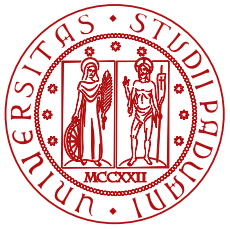


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

MASTER DEGREE IN CONTROL SYSTEMS ENGINEERING

Heterogeneous robots: Model Predictive Control for bearing-only formation and tracking

MASTER STUDENT

Elena Ferrari

ID 2021436

SUPERVISOR

Prof. Angelo Cenedese

University of Padova

Co-SUPERVISOR

Prof. Isabelle Fantoni

CNRS, École Centrale de Nantes

GRADUATION DATE 17TH OCTOBER 2022

ACADEMIC YEAR 2021 - 2022

To my parents
and friends

Abstract

Multi-agent systems are systems consisting of several autonomous robots that usually work under the assumption that they can communicate by sending and receiving the positions of other robots operating in the network.

The introduction of this type of systems is due to the fact that, in many situations, it is preferable to use more than one robot to achieve more complex goals without human help, especially in dangerous situations.

In this thesis, the focus is on a heterogeneous multi-agent system, i.e. a system whose component robots are heterogeneous with each other in terms of their implementation capabilities. In particular, a heterogeneous multi-agent system consisting of two UGVs and two UAVs is developed. In fact, although the two UGVs and the two UAVs have the same characteristics, the implementation capabilities of the UAV and those of the UGV are different.

The aim of the thesis is to maintain the formation and have the four agents follow a desired trajectory through a Leader-Follower approach based on the Bearing Rigidity Theory and implemented through Model Predictive Controllers. In the multi-agent system considered below, one of the two UAVs is assumed to be the leader, while all the others are followers. The leader's role is to plot the desired trajectory, while the followers must train and maintain the training even while following the leader's trajectory. However, the followers do not know the trajectory to be followed, nor the distance to the other agents and the leader.

Therefore, to solve this problem, a decentralised Leader-Follower with Bearing-Only control approach was implemented. In addition, Model Predictive controllers were chosen because this type of control makes it possible to prevent critical situations by solving an optimisation problem online at each instant of time so that the predicted output follows the reference.

The advantage of Model Predictive Control is that it is a multivariable controller that controls the outputs simultaneously, taking into account all the interactions between the system's variables, even if this involves adjusting numerous controller gains. Another advantage of Model Predictive Control is that it can handle constraints that play an important role in avoiding unintended consequences.

The proposed approach has been implemented in MATLAB and SIMULINK and the results obtained from the simulations will be discussed.

Sommario

I sistemi multi-agente sono sistemi composti da più robot autonomi che, solitamente, lavorano sotto il presupposto di poter comunicare inviando e ricevendo le posizioni degli altri robot che operano nella rete.

L'introduzione di questo tipo di sistemi è dovuta al fatto che, in molte situazioni, è preferibile utilizzare più di un robot per raggiungere obiettivi più complessi senza l'aiuto dell'uomo, soprattutto in situazioni di pericolo.

In questa tesi, l'attenzione è rivolta ad un sistema multi-agente eterogeneo, ovvero un sistema i cui robots che lo compongono sono eterogenei tra loro, in termini di capacità di attuazione. In particolare, viene sviluppato un sistema multi-agente eterogeneo composto da due UGV e due UAV. Infatti, anche se i due UGV e i due UAV hanno le stesse caratteristiche, le capacità di attuazione dell'UAV e quelle dell'UGV sono differenti.

L'obiettivo della tesi è quello di mantenere la formazione e di far seguire ai quattro agenti una traiettoria desiderata attraverso un approccio Leader-Follower basato sulla Bearing Rigidity Theory e implementato attraverso Model Predictive Controllers. Nel sistema multi-agente considerato in seguito, uno dei due UAV si assume essere il leader, mentre tutti gli altri sono followers. Il ruolo del leader è quello di tracciare la traiettoria desiderata, mentre i follower devono formare e mantenere la formazione anche durante l'inseguimento della traiettoria del leader. Tuttavia, i follower non conoscono la traiettoria da seguire, né la distanza dagli altri agenti e dal leader.

Quindi, per risolvere il problema, è stato implementato un approccio basato sul controllo decentralizzato Leader-Follower with Bearing-Only. Inoltre, è stato scelto di utilizzare controllori di tipo Model Predictive poichè questo tipo di controllo permette di prevenire le situazioni critiche, risolvendo online un problema di ottimizzazione ad ogni istante di tempo per far sì che l'uscita prevista segua il riferimento.

Il vantaggio del Model Predictive Control è che si tratta di un controllore multivariabile che controlla le uscite simultaneamente tenendo conto di tutte le interazioni tra le variabili del sistema, anche se ciò comporta la necessità di regolare numerosi guadagni del controllore. Un altro vantaggio del Model Predictive Control è che può gestire i vincoli che svolgono un ruolo importante per evitare conseguenze indesiderate.

L'approccio proposto è stato implementato in MATLAB e SIMULINK e i risultati ottenuti dalle simulazioni saranno discussi.

Contents

List of Figures **xi**

List of Tables **xiii**

List of Acronyms **xix**

1 Introduction **1**
1.1 State of art . . . 1
1.2 Thesis structure . . . 2
1.3 Notation . . . 3

2 From single agent to multi-agent systems **5**
2.1 Single Agent Model . . . 5
2.1.1 Agent categorization . . . 7
2.1.2 Kinematics and dynamics of an agent . . . 8
2.1.3 UGV: unmanned ground vehicle . . . 10
2.1.4 Example: UGV trajectory tracking . . . 15
2.1.5 UAV: unmanned aerial vehicle . . . 20
2.1.6 Example: UAV (quadrotor) trajectory tracking . . . 22
2.2 Heterogeneous Multi-Agent System . . . 27
2.2.1 Graph theory . . . 29
2.2.2 Task taxonomy and Multi-robot system workflow . . . 33

3 Types of controllers **35**
3.1 PID controller . . . 36
3.1.1 Proportional action . . . 36
3.1.2 Integral action . . . 37
3.1.3 Derivative action . . . 37
3.1.4 PID configurations . . . 38
3.2 MPC controller . . . 38
3.3 Leader-Follower Approach . . . 45

3.3.1 Leader-Follower approach based on distance 45
3.3.2 Example: Numerical results of the simulation 47

4 Bearing Rigidity Theory **51**
4.1 Bearing Rigidity Theory in $\mathbb{SE}(3)$ 57
4.2 Bearing Rigidity theory applied to a group of UAVs quadrotors . 59
4.3 Bearing Rigidity theory applied to a group of UGVs unicycles . . 61

5 Leader-Follower approach using MPC with Bearing **65**
5.1 Leader-Follower approach between the 2 UAVs 66
5.2 Leader-Follower approach between a UAV and a UGV 88
5.3 Leader-Follower approach between a UAV and two UGVs 95
5.4 Final results considering 2 UGVs and 2 UAVs 97
5.5 Adding the disturbance 98

6 Conclusions **103**

A Quaternions **105**

B Rotation matrices **109**

C Relation between quaternions and rotation matrices **113**

D Leader-Follower approach based on MPC with bearing between two UGVs **115**

List of Figures

2.1 *MRS architecture: robot, locally connected MRS, group of MRS connected through the cloud [**5**].* 8
2.2 *Example of World and Body Reference Frames [**8**].* 9
2.3 *Angles and vectors definitions.* 12
2.4 *Desired linear and angular velocities of the unicycle.* 16
2.5 *Reference trajectory of the UGV.* 17
2.6 *Tracking errors of the UGV.* 17
2.7 *Real linear and angular velocities of the UGV.* 18
2.8 *Wheel speeds and check of the saturation.* 19
2.9 *Comparison between the reference and the measured trajectory of the UGV.* 20
2.10 *UAV quadrotor cascaded control scheme.* 20
2.11 *Desired trajectory of the quadrotor.* 23
2.12 *Tracking errors of the quadrotor.* 24
2.13 *Desired angles of the quadrotor.* 24
2.14 *Errors on the ϕ, θ, ψ angles of the UAV.* 25
2.15 *Measured trajectory of the UAV.* 26
2.16 *Measured and desired trajectory of the UAV.* 27
2.17 *Examples of directed and undirected graphs.* 30
2.18 *Workflow for the multi-robot system [**5**].* 34

3.1 *Generalized scheme of a feedback control, considering a PID controller [**2**].* 35
3.2 *Generalized scheme of a PID controller [**39**].* 36
3.3 *Basic generalized scheme of a MPC controller [**34**].* 39
3.4 *General concept of a discrete MPC [**41**].* 43
3.5 *Graphical representation of the Leader-Follower approach and its parameters applied to two unicycles [**30**].* 46
3.6 *Leader-Follower trajectories obtained applying the Leader-Follower approach based on distance.* 48
3.7 *Follower velocities obtained applying the Leader-Follower approach based on distance.* 49

3.8 *Leader-Follower velocities obtained applying the Leader-Follower approach based on distance.* 50

4.1 *Examples of bearing rigid formations [**44**].* 52
4.2 *Examples of nontrivial infinitesimal bearing motions, that imply non-infinitesimal bearing networks [**44**].* 54
4.3 *Example of infinitesimal bearing networks [**44**].* 54

5.1 *Directed graph considered for the Leader-Follower approach between two UAVs.* 68
5.2 *UAV follower trajectory.* 78
5.3 *Comparison between the UAV leader and the UAV follower trajectories.* 78
5.4 *UAV follower thrust and torques.* 79
5.5 *UAV leader and UAV follower torques.* 80
5.6 *UAV leader and UAV follower thrust.* 81
5.7 *UAV follower linear velocities.* 82
5.8 *UAV leader and UAV follower linear velocities.* 83
5.9 *UAV follower angular velocities.* 84
5.10 *UAV leader and UAV follower angular velocities.* 85
5.11 *UAV leader and UAV follower trajectories in x, y and z directions.* . . 86
5.12 *UAV follower angles.* 87
5.13 *Directed graph considered for the Leader-Follower approach between a UAV and a UGV.* 90
5.14 *UGV Follower trajectory.* 93
5.15 *Leader-Follower trajectories: UAV leader and UGV follower.* 94
5.16 *UGV follower linear velocities.* 95
5.17 *UAV leader and UGV follower linear velocities.* 96
5.18 *UGV2 follower trajectory.* 97
5.19 *UAV leader and UGVs followers trajectories.* 98
5.20 *Final trajectories of the MAS without noise.* 99
5.21 *Comparison between the UAV leader trajectory and the first UGV follower trajectory with noise.* 100
5.22 *Comparison between the UAV leader trajectory and the UGVs followers trajectories with noise.* 101
5.23 *Final system result with noise.* 102

B.1 *World and Body Frames.* 109

D.1 *Undirect graph representing the MAS composed by two unicycles: node 1 represents the leader, while node 2 represents the follower.* 116

D.2 *Linear and angular velocities of the unicycle follower.* 118
D.3 *Comparison between the linear and angular velocities of the unicycle leader and follower.* 119
D.4 *Trajectory of the UGV follower.* 119
D.5 *Trajectories of the leader and follower UGV.* 120

List of Tables

2.1 *Unicycle's parameters.* 15
2.2 *Quadrotor's parameters.* 22

3.1 *Leader-follower approach's parameters.* 48

5.1 *Initial positions of the UAV leader and UAV follower.* 67
5.2 *Initial positions of the UAV leader and UGV1 follower.* 89
5.3 *Initial positions of the UAV leader and UGVs followers.* 95

D.1 *Coordinates of the initial pose of the unicycle follower.* 115

List of Acronyms

IBR Infinitesimally Bearing Rigid

L-F Leader-Follower

LQR Linear Quadratic Regulator

MAS Multi-Agent system

MIMO Multi input multi output system

MPC Model Predictive control

QP Quadratic Programming

SISO Single input single output system

UAV Unmanned aerial vehicle

UGV Unmanned ground vehicle

NLP Non Linear Problem

OCP Optimal Control Problem

IVP Initial Value Problem

1 Introduction

1.1 State of art

Multi-agent systems (MAS) are systems composed by multiple agents and they have found a lot of applications in the last years since there exist many different types of robots and a single agent cannot be used or perform well in all the circumstances. Hence, MASs are used to solve more complex tasks, in dangerous scenarios but also in market simulation, monitoring, system diagnosis and remedial actions [19][42].

In details, heterogeneous MAS are characterized by the diversity of the robots in terms of capabilities such that each one can contribute in a different way. Due to the different capabilities, a key element in multi-robots system (MRS) is to assign tasks to robots in order to reach a meaningful division. So, the goal of task allocation is to find robot-task assignments such that the overall utility is maximized [11].

In order to control an heterogeneous MRS, the new position feedback based formation control was though. It is possible to identify three conventional methods of formation control: *Behavior-Based Strategy*, *Leader-Follower Approach*, *Potential Field Approach* [24].

Between these, in the thesis, the Leader-Follower approach plays a central role since, using it, controlling multiple robots in a desired formation is easy and suitable for describing the formation of robots even though it is difficult to consider the ability gap in heterogeneous robot teams. Such kind of approach can be easily implemented based on distance, measuring the relative positions of the neighbors using GPS and shared via wireless network.

Despite this, the GPS does not satisfy the high accuracy requirements of

the formation control tasks and cannot be used indoors, underwater or in deep space. Hence, to sense the neighbors, it is preferable to use onboard sensors such as optical cameras for ground and aerial vehicles. These sensors are bearing-only sensors and, since it is easy for vision to measure bearings, but relatively difficult to obtain accurate range information, vision can be effectively modeled as a bearing-only sensing approach in MAS formation control. So, here it is implemented the Leader-Follower approach based on MPC with bearing, instead of distance.

Note that the project will be about four robots: two unicycles and two quadrotors so that the resulting system is heterogeneous. In order to control this MAS, the Leader-Follower approach based on MPC with bearing is implemented, where the MPC is used to indicate the Model Predictive Control.

During the last years, MPC was studied a lot and it is an advanced popular control method. MPC is used in a lot of different applications thanks to its easily way to include constraints in the model and to solve nonlinear systems. The basic idea of MPC is to predict the future behaviour of the controlled system over a finite time horizon and compute an optimal control input that minimizes an a priori defined cost functional while ensuring satisfaction of given system constraints.

1.2 Thesis structure

The thesis is structured in 5 chapters. The first four chapters are about the theoretical notions used to implement the thesis' project with some examples in the between used to better understand the notions. Instead, the fifth chapter is about the project implementation and results obtained. In particular:

- in Chapter 2, two main notions are presented: the concept of agent and the concept of multi-agent system. In details, about the concept of agent, the dynamics and kinematics of a rigid body are demonstrated, with particular attention to the UGV and UAV tracking models. About the tracking models of the UGV and the UAV, two relative examples are also shown, one about the unicycle and the other one about the quadrotor. Then, the basic notions of graph theory are reported since it constitutes the way used to represent a MAS.

- in Chapter 3, the different kinds of controllers used in the project are presented. About the tracking of a single agent, the PID controller is presented. Then, also the Leader-Follower approach is introduced since it

is used in the project, where the cooperation between two UGVs and two UAVs has to be realized.

- in Chapter 4, the bearing rigidity theory is explained, with particular attention to this theory applied to the case of the unicycle and of the quadrotor, being the two types of agents considered in the project. This theory has been introduced because the main goal of the project consists in realizing the cooperation between the agents without using the distance between them, but just the bearing.

- in Chapter 5, the project implementation and results are presented. The project is about the cooperation between two unicycles and two quadrotors. About these four agents, just one of them is the leader, while the others are followers that have to follow the same trajectory of the leader without knowing it and the distance between themselves.

- in Chapter 6, the conclusions relative to the project and the suggestions for future works are given.

1.3 Notation

In the thesis, the vectors belonging in $\mathbb{R}^n$ are denoted with a bold lowercase letter, while the matrices in $\mathbb{R}^{n \times m}$ are indicated with a bold uppercase letter.

The sets, the graph and the matrices used to describe it (Node degree matrix, Adjacency matrix, Incidence matrix, Laplacian matrix), other than the Bearing Laplacian matrix are represented through calligraphic letters.

Then, particular notations to represent a matrix are: $\mathbf{0}_{n \times m}$ used to represent a null matrix of dimension $n \times m$ and $\mathbf{I}_n$ which represents an identity square matrix of dimension n.

$\mathbf{1}_n \in \mathbb{R}^4$ represents the unit quaternion vector, while $\otimes$ denotes the Kronecker product.

2

From single agent to multi-agent systems

Designing an autonomous robot that can perform well in all the circumstances is difficult because of the dynamical and unpredictable nature of the robot. Due to this and to the variety of robots in terms of shapes, sizes and capabilities, it is possible to solve efficiently more complex tasks creating a robotic system. In fact, it is usually preferable to use a more complex robotic system rather than requiring human-robot cooperation, especially in dangerous scenarios such as search and rescue missions.

2.1 Single Agent Model

In the literature, there exist multiple definitions of "agent". One of these is:

Definition 2.1.1 (Agent) [1] *A single intelligent agent is a physical (robot) or virtual (software program) entity that can autonomously perform actions on an environment while perceiving this environment through the sensors connected to it, to accomplish a goal. A rational agent seeks to perform actions that result in the best outcome. A cognitive architecture is the "underlying infrastructure for an intelligent agent": the agents brain. It consists of perception, reasoning, learning, decision making, problem solving, interaction and communication.*

or, another one is:

[1] Awad M., Rizk Y., Tunstel E., *Cooperative Heterogeneous Multi-Robot Systems: A Survey*, 20 March 2019

Definition 2.1.2 (Agent) [2] *An agent is an encapsulated computational system that is situated in some environment and this is capable of flexible, autonomous action in the environment in order to meet its design objective.*

From these, it is possible to observe that the concept of "agent" is rather generic. Hence, a more generalized definition based on the main abilities and features of the agents is exploited in Def.2.1.3.

Definition 2.1.3 (Agent) [3] *An agent is an entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. The entity performs the necessary action on the environment based on this decision.*

Looking at Def.2.1.3, five keywords are identified:

1. Entity: it refers to the type of the agent, which can be an hardware, software or a combination of both.

2. Environment: it refers to the place where the agent is located and the information sensed from the environment by the agent is used for decision making. The complexity of an agent-based system is affected by multiple features which are related to the environment, such as:

   - Accessibility: it refers to the accuracy with which agents can sense data from the environment.
   - Determinism: it refers to the predictability of the results of an action.
   - Dynamism: it refers to the changes that occur in the environment that are independent of the actions taken by the agents.

3. Parameters: they refer to the different types of data that an agent can sense from the environment.

4. Action: it refers to a change in the environment. An agent can perform a set of discrete or continuous actions. In a continuous set of actions, the agent can perform unlimited actions while, in a discrete set of actions, the agent can perform finite set of actions.

[2] N.R.Jennings, *On agent-based software engineering*, 21 September 1999

[3] Dorri A., Salil S. Kanhere, Jurdak R., *Multi-Agent Systems: A Survey*, pg. 2, in School of Computer Science and Engineering, University of New South Wales, June 19, 2018, Sydney, Australia.

5. Continuity: the environment can be classified as: continuous or discrete. In particular, a continuous environment affects the agent's state through a continuous function while a discrete environment constraints the agent to follow a set of known states.

Then, in order to solve more complex tasks, other agents features are:

- Sociability: agents share knowledge and information with each other to reach the goal.
- Autonomy: each agent can independently execute the decision making process.
- Proactivity: each agent can predict the possible future actions knowing its history, sensed parameters, and information of others agents.

Due to this multitude of characteristics, the agent's evolution is based on domain specific performance measures, generality, versatility, rationality, optimality, efficiency, scalability, autonomy and improvability.

2.1.1 Agent categorization

It is possible to identify a lot of agent categorizations. A first classification distinguishes three types of agents, whose workflow is shown in Fig.2.1:

- reactive agents: they just react to environmental changes. Their workflow is composed by sense (S) and act (A);
- deliberative agents: they initiate actions based on planning, with no external activation. In this case the workflow contains sense (S), plan (P) and act (A), such that it is called "sense-plan-act" or "sense-model-plan-act";
- hybrid agents: they are agents that can perform actions after a planning algorithm or reacting to current perceptions.

Instead, a more accurate categorization classifies agents into:

- simple reflex agents: the action performed is due to a sensory input;
- model-based reflex agents: the internal state of the environment is kept;
- goal-based agents: actions are performed to reach a goal;
- utility-based agents: a utility function is maximized.

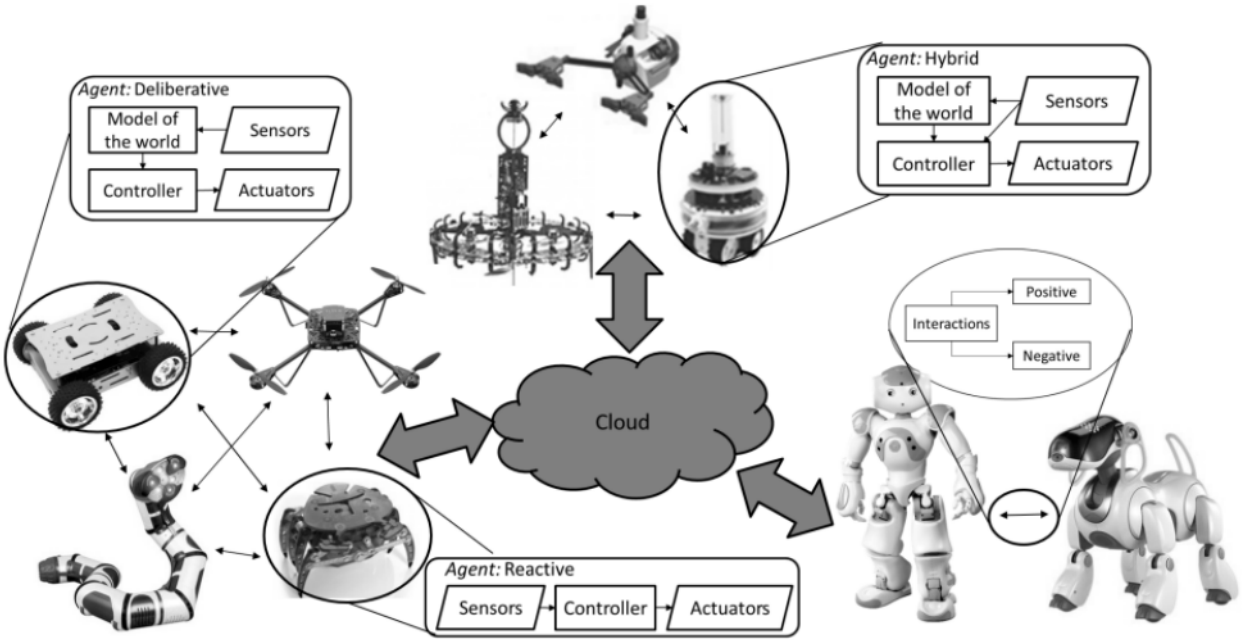


Figure 2.1: *MRS architecture: robot, locally connected MRS, group of MRS connected through the cloud [**5**].*

2.1.2 Kinematics and dynamics of an agent

A single agent acting in the 3D space is considered as a rigid body and its spatial displacement can be described with respect to two different reference frames:

- the Body Frame (also called Local Frame) $\mathcal{F}_i$, whose origin O_i coincides with the center of mass of the agent and whose axes are identified by the unit vectors $\{e_1, e_2, e_3\}$, which define the canonical basis of $\mathbb{R}^3$;
- the Inertial World Frame $\mathcal{F}_w$ that is fixed, unknown and common to all agents.

Looking at Fig.2.2, (x_W, y_W, z_W) is used to describe the Inertial World Frame, while (x_a, y_a, z_a), (x_o, y_o, z_o) and (x_p, y_p, z_p) indicate the Body Frames of the rigid bodies a, o and p, respectively.

Each agent is characterized by a maximal number of controllable DoFs (cDoFs) that is equal to 6. Between these 6 cDoFs, 3 are for the translation and 3 are for the rotation.

In particular, when:

- $\mathcal{V}_i = \Omega_i = \mathbb{R}^3$, the agent is fully actuated and, hence, it can translate and rotate in any direction of the 3D space having 3 translational and 3 rotational cDoFs;

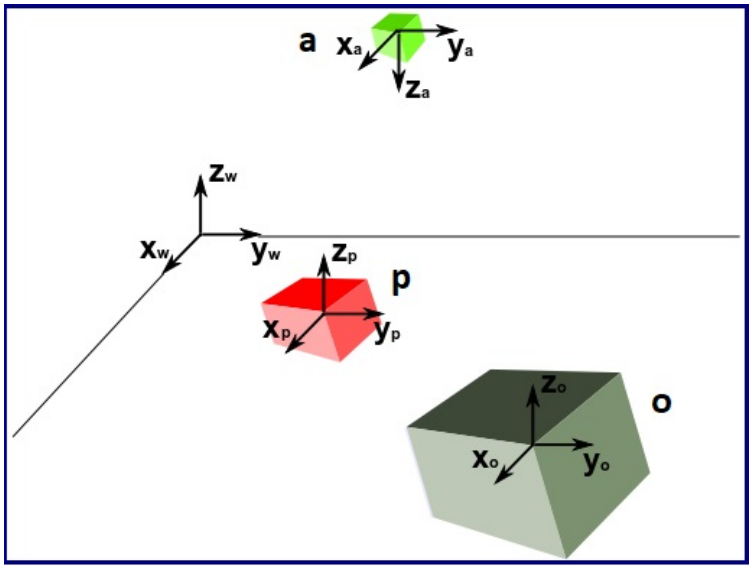


Figure 2.2: *Example of World and Body Reference Frames [8].*

- when $\mathcal{V}_i$ or $\Omega_i \subset \mathbb{R}^3$, the agent is under actuated and its movement is constrained only in some direction (it has less than 6 DoFs).

Indicating with $c_i \in \{0...6\}$ the agent cDoFs, it is suitable to introduce the agent commands vector δ_i belonging to the agent instantaneous variation domain $\mathcal{I}_i \subseteq \mathbb{R}^{c_i}$, which specifies the agent actuation capabilities through the selection map:

$$\mathcal{S}_i : \mathcal{I}_i \rightarrow \mathcal{V}_i \times \Omega_i \qquad \delta_i \mapsto [v_i^T \quad \omega_i^T]^T \tag{2.1}$$

At this point, it is possible to consider the decoupling hypothesis, in regard to the agent translation and rotation movements, that is formally expressed as in 2.1.1.

Assumption 2.1.1 *Any $i-th$ agent can provide decoupled translation and rotation commands, meaning that δ_i in 2.1 is made up of two components that can be independently assigned. Formally, $\delta_i = [\delta_{p,i}^T \quad \delta_{o,i}^T]^T \in \mathcal{I}_i = \mathcal{I}_{p,i} \times \mathcal{I}_{o,i}$ with $\delta_{p,i}$ and $\delta_{o,i}$ respectively associated to the agent linear and angular velocity $\mathcal{I}_{p,i}$ and $\mathcal{I}_{o,i}$ representing the $i-th$ agent instantaneous position and orientation variation domains.*

Considering Assumption 2.1.1, the agent total number of cDoFs results:

$$c_i = c_{t,i} + c_{r,i}$$

with:

- $c_{t,i} = dim(\mathcal{I}_{p,i}) = dim(\mathcal{V}_i)$ that represents the translational cDoFs;
- $c_{r,i} = dim(\mathcal{I}_{o,i}) = dim(\Omega_i)$ that indicates the rotational cDoFs.

Doing so, the selection map in Eq.2.1 can be splitted into the terms:

$$\mathcal{S}_{P,i} : \mathcal{I}_{P,i} \rightarrow \mathcal{V}_i \qquad \delta_{P,i} \mapsto v_i$$

$$\mathcal{S}_{O,i} : \mathcal{I}_{O,i} \rightarrow \Omega_i \qquad \delta_{O,i} \mapsto \omega_i$$

where $\mathcal{S}_{P,i}$ is bijective.

In details, considering the real world scenarios, the structure of the maps $\mathcal{S}_{P,i}$ and $\mathcal{S}_{O,i}$ is assumed as in 2.1.2.

Assumption 2.1.2 *The maps $\mathcal{S}_{P,i}$, $\mathcal{S}_{O,i}$ are linear, hence it is:*

$$v_i = \mathcal{S}_{P,i}(\delta_{P,i}) = \mathcal{S}_{P,i}\delta_{P,i} \qquad \mathcal{S}_{P,i} \in \mathbb{R}^{3\times c_{P,i}}$$

$$\omega_i = \mathcal{S}_{O,i}(\delta_{O,i}) = \mathcal{S}_{O,i}\delta_{O,i} \qquad \mathcal{S}_{O,i} \in \mathbb{R}^{3\times c_{O,i}}$$

2.1.3 UGV: UNMANNED GROUND VEHICLE

About the unmanned ground vehicles, the unicycle model is considered in the following.

The unicycle is a planar vehicle with only one orientable wheel and its pose in the World Frame can be described through the vector:

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \theta \end{bmatrix} \in \mathbb{R}^n \qquad with \quad i = 1, ..., n \tag{2.2}$$

where the first two components ($[x, y]^T = \mathbf{p} \in \mathbb{R}^2$) indicate the position of robot at time t in Cartesian coordinates, while the third component ($\theta \in (-\pi, \pi], \theta \in \mathbb{S}^1$) is the heading direction of the robot in the World Frame.

Alternatively, in order to describe the agent pose, it is possible to use the vector $\mathbf{x}_i(t) = [\mathbf{p}_i^T \quad \mathbf{q}_i^T]^T \in \mathbb{R}^3 \times \mathbb{S}^3$, where:

- the vector $\mathbf{p}_i \in \mathbb{R}^3$ describes the coordinates of O_i in $\mathcal{F}_w$;
- the unit quaternion[4] $\mathbf{q}_i = [\eta_i \quad \epsilon_i^T]^T \in \mathbb{S}^3$ defines the rotation of $\mathcal{F}_i$ with respect to $\mathcal{F}_w$.

[4]For more details about the quaternions, read the Appendix A

At this point, knowing the pose, it is possible to derive the linear ($\dot{\mathbf{p}}_i$) and angular ($\dot{\mathbf{q}}_i$) velocities, both expressed in the Body Frame and considered as controllable variables of the single agent. So, the single agent kinematics is described by the relations 2.3:

$$\begin{aligned} \dot{\mathbf{p}}_i &= \mathbf{R}_i v_i \\ \dot{\mathbf{q}}_i &= \frac{1}{2}\mathbf{M}(\mathbf{q}_i)\omega_i \end{aligned} \tag{2.3}$$

where:

- $v_i \in \mathcal{V}_i \subseteq \mathbb{R}^3$ is the linear velocity of O_i with respect to $\mathcal{F}_w$;
- $\omega_i \in \Omega_i \subseteq \mathbb{R}^3$ is the angular velocity of the Body Frame $\mathcal{F}_i$ with respect to $\mathcal{F}_w$;
- $\mathbf{R}_i \in \mathbb{SO}(3)$[5] (that is the special orthogonal group) indicates the rotation matrix associated to $\mathbf{q}_i$;
- $\mathbf{M}(\mathbf{q}_i) \in \mathbb{R}^{4\times3}$ maps the agent angular velocity into the time derivative of its quaternion based on orientation.

Under the assumption of non-holonomic pure rolling constraint, i.e. $\Phi(\mathbf{q}, \dot{\mathbf{q}}) = 0$, and considering the driving (linear) velocity and the steering (angular) velocity, the kinematic model of the unicycle takes the form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.4}$$

Considering these relations, the control inputs, i.e. the velocities v and ω, are obtained from the the angular speed ω_R and ω_L of the wheels:

$$v = \frac{r(\omega_R + \omega_L)}{2} \qquad \omega = \frac{r(\omega_R - \omega_L)}{d} \tag{2.5}$$

where r represents the wheels radius, while d is the distance between the centers of the wheels.

About the unicycle, three different types of controllers are presented:

- Cartesian control, which drives the unicycle to a desired configuration in terms of position $\mathbf{p}$ regardless the orientation θ.

[5]For more details about the rotation matrices, read the Appendix

In this case, without loss of generality, the following desired target configuration is considered:

$$\mathbf{q}^{des} = \begin{bmatrix} \mathbf{p}^{des} \\ \forall \end{bmatrix} = \begin{bmatrix} x^{des} \\ y^{des} \\ \forall \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \forall \end{bmatrix} \tag{2.6}$$

Then, with respect to the current position (x, y) and orientation θ, the position error vector $\mathbf{e}_P$ and the unit vector sagittal axis $\mathbf{n}$ are defined as in Eq.2.7, while their graphical meaning is presented in Fig.2.3.

$$\mathbf{e}_P = \begin{bmatrix} 0 - x \\ 0 - y \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix} \qquad \mathbf{n} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{2.7}$$

Hence, the projection of the error on the sagittal axis is:

$$< \mathbf{e}_P, \mathbf{n} >= \mathbf{e}_P^T \mathbf{n} = -x\cos(\theta) - y\sin(\theta) \tag{2.8}$$

instead, about the angles, it holds that:

$$\alpha = atan2(y, x) \qquad \delta = \alpha + \pi \qquad \gamma = \delta - \theta = atan2(y, x) + \pi - \theta \tag{2.9}$$

and the aim of the controller is to take the projection and the angle γ to zero.

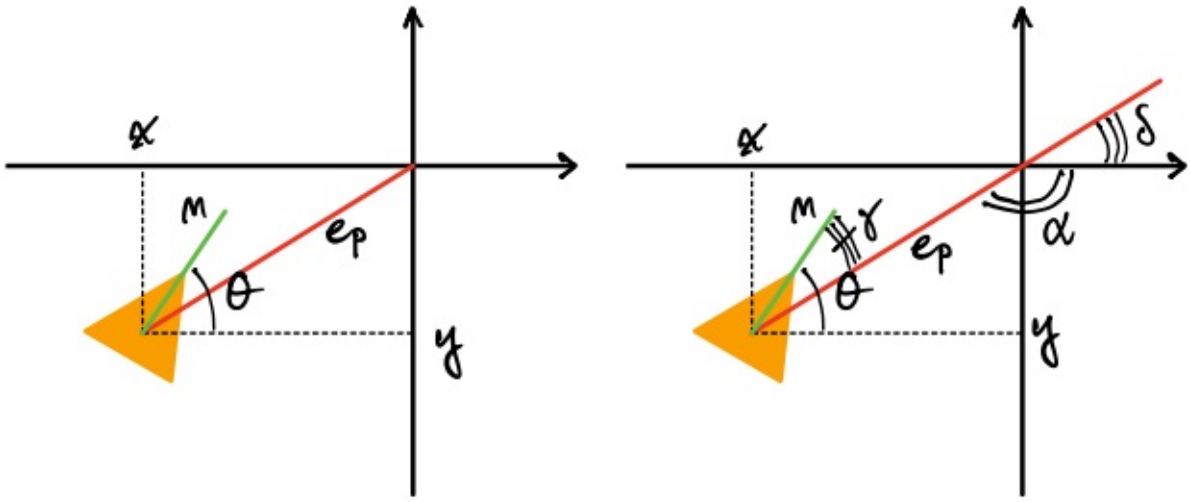


Figure 2.3: *Angles and vectors definitions.*

In order to reach the goal, the adopted control law is shown in Eq.2.10 and acts on the driving and steering velocity.

$$v = k_v \mathbf{e}_P^T \mathbf{n} = k_v(-x\cos(\theta) - y\sin(\theta)) \qquad \omega = k_\omega \gamma = k_\omega(atan2(y,x) + \pi - \theta) \tag{2.10}$$

with k_v and k_ω positive constant gains chosen through a trial and error approach by the user.

- Posture control, which aim is to drive the unicycle to a desired configuration in terms of full position $\mathbf{p}$ and orientation θ.

  In this case, without loss of generality, the desired target configuration considered is expressed as in Eq.2.11.

$$\mathbf{q}^{des} = \begin{bmatrix} \mathbf{p}^{des} \\ \theta^{des} \end{bmatrix} = \begin{bmatrix} x^{des} \\ y^{des} \\ \theta^{des} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.11}$$

  With respect to the current position (x, y) and orientation θ, the position error vector $\mathbf{e}_P$ and the unit vector sagittal axis $\mathbf{n}$ are defined as in Eq.2.12.

$$\mathbf{e}_P = \begin{bmatrix} 0 - x \\ 0 - y \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix} \qquad \mathbf{n} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{2.12}$$

  The length of the error vector $\mathbf{e}_P$, i.e. $\rho = \sqrt{x^2 + y^2}$, together with the angles γ and δ as defined above, allow to obtain a polar representation of the unicycle pose. The aim of this controller is to bring these polar quantities to zero. Hence, a possible control law is:

$$v = k_v \rho \cos(\gamma) \qquad \omega = k_\omega \gamma + k_v \frac{\sin\gamma\cos\gamma}{\gamma}(\gamma + k_\delta \delta) \tag{2.13}$$

- Tracking control, which aim is to drive the unicycle in such a way it follows a desired trajectory that is a function of time. In trajectory tracking approach, the feasible desired and dependent on time trajectory to track has to be known before the UGV starts to move to the desired pose:

$$\mathbf{q}^{des}(t) = \begin{bmatrix} \mathbf{p}^{des}(t) \\ \theta^{des}(t) \end{bmatrix} = \begin{bmatrix} x^{des}(t) \\ y^{des}(t) \\ \theta^{des}(t) \end{bmatrix} \tag{2.14}$$

In trajectory tracking, one goal is to minimize the difference between the reference state vector $\mathbf{q}^{des}$ and the current state vector $\mathbf{q}$ which is called tracking error and defined in the World Frame as in Eq.2.15.

$$\mathbf{e}_W = \begin{bmatrix} x^{des} - x \\ y^{des} - y \\ \theta^{des} - \theta \end{bmatrix} \tag{2.15}$$

or as in Eq.2.16 if defined in the Body Frame.

$$\mathbf{e}_B = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \mathbf{R}_z^T(\theta)\mathbf{e}_W = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{e}_W \tag{2.16}$$

Hence, deriving Eq.2.16 and considering the invertible input transformation:

$$v = v^{des}\cos(e_3) - u_1 \qquad \omega = \omega^{des} - u_2 \Rightarrow u_1 = -v + v^{des}\cos(e_3) \qquad u_2 = \omega^{des} - \omega \tag{2.17}$$

it holds that:

$$\dot{\mathbf{e}}_B = \begin{bmatrix} 0 & \omega^{des} & 0 \\ -\omega^{des} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e}_B + \begin{bmatrix} 0 \\ \sin(e_3) \\ 0 \end{bmatrix} v^{des} + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.18}$$

Linearizing it around the tracking error where $\mathbf{e} = 0$ into Eq.2.19

$$\dot{\mathbf{e}}_B = \begin{bmatrix} 0 & \omega^{des} & 0 \\ -\omega^{des} & 0 & v^{des} \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e}_B + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.19}$$

a possible control law results as in Eq.2.20.

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ 0 & -k_2 & -k_3 \end{bmatrix} \mathbf{e}_B \tag{2.20}$$

In this way, the dynamics of the closed-loop error results to be:

$$\dot{\mathbf{e}}_B = \begin{bmatrix} -k_1 & \omega_{des} & 0 \\ -\omega_{des} & 0 & v^{des} \\ 0 & -k_2 & -k_3 \end{bmatrix} \mathbf{e}_B \tag{2.21}$$

where k_1, k_2, k_3 are positive control gains chosen in such a way that the eigenvalues of the error dynamics matrix asymptotically converge to zero.

2.1.4 Example: UGV trajectory tracking

In this example, the implementation of a UGV trajectory tracking is shown and the parameters of the unicycle are defined in Tab.2.1.

