POLYTECHNIC OF TURIN

MASTER's Degree in Mathematical Engineering

Politecnico di Torino

MASTER's Degree Thesis

Game-theoretic approach for robust nonlinear Model Predictive Control on network dynamics

Supervisors	Candidate
Prof. Michele PAGONE	Giovanni MARINELLO
Prof. Lorenzo ZINO	

Academic year 2023/2024

Summary

In this thesis, we explore the formation control of unmanned ground vehicles (UGVs) using a Nonlinear Model Predictive Control (NMPC) approach based on the Pontryagin Minimum (maximum, in the original form) Principle. The UGV model incorporates essential information about the involved agents dynamics, guiding our pursuit of optimal trajectories while concurrently establishing a dynamic network among them.

The main goal of the thesis is to bridge the theoretical constructs with practical realities by introducing disturbances into the model. These disturbances, representing real-world uncertainties, contribute to the complexity of the control problem. To address the robust NMPC problem, we formulate the optimal control problem, in presence of exogeneous disturbance, as min-max optimization, wherein the goal is to minimize the trajectories for formation while maximizing the disturbances. To this end, we employ game theoretic approach, where the min-max optimal control problem can be viewed as a zero-sum game, with the aim of determining the minimum control inputs required for the formation while strategically maximizing the impact of disturbances.

The incorporation of game theory introduces a strategic dimension to the control problem, seeking a Nash equilibrium point which coincides with a saddle node point of the Hamiltonian. By determining the parameters leading to this equilibrium, we establish the optimal control inputs necessary for formation under the influence of disturbances.

Extending our focus beyond UGVs, we also propose the management of collision avoidance for drones within the formation. This is achieved through the implementation of the Artificial Potential Field (APF) method, enhancing the overall robustness and safety of the multi-agent system.

The comprehensive approach outlined in this thesis combines advanced control techniques, disturbance modeling, and game theory, offering a nuanced perspective

on the challenges associated with formation control in dynamic environments. The findings contribute to the growing body of knowledge in autonomous systems, paving the way for practical implementations in real-world scenarios.

Acknowledgements

I want to thank my family – my dad, mom, and brother – for their constant support and love. They've been my main source of support throughout this journey.

A big thanks to my professors for their help and guidance. I'm grateful for all they've done to help me reach my goals.

I also want to remember my grandma, who passed away recently. She won't be here to see me finish my master's, but her memory motivates me every day.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent decades, there has been a significant surge in the utilization and development of Unmanned Vehicles (UVs) **Liu et al.** [**1**]. The ability to operate without human intervention and enhance autonomous systems has led UVs to find applications in different fields. While initially predominantly employed in military operations, there has been a notable shift towards extending their use to civilian operations, such as search and rescue.

Depending on the specific application, various categories of UVs exist, including Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs), Unmanned Surface Vehicles (USVs), and Autonomous Underwater Vehicles (AUVs).

When assessing the applications of these platforms, a frequently observed limitation is their typically small size and limited capability, rendering them suitable primarily for relatively straightforward missions. Additionally, a significant number of current unmanned vehicle platforms exhibit low levels of autonomy, with some being remotely controlled or only semi-autonomous.

To address and manage these limitations more effectively, deploying these small vehicles in formation, constituting a fleet, proves to be a valuable strategy. This approach allows for the coverage of wider mission areas with improved system robustness, coordination, and fault-tolerant capabilities.

In the context of this thesis, the focus will be on the formation of Unmanned Ground Vehicles (UGVs). Specifically, the study will delve into the application of Nonlinear Model Predictive Control (NMPC) and its extensions to aid in the formation of these devices. Additionally, Artificial Potential Field (APF) will be employed to enhance management in this aspect, particularly in avoiding collisions within the formation.

1.1 Formation and State of Art

The fundamental notion of formation about UV systems finds its inspiration in the intricate behaviors exhibited by animals, particularly in phenomena like bird flocking or fish schooling. By closely mimicking these patterns, unmanned vehicles can seamlessly navigate and operate within well-structured formations. This emulation of nature's inherent order not only facilitates the execution of various tasks but also contributes significantly to the enhancement of overall system autonomy. The utilization of formations enables a synchronized and cooperative approach, enabling unmanned vehicles to accomplish complex missions with heightened efficiency and adaptability. This paradigm not only leverages the wisdom of nature but also represents a crucial stride in the evolution of autonomous systems, promoting a more robust and intelligent integration of unmanned vehicles into diverse operational landscapes **Liu et al.** [1].

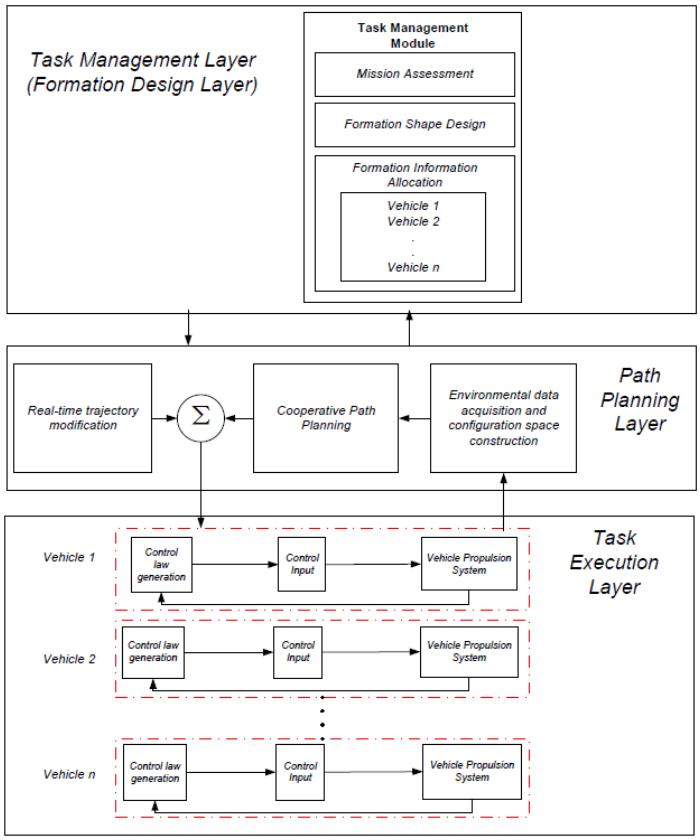

Figure 1.1: Formation Layers

A generic hierarchical architeture for the formation has been proposed by Liu

and Bucknall **Liu et al.** [2]. The structure is divided in 3 important layers: **Task Management Layer**, **Path Planning Layer** and **Task Execution Layer** (as we can observe in Figure 1.1). The **Task Management Layer** is designed to address a particular type of mission in the most time-efficient manner possible.
The **Path Planning Layer** is dedicated to determining the optimal trajectory for the formation. It is characterized by a real-time trajectory modification module, a data acquisition module, and a cooperative path planning module.
Additionally, there is the **Execution Layer**, which establishes a direct connection with the propulsion system of the unmanned vehicle and generates control laws. To enhance system performance, real-time information, such as vehicle velocity and position, is fed back to the upper layer to modify the trajectory in the near future. This process creates a closed control loop, contributing to the improvement of overall system efficiency.

1.1.1 Task Management Layer

In this layer, the primary objective is the optimization of a designated application, with a focal point on minimizing the execution time required.

An illustrative example of such an application is discernible in various research studies, as exemplified by references **Song et al.** [3] and **Guerra et al.** [4], where exclusively Unmanned Aerial Vehicles (UAVs) are utilized. The central concept revolves around configuring a formation through the utilization of a swarm, concurrently or subsequently engaging in the pursuit of an unidentified object (Figure 1.2). Another noteworthy application involving the collaborative operation of UAVs and UGVs pertains to wildfire hotspot surveillance. In this context, UAVs are employed to detect fires in the air, while UGVs focus on identifying specific suspected hotspots, as depicted in the work presented by **Pasini et al.** [5].

