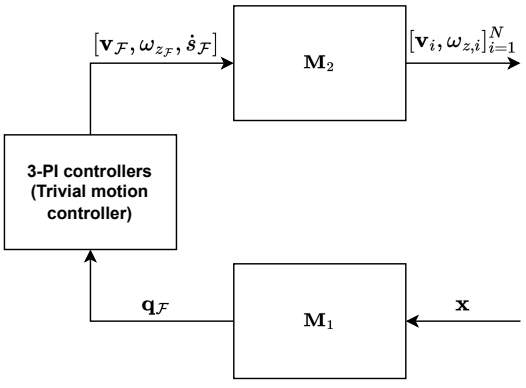


FIGURE 3.2: Scheme of the trivial motion controller with the transformation maps

In the figure below, we can see the TMC scheme above with a global point of view where it is connected with NMPC:

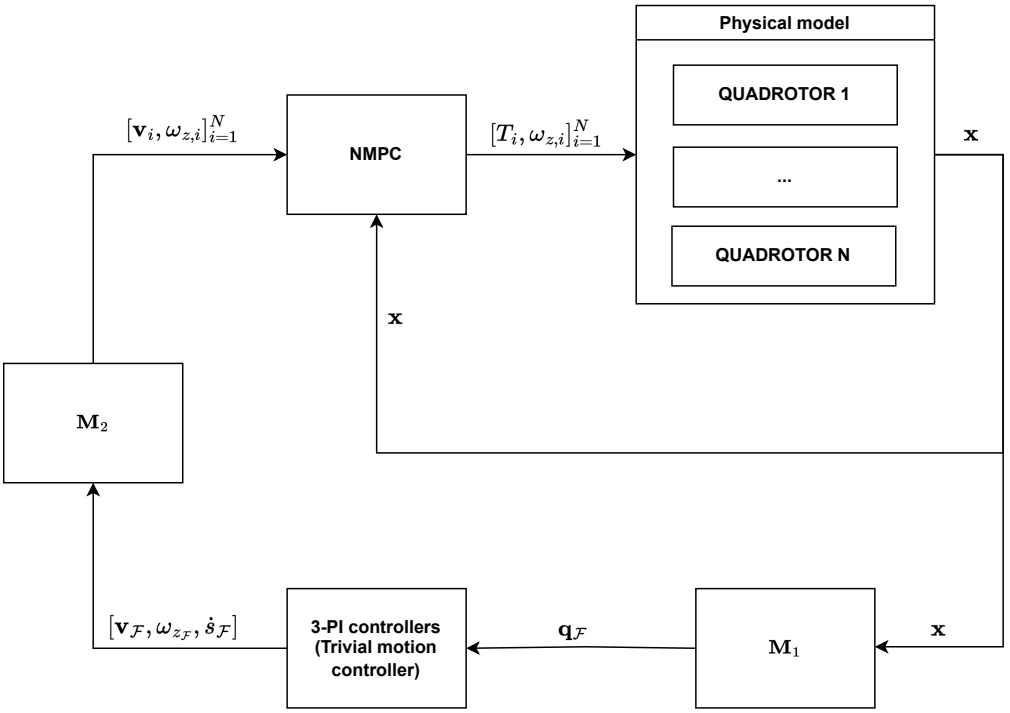


FIGURE 3.3: NMPC and TCM in the control loop

We start to define the formation state like

$$\mathbf{q}_\mathcal{F} = \begin{bmatrix} \mathbf{c}_\mathcal{F} & \psi_\mathcal{F} & s_\mathcal{F} \end{bmatrix}^T \in \mathbb{R}^5 \text{ with } \mathbf{c}_\mathcal{F} \in \mathbb{R}^3 \tag{3.69}$$

instead $\mathbf{x} \in \mathbb{R}^{3m+9N} = [\mathbf{x}_\beta \quad \mathbf{x}_\mathcal{A}]^T$ contains the bearing information $\mathbf{x}_\beta$ and the agents state $\mathbf{x}_\mathcal{A}$ namely the positions, rotations, and linear velocities of all agents. After formalizing these concepts, we define the first map that transforms from the state of agents $\mathbf{x}_\mathcal{A}$ to formation state $\mathbf{q}_\mathcal{F}$ in this way (N is the agent number of formation):

$$\mathbf{M}_1 : \mathbb{R}^{9N} \to \mathbb{R}^5 \tag{3.70}$$

$$\mathbf{x}_\mathcal{A} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_\psi \\ \mathbf{x}_\mathbf{v} \end{bmatrix} \mapsto \mathbf{q}_\mathcal{F} = \begin{bmatrix} \mathbf{c}_\mathcal{F}(\mathbf{x}_p) \\ \psi_\mathcal{F}(\mathbf{x}_\psi) \\ s_\mathcal{F}(\mathbf{x}_p) \end{bmatrix} \tag{3.71}$$

where $\mathbf{x}_p \in \mathbb{R}^{3N}$, $\mathbf{x}_\psi \ \in \mathbb{R}^{3N}$, $\mathbf{x}_\mathbf{v} \in \mathbb{R}^{3N}$ and each map is defined as follows:

$$\mathbf{c}_\mathcal{F} : \mathbb{R}^{3N} \to \mathbb{R}^3, \quad \mathbf{x}_p \mapsto \mathbf{c}_\mathcal{F} = \sum_i^N \frac{\mathbf{x}_{p_i}}{N} \tag{3.72}$$

$$\psi_\mathcal{F} : \mathbb{R}^{3N} \to \mathbb{R}, \quad \mathbf{x}_\psi \mapsto \psi_\mathcal{F} = \sum_i^N \frac{\mathbf{x}_{\psi_i}}{N} \tag{3.73}$$

$$s_\mathcal{F} : \mathbb{R}^{3N} \to \mathbb{R}, \quad \mathbf{x}_p \mapsto s_\mathcal{F} = \frac{A_\mathcal{F}}{A_\mathcal{F}^d} \tag{3.74}$$

where:

- $\mathbf{x}_{p_i}$ is the position of $\mathcal{A}_i$
- $\mathbf{x}_{\psi_i}$ is the yaw of $\mathcal{A}_i$
- $A^d_{\mathcal{F}}$ is the desired formation area
- $A_{\mathcal{F}}$ is the current formation area

For our case, the equation to compute the areas $A_{\mathcal{F}}$ and $A^d_{\mathcal{F}}$ is the fundamental equation for a triangle's area.

In the end, we define the second map that transforms the trivial motions $\mathcal{M}$ into the controller input for each agent in $\mathbb{R}^3 \times \mathbb{S}^1$ (as defined in the Equation 2.1 or in the Equation 2.6):

$$\mathbf{M}_2 : \mathbb{R}^5 \to \mathbb{R}^{4N} \tag{3.75}$$

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} {}^{\mathcal{W}}\mathbf{v}_{\mathcal{F}} \\ \dot{s}_{\mathcal{F}} \\ \dot{\psi}_{\mathcal{F}} \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{v}_1 \\ \cdots \\ \mathbf{v}_N \\ \omega_{z,1} \\ \cdots \\ \omega_{z,n} \end{bmatrix} \tag{3.76}$$

where the maps are defined as follows:

$$\mathbf{v}_i : \mathbb{R}^5 \to \mathbb{R}^3, \boldsymbol{\mathcal{M}} \mapsto \mathbf{v}_i = \mathbf{v}_{\mathcal{F}} + \dot{\psi}_{\mathcal{F}}[\mathbf{z}_0]_{\times}(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_i) + \dot{s}_{\mathcal{F}}(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_i) \tag{3.77}$$

$$\omega_{z,i} : \mathbb{R}^5 \to \mathbb{R}^3, \boldsymbol{\mathcal{M}} \mapsto \omega_{z,1} = \dot{\psi}_{\mathcal{F}} \tag{3.78}$$

The Equation 3.77 can be explained so:

- $\mathbf{v}_{\mathcal{F}}$ represents the linear velocity of the formation that is equal to the linear velocities of agent i
- $\dot{\psi}_{\mathcal{F}}[\mathbf{z}_0]_{\times}(\mathbf{c}_{\mathcal{F}} - \mathbf{p}_i)$ represents the formation rotation applied around z-axis with the distance between the formation barycenter and agent i position that represents the arm in the vector product
- $\dot{s}_{\mathcal{F}}(\epsilon)(\mathbf{c}_{\mathcal{F}}(\epsilon) - \mathbf{p}_i)$ represents the formation scale rate for agent i applied on the distance between the formation barycenter and agent i position i

3.5 Evolution of the cost functions

In this last section of this chapter, we want to show the evolution of cost function throughout the thesis project with their reasons.

3.5.1 Cost function for the quadrotor's simple model

First, we begin with the simplest cost function, namely a quadratic cost function:

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{x}^d, \Delta t) =& \frac{1}{2}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))^T\mathbf{S}_{\boldsymbol{\beta}}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))+ \\ & \frac{1}{2}\sum_{k=1}^{N_p}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))^T\mathbf{Q}_{\boldsymbol{\beta}}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))\end{aligned} \tag{3.79}$$

where the error on bearing is designed so $\mathbf{e}_{\boldsymbol{\beta}}(\cdot) = \boldsymbol{\beta}^d - \boldsymbol{\beta}(\cdot)$ instead, the matrices have the following meanings:

- $\mathbf{Q}_{\boldsymbol{\beta}} \geq \mathbf{0}$ is an weighted matrix on bearing error information
- $\mathbf{S}_{\boldsymbol{\beta}} > \mathbf{0}$ is a weighted matrix on bearing error for final term cost

also, we recall that N_p denotes the number of steps of the prediction horizon for our NMPC.

This first cost function achieves our goal of creating the formation. However, the disadvantage is that the NMPC continues to put effort into controlling input, resulting this is a waste of energy. Therefore, we have included an input cost term in the cost function:

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{x}^d, \mathbf{u}, \Delta t) =& \frac{1}{2}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))^T\mathbf{S}_{\boldsymbol{\beta}}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))+ \\ & \frac{1}{2}\sum_{k=1}^{N_p}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))^T\mathbf{Q}_{\boldsymbol{\beta}}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t)) + \mathbf{u}(k\Delta t)^T\mathbf{R}\mathbf{u}(k\Delta t)\end{aligned} \tag{3.80}$$

- $\mathbf{R} > \mathbf{0}$ is a weighted matrix that always allows to penalize the input

These two cost functions are effective for an NMPC problem with the first internal system model designed in the Section 3.2.1. But to apply the same cost function to a more complex model as we described above, modifications must be made.

3.5.2 Cost function for the quadrotor's complex model in SE(3)

The first step is to introduce a control input error in the cost function which allows NMPC to determine a control input that can counteract the gravitational force:

$$\mathbf{e}_{\mathbf{u}}(\cdot) = \mathbf{u}^d - \mathbf{u}(\cdot) \qquad \mathbf{u}^d = \begin{bmatrix}\mathbf{u}_1^d \\ \dots \\ \mathbf{u}_N^d\end{bmatrix} \qquad \mathbf{u}_i^d = \begin{bmatrix}gm_i \\ 0 \\ 0 \\ 0\end{bmatrix} \tag{3.81}$$

The second step is to introduce a cost term relating to the angles of the quadrotor. The motivation arose when, after some simulation, the NMPC had found a control input to achieve the desired formation. However, this had the side effect of imposing a 180 degree rotation around the pitch axis for all drones, causing the formation to move toward negative infinite space. We define the quadrotor's angles vector for each drone (Roll (ϕ), Pitch(θ) and Yaw(ψ)) and its angles error:

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 \end{bmatrix} \qquad \boldsymbol{\alpha}_i = \begin{bmatrix} \phi_i \\ \theta_i \\ \psi_i \end{bmatrix} i = 1, ..., n \qquad \mathbf{e}_{\boldsymbol{\alpha}} = \mathbf{0}_{12} - \boldsymbol{\alpha} \tag{3.82}$$

Therefore, we have arrived at this updated cost function:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{x}^d, \boldsymbol{\alpha}, \mathbf{u}, \Delta t) = & \frac{1}{2}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\mathbf{e}_{\boldsymbol{\alpha}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\boldsymbol{\alpha}} \mathbf{e}_{\boldsymbol{\alpha}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\sum_{k=1}^{N_p} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))^T \mathbf{Q}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t)) \\ & + \mathbf{e}_{\boldsymbol{\alpha}}^T(k\Delta t)\mathbf{Q}_{\boldsymbol{\alpha}}\mathbf{e}_{\boldsymbol{\alpha}}(k\Delta t) \\ & + \mathbf{e}_{\mathbf{u}}(k\Delta t)^T \mathbf{R}\mathbf{e}_{\mathbf{u}}(k\Delta t) \end{aligned} \tag{3.83}$$

where the meaning of new matrices are:

- $\mathbf{Q}_{\boldsymbol{\alpha}} \geq \mathbf{0}$ is a weighted matrix on angles error
- $\mathbf{S}_{\boldsymbol{\alpha}} \geq \mathbf{0}$ is a weighted matrix on angles error for final term cost

Now, we want NMPC to have the capability to generate a control input, which will maintain the formation at a specific altitude. To accomplish this, a vector is defined that comprises the altitudes of all agents and another with their altitudes error :

$$\mathbf{z}(\cdot) = \begin{bmatrix} z_1(\cdot) \\ z_2(\cdot) \\ z_3(\cdot) \end{bmatrix} \text{ where } z_i \in \mathbf{p}_i \qquad \mathbf{z}^d = \begin{bmatrix} z_1^d \\ z_2^d \\ z_3^d \end{bmatrix} \qquad \mathbf{e}_{\mathbf{z}}(\cdot) = \mathbf{z}^d - \mathbf{z}(\cdot) \tag{3.84}$$

where:

- $\mathbf{z}$ is the altitude all agents in $\mathcal{F}_{\mathcal{B}}$
- $\mathbf{z}^d$ is the desired altitude for the whole formation express in $\mathcal{F}_{\mathcal{B}}$

In addition, a term is included to maintain the linear velocities as close to zero, preventing the quadrotors from executing aggressive maneuvers:

$$\mathbf{v}(\cdot) = \begin{bmatrix} \mathbf{v}_1(\cdot) \\ \mathbf{v}_2(\cdot) \\ \mathbf{v}_3(\cdot) \end{bmatrix} \qquad \mathbf{e}_{\mathbf{v}}(\cdot) = \begin{bmatrix} 0 \\ \dots \\ 0 \end{bmatrix} - \mathbf{v}(\cdot) \tag{3.85}$$

Then, this is the cost function with an error term for altitude positions and linear velocities:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{x}^d, \boldsymbol{\alpha}, \mathbf{z}, \mathbf{z}^d, \mathbf{u}, \Delta t) = & \frac{1}{2}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\mathbf{e}_{\mathbf{z}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\mathbf{z}} \mathbf{e}_{\mathbf{z}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\mathbf{e}_{\boldsymbol{\alpha}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\boldsymbol{\alpha}} \mathbf{e}_{\boldsymbol{\alpha}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\mathbf{e}_{\mathbf{v}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\mathbf{v}} \mathbf{e}_{\mathbf{v}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\sum_{k=1}^{N_p} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))^T \mathbf{Q}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t)) \\ & + \mathbf{e}_{\mathbf{z}}(k\Delta t)^T \mathbf{Q}_{\mathbf{z}} \mathbf{e}_{\mathbf{z}}(k\Delta t) \\ & + \mathbf{e}_{\boldsymbol{\alpha}}(k\Delta t)^T \mathbf{Q}_{\boldsymbol{\alpha}} \mathbf{e}_{\boldsymbol{\alpha}}(k\Delta t) \\ & + \mathbf{e}_{\mathbf{v}}(k\Delta t)^T \mathbf{Q}_{\mathbf{v}} \mathbf{e}_{\mathbf{v}}(k\Delta t) \\ & + \mathbf{e}_{\mathbf{u}}(k\Delta t)^T \mathbf{R} \mathbf{e}_{\mathbf{u}}(k\Delta t) \end{aligned} \tag{3.86}$$

where the meaning of new matrices are:

- $\mathbf{Q}_{\mathbf{z}} \geq \mathbf{0}$ is a weighted matrix on altitude error
- $\mathbf{Q}_{\mathbf{v}} \geq \mathbf{0}$ is a weighted matrix on linear velocities error
- $\mathbf{S}_{\mathbf{z}} \geq \mathbf{0}$ is a weighted matrix on altitude error for final term cost
- $\mathbf{S}_{\mathbf{v}} \geq \mathbf{0}$ is a weighted matrix on linear velocities for final term cost

3.5.3 Cost function with a trivial motion controller in the control loop

With the above cost functions, we achieve all our aims for handling a formation task. However, when we add the trivial motion control into the control loop in addition to NMPC, then it is necessary to make some adjustments to the cost function. Trivial motion control handles the three aspects of formation control, namely translation, rotation and scale of the formation. Therefore, we can omit the angles term and altitude term because they are **redundant**. In addition, we need to make a slight change in the definition of the

linear velocity error. Here, we replace the zero vector, representing the desired reference with the trivial motion velocities (obtained through the second map as specified in the Equation 3.77) which correspond to the desired linear velocities:

$$\mathbf{e_v}(\cdot) = \mathbf{v}^d - \mathbf{v}(\cdot) \qquad \mathbf{v}^d = \begin{bmatrix} \mathbf{v}^d_{\mathcal{M},1} \\ \dots \\ \mathbf{v}^d_{\mathcal{M},N} \end{bmatrix} \tag{3.87}$$

and we add the term of angular velocities (that we obtained by the second map defined in the Equation 3.77 and we denote them so $\omega^{\mathcal{M}}_{z,i}$) into the control input:

$$\mathbf{e_u}(\cdot) = \mathbf{u}^d - \mathbf{u}(\cdot) \qquad \mathbf{u}^d = \begin{bmatrix} \mathbf{u}^d_1 \\ \dots \\ \mathbf{u}^d_N \end{bmatrix} \qquad \mathbf{u}^d_i = \begin{bmatrix} gm_i \\ 0 \\ 0 \\ w^{\mathcal{M}}_{z,i} \end{bmatrix} \tag{3.88}$$

We assume the trivial motion velocities to be constant for the entire prediction horizon. Therefore, the following is the final our cost function with new changes:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{x}^d, \mathbf{u}, \Delta t) = & \frac{1}{2}\mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\mathbf{e}_{\mathbf{v}}(\mathbf{x}(N_p\Delta t))^T \mathbf{S}_{\mathbf{v}} \mathbf{e}_{\mathbf{v}}(\mathbf{x}(N_p\Delta t)) + \\ & \frac{1}{2}\sum_{k=1}^{N_p} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t))^T \mathbf{Q}_{\boldsymbol{\beta}} \mathbf{e}_{\boldsymbol{\beta}}(\mathbf{x}(k\Delta t)) \\ & + \mathbf{e}^T_{\mathbf{v}}(k\Delta t)\mathbf{Q}_{\mathbf{v}}\mathbf{e}_{\mathbf{v}}(k\Delta t) \\ & + \mathbf{e}_{\mathbf{u}}(k\Delta t)^T \mathbf{R}\mathbf{e}_{\mathbf{u}}(k\Delta t) \end{aligned} \tag{3.89}$$

The advantage of using this cost function is that we require less information from MAS compared to the previous cost functions. Another advantage of this one respect the cost function defined in the Equation 3.86 is that it allows to have a formation with agents placed at different altitudes with one change on the desired bearing information.

We notice that in this last cost function we have applied the weighted method to resolve the MOO problem (we explained in the Section 2.5.2) where we have the need to achieve two tasks:

- **Desired formation**: it is described through the error of bearing information
- **Move the formation**: it is described through the error on the velocities given by TMC

and the desired formation task is more relevant than other.

3.6 NMPC constraint sets

As we saw in the description of the NMPC, it computes a set of control inputs based on the internal system model and the constraints set on the state and the input. Therefore, the set of control input that the NMPC finds will satisfy these constraints.

In our case, we decided to use only the set of input constraints because this allows us to avoid infeasible solutions. We took this decision basis of the theoretical concepts and after some experiments with state constraints which we carried out. We noticed that if the constraints were too tight; the NMPC was could not find a feasible solution in most cases. For example, we put constraints on the quadcopters's angles because we wanted to avoid the quadcopters to perform aggressive manoeuvres so we chose $\pm 15^{\circ}$ as the constraints on the angles. It was a sensible constraint for us, but it turned out to generate infeasible solutions.

We understood that in this type of implementation it was not feasible to impose hard constraints on the state and the only solution was to include the constraints in the cost function as error terms, namely they appear as weak constraints.

In the end, we have described the input constraints for the internal model system with the complex dynamics of the quadcopter:

$$\begin{bmatrix} \mathbf{lb}_1 \\ \dots \\ \mathbf{lb}_N \end{bmatrix} \leq \begin{bmatrix} \mathbf{u}_1 \\ \dots \\ \mathbf{u}_N \end{bmatrix} \leq \begin{bmatrix} \mathbf{ub}_1 \\ \dots \\ \mathbf{ub}_N \end{bmatrix} \qquad \mathbf{lb}_i = \begin{bmatrix} lb_T \\ lb_{w_x} \\ lb_{w_y} \\ lb_{w_z} \end{bmatrix} \qquad \mathbf{ub}_i = \begin{bmatrix} ub_T \\ ub_{w_x} \\ ub_{w_y} \\ ub_{w_z} \end{bmatrix} \tag{3.90}$$

where lb refers to "lower bound" and ub refers to "upper bound" instead of the subscripts refer to thrust (T) and angular velocity (w_x,w_y,w_z).

Instead, for the internal model system with the quadcopter's model as a point with mass has the same dimension as the definition above but the upper and lower bounds are defined on the linear velocities (v_x,v_y,v_z) and the angular velocity along the z-axis (w_z).

Part II

Experimental approach

Chapter 4

Implementation

4.1 ROS and Gazebo

4.1.1 ROS

The Robotic Operatic System (or ROS) is a set of software frameworks that define the components, interfaces, and tools for building advanced robots. It provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where the processes are represented as nodes and communicate with each other using a publisher/subscriber protocol and a client/services protocol.

FIGURE 4.1: ROS pipeline

ROS has three main ways of communicating between nodes:

- Topics
- Services
- Actions

Publisher and subscriber

A publish/subscriber system is one in which there are producers of data (publishers) and consumers of data (subscribers). The publishers and subscribers know how to contact each other through the concept of a "topic", which is a common name so that the entities can find each other. For example, when you create a publisher, you must also give it a string that is the name of the topic; the same goes for the subscriber. All publishers and subscribers that are on the same topic name can communicate directly with each other. There can be zero or more publishers and zero or more subscribers on a given topic. When data is published on the topic by any of the publishers, all subscribers in the system will receive the data.

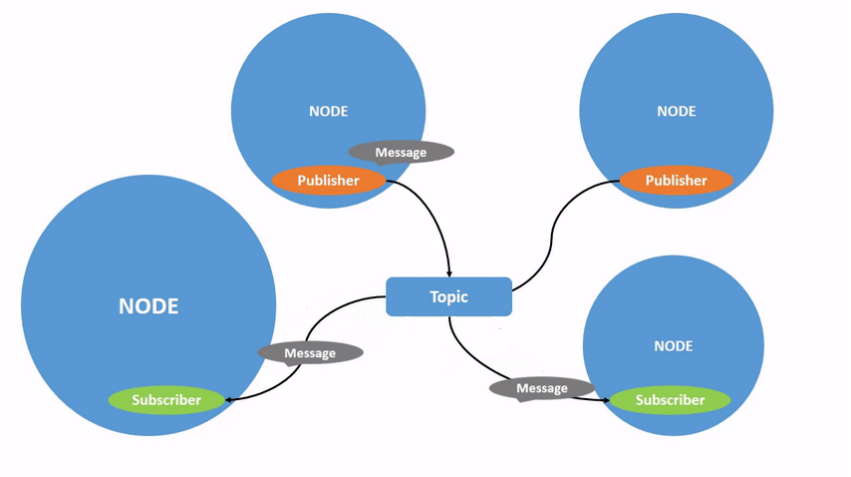


FIGURE 4.2: Publisher and subscriber communication. Source: ROS website

Service client/server

The service client/server is a structure that allows a node to make a call to another node and can receive a response from that call. It is used to get information from another node that doesn't take too much time to compute them.

The service consists of two parts:

- **Client**: it is able to make a call to a server
- **Server**: it receives a request from clients and it returns a response

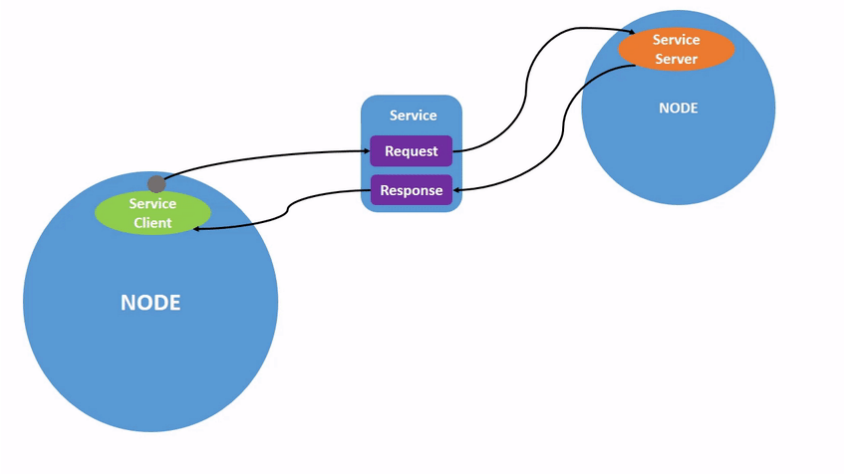


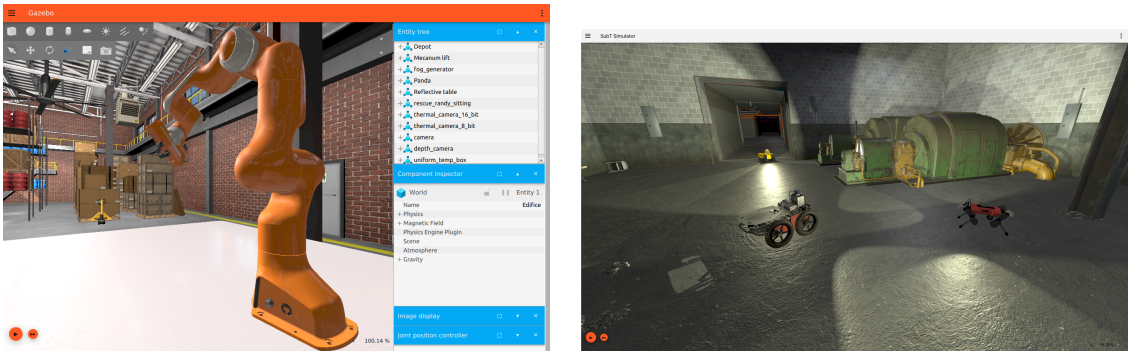
FIGURE 4.3: Service communication. Source: ROS website

Action client/server

Unlike publish/subscribe and service, the actions server is useful to ask information to processes that they take much time to complete. For example, to send a destination goal to the navigation stack and while it is completing the task, it returns some intermediate information like waypoint or other.

4.1.2 Gazebo simulator

Gazebo is an open-source 3D robotics simulation. It offers realistic rendering of environments, including high-quality lighting, shadows, and textures. It can model sensors that sense the simulated environment, such as laser range finders, cameras, etc...



(a) (b)

FIGURE 4.4: Examples of robots in Gazebo simulator

In our case, we will use Gazebo to simulate the flight of a quadcopter. We can see an example of a quadrotor on Gazebo in the figure below:

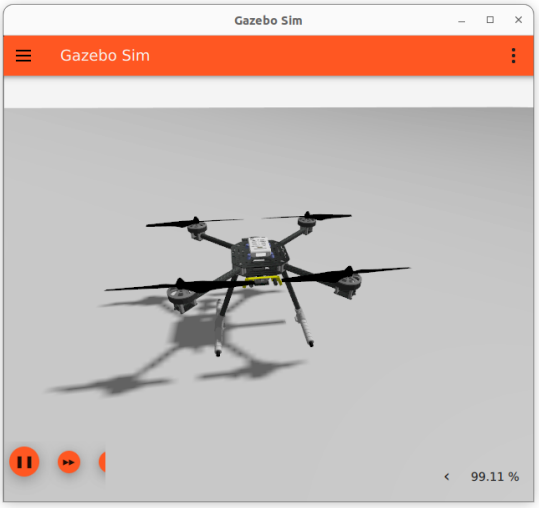


FIGURE 4.5: Simulation of X500 quadcopter in Gazebo environment. Source: PX4 website

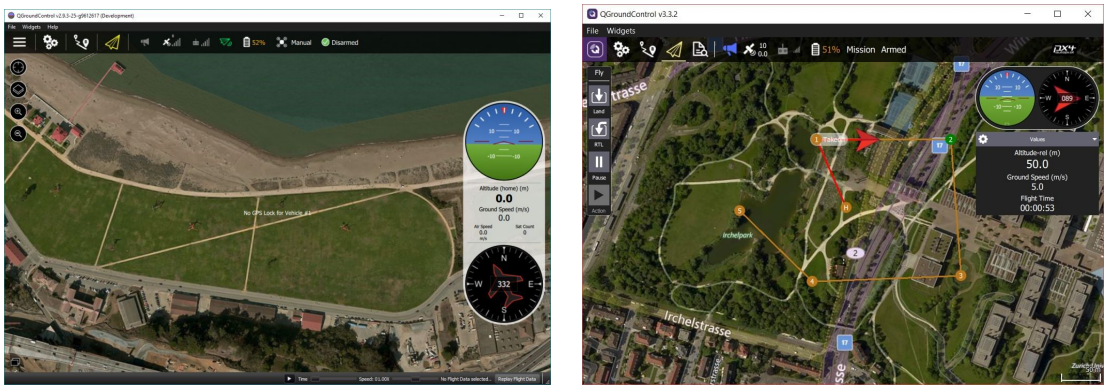
4.2 PX4

PX4 is the Professional Autopilot open source that powers all types of vehicles from racing and cargo drones through to ground vehicles and submersibles. It implements a complete flight stack on the supported hardware (an example is shown in the figure below).

(a) (b)

FIGURE 4.6: PX4 showcases. Source: PX4 website

There is a ground control station that is called *QGroundControl* which allows to load the PX4 firmware, setting up of vehicle settings, changing different parameters, getting real-time flight/mission information through either wired or wireless communication and creating fully autonomous missions.



(a) Interface (b) Mission planning

FIGURE 4.7: QGroundControl. Source: PX4 website

4.2.1 Chain of controllers

In the Figure 4.8 is depicted the standard cascaded control architecture that is implemented on a PX4 supported hardware. It is composed of:

- Position control in $\mathcal{F}_{\mathcal{W}}$
- Velocity control in $\mathcal{F}_{\mathcal{W}}$
- Conversion from Acceleration and Yaw control to Attitude control
- Angle control in $\mathcal{F}_{\mathcal{B}}$
- Angular rate control in $\mathcal{F}_{\mathcal{B}}$
- Mixer that converts torque and thrust in spinning rotors in $\mathcal{F}_{\mathcal{B}}$

and these controllers are a mix of P and PID controllers. The PX4 uses an EKF2 to estimate the state information that it needs them to use the controllers.

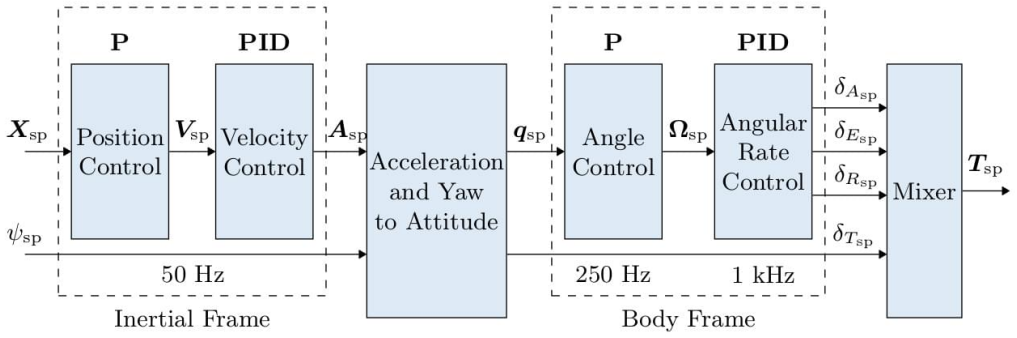


FIGURE 4.8: Cascaded control architecture for quadcopters. Source: PX4 website

We can notice in the figure above that a person can implement any controller in the chain, such as an offboard controller. It is important that the self-implemented controller provides the control input at a specific rate and with the correct reference frame.

In our experiments, we implemented the NMPC as an offboard controller with thrust and angular velocities that it provides for angular rate control. We had tried using an NMPC with thrust and torque but the controller was not fast enough to achieve the velocities required by the PX4's internal controller.

4.2.2 PX4 and ROS2

The ROS-PX4 architecture provides an integration between ROS2 and PX4, allowing ROS2 subscriber or publisher nodes to interface directly with PX4 uORB topics. This is provided by the uXRCE-DDS (Micro eXtremely Resource Constrained Data Distribution Service) communication middleware and in the Figure 4.9 is depicted, the pipeline implemented in PX4:

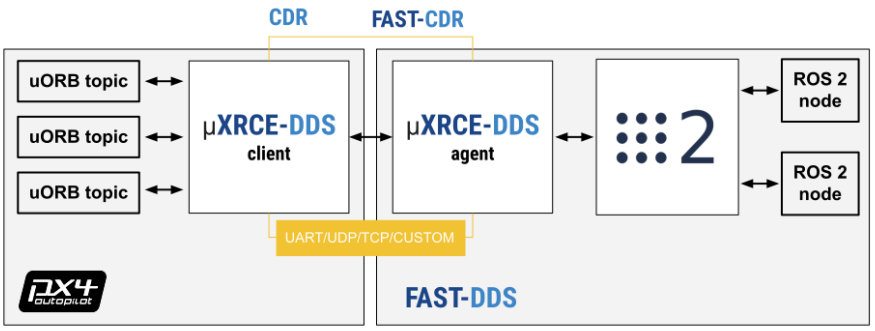


FIGURE 4.9: uXRCE-DDS middleware. Source: PX4 website

The uXRCE-DDS middleware consists of a client running on the PX4 and an agent running on the companion computer (such as Raspberry). The agent acts as a proxy for the client to publish and subscribe to topics in the global DDS data space.

Thanks to DDS middleware, a person is able to implement his offboard controller in ROS2 and he places it on a companion computer such as a PC or directly on a quadcopter such as a Raspberry PI, a PINE64, etc.

4.3 MATMPC [5]

The MATMPC is an open source software built in MATLAB for nonlinear model predictive control (NMPC). It is designed to facilitate modelling, controlled design and simulation for a wide class of NMPC applications. MATMPC has a number of algorithmic modules,

including automatic differentiation, direct multiple shooting, condensing, linear quadratic program (QP) solver and globalization.

MATMPC has been designed to provide state-of-the-art performance while making the prototyping easy, also with limited programming knowledge. This is achieved by writing each module directly in MATLAB API for C. As a result, MATMPC modules can be compiled into MEX functions with performance comparable to plain C/C++ solvers. MATMPC has been successfully used in operating systems including WINDOWS, LINUX and OS X.