Parameter's name	Notation	Value
wheel radius	r	$0.02\ [m]$
wheel distance	d	$0.2\ [m]$
time constant low pass filter	T_d	$0.01\ [s]$
maximum wheel speed	ω_{max}	$20\ [rad/s]$
minimum wheel speed	ω_{min}	$-20\ [rad/s]$
desired x initial position	x^{des}	$0\ [m]$
desired y initial position	y^{des}	$0\ [m]$
desired θ initial position	θ^{des}	$0\ [rad]$
real x initial position	x^{real}	$0.10\ [m]$
real y initial position	y^{real}	$0.10\ [m]$
real θ initial position	θ^{real}	$0\ [rad]$

Table 2.1: *Unicycle's parameters.*

First of all, the desired trajectory must be defined. In the shown solution, the desired trajectory is found starting from the desired linear and angular velocities, imposing different values for them depending on time in order to obtain a curvilinear trajectory and not just a straight line or a circular path.

The shown results refer to a constant linear velocity ($v^{des} = 0.05\ m/s$) and a time varying angular velocity. In details, the angular velocity varies depending on time so that:

- from 0 to 10 s: it is equal to 0 rad/s;
- from 10 to 20 s: it is equal to 0.02 rad/s, which implies a curve on the left;
- from 20 to 30 s: it is equal to 0 rad/s;
- from 30 to 40 s: it is equal to -0.02 rad/s, which implies a curve on the right;

- from 40 to 50 s: it is equal to 0 rad/s;

Hence, the desired values of the linear and angular velocities so obtained are represented in Fig.2.4.

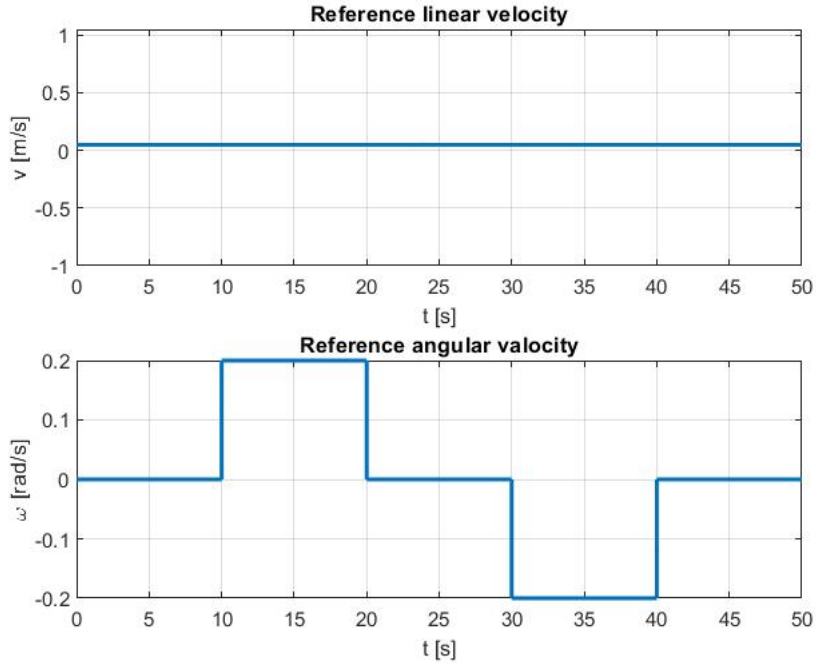


Figure 2.4: *Desired linear and angular velocities of the unicycle.*

Knowing the desired linear and angular velocities, it is possible to compute the desired trajectory of the unicycle reversing the kinematic model of the UGV in Eq.2.4.

In fact, integrating the desired angular velocity, the heading angle is obtained. Knowing the desired linear velocity and the heading angle, all the variables needed to compute the position (x, y) are given.

The resulting desired trajectory to track is reported in Fig.2.5.

Then the tracking errors, defined as the difference between the desired values of the pose of the UGV minus the measured ones as in Eq.2.22, in the Body Frame have to be computed.

$$\begin{aligned} e_x &= x^{des} - x \\ e_y &= y^{des} - y \\ e_\theta &= \theta^{des} - \theta \end{aligned} \tag{2.22}$$

Even if these errors have to be evaluated in the Body Frame, their behaviours in the World Frame are shown in Fig.2.6 in order to observe their convergence to

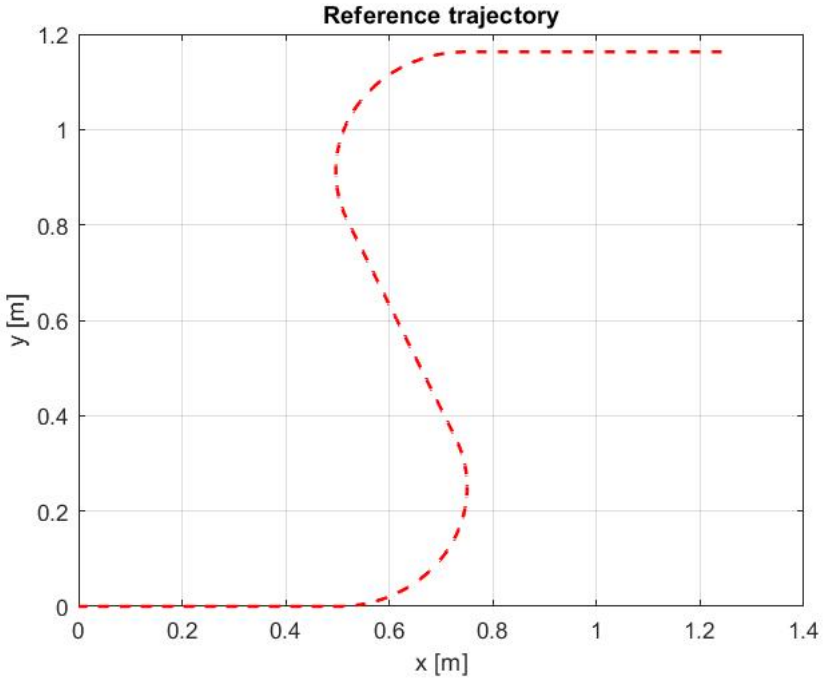


Figure 2.5: *Reference trajectory of the UGV.*

zero, according with the expectations since the measured values have to coincide with the desired ones.

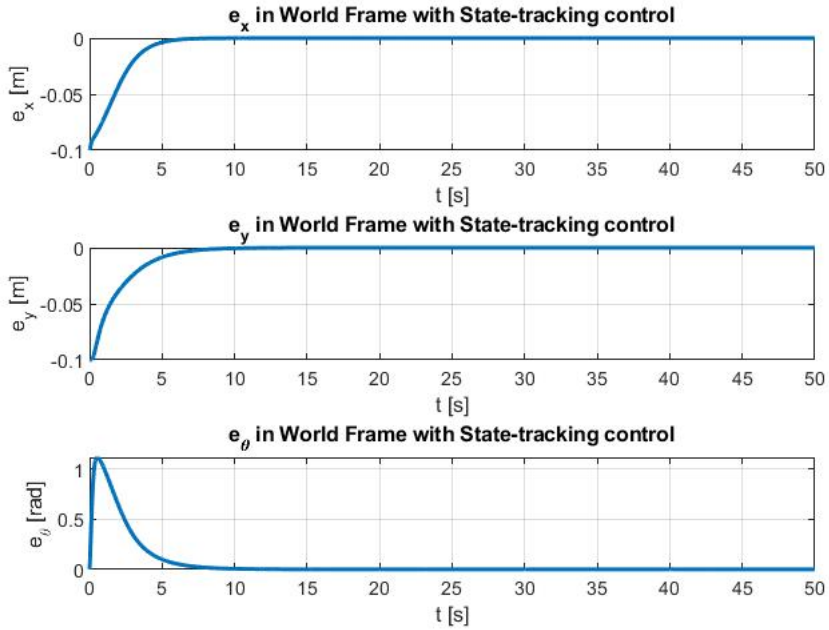


Figure 2.6: *Tracking errors of the UGV.*

From the errors defined in the Body Frame, the third component related to the heading angle is extracted and a PID state-tracking controller is applied to

the whole errors' vector according with Eq.2.20.

So, knowing the control inputs vector, the desired linear and angular velocities and the third component of the error, a change of input has to be performed in order to obtain the measured values of the linear and angular velocities. This is performed following the formula Eq.2.23.

$$\begin{cases} v = v^{des}\cos(e_3) - u_1 \\ \omega = \omega^{des} - u_2 \end{cases} \tag{2.23}$$

The measured values of the UGV velocities are shown in Fig.2.7

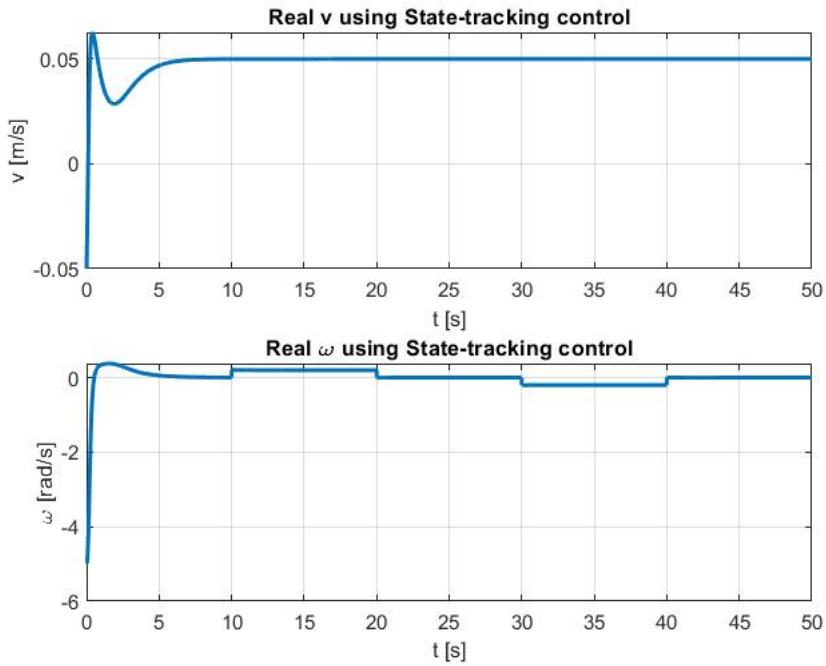


Figure 2.7: *Real linear and angular velocities of the UGV.*

Looking at this plot, it is possible to observe that the measured values converge to the desired ones in almost 5 s.

After having performed the change of input and obtained the real values of the velocities, the wheel speeds saturation must be verified because, if saturation occurs, it means that the model cannot be implemented in a real simulation.

As it is possible to observe from Fig.2.8, saturation occurs due to the chosen initial condition of the UGV and to the fact that, at the beginning, it has null linear and angular velocities.

In the end, knowing the real linear and angular velocity, the measured pose of the UGV is given by reversing and integrating Eq.2.4, as previously done for the desired trajectory.

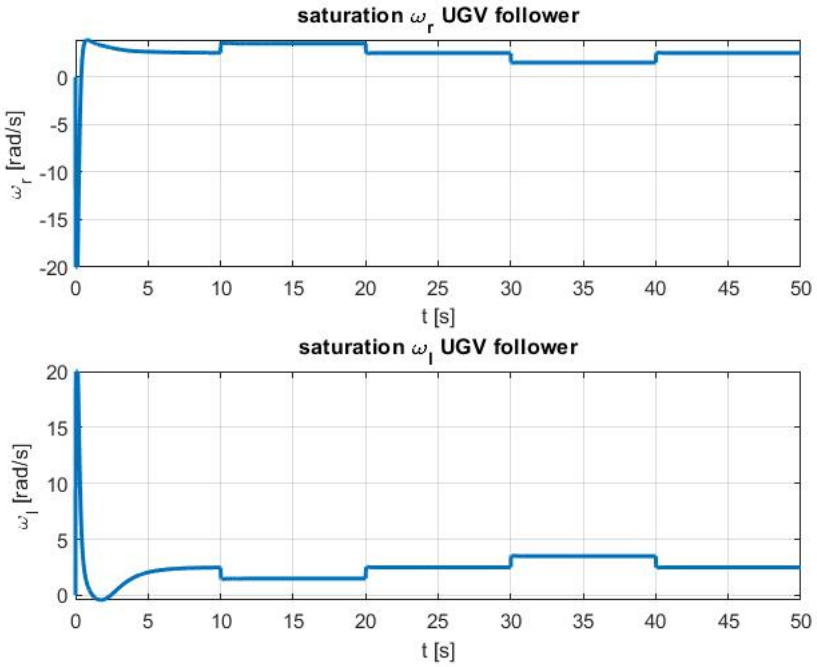


Figure 2.8: *Wheel speeds and check of the saturation.*

The plot of the real trajectory is shown in Fig.2.9. Looking at this, it is possible to note that the PID controller implemented performs well since the measured trajectory reaches quickly the desired values and they coincide except at the beginning due to the different established initial poses, i.e. the desired and the real ones.

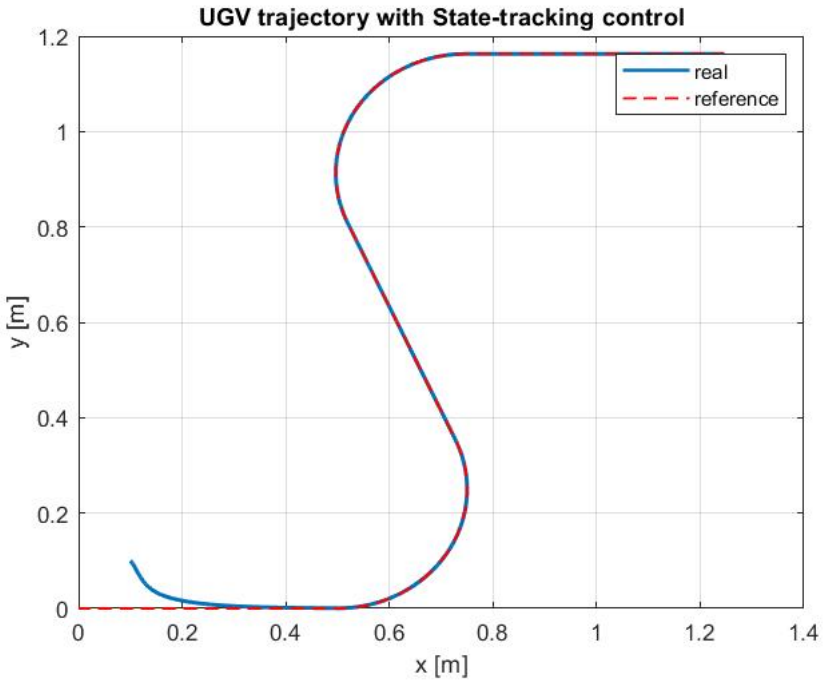


Figure 2.9: *Comparison between the reference and the measured trajectory of the UGV.*

2.1.5 UAV: UNMANNED AERIAL VEHICLE

About the UAVs, the case of quadrotor is considered. First of all it is necessary to observe that the quadrotor has only four degrees of freedom, from which its underactuated structure derives. In details, the three positions and the yaw angle are considered, while the roll and pitch angle are given respectively by the position (x, y) controller shown in Fig.2.10.

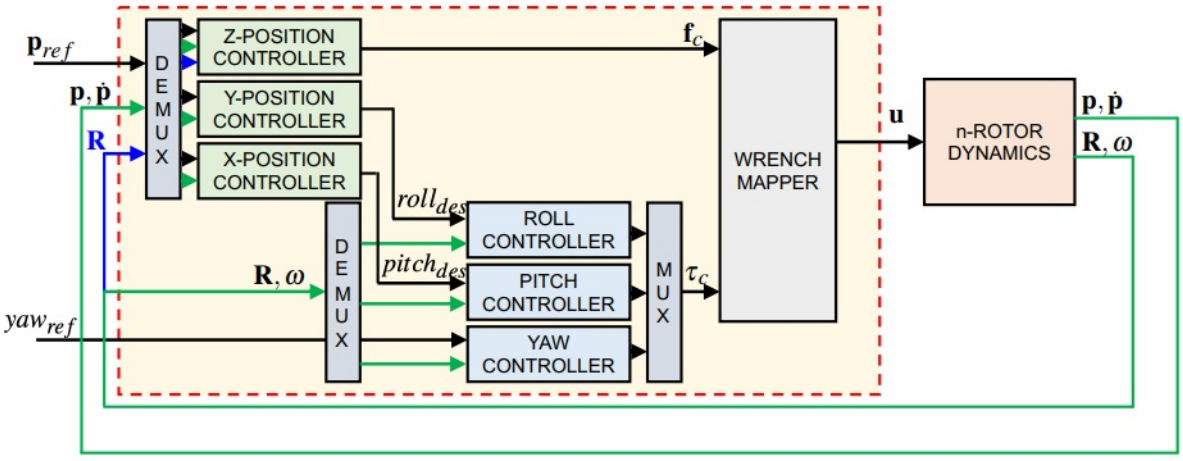


Figure 2.10: *UAV quadrotor cascaded control scheme.*

A planar quadrotor can be modeled as a 3D rigid body subjects to the gravi-

tational force and the thrusts generated by its four propellers. As for every rigid body, two reference frames are considered: $\mathcal{F}_W$, the common inertial reference frame, and $\mathcal{F}_B$, the reference frame attached to the drone and centered in its center of mass. The body to world transformation is expressed through the rotation matrix $\mathbf{R}(\alpha) \in \mathbb{SO}(3)$ and it is composed by the concatenation of the three basic rotations.

$$\mathbf{R}(\alpha) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{2.24}$$

Hence, the quadrotor state can be defined as $\mathbf{x} = [p^T \quad \alpha^T \quad v^T \quad \omega^T]^T$ and it is subject to the following differential equation $\dot{x} = f(x, u, f_d)$:

$$\begin{aligned} \dot{p} &= v \\ \dot{\alpha} &= \mathbf{W}^{-1}\omega \\ m\dot{v} &= -\begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathbf{R}(\alpha)\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + f_d \\ \mathbf{J}\dot{\omega} &= -\omega \times \mathbf{J}\omega + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \end{aligned} \tag{2.25}$$

where v is the quadrotor velocity in the Inertial Frame, ω is the angular rate in the Body Frame, g is the gravitational acceleration, m is the quadrotor mass, $\mathbf{J}$ is the quadrotor inertia matrix expressed in the Body Frame and f_d is an unknown disturbance thrust.

Considering such a model, the inputs are the common thrust T, which is applied along the z-axis of the Body Reference Frame, and the torques τ_x, τ_y, τ_z applied on the Body Frame. These inputs can be uniquely obtained from the four propeller spinning rates squared $\mathbf{u}^2 = [\omega_1^2 \quad \omega_2^2 \quad \omega_3^2 \quad \omega_4^2]^T$ through Eq.2.26.

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = Fu^2 = c_f \begin{bmatrix} 1 & 1 & 1 & 1 \\ p_{1,y} & p_{2,y} & p_{3,y} & p_{4,y} \\ -p_{1,x} & -p_{2,x} & -p_{3,x} & -p_{4,x} \\ c_1 c_\tau & c_2 c_\tau & c_3 c_\tau & c_4 c_\tau \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.26}$$

where c_f is the propeller thrust constant, c_τ is the propeller drag/thrust coefficient, $c_i = 1$ if the $i-th$ propeller spins clockwise and $c_i = -1$ otherwise and $\mathbf{p}_{i,\{x,y\}}$ is the (x, y) - coordinate of the $i-th$ propeller spinning centre with respect to the quadrotor center of mass expressed in the Body Reference Frame.

2.1.6 Example: UAV (quadrotor) trajectory tracking

In this example, the implementation of a trajectory tracking for a quadrotor is shown and the quadrotor parameters are defined in Tab.2.2.

Parameter's name	Notation	Value
gravitational acceleration	g	$9.81\ [m/s^2]$
mass	m	$m = 1.5\ [Kg]$
inertia matrix	J	$\begin{bmatrix} 0.029125 & 0 & 0 \\ 0 & 0.029125 & 0 \\ 0 & 0 & 0.055225 \end{bmatrix} [Kg * m^2]$
quadrotor's geometry	p_{mot}	$\begin{bmatrix} 0.13 & -0.22 & 0 \\ -0.13 & 0.2 & 0 \\ 0.13 & 0.22 & 0 \\ -0.13 & -0.2 & 0 \end{bmatrix}$
rotors' maximum spinning rate		1100 [rad/s]
thrust coefficient	c_f	$5.84 \times 10^{-6}\ [N/s^2]$
torque coefficient	c_τ	$0.06 c_f\ [Nm/s^2]$
rotors' direction		$\begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix}$
force matrix	F	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 5.84 \times 10^{-6} & 5.84 \times 10^{-6} & 5.84 \times 10^{-6} & 5.84 \times 10^{-6} \end{bmatrix}$
moments' matrix	M	$\begin{bmatrix} -1.2848 \times 10^{-6} & 1.1680 \times 10^{-6} & 1.2848 \times 10^{-6} & -1.1680 \times 10^{-6} \\ -7.5920 \times 10^{-7} & 7.5920 \times 10^{-7} & -7.5920 \times 10^{-7} & 7.5920 \times 10^{-7} \\ -3.5040 \times 10^{-7} & -3.5040 \times 10^{-7} & 3.5040 \times 10^{-7} & 3.5040 \times 10^{-7} \end{bmatrix}$
wrench mapper		$\begin{bmatrix} -2.0385 \times 10^5 & -3.2929 \times 10^5 & -6.7950 \times 10^5 & 4.2808 \times 10^4 \\ 2.0385 \times 10^5 & 3.2929 \times 10^5 & -7.4745 \times 10^5 & 4.2808 \times 10^4 \\ 2.0385 \times 10^5 & -3.2929 \times 10^5 & 6.7950 \times 10^5 & 4.2808 \times 10^4 \\ -2.0385 \times 10^5 & 3.2929 \times 10^5 & 7.4745 \times 10^5 & 4.2808 \times 10^4 \end{bmatrix}$

Table 2.2: *Quadrotor's parameters.*

As recalled in section 2.1.5, quadrotors have an underactuated structure. This means that, in order to reach the goal, 4 of its 6 DoFs have to be established.

The chosen four DoFs are the x, y and z positions and the yaw angle (ψ) and the same trajectory of the UGV previously shown in the (x, y)-plane is adopted as the desired one, with a constant altitude ($z = 2\ m$). The desired trajectory that the quadrotor has to track is shown in Fig.2.11.

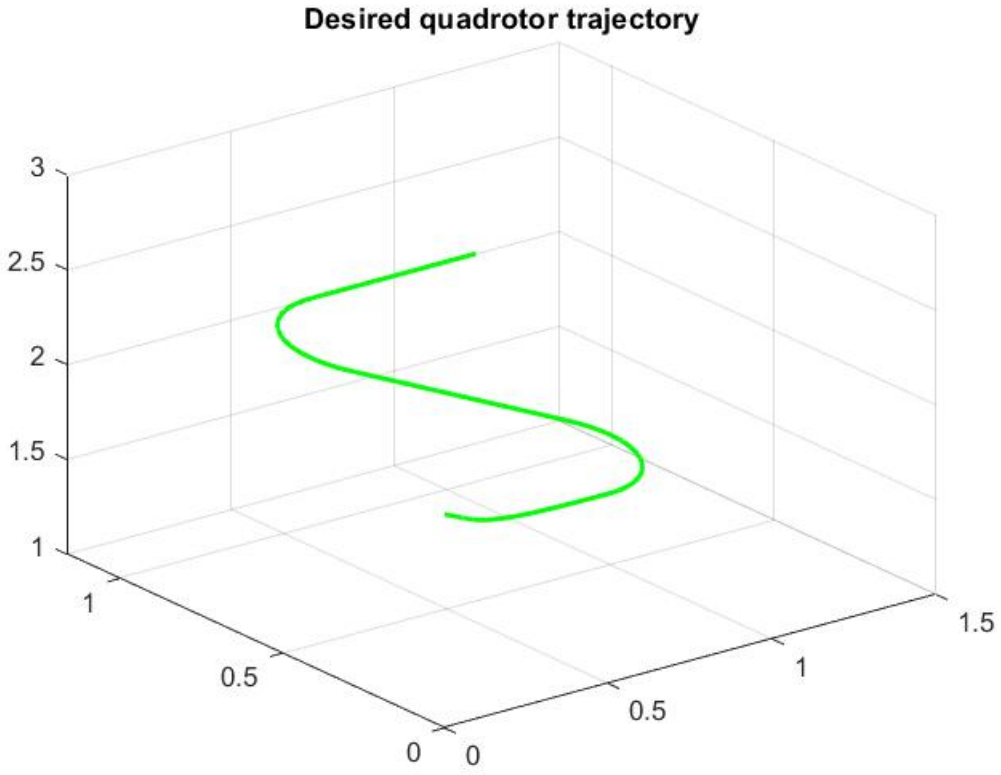


Figure 2.11: *Desired trajectory of the quadrotor.*

Hence, knowing these four values, also the other two DoFs, that are the roll (ϕ) and pitch (θ) angles, have to be computed. In order to do this, the trajectory errors on the x, y and z coordinates have to be known. Then, a PID controller is applied to such tracking errors (one PID controller for each tracking error) to obtain the desired values of the linear acceleration along the x, y and z axis. Note that, as the case of the UGV, the tracking errors have to converge to zero and this goal is reached as presented in Fig.2.12.

So the desired roll and pitch angles are given by the formulas in Eq.2.28.

$$\phi^{des} = \frac{1}{g}(\sin(\psi^{des})\ddot{x}^{des} - \cos(\psi^{des})\ddot{y}^{des}) \tag{2.27}$$

$$\theta^{des} = \frac{1}{g}(\cos(\psi^{des})\ddot{x}^{des} + \sin(\psi^{des})\ddot{y}^{des}) \tag{2.28}$$

Doing so, the desired behaviours of the roll, pitch and yaw angles are shown in Fig. 2.13.

Later also the errors on the angles, defined as the difference between the desired and real values, are computed and their behaviour is shown in Fig.2.14. Looking at these plots, the error relative to ϕ and θ converge to zero, while the

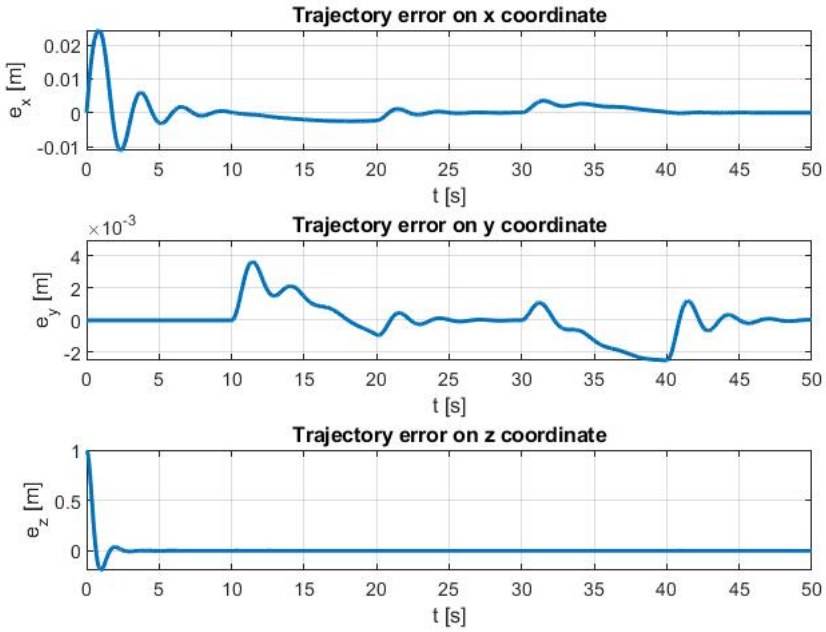


Figure 2.12: *Tracking errors of the quadrotor.*

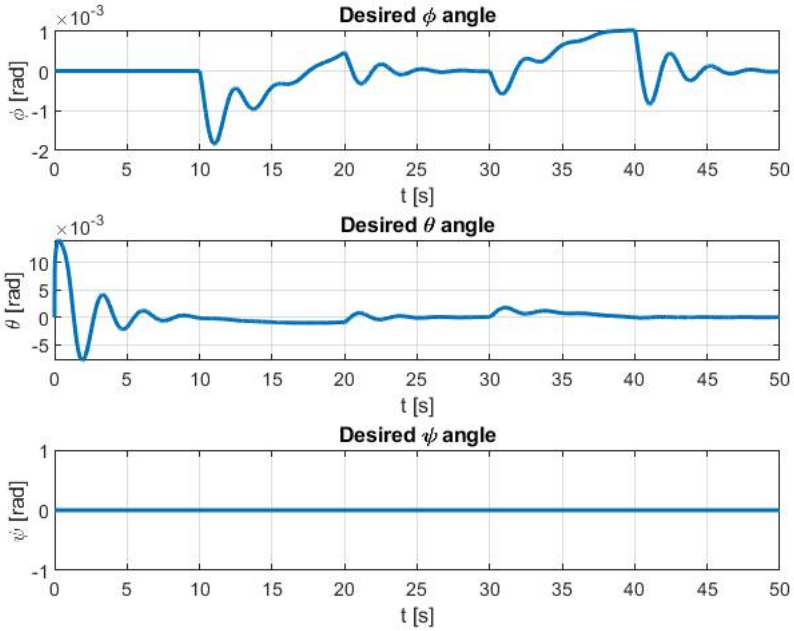


Figure 2.13: *Desired angles of the quadrotor.*

one relative to ψ oscillates but its of order -18 so that it is possible to consider this as a right behaviour.

At this point, the elevation and attitude controllers can be applied in order to have the values of the desired common thrust (T) and the desired body torques

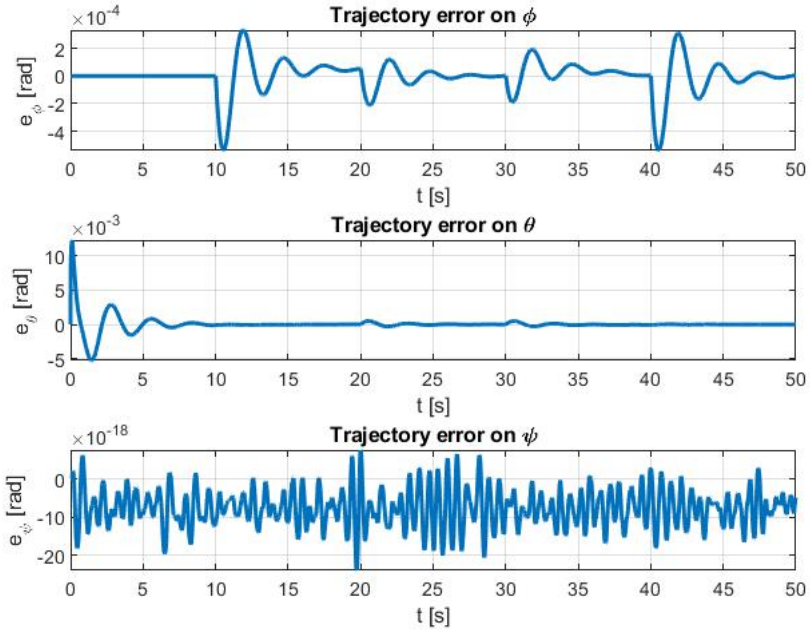


Figure 2.14: *Errors on the ϕ, θ, ψ angles of the UAV.*

(τ_ϕ, τ_θ, τ_ψ), respectively.

In details, the attitude controllers consist of three independent PID controllers, while the elevation controller is a PID with the addition of a nonlinear compensation term for gravity and ϕ and θ angles, and a feedforward term. In both these two cases, the integral term can be neglected.

Mathematically, the controllers are implemented accordingly with formulas Eq.2.29.

$$\begin{aligned} T &= mg + K_P e_z - K_D \dot{z} \\ \tau_\phi &= K_P e_\phi - K_D \dot{\phi} \\ \tau_\theta &= K_P e_\theta - K_D \dot{\theta} \\ \tau_\psi &= K_P e_\psi - K_D \dot{\psi} \end{aligned} \tag{2.29}$$

where z, ϕ, θ, ψ indicate the measured value of the altitude, roll, pitch and yaw angle, respectively.

Hence, from these quantities, the control input vector, which corresponds to the rotor spinning rates squared, can be found. A wrench mapper, described in Tab.2.2 is designed to convert the desired common torque and the desired body torques into the required input vector.

Then, the thrust and torques are computed in the Body Frame and, from

them, the linear and angular acceleration are obtained. Knowing the linear and angular acceleration and integrating them (double integration to have the position and the angles, while a single integration to obtain the linear and angular velocities), the values of the position (x, y, z coordinates), linear velocities (v_x, v_y, v_z), angular velocities (ω_x, ω_y, ω_z) and the angles (ϕ, θ, ψ) are found. The measured trajectory results the one shown in Fig.2.15.

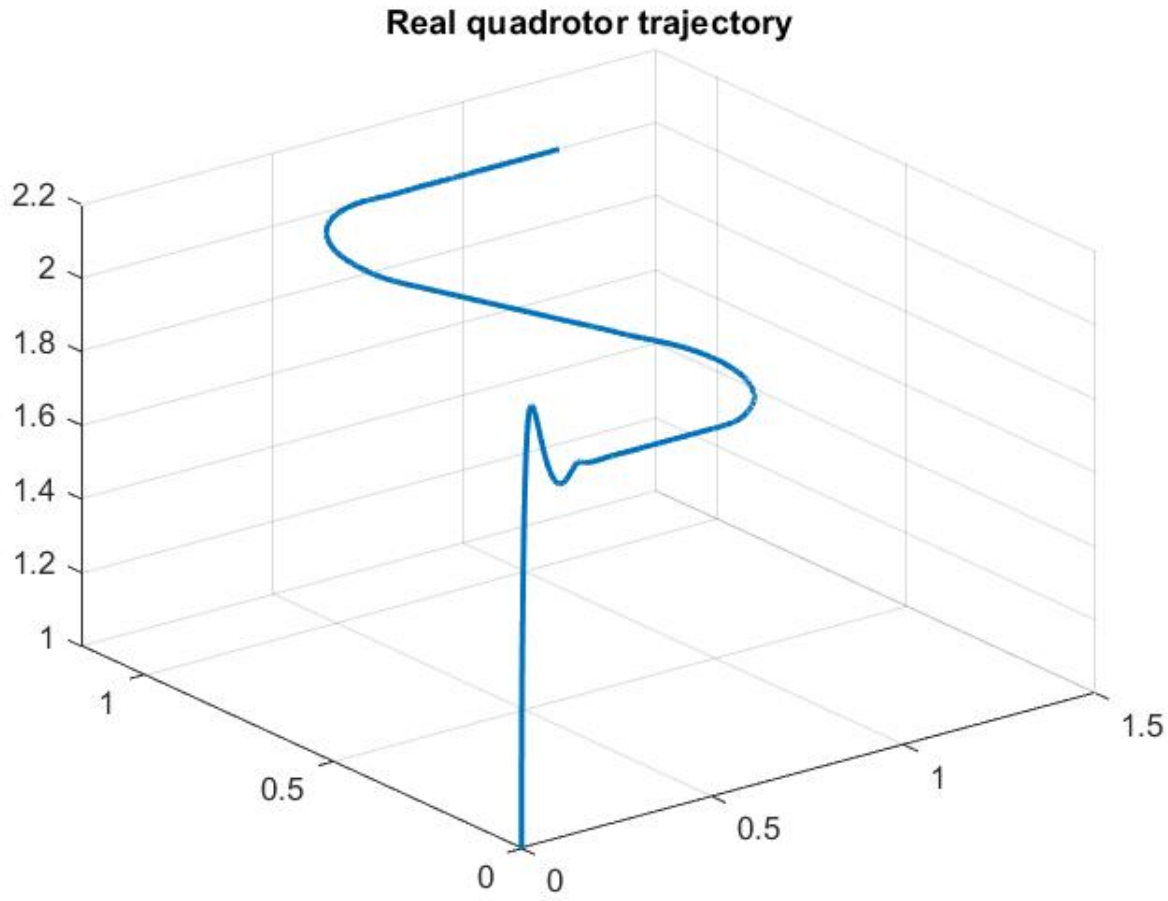


Figure 2.15: *Measured trajectory of the UAV.*

Instead, in order to compare the desired and the real trajectory of the UAV, a plot of both of them together is shown in Fig.2.16.

From the comparison between the desired and measured trajectory, it is possible to observe that the measured trajectory reaches the desired one except at the beginning due to the fact that the real initial position is imposed at $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. The goal is achieved as expected since the errors previously shown, converge to zero.

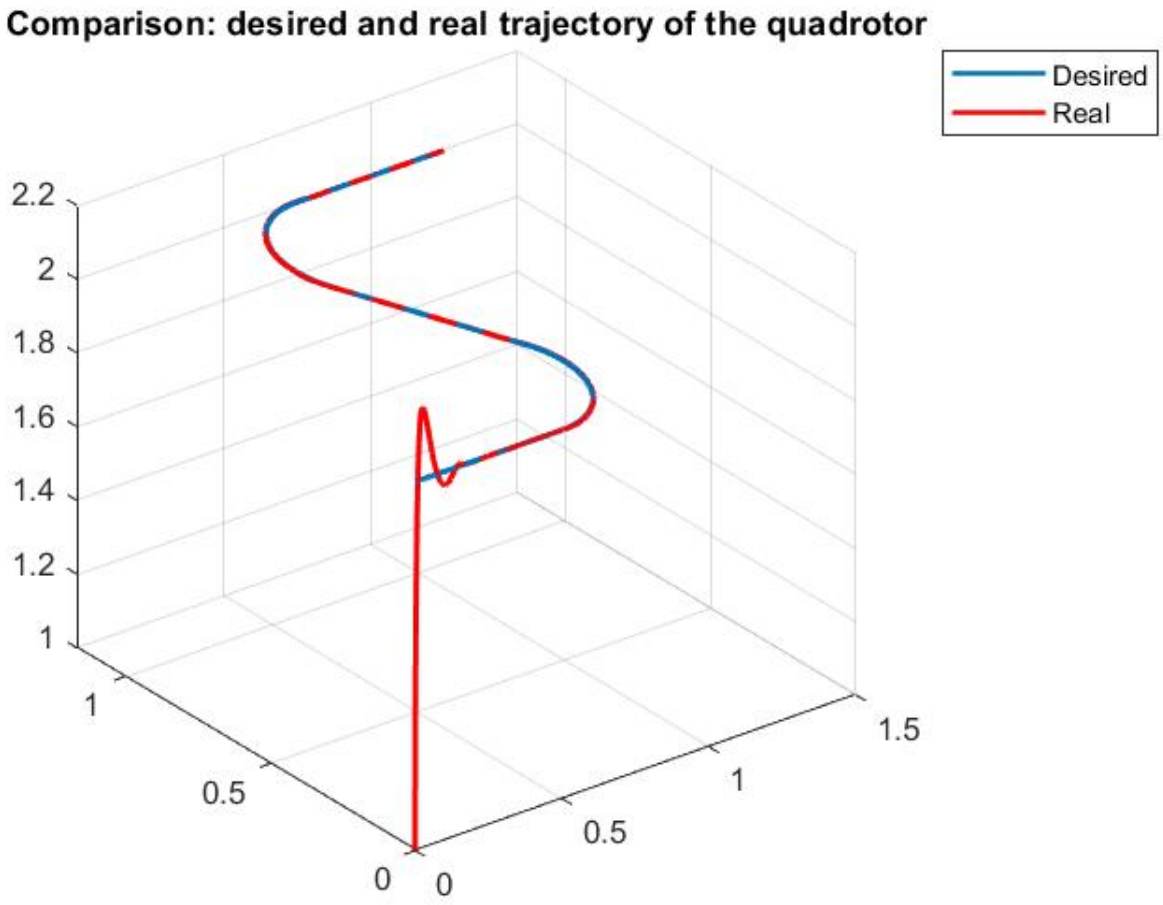


Figure 2.16: *Measured and desired trajectory of the UAV.*

2.2 Heterogeneous Multi-Agent System

An heterogeneous multi-robot system is composed by N autonomous robots, indexed as $R_1, ..., R_N$. Usually these robots work under the assumption that they can communicate sending own position and receiving positions of other robots among the team through wireless network.