However, this thesis does not extend to further exploration of information within this layer, whereas it focuses on the path planning and the task execution layers.

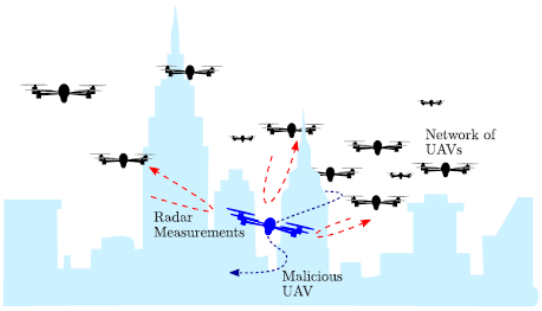

Figure 1.2: Swarm Pursuit

1.1.2 Path Planning Layer

The path planning layer is the heart of the shape of UVs formation. Our propose is the usage of a leader-follower structure, also known as a master-slave configuration, as outlined in **Park et al.** [**6**]. In this arrangement, a master or leader Unmanned Ground Vehicle (UGV) occupies a designated position in space, while the other vehicles calculate their positions based on the leader's information. After that, the scheme reached has to move maintaining the formation. All this is also described and illustrated in Figure 1.3.

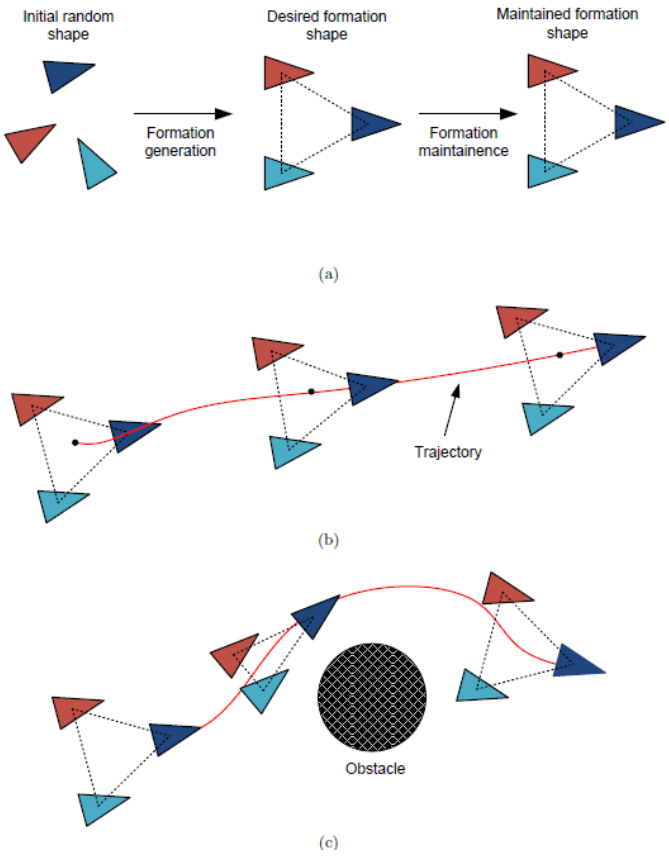

Figure 1.3: (a) Formation generation and maintenance. (b) Formation maintenance while tracking trajectory. (c) Formation shape variation and re-generation. **Liu et al.** [**1**]

In this layer, the Artificial Potential Fields (APF) method can be inserted. It is used to avoid possible collisions between UVs and also between UVs and objects (see (c) in Figure 1.3). It consists of potentials/forces, which can be repulsive or attractive. Repulsive potentials/forces are used to avoid a certain collision, and attractive potentials/forces are used to reach a certain objective (Figure 1.4 from

Ravankar et al. [7]), optimizing the results and the time spent to reach the objective.

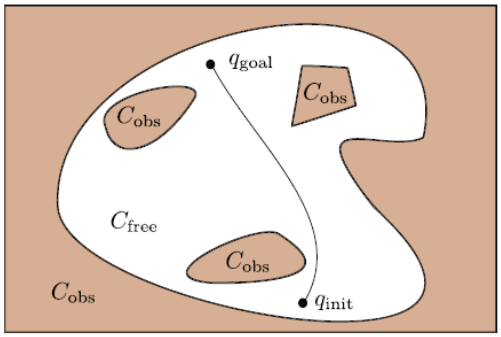

Figure 1.4: C_{obs}: repulsive field; q_{goal}: attractive field

We can view, for example, an application in automated parking as in **Dong et al.** [8], where a location contains an attractive potential, and the borders around the location contain a repulsive potential, as we can observe in Figure 1.5.

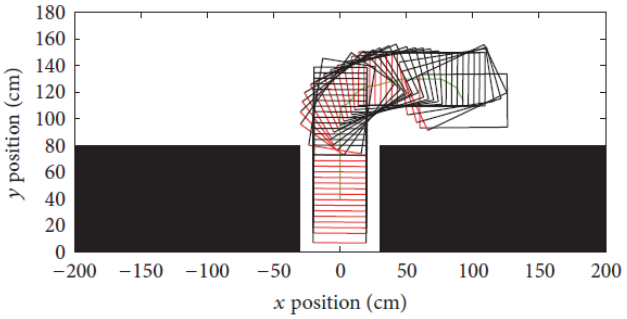

Figure 1.5: Parking with APF

Another example can be viewed in **Souza et al.** [9], where APF method is used to avoid collision for UAVs with obstacles on a 3D simulation (example in Figure 1.6).

Another interesting example is the usage of APF applied to autonomous rendezvous maneuvers to avoid obstacles, as detailed in **Mancini et al.** [10].

So in this layer, thanks to the decision of the shape of formation, it is possible, in real-time, to find the right trajectory for the UVs, communicating information with the **Task Execution Layer** and also with the **Task Management Layer** for possible practical applications.

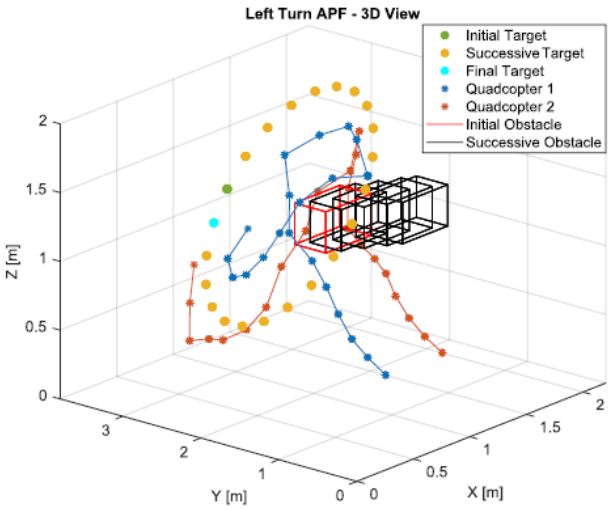

Figure 1.6: APF application 3D

1.1.3 Task Execution Layer

In this layer, the incorporation of data derived from the Path Planning Layer is essential for identifying optimal control parameters necessary for establishing a specific trajectory and, consequently, the formation. Diverse models, encompassing both stochastic and deterministic methodologies, are employed to ascertain suitable parameters for the control of Unmanned Vehicles (UVs).

A prominent model within this layer, which interfaces concurrently with the Path Planning Layer and is to be utilized in this study, is the Nonlinear Model Predictive Control (NMPC). The significant interdependence with the aforementioned layer is noteworthy, as the primary goal is to minimize a predetermined trajectory while simultaneously optimizing control parameters.