We decided to use this software because it allows us to implement the controller in MATLAB in the easiest way compared to other software/libraries and also it allows us to try different solvers to understand which ones are more suitable for our problem.

4.4 Simulink model of the control pipeline

In this section, we see the implementation of the control pipeline on Simulink divided into the main blocks. In the figure below, we can observe the whole control pipeline, which is composed of 3 main blocks:

- NMPC block contains the implementation of NMPC
- MAS block contains the N block of the quadcopters' model
- TMC block contains the 3 PID

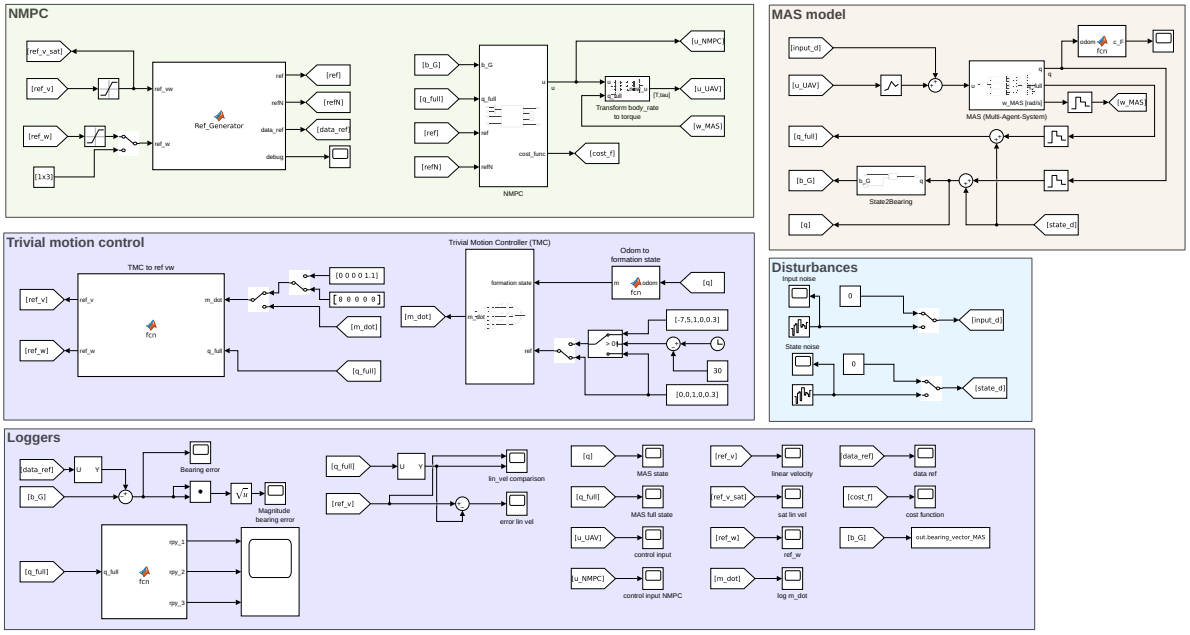


FIGURE 4.10: The Simulink scheme with NMPC, TMC and MAS physical model

In the Figure 4.11, we can see that the MAS model is composed of three quadcopter models because we need to simulate three different quadcopter dynamics.

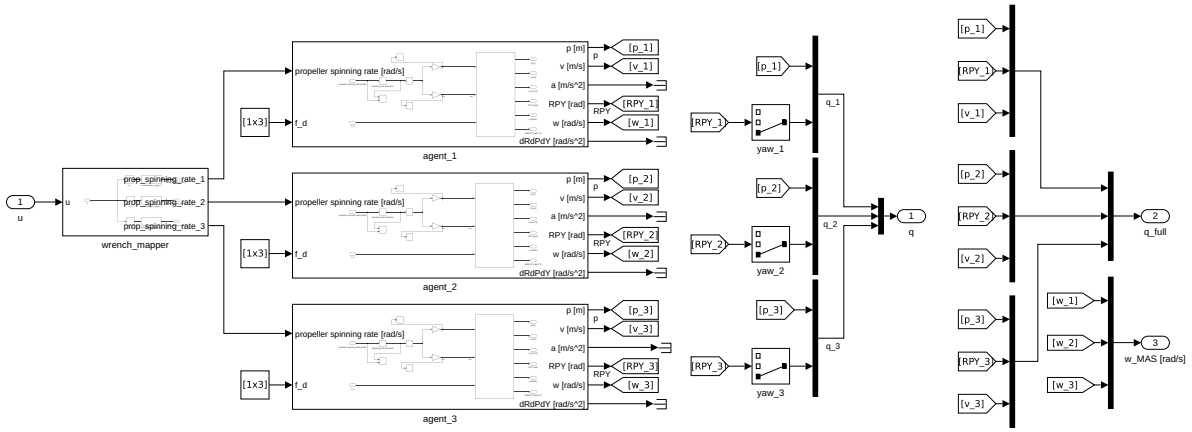


FIGURE 4.11: The Simulink scheme of the MAS physical model

The following figure shows how we implemented the quadcopter's model that we described in the Equation 2.2 and the Equation 2.2:

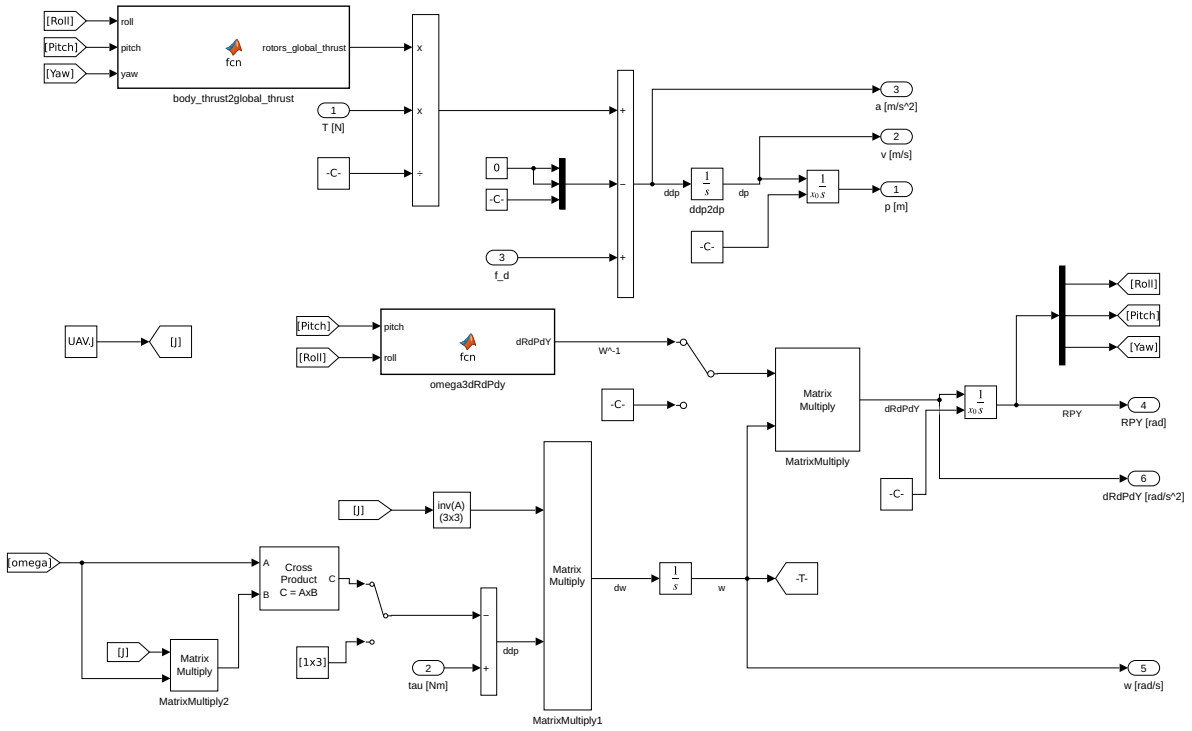


FIGURE 4.12: The Simulink scheme of an agent physical model

We can see in the figure below how we implemented the TMC on the Simulink model:

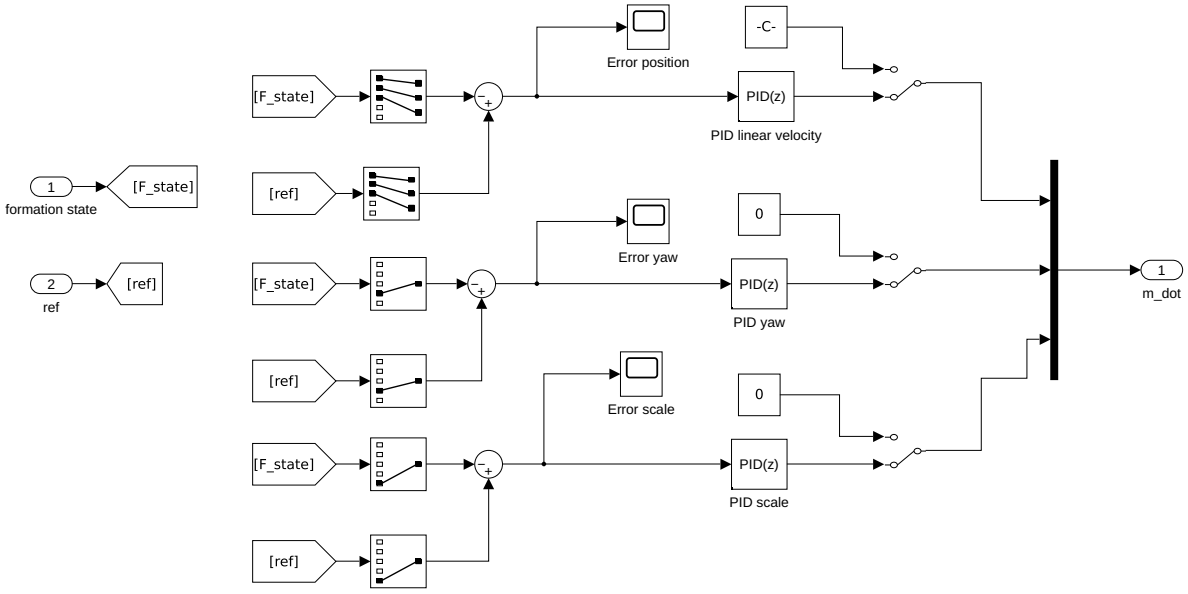


FIGURE 4.13: The Simulink scheme of the TMC

Finally, we show the rate controller that we have implemented on each quadrotor axis to handle the NMPC's control input. The gains we used are shown in the Table A.8 and are the same as those used by PX4.

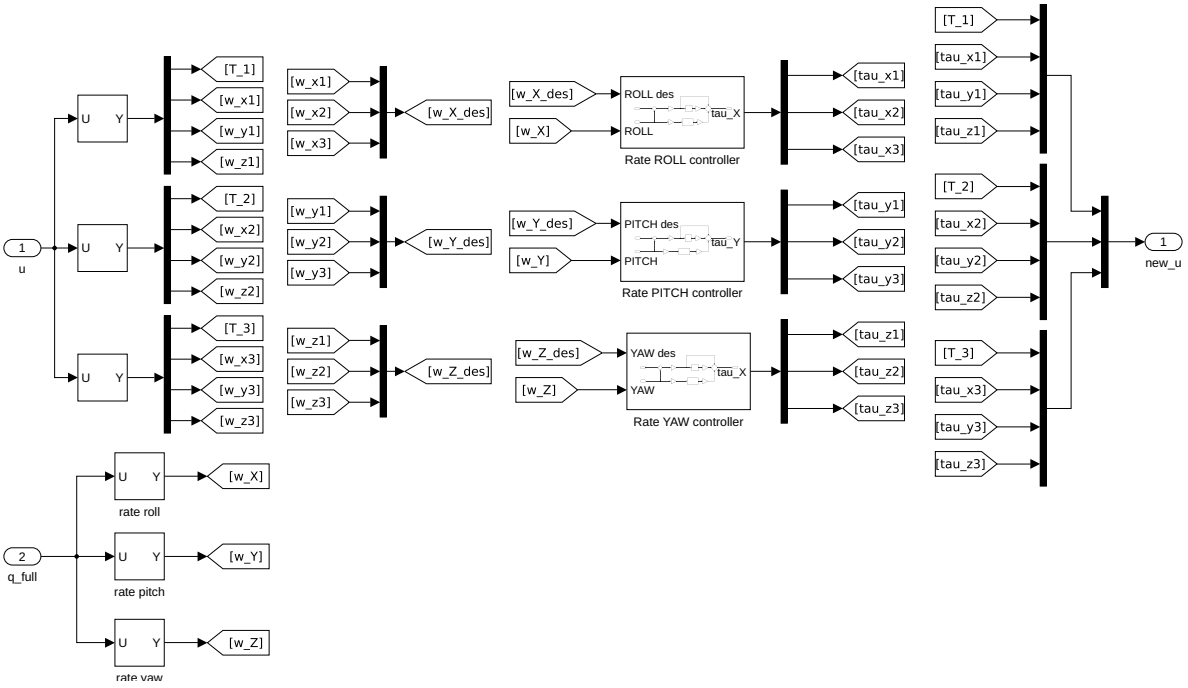


FIGURE 4.14: Rate controller for the axis x,y,z of the MAS

In the figure above, we can see that the rate controller is implemented as a K-PID. We add a derivative kick method to avoid the derivative spikes when the setpoint changes.

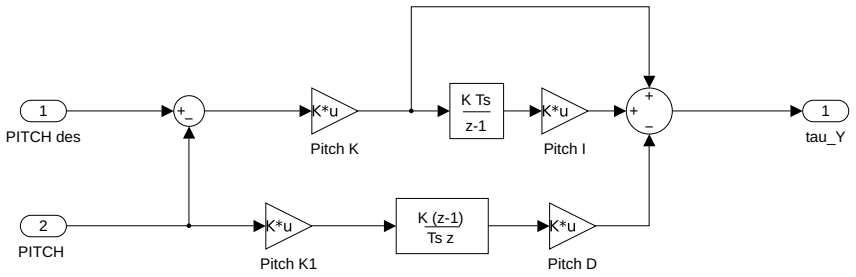


FIGURE 4.15: One of three rate controllers

4.5 Control pipeline on ROS

In the previous section, we saw the whole ecosystem of controllers and models implemented in Simulink and how they communicate with each other. Now, we need to translate and implement that ecosystem in another environment that it allows us to simulate our systems close to the real world. For this reason, we decided to use the ROS framework and implement a control pipeline with different objectives and features.

The control pipeline is designed with the goal in mind:

- **Adaptable**: it can be used with different controllers
- **Configurable**: it can handle N agents where N is variable
- **Centralized/Decentralized**: it can be configured with two different architecture

and it is depicted in the Figure 4.16. In origin, we thought the control pipeline to work only with Simulink model and after it arose the need to work also with NMPC implemented on C++ ROS2 node. For Simulink reason, we had need to create solution in C++ pipeline with the goal that it avoided to overload too much the Simulink model. For this reason, the middleware node clusters the odometry information in 4 main topics and the same for the control input that received from the NMPC node.

This need could be neglected with the node written in C++ with the implementation of NMPC, but this solution is born after some tests where the Simulink model is too much slower to control the formation in the Gazebo environment.

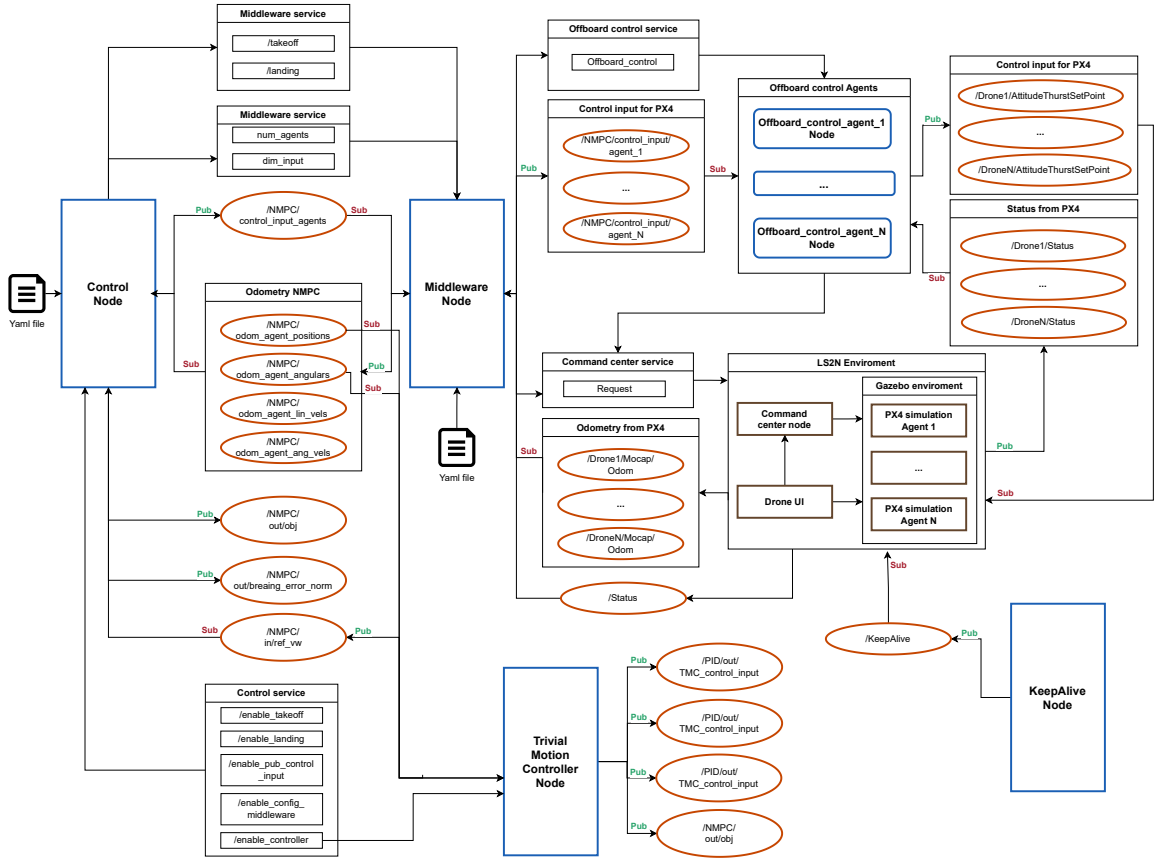


FIGURE 4.16: Control pipeline of the whole system

As we can see, the control pipeline is made up of many nodes and each node has a specific and important task:

- **Keepalive** node must continue to publish a message to enable the PX4 to receive offboard's message. Its flowchart is depicted in the Figure 4.17
- **Middleware PX4** node has the goal of taking all the information from the offboard control PX4 and translate them into information for the control node. The parallel goal is to take the information from the control node and send them in the correct way to a specific agent for the whole formation. Its flowchart is depicted in the Figure 4.18
- **NMPC/Control** node has to get information from middleware and computes the control input to control the agents of the formation. Its flowchart is depicted in the Figure 4.20
- **Offboard control** node runs for a specific formation's agent and it has the goal of managing all input and output message for its PX4 drone. Its flowchart is depicted in the Figure 4.19
- **TMC** node has the task of controlling the whole formation and it already implements the map M_1 and M_2 that we explained in the 3.4. Its flowchart is depicted in the Figure 4.21

4.5.1 Keepalive node

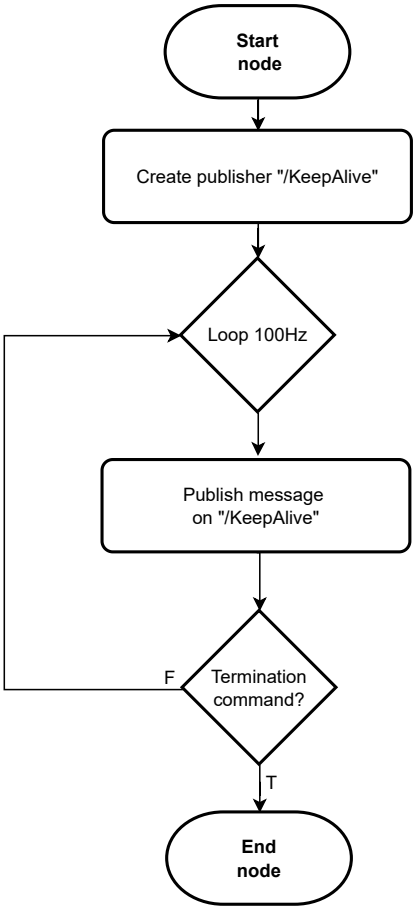


FIGURE 4.17: Flowchart of keepalive node

The keepalive node is very simple to accomplish its task to publish the keep alive message. This protocol is required by the PX4 architecture in order to enable the PX4 firmware to receive offboard control input.

4.5.2 Middleware node

The middleware node is at the heart of this control pipeline. Its main objective is divided into two parts:

- **Offboard control -> middleware -> NMPC controller**: The middleware node receives odometry from multiple offboard controllers and it groups them into four main topics which are:
  - positions odometry
  - rotation odometry

  - linear velocities odometry
  - angular velocities odometry

- **NMPC controller -> middleware -> offboard control**: The middleware receives a large vector containing the control input for each agent. The middleware divides the vector into groups, and each group represents a control input for a specific agent. Once the middleware has finished to divide the vector, it publishes it to the correct agent's topic.

The dashed diagonal rectangles represent the asynchronous call that other nodes may make during the node's life, such as a new message in a subscribed topic or a called service.

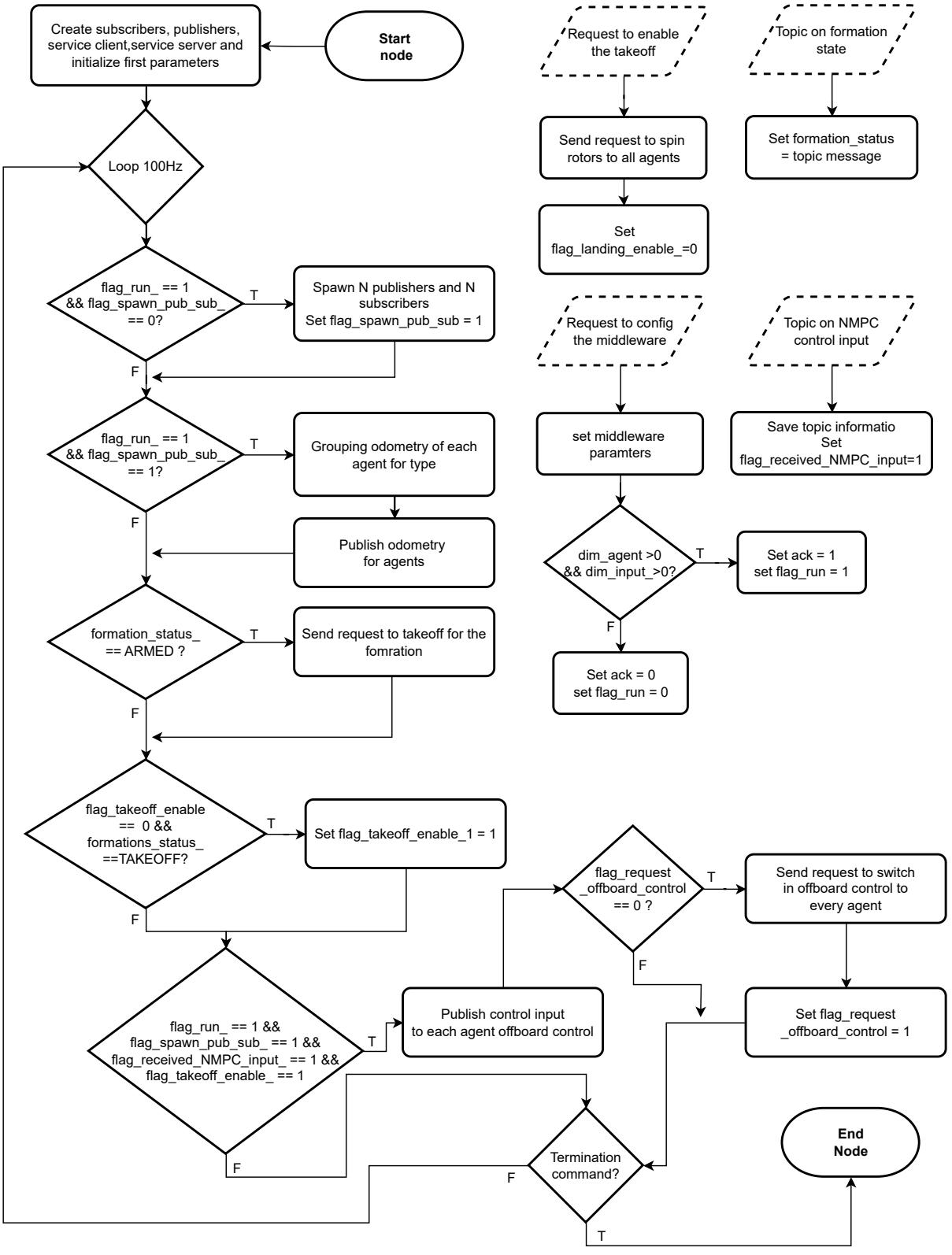


FIGURE 4.18: Flowchart of the middleware node

4.5.3 Offboard control node

The offboard control has the task of controlling the agent and managing the PX4 protocol, such as checking the keepalive message, when it has to enable or disable the PX4 offboard feature, which controls it needs to use, etc. If we work in multi-agent system mode, when we run the whole pipeline, we see that there are more offboard controllers working at the same moment. This is because each agent needs to have an offboard control with its PX4 firmware.

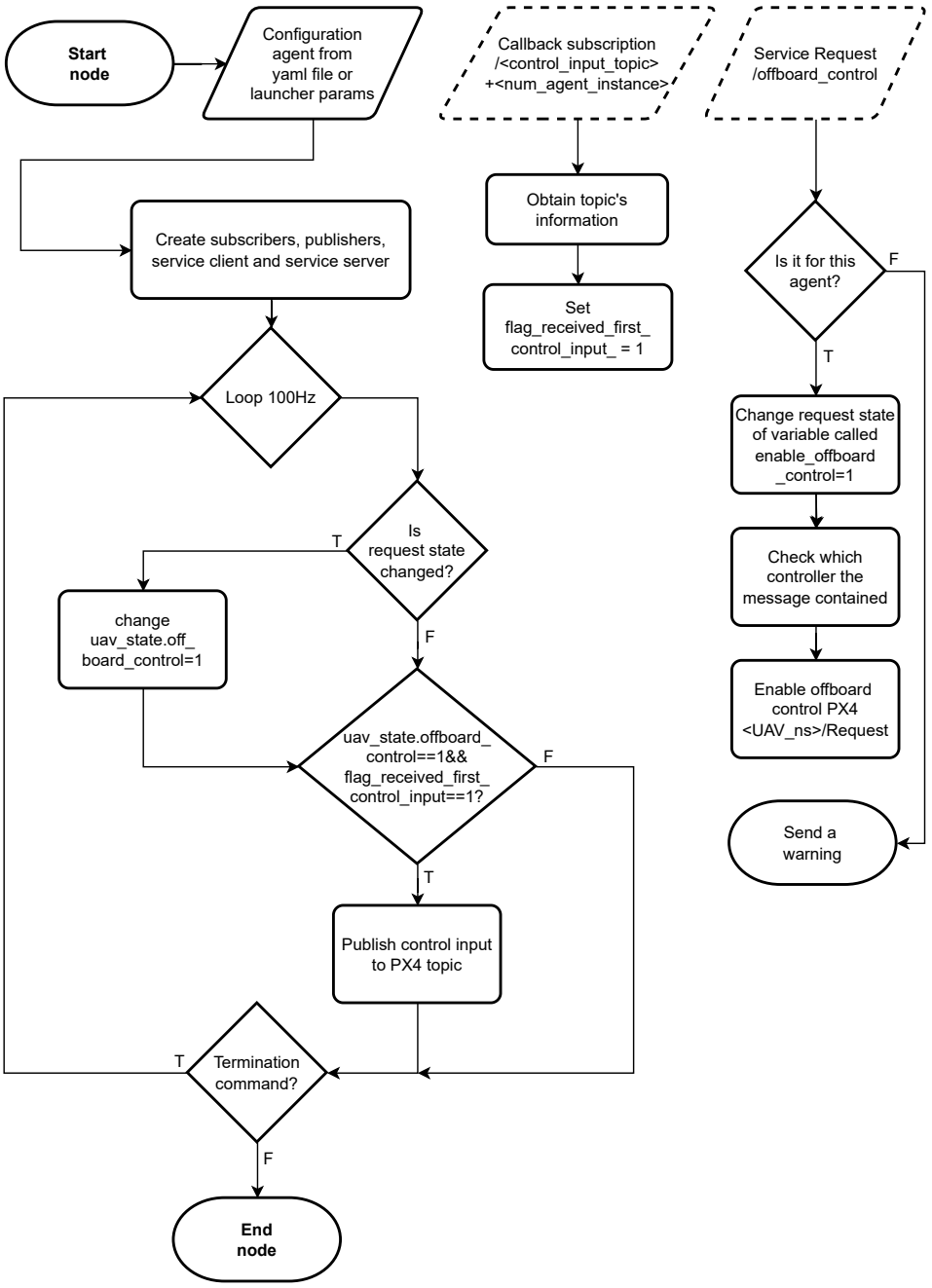


FIGURE 4.19: Flowchart of the offboard control node for an agent

4.5.4 NMPC control node

The NMPC control node is the brain of our pipeline. It has implemented the MATMPC(NMPC) and it uses the MATMPC to calculate the control input for our formation. It has to perform some checks (shown in the Figure 4.20) before publishing the control input to the middleware node. We also note that the NMPC operates at 100Hz or $T_s = 0.01ms$.

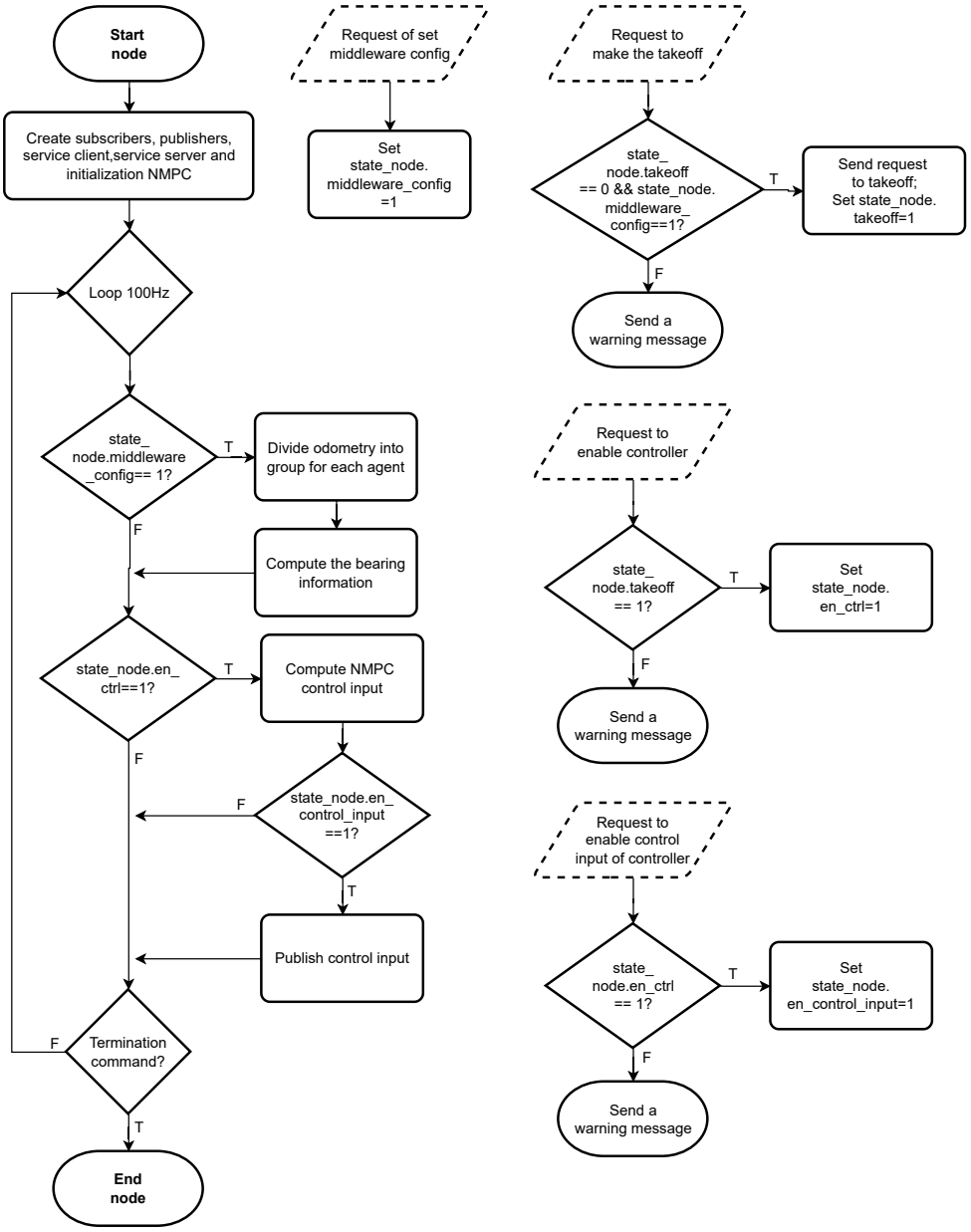


FIGURE 4.20: Flowchart of the NMPC node

4.5.5 TMC node

The Trivial Motion Control (or TMC) node has the task of managing the formation control with the three trivial motions. It has already implemented the map M_1 and M_2. In this

way, it has the odometry agent as the input and the output that it sends one to the NMPC node is the linear and angular velocity of each agent. We also note that the TMC operates at 20Hz or $T_s = 0.05ms$

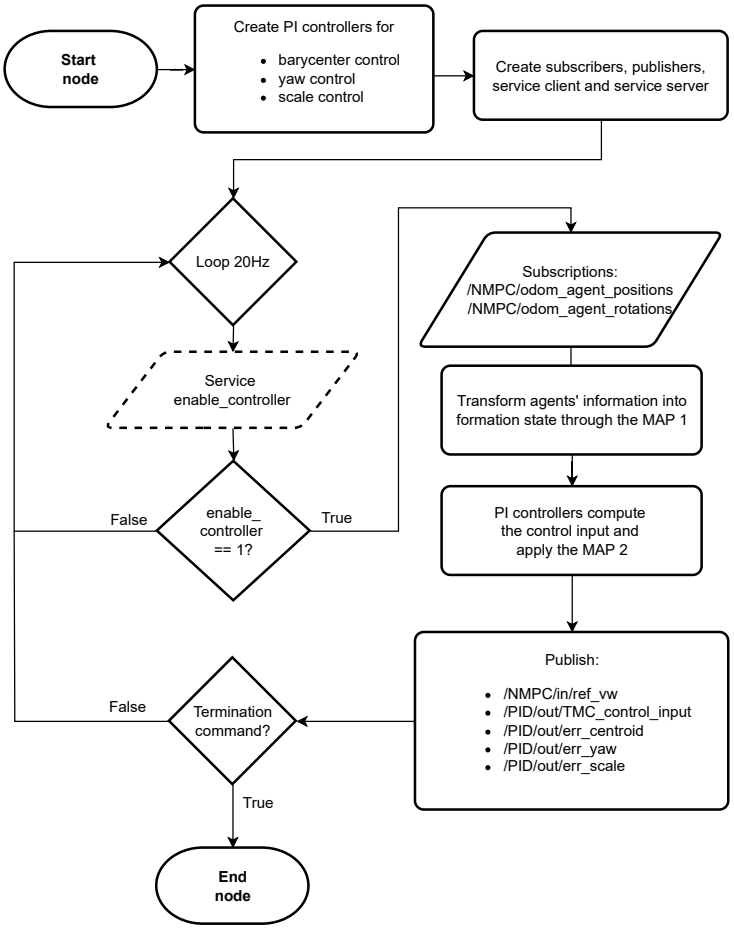


FIGURE 4.21: Flowchart of the trivial motion control (TMC) node

Chapter 5

Experiments

5.1 Experiments NMPC - Model with quadrotor like a singular integrator

In this section, we will refer to "the first cost function" as the cost function defined in the Equation 3.79 instead with "the second cost function" we will refer to the cost function defined in the Equation 3.80.