The main features of MAS are eight[6]:

1. Leadership: there could be the existence of a leader, i.e. an agent that defines goals and tasks for the other agents. The presence or absence of such a leader can be used to categorize MAS as *leaderless* or *Leader-Follower*.

   In a leaderless MAS, each agent autonomously decides on its actions based

[6]Dorri A., SALIL S. KANHERE1, JURDAK R., Multi-Agent Systems: A Survey, in School of Computer Science and Engineering, University of New South Wales, June 19, 2018, Sydney, Australia.

on its own goals. The decision of each agent is affected by the decision of other agents if agents collaborate to reach consensus[7] on a particular feature. Instead, in Leader-Follower, the leader agent establishes actions for the other agents, called followers. Followers communicate and share knowledge to find the position of the leader. The leader is either predefined or is collaboratively chosen by agents. In a MAS, it is possible to have just a mobile leader, that can move from one position to another, or a group of agents acting as leaders.

2. Decision function: MASs are categorized based on the proportionality of the output changes with respect to its input changes. According to this, MASs are categorized as *linear* and *non-linear*.

   In a linear MAS, there is a proportional relation between the decision of an agent and the sensed parameters from the environment. This feature makes linear agents easier to be analyzed mathematically.

   In a non-linear MAS, there is no proportion between the decision of the agent and the sensed metrics due to the non-linearity of the input to the decision making process.

3. Heterogeneity: MASs can be divided into two categories: *homogeneous* and *heterogeneous*.

   A homogeneous MAS includes agents with the same characteristics and functionalities, while a heterogeneous MAS includes agents with different actuation capabilities and whose components have different DoFs as controllable variables.

4. Agreement parameters: in some applications of MAS, agents have to be adjusted depending on particular parameters known as metrics. Based on the number of metrics, MASs are classified as *first*, *second* and *higher order*.

   In first order, agreement on one metric has to be achieved by the agents. In second-order, agents must agree on two metrics while, in higher-order MAS, the agents are in agreement if the metrics (either one or two) and their high-order derivatives converge to a common value.

5. Delay consideration: in performing a task, agents can be affected by multiple sources of delay. In this case, MASs can be classified into two groups

[7] In MAS consensus refers to achieving a global agreement over a particular feature of interest. Multiple MAS features impact the consensus problem as they affect communication and collaboration between agents.

namely *with delay* or *without delay*, depending if the delays are relevant or negligible, respectively. In the first case, the delay is taken into account while, in the second case, no delay is considered.

6. Topology: it refers to the location and relations of agents. MAS topology can be either *static* or *dynamic*.

   In a static topology, the position and relations of an agent remains unchanged over the lifetime of the agent. In dynamic topology, the position and relations of an agent change as the agent moves, leaves or joints the MAS, or establishes new communications/ relations with other agents.

7. Data Transmission frequency: the data sensed from the environment are sensed and shared in a *time-triggered* or an *event-triggered* manner.

   In time-triggered MAS, the agent always senses the environment, collects data and sends all newly sensed data to other agents in pre-defined time intervals. In event-triggered MAS, the agent senses the environment and shares the data only when particular event occurs.

8. Mobility: an agent can be classified as *static* or *mobile* agents, depending on its dynamism.

   An agent is static if it is always in the same position in the environment, while it is mobile if it moves.

The two main tasks, that enable multi-agent systems to reach complex goals, are: the network localization and the formation control law. The goal of network localization is to estimate the location of each agent in a network through information perceived or shared by neighbouring agents. Instead, the goal of formation control is to reach a desired geometric pattern controlling each agent through local information from neighbors.

The first step that has to be implemented is the network localization. This task must be completed before a sensor network provides other services like positioning mobile robots or monitoring areas of interest.

2.2.1 Graph theory

Any formation can be modeled according to the graph based multi-agent system representation since it is considered as a networked architecture. Doing so, each vertex of the graph represents an agent, while the edge connecting two vertices indicates the communication between them.

So, an heterogeneous formation made up of N-agents can be associated to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. A graph can be distinguished into a direct graph and an undirect one.

The node $v_i \in \mathcal{V}$ corresponds to the $i-th$ agent with $i \in \{1...n\}$ and the directed edge $e_k = e_{ij} = (v_i, v_j) \in \mathcal{E}$ indicates that the $i-th$ agent can communicate with $j-th$ one while, if the edge is undirected, $e_k = e_{ij} = (v_i, v_j) \in \mathcal{E}$ means that the $i-th$ agent can communicate with the $j-th$ one and also the opposite, i.e. the $j-th$ agent can exchange information with the $i-th$ one. In addition, if $e_{ij} \in \mathcal{E}$, then v_i and v_j are neighbours. So, the set of neighbours of a given node v_i is defined as

$$\mathcal{N}(v_i) = \mathcal{N}_i = \{v_j : e_{ij} \in \mathcal{E}\}$$

From the number of neighbours, it is possible to define the *degree* of a given node v_i since it is equal to the number of its neighbours. In particular, a graph is said to be *regular* if all the nodes have the same degree.

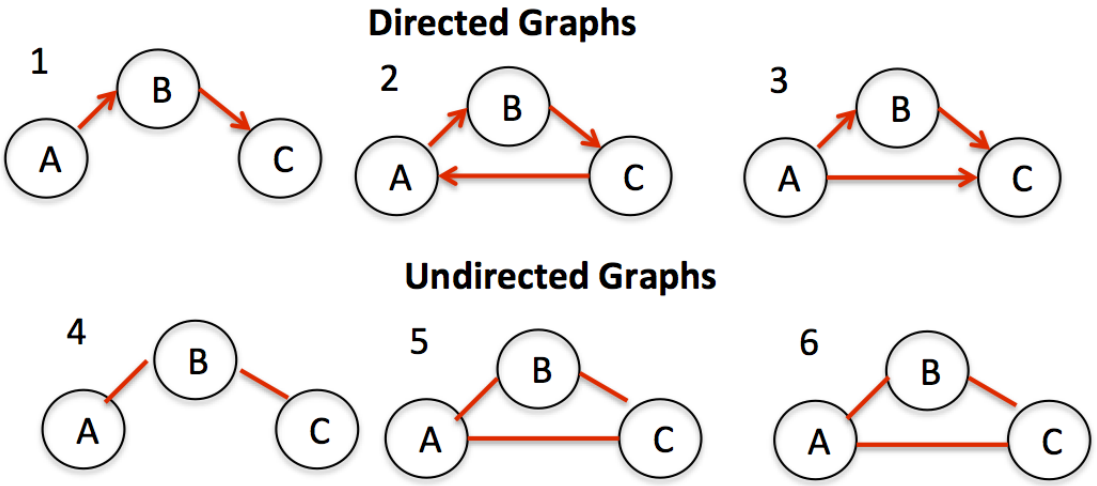


Figure 2.17: *Examples of directed and undirected graphs.*

Hence, considering the examples in Fig.2.17, the graphs number 1, 2 and 3 are directed ones, while the numbers 4, 5, 6 are undirected. Then, in details, looking at graph 1, A and B are neighbours and also B and C are neighbours since they are connected through an edge. Instead, note that there is no connection between A and C. So node A and B have degree equal to 1, while C has degree equal to 0. Looking at graph 2, A and B are neighbours, B and C are neighbours and also C and A are neighbours. Hence, all the nodes have degree 1 so that the graph is regular.

Instead, the graphs number 4, 5 and 6 are undirected ones and, considering

the graph 5 as an example, A and B are neighbours, B and C are neighbours and also A and C are neighbours. However, being the graph an undirected one, A has degree 2, B has degree 2 and C has degree 2, so again the graph is regular.

The formal definition of graph is the one reported in Def.2.2.1

Definition 2.2.1 (Simple graph) [8] *A simple graph $\mathcal{G}$ consists of a non-empty finite set $\mathcal{V}(\mathcal{G})$ of elements called vertices (or nodes), and a finite set $\mathcal{E}(G)$ of distinct unordered pairs of distinct elements of $\mathcal{V}(\mathcal{G})$ called edges. We call $\mathcal{V}(\mathcal{G})$ the vertex set and $\mathcal{E}(G)$ the edge set of $\mathcal{G}$. An edge v, w is said to join the vertices v and w, and is usually abbreviated to vw.*

Then, it is also possible to identify some properties of the graphs:

1. Isomorphism

   Definition 2.2.2 (Isomorphic graphs) [9] *Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if there is a one-one correspondence between the vertices of $\mathcal{G}_1$ and those of $\mathcal{G}_2$ such that the number of edges joining any two vertices of $\mathcal{G}_1$ is equal to the number of edges joining the corresponding vertices of $\mathcal{G}_2$.*

2. Connectedness

   Definition 2.2.3 (Connection between graphs) [10] *If two graphs $\mathcal{G}_1 = (\mathcal{V}(\mathcal{G}_1), \mathcal{E}(G_1))$ and $\mathcal{G}_2 = (\mathcal{V}(\mathcal{G}_2), \mathcal{E}(G_2))$, where $\mathcal{V}(\mathcal{G}_1)$ and $\mathcal{V}(\mathcal{G}_2)$ are disjoint, then their union $\mathcal{G}_1 \cup \mathcal{G}_2$ is the graph with vertex set $\mathcal{V}_1 \cup \mathcal{V}_2$ and edge family $\mathcal{E}_1 \cup \mathcal{E}_2$. A graph is connected if it cannot be expressed as the union of two graphs, and disconnected otherwise. Clearly any disconnected graph $\mathcal{G}$ can be expressed as the union of connected graphs, each of which is a component of $\mathcal{G}$.*

3. Adjacency

   Definition 2.2.4 (Adjacency) [11] *Two vertices v and w of a graph $\mathcal{G}$ are adjacent if there is an edge vw joining them, and the vertices v and w are then incident with such an edge. Similarly, two distinct edges e and f are adjacent if they have a vertex in common.*

4. Subgraphs

[8] Robin J. Wilson, Introduction to graph theory, Fourth Edition, Prentice Hall, 1998
[9] ibidem
[10] ibidem
[11] ibidem

Definition 2.2.5 (Subgraphs) [12] *A subgraph of a graph $\mathcal{G}$ is a graph: each of whose vertices belongs to $\mathcal{V}(\mathcal{G})$ and each of whose edges belongs to $\mathcal{E}(\mathcal{G})$.*

5. Matrix representation

   A graph can be described through matrices, in particular through the *Node degree matrix* ($\Delta_{\mathcal{G}}$), *Adjacency matrix* ($A_{\mathcal{G}}$), *Incidence matrix* ($D_{\mathcal{G}}$) and *Laplacian matrix* ($\mathcal{L}_{\mathcal{G}}$).

   The Node degree matrix of a (non-oriented) graph $\mathcal{G}$ is defined as the diagonal matrix $\Delta\mathcal{G} \in \mathbb{R}^{n\times n}$ with entries the degrees of the nodes. For oriented graphs, the diagonal matrices $\Delta\mathcal{G}_{IN}$ and $\Delta\mathcal{G}_{OUT}$ are defined, whose elements are, respectively, the in-degrees (in-coming edges) and out-degrees (out-going edges) of the nodes.

   Hence, if $\mathcal{G}$ is a graph with vertices labelled (i.e. the edges are named) $\{1, 2, ..., n\}$, its adjacency matrix A is the $n \times n$ matrix whose $i-th$ entry is the number of edges joining vertex i and vertex j. Hence, for an undirected graph, it holds that:

$$A_{\mathcal{G}}(i,j) = \begin{cases} 1 & if \quad (i,j) \in \mathcal{E} \\ 0 & otherwise \end{cases} \tag{2.30}$$

   while for a directed graph, it is:

$$A_{\mathcal{G}}(i,j) = \begin{cases} 1 & if \quad (i \rightarrow j) \in \mathcal{E} \\ 0 & otherwise \end{cases} \tag{2.31}$$

   In addition, if the edges are labelled $\{1, 2, ..., m\}$, its incidence matrix is the $n \times m$ matrix whose $ij-th$ entry is 1 if vertex i is incident to edge j, and 0 otherwise. Hence, for an undirected graph it holds that:

$$D_{\mathcal{G}}(i,k) = \begin{cases} 1 & if \quad v_i \in e_k \\ 0 & otherwise \end{cases} \tag{2.32}$$

[12] Robin J. Wilson, Introduction to graph theory, Fourth Edition, Prentice Hall, 1998

while for a directed one, it is.

$$D_{\mathcal{G}}(i,k) = \begin{cases} +1 & if \quad v_i \quad tail \quad of \quad e_k \\ -1 & if \quad v_i \quad head \quad of \quad e_k \\ 0 & otherwise \end{cases} \tag{2.33}$$

Knowing the Node degree matrix and the Adjacency matrix, the definition of Laplacian matrix follows as the difference between the Node degree matrix and the Adjacency matrix:

$$\mathcal{L}_{\mathcal{G}} = \Delta_{\mathcal{G}} - A_{\mathcal{G}} \tag{2.34}$$

2.2.2 Task taxonomy and Multi-robot system workflow

Task complexity represents the difficulty of a task and, depending on it, the number and type of robots needed to complete the task can be determined. Hence, in order to solve a task, it can be decomposed into multiple simpler sub-tasks.

Single-robot tasks, such as small-scale mapping, pick and place, and navigation problems, can be accomplished by one robot.

Instead, multi-robot tasks require multiple cooperating robots. In addition, multi-robot tasks can also be classified according to the level of cooperation required to achieve the objective, ranging from loosely to tightly coordinated. Loosely coordinated tasks can be decomposed to be sub-tasks that can be independently executed with minimum interaction among robots. Examples include large-scale exploration and mapping, hazardous material clean-up, tracking and surveillance. In such scenarios, the environment can be divided into disjoint areas and the robots operate within their specified areas. Tightly coupled tasks are not decomposable and require coordinated execution with significant interaction among robots. Examples include robot soccer, object transport and large-scale construction.

To systematically design MAS capable of accomplishing complex tasks, four main design blocks are identified:

1. task decomposition: sub-division of complex tasks into simpler and smaller tasks.

2. coalition formation: formation of agent teams.

3. task allocation: the sub-tasks are assigned to agent teams.

4. task execution/planning and control: achieving an objective through a sequence of actions on the environment.

This subdivision is summarized in Fig.2.18.

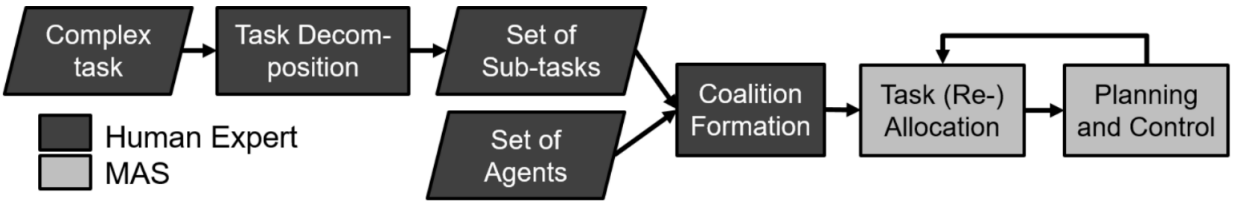


Figure 2.18: *Workflow for the multi-robot system [**5**].*

Looking at the scheme in Fig.2.18, it is possible to see that, in order to subdivide complex tasks into simpler sub-tasks and to form coalitions of agents given a set of them, a human designer is required.

Task decomposition consists in dividing a complex task into a set of simpler or more primitive sub-tasks that are either independent or sequentially dependent on each other. Then, the simpler tasks have to be executed. Hence, these sub-tasks should be allocated to a group of robots or a single robot, depending if the sub-tasks are multi-robot tasks or not, respectively. If the sub-task is a multi-robot task, first a coalition formation of robots has to be created. Then, the sub-tasks have to be assigned to them before task execution can be performed.

Note that task allocation and robot planning and control are autonomously performed by the robot teams. In addition, performing simultaneous coalition formation and task allocation has also been tried to improve the performance.

3

Types of controllers

A controller is a dynamical system whose purpose is to minimize an error defined as the difference between the actual value of the system (called *process variable*) and the desired one (called *setpoint*). Hence, the input of the controller is the error signal ($e(t) = r(t) - y(t)$), while the output results in a control signal ($u(t)$). The scheme of a feedback control is shown in Fig.3.1.

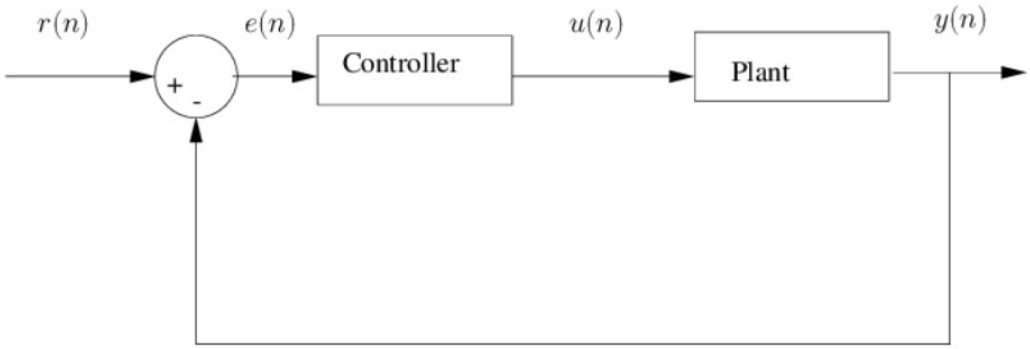


Figure 3.1: *Generalized scheme of a feedback control, considering a PID controller [**2**].*

In particular, a controller is used for:

- steady-state accuracy improvement by reducing steady-state error so that stability is also improved;
- reduce unnecessary offsets generated by the system;
- maximum system overshoot control;
- reducing noise generated by the system;

- speeding up the slow response of an overdamped system.

In details, in the next sections, the PID and MPC controller are explained, since they are used in the project. Then, it is also presented the Leader-Follower approach implemented in order to define the cooperation between the agents.

3.1 PID CONTROLLER

The PID controller is a dynamical mechanism with a predefined structure, whose efficiency is adjusted through the choice of at most 3 parameters (K_P, K_I, K_D), called gains or DoFs of the controller and found through a trial and error approach. Each parameter represents a different action: proportional, integral and derivative, respectively.

The general structure of a PID controller is reported mathematically in Eq.3.1 while, graphically, in Fig.3.2.

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{3.1}$$

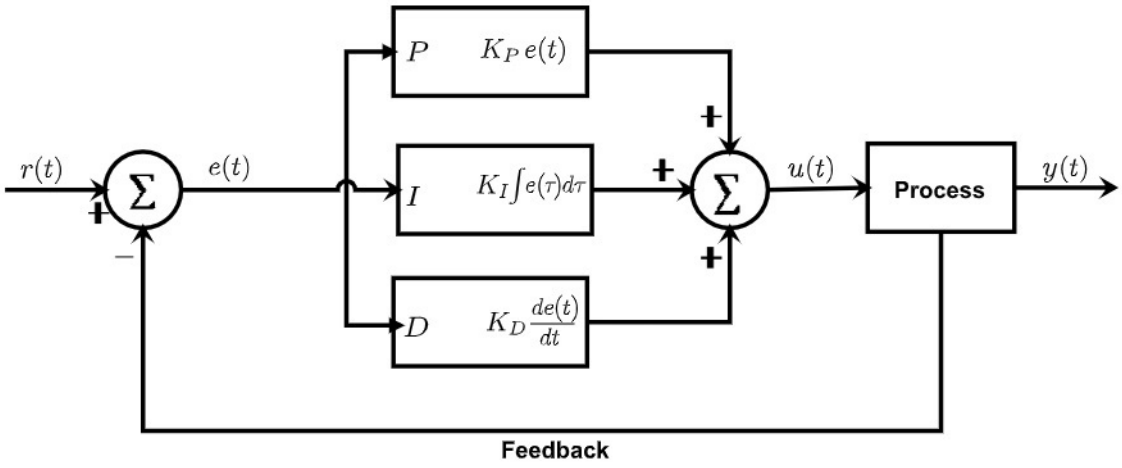


Figure 3.2: *Generalized scheme of a PID controller [**39**].*

Instead, the details of each action are presented in the following.

3.1.1 PROPORTIONAL ACTION

The output of this action is proportional to the input signal (the error), accordingly with the value of the proportional gain K_P. First of all, consider the case where only this action influences the system, i.e. $K_I = K_D = 0$. If the

proportional gain increases, the response speed of the system improves, but the stability decreases and more oscillations occur. Because of this, it is not possible to choose any value for K_P: there is a threshold value beyond which the system is unstable.

Another drawback of the P controller is due to the fact that it increases the maximum overshoot of the system.

3.1.2 Integral action

The output of this action is proportional to the integral of the input signal (the error). The integral gain K_I is given by the inverse of the integral time constant T_I (called *reset time*) multiplied by K_P as shown in Eq.3.2.

$$K_I = \frac{K_P}{T_I} \tag{3.2}$$

The variable parameter that has to be established is the constant of integration time T_I observing that the integral effect increases as T_I decreases. In particular:

- if $T_I \to \infty$, the integral effect is negligible;
- if $T_I \to 0$, the integral reaches quickly the value of the input signal (the error), with the drawback of the generation of big oscillations which require some time to stabilize.

The main goal of the integral action is on eliminating the steady-state error caused by the proportional action.

3.1.3 Derivative action

The output of this action is the first derivative with respect to time of the error signal, i.e. $\dot{e}(t)$. The derivative gain K_D is given by the product between the proportional gain (K_P) and the derivative time constant (T_D) as in Eq.3.3.

$$K_D = K_P T_D \tag{3.3}$$

The main objective of this action is to anticipate corrective action since, by deriving the input signal (the error), it is possible to account for rapid signal changes.

The parameter that determines the efficiency of this action is T_D, the value of which determines the rate at which the signal rises. Its main drawback is

verified in the case of signals with harmonic content at high frequencies since, in this case, it implies such kind of signals.

3.1.4 PID configurations

In order to control the system and, hence, to have an error converging to zero, the three actions introduced above can be combined. In details, these can be combined to generate the following controllers: PI controller, PD controller and PID controller, while the ID and D controllers are inefficient.

- PI controller: it consists on the combination of a P and I controller. It allows to obtain a better accuracy and speed of response, without worsening stability.
- PD controller: it consists on the combination of a P and D controller. It allows to anticipate the steady-state error.
- PID controller: it consists on the combination of a P, I and D controller. It is used to avoid/reduce the presence of oscillations but without providing an immediate stress response.

Note that the presented controllers are the most common kinds of PID controller.

3.2 MPC controller

Model Predictive Control is a feedback control algorithm implemented to solve an optimization problem. It uses a model to make predictions about future outputs of a process solving an online optimization problem at each time instant to select the best control action that drives the predicted output to the reference.

MPCs have been used in the process industry since the 1980s. With the increasing computing power of microprocessors, their use has spread to other fields as: automotive, aerospace, energy, food processing, industrial manufacturing (to design technical systems like bridges, cars, aircraft and digital devices), robotics and metallurgy, business (to allocate the resources in logistic, investment), science (to estimate and fit of models to measurement data and design experiments)

The advantage of MPC is that it is a multivariable controller that controls the outputs simultaneously by taking into account all the interactions between system variables, even if this implies the requirement of tuning too many controller

gains. Another advantage of the MPC is that it can handle constraints that play an important role in order to avoid undesired consequences. In addition, MPC can easily incorporate future reference information into the control problem to improve controller performance.

Even if MPC has all these benefits, it requires a powerful fast processor with a large memory due to the fact that it solves an online optimization problem at each time step.

In a control problem, the goal of the controller is to compute the input to the plant such that the plant output follows a desired reference. The control scheme of a system with an MPC controller is shown in Fig.3.3.

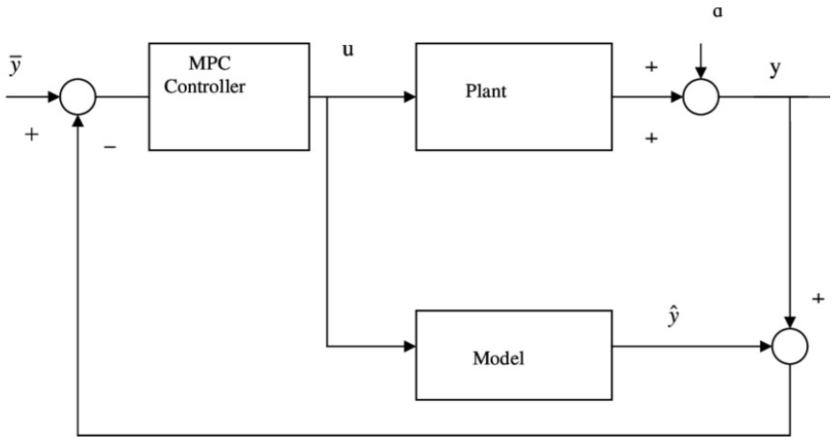


Figure 3.3: *Basic generalized scheme of a MPC controller [**34**].*

Another fundamental aspect to realize the MPC controller is about the choice of the value of the parameters:

- sample time T_s, that is the sample time at which the control algorithm is executed by the controller. If it is too big, when a disturbance comes in, the controller will not be able to react to the disturbance fast enough. On the contrary, if the sample time is too small, the controller can react much faster to disturbances and setpoint changes, but this causes an excessive computational load. To find the right balance between performance and computational effort, it is fine to fit 10 to 20 samples within the rise time of the open loop system response.

- weights. The outputs have to track as close as possible to their setpoints, but at the same time it is desired to have smooth control moves to avoid aggressive control maneuvers. The way to achieve a balanced performance between these competing goals is to weight the input rates and outputs relative to each other and the relative weights within the groups, too.

- constraints. These can be either soft or hard constraints. Hard constraints cannot be violated, while the soft constraints can be violated. Despite this, having hard constraints on both inputs and outputs is not a good idea because these constraints may conflict with each other leading to an unfeasible solution for the optimization problem. The recommendation is to set output constraints as soft and avoid having hard constraints both on the inputs and the rate of change of the inputs.

- control horizon. Each control horizon can be thought of as a free variable that needs to be computed by the optimizer. So, the smaller the control horizon, the fewer the computations. Increasing the control horizon, it is possible to have better predictions but at the cost of increasing the complexity. A good rule of thumb for choosing the control horizon is setting it to 10 to 20% of the prediction horizon, and having minimum 2-3 steps.

- prediction horizon. The number of predicted future time step is called the prediction horizon and shows how far the controller predicts into the future. The recommendation for choosing the prediction horizon is to have 20 to 30 samples covering the open loop transient system response.

The choice of these parameters is important since they affect the controller performance and the computational complexity of the MPC algorithm.

An MPC problem that has a linear system, linear constraints and a quadratic cost function gives rise to a convex optimization problem where the cost function has a single global optimum and the goal of the optimization is to find this optimum.

Instead, if the system is not linear (nonlinear), it is possible to use linear MPC and the benefits from the properties of the convex optimization problem. The available methods to use in this case are the Adaptive and Gain-scheduled MPC controllers. The way these controllers deal with a nonlinear system is based on linearization.

In Adaptive MPC, a linear model is computed on the fly as the operating conditions change and, at each time step, you update the internal plant model used by the MPC controller with this linear model. In Adaptive MPC, the structure of the optimization problem remains the same across different operating points. This means that the number of states and the number of constraints do not change for different operating conditions over the prediction horizon. However, if they do change, then Gain-Scheduled MPC has to be used.

In Gain-Scheduled MPC, the operating points of interest are linearized online and, for each operating point, a linear MPC controller is designed. Each controller is independent from each other and therefore may have a different number of states and of constraints. Note that in this approach, it is also required to design an algorithm that will switch between the predefined MPC controllers for different operating conditions. Although, having independent controllers is an advantage of Gain-Scheduled MPC, it uses more memory than Adaptive MPC. Hence, the recommendation is to use Adaptive MPC if a linear plant model can be found at run-time and the structure of the optimization problem remains fixed across different operating conditions. If they change, then it is possible to use Gain-Scheduled MPC, where independent MPC controllers are designed for the changing operating conditions.

Despite this, nonlinear MPC (approximated well by linear model), has to be used. This method is the most powerful one as it uses the most accurate representation of the plant, namely a nonlinear plant model. Therefore, the predictions made by the controller are more accurate, which also lead to better control actions. In addition, it is also the most challenging one to solve in real time because when there are nonlinear constraints and a nonlinear cost function, the optimization problem becomes nonconvex. The cost function may have multiple local optima and finding the global optimum may be hard. The efficiency of solving the non convex optimization problem, that requires a large number of computations, depends on the available nonlinear solver. In fact, MPC is computationally complex: it is formulated as a QP (Quadratic Programming) problem that tries to minimize a quadratic cost function. MPC computations become more complex with the increasing number of states, constraints, the length of the control and prediction horizons.

In order to solve an optimization problem, the following ingredients have to be defined:

1. an objective function $\Phi(\omega)$, that shall be minimized or maximized;

2. decision variables (ω);

3. constraints that shall be respected. These can be of the form $g_1(\omega) = 0$ (equality constraints) or $g_2(\omega) \geq 0$ (inequality constraints).

The mathematical formulation in standard form of the nonlinear program-

ming problem is shown in Eq.3.4.

$$\begin{aligned} &\min_{\omega} \Phi(\omega) \\ such \quad that \quad & g_1(\omega) \leq 0 \\ & g_2(\omega) = 0 \end{aligned} \tag{3.4}$$

where $\Phi(\cdot)$, $g_1(\cdot)$, $g_2(\cdot)$ are usually assumed to be differentiable.

In the following it is considered the case of nonlinear dynamic programming since the linear and quadratic cases can be assumed to be special cases (subproblems) of the nonlinear one. In particular, there is:

- linear programming (LP) when $\Phi(\cdot)$, $g_1(\cdot)$, $g_2(\cdot)$ are affine and, hence, these functions can be expressed as linear combinations of the elements of ω;
- quadratic programming (QP) when $g_1(\cdot)$, $g_2(\cdot)$ are affine but the objective function $\Phi(\cdot)$ is a linear quadratic function

In order to solve the optimization problem, it is necessary to compute the minimization or maximization of a given objective function. Usually, it is considered a minimization problem, but it can also be expressed as a maximization one since maximization can be treated as a minimization of the negative objective. Hence, the problem to solve is expressed as in Eq.3.5 or, equivalently, Eq.3.6.

$$\begin{aligned} &\min_{\omega} \Phi(\omega) \\ such \quad that \quad & g_1(\omega) \leq 0 \\ & g_2(\omega) = 0 \end{aligned} \tag{3.5}$$

$$\begin{aligned} &\max_{\omega} -\Phi(\omega) \\ such \quad that \quad & g_1(\omega) \leq 0 \\ & g_2(\omega) = 0 \end{aligned} \tag{3.6}$$

Despite this, normally, the value of ω that minimizes the objective expressed as in Eq.3.5 has to be found. So, the solution has the expression shown in Eq.3.7.

$$\omega^* = \min_{\omega} \Phi(\omega) \tag{3.7}$$

Directly substituting Eq.3.7 in Eq.3.5, the found value of the objective function is the one reported in Eq.3.8.

$$\Phi(\omega^*) = \Phi(\omega)|_{\omega} = \min_{\omega} \Phi(\omega) \tag{3.8}$$

More in details, the problem is analyzed starting to consider a SISO system described by the expression in Eq.3.9.

$$x(k+1) = f(x(k), u(k)) \tag{3.9}$$

At time k, a new decision or a new action would like to be taken so that the state x eventually reaches some reference. In MPC, it is looked in the future for n time steps and it is tried to find the best or optimal sequence of control actions that makes the prediction for the state approaches or actually goes to the state reference. Then, only the first portion of that sequence of control actions is applied to the system. So, when this first control action is applied and the current state is in point 1 in Fig.3.4, it is predicted that the state goes to point 2. However, what happens in reality does not exactly coincide with the prediction. This is why, after applying the first portion of the control action, another step of MPC is solved and a new control sequence is got by applying the first control action in that sequence.

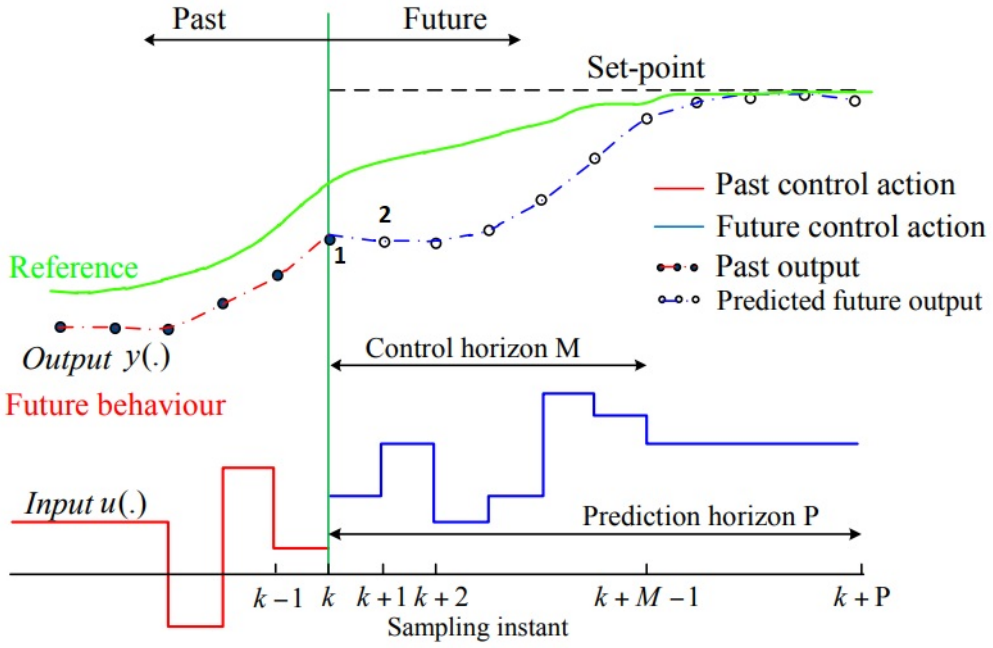


Figure 3.4: *General concept of a discrete MPC [**41**].*

Hence, in MPC, three steps have to be performed:

1. Prediction;
2. Online Optimization (minimizing the difference);
3. Receding Horizon.

The mathematical formulation of MPC is very similar to LQR. A running cost (also called *stage cost*) is defined where the difference between the predicted state and the reference and the difference between the control action and its reference value are penalized, as shown in Eq.3.10.

$$l(x,u) = ||x_u - x^r||_Q^2 + ||u - u_r||_R^2 = (x_u - x^r)^T Q(x_u - x^r) + (u - u_r)^T R(u - u_r) \quad (3.10)$$

with $||(\cdot)||$ representing the second norm of the difference. Note that it can also be rewritten as in Eq.3.11.

$$l(x,u) = (x_u - x^r)^T Q(x_u - x^r) + (u - u_r)^T R(u - u_r) \quad (3.11)$$

Then, the cost function is defined as a summation of the running cost over all the steps (along the whole prediction horizon) as shown in Eq.3.12.

$$J_N(x,u) = \sum_{k=0}^{N-1} l(x_u(k), u(k)) \quad (3.12)$$

Now, the MPC problem can be formulated as an optimal control problem to find a minimizing control sequence as in Eq.3.13.

$$\begin{aligned} \min_u J_N(x_0,u) &= \sum_{k=0}^{N-1} l(x_u(k), u(k)) \\ subject \quad to \quad x_u(k+1) &= f(x_u(k), u(k)) \\ x(0) &= x_0 \\ u(k) &\in \mathcal{U}, \quad \forall k \in [0, N-1] \\ x_u(k) &\in \mathcal{X}, \quad \forall k \in [0, N] \end{aligned} \quad (3.13)$$

Then, the value function that is the minimum of the cost function is computed as in Eq.3.14.

$$V_N(x) = \min_u J_N(x_0,u) \quad (3.14)$$

Note that, even if this strategy is shown for a SISO system, it can also be generally applied to nonlinear MIMO (*multiple input multiple output*) systems.

In addition, the main differences between the MPC and PID controllers are:

- PID controller handles only a single input and a single output (SISO systems);
- MPC controller is a more advanced method of process control used for MIMO systems (Multiple Inputs, multiple Outputs);

- In PID controller, there is no knowledge of constraints;
- The primary advantage of MPC is its ability to deal with the constraints;
- PID controller does not have the ability to deal with the constraints;
- PID controller does not require a model of process;
- MPC controller requires the model of a process.

3.3 Leader-Follower Approach

Leader-Follower approach is used to design a controller that allows each robot to reach its desired geometric shape in the formation. The idea behind this strategy is based on the natural behaviours of animals operating as a team so that it is possible to investigate the possibilities of networking a group of systems to accomplish a given set of tasks without requiring an explicit supervisor.

In the considered case, it is assumed to have formations under decentralized control based on a Leader-Follower approach, which means that the robot motion coordination is controlled by the individual robot controllers. Hence, the follower vehicle is controlled maintaining constant the distance between the leader and the follower and having the orientation angle of the follower almost the same of the one of the leader.

The Leader-Follower approach is based on the fact that the followers do not know the trajectory that the leader tracks and have no influence on its movements. Hence, they can only rely on their own range and bearing sensors to adjust its linear and angular velocity to follow the leader. In particular, the focus is about a teleoperation strategy for a MAS and, hence, selection of leaders in the group of mobile robots, in order to achieve some optimization goals.

3.3.1 Leader-Follower approach based on distance

It is considered the case of an homogeneous MAS composed by two UGVs, where one of them is the leader and the other one is the follower. In details, the two vehicles are assumed to be unicycles, whose control inputs are the linear and angular velocities.

Considering the Cartesian plane, the leader is described by (x_L, y_L, θ_L), while the follower by (x_F, y_F, θ_F) and there exists a distance between the leader and the

follower expressed by ρ, defined as the Euclidean distance between the center of mass of the leader and of the follower as in Eq.3.15.

$$\rho = \sqrt{(x_L - x_F)^2 + (y_L - y_F)^2} \tag{3.15}$$

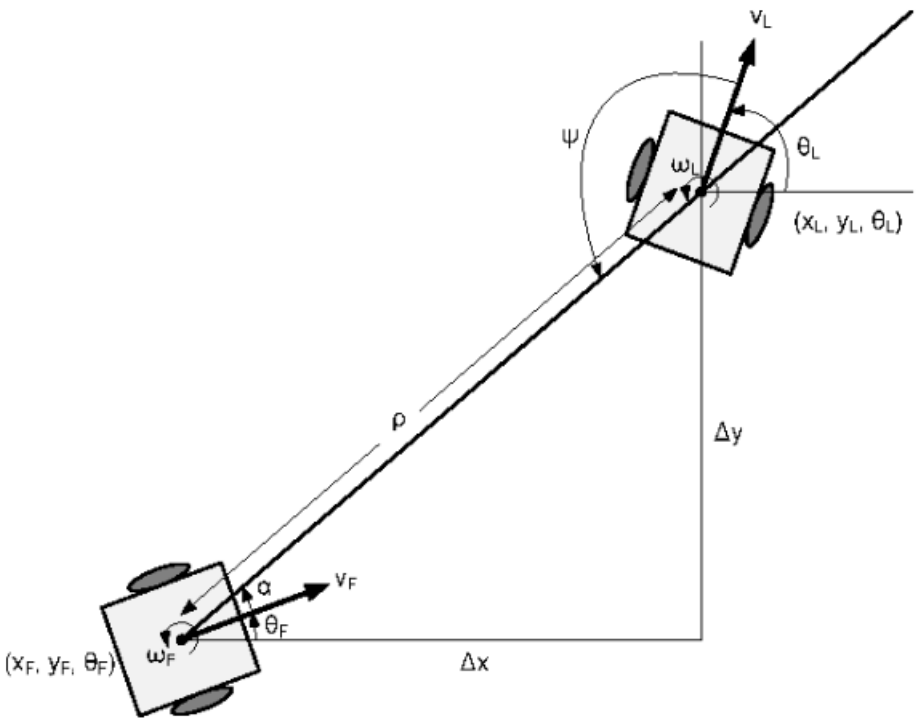


Figure 3.5: *Graphical representation of the Leader-Follower approach and its parameters applied to two unicycles [**30**].*

As it is possible to see from Fig.3.5, the follower robot is also defined by the angle α, that is the angle between follower's motion and the new line defined by ρ, mathematically given by Eq.3.16.

$$\alpha = \tan^{-1} \frac{y_L - y_F}{x_L - x_F} - \theta_F \tag{3.16}$$

Instead, the angle describing the leader orientation with respect to the one of the follower is indicated by ϕ. Such an angle is defined as the angle between the motion of the leader and the new line defined by ρ as in Eq.3.17.

$$\phi = \alpha + \theta_F - \theta_L \tag{3.17}$$

Hence, knowing α, ρ and ϕ, it is possible to define the kinematic model in polar coordinates (ρ, α, ψ), instead of using cartesian coordinates (x, y, θ), in order to coordinate the motions of the two vehicles taken into account.The

kinematic model, used to design the controller, is shown in Eq.3.18.