NMPC finds widespread application across various domains such as finance **Primbs et al.** [**11**], epidemic environments **Stursberg et al.** [**12**], automotive environments **Di Cairano et al.** [**13**], and so on.
Its application varies across contexts, adapting and strengthening according to specific requirements. For instance, in a study **Pagone et al.** [**14**], NMPC is applied to an academic model known as the Lotka-Volterra model, demonstrating its utilization in a nonlinear context with the Pontryagin approach. Another instance involves the application of NMPC with the introduction of noise through the utilization of game theory **Pagone et al.** [**15**]. This variant, known as robust NMPC, aims to minimize a defined trajectory despite the challenges given by noise.

Here, the academic model considered is the Van Der Pol model.

1.2 Resume of Objectives

The primary objective of our thesis is to develop an advanced model for Unmanned Ground Vehicles (UGVs) and implement effective formation control using the Nonlinear Model Predictive Control (NMPC) algorithm with Pontryagin approach. We plan to adopt a leader(master)-follower(slave) scheme where a master drone defines a predetermined position, and other drones adjust their positions based on the leader's/master information. The goal will be to determine the optimal trajectory and control parameters through the application of NMPC.

In addition, our goal is to inject a challenging element into the formation process by introducing noise into the system, as form of friction for UGVs. We plan to utilize game theory to optimize the formation in the presence of disturbances. This will involve extending NMPC with a robust Pontryagin approach, employing a non-cooperative game with two players, specifically a zero-sum game.

Moreover, an interesting aspect of our work will be addressing the issue of collisions between drones during formation. To achieve this, we intend to use the Artificial Potential Field (APF) method to avoid collisions, ensuring safe and coordinated movement within the formation.

In summary, our research focuses on creating an advanced UGV model capable of forming a coordinated swarm through the use of NMPC and APF, effectively managing noise through game theory, and implementing a robust NMPC approach. This integrated approach aims to enhance the autonomy and overall efficiency of vehicle operations without compromising safety, giving good performances in trajectories and in time.

Chapter 2

Theoretical Background

This chapter aims to provide a comprehensive overview of the theoretical constructs that form the basis for the models discussed herein, offering a detailed exploration of their conceptual foundations and methodological grounds.

2.1 Unmanned Ground Vehicle (UGV) Model

2.1.1 UGV model without disturbances

Within the context of this thesis, the selected model for simulating a single Unmanned Ground Vehicle (UGV) is characterized by the following set of motion equations:

$$\ddot{x} = F\cos(\theta) \tag{2.1}$$

$$\dot{\theta} = \omega \tag{2.2}$$

A simplified point model is employed for individual agents/UGVs, where F and ω represent the control input parameters. Here, F denotes the input force that we want to control, acting on the agent, and ω represents its angular velocity, also under our control. Schematically, we define the control input as $u = (F, \omega) = (u_1, u_2)$. These parameters will systematically determined by a governing algorithm to ensure the accurate derivation of the trajectory for the formation, that we will describe further on. The motion equations are then reformulated to operate solely at the first order, resulting in:

$$\dot{x} = v_x \tag{2.3}$$
$$\dot{v_x} = u_1 \cos\theta \tag{2.4}$$
$$\dot{\theta} = u_2 \tag{2.5}$$

This formulation allows us to model the various UGVs considered in this work. Consequently, we can extract the triple (x, v_x, θ) from these equations. And subsequently, utilizing polar coordinates, we can extract y through polar coordinates as follows:

$$y = x\tan(\theta) \tag{2.6}$$

This model is observable in figure 2.1.

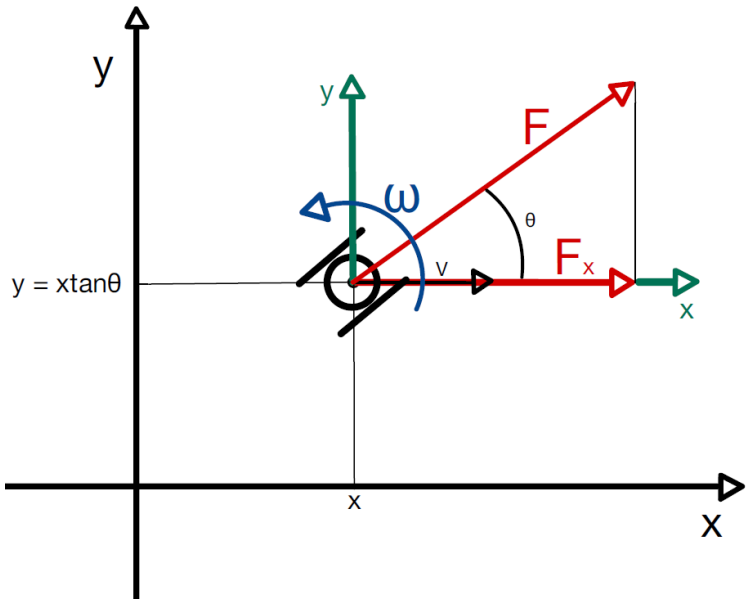

Figure 2.1: UGV model without disturbances

2.1.2 UGV model with disturbances

In this subsection, we extend the UGV model (described in subsection 2.1.1) to incorporate disturbances, into the second and third motion equations. The disturbances, denoted as w_1 and w_2, are extracted from a uniform distribution in the interval $(-0.2, 0.2)$.

The modified motion equations become:

$$\dot{x} = v_x \tag{2.7}$$
$$\dot{v_x} = u_1 \cos\theta + w_1 \tag{2.8}$$
$$\dot{\theta} = u_2 + w_2 \tag{2.9}$$

Here, w_1 represents the disturbance affecting the variation on time of velocity along x axis, acting as a form of friction, and w_2 represents the disturbance affecting the angular velocity.

Including disturbances in the model allows for a more realistic representation of UGV behavior, considering uncertainties and external factors that may influence the motion of the vehicle, such as frictional forces. This model is graphically shown in figure 2.2.

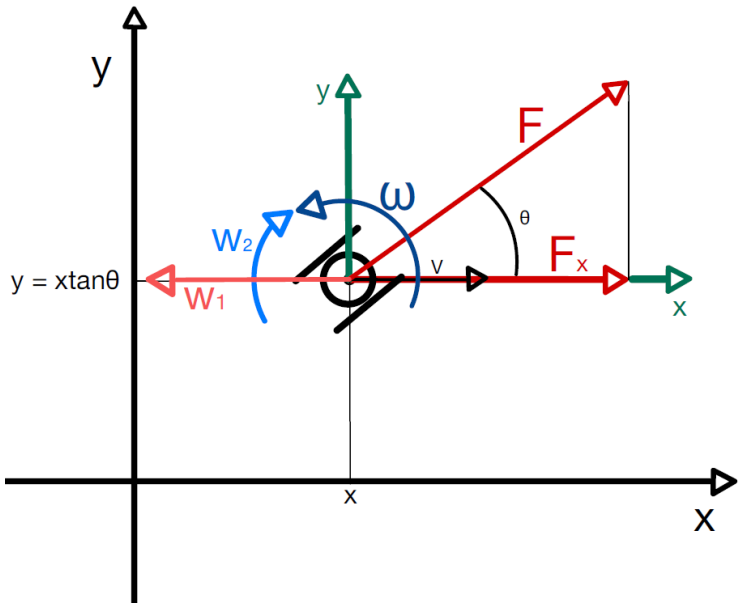

Figure 2.2: UGV model with disturbances

2.2 Nonlinear Model Predictive Control (NMPC)

Nonlinear Model Predictive Control (NMPC), as discussed in Chapter 1, is widely employed across diverse domains, primarily for computing optimal trajectories in various contexts. In this context, we will explore two distinct implementations for optimization problems characterized by a specific form.