5.1.1 Constraint set

Now, we describe the constraints set that we used for the simulations and we refer to the Section 3.6. In the first moment, we tried the following first input constraint set:

$$\mathbf{0}_{12\times 1} \leq \mathbf{u} \leq \mathbf{I}_{12\times 1} \cdot 5 \tag{5.1}$$

but after some simulations, we noticed that the NMPC continued to apply an input and it could not reverse the quadcopters.

To solve these problems, we added a penalising term to the cost function and this new input constraint set so that the drones were able to reverse:

$$-\mathbf{I}_{12\times 1} \cdot 5 \leq \mathbf{u} \leq \mathbf{I}_{12\times 1} \cdot 5 \tag{5.2}$$

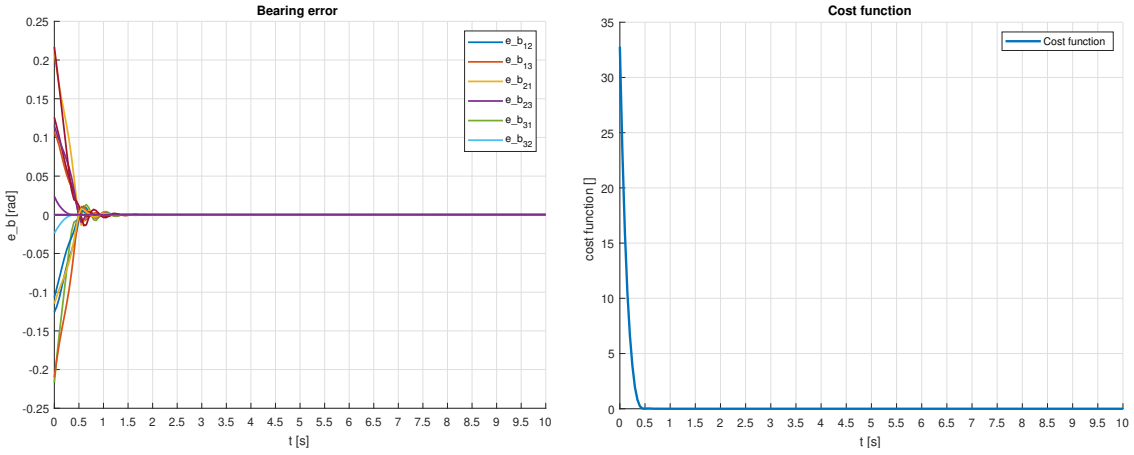
All simulations for this internal system model have this setup:

- $N_p = 10$ number of steps of the prediction horizon
- Ts= 0.05 is the sampling time of the whole system
- we used the Forward Euler method as the integration method

- the NMPC is implemented though *fmincon* Matlab function
- the **Q** and **R** are diagonal matrices and we refer to them as R=number and Q=number for simplicity

In the first simulation was activated the first input constraint set (eq (5.1)) instead in the second simulation was activated the second input constraint set (eq (5.2))

5.1.2 First cost function and First input constraint set



(a) Bearing error

(b) Cost function

FIGURE 5.1: Bearing error and cost function with Q=100

In the figures above, we can see that the bearing error went to zero faster and the cost function has the same behaviour. This means that our design of the internal system model described in the Section 3.2.1 worked as we wanted it to.

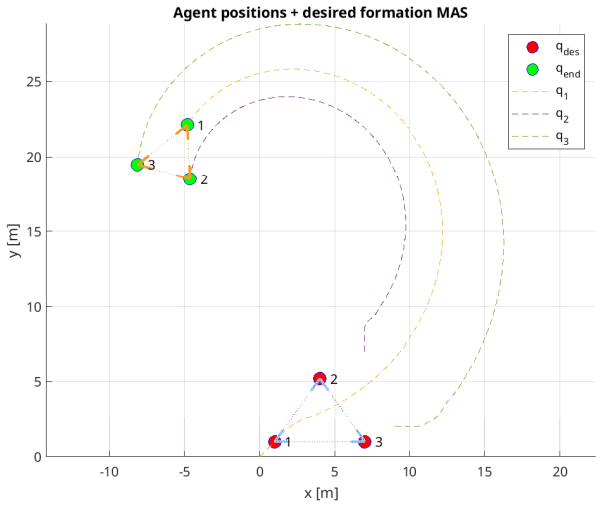


FIGURE 5.2: Trajectory and formation of MAS

In the Figure 5.2 the formation has a circular behaviour because there is not a penalization term on the cost function and we find another confirmation in Figure 5.3 because we can notice that the control input never went to zero.

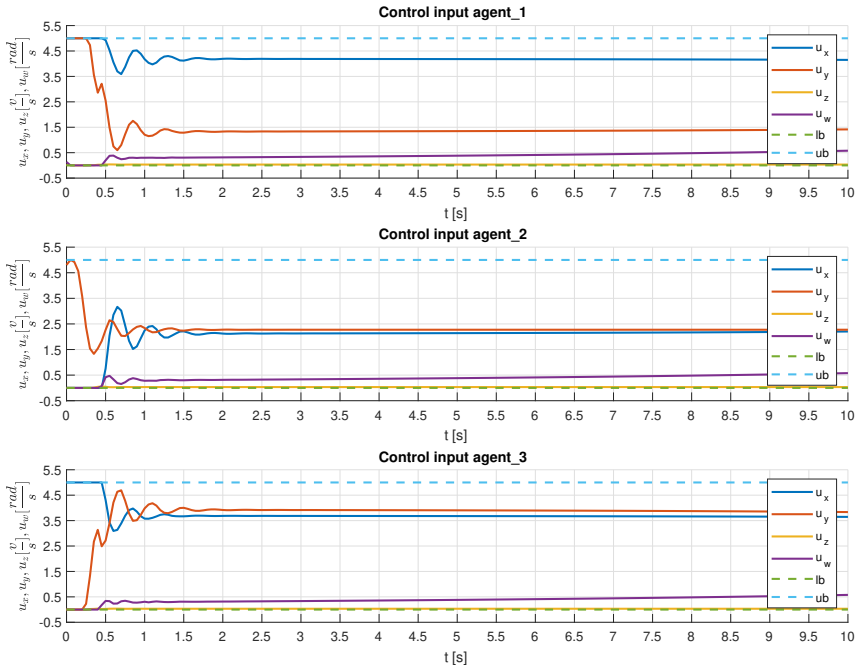


FIGURE 5.3: Control input for each agent

Furthermore, the figure below gives us a confirmation that the bearing information went to the desired bearing, hence the MAS has achieved the desired formation.

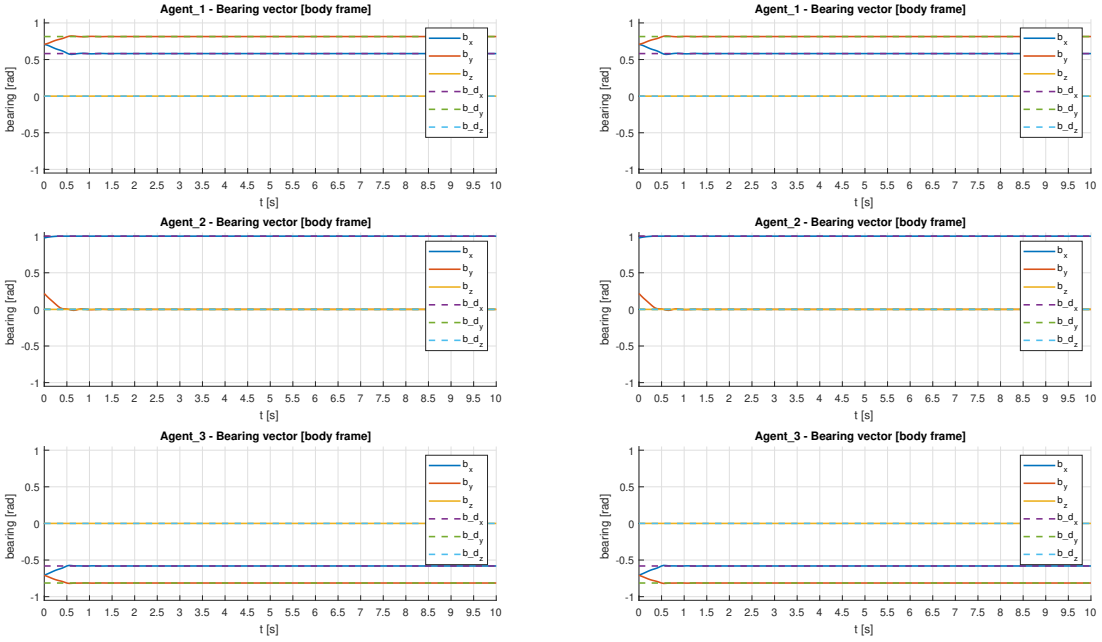
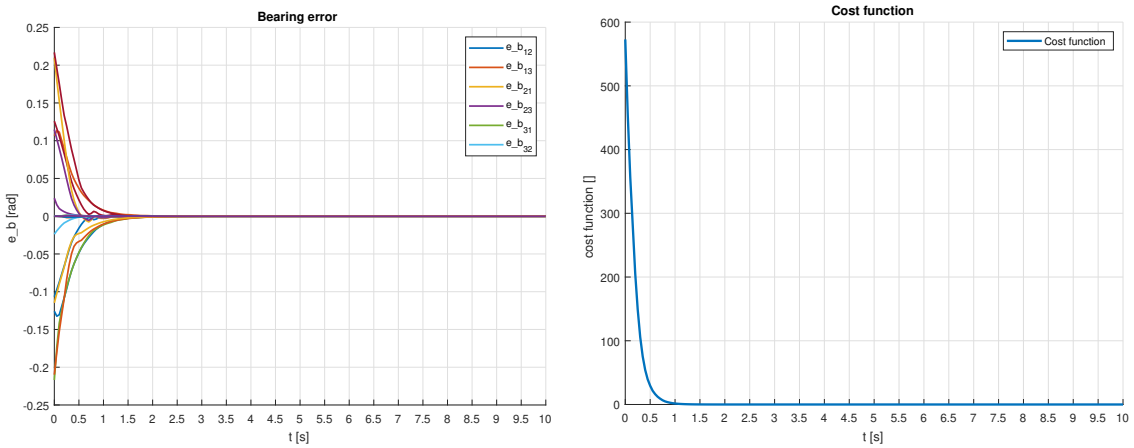


FIGURE 5.4: Bearing vector of each agent

5.1.3 Second cost function and second input constraint set



(a) Bearing error

(b) Cost function

FIGURE 5.5: Bearing error and cost function with Q = 1000 and R=1

The figures above show the similar behaviour in the previous experiment but in this case in the figure below we can notice that when the agents reached the formation and they didn't move any further but they was stationary in a specific point in order to maintain the desired bearing.

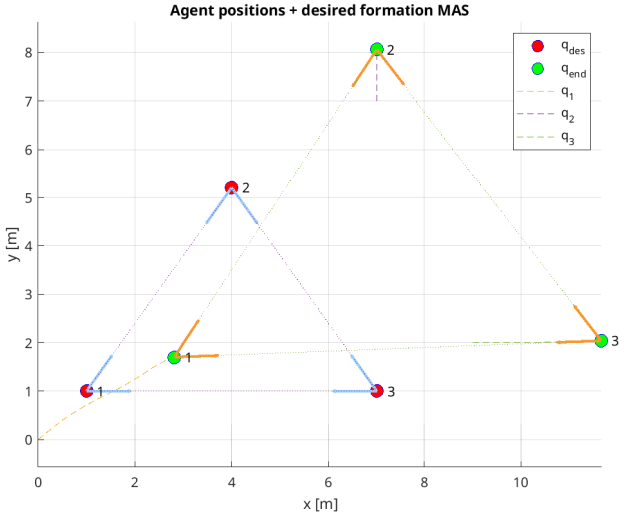


FIGURE 5.6: Trajectory and formation of MAS with Q = 1000 and R=1

The figure below shows us that the control input went to zero when the bearing error achieved the zero value or to close it:

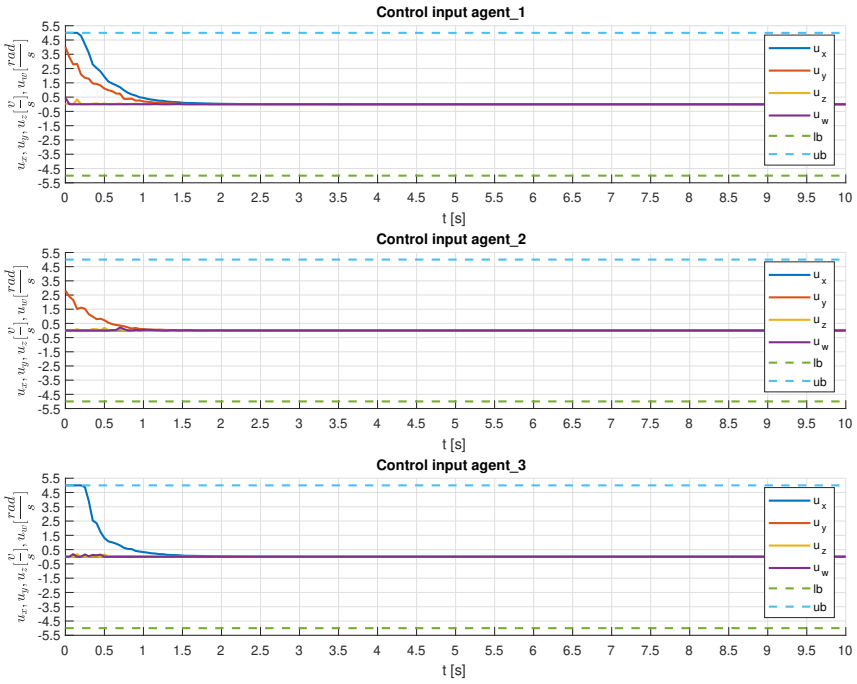


FIGURE 5.7: Control input for each agent with Q = 1000 and R=1

The figure below gives us a confirmation that the bearing behaviour doesn't have a negative effect on the new cost function and the new input constraint set.

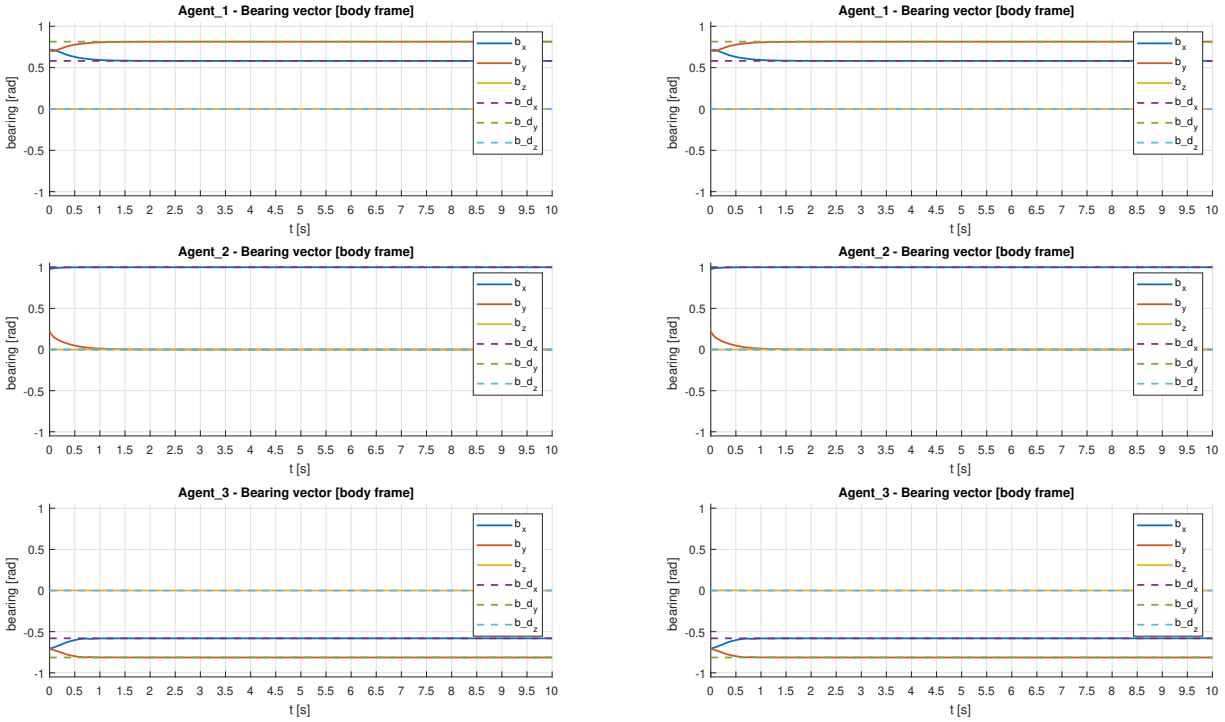


FIGURE 5.8: Bearing vector of each agent with Q = 1000 and R=1

5.2 Experiments NMPC - Centralized MAS model with thrust and angular velocities control

In this section, we will look at the experiments done before in Simulink and after in ROS/Gazebo. The idea is to prove that the model tested in Simulink works with similar behaviour also in the Gazebo environment. In general, when the Simulink and Gazebo report the similar behaviour, it means that the Simulink model was well designed and when we test the model also in the real world, we will not find any unexpected behaviour. In all experiments both Simulink and Gazebo, the input constraints that we used are reported in the Table A.4.

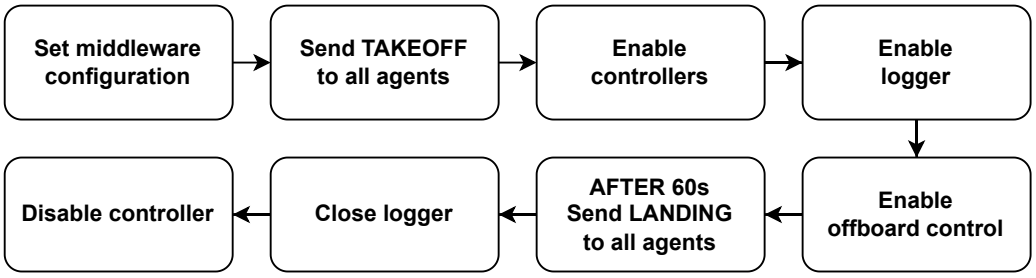


FIGURE 5.9: Flowchart of the procedure to do an experiment

For all experiments, we used the same desired bearing (Table A.1) and initial agent positions (Table A.2). In the Appendix B.2 contains some additional experiments with different initial agent positions. Also in the Appendix A.1, there are other tables that containing gains and settings used for these experiments. In the Figure 5.9 shows the procedure that we have used to simulate the system and retrieve the data from the Gazebo environment. We can notice in the figure above that the agents in the takeoff reach the 1mt height hence in the Gazebo simulations we decide to bring the agents at 2mt of height so we show the transient from one height to another.

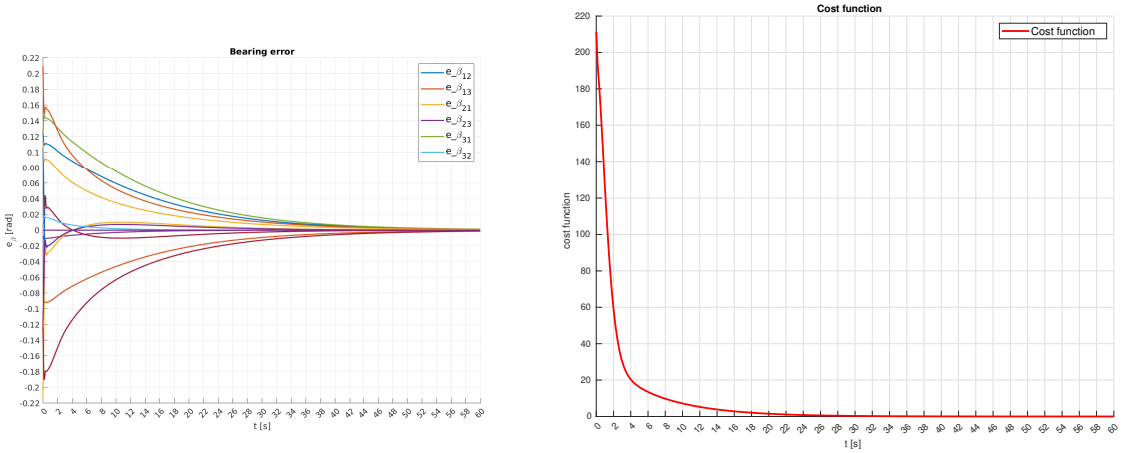
In addition, the experiment of the internal system model of MAS with 2 fixed agents and 1 free agent, which we described in the Section 3.2.2.1, is located in the Appendix B.2.1 because we have designed this model to understand the MATMPC limitations implemented on simulink.

5.2.1 Experiments NMPC without TMC

The following experiments aim to test the NMPC model (designed in the Section 3.2.2.3) without TMC, using the cost function cited in the Equation 3.86.

Simulink

The weights of the cost function are shown in the Table A.12.



(a) Bearing error (b) Cost function

FIGURE 5.10: Bearing error and cost function without TMC

The figures above represent the bearing error (left side) for each edge and the cost function (right side). We notice that the NMPC is able to minimize the cost function, reducing it to almost zero. The error is less than 0.05 rad of the interval and it is an acceptable error, moreover it is also confirmed by the figure below that represents the magnitude error:

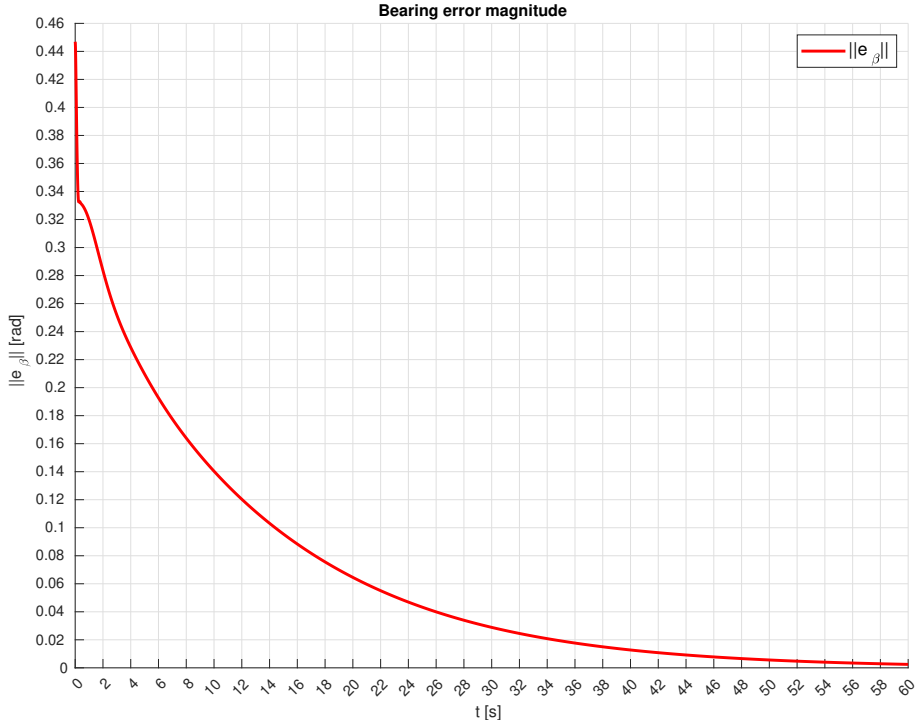


FIGURE 5.11: Bearing error magnitude

The following figure represents the comparison between the desired bearing and measured bearing for each the agent's edges:

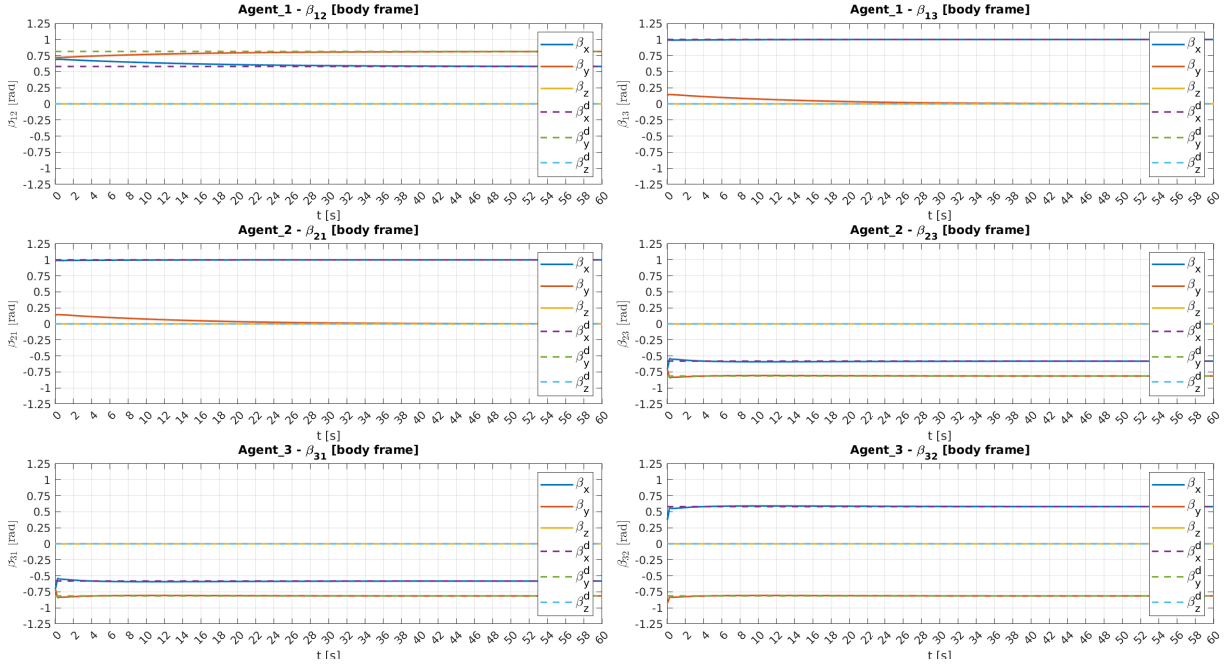


FIGURE 5.12: Bearing vector of each agent without TMC

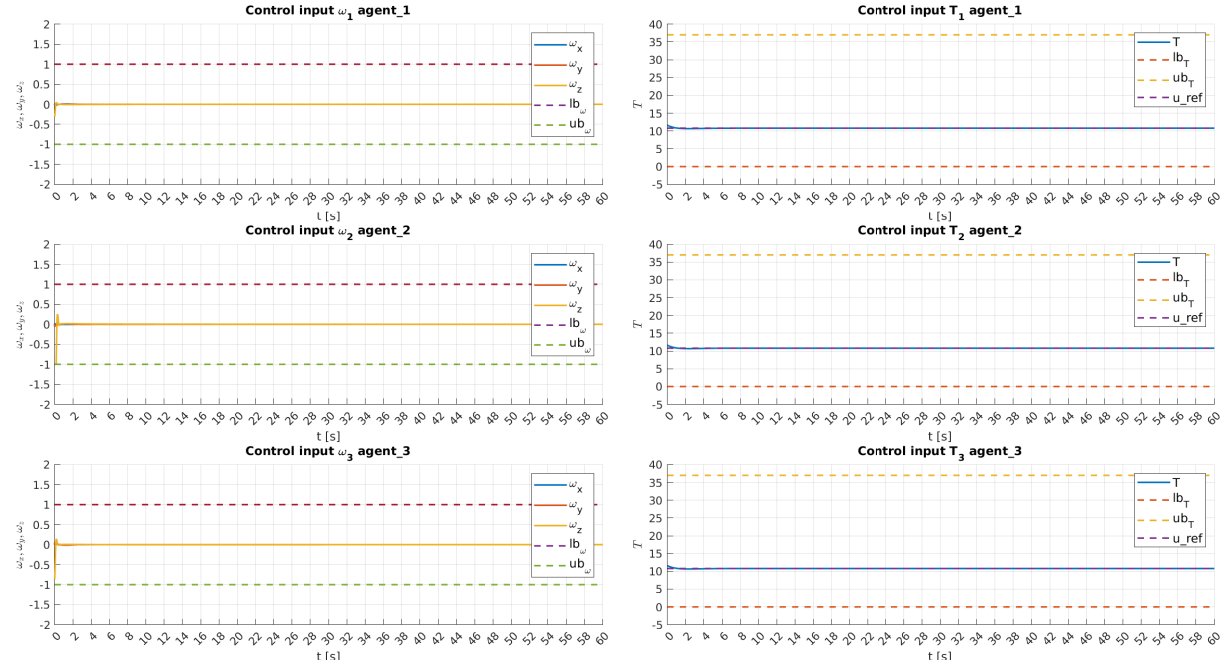


FIGURE 5.13: Control input for each agent without TMC

The Figure 5.13 shows the control input for each agent and we notice that the angular velocity is not zero. This is because the NMPC is trying to bring the bearing error to zero and the only way to do this is to change the yaw of the agents. Instead, the thrust is not zero because it has to compensate for the gravitational force. We can see the effect on linear velocities in the figure below:

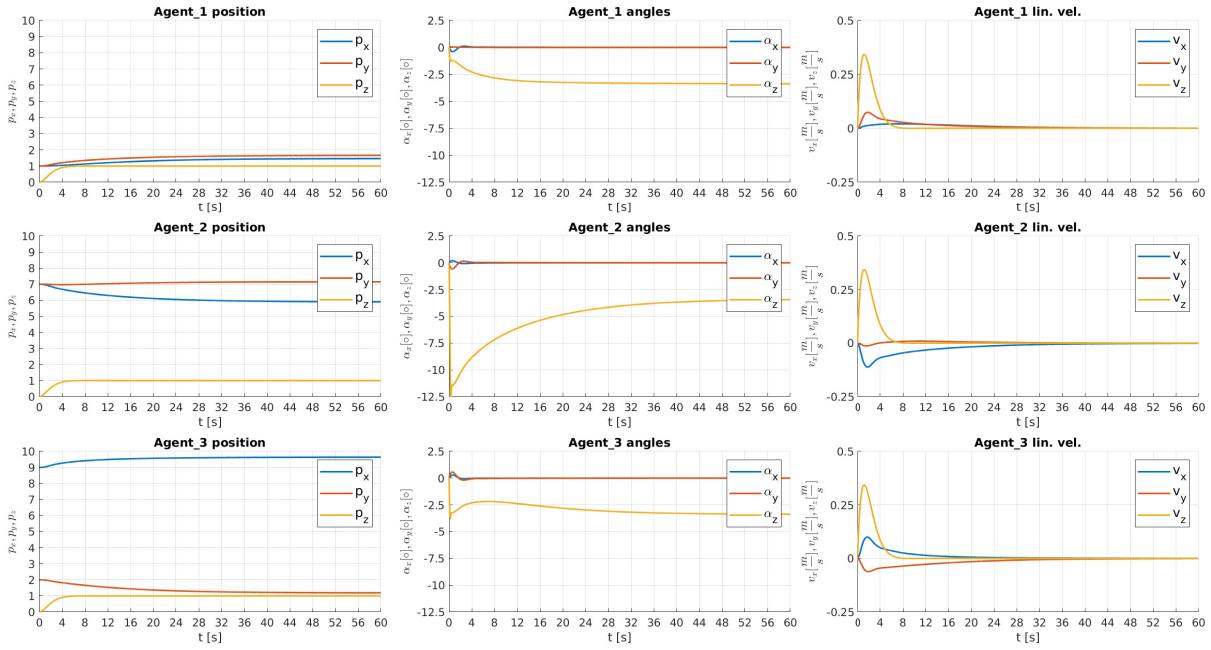


FIGURE 5.14: MAS state without TMC

The Figure 5.15 shows us the trajectory that the agents made to reach the desired formation through the control of the NMPC:

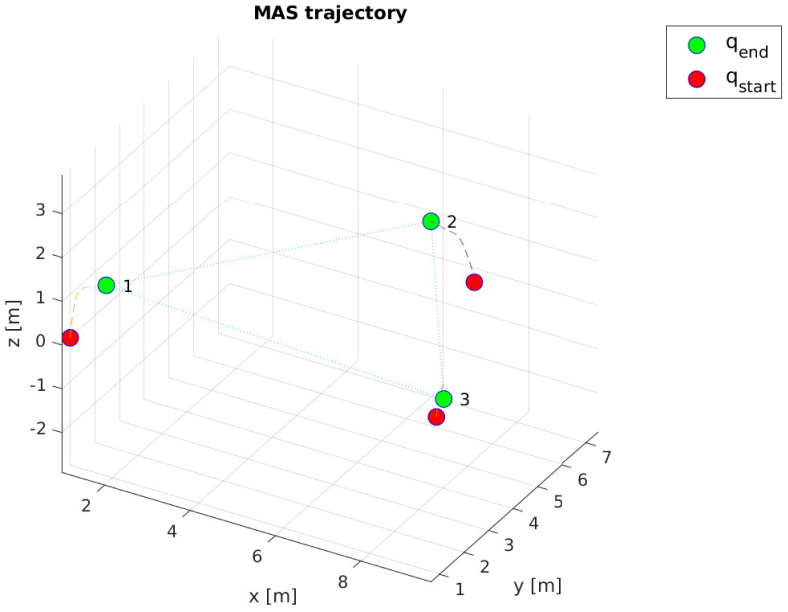
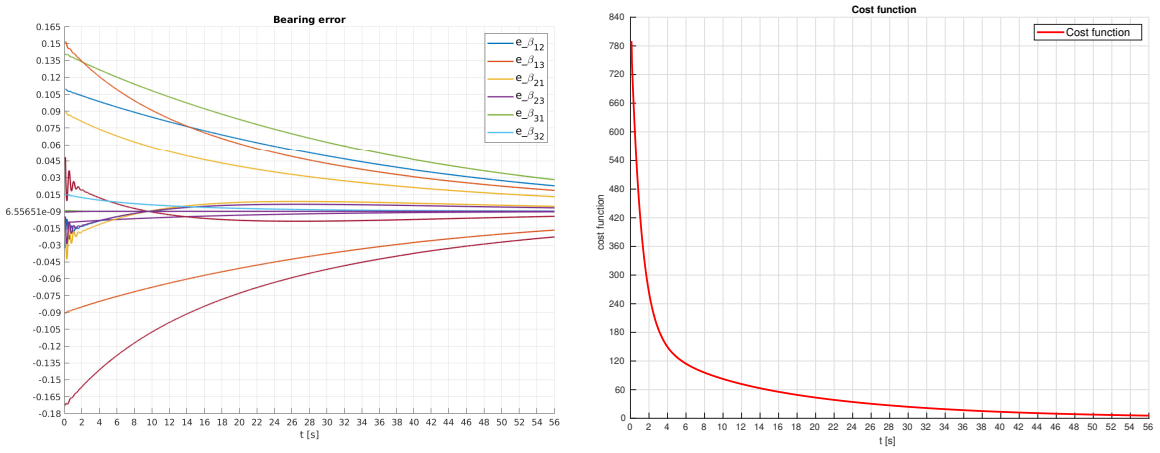


FIGURE 5.15: Trajectory and formation of MAS without TMC

ROS 2/Gazebo

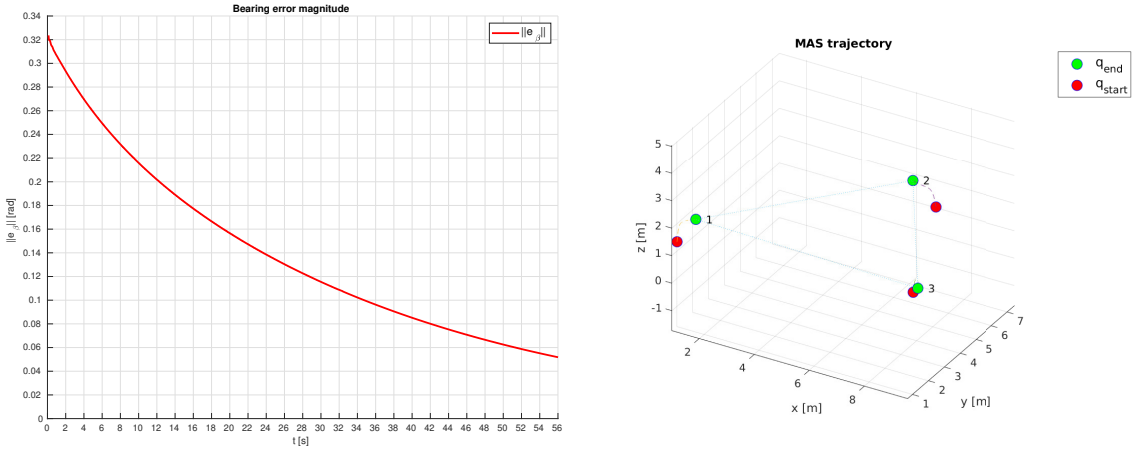
We now see the results obtained though the Gazebo environment with the same NMPC internal model system from the previous Simulink experiment. The weights of the cost function are shown in the Table A.12 instead of the initial positions are in the Table A.2 and the desired bearings are in the Table A.1.