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & 0 \\ \sin(\alpha/\rho) & -1 \\ \sin(\alpha/\rho) & 0 \end{bmatrix} \begin{bmatrix} v_F \\ \omega_F \end{bmatrix} + \begin{bmatrix} -\cos(\phi) & 0 \\ \sin(\phi/\rho) & 0 \\ \sin(\phi/\rho) & -1 \end{bmatrix} \begin{bmatrix} v_L \\ \omega_L \end{bmatrix} \tag{3.18}$$

Such formulation of the controller aims to define the inputs v_F and ω_F of the follower such that the goal is reached. i.e. the follower follows the leader maintaining a constant relative distance with the leader.

From Eq.3.18, it is possible to observe that to define the controller and reach the goal, only the distance with respect to the leader (ρ) and the angle α must be known. Hence, the desired distance between the leader and the follower (ρ^{des}) and the desired angle between ρ^{des} and the follower's direction of motion (α^{des}) are chosen. So, the goal is to have that the real values of these two variables converge to the desired ones as the time goes to infinity, accordingly with the control law in Eq.3.19.

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \begin{bmatrix} \rho^{des} - \rho \\ \alpha^{des} - \alpha \end{bmatrix} \tag{3.19}$$

where the values of k are chosen in such a way that the system eigenvalues have a negative real part, meaning that the system is stable. In the end, the values of the linear and angular velocities of the follower (v_F and ω_F) are computed as in Eq.3.20. To reconvert back the result in the Cartesian coordinates, Eq.3.21 holds.

$$\begin{bmatrix} v_F \\ \omega_F \end{bmatrix} = \begin{bmatrix} -\frac{1}{\cos(\alpha)} & 0 \\ -\frac{\tan(\alpha)}{\rho} & -1 \end{bmatrix} \left[\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \end{bmatrix} - \begin{bmatrix} -\cos(\phi) & 0 \\ \frac{\sin(\phi)}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v_L \\ \omega_L \end{bmatrix} \right] \tag{3.20}$$

$$\begin{aligned} x_F &= x_L \cos(\theta_L) - \rho \cos(\alpha) + \theta_F \\ y_F &= y_L \sin(\theta_L) - \rho \sin(\alpha) + \theta_F \\ \theta_F &= \phi - \alpha + \theta_L - \pi \end{aligned} \tag{3.21}$$

3.3.2 Example: Numerical results of the simulation

Consider an homogeneous (with the same actuation capabilities) MAS composed by two UGVs such that one of them is the leader, while the other one is the follower. The leader knows the trajectory to track and it can be assumed as the one described in section 2.1.4.

In addition also the initial posture of both the leader and the follower other

than the values of α^{des}, ρ^{des} and k are assumed to be known and they have the values shown in Tab.3.1.

Parameter's name	Value
x_2	0.05 [m]
y_2	0.05 [m]
θ_2	0 [rad]
α^{des}	0.3
ρ^{des}	$\pi/6[rad]$
k	1

Table 3.1: *Leader-follower approach's parameters.*

Implementing the reasoning explained in section 3.3.1 and considering the leader as the UGV defined in section 2.1.4, the obtained trajectories of the leader and of the follower are the ones shown in Fig.3.6.

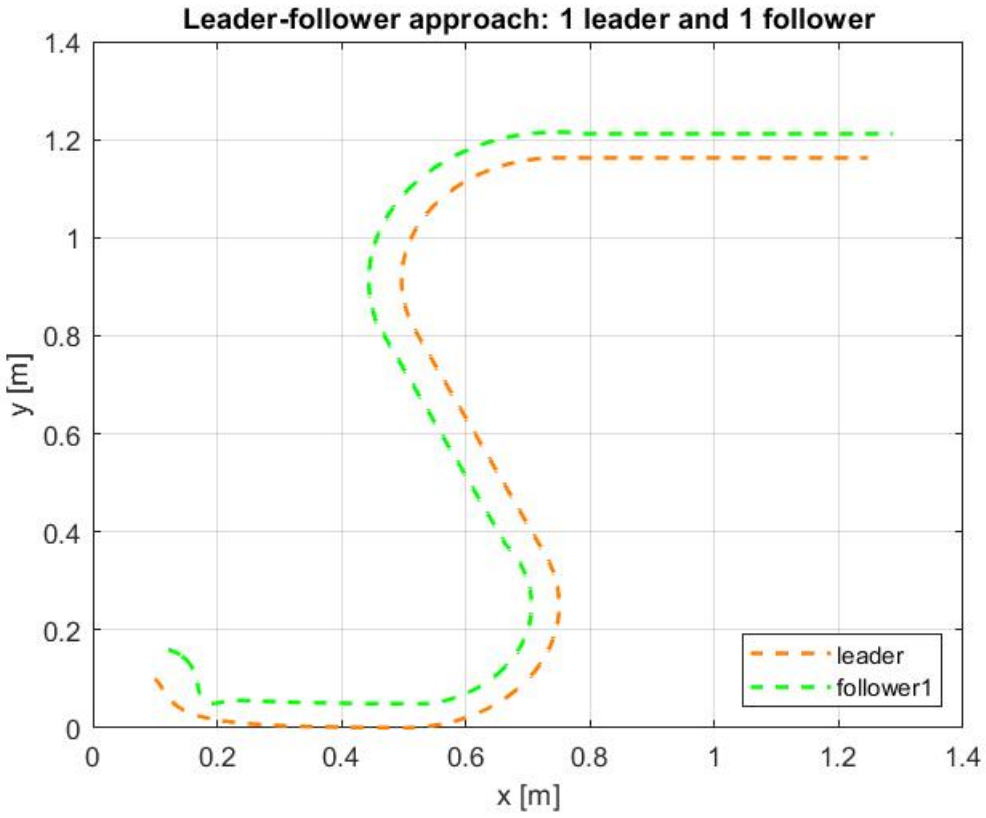


Figure 3.6: *Leader-Follower trajectories obtained applying the Leader-Follower approach based on distance.*

Looking at Fig.3.6, it is possible to note that except at the beginning, the follower tracks the same trajectory of the leader maintaining a constant distance with it.

Despite this, considering the velocities, the ones of the follower are represented in Fig.3.7. Observing this result, it is possible to note that the velocities of the follower does not coincide with the ones of the leader since, basing the approach on distance, it aims at maintaining the same distance and orientation at each time instant. Distance and orientation are the unique constraints for the follower. No requirements are on the velocities. To better understand the difference on the velocities, these are shown in Fig.3.8.

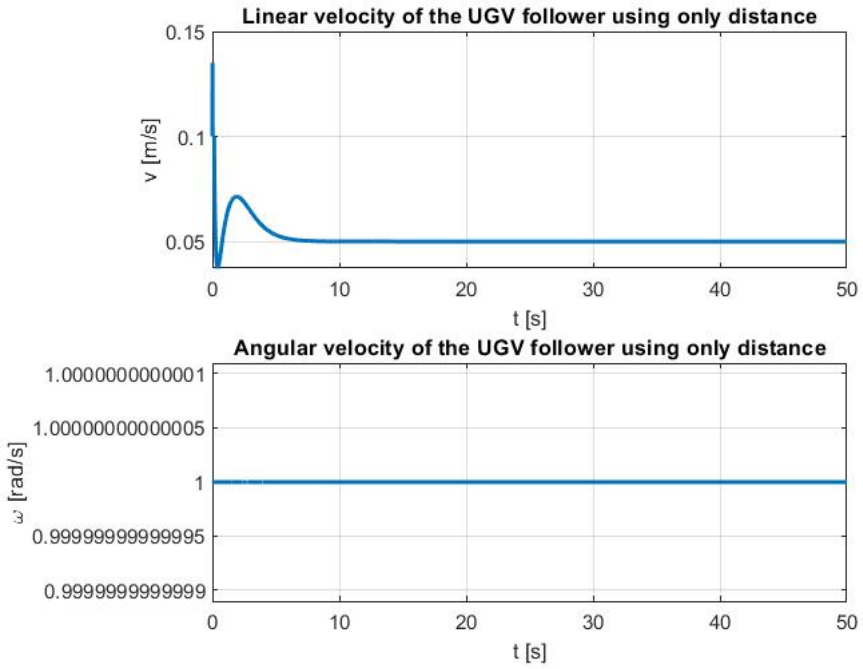


Figure 3.7: *Follower velocities obtained applying the Leader-Follower approach based on distance.*

A different way used to implement the Leader-Follower approach is based just on the knowledge of the bearing vector and of the velocities, instead of the knowledge of the distance and orientation. This strategy will be used together with the MPC control and it will be explained in details in Chapter 5.

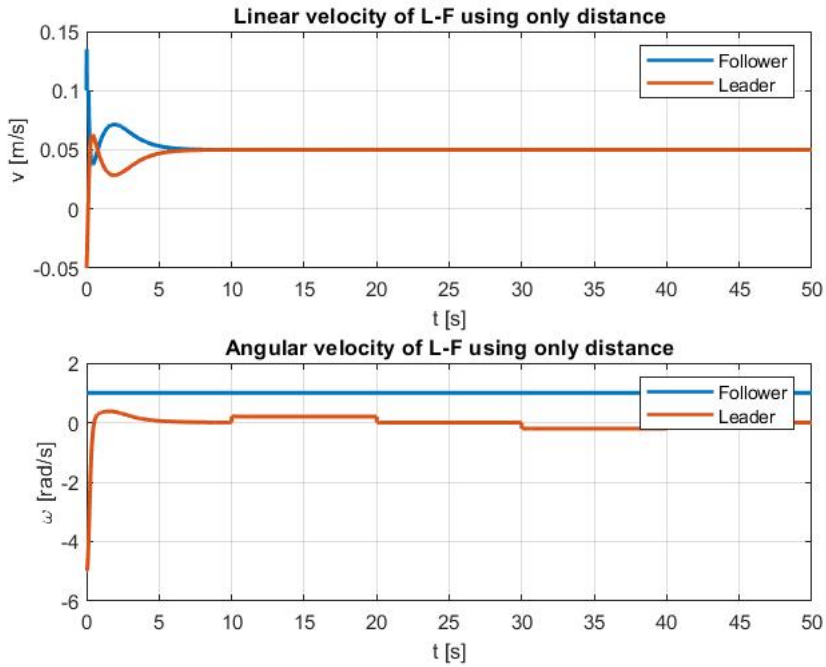


Figure 3.8: *Leader-Follower velocities obtained applying the Leader-Follower approach based on distance.*

4

Bearing Rigidity Theory

Instead of considering the distance between agents to formulate a formation control law, the bearing rigidity theory (also called *parallel rigidity theory*) can be taken into account.

This theory was developed because, even if the majority of the existing techniques for the formation assume that each agent can measure the relative positions of their nearest agents using GPS, the high accuracy requirements of the formation control task are not satisfied by this device. In addition, GPS cannot be employed indoors, underwater or in deep space, for example.

Due to this, it is preferable to use onboard sensors called bearing-only sensors that are able to measure easily the bearings. Some types of bearing-only sensors used to measure the relative bearings are: optical cameras for ground and aerial vehicles, passive radars, passive sonars and sensor arrays.

Specifically, the Bearing Rigidity theory is used to model an heterogeneous formation as a generalized framework[1] and its aim is to find the stiffness properties of given multi-element systems whose components are mutually constrained in terms of relative orientation[2].

Hence, if the bearing of each edge in the network is fixed, the conditions that define the uniqueness of the geometric pattern of a network have to be

[1]

Definition 4.0.1 (Generalized Framework) *A generalized framework is an ordered triple* $(\mathcal{G}, x, \mathcal{H})$ *consisting of a connected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with* $|\mathcal{V}| = n \geq 3$ *and* $|\mathcal{E}| = m$*, a configuration* $x \in \mathbb{R}^{3n} \times \mathbb{S}^{3n}$ *and a collection of instantaneous variation domains* $\mathcal{H} = \mathcal{I}_1, ..., \mathcal{I}_n$

[2]Cenedese A., Michieletto G., Pozzan B., Zelazo D., *Heterogeneous Formation Control: a Bearing Rigidity Approach*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021, Austin, Texas

established. The bearing rigidity theory is based on the assumption that all the agents of the considered system are characterized by local communication and bearing sensing capabilities.

So, the Bearing Rigidity theory studies the conditions such that two networks have the same geometric pattern if they have the same bearings. In particular, for a given formation, described by a network $(\mathcal{G}, p)$, the edge and the bearing vectors for $(i, j) \in \mathcal{E}$ are defined as $e_{ij} = p_j - p_i$ and $g_{ij} = \frac{e_{ij}}{||e_{ij}||}$, respectively. The bearing vector g_{ij} is the unit vector pointing from p_i to p_j and it represents the relative bearing of p_i with respect to p_j[3].

To better understand this, looking at Fig.4.1(a), the two formations have the same bearings but different geometric patterns, which imply that they are not bearing rigid. Instead, looking at Fig.4.1(b), the two formations have the same bearings and geometric pattern, so they are bearing rigid.

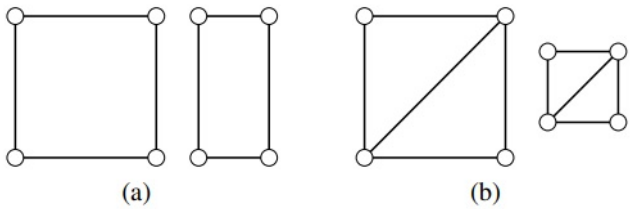


Figure 4.1: *Examples of bearing rigid formations [**44**].*

The key ingredients of the Bearing Rigidity theory are the *Bearing Function* and the *Bearing Rigidity matrix*. The *Bearing Function* summarizes the information on the available measurements and its formal definition is reported in Def.4.0.2.

Definition 4.0.2 (Bearing Function) [4] *Given a formation modeled as a generalized framework $(\mathcal{G}, x, \mathcal{H})$, the bearing function is the map associating the configuration $x \in \mathbb{R}^{3n} \times \mathbb{S}^{3n}$ to the vector $\boldsymbol{b}_{\mathcal{G}}(x) = [b_1^T ... b_m^T] \in \mathbb{S}^{2m}$ stacking all the available bearing measurements.*

Instead, the *Bearing Rigidity matrix* describes the relation between the command vector and the time derivative of the bearing measurements vector of an heterogeneous MAS and it is defined as in Def.4.0.3.

[3] Zelazo D., Zhao S., Bearing Rigidity Theory and its Applications for Control and Estimation of Network Systems, March 14, 2018

[4] Cenedese A., Michieletto G., Pozzan B., Zelazo D., *Heterogeneous Formation Control: a Bearing Rigidity Approach*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021, Austin, Texas, Definition 2.2, pg.3

Definition 4.0.3 (Bearing Rigidity matrix) [5] *Given a formation modeled as a generalized framework* $(\mathcal{G}, x, \mathcal{H})$*, the bearing rigidity matrix is the matrix* $\boldsymbol{B}_{\mathcal{G}}(x) \in \mathbb{R}^{3m \times c}$ *that satisfies the relation* $\dot{\boldsymbol{b}}_{\mathcal{G}}(x) = \boldsymbol{B}_{\mathcal{G}}\delta$.

The computation of such matrix allows to investigate the configurations that affect the formation shape, defined in Def.4.0.4.

Definition 4.0.4 (Formation shape) [6] *Given a formation modeled as a generalized framework* $(\mathcal{G}, x, \mathcal{H})$*, its shape is characterized by the collection of all possible bearing measurements, namely by the vector* $b_k(x) \in \mathbb{S}^{2n(n-1)}$ *where* $\mathcal{K}$ *is the complete graph associated to* $\mathcal{G}$.

In addition, about Bearing Rigidity, it is possible to identify three different notions. These are:

- *Bearing Rigidity*: A framework $(\mathcal{G}; \mathbf{p})$ is bearing rigid if there exists an $\epsilon > 0$ such that every framework $(\mathcal{G}; \mathbf{p'})$ which is bearing equivalent to $(\mathcal{G}; \mathbf{p})$ and satisfies $||\mathbf{p}_i - \mathbf{p}'_i|| < \epsilon$ for all $v_i \in \mathcal{V}$, is bearing congruent to $(\mathcal{G}; \mathbf{p})$.
- *Global Bearing Rigidity*: A framework $(\mathcal{G}; \mathbf{p})$ is globally bearing rigid if all bearing equivalent frameworks to $(\mathcal{G}; \mathbf{p})$ are also bearing congruent to $(\mathcal{G}; \mathbf{p})$.
- *Infinitesimal Bearing Rigidity*: A framework $(\mathcal{G}; \mathbf{p})$ is infinitesimally bearing rigid if every possible infinitesimal motion is trivial. The term "infinitesimal" is due to the fact that the bearing rigidity matrix is the first order derivative of the bearing vectors with respect to the position of the nodes.

Between them, just the Infinitesimal Bearing rigidity is a global property[7] practically used, while the first two cannot ensure unique geometric patterns of networks.

To allow the definition of a mathematical condition and to ensure a unique geometric pattern of a network, the Infinitesimal Bearing Rigidity is based on two main assumptions:

1. The first one is a geometric property: if the network is infinitesimally bearing rigid, the uniqueness of the positions of the nodes in the network

[5] Cenedese A., Michieletto G., Pozzan B., Zelazo D., *Heterogeneous Formation Control: a Bearing Rigidity Approach*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021, Austin, Texas, Definition 3.1, pg.3

[6] ibidem

[7] It means that the bearings can uniquely determine the geometric pattern of a network.

is guaranteed up to a translational and scaling factor. The translational and scaling motions that guarantee this belong to the vectors in $span\{\mathbf{1}_n \otimes I_d, p\}$.

According to this, looking at Fig.4.2, the networks are not bearing rigid because they have nontrivial infinitesimal bearing motions.

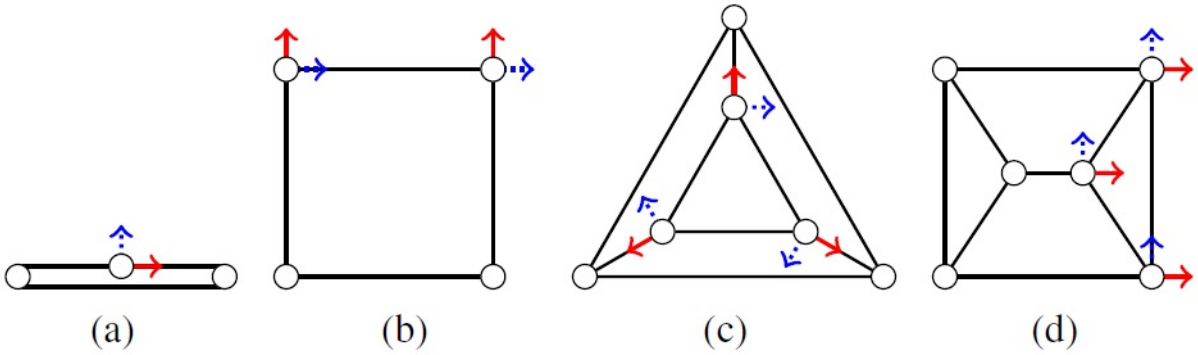


Figure 4.2: *Examples of nontrivial infinitesimal bearing motions, that imply non-infinitesimal bearing networks [**44**].*

2. The second property is an algebraic property: a network is infinitesimally bearing rigid in d-dimensional space if and only if the bearing rigidity matrix $\mathbf{R}_B$ satisfies Eq.4.1 or, equivalently, Eq.4.2.

$$Null(\mathbf{R}_B) = span\{\mathbf{1}_n \otimes \mathbf{I}_d, \mathbf{p}\} \tag{4.1}$$

$$rank(\mathbf{R}_B) = dn - d - 1 \tag{4.2}$$

For example, looking at Fig.4.3, it can be noted that all the networks satisfy Eq.4.2.

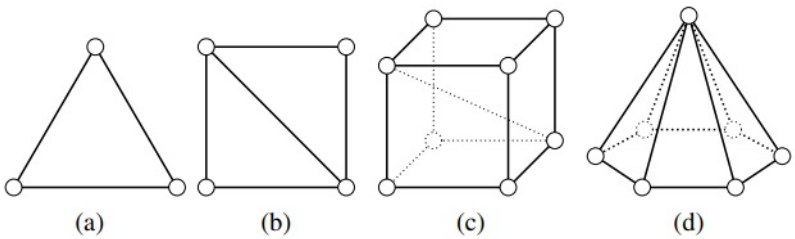


Figure 4.3: *Example of infinitesimal bearing networks [**44**].*

Alternatively, the *Bearing Laplacian* matrix defines a necessary and sufficient condition to have Infinitesimal Bearing Rigidity. Such matrix of a network can

be considered as a Weighted Graph Laplacian matrix. In details, the weights of this matrix are matrices, too. For this reason, the Bearing Laplacian matrix provides the values of the edge bearings and describes the topological structure of the network.

So, considering the Bearing Laplacian matrix, a network is IBR if and only if Eq.4.3 or, equivalently, Eq.4.4 are satisfied.

$$Null(\mathcal{B}) = span\{\mathbf{1}_n \otimes \mathbf{I}_d, \mathbf{p}\} \tag{4.3}$$

$$rank(\mathcal{B}) = dn - d - 1 \tag{4.4}$$

The main differences between the Bearing Laplacian matrix and the Bearing Rigidity matrix are in the case of direct graphs, where these two matrices have different ranks and null spaces. Instead, for undirect graphs, the Bearing Laplacian matrix is symmetric and positive semi-definite, while the Bearing Rigidity matrix no.

Another aspect of great interest is the construction of a bearing rigid network by adding well-placed edges and nodes in a network. This is due to the fact that the IBR of a network is more dependent on the graph than its configuration, while a network is described by its graph and configuration, with the same relevance.

In addition also the concept of *generically bearing rigid* can be introduced: given a graph, if there exists at least one configuration such that the network is IBR, then for almost all configurations the corresponding networks are IBR.

It follows that: if a graph is not generically bearing rigid, then the corresponding network is not IBR for any configuration. Hence, in order to construct IBR networks, generically bearing rigid graphs have to be constructed. This can be summarized through Theorem 4.0.1.

Theorem 4.0.1 (Rigidity Properties) [8] *For a framework* $(\mathcal{G}; \boldsymbol{p})$ *in* $\mathbb{R}^3$*, the following implications are valid*

- *bearing global rigidity* $\Longleftrightarrow$ *bearing rigidity;*
- *bearing infinitesimally rigidity* $\Rightarrow$ *bearing global rigidity;*
- *bearing infinitesimally rigidity* $\Rightarrow$ *bearing rigidity.*

[8] Cenedese A., Franchi A., Michieletto G.,*Bearing Rigidity Theory in SE(3)*, in IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, USA, Theorem 2, pg.4

One of the most common methods to construct rigid graph is the *Henneberg construction*, originally proposed for the distance rigidity theory. In the particular case of a network composed by two vertices connected by an edge, the Henneberg construction results in a *Laman graph*. In the Bearing Rigidity theory, the main result about Laman graphs is that:

If the underlying graph of a network is Laman, then the network is infinitesimally bearing rigid for almost all configurations in an arbitrary dimension. Again, for networks in the plane, a graph is generically bearing rigid if and only if it is Laman[9].

Instead, the general conditions that have to hold to obtain Infinitesimal bearing rigidity are exploited in Theorem 4.0.2.

Theorem 4.0.2 (Condition for Infinitesimal Bearing Rigidity) [10] *For a framework* $(\mathcal{G}; \boldsymbol{p})$ *in* $\mathbb{R}^3$*, the following statements are equivalent:*

1. $(\mathcal{G}; \boldsymbol{p})$ *is infinitesimally bearing rigid;*
2. $rank(\boldsymbol{B}_{||,\mathcal{G}(\boldsymbol{p})}) = 3n - 4$*;*
3. $\mathcal{N}(\boldsymbol{B}_{||,\mathcal{G}(\boldsymbol{p})}) = span\{\boldsymbol{1} \otimes \boldsymbol{I}_3, \boldsymbol{p}\}$.

One of the main applications of the Bearing Rigidity theory is the Bearing-Only Formation Control. The aim of the Bearing Only Formation Control consists in steering a group of mobile agents to form a desired geometric pattern predefined by inter-neighbor bearings using bearing-only measurements. The unique requirement is that each agent measures the relative bearings of their neighbors, while the relative positions cannot be measured or estimated.

Hence, it is assumed that the target formation is specified by constant bearing constraints $\{\mathbf{g}_{ij}^*\}_{i,j\in\mathcal{E}}$ and there are no leaders. The control objective is to control the positions of the agents $\{\mathbf{p}_i(t)\}_{i\in\mathcal{V}}$ such that $\mathbf{g}_{ij}(t) \to \mathbf{g}_{ij}^*$ for all $(i,j) \in \mathcal{E}$ as $t \to \infty$, where all the bearings are defined in the Inertial World Frame.

The nonlinear control law that can be used to solve the bearing-only formation control problem is in Eq.4.5.

$$\dot{\mathbf{p}}_i(t) = -\sum \mathbf{P}_{\mathbf{g}_{ij}(t)}\mathbf{g}_{ij}^* \qquad i \in \mathcal{V} \tag{4.5}$$

where $\mathbf{P}_{\mathbf{g}_{ij}(t)} = \mathbf{I}_d - \mathbf{g}_{ij}(t)\mathbf{g}_{ij}^T(t)$.

[9] H.-S. Ahn, M. H. Trinh, S. Zhao, Z. Sun, D. Zelazo, *Laman Graphs are Generically Bearing Rigid in Arbitrary Dimensions*, 11 March 2017

[10] Cenedese A., Franchi A., Michieletto G.,*Bearing Rigidity Theory in SE(3)*, in IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, USA, Theorem 1, pg.4

The control law Eq.4.5 has three main properties:

- the control of each agent only requires bearing measurements and does not require distance or position estimation;
- the control input of such control law is always bounded as $||\dot{\mathbf{p}}_i(t)|| \leq \sum||\mathbf{P}_{\mathbf{g}_{ij}(t)}||||\mathbf{g}^*_{ij}|| = |\mathcal{N}_i|$ since $||\mathbf{P}_{\mathbf{g}_{ij}(t)}|| = ||\mathbf{g}^*_{ij}|| = 1$;
- the centroid, defined as the average position of the agents, and the scale, defined as the standard deviation of the distances from the agents to the centroid, of the formation are invariant under the control law.

4.1 Bearing Rigidity Theory in $\mathbb{SE}(3)$

Bearing Rigidity theory can also be applied to describe the rigidity properties for frameworks embedded in the 3-dimensional special euclidean space $\mathbb{SE}(3)$, defined as in Def.4.1.1.

Definition 4.1.1 ($\mathbb{SE}(3)$ framework) [11] *An SE(3) framework is a triple $(\mathcal{G}, \boldsymbol{p}, \boldsymbol{a})$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph, $\boldsymbol{p} : \mathcal{V} \to \mathbb{R}^3$ is a function mapping each node to a point in $\mathbb{R}^3$ (position) and $\boldsymbol{a} : \mathcal{V} \to \mathbb{SO}(3)$ is a function associating each node with an element of $\mathbb{SO}(3)$ (attitude).*

The goal is to maintain a rigid formation preserving the measurements and identifying the framework transformations, which implies that the relative poses among the agents in terms of distance and/or directions are preserved. Hence, the aim consists in finding the allowed motions that do not modify the whole system in terms of inter-agent bearings.

Recalling that a MAS is represented through a direct graph and that, in $\mathbb{SE}(3)$, each agent is described by a position and attitude in the 3-dimensional space, the agents orientation are expressed through rotation matrices, which belong to the Special Orthogonal Group $\mathbb{SO}(3)$[12].

The position and the attitude of node $v_i \in \mathcal{V}$ is denoted by $(\mathbf{p}(v_i), \mathbf{a}(v_i)) = (\mathbf{p}_i, \mathbf{R}_i) = (\mathbf{X}_p(i), \mathbf{X}_a(i)) = \mathbf{X}(i) \in \mathbb{SE}(3)$, whereas $\mathbf{p}(\mathcal{V}) = \mathbf{X}_p(\mathcal{V})) \in \mathbb{R}^{3n}$ and

[11] Cenedese A., Franchi A., Michieletto G.,*Bearing Rigidity Theory in SE(3)*, in IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, USA, Definition 4, pg.4

[12] This group includes all the 3×3 orthogonal matrices having unit determinant. The attitude of each agent i has to be interpreted as a rotation matrix $\mathbf{R}_i \in \mathbb{R}^{3\times3}$ such that $\mathbf{R}_i\mathbf{R}_i^T = \mathbf{I}_3$ and $det(\mathbf{R}_i) = +1$.

$\mathbf{a}(\mathcal{V}) = \mathbf{X}_a(\mathcal{V}) \in \mathbb{SO}(3)^n$ indicate the position and attitude components of the complete framework configuration, respectively[13].

In addition, in $\mathbb{SE}(3)$, the following assumptions[14] hold:

1. the underlying graph is direct;
2. the bearing vectors are expressed in the Body Frame of each agent.

Hence, agent i accesses the bearing of agent j if and only if the directed edge (v_i, v_j) belongs to the graph $\mathcal{G}$. Furthermore, the relative bearing $\mathbf{b}_{ij} \in \mathbb{S}^2$ is measured from the Body Frame of the $i-th$ agent, but it can be expressed in terms of the relative positions and attitudes of the two points with respect to the World Frame. So, it holds:

$$\mathbf{b}_{ij} = \mathbf{R}_i^T \frac{\mathbf{p}_i - \mathbf{p}_j}{||\mathbf{p}_i - \mathbf{p}_j||} = \mathbf{R}_i^T \bar{\mathbf{p}}_{ij}$$

where:

- the matrix $\mathbf{R}_i^T$ is the rotation matrix describing the orientation of the World Frame with respect to the Body Frame of agent i;
- $\bar{\mathbf{p}}_{ij}$ represents the normalized relative position vector from i to j.

The information about bearing measurements can be handled by defining the $\mathbb{SE}(3)$-rigidity function associated to the framework, namely the function $\mathbf{b}_{\mathcal{G}} : \mathbb{SE}(3)^n \to \mathbb{S}^{2m}$ such that

$$\mathbf{b}_{\mathcal{G}}(\mathbf{X}(\mathcal{V})) = [\mathbf{b}_1^T ... \mathbf{b}_m^T]^T$$

where $\mathbf{b}_k$ denotes the measurement on the $k-th$ directed edge in the graph $\mathcal{G}$. It can be proven that the $\mathbb{SE}(3)$-rigidity function can be rewritten in the more compact form:

$$\mathbf{b}_{\mathcal{G}}(\mathbf{X}(\mathcal{V})) = -diag(\{\frac{\mathbf{R}_i^T}{||\mathbf{p}_i - \mathbf{p}_j||}\})\bar{\mathbf{E}}^T \mathbf{X}_p(\mathcal{V})$$

[13]Cenedese A., Michieletto G., Pozzan B., Zelazo D., *Heterogeneous Formation Control: a Bearing Rigidity Approach*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021, Austin, Texas

[14]These assumptions are justified by real multi-agent scenarios where a robot can gather the relative bearings between itself and other robots through sensors attached to its Body Frame such as robots flying in 3D with onboard cameras.

In such a scenario, a framework $(\mathcal{G}, \mathbf{p}, \mathbf{a})$ is $\mathbb{SE}(3)$-rigid if and only if for any $\mathbf{X}'(\mathcal{V}) \in \mathbb{SE}(3)^n$ sufficiently close to $\mathbf{X}(\mathcal{V})$ with the same bearing measurements, namely $\mathbf{b}_{\mathcal{G}}(\mathbf{X}(\mathcal{V})) = \mathbf{b}_{\mathcal{G}}(\mathbf{X}'(\mathcal{V}))$, there exists a local bearing preserving transformation taking $\mathbf{X}(\mathcal{V})$ to $\mathbf{X}'(\mathcal{V})$.

Definition 4.1.2 (Rigidity in $\mathbb{SE}(3)$) [15] *A framework $(\mathcal{G}, \boldsymbol{p}, \boldsymbol{a})$ is rigid in $\mathbb{SE}(3)$ if there exists a neighborhood $S \subset \mathbb{SE}(3)^n$ of $\boldsymbol{X}(\mathcal{V})$ such that :*

$$\boldsymbol{b}_{k_n}^{-1}(\boldsymbol{b}_{k_n}(\boldsymbol{X}(\mathcal{V}))) \cap \boldsymbol{S} = \boldsymbol{b}_{\mathcal{G}}^{-1}(\boldsymbol{b}_{\mathcal{G}}(\boldsymbol{X}(\mathcal{V}))) \cap \boldsymbol{S}$$

where $\boldsymbol{b}_{k_n}^{-1}(\boldsymbol{b}_{k_n}(\boldsymbol{X}(\mathcal{V})))$ denotes the pre-image of the point $\boldsymbol{b}_{k_n}(\boldsymbol{X}(\mathcal{V})))$ under the $\mathbb{SE}(3)$-rigidity map.

4.2 Bearing Rigidity theory applied to a group of UAVs quadrotors

As said in the previous section, Bearing Rigidity theory is used to formulate a formation control law. Hence, in this section, it is presented the application of the Bearing Rigidity theory to define a formation control law for a group of N quadrotors.

It is assumed that the dynamics of the quadrotor are described by the simplified model in Eq.4.6.

$$\begin{bmatrix} \dot{\mathbf{p}}_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \omega_i \end{bmatrix} \tag{4.6}$$

where $\dot{\mathbf{p}}_i \in \mathbb{R}^3$ is the UAV position in $\mathcal{F}_W$, ϕ_i is the yaw angle and $\mathbf{R}_i = \mathbf{R}_z(\phi_i) \in \mathbb{SO}(3)$ is the elementary rotation around the z-axis in $\mathcal{F}_W$, $\mathbf{u}_i \in \mathbb{R}^3$ is the known and controllable linear velocity in $\mathcal{F}_B$ and $\omega_i \in \mathbb{R}$ is the known and controllable yaw rate in $\mathcal{F}_B$. In addition, it is also assumed that each quadrotor is equipped with onboard cameras able to measure the relative bearings in its Body Frame with respect to the other UAVs.

A bearing rigid framework $(\mathcal{G}, \mathbf{q}) \in \mathbb{R}^3 \times \mathbb{S}^1$ is considered and the Bearing Rigidity matrix in the World Frame $(B_{\mathcal{G}}^W)$, defined as in Eq.4.7, is constructed. In particular, its $k-th$ row block associated to the edge $e_k = (i, j)$ is defined as in

[15] Cenedese A., Franchi A., Michieletto G.,*Bearing Rigidity Theory in SE(3)*, in IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, USA, Definition 7, pg.5

Eq.4.8.

$$\begin{aligned} \mathbf{B}_{\mathcal{G}}^{W}(\mathbf{q}) &= \frac{\partial \beta_{\mathcal{G}}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{3|\mathcal{E}|\times 4N} \\ \beta_{\mathcal{G}}(\mathbf{q}) &= [\beta_{e_1}^T ... \beta_{e_{|\mathcal{E}|}}^T]^T \end{aligned} \tag{4.7}$$

$$\begin{bmatrix} -\mathbf{0}- & -\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}} & -\mathbf{0}- & \frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}} & -\mathbf{0}- & \dots & -\mathbf{S}\beta_{ij} & -\mathbf{0}- \end{bmatrix} \in \mathbb{R}^{3\times 4N} \tag{4.8}$$

where $-\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}$ is in column i, $\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}$ is in column j and $-\mathbf{S}\beta_{ij}$ is in column $3N + i$. In details, $d_{ij} = ||\mathbf{p}_i - \mathbf{p}_j||$, $\mathbf{P}_{ij} = \mathbf{I}_3 - \beta_{ij}\beta_{ij}^T$ is the orthogonal projector onto the orthogonal complement of β_{ij}, and $\mathbf{S} = [[0 \quad 0 \quad 1]^T]_x$, where $[\cdot]_x$ represents the skew-symmetric operator.

Looking at Eq.4.8, it is possible to observe that $\mathbf{B}_{\mathcal{G}}^{W}$ is function of interdistances (d_{ij}), relative bearings (β_{ij}), and absolute yaw rotations ($\mathbf{R}_i^T$). This matrix is employed to describe the relation between the bearing function ($\beta_{\mathcal{G}}$) and the World frame velocities ($\dot{\mathbf{q}} = (\dot{\mathbf{p}}, \dot{\phi})$) as in Eq.4.9.

$$\dot{\beta}_{\mathcal{G}} = \mathbf{B}_{\mathcal{G}}^{W} \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\phi} \end{bmatrix} \tag{4.9}$$

The Bearing Rigidity matrix can also be defined in the Body Frame assuming to know the Body Frame velocity inputs of the quadrotors ($\mathbf{u} = [...\mathbf{u}_i^T ...]^T \in \mathbb{R}^{3N}$ and $\omega = [...\omega_i ...]^T \in \mathbb{R}^N$) as shown in Eq.4.10.

$$\dot{\beta}_{\mathcal{G}} = \mathbf{B}_{\mathcal{G}}^{W} \begin{bmatrix} diag(\mathbf{R}_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_N \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \omega \end{bmatrix} = \mathbf{B}_{\mathcal{G}}(\mathbf{q}) \begin{bmatrix} \mathbf{u} \\ \omega \end{bmatrix} \tag{4.10}$$

In this case, the $k-th$ row block of the Body Frame Rigidity matrix $B_{\mathcal{G}}(\mathbf{q})$ associated to the edge $e_k = (i, j)$ is:

$$\begin{bmatrix} -\mathbf{0}- & -\frac{\mathbf{P}_{ij}}{d_{ij}} & -\mathbf{0}- & \frac{\mathbf{P}_{ij}\mathbf{R}_j^i}{d_{ij}} & -\mathbf{0}- & \dots & -\mathbf{S}\beta_{ij} & -\mathbf{0}- \end{bmatrix} \in \mathbb{R}^{3\times 4N} \tag{4.11}$$

where $\mathbf{R}_j^i = \mathbf{R}_z(\phi_{ij})$ with $\phi_{ij} = \phi_j - \phi_i$.