2.2.1 NMPC with Pontryagin approach

In this study, we consider a dynamic system with specific conditions outlined by the following affine-in-the-input differential equation:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{2.10}$$

The system is subject to initial and terminal conditions:

$$x(t_0) = x_0, \qquad x(t_f) = x_f. \tag{2.11}$$

Here, t represents time, $x \in \mathbf{R}^n$ denotes the system variable, and $u(t) \in U$ stands for the control function, with U compact and convex subset of $\mathbf{R}^n$. Our primary goal is to find u^* that minimizes the cost function J, where J is a cost function defined on a Banach space as follows:

$$J : B(\cdot) \to \mathbf{R} \tag{2.12}$$

The optimization problem is a minimization problem. We assume that:

Assumption 1. *$f(x), g(x)$ are C^1.*

Assumption 2. *u is piece-wise continuous.*

In our Nonlinear Model Predictive Control (NMPC) framework, we formulate an optimization problem as a minimization problem, and we consider $[t_k, t_k + T_p]$ as a sampled discretized time interval from a larger time interval, where $t_k = T_s k$ with $k = 0,1,...$ (the different values for k permit to sample the entire time interval considered), T_s is called sampling time (with $T_s \leq T_p$) and T_p is called prediction time horizon. The minimization problem takes the form:

$$\text{minimize:} \qquad J(u; t_k, x_k) := \psi(x(T)) + \int_{t_k}^{t_k+T_p} L(t, x(t), u(t))dt. \tag{2.13}$$

Here, the components of $J(u; t_k, x_k)$ are defined as:

$$L(t, x(t), u(t)) := \tilde{x}^T Q \tilde{x} + u^T R u; \tag{2.14}$$

$$\psi(x(T)) := \tilde{x}^T(t_f) P \tilde{x}(t_f), \tag{2.15}$$

with Q, P, R being diagonal positive-definite matrices. Specifically, $\tilde{x} = \hat{x}_k - x_R$, where:

- $\tilde{x}$ denotes the error in the component variables of the dynamic system under consideration,
- $\hat{x}_k$ represents the estimated values of the component variables of the dynamic system,
- x_R corresponds to the real/reference values for the component variables of the dynamic system.

Moreover:

- Q includes positive values on the diagonal to mitigate errors in the component variables related to the dynamic system, when these values are high;
- R contains positive values on the diagonal that gives less considerations on actions of control inputs, if the values are high;
- P incorporates positive values on the diagonal to minimize errors in the component variables concerning the dynamic system at the end of the prediction horizon, when these values are high.

Remark 1. *So, the matrices in our particular work have the following forms:*

- $Q = \begin{pmatrix} Q_{11} & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & Q_{33} \end{pmatrix}$,
- $R = \begin{pmatrix} R_{11} & 0 \\ 0 & R_{22} \end{pmatrix}$,
- $P = \begin{pmatrix} P_{11} & 0 & 0 \\ 0 & P_{22} & 0 \\ 0 & 0 & P_{33} \end{pmatrix}$.

Remark 2. *Notice that the quadratic forms on J give the possibility to use this algorithm in non linear cases.*

NMPC is constituted by three fundamental phases:

- the prediction of the dynamic, in a certain time interval $[t_k, t_k + T_p]$;
- an optimization phase;
- The receding of horizon. In particular, at the k-th step, obtaining $u^* = [u_k^*, u_{k+1}^*, ...]^T \in [t_k, t_k + T_p]$, we apply to the system only the component u_k^*, discarding all the other ones.

Remark 3. *The appropriate selection of T_p and T_s values is critical:*

- *T_p should be sufficiently large for effective operation, but excessive values may compromise short-time tracking accuracy.*
- *T_s should be kept relatively small to maintain algorithmic efficiency, though excessively small values may lead to significant slowdowns.*

Remark 4. *In general, a tradeoff must be found between consumption and the path taken to reach the final objective.*

The complete optimization problem can be expressed as:

$$u^* = \arg\min_{u \in U} \int_{t_k}^{t_k+T_p} L(t, x(t), u(t))dt + \psi(x(T)). \tag{2.16}$$

s.t.

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ t_0 = t_k \\ t_f = t_k + T_p \\ x(t_0) = x_0 \end{cases}$$

with $x \in \mathbf{R}^n$ and $u \in U$. This represents a constrained problem.

Remark 5. *For simplicity, we write all the conditions*

$$\begin{cases} t_0 = t_k \\ t_f = t_k + T_p \\ x(t_0) = x_0 \end{cases}$$

as

$$\Psi = 0. \tag{2.17}$$

Then, we passed to the augmented function cost, incorporating Lagrange multipliers μ and costates λ, as follows:

$$\hat{J} = \psi(x(t_f)) + \mu^T \Psi + \int_{t_0}^{t_f} \{L(t, x, u) + \lambda^T (f(x) + g(x)u - \dot{x})\}dt, \tag{2.18}$$

leading to:

$$\arg\min_{u \in U} \hat{J}. \tag{2.19}$$

The necessary condition for optimality is then described, assuming that the sufficient condition, related to second derivative, is granted.

Necessary Condition for Optimality

To ascertain the necessary condition for optimality, the approach involves computing the Gâteaux derivative of $\hat{J}$ and subsequently setting $d\hat{J} = 0$. This derivative is employed due to the functional nature of the expression under consideration.

In this case we can calculate the Gâteaux derivative, using also the definition of Hamiltonian given by:

$$H(t, x, u, \lambda) := L(t, x, u) + \lambda^T(f(x) + g(x)u) \in \mathbf{R} \tag{2.20}$$

so we obtain:

$$\begin{aligned} d\hat{J} = (\mu^T \frac{\partial \Psi}{\partial t_f} + H(t_f, x(t_f), u(t_f), \lambda(t_f)) + \lambda_{t_f}^T \dot{x}_{t_f}))dt_f \\ +(\mu^T \frac{\partial \Psi}{\partial t_k} - H(t_k, x(t_k), u(t_k), \lambda(t_k)) - \lambda_{t_k}^T \dot{x}_{t_k}))dt_k \\ +\frac{\partial \psi}{\partial x_f}dx_f + \mu^T \frac{\partial \Psi}{\partial x_k}dx_k + \mu^T \frac{\partial \Psi}{\partial x_f}dx_f + \frac{\partial \psi}{\partial t_f}dt_f + \\ \int_{t_k}^{t_f} [\frac{\partial (H(t, x, u, \lambda) - \lambda^T \dot{x})}{\partial x}\delta x + \frac{\partial (H(t, x, u, \lambda) - \lambda^T \dot{x})}{\partial u}\delta u]dt \end{aligned} \tag{2.21}$$

using $\mu = 0$ and neglecting what remains constant along certain directions, we obtain:

$$\begin{aligned} d\hat{J} = \frac{\partial \psi}{\partial x_f}dx_f + \frac{\partial \psi}{\partial t_f}dt_f + H(t_f, x(t_f), u(t_f), \lambda(t_f))dt_f \\ -H(t_k, x(t_k), u(t_k), \lambda(t_k))dt_k + \int_{t_k}^{t_f} [\frac{\partial (H(t, x, u, \lambda))}{\partial x}\delta x + \frac{\partial (H(t, x, u, \lambda))}{\partial u}\delta u - \lambda^T \delta \dot{x}]dt \end{aligned} \tag{2.22}$$

Now, knowing that in 2.20 we can rewrite it as $H(t, x, u) = L(t, x, u) + \lambda^T \dot{x}$, using $\delta x = dx - \dot{x}dt$ and integrating by part $\int_{t_k}^{t_f} [-\lambda^T \delta \dot{x}]dt$ neglecting an invarint piece, we arrive to:

$$\begin{aligned} d\hat{J} = \frac{\partial \psi}{\partial t_f}dt_f + H(t_f, x(t_f), u(t_f), \lambda(t_f))dt_f + \frac{\partial \psi}{\partial x_f}dx_f - \lambda^T dx_f + \lambda^T \delta x_{t_0} + \\ \int_{t_k}^{t_f} [(\frac{\partial (H(t, x, u, \lambda))}{\partial x} + \dot{\lambda}^T)\delta x + \frac{\partial (H(t, x, u, \lambda))}{\partial u}\delta u]dt - L(t_k, x(t_k), u(t_k), \lambda(t_k))dt_k \end{aligned} \tag{2.23}$$

Posing to 0 each component of $d\hat{J}$, we arrive to the **Pontryagin Principle** (necessary condition of optimality):

- $\dfrac{\partial H(t,x,u,\lambda)}{\partial u} = 0$ which is the **law of optimal control**;
- $\dot{\lambda}(t) = -\dfrac{\partial H}{\partial x}$ which is called **Eulero-Lagrange equation**;

with the following conditions:

- $\lambda(t_f) = 2\tilde{x}^T(t_f)P$;
- $\lambda(t_k) = \mu(t_k)$.