(a) Bearing error (b) Cost function

FIGURE 5.16: Bearing error and cost function without TMC

In the figures above, we can see that the NMPC achieves the goal of minimizing the cost function with a bearing error magnitude less than ± 0.05 rad but it is slower than the above experiment on Simulink in the Figure 5.10(a) and the Figure 5.10(b).



(a) Bearing error magnitude without TMC (b) Trajectory and formation of MAS without TMC

FIGURE 5.17: Bearing error magnitude and MAS trajectory without TMC

In the Figure 5.17(a) remains below the value of 0.1 rad which is an acceptable value to take into account for the value for the desired formation is made. Instead, we notice that

the agents already start at an altitude of 1 mt and reach the 2mt altitude and this is what we explained at the beginning of the paragraph Section 5.2.

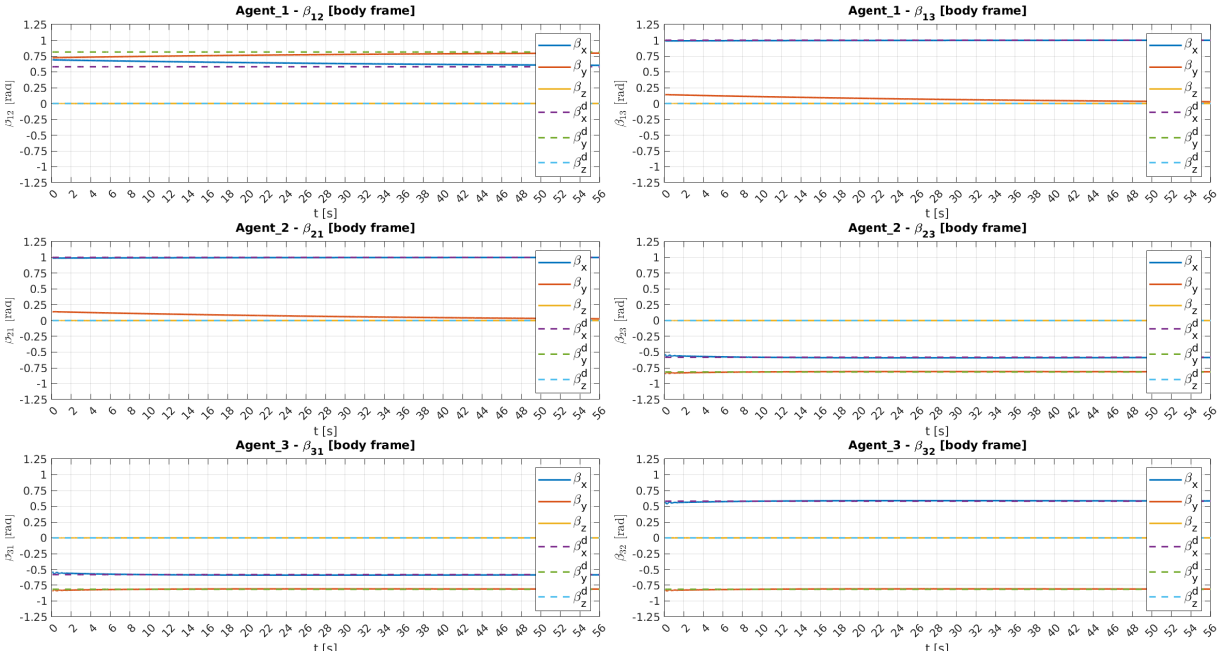


FIGURE 5.18: Bearing vector of each agent without TMC

We notice in Figure 5.18 that the bearings almost achieve the desired bearing value.

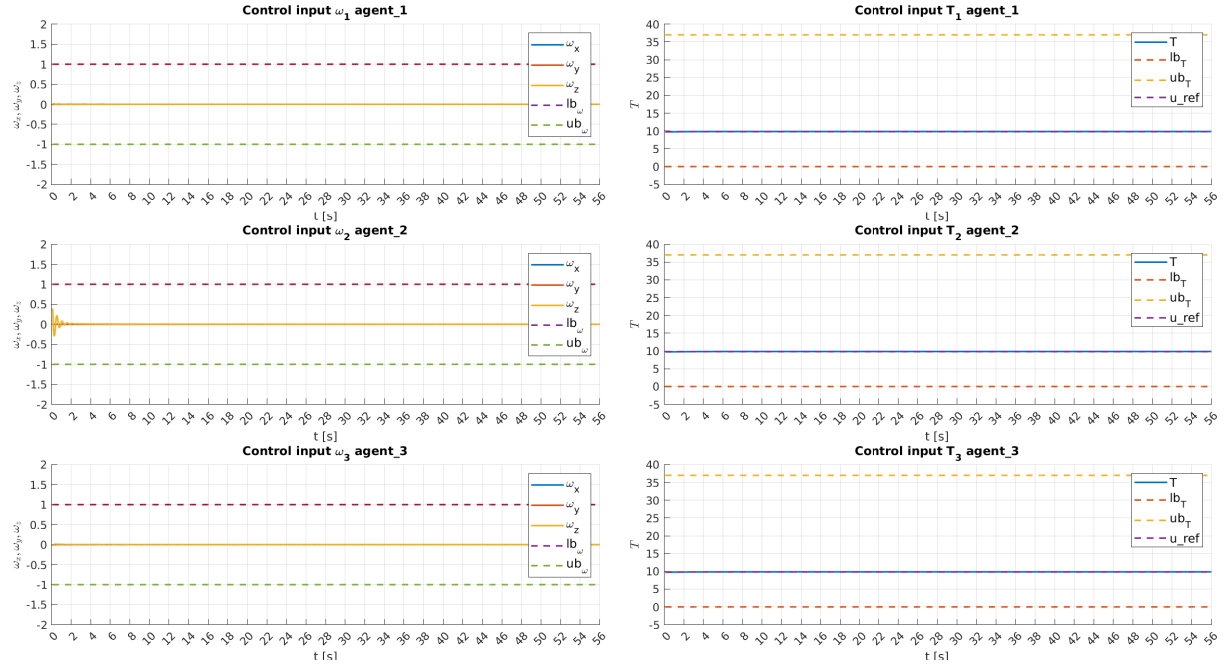


FIGURE 5.19: Control input for each agent without TMC

and we notice that in the Figure 5.19 the NMPC doesn't make big change to the thrust because the agents are already at 1 mt of altitude and the NMPC makes only small variations in angular velocities.

In the Figure 5.20 we notice that each agent achieves the 2 mt height (we ave wanted to show the transient from start height to end height) and the linear velocities go to zero after a first time transient:

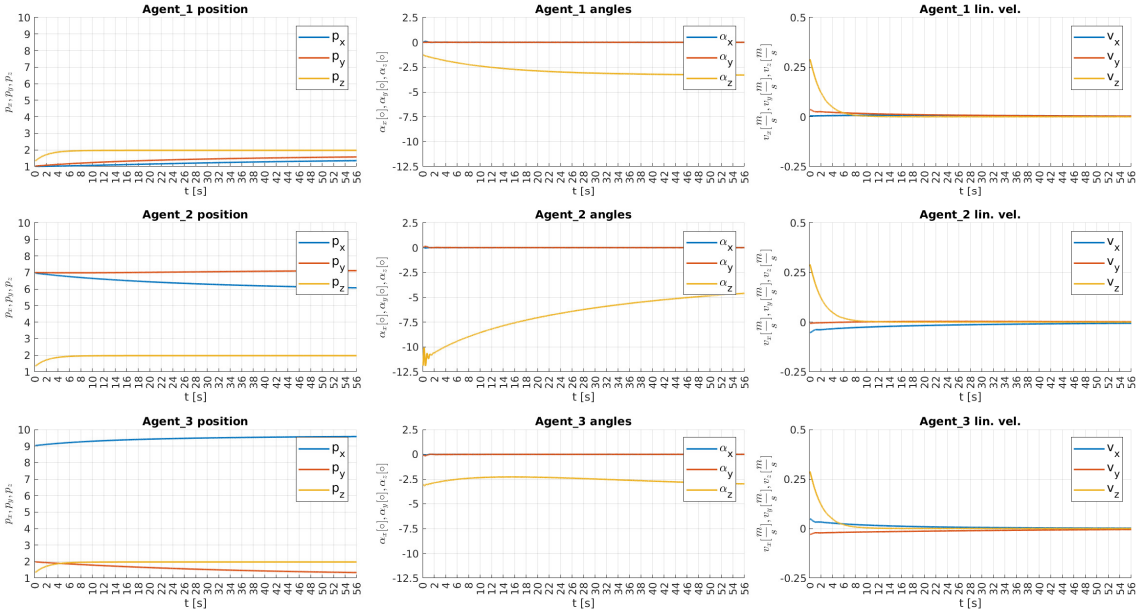


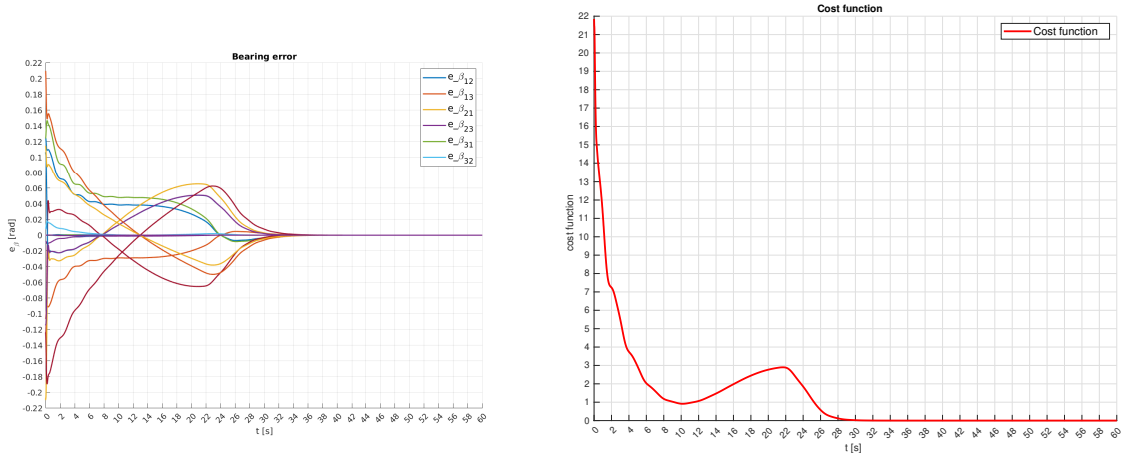
FIGURE 5.20: MAS state without TMC

5.2.2 Experiments NMPC with trivial motion controller (TMC)

The following experiments have the goal to test the NMPC model (designed in the Section 3.2.2.3) with TMC using the cost function cited in the Equation 3.89.

Simulink

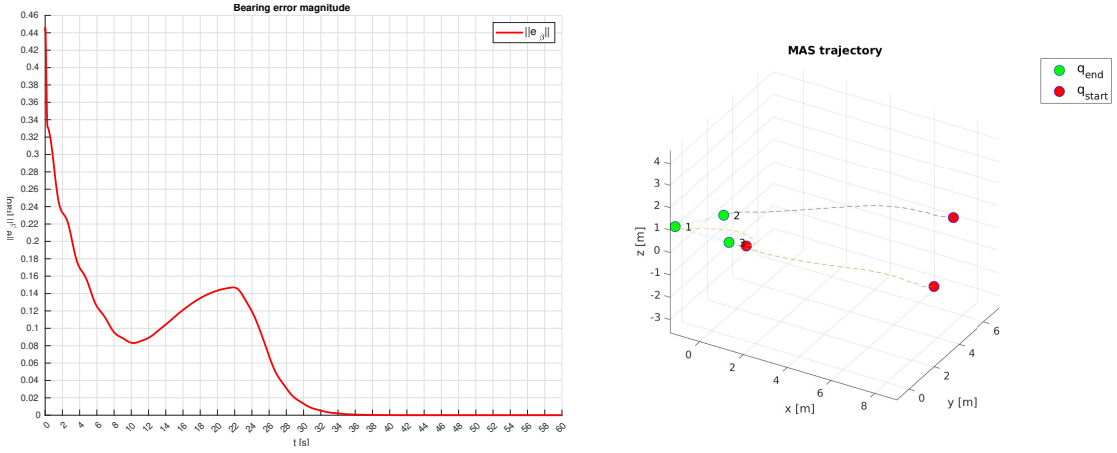
The Figure 5.21(a) shows us that the NMPC works very well with this TMC because the bearing errors are below the threshold of ± 0.05 and the cost function is went to zero with a nice transition time and it is depicted on the Figure 5.21(b).



(a) Bearing error (b) Cost function

FIGURE 5.21: Bearing error and cost function with TMC

In the figure above and in the Figure 5.22(a) show us that there is a pick this because the formation has not been created and a drone try to get closer to other two agents in opposite their direction but in a specific moment the NMPC under the control of TMC has to move the all the agents in the same direction. At that moment, the specific agent has to change direction and to align with the same direction of other two agents.



(a) Bearing magnitude (b) Trajectory and formation of MAS with TMC

FIGURE 5.22: Bearing error and MAS trajectory with TMC

The Figure 5.23 shows the bearings behaviour respect to the desired bearing and we notice that there is a good convergence.

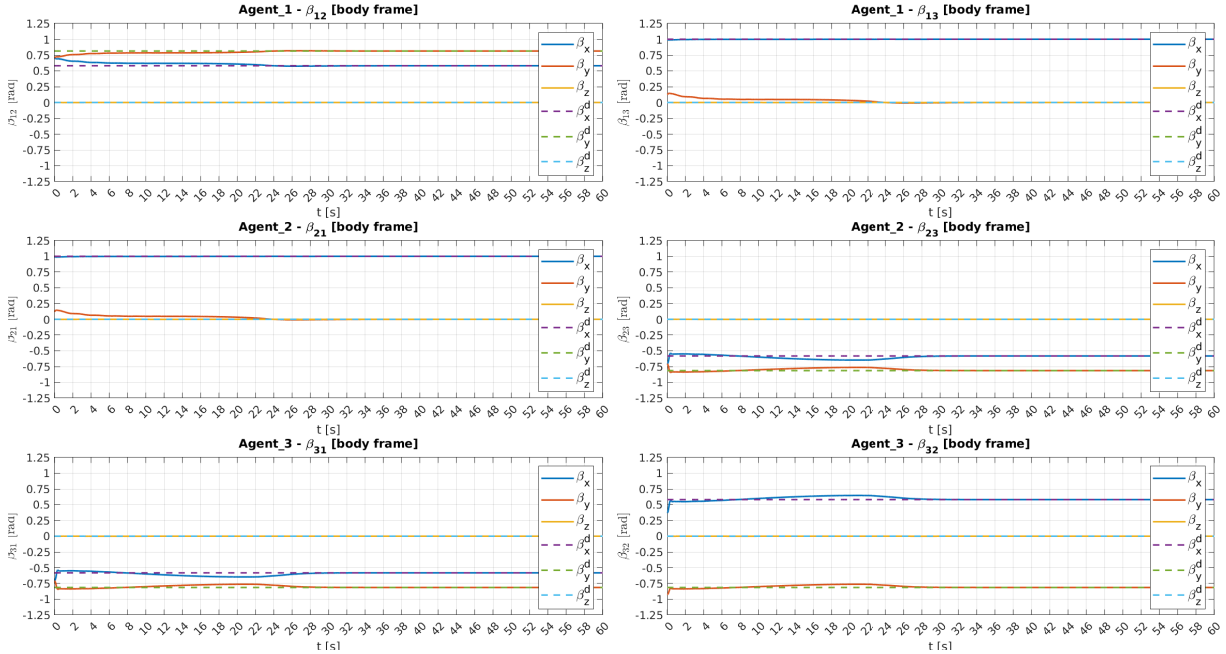


FIGURE 5.23: Bearing vector of each agent with TMC

In the figure below, we notice that in the initial transition there are some oscillations of the agents (we notice them on the agents' angles). This is because in the cost function there is not a term to manage the angles, so the NMPC is free to move the agents with aggressive manoeuvres, causing this behaviour.

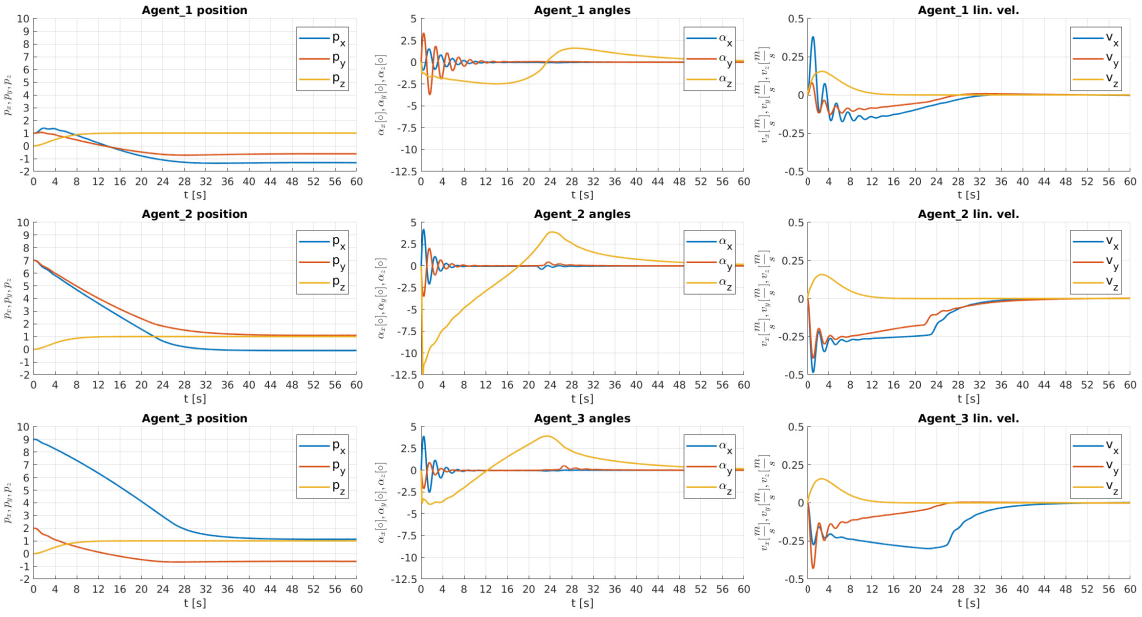
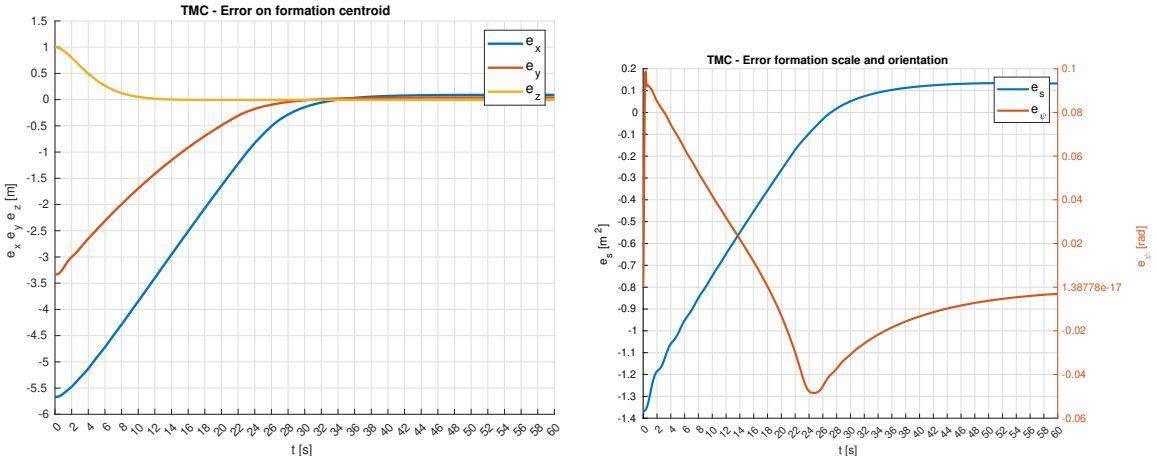


FIGURE 5.24: MAS state with TMC

In Figure 5.25 is depicted the error on barycenter position of the formation, the formation yaw and the formation scale. We notice without difficulty that the TMC is giving the right control input to NMPC to reduce the errors to (almost) zero:



(a) TMC PI error on formation centroid

(b) TMC PI error on formation scale

FIGURE 5.25: TMC errors on formation centroid and formation scale

In the Figure 5.26 is depicted the control input that the TMC generates ones instead, in the Figure 5.27 is depicted the NMPC reference linear velocities and angular velocities that are found through the second map expressed in the Equation 3.77.

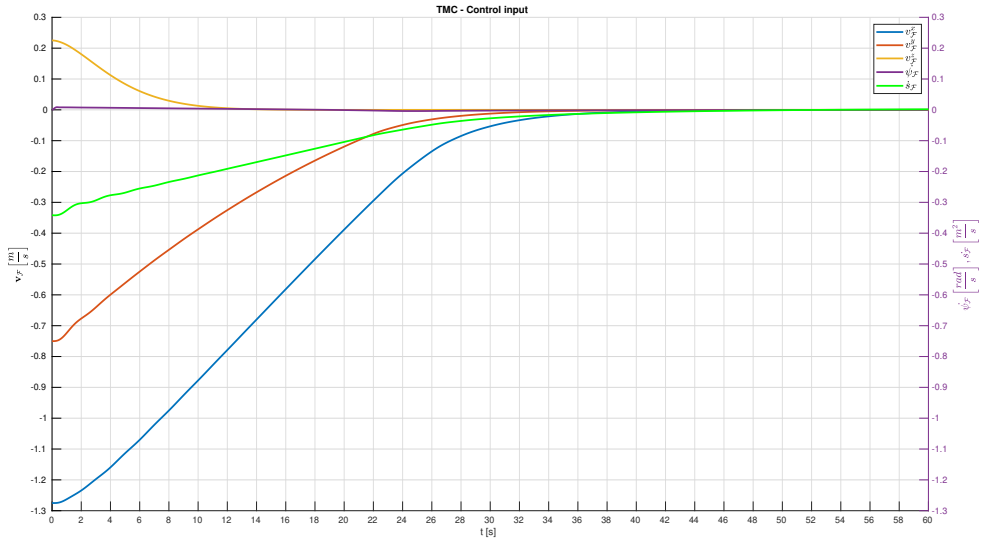


FIGURE 5.26: TMC on formation velocities

In the implementation, we set a saturation of ± 0.25 on the linear velocities of the agents. The motivation for this choice is that if the TMC is tuned too much aggressive the NMPC will not be able to handle these velocities, so the saturation is the better choice to solve the problem. We can see the effect on the left side of the Figure 5.27:

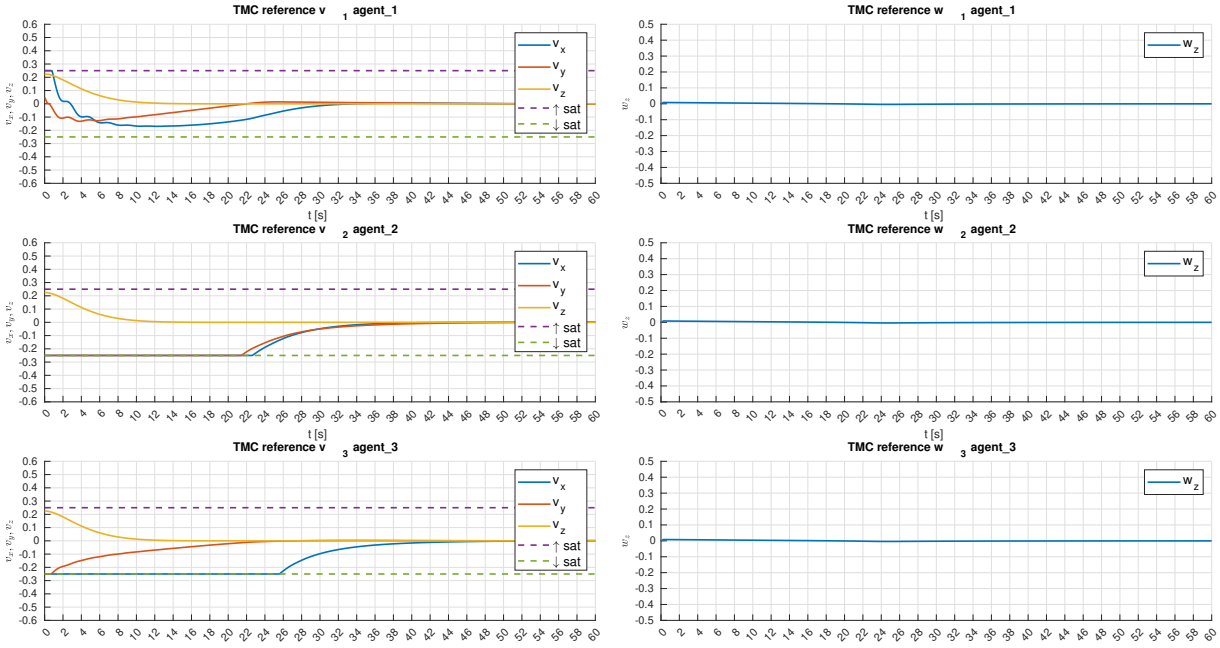
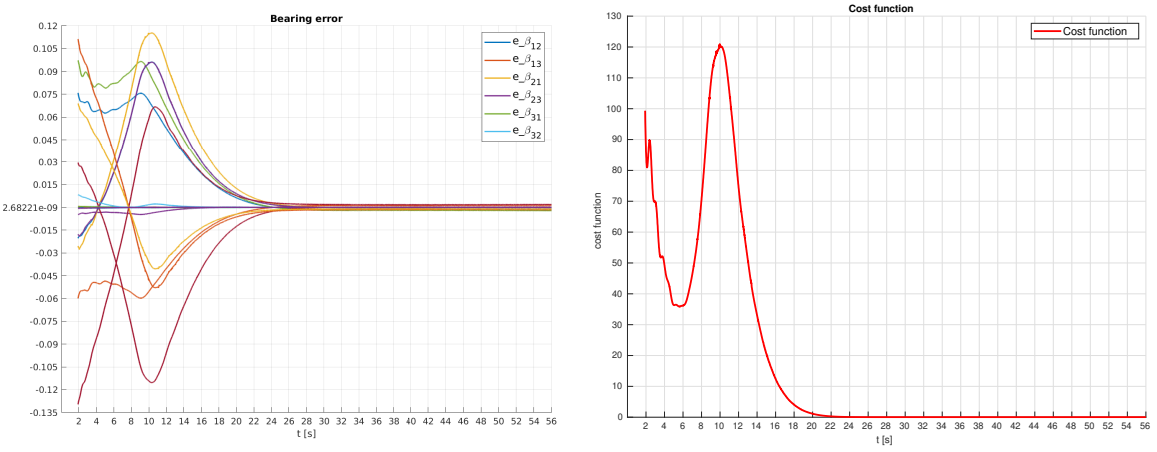


FIGURE 5.27: TMC on agent velocities

ROS 2/Gazebo

Now, we will see the result obtained from the Gazebo experiment of the same system that we tested above in the Matlab/Simulink environment. The weights of the cost function are shown in the Table A.12 and instead the initial positions are in the Table A.2 and the desired bearings are in the Table A.1. In the figure above, we notice that the bearing error after a transient time achieves a good value close to zero.



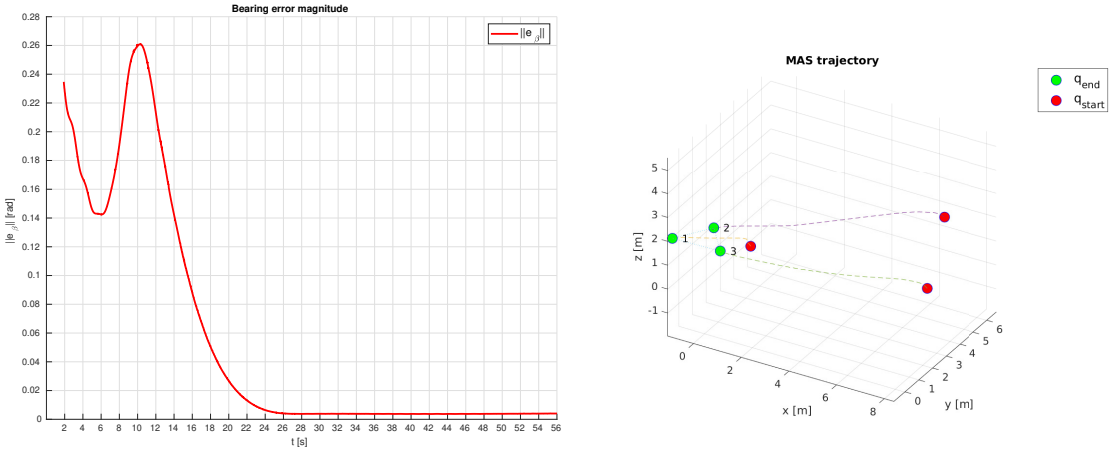
(a) Bearing error

(b) Cost function

FIGURE 5.28: Bearing error and cost function with TMC

The bearing error is better than the one obtained in the Simulink environment because the value is around ± 0.01 instead of before being around ± 0.05. The cost function has

a peak like in the Simulink experiment and then goes to (almost) zero. We notice that similar peak behaviour on trend of the bearing error magnitude in the Figure 5.29(a) and this happens because the cost function has a big value on the bearing term with respect to others hence it influences the cost function behaviour:



(a) Bearing magnitude

(b) Trajectory and formation of MAS with TMC

FIGURE 5.29: Bearing error and cost function with TMC

The following figure gives us further confirmation that the NMPC achieves the goal of creating the desired formation controlling the individual agents:

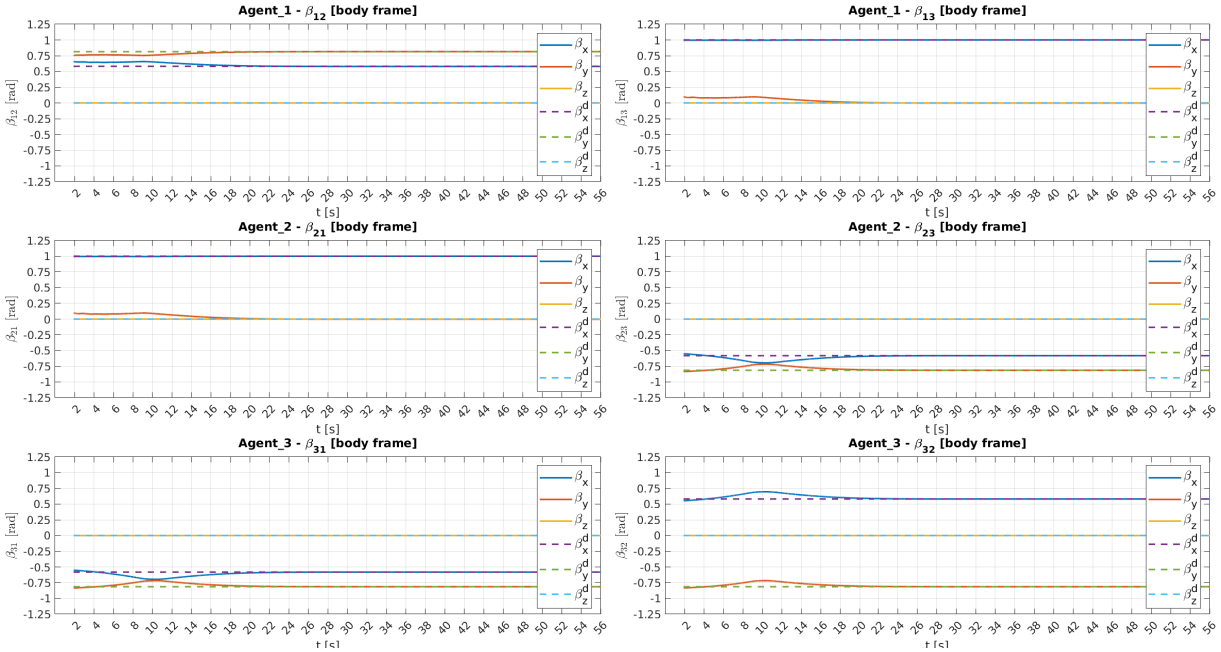


FIGURE 5.30: Bearing vector of each agent with TMC

The figure below shows the oscillation effect on angles caused by the NMPC control input (in the Figure 5.32)

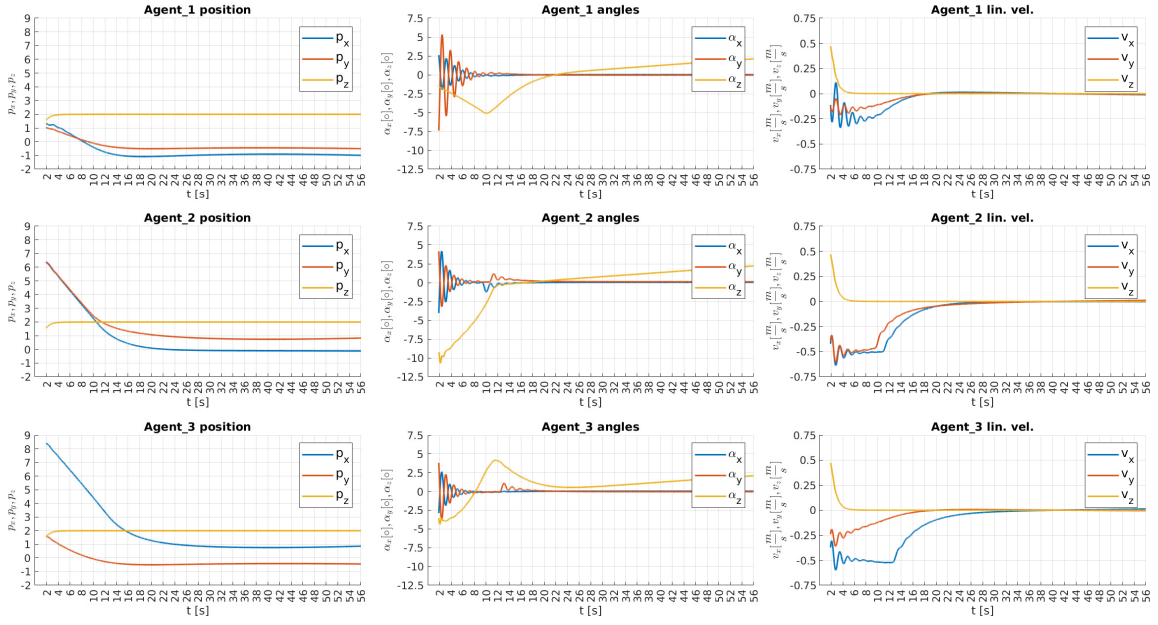


FIGURE 5.31: MAS state with TMC

In addition, we can notice that the linear velocity of the agents which are showed in the last columns on the right side is almost equal to the linear velocity obtained from the TMC in the Figure B.24.