As for the Bearing Rigidity matrix defined in the World Frame, the Body Frame Bearing Rigidity matrix depends on the measured bearings β_{ij} and interdistances d_{ij}. However, it does not depend anymore on the absolute yaw rotations, but on the relative orientations $\phi_j - \phi_i$.

Then, the bearing formation control error has to be defined as:

$$e_F(q) = \mathbf{b}_{\mathcal{G}}^{d} - \beta_{\mathcal{G}}(q) \tag{4.12}$$

and it has to converge to zero to solve the formation stabilization problem. Such a minimization can be implemented by the scale-free controller based on the Body Frame rigidity matrix as in Eq.4.13.

$$\begin{bmatrix} \mathbf{u} \\ \omega \end{bmatrix} = k_c \begin{bmatrix} diag(d_{ij}) & 0 \\ 0 & \mathbf{I}_N \end{bmatrix} \mathbf{B}_{\mathcal{G}}(\mathbf{q})^T b^d_{\mathcal{G}}, \qquad k_c > 0 \tag{4.13}$$

so that the $i-th$ agent velocity is given by Eq.4.14.

$$\begin{cases} u_i = -k_c \sum_{(i,j)\in\mathcal{E}} \mathbf{P}_{ij}\beta^d_{ij} + k_c \sum_{(j,i)\in\mathcal{E}} \mathbf{R}^i_j \mathbf{P}_{ji}\beta^d_{ji} \\ \omega_i = k_c \sum_{(i,j)\in\mathcal{E}} \beta^T_{ij}\mathbf{S}\beta^d_{ij} \end{cases} \tag{4.14}$$

From Eq.4.14, it can be observed that the velocities depend on the graph $\mathcal{G}$ and on the relative quantities ($\mathbf{P}_{ij}$, β_{ij}, $\mathbf{R}^i_j$) according to the requirement about the communication between agents. Instead, no distance measurements and World Frame have to be known.

4.3 Bearing Rigidity theory applied to a group of UGVs unicycles

It is considered a group of N mobile agents, in particular UGVs, in $\mathbb{R}^2$, where $N \geq 2$. The position of an agent $i \in \{1, ..., N\}$ at time t is represented by the vector $\mathbf{p}_i(t) \in \mathbb{R}^2$, while the configuration of the group of N UGVs is represented by $\mathbf{p} = [p_1^T ... p_N^T]$. As said in Chapter 2, a MAS is represented through a graph $\mathcal{G}$, with the characteristics presented in section 2.2.1. In particular, considering a generic edge $(i, j) \in \mathcal{E}$, it holds that agent i can measure the relative bearing of agent j, which means that agent j is a neighbor of agent i. Hence, the formation denoted as $(\mathcal{G}, \mathbf{p})$ is $\mathcal{G}$ with its vertex i mapped to p_i for all $i \in \mathcal{V}$. The edge vector for (i, j) is defined as:

$$e_{ij} = p_j - p_i \tag{4.15}$$

while the bearing vector for the edge (i, j), which represents the relative bearing of p_j with respect to p_i, is defined as:

$$\mathbf{g}_{ij} = \frac{e_{ij}}{||e_{ij}||} \tag{4.16}$$

where $||\cdot||$ denotes the Euclidean norm of a vector or the spectral norm of a matrix. Consequently, for the bearing vector g_{ij}, the orthogonal projection

matrix is defined as:

$$\mathbf{P}_{\mathbf{g}_{ij}} = \mathbf{I}_2 - \mathbf{g}_{ij}\mathbf{g}_{ij}^T \in \mathbb{R}^{2\times 2} \tag{4.17}$$

with $I_2 \in \mathbb{R}^{2\times 2}$ representing the identity matrix. Then, from Eq.4.16, it follows that the time derivative of g_{ij} is given by:

$$\dot{\mathbf{g}}_{ij} = \frac{\mathbf{P}_{\mathbf{g}_{ij}}}{||e_{ij}||}\dot{e}_{ij} \tag{4.18}$$

In this way, since $\mathbf{P}_{\mathbf{g}_{ij}}\mathbf{g}_{ij} = 0$, it holds that $\dot{\mathbf{g}}_{ij}$ is orthogonal to $\mathbf{g}_{ij}$ and e_{ij}. It follows that: $\mathbf{g}_{ij}^T\dot{\mathbf{g}}_{ij} = 0$ and $e_{ij}^T\dot{\mathbf{g}}_{ij} = 0$.

Without loss of generality, it is also possible to assume that the first n_l agents are leaders belonging to the set $\mathcal{V}_l = \{1, ..., n_l\}$, while the others $n_f = n - n_l$ agents are followers, belonging to the set $\mathcal{V}_f = \mathcal{V}/\mathcal{V}_l$. The positions of the leader is denoted as $\mathbf{p}_l = [p_1^T, ..., p_{n_l}^T]^T$, while the ones of the followers are represented as $\mathbf{p}_f = [p_{n_l+1}^T, ..., p_n^T]^T$, so that $\mathbf{p} = [\mathbf{p}_l^T \quad \mathbf{p}_f^T]^T$. From these, it follows that the velocities of the leaders are $\dot{\mathbf{p}}_l = v_l$, while the ones of the followers are $\dot{\mathbf{p}}_f = v_f$.

In order to apply the Bearing Rigidity theory to the group of UGVs, a desired target formation that the agents should achieve has to be defined as in Def.4.3.1, under the assumption that the leaders move at a constant velocity $v_l \in \mathbb{R}^2$.

Definition 4.3.1 (Target Formation) [16] *The target formation $(\mathcal{G}, p^*(t))$ satisfies the constant interneighbor bearings $\{\boldsymbol{g}_{ij}^*\}_{(i,j)\in\mathcal{E}}$ and the time-varying leader positions $\{\boldsymbol{p}_i^*(t)\}_{i\in\mathcal{V}_l}$.*

The bearing Laplacian matrix $\mathcal{B} \in \mathbb{R}^{2n\times 2n}$ is now defined in sub-blocks as shown in Eq.4.19

$$[\mathcal{B}]_{ij} = \begin{cases} \mathbf{0}_{d\times d}, & i \neq j, (i,j) \notin \mathcal{E} \\ -\mathbf{P}_{\mathbf{g}_{ij}^*}, & i \neq j, (i,j) \in \mathcal{E} \\ \sum_{k\in\mathcal{N}_i} \mathbf{P}_{\mathbf{g}_{ik}^*}, & i = j, i \in \mathcal{V} \end{cases} \tag{4.19}$$

In addition, according to the partition of leader and follower agents, $\mathcal{B}$ can be partitioned to:

$$\mathcal{B} = \begin{bmatrix} \mathcal{B}_{ll} & \mathcal{B}_{lf} \\ \mathcal{B}_{fl} & \mathcal{B}_{ff} \end{bmatrix} \tag{4.20}$$

Given such partition, the condition of uniqueness of the target formation is

[16] Zhao S., Zhengtao D, Zhenhong Li, Bearing-Only Formation Tracking Control of Multi-Agent Systems, in IEEE Transactions on Automatic Control, 2019, Manchester

guaranteed by the following Lemma.

Lemma 4.3.1 (Condition for unique target formation) [17] *The target configuration $p^*(t)$ can be uniquely determined by the bearings $\{g^*_{ij}\}_{(i,j)\in\mathcal{E}}$ and leader positions $\{p^*_i\}_{i\in\mathcal{V}_l}$ if and only if $\mathcal{B}_{ff}$ is nonsingular.*

Precisely, if $\mathcal{B}_{ff}$ is non-singular, the position and the velocity of the followers in the target formation are uniquely determined as:

$$\mathbf{p}^*_f(t) = -\mathcal{B}_{ff}^{-1}\mathcal{B}_{fl}\mathbf{p}^*_l(t) \tag{4.21}$$

$$v^*_f(t) = -\mathcal{B}_{ff}^{-1}\mathcal{B}_{fl}v^*_l(t) \tag{4.22}$$

The control problem that has to be solved is to steer the agents to achieve the target formation based on the bearing measurements, which can be alternatively expressed as designing a control input for agent $i \in \mathcal{V}_f$ based merely on the bearing measurements $\{\mathbf{g}_{ij}(t)\}_{j\in\mathcal{N}_i}$ and the varying rate of the bearings $\{\dot{\mathbf{g}}_{ij}(t)\}_{j\in\mathcal{N}_i}$ such that $\mathbf{g}_{ij} \to \mathbf{g}^*_{ij}$ for all $(i,j) \in \mathcal{E}$ as $t \to \infty$[18]. In order to solve this problem, the UGVs are considered as single integrators with model:

$$\dot{\mathbf{p}}_i(t) = \mathbf{u}_i(t) \tag{4.23}$$

where $\mathbf{u}_i(t)$ is the velocity input to be designed. It is first considered the case of a stationary formation, where one UGV leader is stationary: $v_l = 0$, such that $\dot{\mathbf{p}}_i = 0$ for $i \in \mathcal{V}_l$. For the other UGVs, i.e. the followers, the relative control law is:

$$\dot{\mathbf{p}}_i(t) = \sum_{j\in\mathcal{N}_i}(\mathbf{g}_{ij}(t) - \mathbf{g}^*_{ij}), \qquad i \in \mathcal{V}_f \tag{4.24}$$

Considering an oriented graph $\mathcal{G}$ and its incidence matrix $D_{\mathcal{G}}$, the bearing vectors $\mathbf{g}(t)$ and the desired bearing vectors $\mathbf{g}^*(t)$, the control law in Eq.4.24 can be rewritten as:

$$\dot{\mathbf{p}} = -\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_{dn_f} \end{bmatrix}\bar{D}^T_{\mathcal{G}}(\mathbf{g} - \mathbf{g}^*) \tag{4.25}$$

whose initial state is $\mathbf{p}(0) = [(\mathbf{p}^*_l)^T \quad \mathbf{p}^T_f(0)]^T$, where $\mathbf{p}_f(0)$ can be arbitrary.

Instead, considering the case of a moving formation, which means that the

[17] Zhao S., Zhengtao D, Zhenhong Li, Bearing-Only Formation Tracking Control of Multi-Agent Systems, in IEEE Transactions on Automatic Control, 2019, Manchester

[18] ibidem

leader moves with nonzero constant velocity v_l, the control law is:

$$\dot{\mathbf{p}}_i = k_P \sum_{j \in \mathcal{N}_i} (\mathbf{g}_{ij}(t) - \mathbf{g}_{ij}^*) + k_I \int_0^t \sum_{j \in \mathcal{N}_i} (\mathbf{g}_{ij}(\tau) - \mathbf{g}_{ij}^*) d\tau, \qquad i \in \mathcal{V}_f \tag{4.26}$$

where k_P and k_I are positive constant gains.

Given such control law, the formation stability can be proved by the theorem Th.4.3.1.

Theorem 4.3.1 (Single integrator formation tracking) *Under the assumptions that:*

1. *the target formation $(\mathcal{G}, \boldsymbol{p}^*)$ is unique, i.e. $\mathcal{B}_{ff}$ is positive definite;*

2. *No neighboring agents collide with each other during the formation evolvement*

$\boldsymbol{p}(t)$ converges to $\boldsymbol{p}^*$ asymptotically by the action of control law in Eq.4.26, where $\boldsymbol{p}^*(t)$ represents the target configuration moving at velocity v_l with a fixed geometric pattern.

5

Leader-Follower approach using MPC with Bearing

In this project it is implemented the Leader-Follower approach through the MPC controller for an heterogeneous MAS. It is assumed that the distances between the robots are unknown. Hence, the MPC is constructed based on bearing and the velocities.

An heterogeneous MAS composed by four agents is considered. Between them, two are UGVs and the other two are UAVs, such that the agents are two by two homogeneous. The leader of the system is one of the two UAVs, while the other UAV and the two UGVs are the followers. Only the leader knows the trajectory to track while the followers track the same trajectory of the leader without knowing the distance with it but just knowing the velocities and the desired bearing vector.

To obtain the final result, the system is constructed by steps:

1. the UAV leader trajectory is implemented as shown in section 2.1.6;
2. the first UAV follower is added;
3. the two UGVs followers are added. Note that they depend on the leader and there is no relation between themselves;
4. a noise is added to the Leader-Follower approach between the UAV leader and the 2 UGVs followers.

In order to implement the MPC controller, an open source MATMPC toolbox, that is a MATLAB based toolbox for real-time non linear model predictive control, is used.

Using MATMPC, a NLP is formulated by applying direct multiple shooting to an OCP over the prediction horizon $T = [t_0, t_f]$, which is divide in N shooting intervals $[t_0, t_1, ..., t_N]$ as:

$$\begin{aligned} \min_{x_k, u_k} \sum_{k=0}^{N-1} \frac{1}{2}||h_k(x_k, u_k)||^2_W + \frac{1}{2}||h_N(x_N)||^2_{W_N} \\ such \quad that \quad x_0 - \hat{x}_0 = 0 \\ x_{k+1} - \phi_k(x_k, u_k) = 0, \quad k = 0, 1, ..., N-1 \\ \underline{r}_k \leq r_k(x_k, u_k) \leq \bar{r}_k, \quad k = 0, 1, ..., N-1 \\ \underline{r}_N \leq r_N(x_N) \leq \bar{r}_N \end{aligned} \tag{5.1}$$

where $\hat{x}_0$ is the measurement of the current state. The system states $x_k \in \mathbb{R}^{n_x}$ are defined at the discrete time point t_k for $k = 0, ..., N$ and the control inputs $u_k \in \mathbb{R}^{n_u}$ for $k = 0, ..., N-1$ are piecewise constant. It also holds that $r(x_k, u_k) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_r}$ and $r(x_N) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_l}$, with lower and upper bound $\underline{r}_k, \bar{r}_k$. $\phi_k(x_k, u_k)$ is a numerical integration operator that solves the following initial value problem (IVP) and returns the solution at t_{k+1}[1].

$$0 = f(\dot{x}(t), x(t), u(t), t), \qquad x(0) = x_k$$

.

5.1 Leader-Follower approach between the 2 UAVs

In order to construct the Leader-Follower approach between the 2 UAVs, first an MPC controller based just on the bearing vector (composed by three components along the x, y, z axis) and velocities (linear velocities along x, y, z axis and angular velocities around x, y, z axis) is considered. Then, in order to improve the quality of such a controller, a new MPC controller based on the bearing vector, velocities, angles, thrust (T) and torques (τ_ϕ, τ_θ, τ_ψ) is used.

As already explained in section 3.2, the necessary ingredients for a well-defined MPC are essentially three: cost function, dynamic system and constraints.

In addition, also the states and control inputs have to be defined such that

[1]Beghi A., Bruschetta M., Chen Y., Picotti E., *MATMPC - A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control*, 2019 18th European Control Conference (ECC), Napoli, Italy, June 25-28, 2019

the states can be expressed as a function of the control inputs in the dynamic system.

Before starting to solve the problem, the following assumptions are done:

- the two UAVs considered are quadrotors described by the same characteristics given in Tab.2.2, since they are assumed to be homogeneous. Instead, they have different initial positions that are known and shown in Tab.5.1. It has to be observed that the initial position of the UAV follower is chosen paying attention to the fact that this must be different from the one of the leader because, otherwise, the dynamical system cannot be solved.

Parameter	**Value**
x_l	$0[m]$
y_l	$0[m]$
z_l	$1[m]$
x_f	$-0.15[m]$
y_f	$0[m]$
z_f	$1.04[m]$

Table 5.1: *Initial positions of the UAV leader and UAV follower.*

- the UAV leader knows its trajectory, that is the one presented in section 2.1.6.

- the desired bearing vector ($\mathbf{g}^*$) is imposed. In details, it is:

$$\begin{bmatrix} g_x^* \\ g_y^* \\ g_z^* \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.52915 \end{bmatrix}$$

  The values of such vector are chosen so that its Euclidean norm is equal to 1 since, as explained in Chapter 4, it is required that $\mathbf{g}_{ij}$ is a unit vector.

- the UAV follower knows the linear and angular velocities of the UAV leader: $v_{x,l}$, $v_{y,l}$, $v_{z,l}$, $\omega_{x,l}$, $\omega_{y,l}$, $\omega_{z,l}$.

- the UAV follower knows the thrust and torques of the UAV leader: T_l, $\tau_{\phi,l}$, $\tau_{\theta,l}$, $\tau_{\psi,l}$.

The resolution has to be based on a graph since the system considered is a MAS composed by two UAVs. In details, a direct graph, as shown in Fig.5.1, is taken into account, where node 2 represents the follower, node 1 represents the leader and $\mathbf{g}_{21}$ represents the unknown bearing vector from follower to leader (from node 2 to node 1).

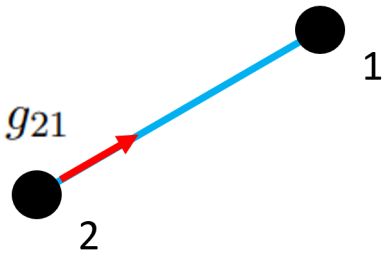


Figure 5.1: *Directed graph considered for the Leader-Follower approach between two UAVs.*

Recalling that the general formulation of the MPC controller is the one expressed in Eq.5.2, in the case of interest the cost function is given by the sum of other four cost functions that have to be minimized as shown in Eq.5.3. These minimisations are related to the bearing error vector, the angle error vector, the motion error vector relative to the velocities and the error vector on thrust and torques.

$$\begin{aligned} \hat{u}^* &= \min_u J \\ such \quad that \quad \dot{x} &= f(x,u) \\ other \quad constraints & \end{aligned} \tag{5.2}$$

$$\begin{aligned} \hat{u}^* = \min_u J &= \min_u J_g + J_m + J_{rpy} + J_{act} \\ such \quad that \quad \dot{\bar{x}} &= f(\bar{x},u) \\ other \quad constraints & \end{aligned} \tag{5.3}$$

In details, the definitions of these errors are:

$$\mathbf{e}_g = \mathbf{g}^* - \mathbf{g}_{21} = \begin{bmatrix} g_x^* \\ g_y^* \\ g_z^* \end{bmatrix} - \begin{bmatrix} g_{21,x} \\ g_{21,y} \\ g_{21,z} \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{5.4}$$

$$\mathbf{e}_m = \mathbf{m}_l - \mathbf{m}_f = \begin{bmatrix} v_{x,l} \\ v_{y,l} \\ v_{z,l} \\ \omega_{x,l} \\ \omega_{y,l} \\ \omega_{z,l} \end{bmatrix} - \begin{bmatrix} v_{x,f} \\ v_{y,f} \\ v_{z,f} \\ \omega_{x,f} \\ \omega_{y,f} \\ \omega_{z,f} \end{bmatrix} \in \mathbb{R}^{6\times 1} \tag{5.5}$$

$$\mathbf{e}_{rpy} = \mathbf{rpy}_l - \mathbf{rpy}_f = \begin{bmatrix} \phi_l \\ \theta_l \\ \psi_l \end{bmatrix} - \begin{bmatrix} \phi_f \\ \theta_f \\ \psi_f \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{5.6}$$

$$\mathbf{e}_{act} = \mathbf{act}_l - \mathbf{act}_f = \begin{bmatrix} \tau_{\phi,l} \\ \tau_{\theta,l} \\ \tau_{\psi,l} \\ T_l \end{bmatrix} - \begin{bmatrix} \tau_{\phi,f} \\ \tau_{\theta,f} \\ \tau_{\psi,f} \\ T_f \end{bmatrix} \in \mathbb{R}^{4\times 1} \tag{5.7}$$

Considering this formulation, the states are the three components of the bearing vector ($\mathbf{g}_{21}$), the linear and angular velocities of the follower and the roll (ϕ), pitch (θ) and yaw (ψ) angles of the follower. Instead, the control inputs are the thrust and torques of the follower.

Then, the states and the control inputs have, as time-varying references, the desired bearing vector, the linear and angular velocities of the leader, the angles of the leader and the thrust and torques of the leader, respectively.

So, following the order of the sum as in Eq.5.3, the cost function J is defined as in Eq.5.8:

$$J = \mathbf{e}_g^T \mathbf{C} \mathbf{e}_g + \mathbf{e}_m^T \mathbf{D} \mathbf{e}_m + \mathbf{e}_{rpy}^T \mathbf{E} \mathbf{e}_{rpy} + \mathbf{e}_{act}^T \mathbf{F} \mathbf{e}_{act} \tag{5.8}$$

where $\mathbf{C} \in \mathbb{R}^{3\times 3}$, $\mathbf{D} \in \mathbb{R}^{6\times 6}$, $\mathbf{E} \in \mathbb{R}^{3\times 3}$, $\mathbf{F} \in \mathbb{R}^{4\times 4}$ are tuning gain matrices (also called *weighting matrices*).

Having defined the cost function that has to be minimized, a dynamic system relating the states with the control inputs must be defined. So, an augmented system is created and, mathematically, it holds that the vector of unknowns

related to the states is as in Eq. 5.9.

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{g}_{21} \\ \mathbf{v} \\ \mathbf{rpy} \end{bmatrix} = \begin{bmatrix} g_{21x} \\ g_{21y} \\ g_{21z} \\ v_{xf} \\ v_{yf} \\ v_{zf} \\ w_{xf} \\ w_{yf} \\ w_{zf} \\ \phi_f \\ \theta_f \\ \psi_f \end{bmatrix} \in \mathbb{R}^{12\times 1} \tag{5.9}$$

since

$$\mathbf{g}_{21} = \begin{bmatrix} g_{21x} \\ g_{21y} \\ g_{21z} \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{5.10}$$

$$\mathbf{v} = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{r\dot{p}y} \end{bmatrix} = \begin{bmatrix} v_{xf} \\ v_{yf} \\ v_{zf} \\ w_{xf} \\ w_{yf} \\ w_{zf} \end{bmatrix} \in \mathbb{R}^{6\times 1} \tag{5.11}$$

$$\mathbf{rpy} = \begin{bmatrix} \phi_f \\ \theta_f \\ \psi_f \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{5.12}$$

$$\tag{5.13}$$

Instead, the unknown control inputs vector is defined as in Eq.5.14.

$$\mathbf{u} = \begin{bmatrix} \tau_{\phi,f} \\ \tau_{\theta,f} \\ \tau_{\psi,f} \\ T_f \end{bmatrix} \in \mathbb{R}^{4\times 1} \tag{5.14}$$

Hence, the dynamic system has to be of the form as in Eq.5.15.

$$\dot{\bar{\mathbf{x}}} = \mathbf{B}\bar{\mathbf{x}} + \mathbf{A}\mathbf{u} \tag{5.15}$$

To construct the matrices **A** and **B**, the dynamics of the bearing vector in Eq.5.16 and the ones of the UAV reported in Eq.5.24, 5.25, 5.26 must be recalled.

$$\dot{\mathbf{g}} = \mathcal{B}^W \mathbf{v} \tag{5.16}$$

where:

- **v** is the ordered vector containing the linear velocities of the leader, the linear velocities of the follower, the angular velocities of the leader and the angular velocities of the follower, formally defined as in Eq.5.17.

$$\mathbf{v} = \begin{bmatrix} v_{x,l} \\ v_{y,l} \\ v_{z,l} \\ v_{x,f} \\ v_{y,f} \\ v_{z,f} \\ \omega_{x,l} \\ \omega_{y,l} \\ \omega_{z,l} \\ \omega_{x,f} \\ \omega_{y,f} \\ \omega_{z,f} \end{bmatrix} \in \mathbb{R}^{12\times 1} \tag{5.17}$$

- $\mathcal{B}^W$ is the Bearing Laplacian Matrix defined in the World Frame and mathematically expressed as in Eq.5.18.

$$\mathcal{B}^W = \begin{bmatrix} -\mathbf{0}- & -\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}} & -\mathbf{0}- & \frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}} & -\mathbf{0}- & \dots & -\mathcal{S}g_{ij} & -\mathbf{0}- \end{bmatrix} \tag{5.18}$$

  where:

  - $-\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}$ is in position i,
  - $\frac{\mathbf{P}_{ij}\mathbf{R}_i^T}{d_{ij}}$ is in position j
  - $-\mathbf{S}\mathbf{g}_{ij}$ is in position $3N + i$.

In details:

$$d_{ij} = ||\mathbf{p}_i - \mathbf{p}_j|| \Rightarrow d_{21} = ||\mathbf{p}_2 - \mathbf{p}_1|| \tag{5.19}$$

$$\mathbf{g}_{ij} = \mathbf{R}_i^T \frac{\mathbf{p}_j - \mathbf{p}_i}{d_{ij}} \Rightarrow \mathbf{g}_{21} = \mathbf{R}_2^T \frac{\mathbf{p}_1 - \mathbf{p}_2}{d_{21}} \in \mathbb{R}^{3\times 1} \tag{5.20}$$

$$\mathbf{P}_{ij} = \mathbf{I}_3 - \mathbf{g}_{ij}\mathbf{g}_{ij}^T \Rightarrow \mathbf{P}_{21} = \mathbf{I}_3 - \mathbf{g}_{21}\mathbf{g}_{21}^T \in \mathbb{R}^{3\times 3} \tag{5.21}$$

$$\mathbf{R}_i(\psi_i) = \mathbf{R}_2(\psi_2) \in \mathbb{R}^{3\times 3} \tag{5.22}$$

$$\mathcal{B} \in \mathbb{R}^{3\times 8} \tag{5.23}$$

$$\frac{d}{dt}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \mathbf{R}_{WB}\frac{T}{m}\begin{bmatrix}0\\0\\1\end{bmatrix} - \begin{bmatrix}0\\0\\g\end{bmatrix} \tag{5.24}$$

where:

- T is the thrust component;
- m is the mass of the quadrotor;
- g is the gravity acceleration, i.e. $g = 9.81[m/s^2]$
- $\mathbf{R}_{WB}$ is the rotation matrix from Body to World Reference Frame, defined as:

$$\mathbf{R}_{WB}(\phi,\theta,\psi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

  where $c = cos(\cdot)$ and $s = sen(\cdot)$.

$$\begin{bmatrix}\dot{\omega}_x\\ \dot{\omega}_y\\ \dot{\omega}_z\end{bmatrix} = \mathbf{I}^{-1}\left(\begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} - \begin{bmatrix}0 & -\omega_z & \omega_x\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix}\mathbf{I}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}\right) \tag{5.25}$$

where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the inertial matrix.

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}c_\theta c_\psi & s_\psi & 0\\ -c_\theta s_\psi & c_\psi & 0\\ s_\theta & 0 & 1\end{bmatrix}^{-1}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \tag{5.26}$$

Hence, matrix **A** and **B** result to be constructed as given in Eq.5.27 and

Eq.5.28.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} \in \mathbb{R}^{12\times4} \tag{5.27}$$

with

$$\mathbf{A}_1 = \frac{\mathbf{R}_{WB,f}}{m} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^{3\times1}$$

$$\mathbf{A}_2 = \mathbf{I}^{-1} \in \mathbb{R}^{3\times3}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathcal{B} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times12} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_2 & \mathbf{0}_{3\times3} \end{bmatrix} \in \mathbb{R}^{12\times18} \tag{5.28}$$

with

$$\mathcal{B} = \begin{bmatrix} \frac{\mathbf{P}_{21}\mathbf{R}_2^T}{d_{21}} & -\frac{\mathbf{P}_{21}\mathbf{R}_2^T}{d_{21}} & \mathbf{0}_{3\times5} & -\mathbf{S}\mathbf{g}_{21} \end{bmatrix} \in \mathbb{R}^{3\times12}, \qquad with \quad \mathbf{S} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_1 = -\mathbf{I}^{-1} \begin{bmatrix} 0 & -\omega_{zf} & \omega_{yf} \\ \omega_{zf} & 0 & -\omega_{xf} \\ -\omega_{yf} & \omega_{xf} & 0 \end{bmatrix} I \in \mathbb{R}^{3\times3}$$

$$\mathbf{B}_2 = \begin{bmatrix} c_\theta c_\psi & s_\psi & 0 \\ -c_\theta s_\psi & c_\psi & 0 \\ s_\theta & 0 & 1 \end{bmatrix}^{-1} \in \mathbb{R}^{3\times3}$$

Combining these results, the dynamics of the system can be summarized in

Eq.5.29.

$$\begin{bmatrix} \dot{g}_{21x} \\ \dot{g}_{21y} \\ \dot{g}_{21z} \\ \dot{v}_{xf} \\ \dot{v}_{yf} \\ \dot{v}_{zf} \\ \dot{w}_{xf} \\ \dot{w}_{yf} \\ \dot{w}_{zf} \\ \dot{\phi}_f \\ \dot{\theta}_f \\ \dot{\psi}_f \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathcal{B} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times12} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_2 & \mathbf{0}_{3\times3} \end{bmatrix} \begin{bmatrix} g_{21x} \\ g_{21y} \\ g_{21z} \\ v_{xl} \\ v_{yl} \\ v_{zl} \\ v_{xf} \\ v_{yf} \\ v_{zf} \\ w_{xl} \\ w_{yl} \\ w_{zl} \\ w_{xf} \\ w_{yf} \\ w_{zf} \\ \phi_f \\ \theta_f \\ \psi_f \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} \begin{bmatrix} \tau_{\phi,f} \\ \tau_{\theta,f} \\ \tau_{\psi,f} \\ T_f \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.29}$$

About the constraints, a constraint for each control input ($\tau_{\phi,f}$, $\tau_{\theta,f}$, $\tau_{\psi,f}$ and T_f) is imposed. In particular the chosen upper and lower bounds for these quantities are:

$$-1[Nm] \leq \tau_{\phi,f} \leq 1[Nm] \tag{5.30}$$
$$-1[Nm] \leq \tau_{\theta,f} \leq 1[Nm] \tag{5.31}$$
$$-1[Nm] \leq \tau_{\psi,f} \leq 1[Nm] \tag{5.32}$$
$$6[N] \leq T_f \leq 35[N] \tag{5.33}$$
$$\tag{5.34}$$

The choice of them is based on the observed values of τ_ϕ, τ_θ, τ_ψ and T of the leader. Instead, no constraints are established on the states.

Summarizing the MPC problem that has to be solved is the one presented in

Eq.5.35.

$$
\begin{aligned}
&\min \sum_{k=0}^{N-1} \frac{1}{2}\left(\mathbf{e}_g^T\mathbf{C}\mathbf{e}_g + \mathbf{e}_m^T\mathbf{D}\mathbf{e}_m + \mathbf{e}_{rpy}^T\mathbf{E}\mathbf{e}_{rpy} + \mathbf{e}_{act}^T\mathbf{F}\mathbf{e}_{act}\right) + \frac{1}{2}\left(\mathbf{e}_g^T\mathbf{G}\mathbf{e}_g + \mathbf{e}_m^T\mathbf{H}\mathbf{e}_m + \mathbf{e}_{rpy}^T\mathbf{I}\mathbf{e}_{rpy}\right) \\
&\quad such \quad that \quad \begin{bmatrix} \dot{g}_{21x} \\ \dot{g}_{21y} \\ \dot{g}_{21z} \\ \dot{v}_{xf} \\ \dot{v}_{yf} \\ \dot{v}_{zf} \\ \dot{w}_{xf} \\ \dot{w}_{yf} \\ \dot{w}_{zf} \\ \dot{\phi}_f \\ \dot{\theta}_f \\ \dot{\psi}_f \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathcal{B} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times12} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times12} & \mathbf{B}_2 & \mathbf{0}_{3\times3} \end{bmatrix} \begin{bmatrix} g_{21x} \\ g_{21y} \\ g_{21z} \\ v_{xl} \\ v_{yl} \\ v_{zl} \\ v_{xf} \\ v_{yf} \\ v_{zf} \\ w_{xl} \\ w_{yl} \\ w_{zl} \\ w_{xf} \\ w_{yf} \\ w_{zf} \\ \phi_f \\ \theta_f \\ \psi_f \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} \begin{bmatrix} \tau_{\phi,f} \\ \tau_{\theta,f} \\ \tau_{\psi,f} \\ T_f \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&\qquad -1[Nm] \leq \tau_{\phi,f} \leq 1[Nm] \\
&\qquad -1[Nm] \leq \tau_{\theta,f} \leq 1[Nm] \\
&\qquad -1[Nm] \leq \tau_{\psi,f} \leq 1[Nm] \\
&\qquad 6[N] \leq T_f \leq 35[N]
\end{aligned}
\tag{5.35}
$$

After having defined the dynamical system, from the implementation point of view, other parameters have to be chosen. Between them, it has been chosen to set:

- the number of shooting intervals (N) equal to 80. This number represents the prediction horizon that is used to minimize the error between the measurement and the reference.
- The shooting interval time T_s is set to 0.001 s. This is chosen in such a way that the model can be solved. In fact, if it is too big, some errors arise due

to the fact that the prediction cannot be performed.

- the algorithm used to solve the non-linear least squares problems is the Gauss-Newton. This method is an extension of the Newton algorithm used to find a minimum of a non-linear function.

- the chosen integrator is $ERK4$ ($Ode4$ is used in MATLAB with fixed step size equal to the shooting interval time) which uses the Runge-Kutta algorithm to solve Ordinary Differential Equations;

- the solver is the $qpoases$;

- the weighting matrices **C**, **D**, **E** and **F** are diagonal matrices chosen by trial and error. These matrices are used to weight the contribution of each term to the cost index itself. The greater the weight, the smaller the deviation from the tolerated reference signal. The choice of these weights is not immediate since it is the result of an iterative procedure that starts from the choice of the initial matrices and then proceeding by adjusting the starting values depending on the optimal solutions obtained. Note that varying the values of the weights influences the dynamics of the signals in terms of shape and amplitude and makes it possible to model the accuracy of the tracking of the reference in such a way as to achieve the objective. In particular, the results shown in the following are obtained considering:

$$\mathbf{C} = \begin{bmatrix} 1050 & 0 & 0 \\ 0 & 1050 & 0 \\ 0 & 0 & 1050 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 16500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 700 & 0 & 0 \\ 0 & 0 & 0 & 0 & 700 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18250 \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 18250 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 1050 & 0 & 0 \\ 0 & 1050 & 0 \\ 0 & 0 & 1050 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 16500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 21500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 700 & 0 & 0 \\ 0 & 0 & 0 & 0 & 700 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18250 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 18250 \end{bmatrix}$$

Implementing the MPC controller as explained above, the final UAV follower trajectory is the one shown in Fig.5.2, while the comparison between the UAV leader and UAV follower trajectories is shown in Fig.5.3.

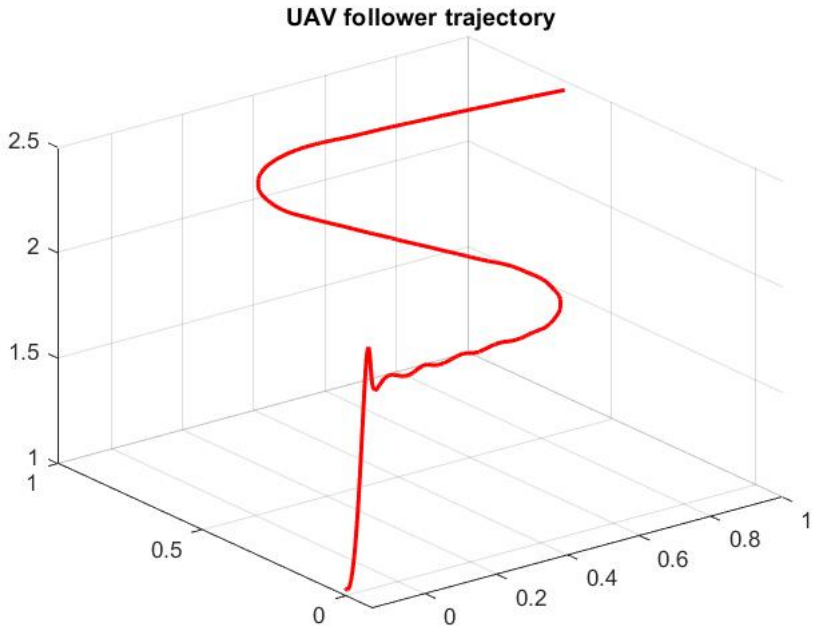


Figure 5.2: *UAV follower trajectory.*

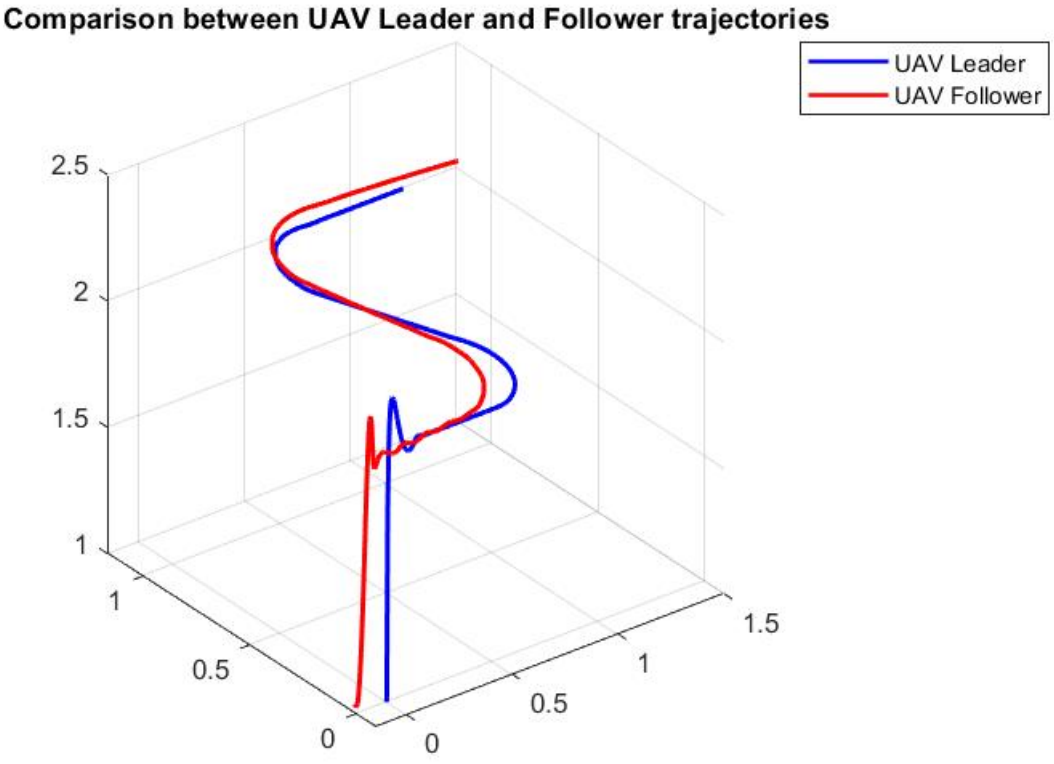


Figure 5.3: *Comparison between the UAV leader and the UAV follower trajectories.*

Looking at Fig.5.3, the UAV follower trajectory follows the UAV leader one

even if they starts from two different initial positions as given in Tab.5.1. However, it must be noted that the UAV leader stabilizes at an altitude equal to 2 m, while the UAV follower reaches an altitude of almost 2.5 m. This is due to the fact that the linear velocity along the z direction stabilizes at a really small positive value different from 0, as it is possible to observe in Fig.5.8(c). In addition, the UAV follower oscillates more along the z coordinate with respect to the UAV leader.

The behaviour of the obtained result can be studied more through the analysis of the thrust and torques, other than linear, angular velocities and angles. The control inputs, given by the MPC, assume the form shown in Fig.5.4, which are also compared to the ones of the leader as shown in Fig.5.5 and 5.6.