So, we can reduce the numerical calculus to a **two points boundary value problem (TPBVP)** 2.2.1.
(For more considerations about Gâteaux derivative, read **Zaffaroni et al.** [**16**] and about all the calculus for the necessary condition of Pontryagin, see Section 2.8 of **Jr Arthur et al.** [**17**]).

2.2.2 More about necessary condition of Pontryagin Principle

Using similar notations we have seen before, consider the initial condition

$$x(t_0) = x_0, \tag{2.24}$$

and let

$$t \to x(t) = x(t; t_0, x_0, u) \tag{2.25}$$

the corresponding trajectory of a controlled system described by

$$\dot{x} = f(t, x, u), \qquad u(t) \in U \tag{2.26}$$

with t time, $x \in \mathbf{R}$ the state variable and f such that:

$$|f(t,x,u)| \leq C(1+|x|) \qquad \forall (t,x,u) \in [0,T] \times \mathbf{R}^n \times U. \tag{2.27}$$

So, consider the optimization problem

$$maximize: \qquad J(u; t_0, x_0) := \psi(x(T)) + \int_{t_0}^{T} L(t, x(t), u(t))dt \tag{2.28}$$

where ψ is a terminal payoff and $L(\cdot)$ the running cost. For initial data (t_0, x_0) (given), J should be maximized over measurable control functions $u : [t_0, T] \to U$.

Consider $t \to u^*(t)$ and $t \to x^*(t) = x(t; u_0, x_0, u^*)$ sequentially as the control optimal function and the optimal trajectory. We use the **Pontryagin Maximum Principle (PMP)** to have a set of necessary conditions for $u^*(\cdot)$ and $x^*(\cdot)$, considering initially the initial point given and the terminal point free.

Theorem 1 (PMP with free terminal point)**.** ***Bressan et al. [18].*** *Consider $t \to u^*(t)$ and $t \to x^*(t)$ the corresponding optimal control and optimal trajectory for 2.26 - 2.28. Define the vector $t \to \lambda(t)$ as the solution of the adjoint system*

$$\dot{\lambda}(t) = -\lambda(t)\frac{\partial f}{\partial x}(t, x^*(t), u^*(t)) + \frac{\partial L}{\partial x}(t, x^*(t), u^*(t)) \tag{2.29}$$

with terminal condition

$$\lambda(T) = \nabla\psi(x^*(T)). \tag{2.30}$$

Then, for almost every $t \in [t_0, T]$, it holds that:

$$\begin{aligned} &\lambda(t) \cdot f(t, x^*(t), u^*(t)) - L(t, x^*(t), u^*(t)) \\ &= \max_{u \in U}\{\lambda(t) \cdot f(t, x^*(t), u(t)) - L(t, x^*(t), u(t))\} \end{aligned} \tag{2.31}$$

So, the idea behind the theorem 1 to find optimal control can be achieved in two steps:

- solve 2.31, obtaining the optimal control u^* as function of t,x,λ:

$$u^*(t, x, \lambda) = \arg\max_{u \in U}\{\lambda f(t, x, \lambda) - L(t, x, \lambda)\}; \tag{2.32}$$

- solve the **two-point boundary value problem (TPBVP)** on time interval $[t_0, T]$:

$$\begin{cases} \dot{x}(t) = f(t, x(t), u^*(t, x, \lambda)) \\ \dot{\lambda}(t) = -\lambda(t)\dfrac{\partial f}{\partial x}(t, x, u^*(t, x, \lambda)) + \dfrac{\partial L}{\partial x}(t, x, u^*(t, x, \lambda)) \end{cases}$$

with initial conditions:

$$\begin{cases} x(t_0) = x_0 \\ \lambda(T) = \nabla\psi(x(T)) \end{cases}$$

An extension of Theorem 1 where initial and terminal points are constrained is the following:

Theorem 2 (PMP with initial and terminal points constrained)**. *Bressan et al. [18].*** *Let $t \to u^*(t)$ and $t \to x^*(t)$ an optimal bounded control function and the correspondent optimal trajectory for the problem*

$$maximize: \quad J := \phi(x(0)) + \psi(x(T)) - \int_0^T L(t, x(t), u(t))dt, \tag{2.33}$$

with dynamics 2.26 and with conditions

$$x(0) \in S_0 \qquad x(T) \in S_T \tag{2.34}$$

where f, L, ψ, ϕ are differentiable functions and $S_0, S_T \in \mathbf{R}^n$ are embedded manifolds, both C^1. Then:

- *there exists $t \to \lambda(t) = (\lambda_0, \lambda_1, ..., \lambda_n)(t)$ absolutely continuous, which never vanishes on $[0, T]$, with λ_0 constant, satisfying*

$$\dot{\lambda_j}(t) = -\sum_{j=1}^{n} \lambda_j(t) \frac{\partial f_j}{\partial x_i}(t, x^*(t), u^*(t)) + \lambda_0 \frac{\partial L}{\partial x_i}(t, x^*(t), u^*(t)) \qquad i = 1, ..., n \tag{2.35}$$

- *the initial and terminal values of p satisfy*

$$(\lambda_1, ..., \lambda_n)(0) = \lambda_0 \nabla \phi(x^*(0)) + n_0 \tag{2.36}$$
$$(\lambda_1, ..., \lambda_n)(T) = \lambda_0 \nabla \phi(x^*(T)) + n_T \tag{2.37}$$

  where n_0 and n_T are vectors orthogonal in the order to manifold S_0 at point $x^*(0)$ and manifold S_T at point $x^*(T)$.

- *The condition*

$$\sum_{i=1}^{n} \lambda_i(t) f_i(t, x^*(t), u^*(t)) - \lambda_0 L(t, x^*(t), u^*(t))$$
$$= \max_{u \in U} \{\sum_{i=1}^{n} \lambda_i(t) f_i(t, x^*(t), u(t)) - \lambda_0 L(t, x^*(t), u(t))\} \tag{2.38}$$

  holds for a.e. $t \in [0, T]$.

Definition 1. ***Bressan et al. [18].*** *We define as **Hamiltonian function** the following:*

$$H(t, x, u, \lambda) := \lambda f(t, x, u) - L(t, x, u). \tag{2.39}$$

Moreover, we define as ***reduced Hamiltonian*** *what follows:*

$$H(t,x,u) := \max_{\omega \in U}\{\lambda f(t,x,\omega) - L(t,x,\omega)\}; \tag{2.40}$$

where L is the running cost that you can find also here 2.28, f is the function related to 2.26, t is time, x and u are in the order the variable which describes our system and the control parameter.