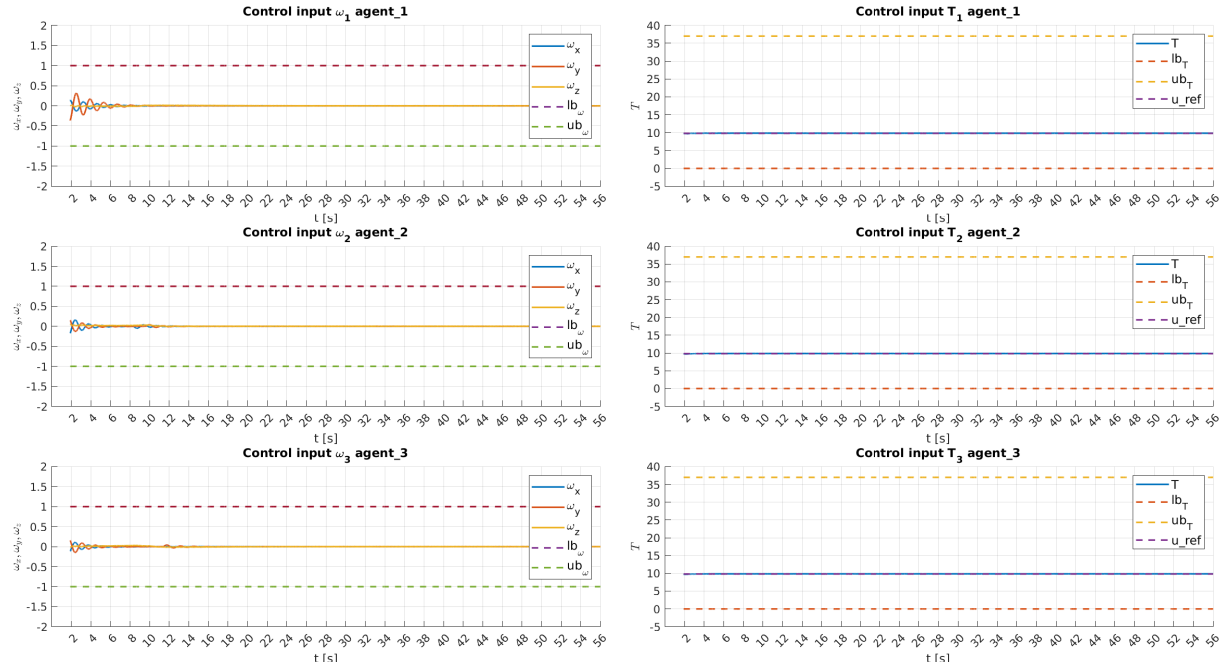


FIGURE 5.32: Control input for each agent with TMC

The behaviour of the TMC control input goes to (almost) zero (the figure below), this means that the errors on TMC reference are went to (almost) zero and this is true thanks

to the Figure 5.35(a) and the Figure 5.35(b). These show the error went to zero during the experiment.

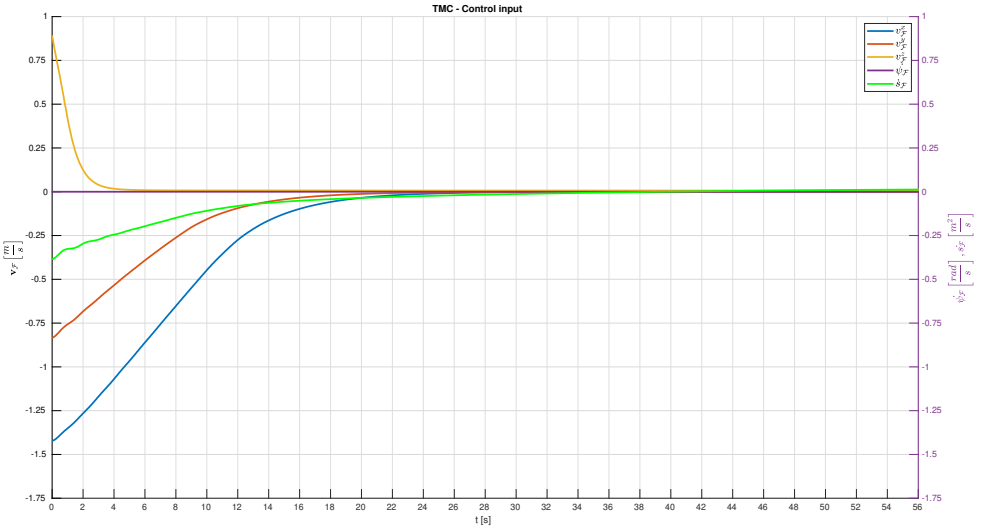


FIGURE 5.33: TMC control input

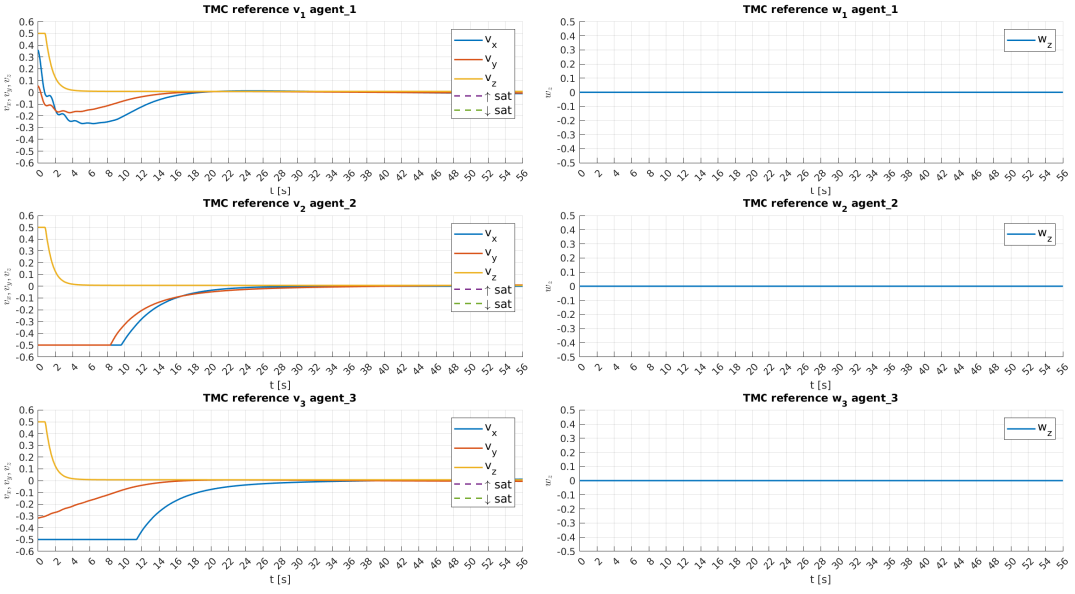
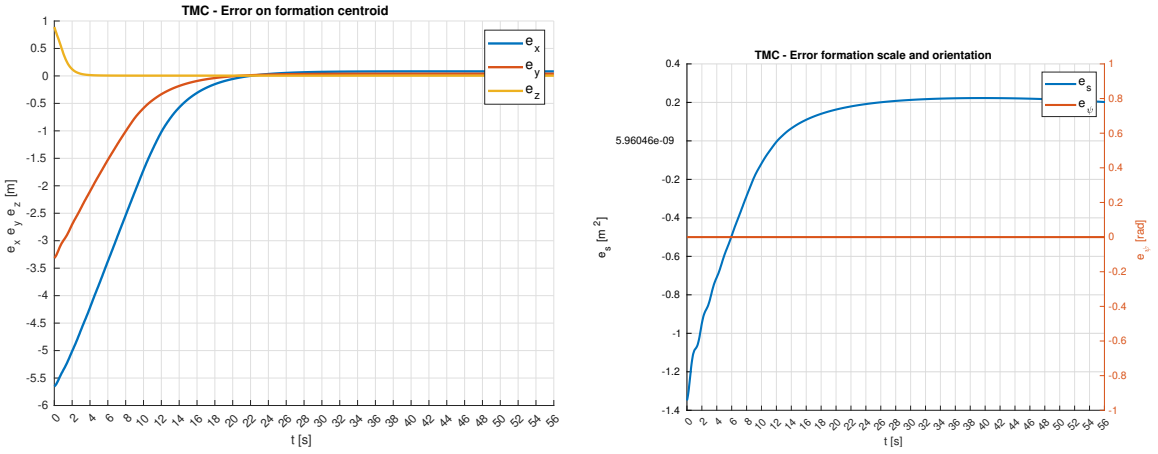


FIGURE 5.34: TMC PI error on formation velocities



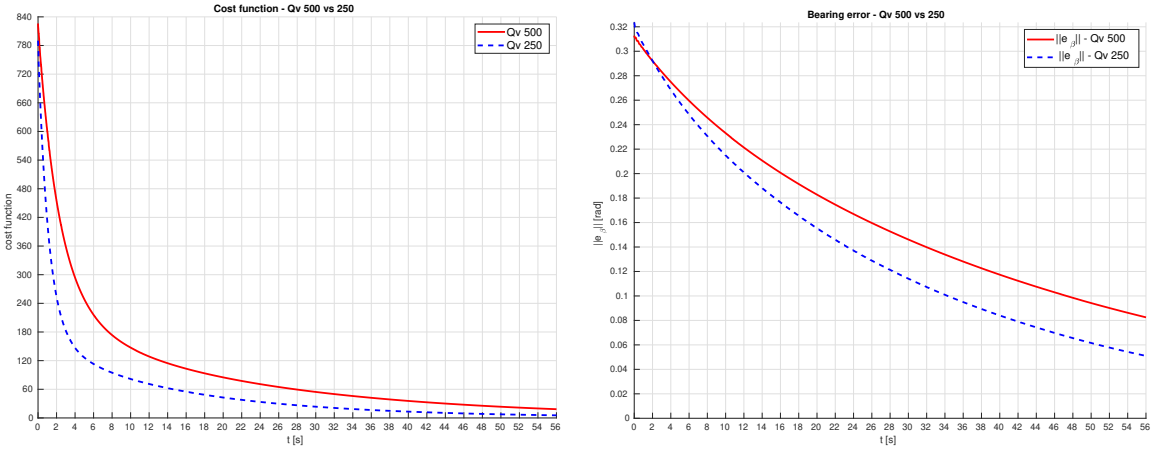
(a) TMC PI error on formation centroid

(b) TMC PI error on formation scale

FIGURE 5.35: Bearing error and cost function with TMC

5.2.3 Comparison between the cost function with Qv 500 vs Qv 250

In this section, we wanted to show the comparison of the cost function described in the Equation 3.86 with two difference values on $\mathbf{Q}_v$. In the figures below show that the $\mathbf{Q}_v$ lower than the other result, it is faster than one. This is because the NMPC gives less relevance to the TMC reference speeds and it can find a better control input to minimize the bearing error. We can remember that this is a MOO problem that we described in the Section 2.5.



(a) Comparison of two cost functions with different value of $\mathbf{Q}_v$

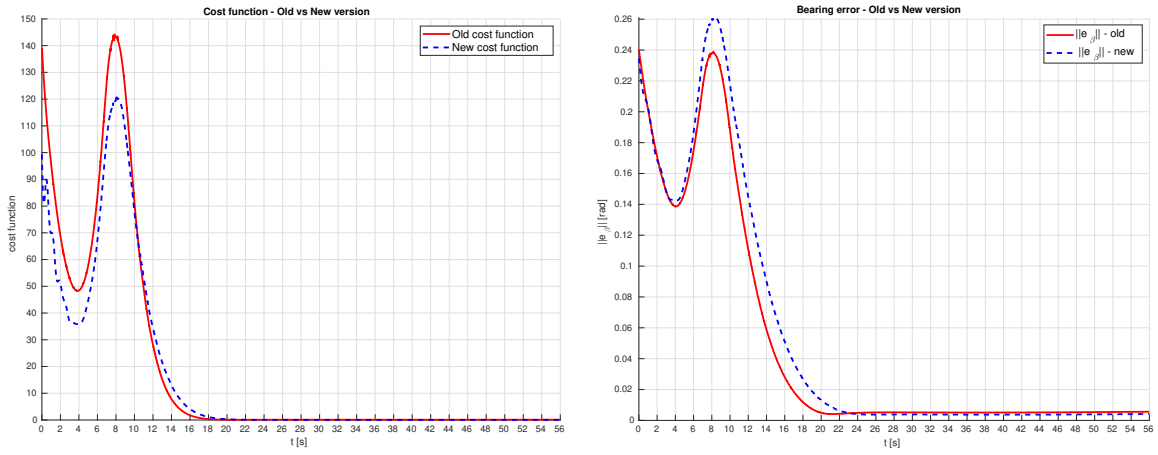
(b) Comparison of two bearing error magnitudes with different value of $\mathbf{Q}_v$

FIGURE 5.36: Comparison of two bearing error magnitudes and cost functions without TMC with old cost function

The weights of the cost function are shown in the Table A.12 instead of the initial positions are in the Table A.2 and the desired bearings are in the Table A.1.

5.2.4 Comparison between old cost function vs new cost function

Instead, in this section we wanted to highlight the difference between using the cost function defined in the Equation 3.86 (which we will call the "old cost function") and the cost function defined in the Equation 3.89 (which we will call the "new cost function). We notice in the Figure 5.37 that the cost functions behave similarly and both achieve the objective of minimizing the bearing error. This proves us that we were right when in deciding to remove the redundancy term with TMC that we explained in the Section 3.5.



(a) TMC PI error on formation centroid

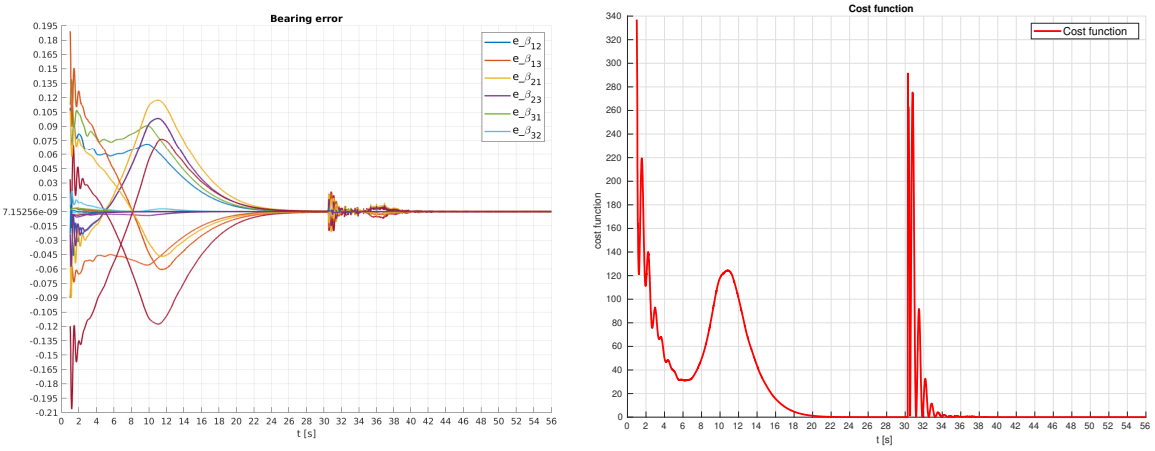
(b) TMC PI error on formation scale

FIGURE 5.37: Comparison of two bearing error magnitudes and cost functions with different cost functions

The weights of the cost function are shown in the Table A.12 instead the initial positions are in the Table A.2 and the desired bearings are in the Table A.1.

5.2.5 Change TMC reference setpoint during the flight

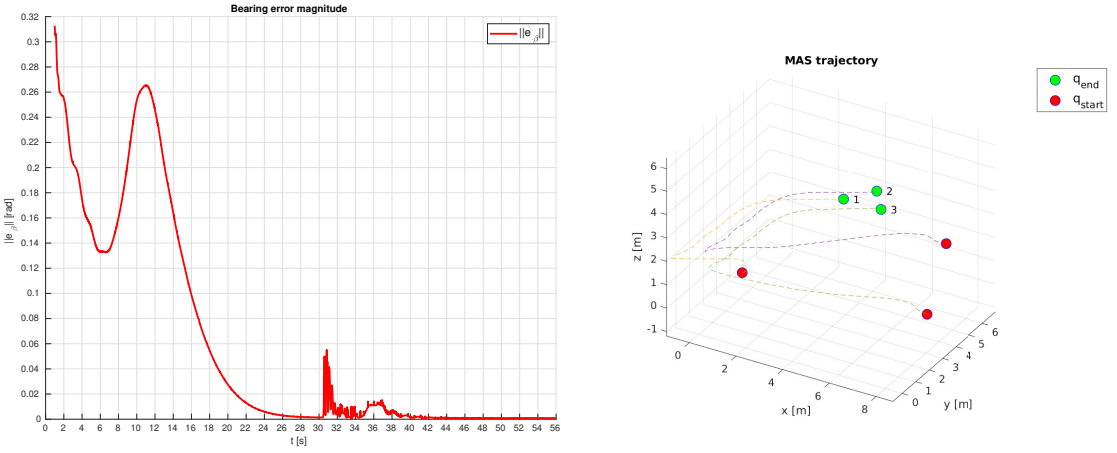
In this experiment, we want to test if the trivial motions theory doesn't destroy a formation after it has been created. The idea is to change the TMC reference setpoint after 30s of the simulation. The weights of the cost function are shown in the Table A.12 instead the initial positions are in the Table A.2 and the desired bearings are in the Table A.1.



(a) Bearing error (b) Cost function

FIGURE 5.38: Bearing error and cost function with change of TMC reference setpoint

As we can see, the peak that there is in the Figure 5.38(b) proves that there has been a change on the setpoint in hence an increase of the cost function value. Nevertheless, in the Figure 5.38(a) there are little perturbations but this proves that the TMC doesn't destroy the formation.



(a) Bearing magnitude (b) Trajectory and formation of MAS with TMC

FIGURE 5.39: Bearing error and MAS trajectory with change of TMC reference setpoint

The same behaviour we can be seen it in the figure above, where the perturbations are very small:

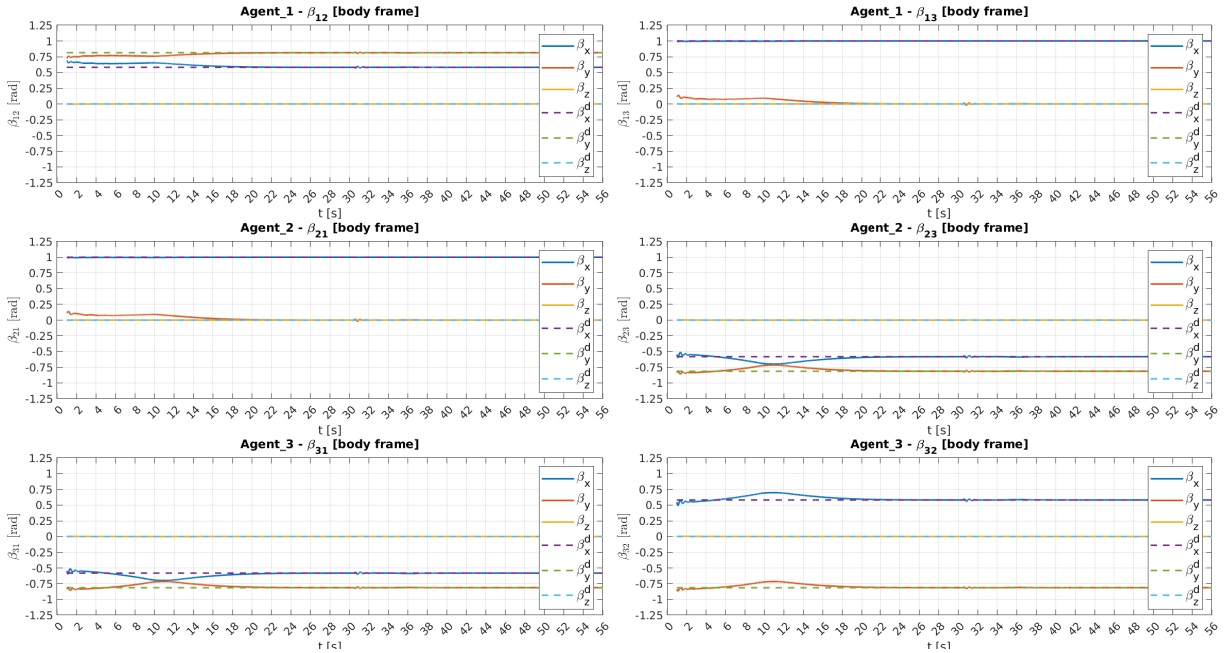


FIGURE 5.40: Bearing vector of each agent with change of TMC reference setpoint

The change of setpoint causes a change in the TMC control input (which is shown in the figure below) that has effects on the NMPC control input in the Figure 5.42

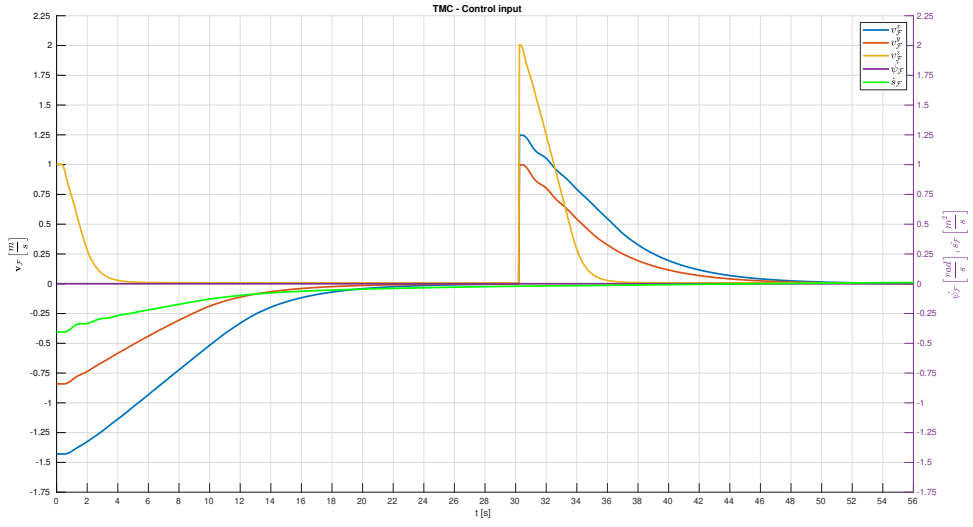


FIGURE 5.41: TMC control input

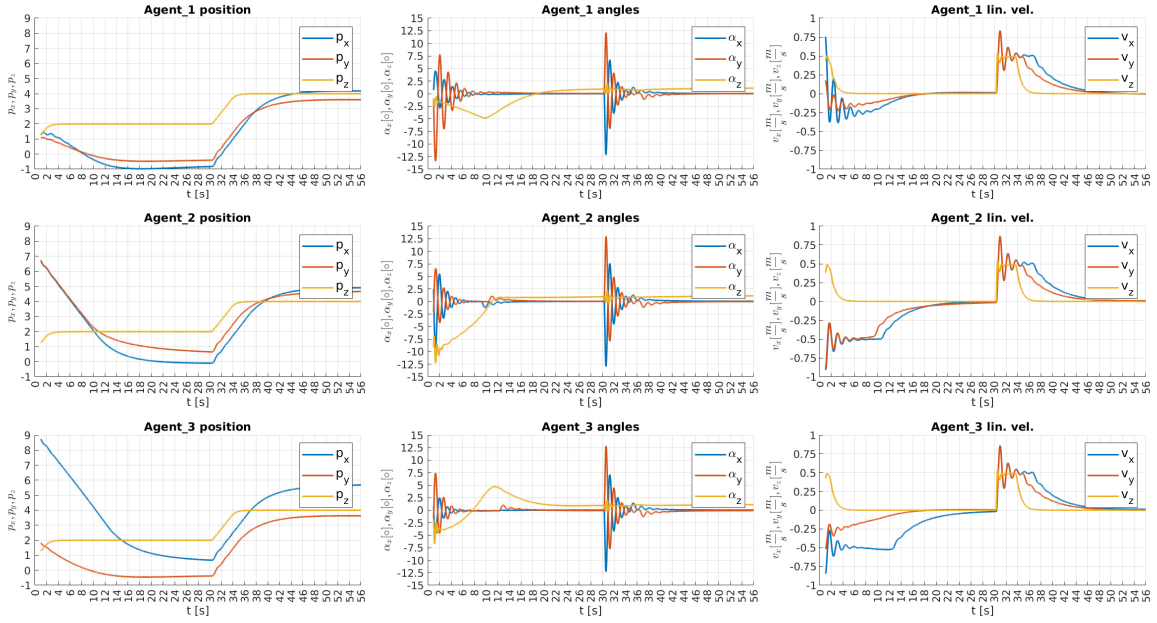


FIGURE 5.42: MAS state with a change of TMC reference setpoint

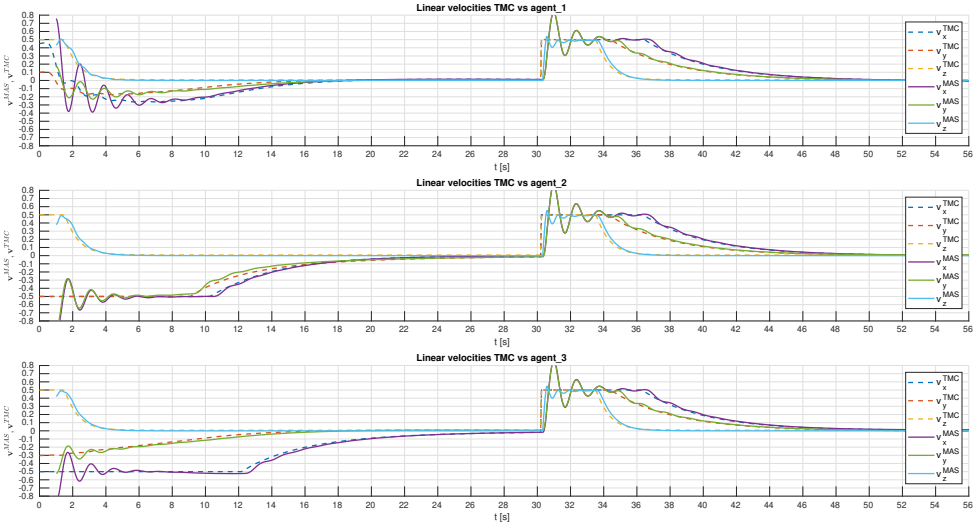
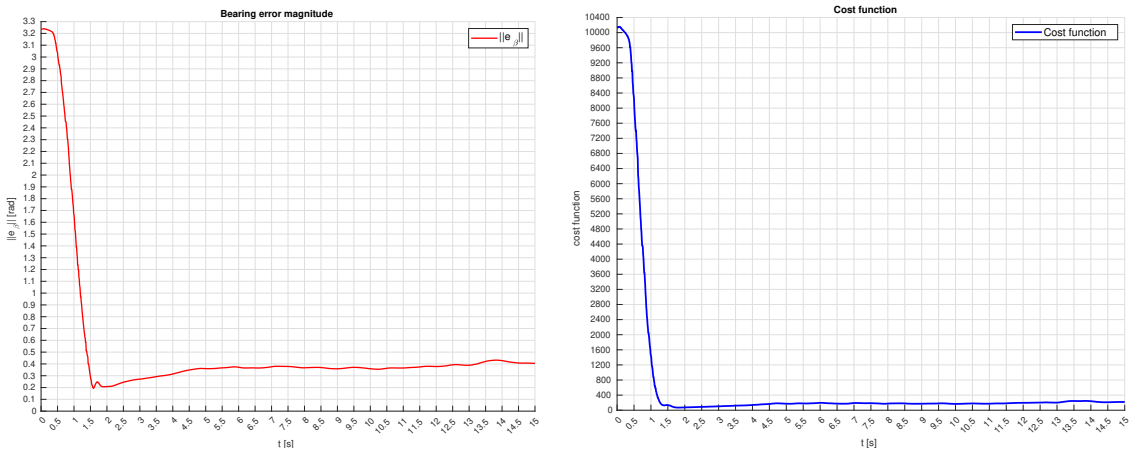


FIGURE 5.43: Comparison between TMC reference velocities and agents velocities

The Figure 5.43 shows a comparison between TMC reference linear velocities and linear velocities of the agents. This figure gives us an important insight into the trivial motions. In the Section 2.3.2 we said that the trivial motions are inside the null space of the bearing rigidity matrix and the TMC has created this reference of trivial motions. However, we notice in the figure that the NMPC doesn't follow the TCM reference velocities perfectly and this creates a small perturbation on the bearing information when there is the change the setpoint. Therefore, this means that the NMPC has generated a control input that causes the agents to reach a velocity that is not in the null space of the bearing rigidity matrix.

5.2.6 Experiments in the laboratory

The pipeline was designed with the idea of doing some experiments in the laboratory. We did several tests but we had big problems with the MoCap system, drones and other things because there was not the time to make a good setup and a good tune of the parameters. The Figure 5.44(b) and the Figure 5.44(a) show the results that we obtained from the last experiment that we did in the laboratory.



(a) Bearing error

(b) Cost function

FIGURE 5.44: Bearing error and cost function with TMC

In the Figure 5.44(b) shows the cost function doesn't go to zero but it keeps a value around 350 that the NMPC is not able to minimize it. We find the cause in the Figure 5.44(a) where it shows us the bearing magnitude error, which doesn't decrease more than 0.4 rad which is equal to $\approx 23°$. This is an unacceptable value because it means that the NMPC is not able to achieve the desired formation.

The causes of this failure to achieve the goal are:

- incorrect tuning of the parameters on the NMPC and TMC
- incorrect physical value of quadrotors

In addition, in the columns on the agent positions of the Figure 5.45 show that the agents start from positions of 1m (it is the value of the desired altitude) but after they get slowly off. This is a strange behaviour because at that moment, the NMPC should improve only the bearing. This fact confirms the likely second point in the list above and in this case with wrong weights, a MOO problem arose which we explained in the Section 2.5.2.

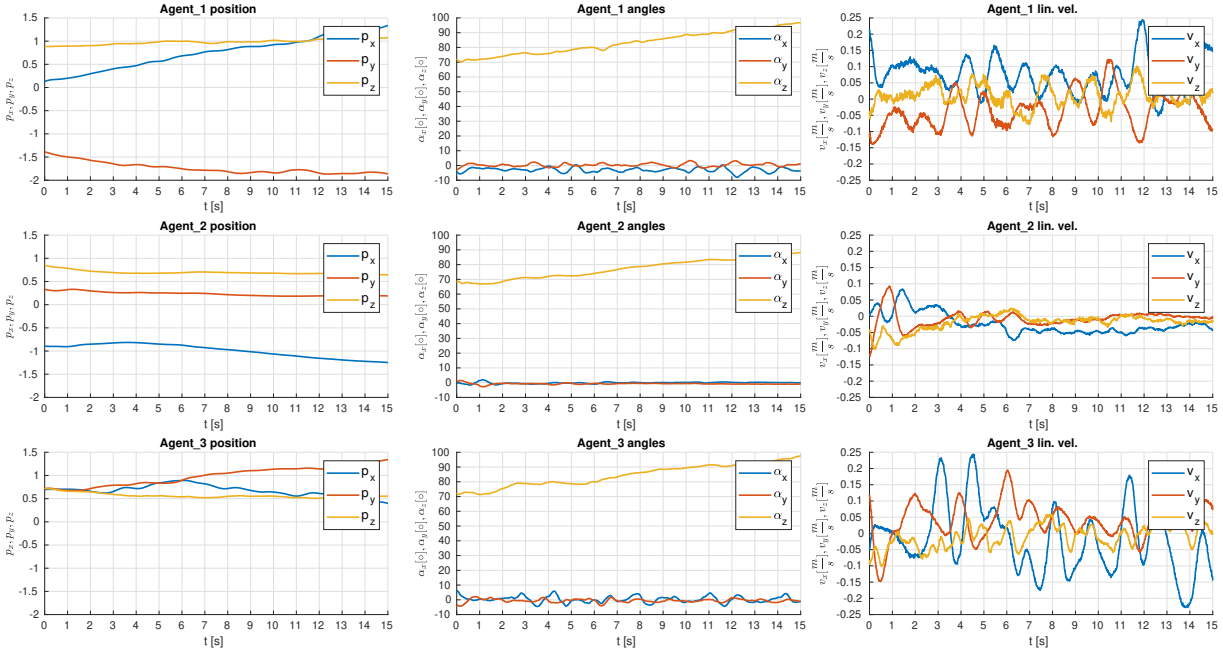


FIGURE 5.45: MAS state with TMC

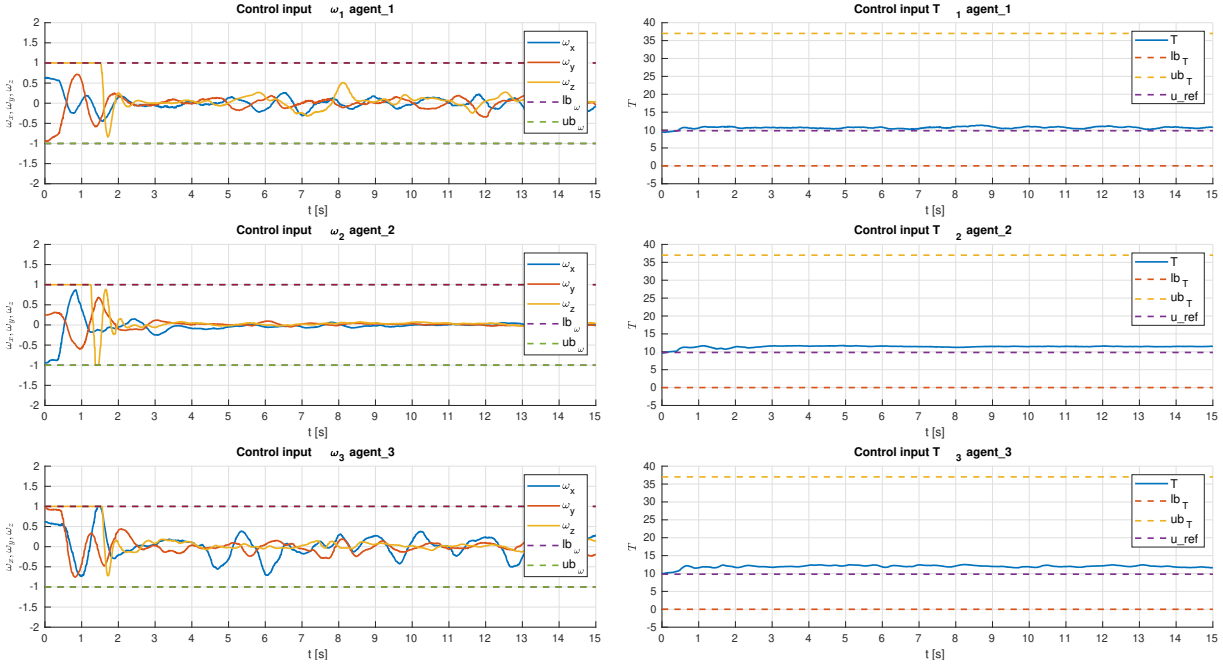


FIGURE 5.46: Control input for each agent with TMC

If we compare the TMC reference velocities (shown in the Figure 5.47) and the linear velocities of the agents (shown in the Figure 5.45), we will notice that the NMPC is not able to follow the TMC reference and this causes that the formation state error (shown in the Figure 5.48) will not go to zero.