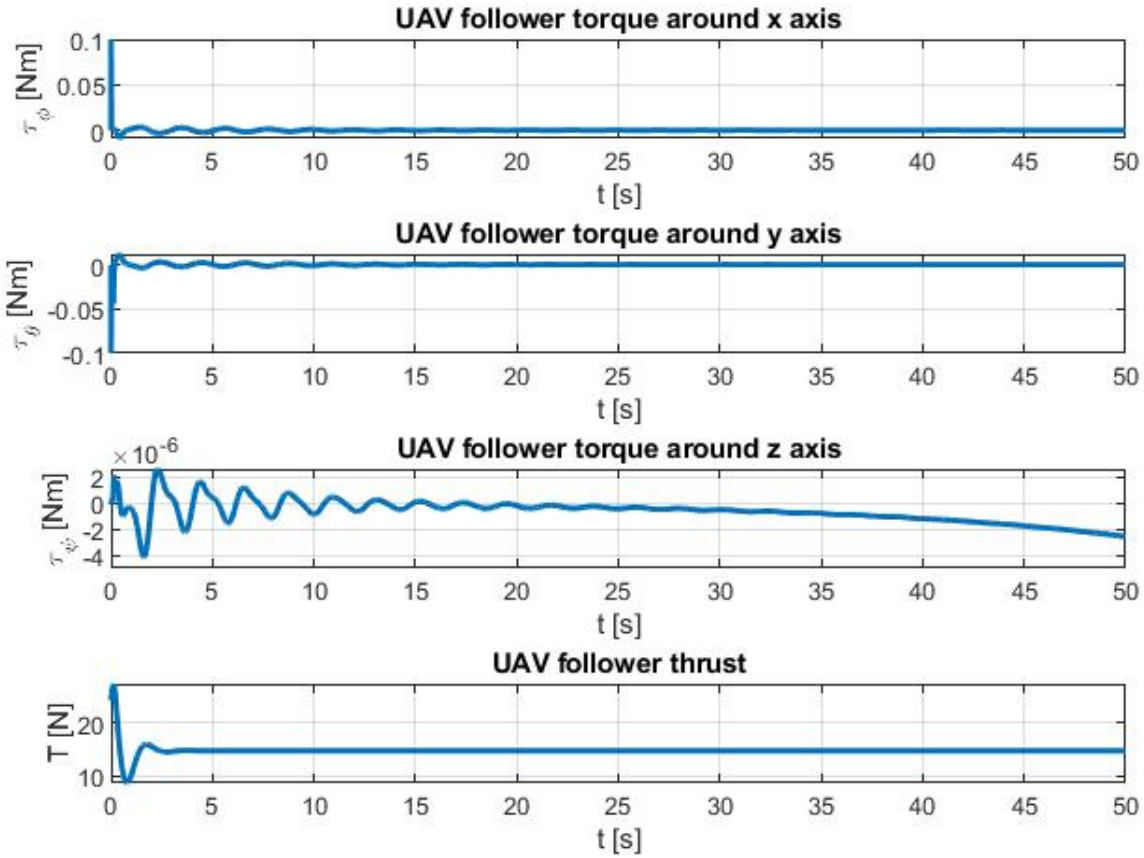
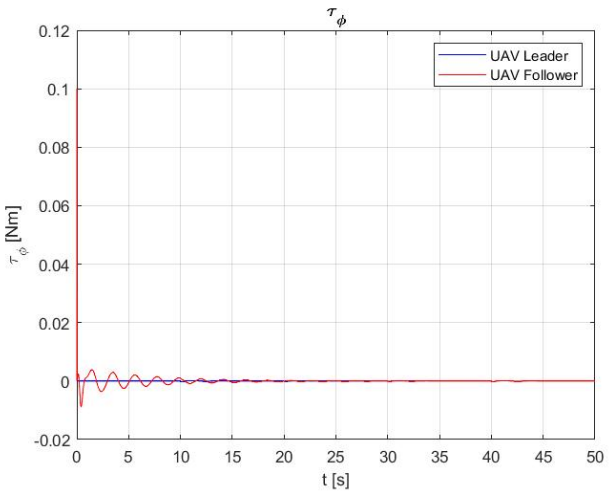
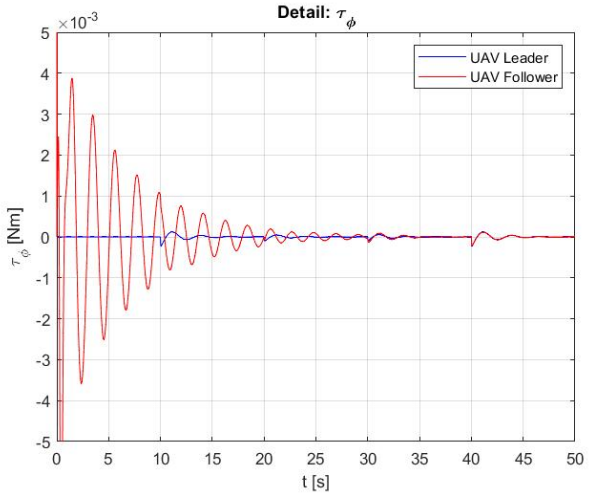


Figure 5.4: *UAV follower thrust and torques.*

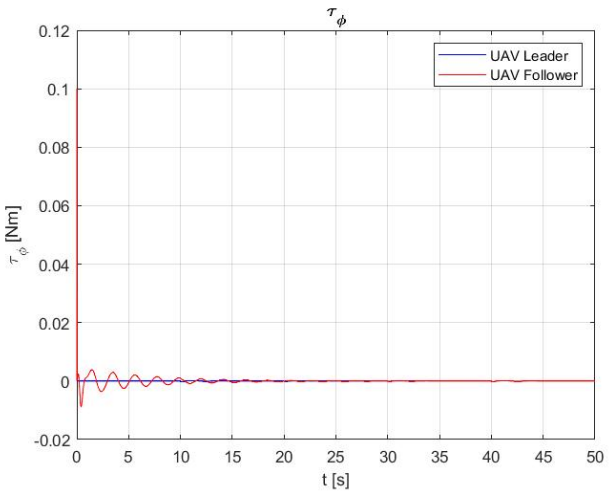
Looking at Fig.5.5, the UAV follower's torques oscillate more than the ones relative to the UAV leader even if the first ones reach the values of the ones of the leader in almost 30 s. According to the trajectory, every time the trajectory changes, i.e. every 10 s, an effect in the torques of both the agents is visible due



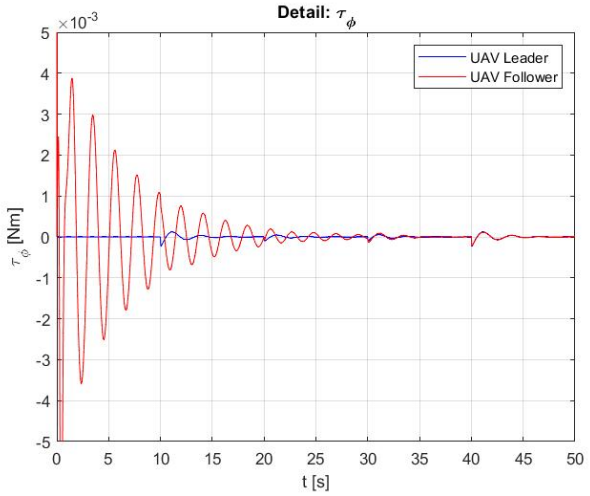
(a) τ_ϕ *UAVs leader and follower.*



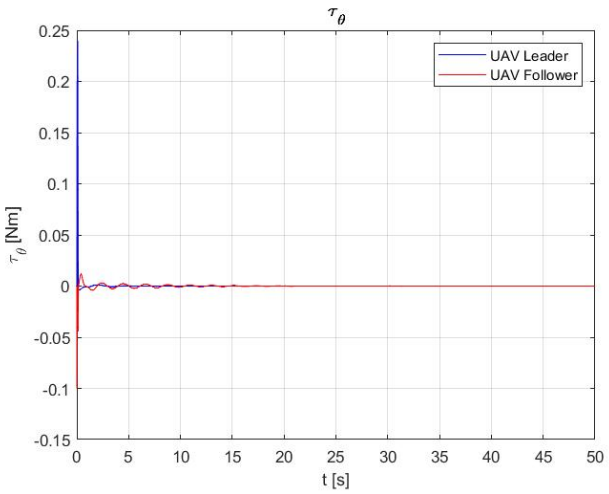
(b) *Detail:* τ_ϕ *UAVs leader and follower.*



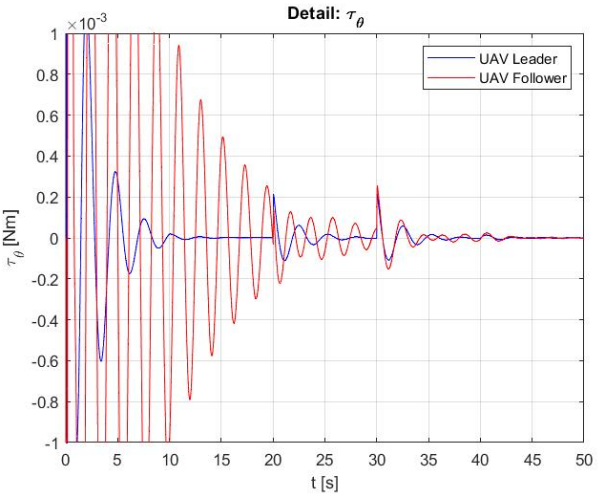
(c) τ_θ *UAVs leader and follower.*



(d) *Detail:* τ_θ *UAVs leader and follower.*

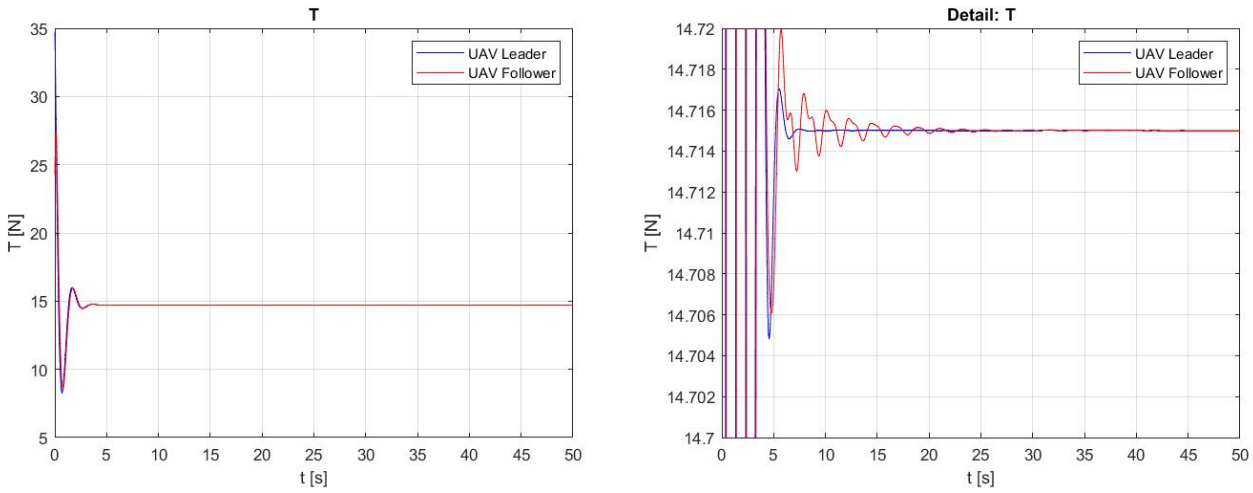


(e) τ_ψ *UAVs leader and follower.*



(f) *Detail:* τ_ψ *UAVs leader and follower.*

Figure 5.5: *UAV leader and UAV follower torques.*



(a) *T UAVs leader and follower.* (b) *Detail: T UAVs leader and follower.*

Figure 5.6: *UAV leader and UAV follower thrust.*

to the fact that the UAVs have to align with the new direction to follow.

Instead, looking at Fig.5.6, the thrusts of the two UAVs start from a value that is different from the force of gravity given by $mg = 14.715\ N$. This behaviour is due to the fact both the UAV leader and UAV follower start from $z = 1\ m$, but they have to reach the altitude of $z = 2\ m$. Hence, they must apply an upward force greater than the force of gravity because if these two forces were equal, the quadrotors would maintain a constant height of $1\ m$. In fact, after having reached the desired altitude, the UAVs leader and follower stabilize, equalising the thrust with the force of gravity.

So, knowing the values of $\tau_{\phi,f}$, $\tau_{\theta,f}$, $\tau_{\psi,f}$ and T_f, the rotor spinning rates squared $\begin{bmatrix} \omega_{1,f}^2 \\ \omega_{2,f}^2 \\ \omega_{3,f}^2 \\ \omega_{4,f}^2 \end{bmatrix}$ are given by the application of the wrench mapper (defined in Tab.2.2) to the vector $\begin{bmatrix} \tau_{\phi,f} \\ \tau_{\theta,f} \\ \tau_{\psi,f} \\ \mathrm{T}_f \end{bmatrix}$. Hence, from the rotor spinning rates squared, the linear and angular accelerations are computed and then, integrating them, the linear and angular velocities are obtained.

The linear velocities' behaviour are shown in Fig.5.8, while a comparison between the UAV leader and UAV follower linear velocities are presented in Fig.5.8.

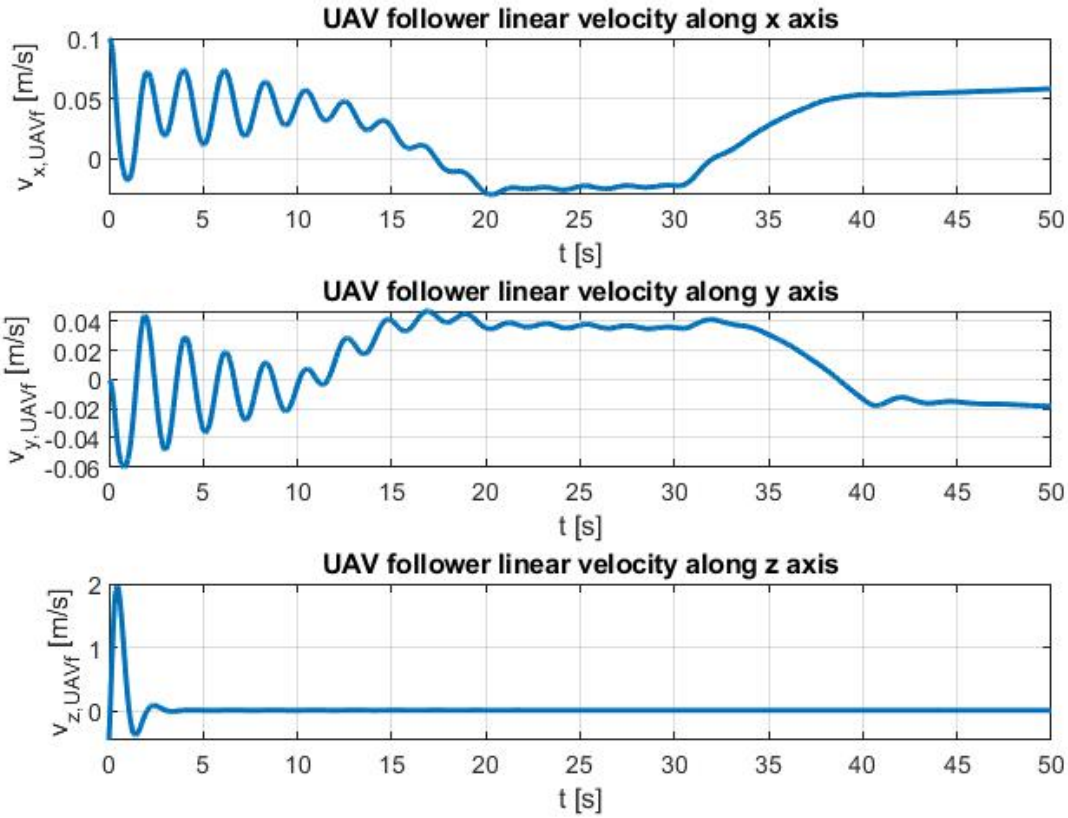
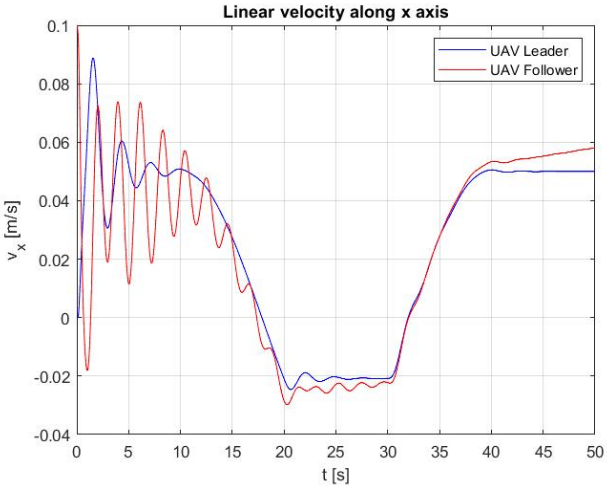


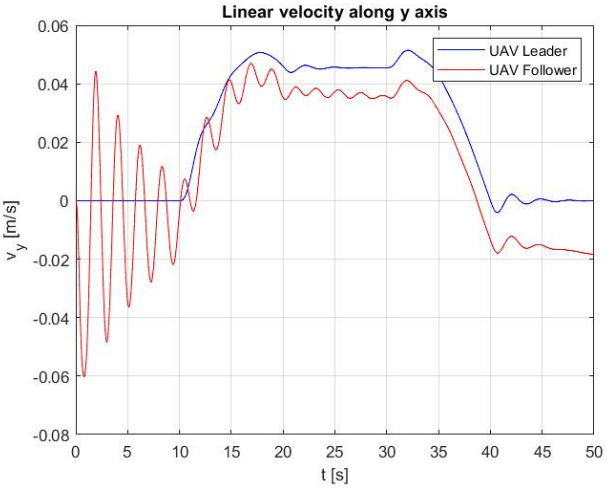
Figure 5.7: *UAV follower linear velocities.*

Looking at Fig.5.8, the linear velocities of the UAV follower almost follow the ones of the leader even if with a more oscillatory behaviour, especially in the first 20 s, due to the fact that the starting positions of the two agents are different and the linear velocities of the follower are initialized to $\begin{bmatrix} 0.1 \\ 0 \\ -0.46 \end{bmatrix} [m/s]$, while the ones of the leader to $\begin{bmatrix} 0.1 \\ 0 \\ 0 \end{bmatrix} [m/s]$.

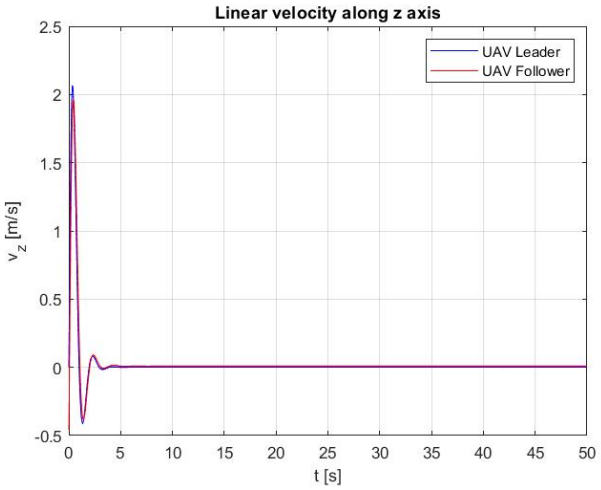
In particular, considering the trajectory that the UAV has to track, $v_x \neq 0$ and $v_y = 0$ from 0 to 10s according to the fact that the drone moves only along the x direction. Then, from 10 to 20s, v_x decreases, while v_y increases since the curve is on the left. From 20 to 30s, v_x and v_y stabilize while, from 30 to 40s, v_x increases and v_y decreases in order to have a curve on the right. In the end, as seen for the first 10s of the simulation, from 40 to 50s, v_x remains constant to a value different from 0, while v_y is almost null since the UAV follower performs a straight line path along the x direction.



(a) v_x *UAVs leader and follower.*



(b) v_y *UAVs leader and follower.*



(c) v_z *UAVs leader and follower.*

Figure 5.8: *UAV leader and UAV follower linear velocities.*

Instead, the angular velocities behaviour is shown in Fig.5.9, while a comparison between the UAV leader and UAV follower angular velocities are presented in Fig.5.10.

Looking at Fig.5.10, as for the case of the linear velocities, it is possible to note the oscillatory behaviour during the first 20 s of the simulation. Then, the angular velocities of the UAV follower along the x and y direction reaches the values of the UAV leader's angular velocities. Instead, about the angular velocity around the z component of the UAV follower, it diverges. To avoid this problem, two possible solutions are:

1. the weights relative to the torques and to the angular velocities have to be changed, accordingly with Eq.5.25;

2. a constraint on the state relative to the angular velocity around the z

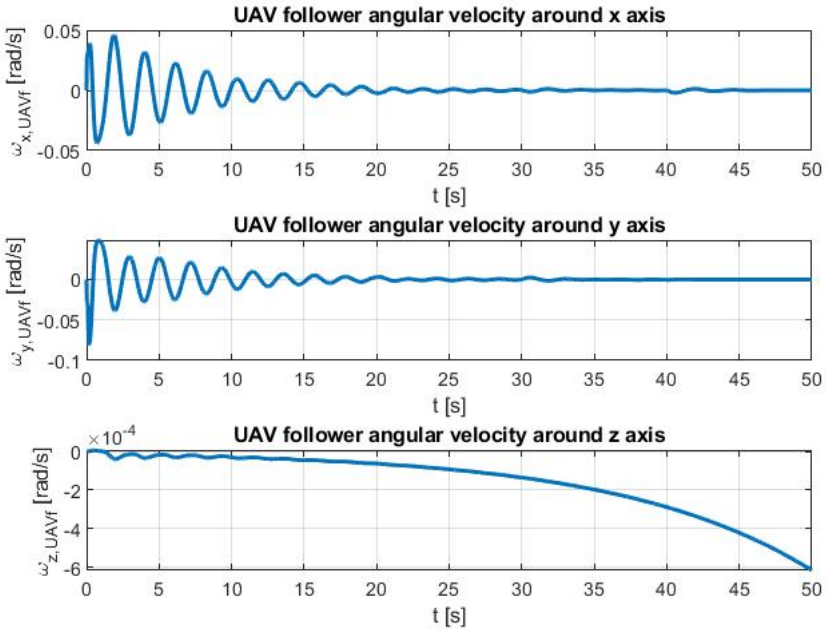
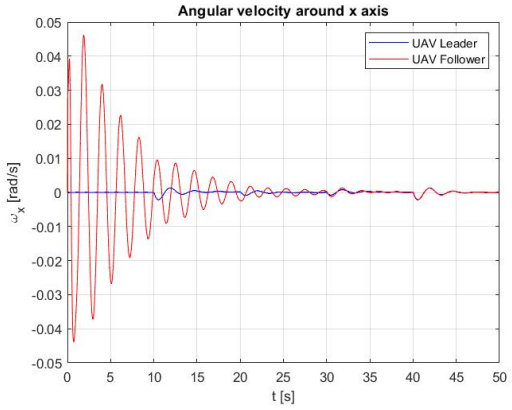
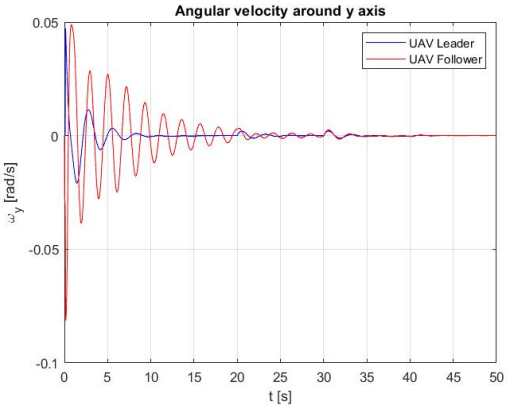


Figure 5.9: *UAV follower angular velocities.*

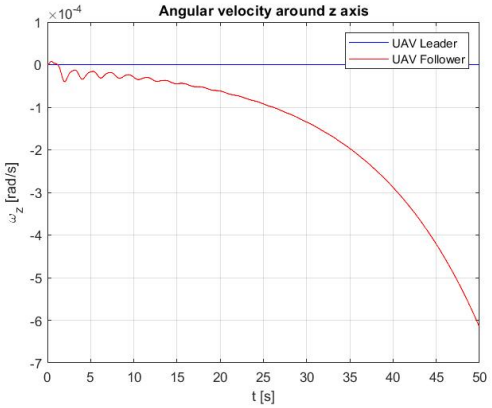
component has to be added.



(a) ω_x *UAVs leader and follower.*



(b) ω_y *UAVs leader and follower.*



(c) ω_z *UAVs leader and follower.*

Figure 5.10: *UAV leader and UAV follower angular velocities.*

In the end, the x, y and z position and the ϕ, θ, ψ angles are given by integrating the linear and angular velocities, respectively. The UAV follower position is shown in Fig.5.11, while the angles in Fig.5.12.

Hence about the angles, considering the Cartesian plane, from 0 to 10s the UAV follower has to track a straight line path how it is possible to note from the fact that ϕ is null and θ oscillates around zero since $\theta_0 = 0\ rad$. Then, from 10 to 20s a curve on the left is performed accordingly with the fact that $\phi < 0$ rad. Then, from 20 to 30s, there is again a straight line path and the angles try to reach a constant value, while from 30 to 40s a curve on the right is performed accordingly with $\phi > 0\ rad$. In the end, at 40s the UAV follower repositions itself to follow a straight line up to 50s, when the simulation ends. Summarizing, about the ϕ angle, it holds that:

- if $\phi < 0\ rad$, then the UAV follower performs a left turn;
- if $\phi > 0\ rad$, then the UAV follower performs a right turn.

In addition, the value of θ is always small but never equal to zero since the

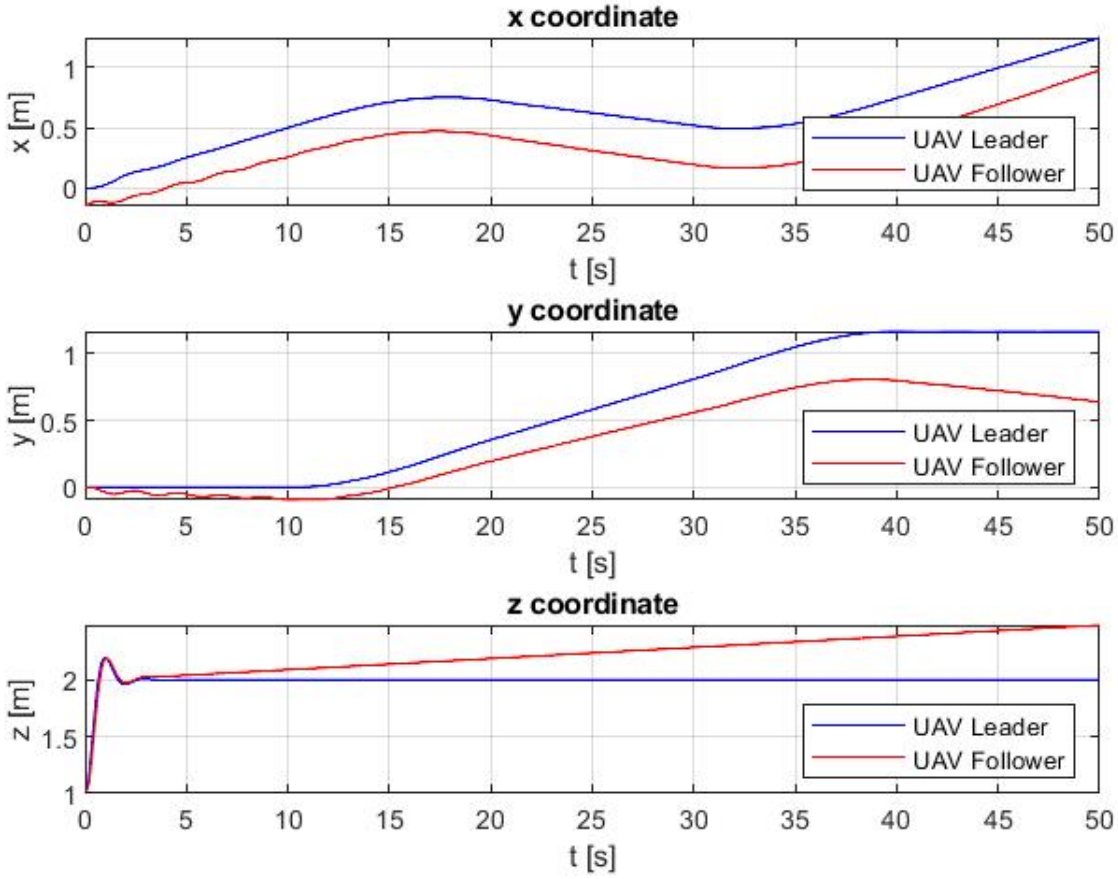


Figure 5.11: *UAV leader and UAV follower trajectories in x, y and z directions.*

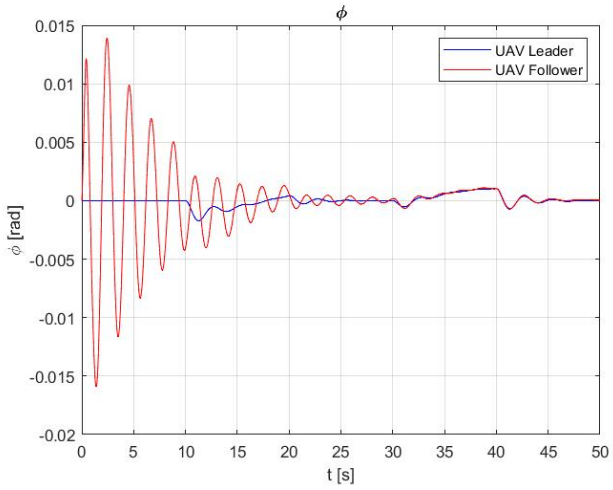
UAV follower must follow the trajectory. In fact, it holds that:

- if $\theta < 0\ rad$, then the UAV follower goes forward;
- if $\theta > 0\ rad$, then the UAV follower goes backward.

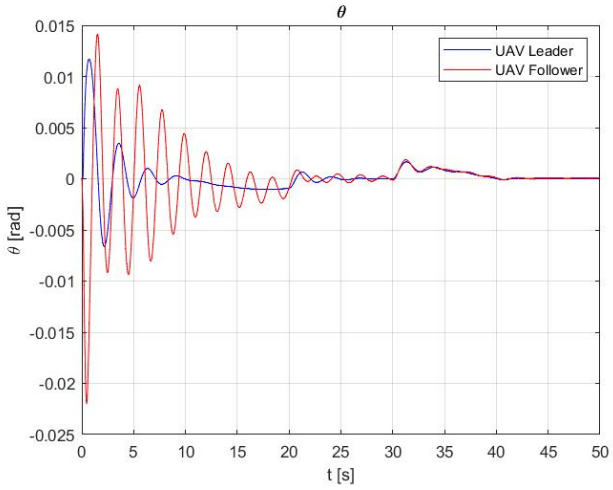
Despite this, when there is a change in the manoeuvre, the angles oscillates since the UAV follower has to change its orientation.

Instead, looking at the plot relative to ψ, for the UAV leader, it is really small (oscillates around zero with an amplitude of order 10^{-16}), while the one of the UAV follower degenerate accordingly with the fact that it is computed by deriving the angular velocity around the z direction that degenerates, too.

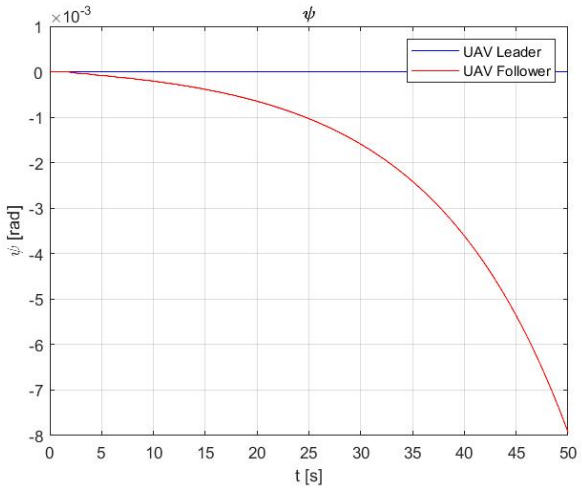
All the results strongly depend on the weights chosen. Such values are chosen by a trial and error approach, even if it possible to observe some rules about this choice. In fact, choosing bigger values for the controls than the ones for the states, the results saturate. Hence, smaller weights are chosen than the



(a) ϕ *UAVs leader and follower.*



(b) θ *UAVs leader and follower.*



(c) ψ *UAVs leader and follower.*

Figure 5.12: *UAV follower angles.*

ones of the states, even if the greater the weight is, the smaller the deviation from the tolerated reference signal is.

Then, also a dependence between the weights can be found. In fact, the weights that adjust the angular acceleration, influence the behaviour of the angular velocities, too. In this way, in order to improve the performance of the system, the weights have to be changed. As an example, choosing the weighting matrices as:

$$\mathbf{C} = \begin{bmatrix} 1050 & 0 & 0 \\ 0 & 1050 & 0 \\ 0 & 0 & 1050 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 16500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 31000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 900000 \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 31000 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 30500 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 1050 & 0 & 0 \\ 0 & 1050 & 0 \\ 0 & 0 & 1050 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 16500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 21500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 700 & 0 & 0 \\ 0 & 0 & 0 & 0 & 700 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18250 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} 700 & 0 & 0 \\ 0 & 700 & 0 \\ 0 & 0 & 18250 \end{bmatrix}$$

the UAV follower stabilizes at an altitude of 2 m, ω_z and ψ do not degenerate anymore. However, a more oscillatory behaviour in the (x, y)-plane appears. Hence, in order to improve the performance of the system, a better weights combination must be found.

5.2 Leader-Follower approach between a UAV and a UGV

To implement the Leader-Follower approach for the heterogeneous MAS composed by a UGV and a UAV, the following assumptions are considered:

- the UGV follower is assumed to be a UAV. Hence, the reasoning is performed in the 3-dimensional space neglecting the behaviour of thrust and torques and imposing the z position component equal to 0 m, since the UGV moves in the 2-dimensional space.

- the UAV is the leader and the UGV is the follower. The UAV characteristics are described in Tab.2.2, while the UGV ones are in Tab.2.1. However, the initial positions of the UAV leader and of the UGV follower are given in Tab.5.2.

Parameter	**Value**
x_l	$0\ [m]$
y_l	$0\ [m]$
z_l	$1\ [m]$
$x_{f,UGV1}$	$-0.3\ [m]$
$y_{f,UGV1}$	$0\ [m]$
$z_{f,UGV1}$	$1\ [m]$

Table 5.2: *Initial positions of the UAV leader and UGV1 follower.*

- the UAV leader knows its trajectory, that is the one shown in section 2.1.6.

- the desired bearing vector ($\mathbf{g}^*$) is known and it is:

$$\begin{bmatrix} g_x^* \\ g_y^* \\ g_z^* \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.52 \end{bmatrix}$$

- the UGV follower knows the UAV leader velocities: $v_{x,l}, v_{y,l}, v_{z,l}, \omega_{z,l}$.

As done in the previous case for the 2 UAVs, it is dealing again with a MAS even if, in this case, it is heterogeneous. Hence, it must be defined a graph to describe it.

As before, for the case of two UAVs, a directed graph is considered and its representation is in Fig.5.13, where it is assumed that node 1 is the leader, node 2 is the follower and $\mathbf{g}_{21}$ is the bearing vector from follower to leader.

An MPC controller for the Leader-Follower approach has to be constructed in order to have that the UGV follows the UAV leader. This controller is constructed based on the bearing vector and on the velocities. As explained before, to construct the MPC, it is necessary to define the cost function, the dynamic system and, eventually, the constraints.

About the cost function that has to be minimized, it is defined as in Eq.5.38.

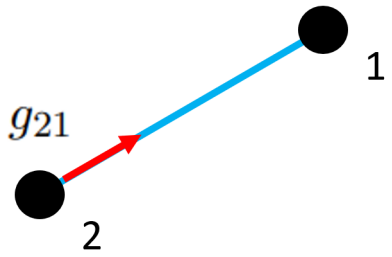


Figure 5.13: *Directed graph considered for the Leader-Follower approach between a UAV and a UGV.*

$$\hat{u}^* = \min_u J = \min_u J_g + J_m \tag{5.36}$$

$$such \quad that \quad \dot{x} = f(x,u) \tag{5.37}$$

$$other \quad constraints \tag{5.38}$$

In this scenario, the states are represented by the bearing vector ($\mathbf{g}_{21}$), while the control inputs are the linear velocities along the x,y and z direction and the angular velocity around the z axis ($\dot{\phi}$). Instead, the time-varying references are the desired bearing vector ($\mathbf{g}^*$) and the linear and angular velocities of the leader. Mathematically, the definitions of such errors are:

$$\mathbf{e}_g = \mathbf{g}^* - \mathbf{g} = \begin{bmatrix} g^*_{21,x} \\ g^*_{21,y} \\ g^*_{21,z} \end{bmatrix} - \begin{bmatrix} g_{21,x} \\ g_{21,y} \\ g_{21,z} \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{5.39}$$

$$\mathbf{e}_m = \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \\ \dot{z}_l \\ \dot{\phi}_l \end{bmatrix} - \begin{bmatrix} \dot{x}_f \\ y_f \\ \dot{z}_f \\ \dot{\phi}_f \end{bmatrix} \in \mathbb{R}^{4\times 1} \tag{5.40}$$

Hence, following the order in Eq.5.38, the cost function becomes as in Eq.5.41.

$$J = \mathbf{e}_g^T \mathbf{L} \mathbf{e}_g + \mathbf{e}_m^T \mathbf{M} \mathbf{e}_m \tag{5.41}$$

where $\mathbf{L} \in \mathbb{R}^{3\times 3}$, $\mathbf{M} \in \mathbb{R}^{4\times 4}$ are weighting diagonal matrices, whose gains are established by trial and error as previously explained in section 5.1.

Defined the cost function, a dynamic system has to be created to relate the dynamics of the bearing vector with the control inputs. As known, Eq.5.42 holds

and represents the system dynamics.

$$\dot{\mathbf{g}} = \mathcal{B}^W \mathbf{u} \tag{5.42}$$

where:

- $$\dot{\mathbf{g}} = \begin{bmatrix} \dot{g}_x \\ \dot{g}_y \\ \dot{g}_z \end{bmatrix}$$

- $\mathcal{B}^W$ is the Bearing Laplacian matrix defined for a direct graph and expressed in the World Frame as in Eq.5.18.

- $$\mathbf{u} = \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \\ \dot{z}_l \\ \dot{x}_f \\ \dot{y}_f \\ \dot{z}_f \\ \dot{\psi}_l \\ \dot{\psi}_f \end{bmatrix} \in \mathbb{R}^{8\times 1}$$

  where $\dot{x}_l$, $\dot{y}_l$, $\dot{z}_l$ and $\dot{\psi}_l$ are known, while $\dot{x}_f$, $\dot{y}_f$, $\dot{z}_f$ and $\dot{\psi}_f$ are given by the control inputs that have to follow the reference given by the linear velocity and the angular velocity around the z axis of the leader.

Hence, knowing $\dot{x}_f, \dot{y}_f, \dot{z}_f$ it is possible to obtain the pose of the UGV follower just integrating such values as in Eq.5.46.

$$x_f = \int \dot{x}_f dt \tag{5.43}$$

$$y_f = \int \dot{y}_f dt \tag{5.44}$$

$$z_f = \int \dot{z}_f dt \tag{5.45}$$

$$\tag{5.46}$$

Note that, since the UGV moves in the 2-dimensional space, the z component will be later set to zero.