Theorem 3 (PMP and concavity for the optimality)**.** ***Bressan et al. [18].*** *Using the settings for theorem 1, let $t \to u^*(t)$ a measurable function and $x^*(\cdot)$, $\lambda(\cdot)$ two absolutely continuous functions satisfying the following system:*

$$\begin{cases} \dot{x} = f(t,x,u^*(t)) \\ \dot{\lambda} = -\lambda(t)\dfrac{\partial f}{\partial x}(t,x,u^*(t)) + \dfrac{\partial L}{\partial x}(t,x,u^*(t)) \end{cases}$$

with initial and terminal conditions

$$\begin{cases} x(t_0) = x_0 \\ \lambda(T) = \nabla\psi(x(T)) \end{cases}$$

all with maximality condition 2.31. Suppose that the set U is convex and that $x \to H(t,x,\lambda(t))$, $x \to \psi(x)$ are concave.
Then $u^*(\cdot)$ is an optimal control, and $x^*(\cdot)$ is the corresponding optimal trajectory.

Proof. Consider $u : [t_0, T] \to U$ measurable control function. Then, using definition 1 and equation 2.26 we obtain:

$$\begin{aligned} J(u) - J(u^*) &= \psi(x(T)) - \psi(x^*(T)) - \int_{t_0}^{T}[L(t,x(t),u(t)) - L(t,x^*(t),u^*(t))]dt \\ &= \psi(x(T)) - \psi(x^*(T)) + \int_{t_0}^{T}[H(t,x(t),\lambda(t),u(t)) - \lambda(t)\dot{x}(t)] \\ &\quad -[H(t,x^*(t),\lambda(t),u^*(t)) - \lambda(t)\dot{x}^*(t)]dt \end{aligned} \tag{2.41}$$

u^* satisfies the maximality condition 2.31, so for a.e. $t \in [t_0, T]$ we have:

$$H(t,x^*(t),\lambda(t),u^*(t)) = H(t,x(t),\lambda(t)), \qquad H(t,x(t),\lambda(t),u(t)) \le H(t,x(t),\lambda(t)) \tag{2.42}$$

Using inequalities 2.42 inside the previous passages in 2.41 we obtain:

$$\begin{aligned} J(u) - J(u^*) &\le \psi(x(T)) - \psi(x^*(T)) \\ &+ \int_{t_0}^{T}\{[H(t,x(t)\lambda(t)) - \lambda(t)\dot{x}(t)] - [H(t,x^*(t),\lambda(t)) - \lambda(t)\dot{x}^*(t)]\}dt. \end{aligned} \tag{2.43}$$

Considering the map $x \to H(t, x, \lambda(t))$ differentiable, the concavity assumption implies

$$\begin{aligned} H(t, x(t), \lambda(t)) &\leq H(t, x^*(t), \lambda(t)) + \frac{\partial H}{\partial x}(t, x^*(t), \lambda(t))[x(t) - x^*(t)] \\ &= H(t, x^*(t), \lambda(t)) - \dot{\lambda}(t)[x(t) - x^*(t)] \end{aligned} \tag{2.44}$$

(All this is possible also for H not differentiable, following convex analysis).

Inserting 2.44 into 2.43 we obtain:

$$\begin{aligned} J(u) - J(u^*) &\leq \psi(x(T)) - \psi(x^*(T)) \\ &- \int_{t_0}^{T} \{\dot{\lambda}(t)[x(t) - x^*(t)] + \lambda(t)[\dot{x}(t) - \dot{x}^*(t)]\} dt \\ = \psi(x(T)) - \psi(x^*(T)) &- \{\lambda(T)[x(T) - x^*(T)] - \lambda(t_0)[x(t_0) - x^*(t_0)]\} \\ &\leq 0 \end{aligned} \tag{2.45}$$

So, with initial and terminal conditions in theorem 3 and concavity of ψ we arrive to the conclusion:

$$x(t_0) = x^*(t_0) = x_0, \quad \psi(x(T)) \leq \psi(x^*(T)) + \nabla\psi(x^*(T))[x(T) - x^*(T)]. \tag{2.46}$$

□

Remark 6. *Since the proof of the preceding theorem does not directly rely on the initial and terminal conditions, it remains valid even when assuming a fixed terminal condition.*

2.2.3 Application of the NMPC to the UGV model

In this work, we consider as dynamic system, the system in 2.10 for the UGVs, described in the subsection 2.1.1. So, considering

$$\dot{x} = v_x \tag{2.47}$$

$$\dot{v_x} = u_1 \cos\theta \tag{2.48}$$

$$\dot{\theta} = u_2 \tag{2.49}$$

where, for simplicity, we call v_x with v, and we have $u = (u_1, u_2)$, $\vec{x} = (x, v, \theta)$, $\tilde{x} = (x - x_{ref}, v - v_{ref}, \theta - \theta_{ref})$;
we have as hamiltonian function

$$H = \tilde{x}^T Q \tilde{x} + u^T R u + \lambda_1 v + \lambda_2 u_1 cos\theta + \lambda_3 u_2. \tag{2.50}$$

Now, we want to obtain the TPBVP.
First we consider the **law of optimal control**, obtaining:

$$\begin{cases} u_1^* = -(\dfrac{\lambda_2 cos\theta}{2R_{11}}) \\ u_2^* = -(\dfrac{\lambda_3}{2R_{22}}) \end{cases}$$

Then we consider the **Euler-Lagrange equation**, obtaining:

$$\begin{cases} \dot{\lambda_1}(t) = -2(x - x_{ref})Q_{11} \\ \dot{\lambda_2}(t) = -2(v - v_{ref})Q_{22} - \lambda_1 \\ \dot{\lambda_3}(t) = -2(\theta - \theta_{ref})Q_{33} + \lambda_2 u_1 sen\theta \end{cases}$$

So, we obtained our TPBVP, and now numerically we can find the optimal control and the corresponding optimal trajectory of single UGVs, fixing "boundary" conditions.

Note: As detailed in the Summary and also outlined in the Objectives section (see 1.2), the next step involves explaining a robust version of Nonlinear Model Predictive Control (NMPC) to address possible disturbances, as exemplified in the model with disturbances described in 2.1.2. This new extension of NMPC incorporates key concepts from Game Theory such as Nash equilibrium and Zero-Sum Games. Therefore, prior to delving into the mechanics of robust NMPC, it's essential to provide a foundational understanding of these theoretical constructs. Subsequently, we will elucidate how this particular typology of robust NMPC operates.

2.2.4 Basic Concepts about Game Theory

In this work, we want to consider two players called "Player A" and "Player B". They are constituted by:

- Two sets of strategies: A and B;
- Two payoff functions: $J^A : A \times B \to \mathbf{R}$ and $J^B : A \times B \to \mathbf{R}$.

So, if the first player chooses the strategy $a \in A$ and the second player chooses the strategy $b \in B$, the payoffs are described, in order, by $J^A(a, b)$ and $J^B(a, b)$.

Definition 2. ***Bressan et al. [18].*** *A game is called* ***zero-sum game*** *if for every pair of strategies* (a, b) *we have:*

$$J^A(a, b) + J^B(a, b) = 0. \tag{2.51}$$

So, a zero-sum game is determined by one single payoff function $J = J^A = -J^B$.

Assumption 3. ***Bressan et al. [18].*** *A and B are compact metric spaces and J^A, J^B are continuous functions.*

Now we want to define what it is a **best reply map** for two players A and B.