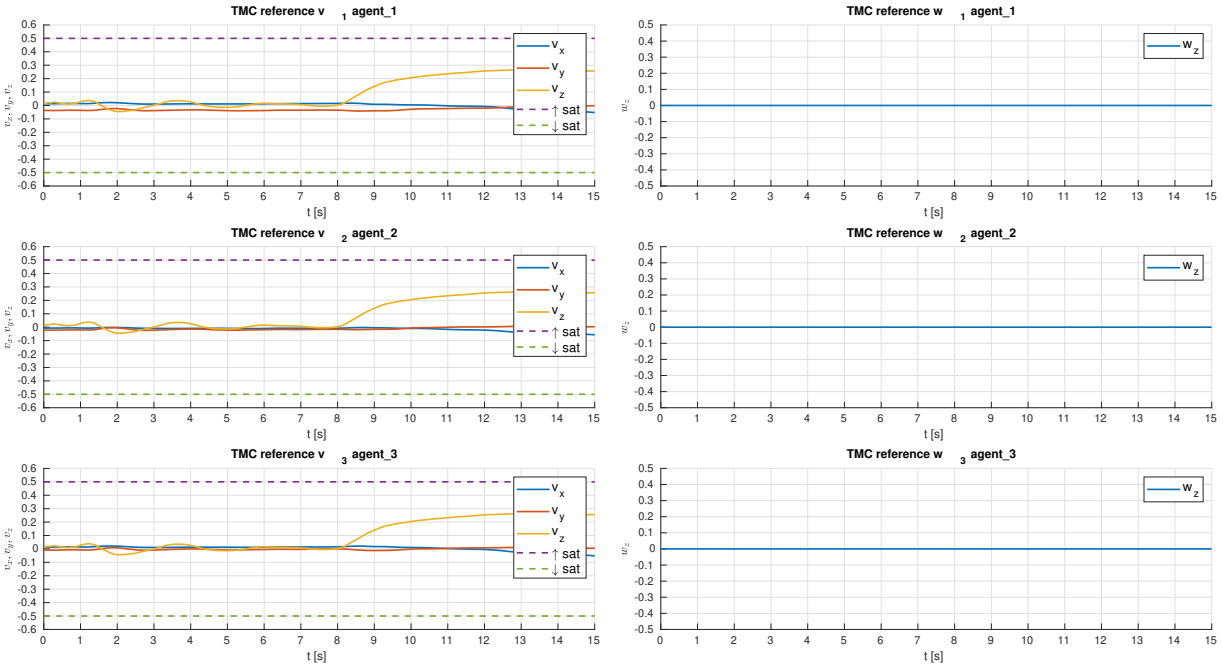
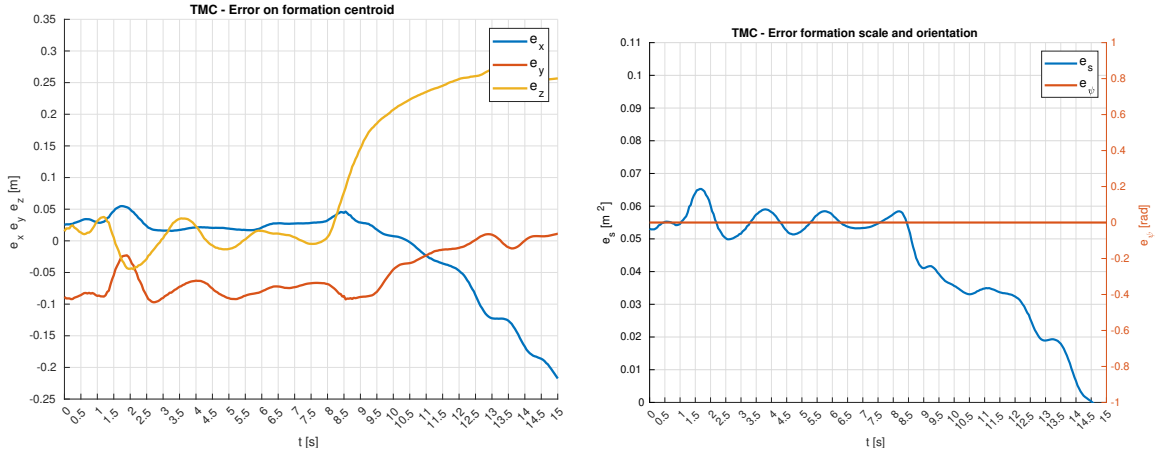


FIGURE 5.47: TMC PI error on formation velocities



(a) TMC PI error on formation centroid

(b) TMC PI error on formation scale

FIGURE 5.48: Bearing error and cost function with TMC

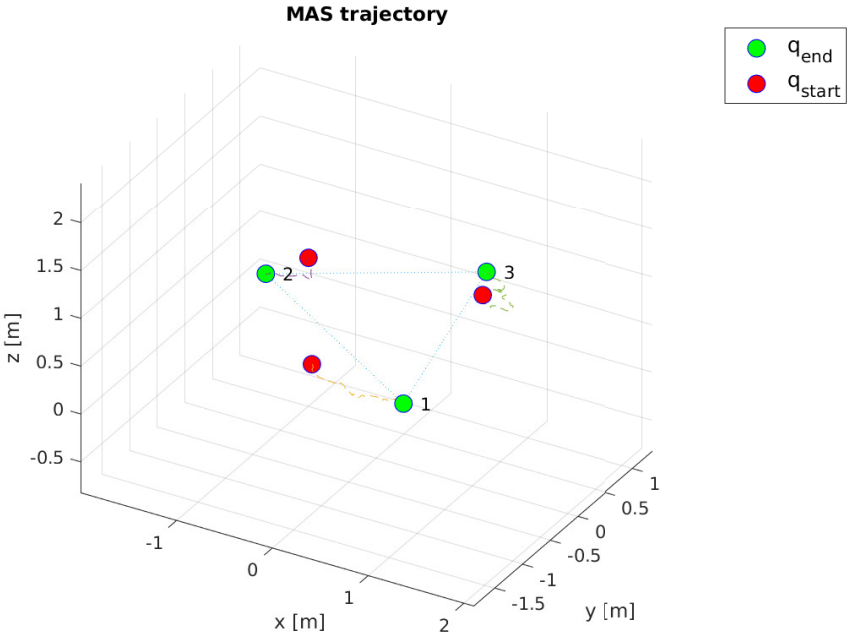


FIGURE 5.49: Trajectory and formation of MAS with TMC

Chapter 6

Conclusions

The contribution of this thesis is to find a way to design an internal system model for NMPC with embedded the bearing rigidity theory because the goal was to make formation control with NMPC and bring it into a real environment. We investigate the advantages and disadvantages of different MPC architectures, such as centralized, distributed, and decentralized architecture. We have designed a control pipeline to manage all implementation aspects with Gazebo, PX4, and Simulink with important features such as adaptability, configurability, and managing different architectures.

We began to design a model with simplified quadrotor dynamics and we implemented it on Simulink and we checked if the concept of the model was correct. After that, we designed new models with complex quadrotor dynamics with different goals. We implemented them on Simulink. When we saw that they worked on Simulink, we designed a control pipeline in order to do the test on the Gazebo with PX4 architecture. We have tested the pipeline with the NMPC models and we obtained satisfactory results on all gazebo results.

We have tried different cost functions with different goals and we implemented a TMC in order to manage the formation and reduce the terms in the cost function. During the study on the cost functions, we faced the problem of MOO problem and we studied how the NMPC can work with it and which solutions were better for our study case.

The thesis has shown the complexity of using the Nonlinear Model Predictive Control with centralized bearing-only control and also it has shown its validity because if the NMPC is complex to tune after finding a good set of parameters, the results that it brings are very interesting and it compensates for the imprecision of the model without problems. In addition, the thesis has shown also the big problem of the centralized NMPC namely it consumes many power resources to compute a feasible solution and it is not feasible to place it on an embedded computer such as a Raspberry PI.

6.1 Future improvements

As we saw in the last section of the Chapter 5, the laboratory tests didn't go well because of a no-good set of parameters and other little problems on the whole setup. The future improvements of this thesis should be:

- bring the whole control pipeline in the laboratory and find good parameters set in order to validate our hypothesis about why the laboratory tests have gone wrong
- implements the bearing-only formation control with NMPC in decentralized architecture and places an NMPC with its subproblems on each specific agent. So, we don't need to have a power-computer to compute the feasible solution anymore
- remove the world frame assumptions in order to see how the control pipeline behaves and if we introduce a new delay caused by calculating some stuff.
- model the delay on the network to understand how the NMPC behaves

Appendix A

An Appendix numerical values

A.1 Gain and settings tables for the experiments

Bearing	Value	Bearing	Value	Bearing	Value
$\boldsymbol{\beta}_{12}$	$\begin{bmatrix} 0.5812 \\ 0.8137 \\ 0 \end{bmatrix}$	$\boldsymbol{\beta}_{21}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\boldsymbol{\beta}_{31}$	$\begin{bmatrix} -0.5812 \\ -0.8137 \\ 0 \end{bmatrix}$
$\boldsymbol{\beta}_{13}$	$\begin{bmatrix} 0.5812 \\ -0.8137 \\ 0 \end{bmatrix}$	$\boldsymbol{\beta}_{23}$	$\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$	$\boldsymbol{\beta}_{32}$	$\begin{bmatrix} -0.5812 \\ 0.8137 \\ 0 \end{bmatrix}$

Table A.1: Table of desired bearing for the triangular formation

Initial position	Value	Initial position	Value	Initial position	Value
$\mathbf{p}_1(0)$	$\begin{bmatrix} 1.0 \\ 1.0 \\ 0 \end{bmatrix}$	$\mathbf{p}_2(0)$	$\begin{bmatrix} 7.0 \\ 7.0 \\ 0 \end{bmatrix}$	$\mathbf{p}_3(0)$	$\begin{bmatrix} 9.0 \\ 2.0 \\ 0 \end{bmatrix}$

Table A.2: Table of the agents initial positions during the simulation

N	Init condition	Value	Init condition	Value	Init condition	Value
1	$\mathbf{p}_1(0)$	$\begin{bmatrix}3\\5\\0\end{bmatrix}$	$\mathbf{p}_2(0)$	$\begin{bmatrix}2\\2\\0\end{bmatrix}$	$\mathbf{p}_3(0)$	$\begin{bmatrix}-5\\-5\\0\end{bmatrix}$
	$\boldsymbol{\alpha}_1(0)$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\boldsymbol{\alpha}_2(0)$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\boldsymbol{\alpha}_3(0)$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$
2	$\mathbf{p}_1(0)$	$\begin{bmatrix}3\\5\\0\end{bmatrix}$	$\mathbf{p}_2(0)$	$\begin{bmatrix}2\\2\\0\end{bmatrix}$	$\mathbf{p}_3(0)$	$\begin{bmatrix}-5\\-5\\0\end{bmatrix}$
	$\boldsymbol{\alpha}_1(0)$	$\begin{bmatrix}0\\0\\\frac{pi}{3}\end{bmatrix}$	$\boldsymbol{\alpha}_2(0)$	$\begin{bmatrix}0\\0\\-\frac{pi}{2}\end{bmatrix}$	$\boldsymbol{\alpha}_3(0)$	$\begin{bmatrix}0\\0\\0\end{bmatrix}$

TABLE A.3: Table of the agents initial conditions for extra experiments

In the table above (Table A.3) contains:

1. the change only the quadrotors positions and the initial bearings information for Simulink and Gazebo experiment
2. the change the quadrotors positions, the quadrotors yaws and the initial bearings information for Simulink and Gazebo experiment

The following table contains the bound setting on the input during the Simulink and Gazebo experiments:

	Lower bound	**Upper bound**
$\mathbf{u}_i = \begin{bmatrix}T_i[N]\\ \omega_{x_i}[\frac{rad}{s}]\\ \omega_{y_i}[\frac{rad}{s}]\\ \omega_{z_i}[\frac{rad}{s}]\end{bmatrix}$	$\mathbf{u}_i = \begin{bmatrix}0\\-1.0\\-1.0\\1.0\end{bmatrix}$	$\mathbf{u}_i = \begin{bmatrix}37\\1.0\\1.0\\1.0\end{bmatrix}$

TABLE A.4: Table of settings on gazebo experiments.

A.1.1 Simulink settings

Exp	$\mathbf{Q}_\beta$	$\mathbf{Q}_\alpha$	$\mathbf{Q}_v$	$\mathbf{Q}_z$	$\mathbf{R}$	$\mathbf{p}_\mathcal{F}$	$s_\mathcal{F}$	$\psi_\mathcal{F}$
1	$\mathbf{I} \cdot 3500$	$\text{diag}\{\begin{bmatrix}500\\500\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 100$	$\mathbf{I} \cdot 500$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$			
2	$\mathbf{I} \cdot 1250$	$\text{diag}\{\begin{bmatrix}0\\0\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 50$	$\mathbf{I} \cdot 0$	$\text{diag}\{\begin{bmatrix}10\\10\\10\\10\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\1\end{bmatrix}$	0.3	0

TABLE A.5: Table of terms weights of cost function during the simulink experiments

Setting	**Value**	**Setting**	**Value**
T_{st}	0.01	N	20
T_s	0.01	N2	5

TABLE A.6: Table with NMPC settings of Simulink model

In the Table A.6 and the Table A.10 the symbols have these meanings:

- T_{st}: Shooting time
- T_s: Sampling time
- N: Number of steps for prediction horizon
- N2: horizon length after partial condensing

	Baricenter	**Yaw**	**Scale**
K_p	$\mathbf{I}_3$*0.225	0.0875	0.25
K_i	$\mathbf{I}_3$*0.00025	0.0005	0.0025
K_d	$\mathbf{I}_3$*0	0	0

TABLE A.7: Table with TMC gains for the Simulink scheme

	Roll	**Pitch**	**Yaw**
K	$\mathbf{I}_3$*3	$\mathbf{I}_3$*3	$\mathbf{I}_3$*1
K_i	$\mathbf{I}_3$*0.0024	$\mathbf{I}_3$*0.0024	$\mathbf{I}_3$*0
K_d	$\mathbf{I}_3$*0.35	$\mathbf{I}_3$*0.35	$\mathbf{I}_3$*0.1

TABLE A.8: Table with rate controllers gains for the Simulink scheme

Setting	Value	Setting	Value
Hessian	Gauss Newton	**QP solver**	hpipm sparse
Integrator	ERK4	**Hotstart**	no
Condensing	no	**ref_type**	1(Time varying)
Shifting	no	**Nonuniform grid**	0
RTI	yes		

TABLE A.9: Table with NMPC optimization settings of Simulink model

A.1.2 Gazebo settings

Setting	Value	Setting	Value
T_{st}	0.01	N	20
T_s	0.01	N2	10

TABLE A.10: Table with NMPC settings of Gazebo experiment

	Baricenter	Yaw	Scale
K_p	$\text{diag}\{\begin{bmatrix}0.25\\0.25\\1\end{bmatrix}\}$	0.0	0.25
K_i	**I***0.00025	0.0	0.005
K_d	**I***0	0	0

TABLE A.11: Table with TMC gains of the gazebo experiments

Exp	$\mathbf{Q}_\beta$	$\mathbf{Q}_\alpha$	$\mathbf{Q}_v$	$\mathbf{Q}_z$	$\mathbf{R}$	$\mathbf{p}_\mathcal{F}$	$s_\mathcal{F}$
1	$\mathbf{I} \cdot 5000$	$\text{diag}\{\begin{bmatrix}500\\500\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 100$	$\mathbf{I} \cdot 500$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\2\end{bmatrix}$	0.3
2	$\mathbf{I} \cdot 5000$	$\text{diag}\{\begin{bmatrix}500\\500\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 500$	$\mathbf{I} \cdot 1000$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$		
3	$\mathbf{I} \cdot 5000$	$\text{diag}\{\begin{bmatrix}500\\500\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 250$	$\mathbf{I} \cdot 1000$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$		
4	$\mathbf{I} \cdot 3500$	$\text{diag}\{\begin{bmatrix}0\\0\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 500$	$\mathbf{I} \cdot 0$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\2\end{bmatrix}$	0.3
5	$\mathbf{I} \cdot 3500$	$\text{diag}\{\begin{bmatrix}0\\0\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 500$	$\mathbf{I} \cdot 0$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\2\end{bmatrix}$	0.3
6	$\mathbf{I} \cdot 3500$	$\text{diag}\{\begin{bmatrix}0\\0\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 500$	$\mathbf{I} \cdot 0$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\2\end{bmatrix}$	0.3
7	$\mathbf{I} \cdot 3500$	$\text{diag}\{\begin{bmatrix}0\\0\\0\end{bmatrix}\}$	$\mathbf{I} \cdot 500$	$\mathbf{I} \cdot 0$	$\text{diag}\{\begin{bmatrix}10\\50\\50\\25\end{bmatrix}\}$	$\begin{bmatrix}0\\0\\2\end{bmatrix}$	0.3

TABLE A.12: Table of terms weights of cost function during the gazebo experiments

For the Table A.12 following there is its legend for the experiments:

- 1 - experiment with old cost function with reference altitude of 2 mt without TCM yaw
- 2 - experiment with old cost function without TMC and with $\mathbf{Q}_v = \mathbf{I} \cdot 500$
- 3 - experiment with old cost function without TMC and with $\mathbf{Q}_v = \mathbf{I} \cdot 250$
- 4 - experiment with last cost function with reference altitude of 2 mt without TCM yaw
- 5 - experiment with last cost function with change of TCM setpoint at 30s of simulation

- 6,7 - extras experiments with different different initial conditions on position and attitude

A.2 Physical table

Name	**Value**	**Name**	**Value**
g $[\frac{m}{s^2}]$	9,81	**r_1** [m]	$\begin{bmatrix} 0.174 \\ -0.174 \end{bmatrix}$
m [kg]	9.81	**r_2** [m]	$\begin{bmatrix} -0.174 \\ 0.174 \end{bmatrix}$
J $[kgm^2]$	diag{ $\begin{bmatrix} 0.0216 \\ 0.0216 \\ 0.040 \end{bmatrix}$ }	**r_3** [m]	$\begin{bmatrix} 0.174 \\ 0.174 \end{bmatrix}$
l	0.24607316	**r_4** [m]	$\begin{bmatrix} -0.174 \\ -0.174 \end{bmatrix}$
c_f $[Ns^2]$	8.54858e-06	c_τ [m]	8.06428e-05
min spinning $[\frac{rad}{s}]$	100	**max spinning** $[\frac{rad}{s}]$	1000

Table A.13: Table with value for quadrotor's model

- g: gravitational acceleration
- m: quadrotor's mass
- l: quadrotor arm length
- r_i is the rotor's position
- J: quadrotor inertial matrix
- c_f: constant of trust force
- c_τ: constant of drag moment

Appendix B

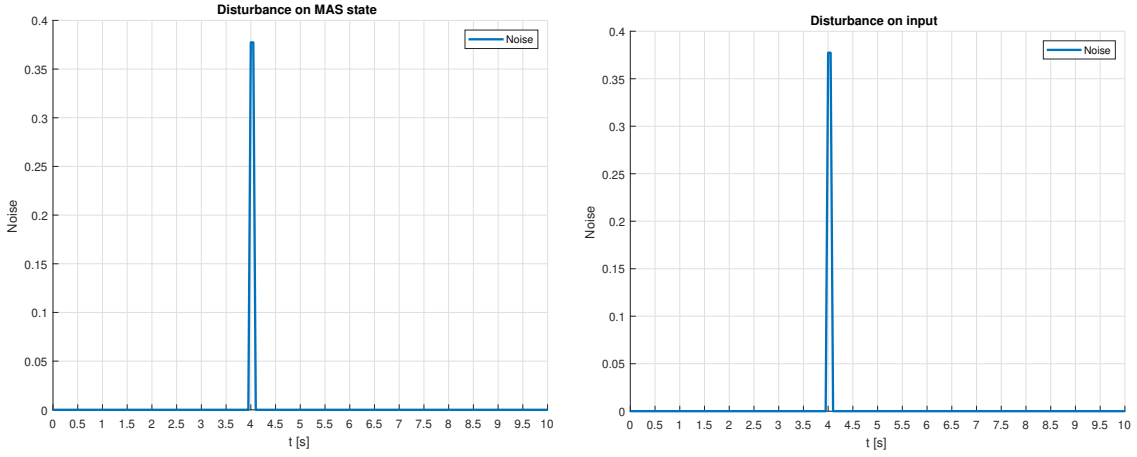
Extra experiments

B.1 Experiments

B.1.1 Second const function - state vs input disturbance

In this experiment, we carried out two simulations in the first we enabled a disturbance on the state and the second with a disturbance on the input for few ms. The idea is to see the behaviour of the NMPC with a Dirac disturbance on the input and on the state. We had noticed that the perturbations didn't cause any major problems for the NMPC and MAS is held the formation even with the disturbance.

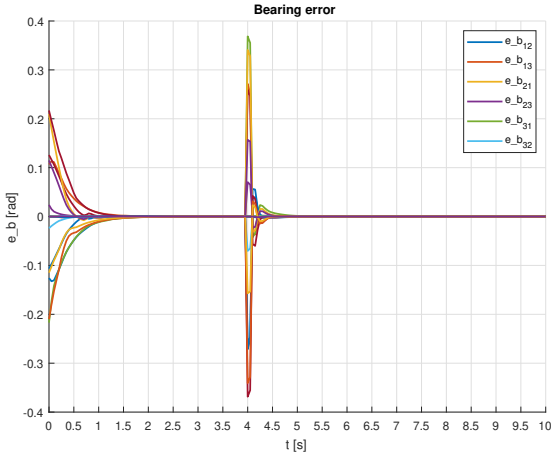
The results of the experiments are shown in columns: on the right side is the experiment with the state disturbance and on the left side there is the experiment with the input disturbance.



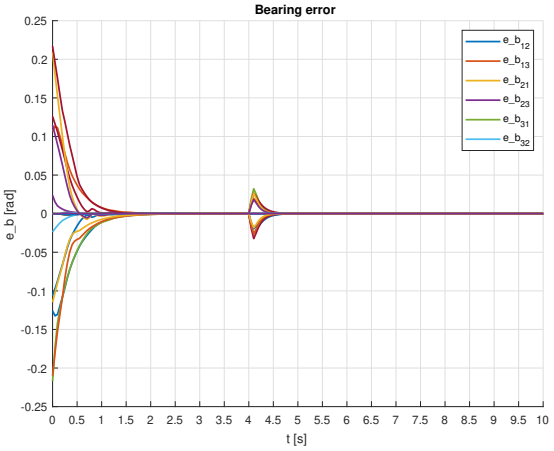
(a) Disturbance for the state

(b) White noise for the input

FIGURE B.1: Disturbances for the state and the input

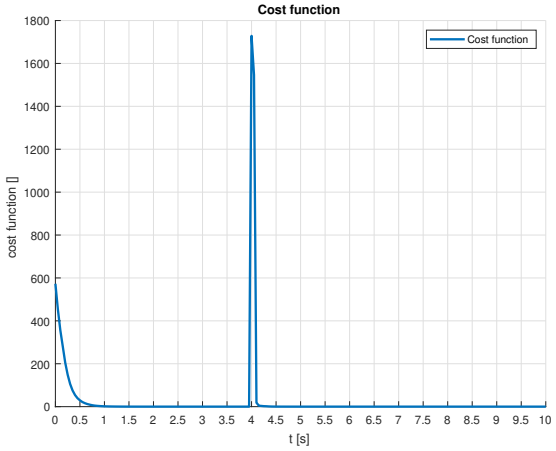


(a) Bearing error with state disturbance

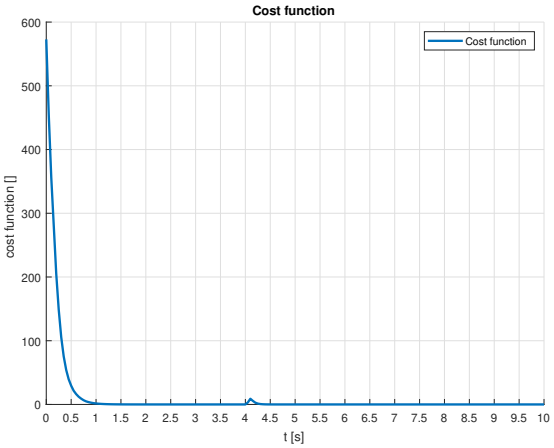


(b) Bearing error with input disturbance

FIGURE B.2: Bearing error with state vs input disturbance

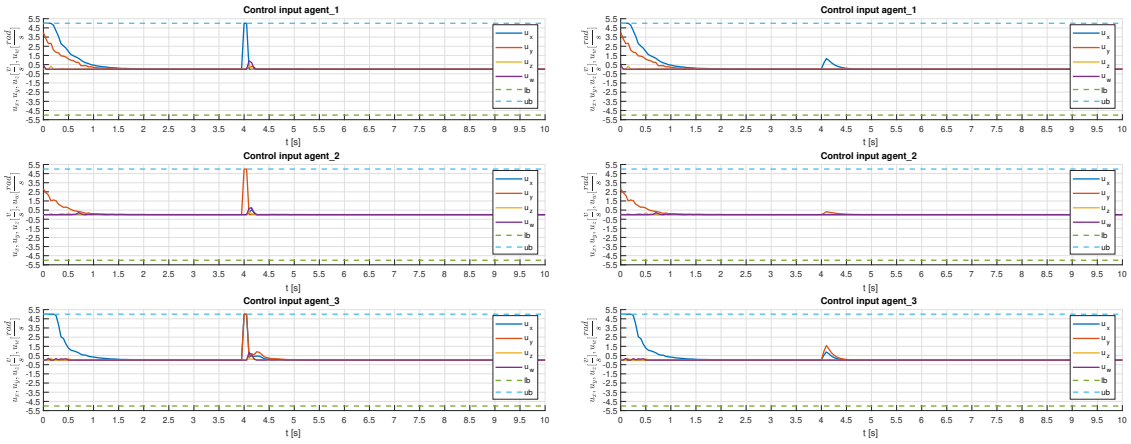


(a) Cost function with state disturbance



(b) Cost function with input disturbance

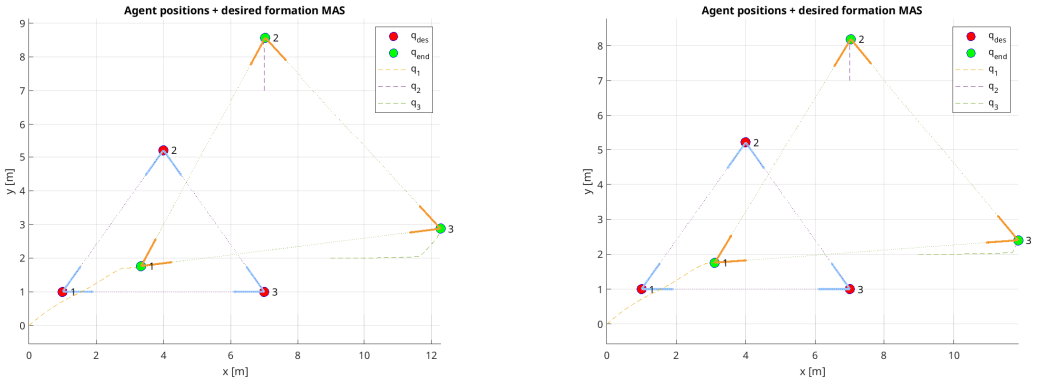
FIGURE B.3: Cost function with state vs input disturbance



(a) Control input with state disturbance

(b) Control input with input disturbance

FIGURE B.4: Control input with state vs input disturbance



(a) MAS position with state disturbance

(b) Control input with input disturbance

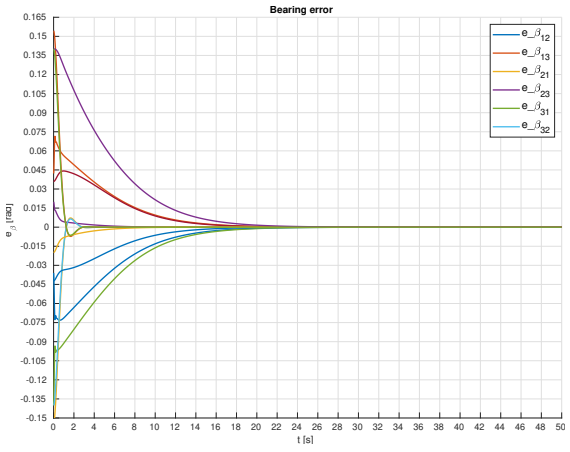
FIGURE B.5: MAS position with state vs input disturbance

B.2 Experiments Simulink GZ

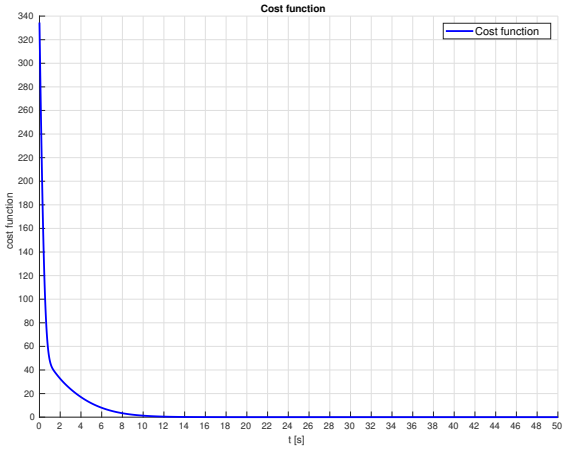
In this section we show the experiments with different initial positions but the same desired bearing or experiment for MAS with 1 free agent and 2 fixed agents.

B.2.1 Experiment on MAS with 1 free agent and 2 fixed agents

In this experiment, we have tested the internal system model described in the Section 3.2.2.1 and the figures below show the result that we obtained.



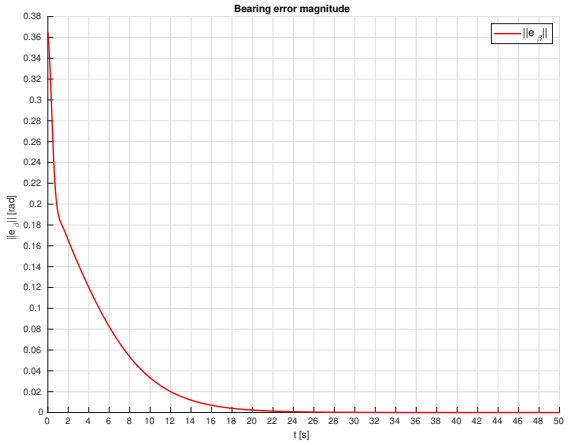
(a) Bearing error



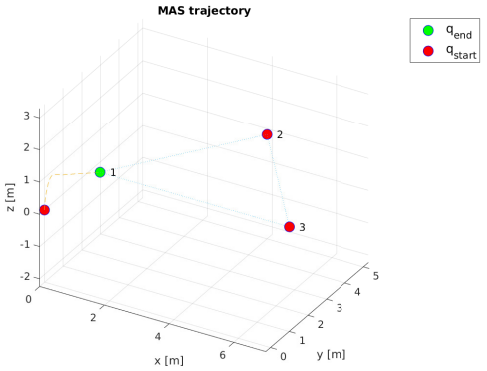
(b) Cost function

FIGURE B.6: Bearing error and cost function

The figures above show that the NMPC brings the cost function to zero and the same behaviour for the bearings error. This means that the agent has reached the desired formation and this is confirmed by also the figure below where the bearing magnitude went to zero.