In this definition, no constraints are imposed on the states and on the control

inputs. So, the final formulation of the MPC can be summarized as in Eq.5.47.

$$\min \sum_{k=0}^{N-1} \frac{1}{2}\left(\mathbf{e}_g^T \mathbf{L} \mathbf{e}_g + \mathbf{e}_m^T \mathbf{M} \mathbf{e}_m\right) + \frac{1}{2}\left(\mathbf{e}_g^T \mathbf{N} \mathbf{e}_g\right)$$
$$such \quad that \quad \begin{bmatrix} \dot{g}_x \\ \dot{g}_y \\ \dot{g}_z \end{bmatrix} = \mathcal{B}^W \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \\ \dot{z}_l \\ \dot{x}_f \\ \dot{y}_f \\ \dot{z}_f \\ \dot{\psi}_l \\ \dot{\psi}_f \end{bmatrix} \tag{5.47}$$

As done before for the case of the two UAVs, from the implementation point of view, some parameters used to define the MPC controller have to be set:

- the number of shooting intervals (N) equal to 80;
- The shooting interval time T_s is set to $0.001[s]$;
- the algorithm used to solve the non-linear least squares problem is the Gauss-Newton;
- the chosen integrator is $ERK4$;
- the solver is the $qpoases$;
- the weighting matrices **G**, **H** are diagonal matrices whose entries are chosen by trial and error. In particular:

$$\mathbf{L} = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 100000 & 0 \\ 0 & 0 & 0 & 100000 \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix}$$

Doing so, the trajectory of the UGV follower is the one shown in Fig.5.14, while the comparison between the UAV leader and the UGV follower trajectories is presented in Fig.5.15.

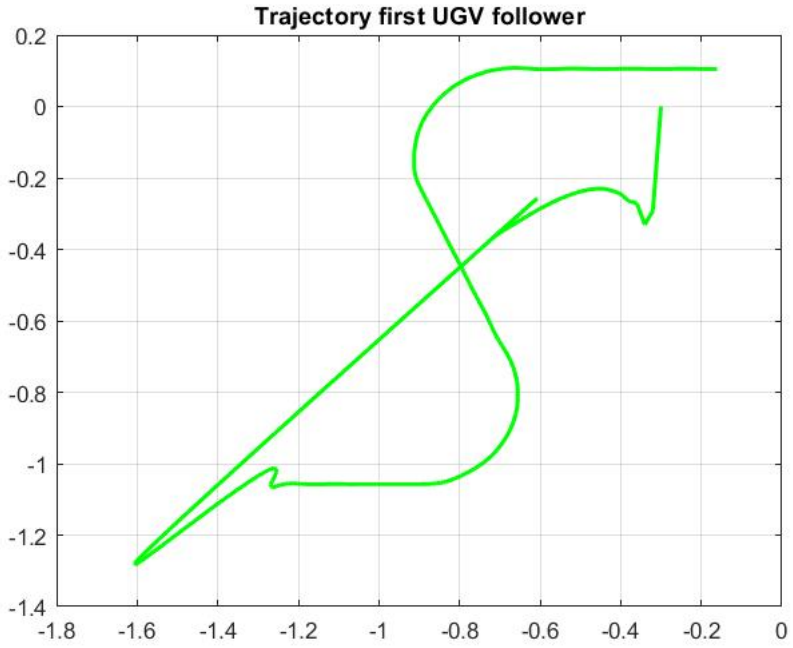


Figure 5.14: *UGV Follower trajectory.*

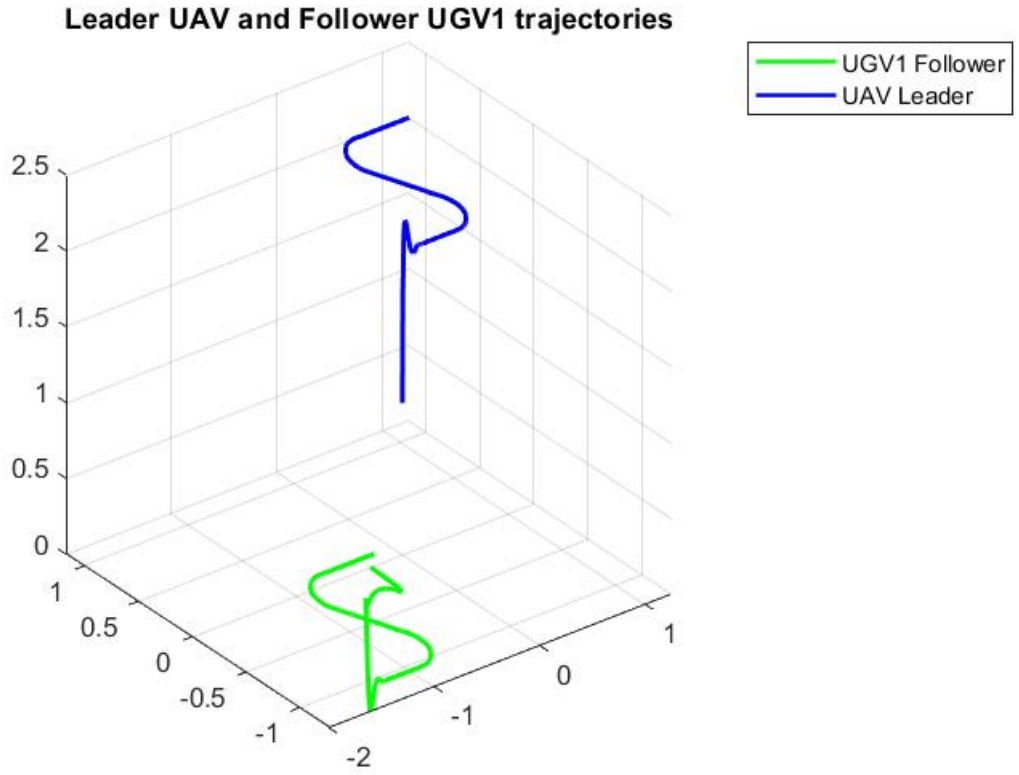


Figure 5.15: *Leader-Follower trajectories: UAV leader and UGV follower.*

Looking at Fig.5.15, it is possible to observe that the UGV follows the UAV leader even if, due to the different starting positions, the UGV has to reach the formation. After having reached such position, it starts to follow the trajectory of the UAV leader.

In detail, the velocities' behaviour of the UGV follower is shown in Fig.5.16, while a comparison between the UAV leader and UGV follower velocities is given in Fig.5.17.

Looking at Fig.5.17, both the linear velocities of the UGV follower oscillate during the first 5 s of the simulation due to the weights' choice and the different initial position chosen with respect to the one of the UAV leader. After the first 5 s of the simulation, the linear velocities coincide according to the fact that the UGV follower tracks the UAV leader trajectory after having reached the formation.

In addition, as explained in section 5.1, the weights can be tuned better through the trial and error approach in order to avoid the big oscillations in the starting instants. To improve the performance of such approach, also the initial position of the UGV can be chosen in such a way to satisfy the condition relative to bearing vector.

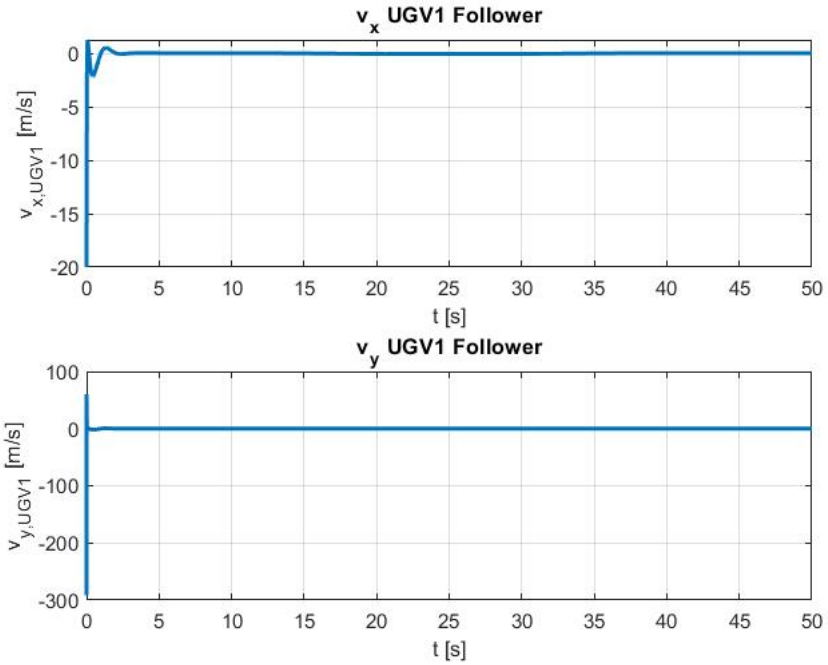


Figure 5.16: *UGV follower linear velocities.*

5.3 Leader-Follower approach between a UAV and two UGVs

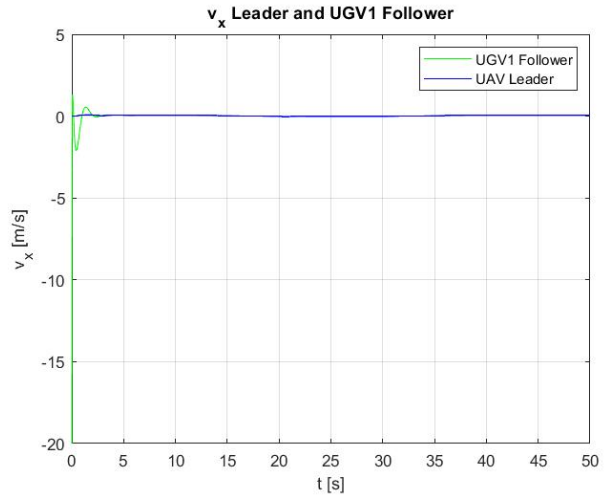
In the next step, it was added another UGV follower so that the heterogeneous MAS is now composed by a UAV leader and two UGVs followers.

It is important to observe that the two considered UGVs have the same actuation capabilities (they are homogeneous) given in Tab.2.1, but the initial positions are different and shown in Tab.5.3.

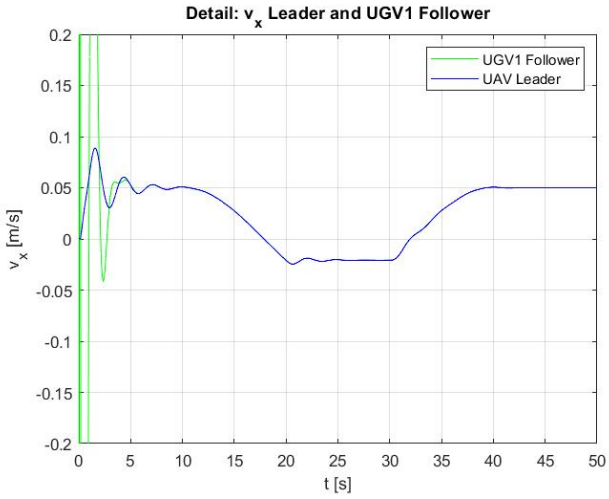
Parameter	Value
x_l	$0\ [m]$
y_l	$0\ [m]$
z_l	$1\ [m]$
$x_{f,UGV1}$	$-0.3\ [m]$
$y_{f,UGV1}$	$0\ [m]$
$z_{f,UGV1}$	$1\ [m]$
$x_{f,UGV2}$	$0.3\ [m]$
$y_{f,UGV2}$	$0\ [m]$
$z_{f,UGV2}$	$1\ [m]$

Table 5.3: *Initial positions of the UAV leader and UGVs followers.*

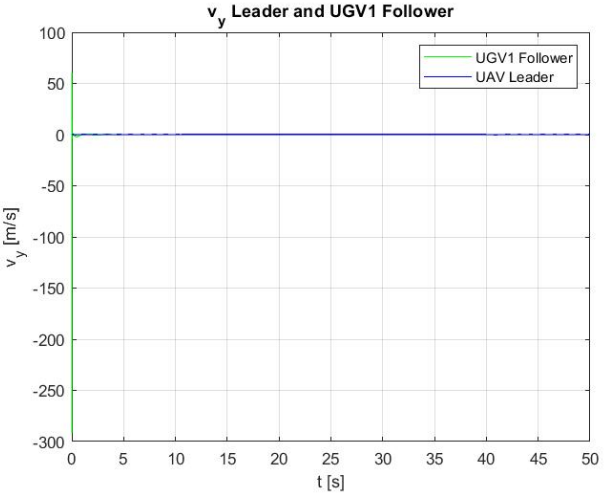
So the two UGVs are controlled separately but they satisfy the same dynamic model, described by the same parameters' values. Therefore, for each UGV, an



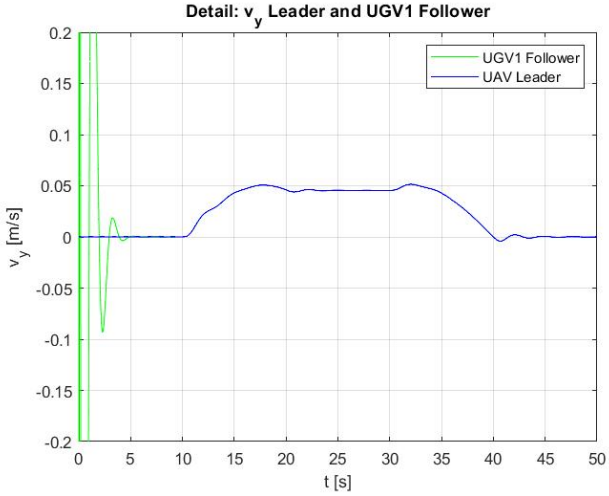
(a) v_x *UAV leader and UGV follower.*



(b) *Detail:* v_x *UAV leader and UGV follower.*



(c) v_y *UAV leader and UGV follower.*



(d) *Detail:* v_y *UAV leader and UGV follower.*

Figure 5.17: *UAV leader and UGV follower linear velocities.*

MPC controller defined as in Eq.5.47 is implemented.

Hence, the trajectory of the second UGV follower is shown in Fig.5.18, while the final result of the simulation comprising a UAV leader and two UGVs followers is shown in Fig.5.19.

According to what previously noted in section 5.2, also the second UGV follower first reaches the position in the formation and then follows the leader trajectory. Despite this, the result can be improved by tuning the weights as explained in section 5.2.

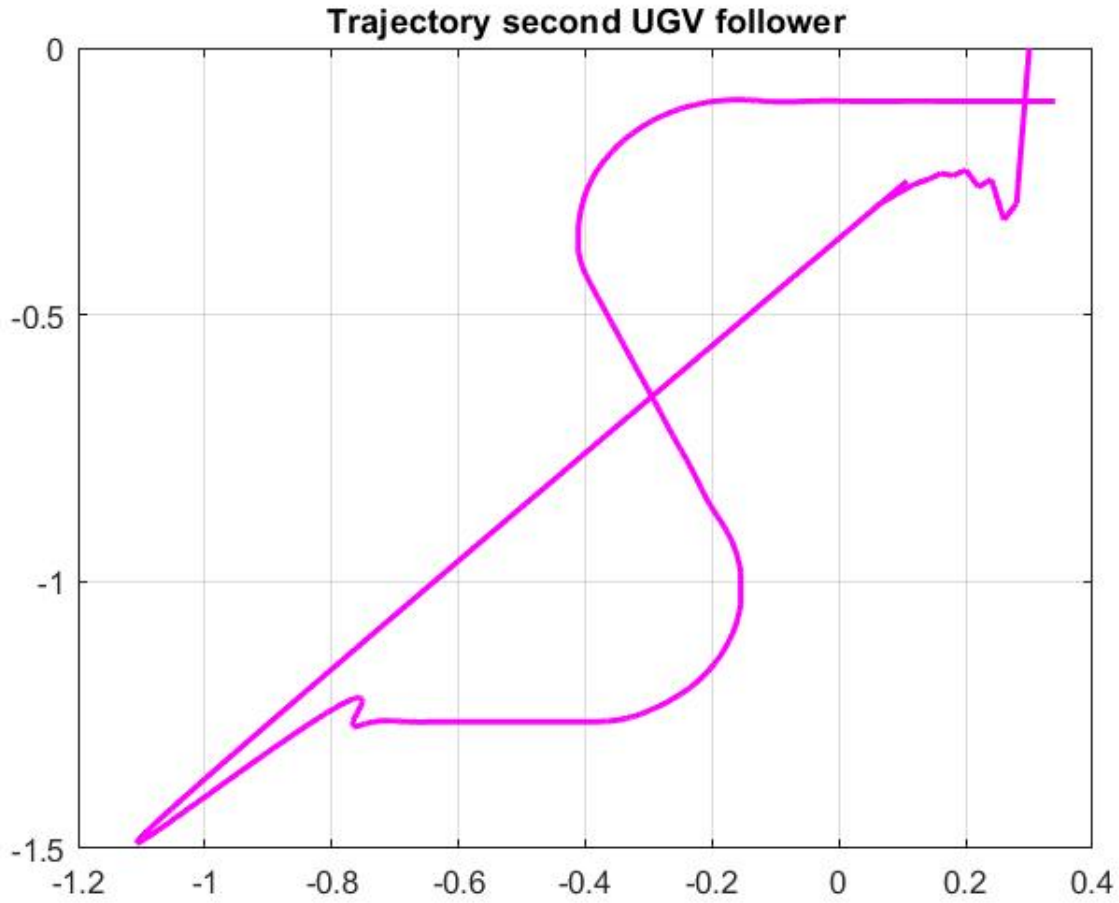


Figure 5.18: *UGV2 follower trajectory.*

5.4 Final results considering 2 UGVs and 2 UAVs

The final goal is reached by joining together the results obtained in sections 5.1 and 5.3 and it is shown in Fig.5.20.

The main problem, in obtaining the final goal, is due to a MATMPC limitation. In fact, this MATLAB Toolbox is created for embedded real-time systems and just one model at the time can be defined. Hence, if in a system there are more than one model and all of them are run, just the last one is considered.

In order to solve this problem, since the model between the two UAVs is different from the one between the UAV and the UGV, considering the UAV as the leader and the other UAV and the two UGVs as followers, two different SIMULINK files are created.

In the first one it is considered the Leader-Follower Approach between the two UAVs and the relative model is defined, while in the second one the Leader-Follower Approach between the UAV and the UGV is considered and the relative

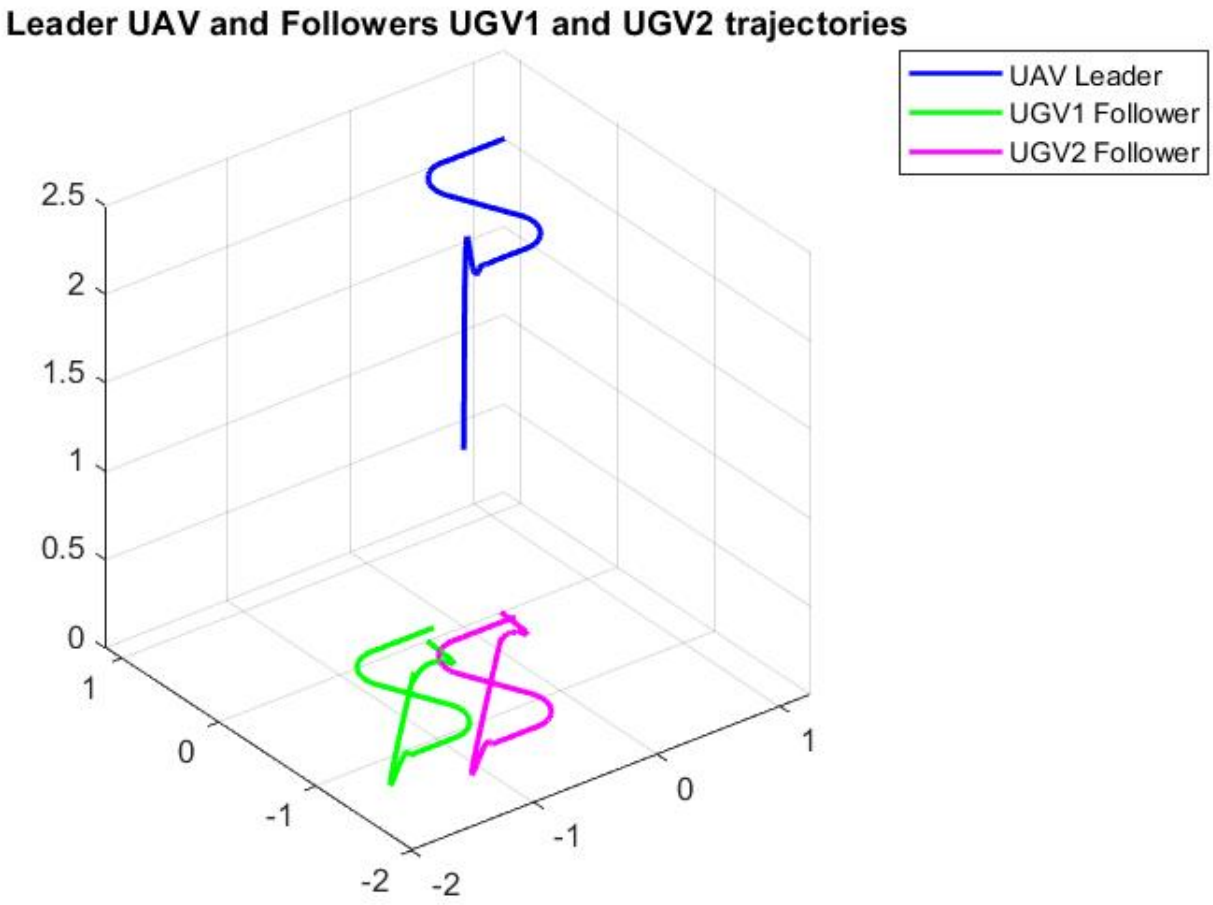


Figure 5.19: *UAV leader and UGVs followers trajectories.*

model is defined.

Then, the results in the scopes of the first SIMULINK file are saved into a file *.m* and used in the main code together with the results obtained by the second SIMULINK file.

5.5 Adding the disturbance

In order to verify the robustness of the obtained results, some white noises are added to the model. In particular the white noises are added to the linear velocities of the UAV leader, in the simulation relative to the Leader-Follower approach between the UAV and the UGV. To better understand the reaction to the disturbances:

- considering the model of the first UGV follower, a white noise of amplitude 0.001 and sample time equal to the shooting time 0.001 s is added to the linear velocity of the UAV leader along the x direction.

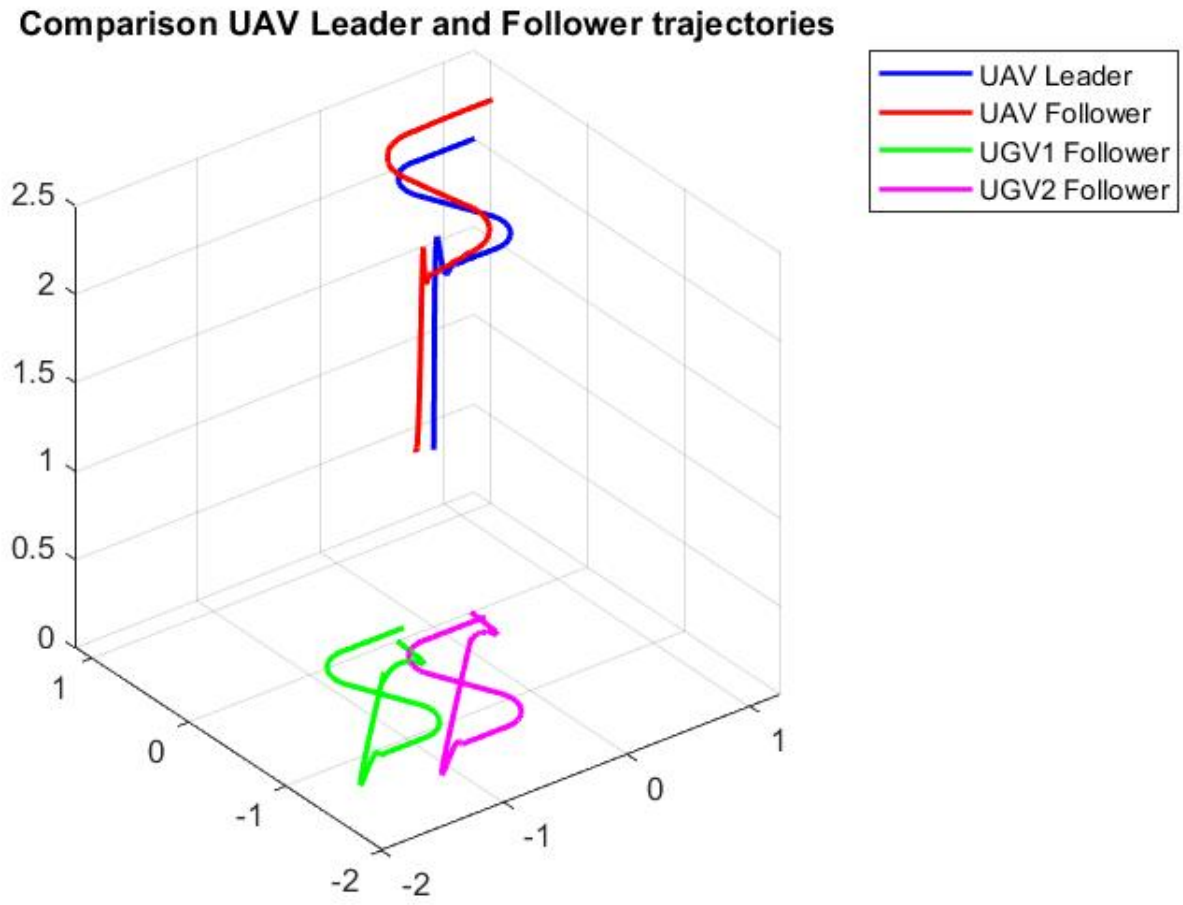


Figure 5.20: *Final trajectories of the MAS without noise.*

- considering the model of the first UGV follower, a white noise of amplitude 0.001 and sample time equal to the shooting time 0.001 s is added to the linear velocity of the UAV leader along the x and y directions.

Considering the first UGV follower with noise, the obtained trajectory, compared to the one of the UAV leader, is the one shown in Fig.5.21.

Instead, taking into account the second UGV follower, the obtained trajectory, compared to the one of the UAV leader, is shown in Fig.5.22.

As before, the UAV follower is then added without considering any noise in this case so that the final result of the entire system is the one shown in Fig.5.23.

Looking at the final plot in Fig.5.23, it is possible to observe that the system react to the disturbances in such a way that the UGVs follow the trajectory even if with some errors, more visible along the curves in both the cases. In details, considering the first UGV follower, the trajectory is more disturbed with respect to the one tracked by the second UGV follower since the amplitude of the noise is bigger in this case.

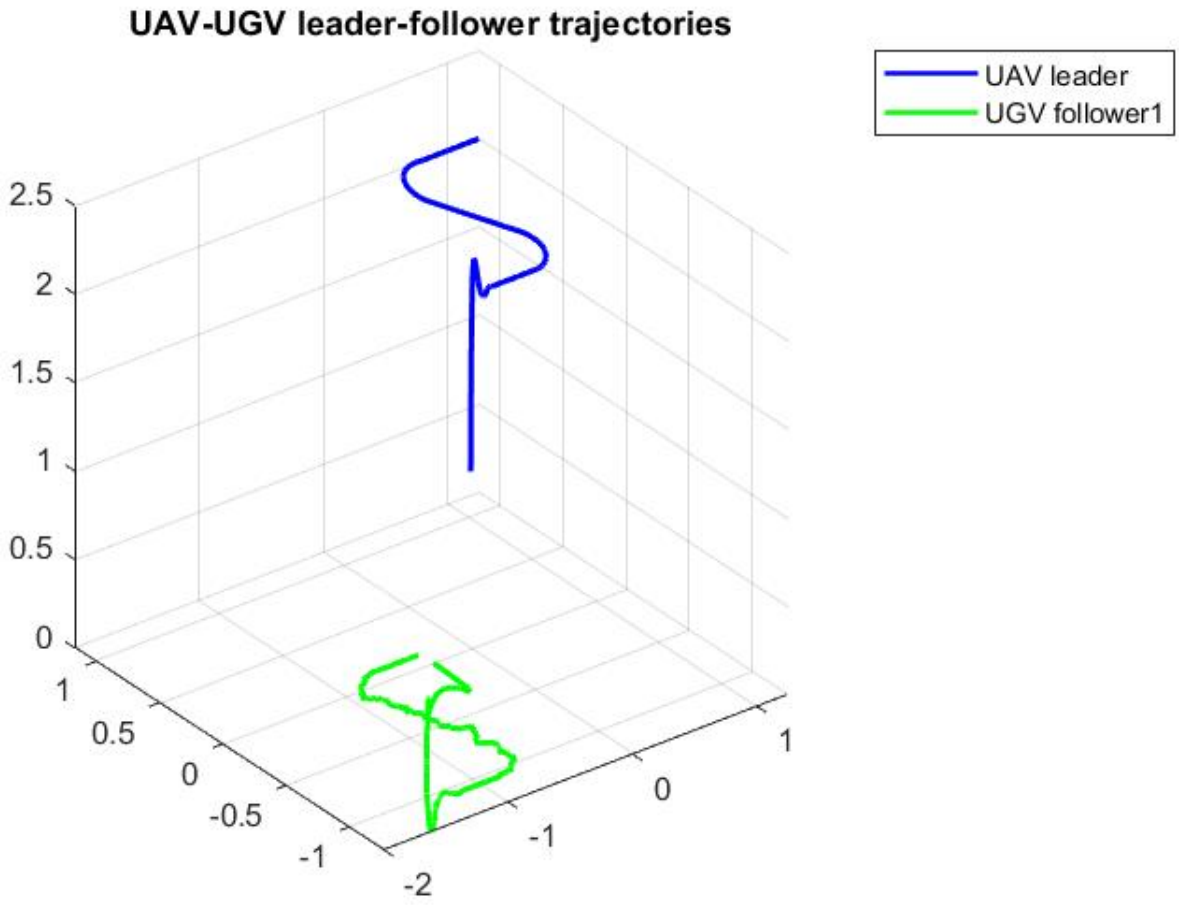


Figure 5.21: *Comparison between the UAV leader trajectory and the first UGV follower trajectory with noise.*

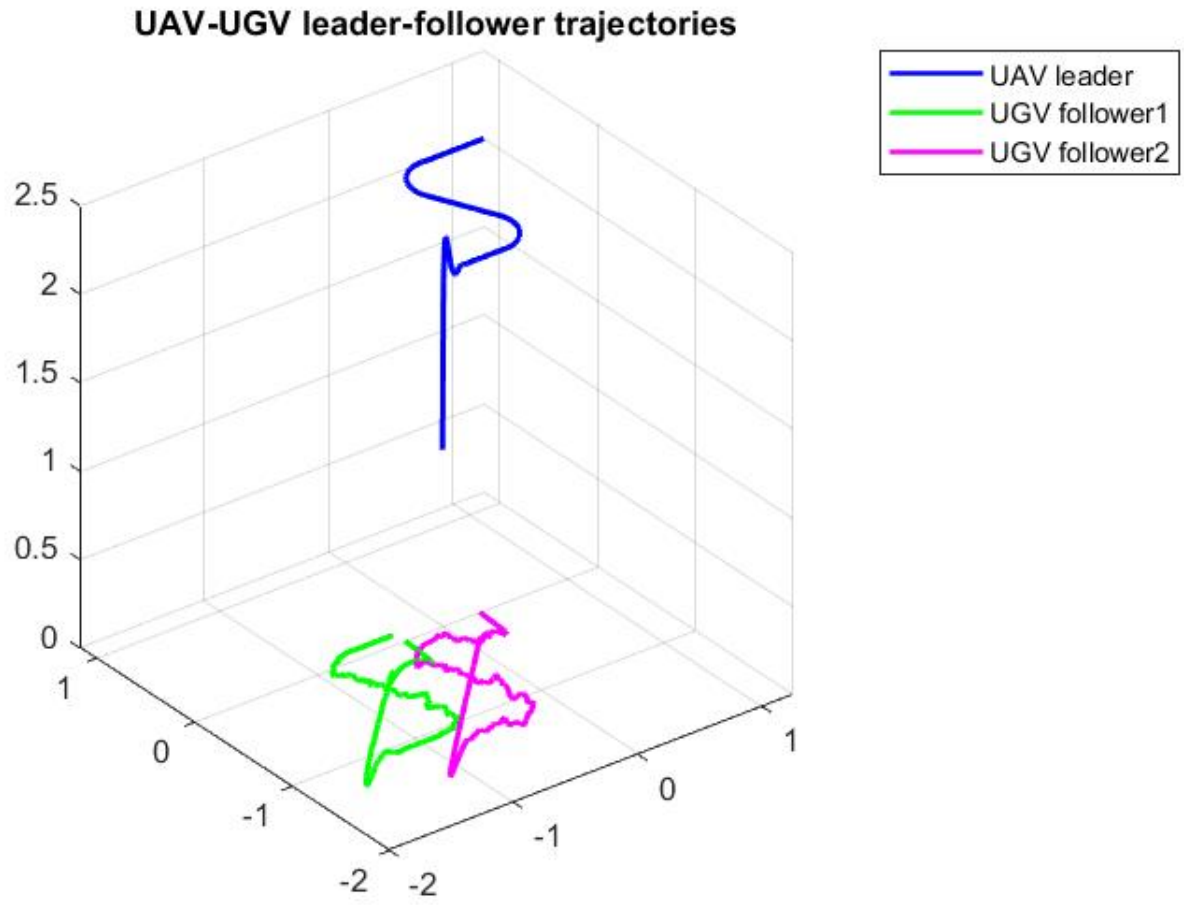


Figure 5.22: *Comparison between the UAV leader trajectory and the UGVs followers trajectories with noise.*

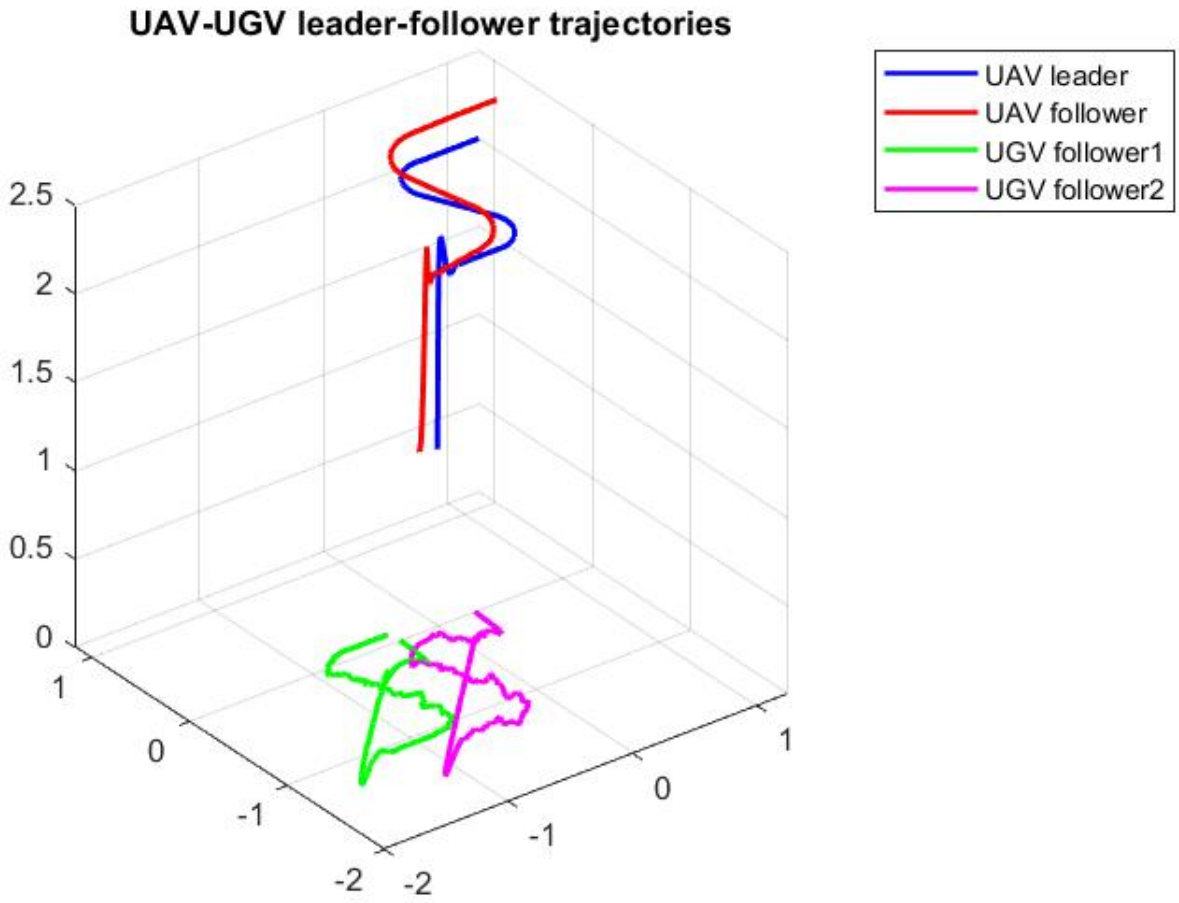


Figure 5.23: *Final system result with noise.*

6

Conclusions

In this thesis, the Leader-Follower approach based on MPC with bearing applied to a MAS, composed by two quadrotors and two unicycles, is studied.

In implementing such approach, instead of considering the Leader-Follower approach based on the distance between the agents, the trajectory tracking for the followers relies on a Model Predictive Controller defined differently depending on the agent considered.

Being the MAS composed by four agents (two UAvs and two UGVs) with different kinematics and dynamics, the problem is studied dividing the MAS in pairs and hence different direct graphs are taken into account.

First the Leader-Follower approach between the UAV leader and UAV follower is implemented. Then, the approach is applied to the UAV leader and a UGV follower and, later, to the UAV leader and the other UGV follower.

In particular, for the case of the Leader-Follower approach applied to a UAV leader and UAV follower, the MPC is based on the bearing vector, linear and angular velocities, angles and thrust and torques in order to develop a more accurate system being agents moving in the 3-dimensional space. In implementing this control system, it is chosen to create an augmented system in order to add the control on thrust and torques to take into account the coupling effect between the attitude and translational dynamics of the quadrotors.

Instead, the Leader-Follower approach applied to the UAV leader and the UGV follower is based on the bearing vector and on the linear velocities being the UGV a 2-dimensional agent.

The implemented MAS works as required since the followers track the same trajectory of the UAV leader after having reached the formation. However, the solution found strongly depends on the desired trajectory to track and on

the weights chosen in the weighting matrices' definitions and on the starting positions of the followers. Hence, changing the starting positions, the weighting matrices have to be adjusted appropriately. In another way, considering the same starting positions of the followers as defined in the previous chapters, in order to improve the performance, different combinations of the weighting matrices can be chosen, paying attention to the fact that some weights influence not only the state or the control input referred to but also other states, as explained in section 5.1.

To verify the robustness of the created system, a noise on the v_x coordinate of amplitude equal to 0.001 and sample time 0.001 s is added in the dynamic system defined for the first UGV follower. Instead, a noise of amplitude 0.001 and sample time 0.001 s is added to v_x and v_y in the dynamic system defined for the second UGV follower. In both these cases, as seen in section 5.5, even if the action of the noises is visible, the followers track the same trajectory of the leader reaching the goal also in this case.

Some future works related to this project may concern:

1. the insertion of obstacles in the trajectory to track. In fact, if the obstacle appears in the UAV leader trajectory, the leader avoids it and, consequently, also the followers change their trajectories. Instead, if the obstacle appears in the followers' trajectory, they cannot avoid the obstacle according with the system created. To realize this improvement, a camera on each agent must be added.