Definition 3. ***Bressan et al. [18].*** *For a given choice $b \in B$ by Player B the set of best possible replies by player A is given by:*

$$R^A(b) := \{a \in A; J^A(a,b) = \max_{\omega \in A} J^A(\omega, b)\} \tag{2.52}$$

Similarly, for player B, for a given choice $a \in A$ we have:

$$R^B(a) := \{b \in B; J^B(a,b) = \max_{\omega \in B} J^B(a, \omega)\} \tag{2.53}$$

Nash equilibria

In general, it is difficult to speak of "optimal solution" in a game. An optimal solution for a player can be a bad solution for the other one. So, there are different concepts of equilibrium. In particular, we want to deal with a particular case: **Nash equilibrium**.
In this case, this typology of equilibrium is used to model a symmetric situation where players do not cooperate and they do not share information about their strategies.

Definition 4. ***Bressan et al. [18].*** *Given a game with 2 players A and B, a pair of strategies (a^*, b^*) is called **Nash equilibrium** for every $a \in A$ and $b \in B$, if it follows that:*

$$J^A(a, b^*) \leq J^A(a^*, b^*), \quad J^B(a^*, b) \leq J^B(a^*, b^*) \tag{2.54}$$

In essence, no player is capable of augmenting their payoff through steadfastly altering their strategy unilaterally, provided that the counterpart adheres to the equilibrium strategy.

Observe that, with reference Definition 3 for best reply maps, a pair of strategies (a^*, b^*) is a Nash equilibrium if and only if it is a fixed point of the best reply map:

$$a^* \in R^A(b^*), \qquad b^* \in R^B(a^*) \tag{2.55}$$

Example 1. ***Bressan et al. [18].*** *In figure 2.3, player A determines the horizontal coordinate, while player B determines the vertical coordinate. The payoff function of A achieves its global maximum at point P, while that of B reaches its global maximum at point Q. The pair of strategies (a_{Nash}, b_{Nash}) represents a Nash equilibrium. It is noteworthy that at this juncture, the level curve of A exhibits a horizontal tangent, while the level curve of B displays a vertical tangent.*

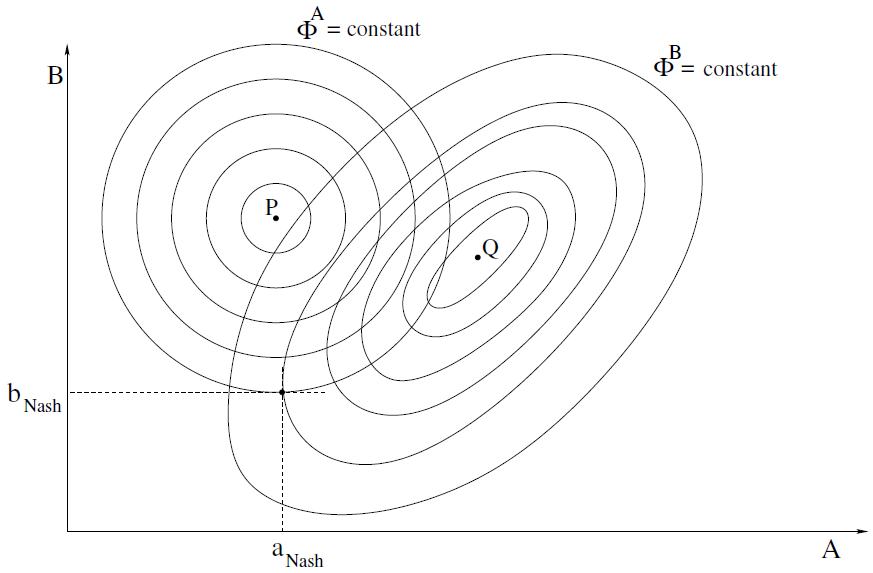

Figure 2.3: Nash equilibrium

We want to show in order, with examples, that:

- a Nash equilibrium may not exist;
- the Nash equilibrium can be not the unique;
- different Nash equilibrium can give different payoffs to players.

Example 2. ***Bressan et al. [18].*** *Assuming that each participant selects a coin, opting to reveal either heads or tails, the outcome is determined by whether the two coins match. In the event of a match, Player A gains 1 while Player B incurs a loss of 1. Conversely, if the coins do not match, Player B earns 1, and Player A suffers a 1 loss. This scenario constitutes a zero-sum game, as outlined in figure 2.4. Upon thorough examination, it is evident that the game lacks any Nash equilibrium solution.*

Example 3. ***Osborne Martin et al. [19]*** *Consider the "Battle of the Sexes" game with two players, Player A and Player B. Both must choose between going to the prize fight (PF) or the ballet (B). The payoff matrix is showed in figure 2.5.*

The Nash equilibrium are:

- (PF, PF)*: If both choose to go to the prize fight, neither has an incentive to deviate, as both prefer this outcome to both going to the ballet.;*

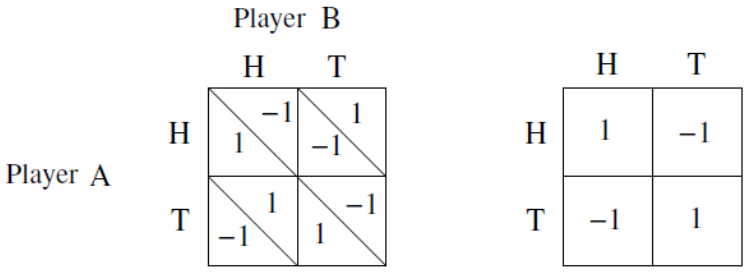

Figure 2.4: Example where Nash equilibrium does not exist

	Prize Fight	Ballet
Prize Fight	10,7	0,0
Ballet	0,0	7,10

Figure 2.5: Example where Nash equilibrium is not unique

- (B, B)*: Similarly, if both choose to go to the ballet, neither has an incentive to deviate.*

Both of these outcomes are Nash equilibria. In this case, there are two stable solutions, each representing a common preference for both players.

Moreover, in this example it can be noticed that different equilibrium gives different payoff to players:

- (PF, PF)*: If both choose to go to the prize fight, the payoff for the first player is 10 and for the second is 7.*
- (B, B)*: this is the case in which both choose to go to the ballet, and the payoff for the first player is 7 and for the second is 10.*

Now, we want to describe a particular situation in which it is always possible to find a Nash equilibrium.

Theorem 4 (Existence of Nash equilibrium)**.** ***Bressan et al. [18].*** *Consider a non-cooperative game, where A and B are compact and convex subsets of $\mathbf{R}^n$, the payoff functions J^A and J^B are continuous and where*

- $a \to J^A(a, b)$ *is a concave function of a, for each $b \in B$;*

- $b \to J^B(a,b)$ *is a concave function of* b*, for each* $a \in A$.

Then the game admits a Nash equilibrium.

Before to demonstrate this theorem, we want to specify that non-cooperative games are used in situations where there are competitions between the players of the game **Jhon et al.** [**20**].
Moreover, zero-sum games are types of non-cooperative games **Khalid et al.** [**21**].

Proof. (idea) Consider best reply maps defined in Definition 3. We demonstrate theorem 4 dividing the explanation in three parts:

- Since B is compact and J^B is continuous, then the function $a \to m(a) := \max_{b \in B} J^B(a,b)$ is continuous too.
  Defining the set $graph(R^B) = \{(a,b) : b \in R^B(a)\} = \{(a,b) : J^B(a,b) = m(a)\}$, it is closed. So, the multifunction $a \to R^B(a)$, with $R^B(a)$ subset of B, is upper semicontinuous. (For more about these concepts view **Bressan et al.** [**18**] and also **Baggs Ivan et al.** [**22**] and **Ursescu Corneliu et al.** [**23**]).

- We want to prove that each set $R^B(a)$ subset of B is convex. So, let $b_1, b_2 \in R^B(a)$, and we have $J^B(a,b_1) = J^B(a,b_2) = m(a)$. Take $\theta \in [0,1]$. Using concavity of $b \to J^B(a,b)$ we obtain: $m(a) \geq J^B(a, \theta b_1 + (1-\theta)b_2) \geq \theta J^B(a,b_1) + (1-\theta)J^B(a,b_2) = m(a)$.
  Since B is convex, we have that $\theta b_1 + (1-\theta)b_2 \in B$. Hence $\theta b_1 + (1-\theta)b_2 \in R^B(a)$ proving the convexity.