(a) Bearing error magnitude



(b) Trajectory and formation of MAS

FIGURE B.7: Bearing error and MAS trajectory

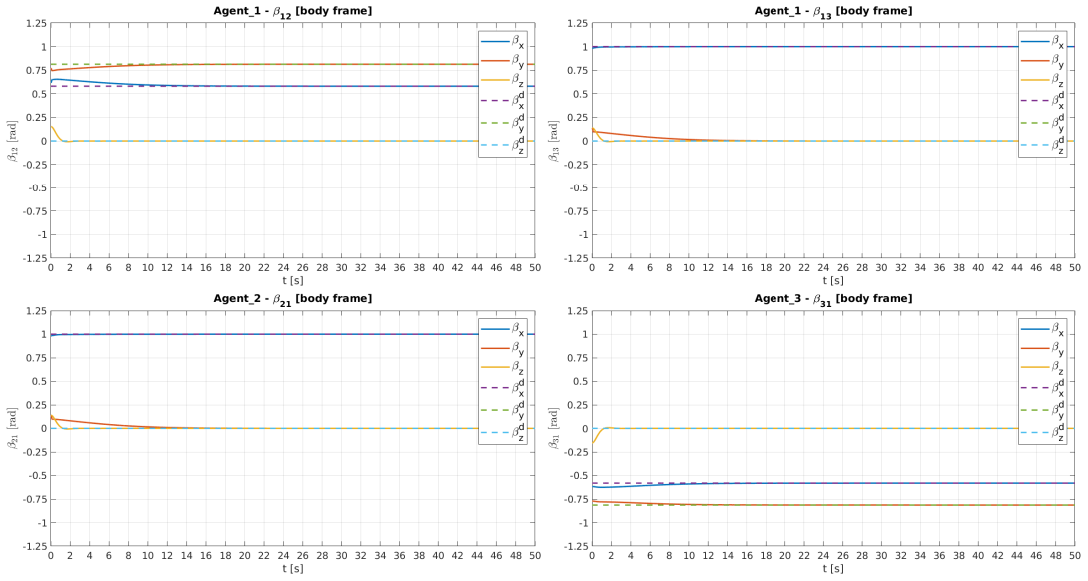


FIGURE B.8: Bearing vector of each agent

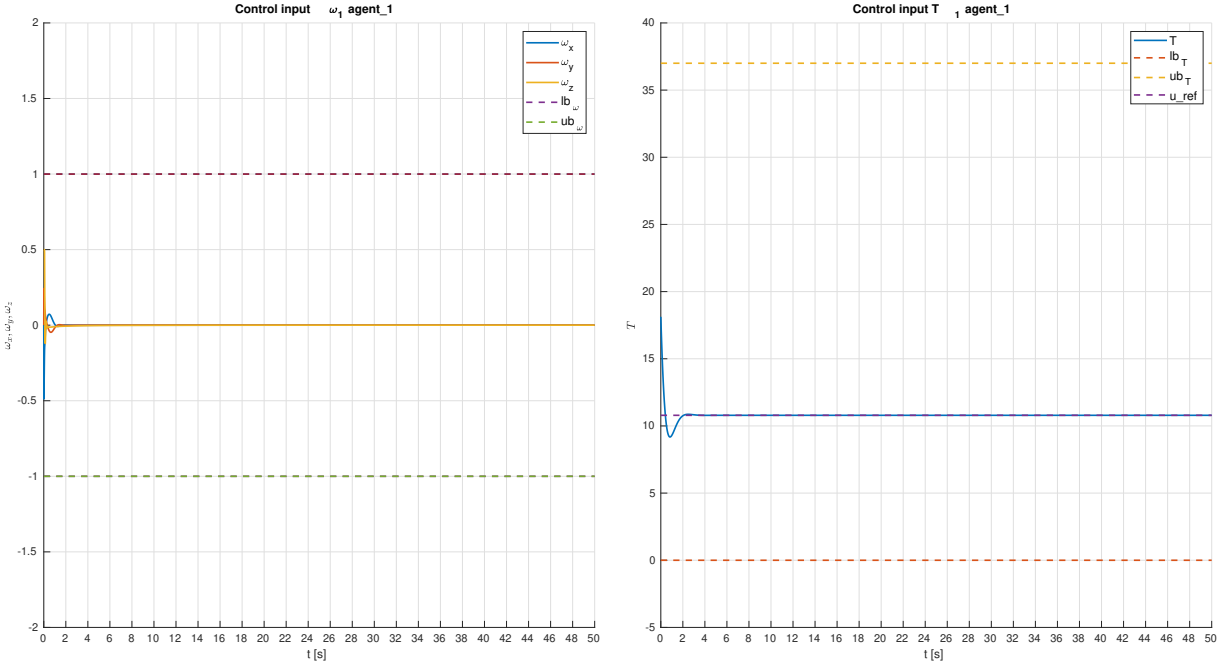


FIGURE B.9: Control input for each agent

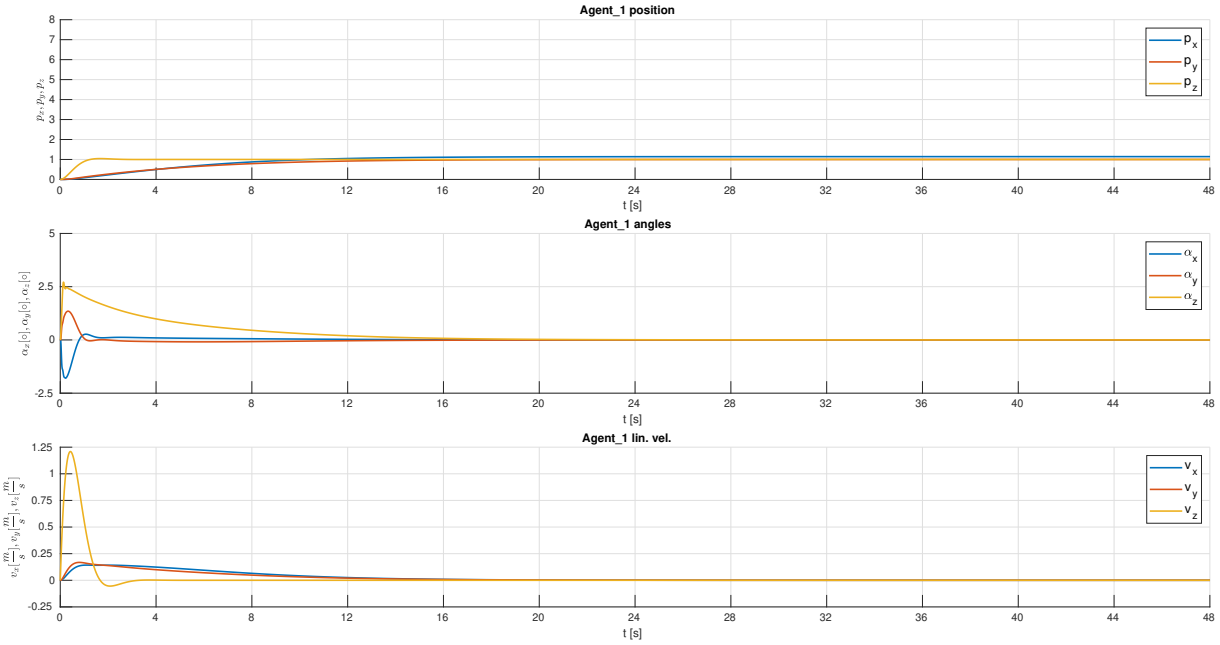
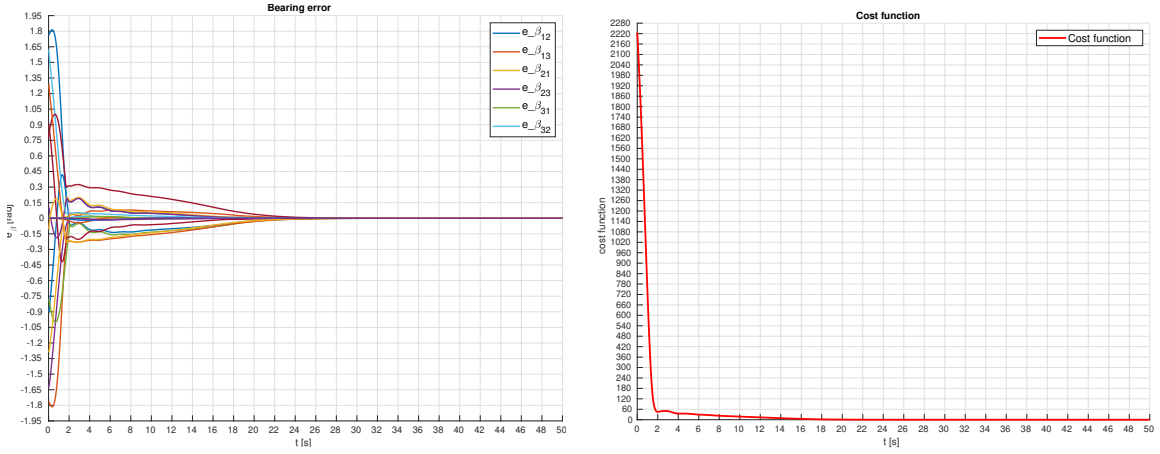


FIGURE B.10: MAS state

B.2.2 Experiment 1

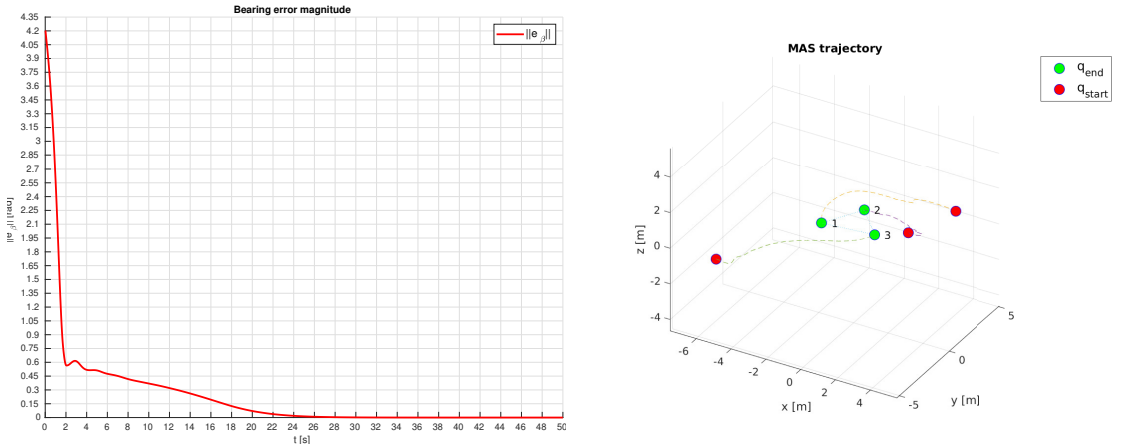
In this experiment we have tested the NMPC with TMC but with a new set of initial positions and bearings information.We can see that this doesn't cause any problems in achieving the desired formation. In the Table A.3 there are the value of this new set of initial positions and in the table Table A.12 there are the gains that we have used. We have report the simulink experiment and then the gazebo experiment. We don't add other comments on these plots because we have already commented extensively in Chapter 5.

Simulink



(a) Bearing error (b) Cost function

FIGURE B.11: Bearing error and cost function with TMC



(a) Bearing magnitude (b) Trajectory and formation of MAS with TMC

FIGURE B.12: Bearing error and MAS trajectory with TMC

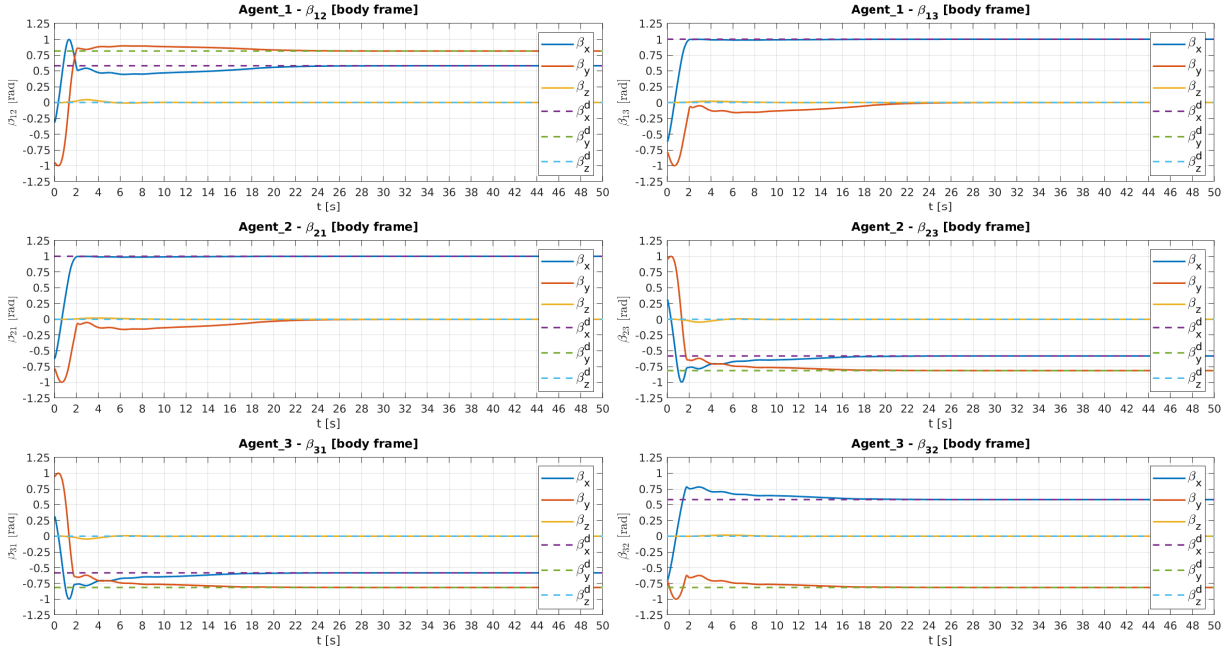


FIGURE B.13: Bearing vector of each agent with TMC

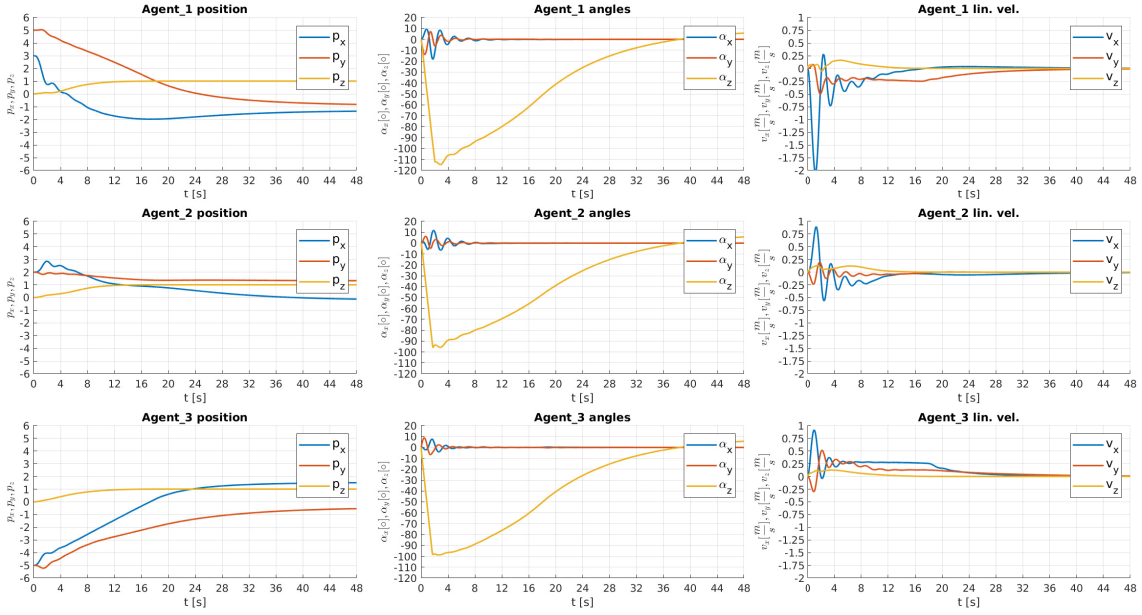
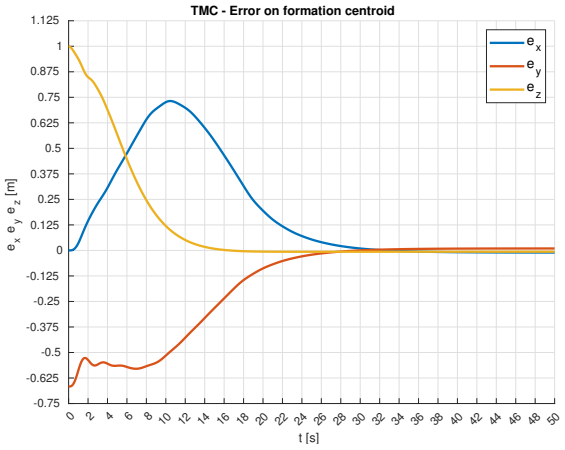
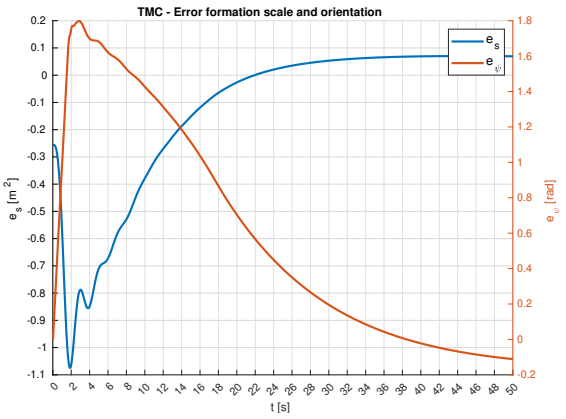


FIGURE B.14: MAS state with TMC



(a) TMC PI error on formation centroid



(b) TMC PI error on formation scale

FIGURE B.15: TMC errors on formation centroid and formation scale

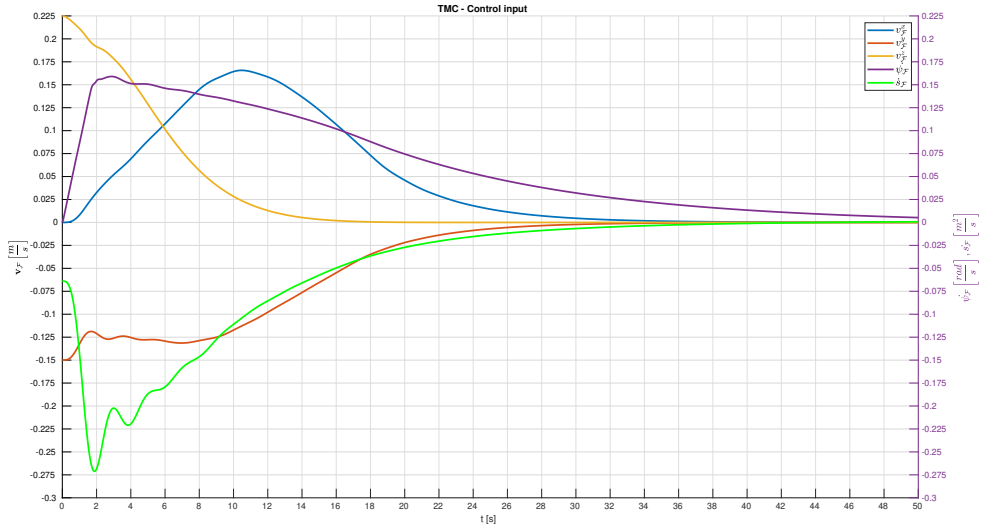


FIGURE B.16: TMC on formation velocities

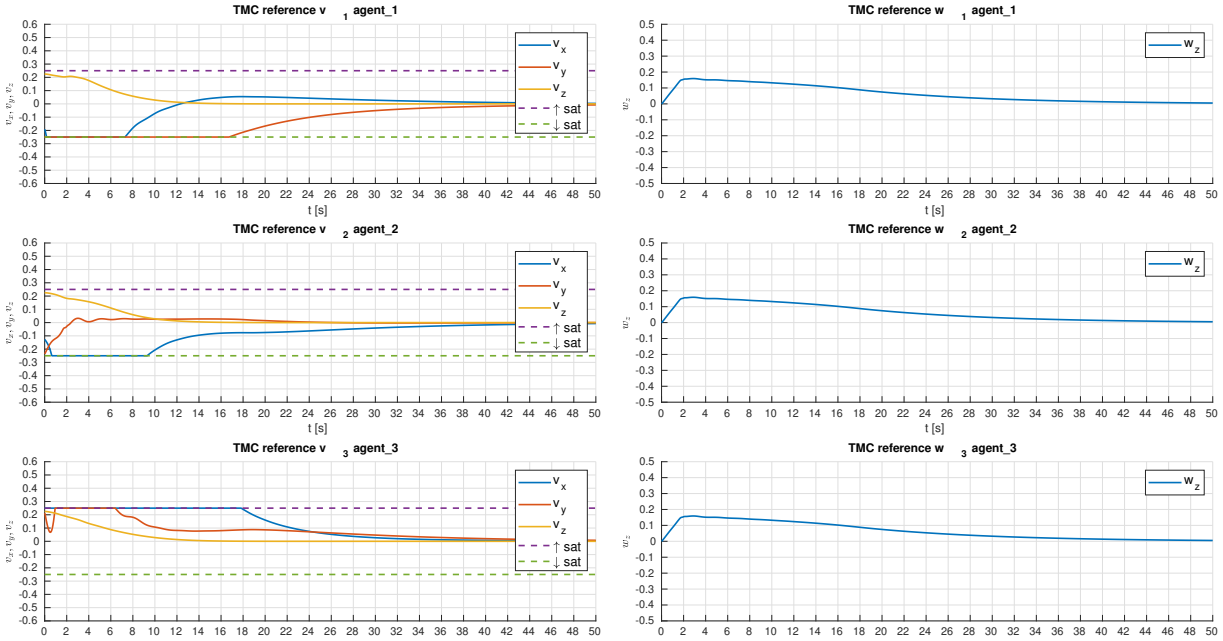
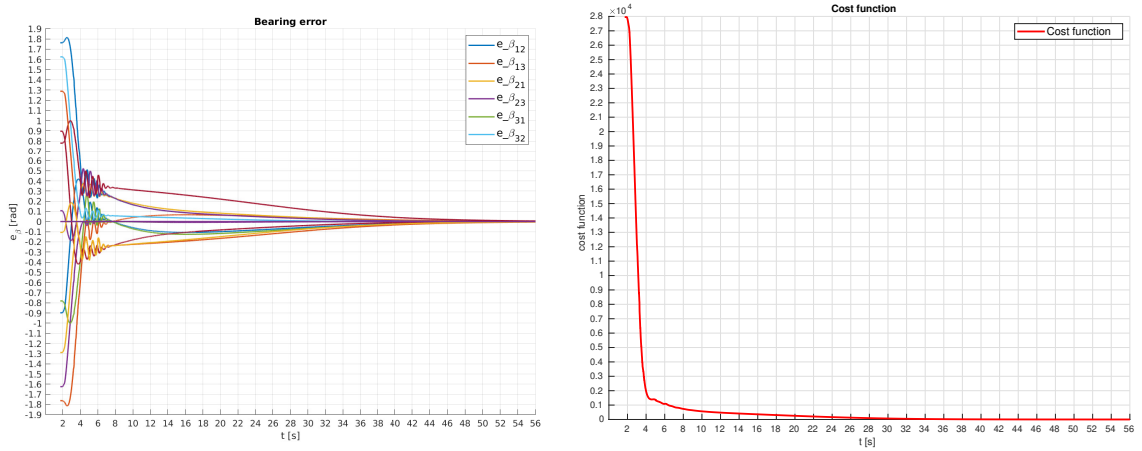


FIGURE B.17: TMC on agent velocities

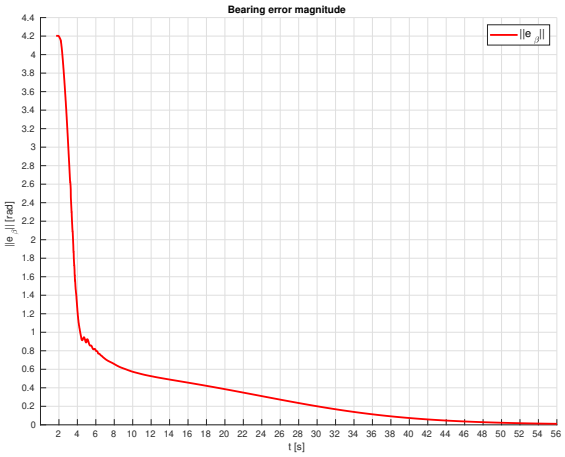
ROS 2/Gazebo



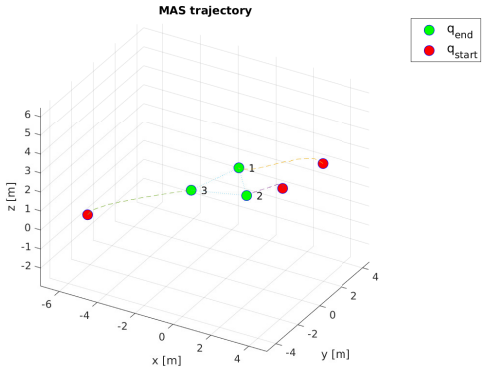
(a) Bearing error

(b) Cost function

FIGURE B.18: Bearing error and cost function with TMC



(a) Bearing magnitude



(b) Trajectory and formation of MAS with TMC

FIGURE B.19: Bearing error and cost function with TMC

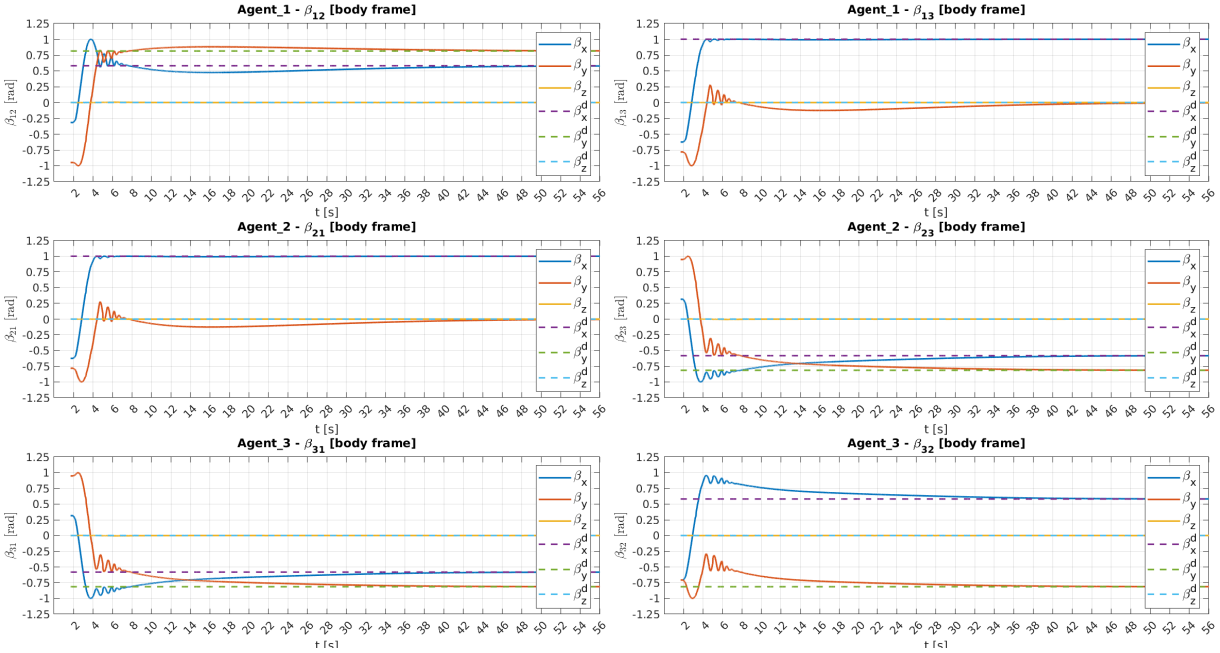


FIGURE B.20: Bearing vector of each agent with TMC

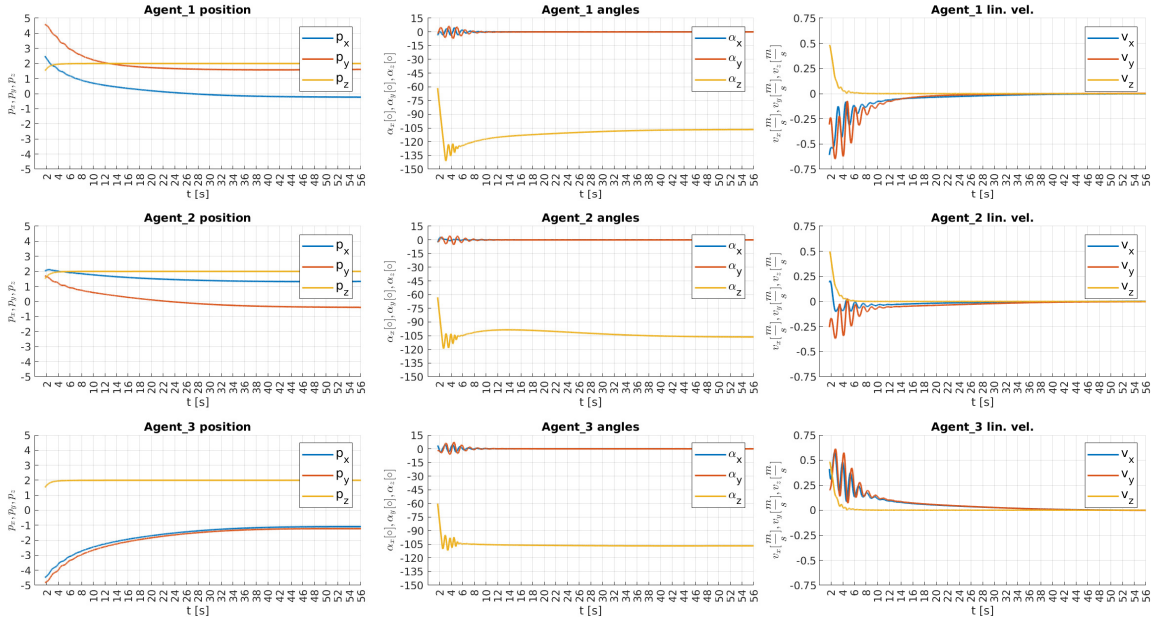


FIGURE B.21: MAS state with TMC

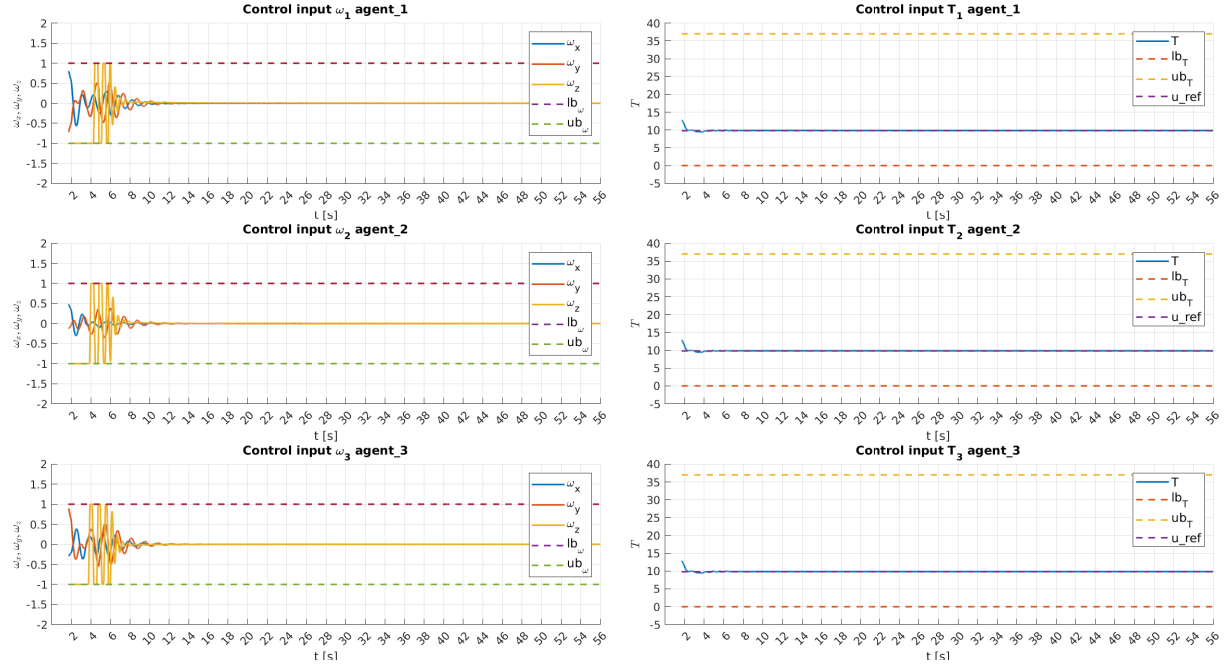


FIGURE B.22: Control input for each agent with TMC

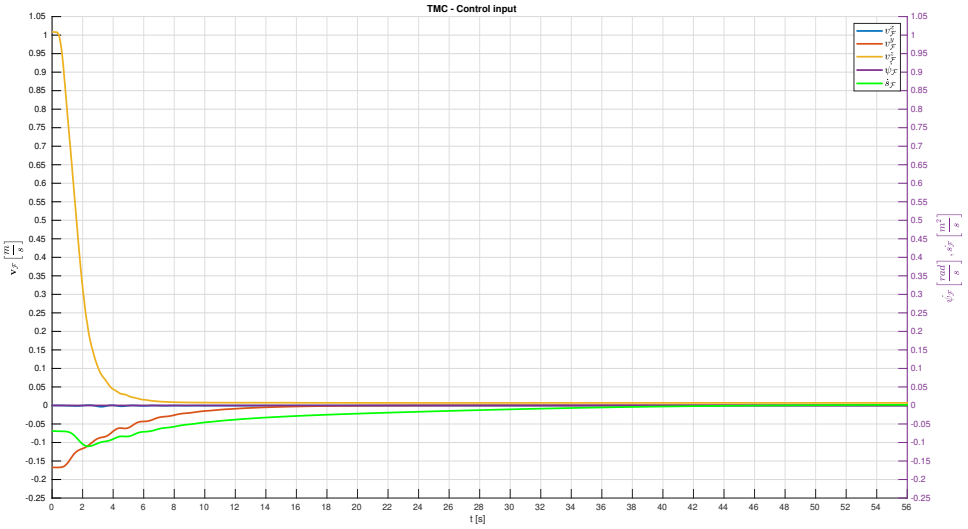


FIGURE B.23: TMC control input

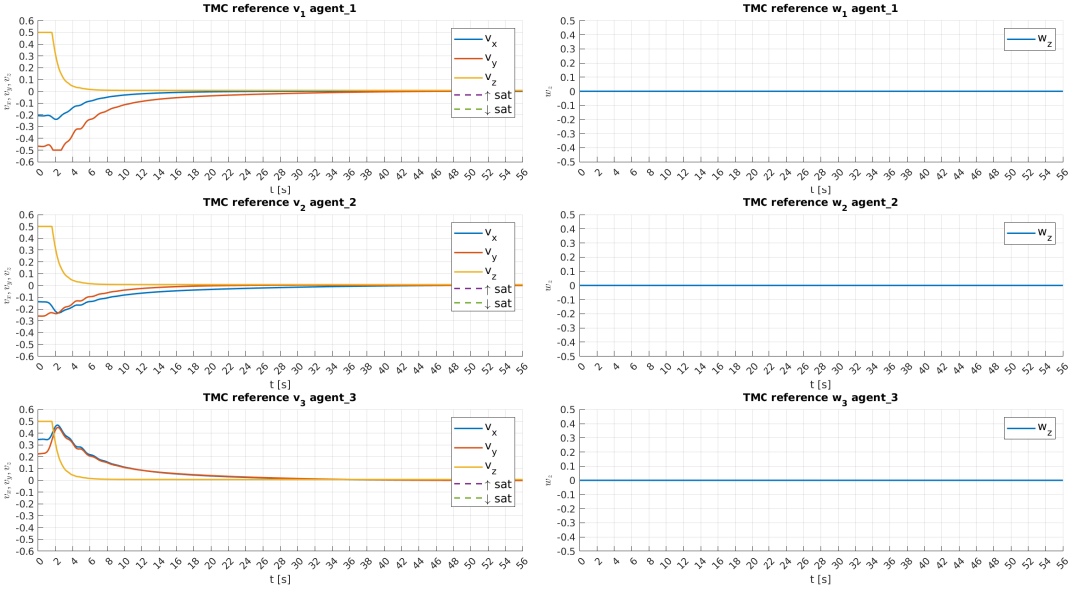
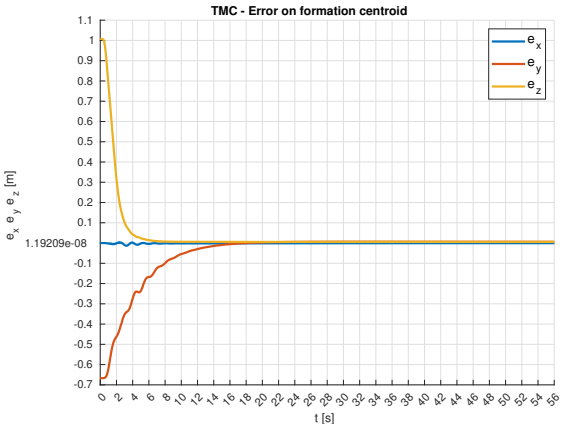
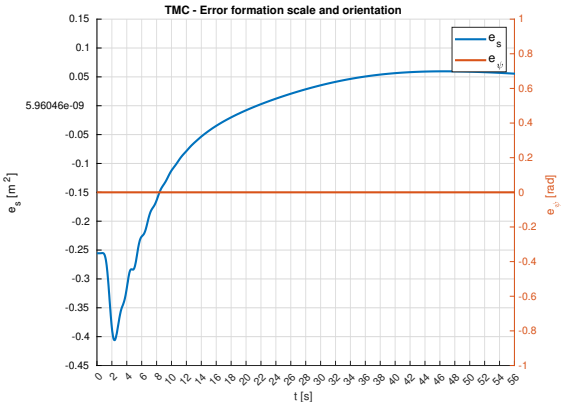


FIGURE B.24: TMC PI error on formation velocities



(a) TMC PI error on formation centroid



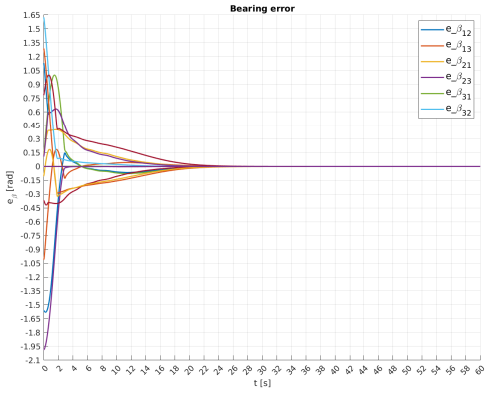
(b) TMC PI error on formation scale

FIGURE B.25: Bearing error and cost function with TMC

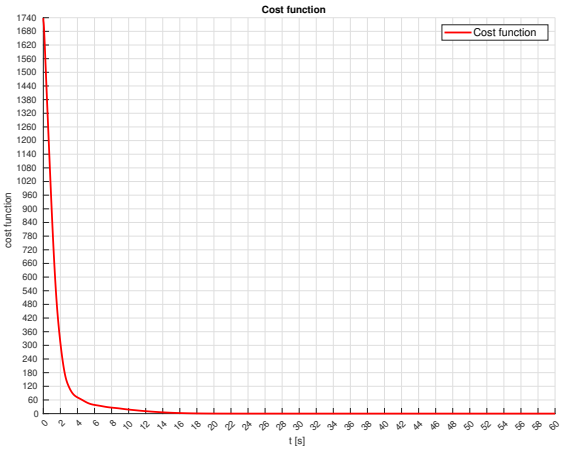
B.2.3 Experiment 2

In this experiment we have tested the NMPC with TMC but with a new set of initial positions, a new set of initial attitudes and bearings information. We can see that this doesn't cause any problems in achieving the desired formation. In the Table A.3 there are the value of this new set of initial positions and in the table Table A.12 there are the gains that we have used. We have report the simulink experiment and then the gazebo experiment. We don't add other comments on these plots because we have already commented extensively in Chapter 5.