2. the consideration of a trajectory to be tracked for UGVs that is not limited to the Cartesian plane but that is developed in a mountainous scenery. In order to realize this improvement, a more convenient definition is on considering a UGV as the leader, the other UGV as the follower and the UAVs as followers, too. Then, the distance between the UGV leader altitude and the UAV follower one has to be taken into account.

A

Quaternions

The quaternion is an hyper complex entity that represents the extension of a complex number in an higher dimensional space and it is defined as $\mathbf{q} = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} \in \mathbb{S}^3$, where:

$$\eta = \cos(\frac{\theta}{2})$$

$$\epsilon = \mathbf{e}\sin(\frac{\theta}{2})$$

so that:

$$\mathbf{q} = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \mathbf{e}\sin(\frac{\theta}{2}) \end{bmatrix} \tag{A.1}$$

where $\mathbf{e}$ represents a unit vector. Hence, $\mathbf{q}$ represents the rotation of θ around $\mathbf{e}$. The definition of quaternion can also be rewritten as:

$$\mathbf{q} = \eta + i\epsilon_i + j\epsilon_j + k\epsilon_k = \eta + \epsilon = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} \tag{A.2}$$

where η is the scalar real part of the quaternion, $\epsilon = [\epsilon_i \quad \epsilon_j \quad \epsilon_k]^T$ is the complex vector part of the quaternion and (i, j, k) is a coordinate frame abiding the Hamilton's rules:

$$\begin{gathered} i^2 = j^2 = k^2 = ijk = -1 \\ ij = k \quad jk = i \quad ki = j \\ ji = -k \quad kj = -i \quad ik = -j \end{gathered} \tag{A.3}$$

The scalar and vector part of the quaternion are constrained by:

$$\eta^2 + \epsilon_i^2 + \epsilon_j^2 + \epsilon_k^2 = 1 \tag{A.4}$$

from which, the name of *unit quaternion* derives. Accordingly with this, the norm computation of the quaternion results:

$$||\mathbf{q}||^2 = \cos^2(\frac{\theta}{2}) + \sin^2(\frac{\theta}{2}) = \eta^2 + \epsilon^T\epsilon = 1 \tag{A.5}$$

In addition, the quaternion conjugate is defined as in Eq.A.6, while the quaternion inverse as in Eq.A.7.

$$\bar{\mathbf{q}} = \eta - i\epsilon_i - j\epsilon_j - k\epsilon_k = \eta - \epsilon = \begin{bmatrix} \eta \\ -\epsilon \end{bmatrix} \tag{A.6}$$

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{||\mathbf{q}||^2} \tag{A.7}$$

Considering a unit quaternion, the inverse of it is defined as in Eq.A.8.

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{1} = \bar{\mathbf{q}} \tag{A.8}$$

Then, the inner and outer product between quaternions result as shown in Eq.A.9 and Eq.A.10, respectively.

$$\mathbf{q} = \mathbf{q}_1 \cdot \mathbf{q}_2 = (\eta_1 + \epsilon_1) \cdot (\eta_2 + \epsilon_2) = \eta_1\eta_2 + \epsilon_1 \cdot \epsilon_2 = \begin{bmatrix} \eta_1\eta_2 + \epsilon_1^T\epsilon_2 \\ \mathbf{0} \end{bmatrix} \tag{A.9}$$

$$\mathbf{q} = \mathbf{q}_1 \times \mathbf{q}_2 = (\eta_1 + \epsilon_1) \times (\eta_2 + \epsilon_2) = \eta_1\epsilon_2 + \eta_2\epsilon_1 + \epsilon_1 \times \epsilon_2 = \begin{bmatrix} \mathbf{0} \\ \eta_1\epsilon_2 + \eta_2\epsilon_1 + \epsilon_1 \times \epsilon_2 \end{bmatrix} \tag{A.10}$$

where the inner product is defined in the domain:

$$\mathbb{S}^3 \times \mathbb{S}^3 \rightarrow \mathbb{R}$$

while the outer product in:

$$\mathbb{S}^3 \times \mathbb{S}^3 \rightarrow \mathbb{R}^3$$

Instead, the quaternion combination is defined as in Eq.A.11.

$$\mathbf{q}_{tot} = \mathbf{q}_1 \circ \mathbf{q}_2 = \eta_1\eta_2 - \epsilon_1^T\epsilon_2 + \eta_1\epsilon_2 + \eta_2\epsilon_1 + \epsilon_1 \times \epsilon_2 \tag{A.11}$$

where $\eta_1\eta_2 - \epsilon_1^T\epsilon_2$ is the real part, while $\eta_1\epsilon_2 + \eta_2\epsilon_1 + \epsilon_1 \times \epsilon_2$ is the hyper-complex part.

In a compact form, the quaternion combination can be written as in Eq.A.12.

$$\mathbf{q}_{tot} = \mathbf{q}_1 \circ \mathbf{q}_2 = \begin{bmatrix} \eta_1 \\ \epsilon_1 \end{bmatrix} \circ \begin{bmatrix} \eta_2 \\ \epsilon_2 \end{bmatrix} = \mathbf{M}(\mathbf{q}_1)\mathbf{q}_2 = \mathbf{N}(\mathbf{q}_2)\mathbf{q}_1 \tag{A.12}$$

where:

$$\mathbf{M}(\mathbf{q}_1) = \begin{bmatrix} \eta_1 & -\epsilon_1^T \\ \epsilon_1 & \eta_1\mathbf{I}_3 + [\epsilon_1]_x \end{bmatrix}$$

$$\mathbf{N}(\mathbf{q}_2) = \begin{bmatrix} \eta_2 & -\epsilon_2^T \\ \epsilon_2 & \eta_2\mathbf{I}_3 - [\epsilon_2]_x \end{bmatrix}$$

B

Rotation matrices

Alternatively to the quaternion, another way used to represent the relative orientation between two different reference frames as, for example, the Fixed Inertial World Frame $\mathcal{F}_W$ and the Body Frame $\mathcal{F}_B$ consists on the rotation matrices, i.e. $\mathbf{R} \in \mathbb{SO}(n)$ with $n = 2$ for planar rotations and $n = 3$ for full 3D rotations.

Considering two different reference frames as shown in Fig.B.1, the vectors in the Body Frame can be described by the ones defined in the World Frame through the relation in Eq.B.1.

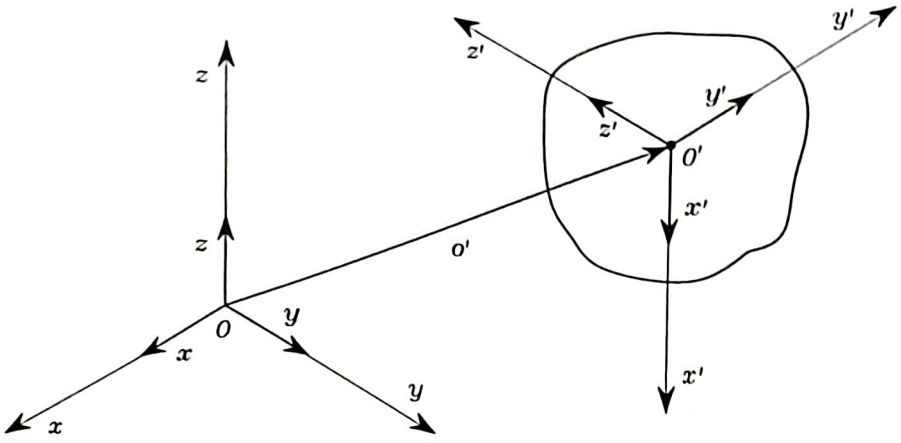


Figure B.1: *World and Body Frames.*

$$\begin{aligned} \mathbf{x}' &= x'_x\mathbf{x} + x'_y\mathbf{y} + x'_z\mathbf{z} \\ \mathbf{y}' &= y'_x\mathbf{x} + y'_y\mathbf{y} + y'_z\mathbf{z} \\ \mathbf{z}' &= z'_x\mathbf{x} + z'_y\mathbf{y} + z'_z\mathbf{z} \end{aligned} \tag{B.1}$$

The relations in Eq.B.1 can be rewritten in a 3×3 matrix to have a compact formulation as shown in Eq.B.2. Such matrix is called rotation matrix.

$$\mathbf{R} = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix} \tag{B.2}$$

Since $\mathbf{R}$ is an orthogonal matrix, this means that:

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$$

where $\mathbf{I}_3$ represents the (3×3) identity matrix.

Another property of the rotation matrix is theta its transpose coincide with its inverse, i.e. $\mathbf{R}^T = \mathbf{R}^{-1}$.

In addition, it holds that:

- $det(\mathbf{R}) = 1$ if the frame is right-handed;
- $det(\mathbf{R}) = -1$ if the frame is left-handed.

Here, the elementary rotations are recalled since the frames can be obtained through such rotations of the reference frame about one of the coordinate axes, where these rotations are positive if they are made counter-clockwise and negative if they are made clockwise.

So:

- the rotation of an angle α around the z axis is expressed as in Eq.B.3.

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

- the rotation of an angle β around the y axis is expressed as in Eq.B.4.

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{B.4}$$

- the rotation of an angle γ around the x axis is expressed as in Eq.B.5.

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{B.5}$$

Considering the elementary rotations, it holds that:

$$\mathbf{R}_k(-\theta) = \mathbf{R}_k^T(\theta) \qquad k = x, y, z$$

Because of the orthogonality condition of the rotation matrices, they give a redundant description of the orientation since the element of each matrix are nine but they are related by six constraints. Due to this three parameters, i.e. three angles, are sufficient to describe the orientation of a rigid body in space.

Hence, a generic rotation matrix can be obtained as the combination of three elementary rotations under the assumption that two successive rotations are not made around parallel axes. This implies that 12 distinct set of angles are allowed out of all 27 combinations, each representing a triplet of Euler Angles.

The two most common triplets of Euler angles are the ZYZ angles and ZYX angles (also called Roll-Pitch-Yaw angles).

The ZYZ angles' rotation is obtained by:

- rotation of the reference frame by the angle ϕ about axis z: $\mathbf{R}_z(\phi)$;
- rotation of the current frame by the angle θ about axis y': $\mathbf{R}_{y'}(\theta)$;
- rotation of the current frame by the angle ψ about axis z'': $\mathbf{R}_{z''}(\psi)$;

In this way, the resulting frame is given by the composition of rotations with respect to the current frame obtained post-multiplying the elementary rotations as shown in Eq.B.6.

$$\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_{y'}(\theta)\mathbf{R}_{z''}(\psi) = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{B.6}$$

Another commonly used triplet of Euler angles are the RPY angles (ZYX angles). In this case, the final rotation is obtained as the combination of the following elementary rotations:

- rotation of the reference frame by an angle ϕ about axis x: $\mathbf{R}_x(\phi)$, yaw;
- rotation of the reference frame by an angle θ about axis y: $\mathbf{R}_y(\theta)$, pitch;
- rotation of the reference frame by an angle ψ about axis z: $\mathbf{R}_z(\psi)$, roll;

Doing so, the resulting frame is given by the composition of rotations with respect to the fixed frame obtained pre-multiplying the elementary rotations as shown in Eq.B.7.

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \tag{B.7}$$

C

Relation between quaternions and rotation matrices

Considering a given quaternion, defined as in AppendixA, the rotation matrix associated to it is expressed as in Eq.C.1.

$$\mathbf{R}(\eta, \epsilon) = \begin{bmatrix} 2(\eta^2 + \epsilon_i^2) - 1 & 2(\epsilon_i\epsilon_j - \eta\epsilon_k) & 2(\epsilon_i\epsilon_k + \eta\epsilon_j) \\ 2(\epsilon_i\epsilon_j + \eta\epsilon_k) & 2(\eta^2 + \epsilon_j^2) - 1 & 2(\epsilon_j\epsilon_k - \eta\epsilon_i) \\ 2(\epsilon_i\epsilon_k - \eta\epsilon_j) & 2(\epsilon_j\epsilon_k + \eta\epsilon_i) & 2(\eta^2 + \epsilon_k^2) - 1 \end{bmatrix} \tag{C.1}$$

Looking at it, it is possible to observe that the rotation $\mathbf{R}(\mathbf{q})$ can be represented by both $\mathbf{q}$ and $-\mathbf{q}$. In a more compact form, the rotation matrix can also be computed as in Eq.C.2.

$$\mathbf{R}(\mathbf{q}) = \mathbf{I} + 2\eta[\epsilon]_x + 2[\epsilon]_x^2 \tag{C.2}$$

where $\mathbf{I} \in \mathbb{R}^{3\times3}$ represents the identity matrix and $[\epsilon]_x$ represents the *skew-symmetric operator* defined as:

$$[\epsilon]_x = \begin{bmatrix} 0 & -\epsilon_k & \epsilon_j \\ \epsilon_k & 0 & -\epsilon_i \\ -\epsilon_j & \epsilon_i & 0 \end{bmatrix}$$

Instead, in order to solve the inverse problem, i.e. from a given rotation matrix to the associated quaternion, it holds that the rotation matrix is generically

defined as in Eq.C.3 and the corresponding quaternion as in Eq.C.4.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{C.3}$$

$$\begin{aligned} \eta &= \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \\ \epsilon &= \frac{1}{2}\begin{bmatrix} sgn(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ sgn(r_{13} - r_{31})\sqrt{r_{22} - r_{11} - r_{33} + 1} \\ sgn(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix} \end{aligned} \tag{C.4}$$

where conventionally it is $sgn(x) = 1$ for $x \geq 0$ and $sgn(x) = -1$ for $x < 0$. In addition it is assumed that $\eta \geq 0$, which means that $\theta \in [-\pi, \pi]$ and any rotation can be described.

The analogy between the inverse of the quaternion and the inverse of the rotation matrix is exploited in the expression Eq.C.5.

$$\mathbf{R}(\mathbf{q}) \rightarrow \mathbf{R}(\mathbf{q})^{-1} = \mathbf{R}(\mathbf{q})^{T} \iff \mathbf{q} \rightarrow \mathbf{q}^{-1} = \bar{\mathbf{q}} \tag{C.5}$$

In the same way, the analogy between the rotation composition with matrices and quaternions is shown in Eq.C.6.

$$\mathbf{R}(\mathbf{q}_{tot}) = \mathbf{R}(\mathbf{q}_1)\mathbf{R}(\mathbf{q}_2) \iff \mathbf{q}_{tot} = \mathbf{q}_1 \circ \mathbf{q}_2 \tag{C.6}$$

D

Leader-Follower approach based on MPC with bearing between two UGVs

In the main project, the Leader-Follower approach between two UAVs and a UAV and a UGV are presented. Instead, in this appendix, the Leader-Follower approach using the MPC with bearing between two UGVs is presented. One of the two UGVs is the leader and knows the trajectory to track, while the other UGV is the follower and does not know the trajectory of the leader neither the distance with it. The two UGVs have the same characteristics: wheel radius, wheel distance, time constant of the low pass filter and maximum and minimum wheel speed described in Tab.2.1. Instead, about the initial poses of the two unicycles, the initial conditions of the UGV leader are the ones shown in Tab.2.1, while the coordinates of the initial pose of the UGV follower are shown in Tab.D.1. where all these parameters are assumed to be known. In addition, it

Parameter's name	**Value**
x_f^{des}	0.15 $[m]$
y_f^{des}	0.15 $[m]$
θ_f^{des}	0 $[rad]$

Table D.1: *Coordinates of the initial pose of the unicycle follower.*

is also assumed that the UGV follower knows the linear and angular velocities of the UGV leader. Hence, the MAS described is modeled as an undirect graph, whose graphical representation is presented in Fig.D.1.

Looking at Fig.D.1, the two vectors represented with the red arrows, i.e. g_{12} and g_{21}, represent the bearing vectors from leader to follower and from follower to leader, respectively.

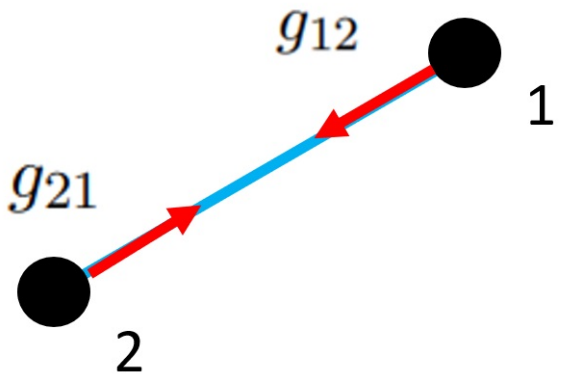


Figure D.1: *Undirect graph representing the MAS composed by two unicycles: node 1 represents the leader, while node 2 represents the follower.*

The leader knows its trajectory and it has been found as shown in section 2.1.4 instead, in order to define the MPC controller to implement the Leader-Follower Approach, three different ingredients have to be defined:

1. cost function;
2. dynamics;
3. eventually constraints.

Considering the definition of the cost function, it is given by the sum of other two cost functions as given by Eq.D.1.

$$\hat{u}^* = \min_u J \quad with \quad J = J_g + J_{v\omega} \tag{D.1}$$

with:

$$J_g = \sum_{k=1}^{N} (\mathbf{g}^* - \mathbf{g})^T \mathbf{A} (\mathbf{g}^* - \mathbf{g}) \tag{D.2}$$

$$J_{v\omega} = \sum_{k=1}^{N} (\mathbf{v}\omega_l - \mathbf{v}\omega_f)^T \mathbf{B} (\mathbf{v}\omega_l - \mathbf{v}\omega_f) \tag{D.3}$$

where $(\mathbf{g}^* - \mathbf{g})$ represents the bearing error vector, $\mathbf{g}^* \in \mathbb{R}^{4\times1}$ represents the desired value of the bearing vector, $\mathbf{g} = \begin{bmatrix} g_{12,x} \\ g_{12,y} \\ g_{21,x} \\ g_{21,y} \end{bmatrix} \in \mathbb{R}^{4\times1}$ is the measured bearing vector, $\mathbf{A} \in \mathbb{R}^{4\times4}$ and $\mathbf{B} \in \mathbb{R}^{2\times2}$ are tuning gain matrices and $(\mathbf{v}\omega_l - \mathbf{v}\omega_f)$ represents the formation motion error vector, where $\mathbf{v}\omega_l = \begin{bmatrix} v_l \\ \omega_l \end{bmatrix}$ gives the linear

and angular velocities of the leader and $\mathbf{v}\omega_f = \begin{bmatrix} v_f \\ \omega_f \end{bmatrix}$ represents the linear and angular velocities of the follower. Hence, the aim of J_g consists in penalizing the bearing error, while $J_{v\omega}$ has to penalize the formation motion error.

Having provided the cost function that must be minimized, the state and control input of the system and its dynamics have to be defined. In this case, the state corresponds to the bearing error vector ($\mathbf{g}^* - \mathbf{g}$), while the control input is the formation motion error vector ($\mathbf{v}\omega_l - \mathbf{v}\omega_f$).

About the dynamics, the relation in Eq.D.4 holds, where $\mathcal{B}^W$ is the Bearing Laplacian matrix defined in the World Frame, while $\mathbf{u} = \begin{bmatrix} v_l \\ v_f \\ \omega_l \\ \omega_f \end{bmatrix}$.

$$\dot{\mathbf{g}} = \mathcal{B}^W \mathbf{u} \tag{D.4}$$

In details, the Bearing Laplacian matrix can be computed as:

$$[\mathcal{B}]_{ij} = \begin{cases} \mathbf{0}_{d\times d} & i \neq j, (i,j) \notin \mathcal{E} \\ -\mathbf{P}_{g_{ij}} & i \neq j, (i,j) \in \mathcal{E} \\ \sum_{j\in\mathcal{N}_i} \mathbf{P}_{g_{ij}} & i = j, (i,j) \in \mathcal{V} \end{cases} \tag{D.5}$$

Doing so, a block structure of the $\mathcal{B}^W$ matrix can be observed as in Eq.D.6.

$$\mathcal{B} = \begin{bmatrix} \mathbf{P}_{g_{12}} & -\mathbf{P}_{g_{12}} \\ -\mathbf{P}_{g_{21}} & \mathbf{P}_{g_{21}} \end{bmatrix} \tag{D.6}$$

where $\mathbf{P}_{g_{ij}}$ is the orthogonal projection of the bearing vector $\mathbf{g}_{ij}$ such that:

$$\begin{aligned} \mathbf{P}_{g_{12}} &= \mathbf{I}_2 - \mathbf{g}_{12}\mathbf{g}_{12}^T \\ \mathbf{P}_{g_{21}} &= \mathbf{I}_2 - \mathbf{g}_{21}\mathbf{g}_{21}^T \end{aligned} \tag{D.7}$$

Hence, the bearing vectors $\mathbf{g}_{12}$ and $\mathbf{g}_{21}$ are defined in the World Frame as in Eq.D.8.

$$\begin{aligned} \mathbf{g}_{12} &= \mathbf{R}_1^T \frac{\mathbf{p}_{12}}{d_{12}} = \mathbf{R}_1^T \frac{\mathbf{p}_2 - \mathbf{p}_1}{||\mathbf{p}_2 - \mathbf{p}_1||} \\ \mathbf{g}_{21} &= \mathbf{R}_2^T \frac{\mathbf{p}_{21}}{d_{21}} = \mathbf{R}_2^T \frac{\mathbf{p}_1 - \mathbf{p}_2}{||\mathbf{p}_1 - \mathbf{p}_2||} \end{aligned} \tag{D.8}$$

Defining the dynamics of the system as in Eq.D.4, the errors (defined as the difference between the known leader's velocities and the unknown follower's velocities) on the velocities are given. So, the value of the linear and angular velocities of the follower can be obtained as the difference between the linear and angular velocity of the leader and the errors on the linear and angular velocity, respectively. The behaviours of the linear and angular velocities of the follower are shown in Fig.D.2. Instead, a comparison between the velocities of the leader

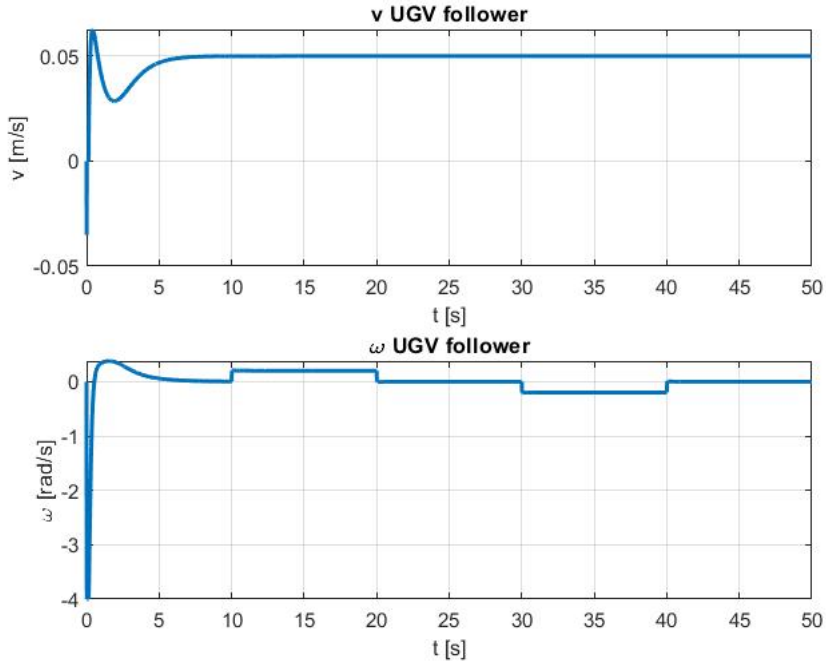


Figure D.2: *Linear and angular velocities of the unicycle follower.*

and the ones of the follower are given in Fig.D.3, from which it is possible to observe that the follower's velocities reach the values of the leader's ones except at the starting instants.

Knowing the velocities of the followers, it is possible to compute the x, y positions and the θ angle such that the resulting trajectory of the unicycle follower is presented in Fig.D.4.

So, a comparison between the trajectories of the leader and of the follower is visible in Fig.D.5. Looking at this plot, it is possible to note that the follower tracks the same trajectory of the leader from a different initial pose. This is confirmed according to the fact that the follower's velocities converge to the leader's ones.

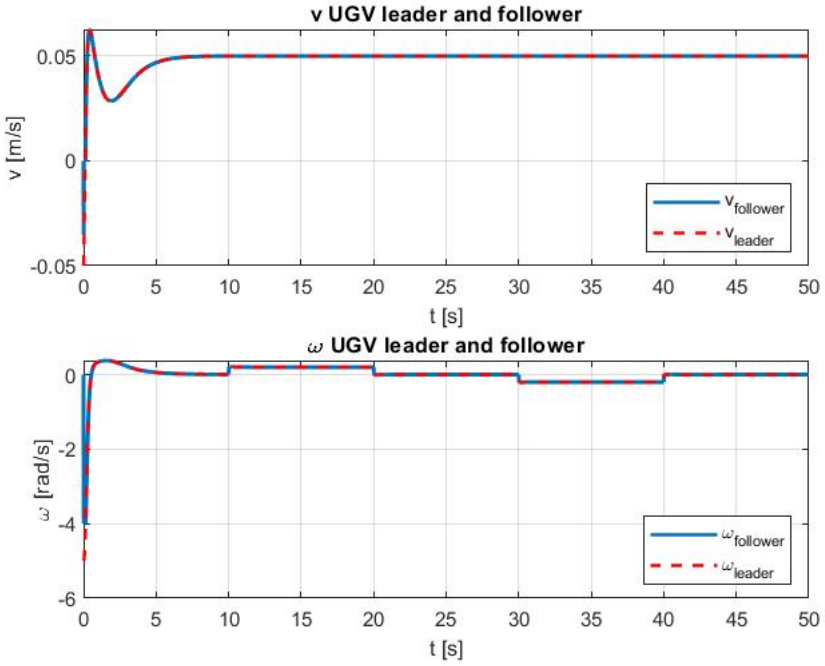


Figure D.3: *Comparison between the linear and angular velocities of the unicycle leader and follower.*

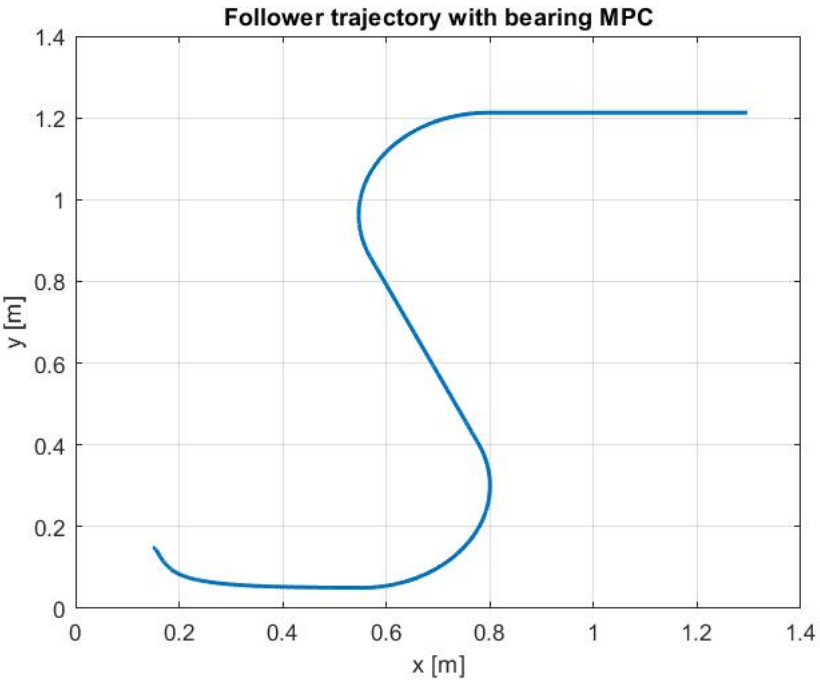


Figure D.4: *Trajectory of the UGV follower.*

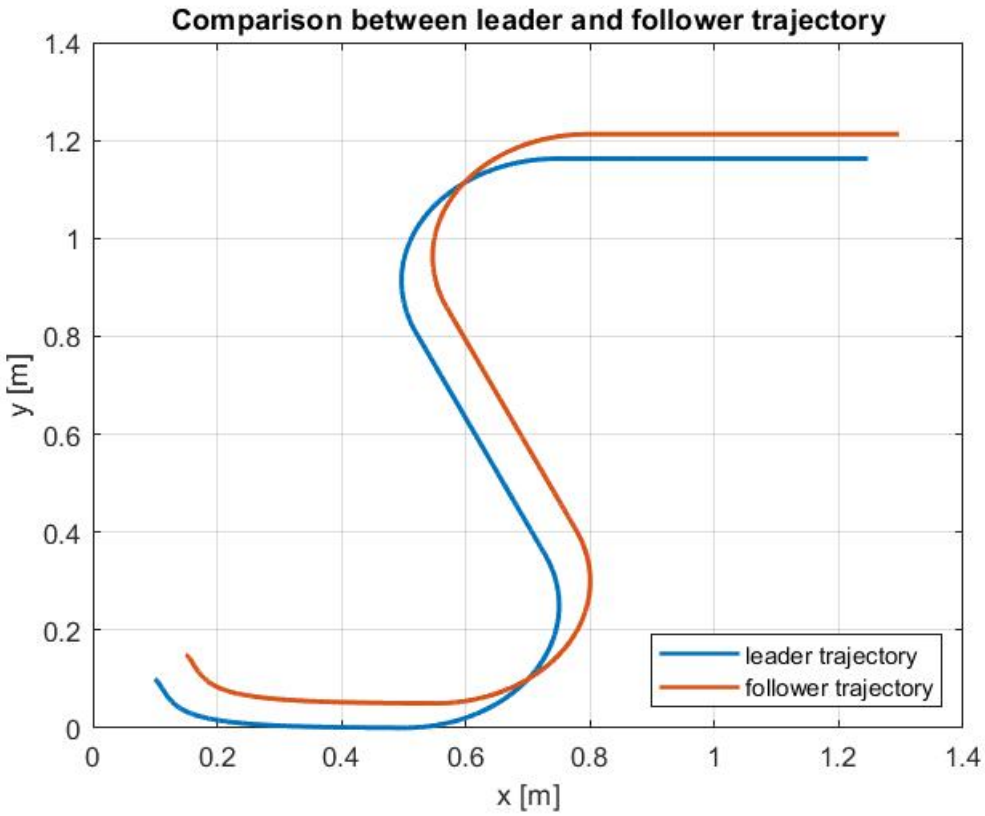


Figure D.5: *Trajectories of the leader and follower UGV.*

References

[1] Aaron D. Ames, Abate A.,Sastry S., *Sufficient Conditions for the Existence of Zeno Behavior*, in Department of Electrical Engineering and Computer Sciences, University of California, Berkeley

[2] Agarwal M.K., Vachhani L., *Development of simulation environment for controllers based on FPGA.*, Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference, July 2012

[3] Ahn H.S., D.O. Anderson B., Sun Z., Trinh M.H., Zelazo D., Zhao S., *Bearing-Based Formation Control of A Group of Agents with Leader-First Follower Structure*, in IEEE Transactions on Automatic Control, 2018

[4] Ahn H.S., Trinh M.H., Sun Z., Zelazo D., Zhao S., *Laman Graphs are Generically Bearing Rigid in Arbitrary Dimensions*, 11 March 2017

[5] Awad M., Rizk Y., Tunstel E., *Cooperative Heterogeneous Multi-Robot Systems: A Survey*, 20 March 2019

[6] Balaji V., Rajaji L.,*Comparative study of PID and MPC controller using LAB VIEW*, in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 2, Issue 11, November 2013

[7] Baotic M., Maurovic I., Petrovic I., *Explicit Model Predictive Control for Trajectory Tracking with Mobile Robots*, in Proceedings of 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July 3-7, 2011, Budapest, Hungary

[8] Barrientos A., Garzon M., Velente J., Towards a ground navigation system based in visual feedback provided by a mini UAV, Conference: 2012 International IEEE Intelligent Vehicles Symposium W5: "Perception in robotics", Alcalá de Henares, Spain, June 2012

[9] Beghi A., Bruschetta M., Chen Y., Picotti E., *MATMPC - A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control*, 2019 18th European Control Conference (ECC), Napoli, Italy, June 25-28, 2019

[10] Bemporad A., Rocchi C., *Decentralized Hybrid Model Predictive Control of a Formation of Unmanned Aerial Vehicles*, University of Trento, Italy

[11] Buehler J., *Capabilities in Heterogeneous Multi-robot Systems*, in ARC Centre of Excellence in Autonomous Systems, 2012, Sydney, Australia

[12] Campobasso M.J., *Leader-Follower Trajectory Generation and Tracking for Quadrotor Swarms*, in Dissertations and Theses, 2017

[13] Castro-Linares R., Vallejo-Alarcťon M.A., Velasco-Villa M., *Unicycle-Type Robot Quadrotor Leader-Follower Formation Backstepping Control*, 2015, Department of Electrical Engineering, Mechatronics Section, Mexico City, Mexico

[14] Cenedese A., Michieletto G., Pozzan B., Zelazo D., *Heterogeneous Formation Control: a Bearing Rigidity Approach*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021, Austin, Texas

[15] Cenedese A., Franchi A., Michieletto G., *Bearing Rigidity Theory in SE(3)*, in IEEE 55th Conference on Decision and Control (CDC), December 12-14, 2016, Las Vegas, USA

[16] Chen J., Yuan L, Wu F., *Leader-Follower Formation Control for Quadrotors*, Conference Paper in IOP Conference Series Materials Science and Engineering, December 2016

[17] Cheng-Heng Fua, Shuzhi Sam Ge, *COBOS: Cooperative Backoff Adaptive Scheme for Multirobot Task Allocation*, in IEEE transactions on robotics, VOL. 21, NO. 6, December 2005

[18] De Gennaro M.C., Jadbabaie A., *Decentralized Control of Connectivity for Multi-Agent Systems*, in Proceedings of the 45th IEEE Conference on Decision Control, San Diego, CA, USA, December 13-15, 2006

[19] Dorri A., SALIL S. KANHERE1, JURDAK R., *Multi-Agent Systems: A Survey*, in School of Computer Science and Engineering, University of New South Wales, June 19, 2018, Sydney, Australia

[20] Franchi A., Robufffo Giordano P., Schiano F., Zelazo D., *A Rigidity-Based Decentralized Bearing Formation Controller for Groups of Quadrotor UAVs*, in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon Convention Center, October 9-14, 2016, Daejeon, Korea

[21] Haifan Su, Cailian Chen, Ziwen Yang, Shanying Zhu, Xinping Guan, *Bearing-Based Formation Tracking Control With Time-Varying Velocity Estimation*, IEEE TRANSACTIONS ON CYBERNETICS, Manuscript received February 18, 2022; accepted April 11, 2022

[22] Hirche S., Lederer A.,Liu Q., Jiao J., Sosnowski S.,Yang Z., *Distributed Learning Consensus Control for Unknown Nonlinear Multi-Agent Systems based on Gaussian Processes*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021. Austin, Texas

[23] Hou Z., Fantoni I., *Interactive Leader-Follower Consensus of Multiple Quadrotors Based on Composite Nonlinear Feedback Control*, in IEEE transactions on Control systems technology, VOL.26, NO.5, University of Nantes, 5 September 2018

[24] Huang Y., Wang L., Wang Q., Wu M., *Formation Control of Heterogeneous Multi-Robot Systems*, in Proceedings of the 17th World Congress, July 6-11, 2008, Seoul, Korea

[25] Jennings N.R., *On agent-based software engineering*,21 September 1999, University of Southampton, UK

[26] Jia R., Yue M., Xu Y., *Adaptive leader-follower formation control of wheeled mobile robots via composite techniques*, in Proceedings of 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, July 19-23, 2018, Tianjin, China

[27] Jianing Zhao, Xiao Yu, Xianwei Li, Hesheng Wang, *Bearing-Only Formation Tracking Control of Multi-Agent Systems With Local Reference Frames and Constant-Velocity Leaders*, IEEE CONTROL SYSTEMS LETTERS, VOL. 5, NO. 1, JANUARY 2021

[28] Johansson K.H., Tsang K. F.E., *Distributed Event-Triggered Learning-Based Control for Nonlinear Multi-Agent Systems*, in 60th IEEE Conference on Decision and Control (CDC), December 13-15, 2021. Austin, Texas

[29] Kefan Wu, Junyan Hu, Barry Lennox, Farshad Arvin, *Finite-Time Bearing-Only Formation Tracking of Heterogeneous Mobile Robots with Collision Avoidance*, in IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS, 2020, Manchester

[30] Kim B., Necsulescu D., Pruner E., Sasiadek J., *Control of Decentralized Geometric Formations of Mobile Robots*, in IEEE 17ř International Conference on MethodsModels in AutomationRobotics (MMAR), 2012, Poland

[31] Klancar G., Skrjanc I., *Tracking-error model-based predictive control for mobile robots in real time, Laboratory of Modelling, Simulation and Control*, received 21 December 2005, received in revised form 4 January 2007, accepted 24 January 2007, available online 3 February 2007, Faculty of Electrical Engineering, University of Ljubljna, Trzaska 25, Slovenia

[32] Krajnťk T., Preucil L., Saska M., Vonťasek V., *Coordination and navigation of heterogeneous UAVs-UGVs teams localized by a hawk-eye approach*

[33] Liang Liu, Xiaopeng Luo, Zhangqing Zhu, *Distributed Formation Control of Unicycle-Like Vehicles Without Direct Distance Measurements*, Department of Control and Systems Engineering, School of Management and Engineering, 2015, Nanjing University

[34] Mali P., Miss. Jagtap Bhagyashree K., Mr.Pawar Kuldeep P., *Nonlinear Process Identification and Model Predictive Control using Neural Network*, in International Journal of Engineering and Technology (IJET), Vol 4 No 5 Oct-Nov 2012

[35] Minh Hoang Trinh, Shiyu Zhao, Zhiyong Sun, Daniel Zelazo, Brian D. O. Anderson, Hyo-Sung Ahn, *Bearing-Based Formation Control of A Group of Agents with Leader-First Follower Structure*, IEEE Transactions on Automatic Control, 2018

[36] Montenegro S., Qasim A., *Role of Graphs for Multi-Agent Systems and Generalization of Eulers Formula*, in IEEE 8th International Conference on Intelligent Systems, 2016

[37] Oriolo G., Sciavicco L., Siciliano B., Villani L., *Robotics. Modeling, Planning and Control*, Springer

[38] Santos Brandao A., Sarcinelli-Filho M., Vago Santana L., *Heterogeneous Leader-Follower Formation based on Kinematic Models*, 2016 International Con-

ference on Unmanned Aircraft Systems, June 7-10, 2016, Arlington, VA USA

[39] Stephen Bassi Joseph, Emmanuel Gbenga Dada, Afeez Abidemi, David Opeoluwa Oyewola, Ban Mohammed Khammas, *Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems*, 5 May 2022

[40] Xiao F., Wanga L., Chenb J., Gaoa Y., *Finite-time formation control for multi-agent systems* , in 2009 Elsevier Ltd, 2009

[41] Xiaoliang Yang, Guorong Liu, Anping Li, Le Van Dai, *A Predictive Power Control Strategy for DFIGs Based on a Wind Energy Converter System*, 26 July 2017

[42] Xie J., Liu C.C., *Multi-agent systems and their applications*, 14 July 2017, Journal of International Council on Electrical Engineering, Published by Informa UK Limited, trading as Taylor Francis Group

[43] Wilson R.J., *Introduction to graph theory*, Fourth Edition, Prentice Hall, 1998

[44] Zelazo D., Zhao S., *Bearing Rigidity Theory and its Applications for Control and Estimation of Network Systems*, March 14, 2018

[45] Zhao S., Zhengtao D., Zhenhong L., *Bearing-Only Formation Tracking Control of Multi-Agent Systems*, in IEEE Transactions on Automatic Control, 2019, Manchester