Simulink

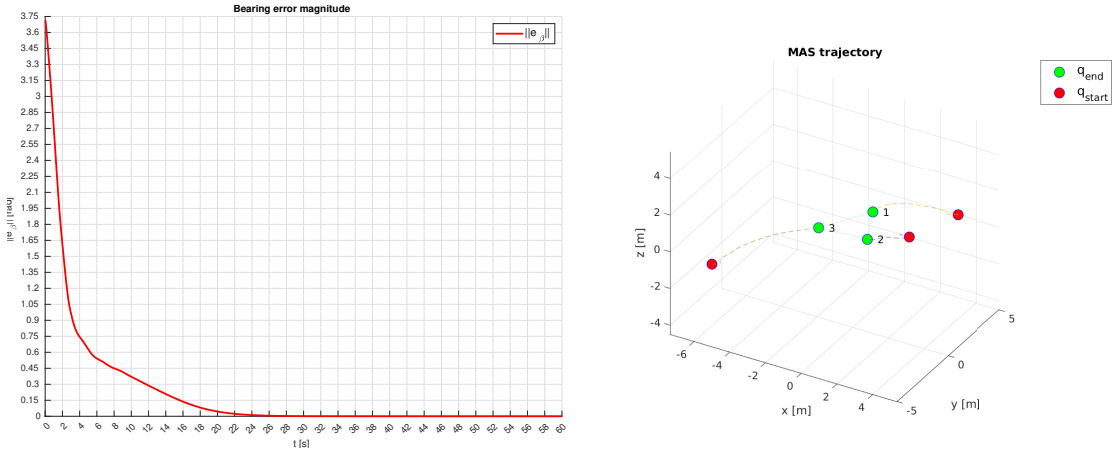


(a) Bearing error



(b) Cost function

FIGURE B.26: Bearing error and cost function with TMC



(a) Bearing magnitude

(b) Trajectory and formation of MAS with TMC

FIGURE B.27: Bearing error and MAS trajectory with TMC

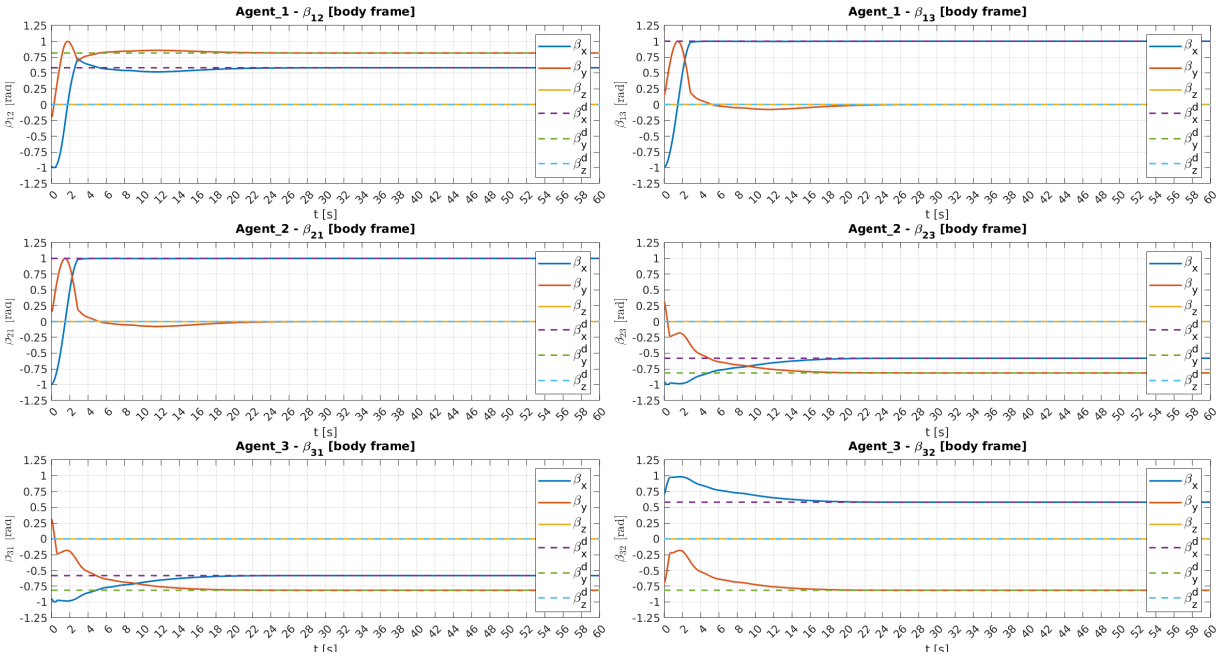


FIGURE B.28: Bearing vector of each agent with TMC

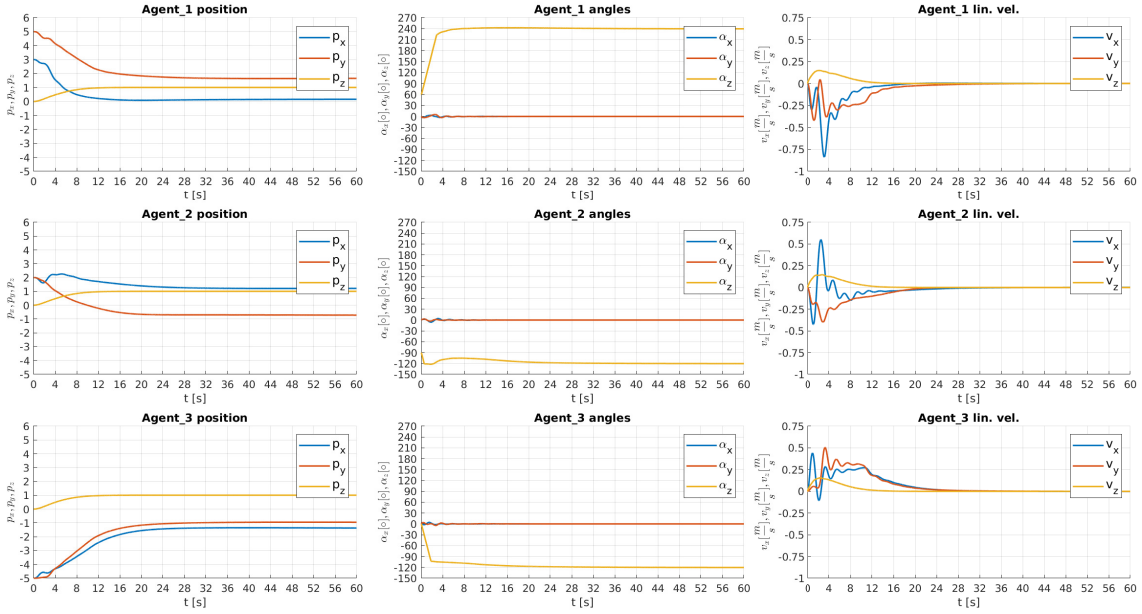
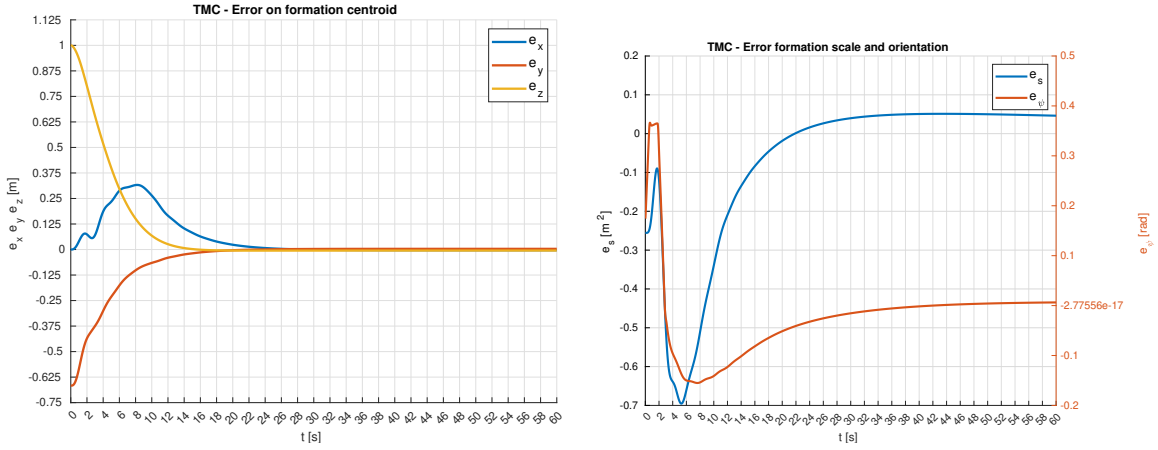


FIGURE B.29: MAS state with TMC



(a) TMC PI error on formation centroid

(b) TMC PI error on formation scale

FIGURE B.30: TMC errors on formation centroid and formation scale

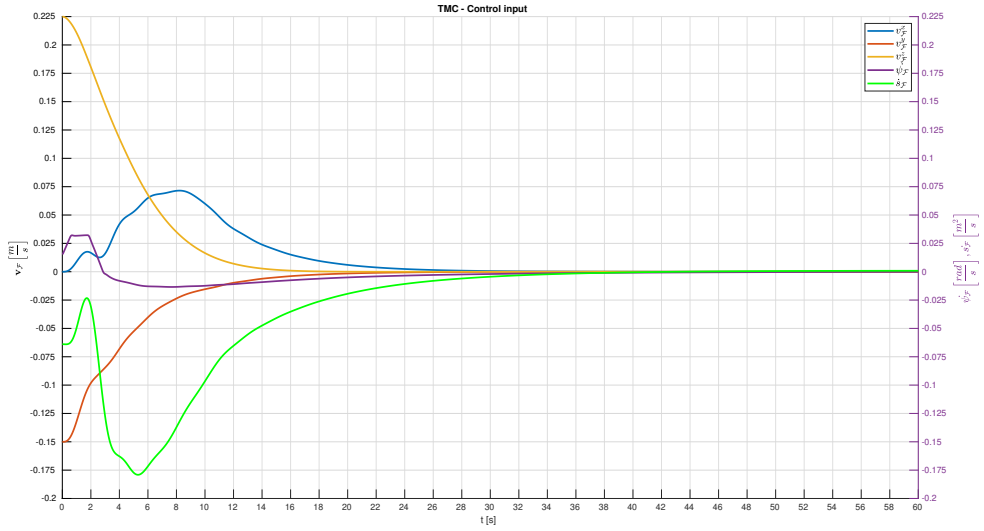


FIGURE B.31: TMC on formation velocities

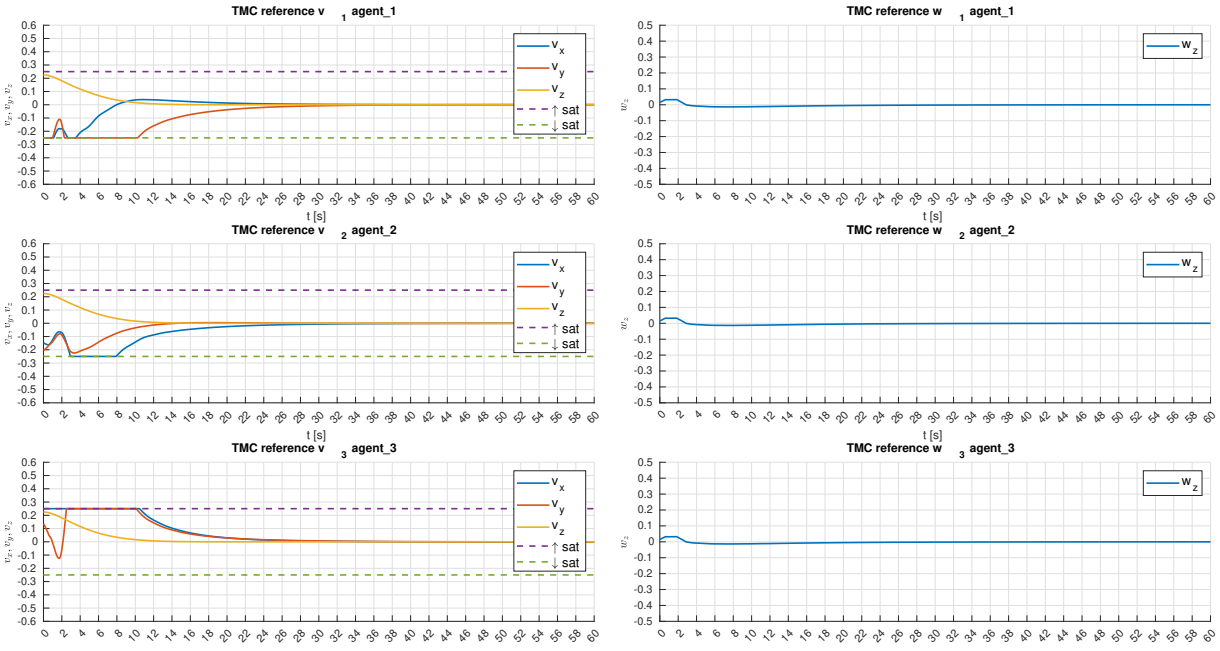
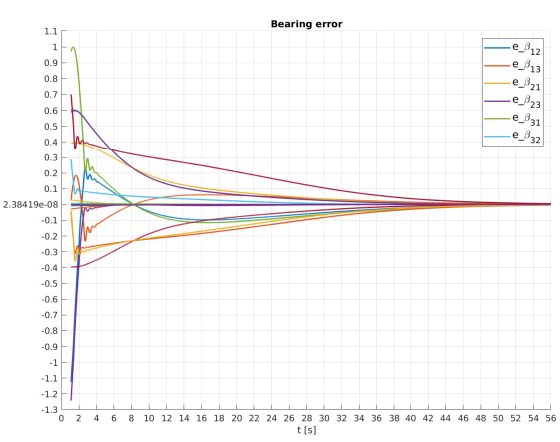
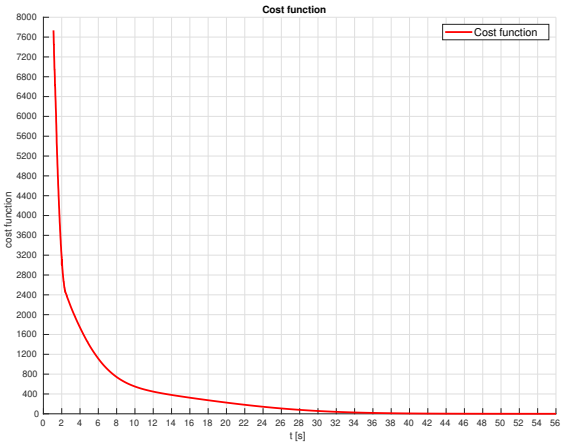


FIGURE B.32: TMC on agent velocities

ROS 2/Gazebo

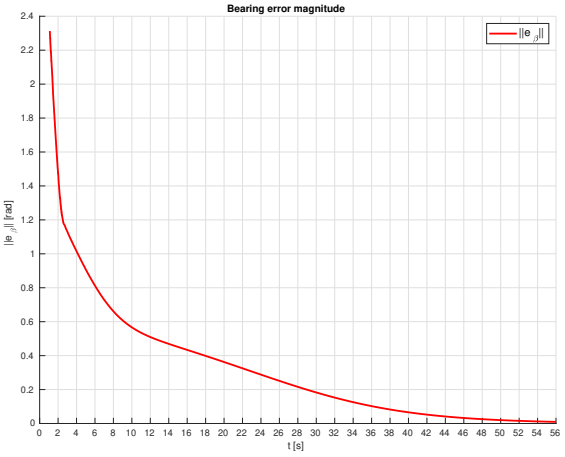


(a) Bearing error

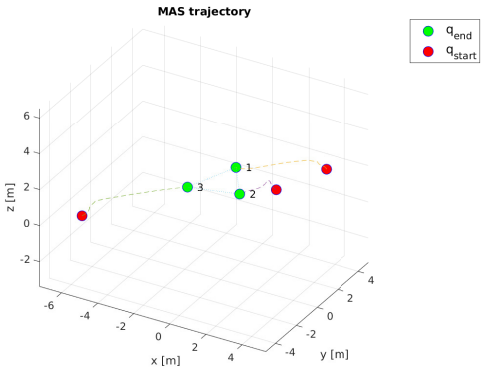


(b) Cost function

FIGURE B.33: Bearing error and cost function with TMC



(a) Bearing magnitude



(b) Trajectory and formation of MAS with TMC

FIGURE B.34: Bearing error and cost function with TMC

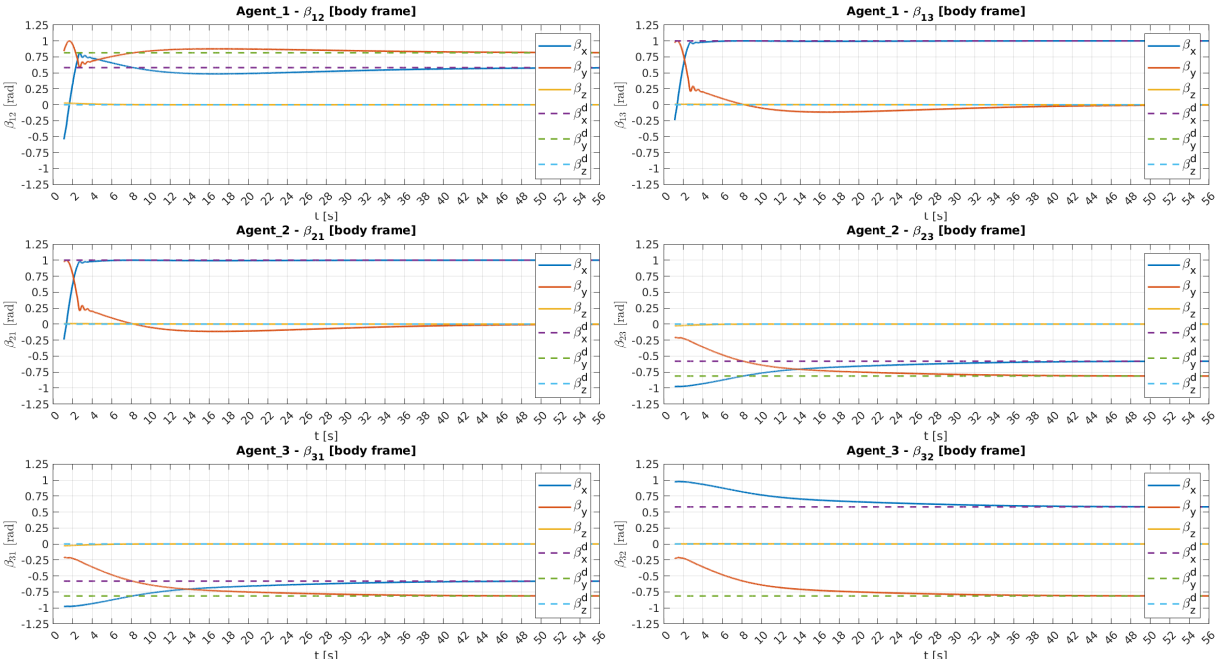


FIGURE B.35: Bearing vector of each agent with TMC

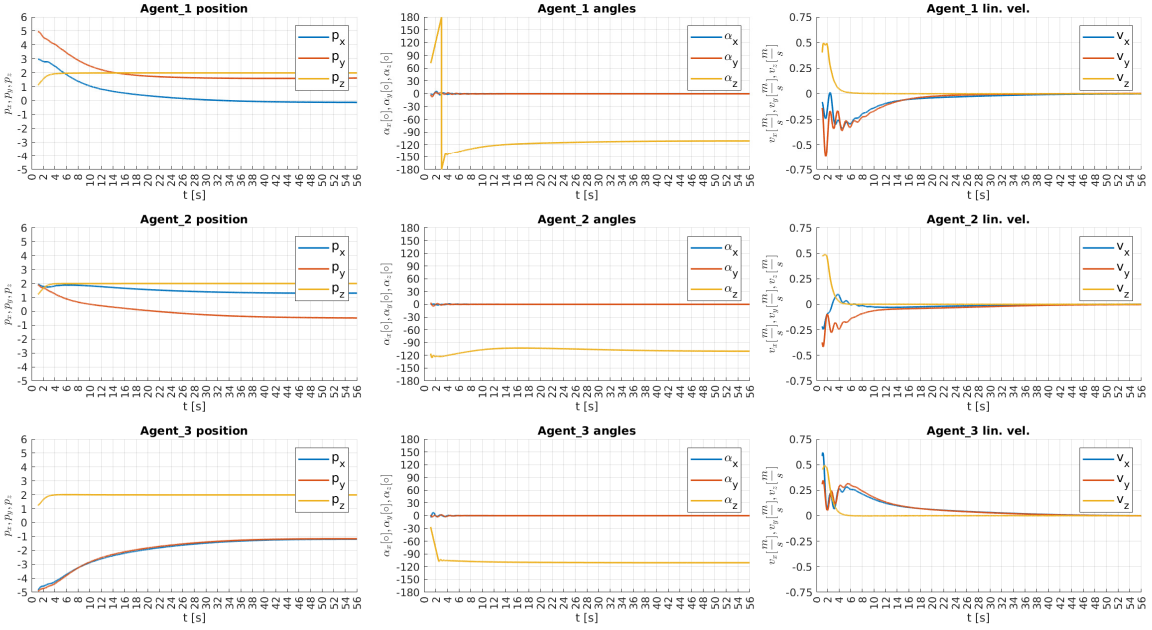


FIGURE B.36: MAS state with TMC

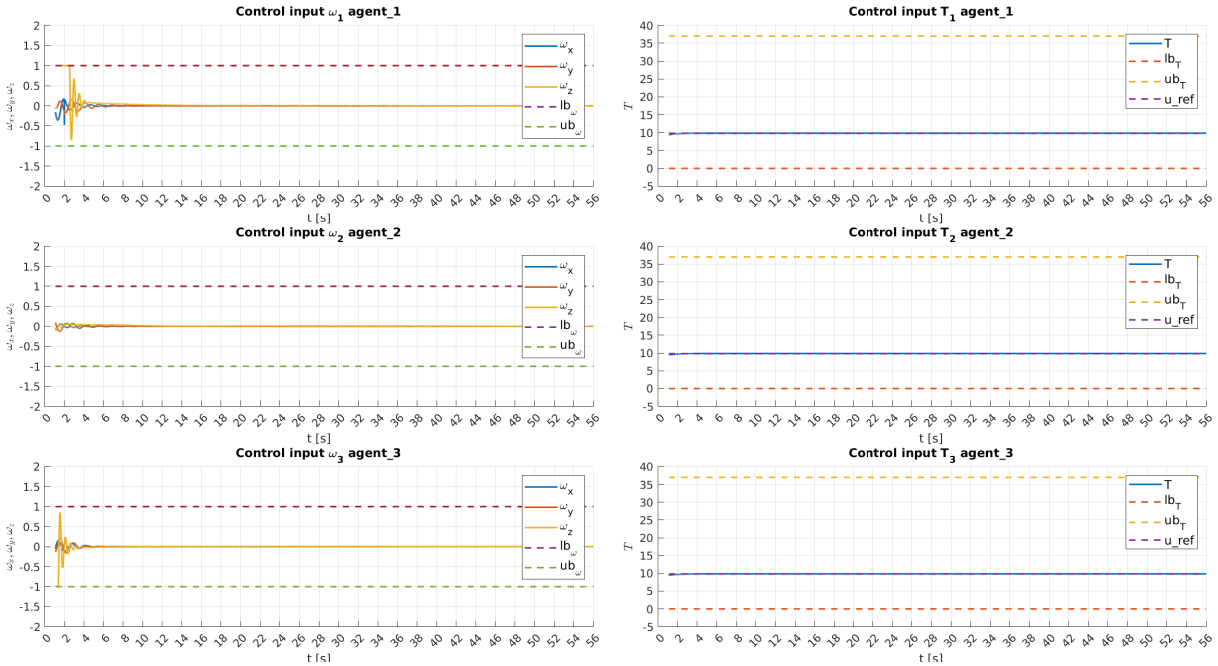


FIGURE B.37: Control input for each agent with TMC

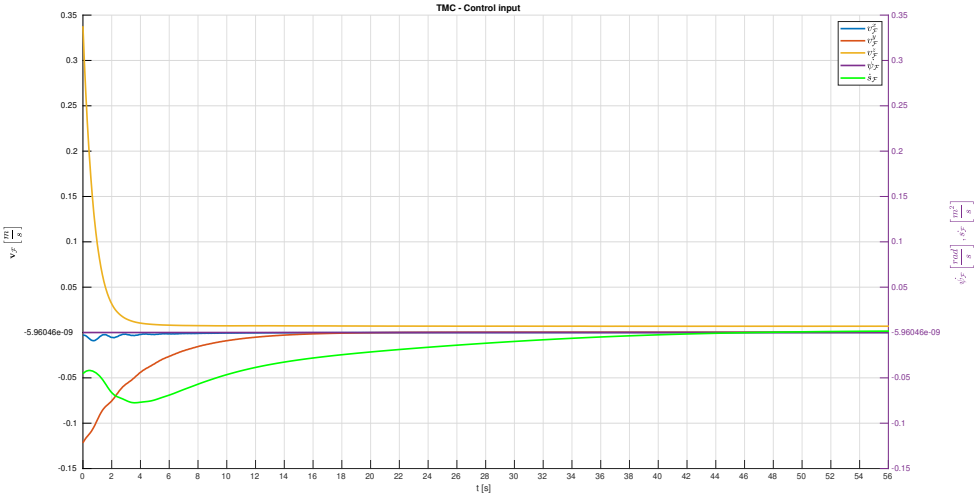


FIGURE B.38: TMC control input

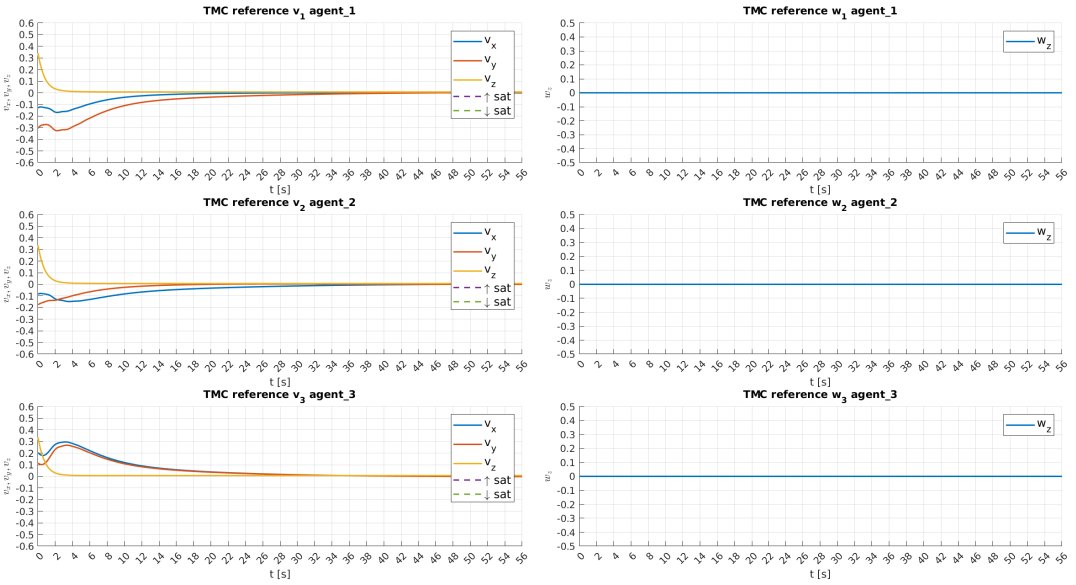
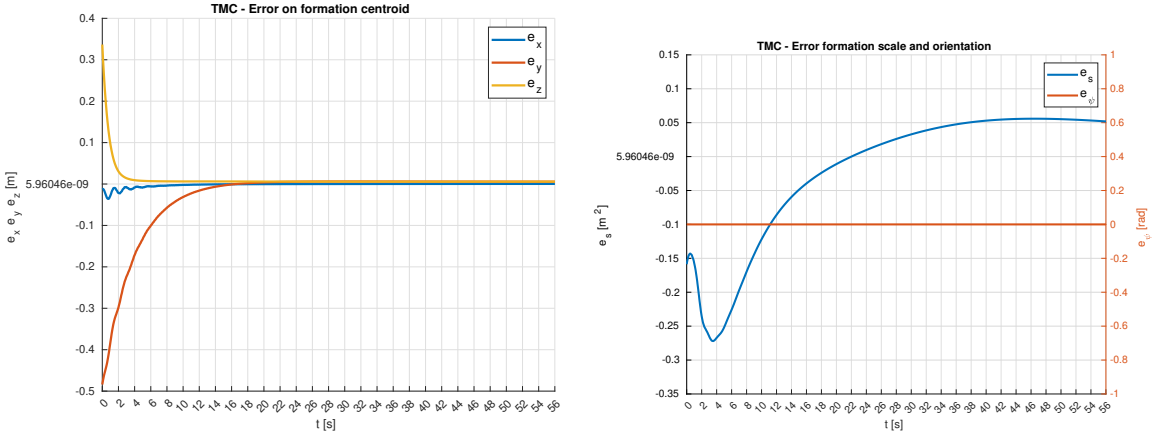


FIGURE B.39: TMC PI error on formation velocities



(a) TMC PI error on formation centroid

(b) TMC PI error on formation scale

FIGURE B.40: Bearing error and cost function with TMC

Bibliography

[1] Shiyu Zhao and Daniel Zelazo. Bearing rigidity theory and its applications for control and estimation of network systems: Life beyond distance rigidity. *IEEE*, 2019. doi: https://ieeexplore.ieee.org/document/8667521.

[2] Angelo Cenedese, the SPARCS group, and the class 2016-17. *Robotics Control 2 Lecture notes*. Angelo Cenedese, 2021.

[3] Hyo-Sung Ahn Kwang-Kyo Oh, Myoung-Chul Park. A review of multi-objective optimization: Methods and its applications. *Automatica*, 2014. doi: https://www.sciencedirect.com/science/article/pii/S0005109814004038.

[4] M. Hosseinipour Goodarzi, E. Ziaei. *Introduction to Optimization Analysis in Hydrosystem Engineering*. Springer, 2014.

[5] Enr. Picotti Y. Chen, Mat. Bruschetta. Matmpc - a matlab based toolbox for real-time nonlinear model predictive control. *IEEE*, 2018. URL https://arxiv.org/abs/1811.08761.

[6] Baris Fidan Brian D.O. Anderson, Changbin Yu and Julien M. Hendrickx. Rigid graph control architectures for autonomous formations. *IEEE*, 2008. doi: https://ieeexplore.ieee.org/abstract/document/4653105/.

[7] Enrico Picotti Alessandro Beghi, Matteo Bruschetta. *Adaptive and Model Predictive Control - Slides*. Alessandro Beghi, 2022.

[8] Sina Sharif Mansouri; George Nikolakopoulos; Thomas Gustafsson. Distributed model predictive control for unmanned aerial vehicles. *IEEE*, 2015. doi: https://ieeexplore.ieee.org/document/7441002.

[9] Daniel Zelazo Fabrizio Schiano, Antonio Franchi and Paolo Robuffo Giordano. A rigidity-based decentralized bearing formation controller for groups of quadrotor uavs. *IEEE*, 2016. doi: https://ieeexplore.ieee.org/abstract/document/7759748/.

[10] J. Erskine; R. Balderas-Hill; I. Fantoni; A. Chriette. Model predictive control for dynamic quadrotor bearing formations. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://ieeexplore.ieee.org/document/9561304.

[11] A. Richards; J. How. Decentralized model predictive control of cooperating uavs. *IEEE*, 2004. doi: https://ieeexplore.ieee.org/document/1429425.

[12] Ugo Rosolia; Francesco Braghin; Andrew G. Alleyne; Stijn De Bruyne; Edoardo Sabbioni. A decentralized algorithm for control of autonomous agents coupled by feasibility constraints. *IEEE*, 2015. doi: https://ieeexplore.ieee.org/document/7963467.

[13] Julian Erskine. Dynamic control and singularities of rigid bearing-based formations of quadrotor. *French site for PHD Thesis*, 2021. doi: https://www.theses.fr/2021ECDN0044.