- From previous facts, $a \to R^B(a)$ is upper semicontinuous with compact and convex values. Similarly, we have the same for $b \to R^A(b)$.
  Now, we consider $(a,b) \to R^A(b) \times R^B(a)$ subset of $A \times B$, on the product space $A \times B$.
  Applying Kakutani's fixed point theorem, that you can find for example in **Yoo et al.** [**24**], we obtain a pair of strategies $(a^*, b^*) \in (R^A(b^*), R^B(a^*))$ which is a Nash equilibrium solution.

□

More about Zero-Sum Games

Considering a game with two Players A and B, we observed in definition 2 what is a **zero-sum game**.
As we know it is possible to describe it with the usage of a single function

$$J : A \times B \to \mathbf{R} \tag{2.56}$$

where if we consider any (a, b) with $a \in A$ and $b \in B$, we can think $J(a, b)$ the amount that B pays to A, if these strategies are chosen.
The goal of Player A is to maximize this payoff, while Player B tries to minimizes it.

We need to assume that:

Assumption 4. ***Bressan et al. [18].*** *A and B are compact metric spaces and the function $J : A \times B \to \mathbf{R}$ is continuous.*

So, we have that

$$b \to \max J_{a\in A}(a, b), \qquad a \to \min J_{b\in B}(a, b) \tag{2.57}$$

are both continuous.

Moreover, in a symmetric situation Players have to choose their choice without a a priori knowledge of the choice of the other Player. But, there are also cases in which one Player can have this advantage of information, as in the following cases.

- Player B chooses $b \in B$, and thus, Player A makes a choice based on the selected strategy of Player B. This advantage in knowledge implies that Player A can determine the best reply, denoted as $\alpha(b) \in A$, satisfying

$$J(\alpha(b), b) = \max_{a\in A} J(a, b). \tag{2.58}$$

  In this way, the minimum payment that the second Player can achieve is:

$$V^+ := \min_{b\in B} J(\alpha(b), b) = \min_{b\in B} \max_{a\in A} J(a, b). \tag{2.59}$$

- Player A chooses $a \in A$, and Player B makes a choice based on the selected strategy of Player A. So, in this case, Player B can determine the best reply, denoted as $\beta(a) \in B$, such that

$$J(a, \beta(a)) = \min_{b\in B} J(a, b). \tag{2.60}$$

  In this way, the maximum payment that the second Player can choose is:

$$V^- := \max_{a\in A} J(a, \beta(a)) = \max_{a\in A} \min_{b\in B} J(a, b). \tag{2.61}$$

Lemma 1. ***Bressan et al. [18].*** *With the previous settings, it holds that:*

$$V^- := \max_{a\in A} \min_{b\in B} J(a, b) \leq \min_{b\in B} \max_{a\in A} J(a, b) =: V^+ \tag{2.62}$$

Proof. Consider the map $a \to \beta(a)$, possibly discontinuous (i.e.: best reply map for Player B). Since (for the possible discontinuity we do not write "max")

$$V^- := \sup_{a \in A} J(a, \beta(a)), \tag{2.63}$$

for any $\epsilon > 0$, there exists $a_\epsilon \in A$ such that:

$$J(a_\epsilon, \beta(a_\epsilon)) > V^- - \epsilon, \tag{2.64}$$

And this fact implies that:

$$V^+ := \min_{b \in B} \max_{a \in A} J(a, b) \geq \min_{b \in B} J(a_\epsilon, b) = J(a_\epsilon, \beta(a_\epsilon)) \geq V^- - \epsilon. \tag{2.65}$$

For the arbitrary of ϵ, this proves the lemma. □

Definition 5. ***Bressan et al. [18].*** *It is called* ***value of the game****, with the previous settings, the situation in which* $V := V^- = V^+$.

Moreover, always under previous settings:

Definition 6. ***Bressan et al. [18].*** *If there exist* $a^* \in A$ *and* $b^* \in B$ *strategies, such that:*

$$\min_{b \in B} J(a^*, b) = J(a^*, b^*) = \max_{a \in A} J(a, b^*), \tag{2.66}$$

the pair (a^*, b^*) *is a* ***saddle point*** *of the game.*

If we nominate V the common values of quantities in definition 6, it holds that:

- if A considers strategy a^*, he surely receives no less than V;
- if B considers strategy b^*, he surely pays no more than V.

Remark 7. *In a zero-sum game the concept of saddle-node point is the same of Nash equilibrium.*

Theorem 5 (connection between value of game and saddle point)**.** ***Bressan et al. [18].*** *Consider assumption 4. The zero-sum game in 2.56 has a value* V *if and only if there is the existence of the saddle point* (a^*, b^*)*. In such case, one has*

$$V = V^- = V^+ = J(a^*, b^*). \tag{2.67}$$

Proof. We divide the proof in two parts:

- Suppose the existence of (a^*, b^*). This implies that:

$$\begin{aligned} V^- &:= \max_{a\in A}\min_{b\in B} J(a,b) \geq \min_{b\in B} J(a^*, b) \\ &= \max_{a\in A} J(a, b^*) \geq \min_{b\in B}\max_{a\in A} J(a,b) =: V^+. \end{aligned} \tag{2.68}$$

  By lemma 1 we arrive to $V := V^+ = V^-$, which is for definition 5 a value.

- Now, consider $V := V^- = V^+$. Let $a \to \beta(a)$ the best reply map for the second Player. For each ϵ, we select $a_\epsilon \in A$ for which 2.64 holds. For the compactness of A and B, consider a subsequence $\epsilon_n \to 0$ for which the corresponding strategies converge, called

$$a_{\epsilon_n} \to a^*, \qquad \beta(a_{\epsilon_n}) \to b^*. \tag{2.69}$$

  We want to prove that (a^*, b^*) is a saddle point.
  The continuity of J yields

$$J(a^*, b^*) = \lim_{n\to\infty} J(a_{\epsilon_n}, \beta(a_{\epsilon_n})). \tag{2.70}$$

  Since

$$V^- - \epsilon_n < J(a_{\epsilon_n}, \beta(a_{\epsilon_n})) \leq \sup_{a\in A} J(a, \beta(a)) = V^+ \tag{2.71}$$

  and holding $\epsilon \to 0$ we arrive to $V^- = V^+$ and so we conclude the proof.

□

Remark 8. *For a zero-sum game, if the Nash equilibrium exists, then all Nash equilibrium give the same payoff:*

$$V = \min_{b\in B}\max_{a\in A} J(a,b) = \max_{a\in A}\min_{b\in B} J(a,b). \tag{2.72}$$

Applying theorem 4 on a zero-sum game, we obtain:

Corollary 1 (Saddle point existence)**. *Bressan et al. [18].*** *Consider a zero-sum game with the assumption 4. Moreover, consider A and B convex and*

- *$a \to J(a,b)$ concave function of a for each $b \in B$*
- *$b \to J(a,b)$ convex function of b for each $a \in A$*

Then, the game has a Nash equilibrium (i.e. a saddle point).

2.2.5 Robust NMPC

The robust Nonlinear Model Predictive Control (NMPC) framework proposed in this study consistently employs the Pontryagin approach **Pagone et al. [15]**. The mathematical procedures involved closely mirror those outlined in subsection 2.2.1. However, the current formulation addresses a min/max optimization problem, specifically within the context of a sum-zero game.

In this work, we identify two key players:

- u representing the control input parameter;
- w signifying the disturbance parameter.

To elaborate further, it is stipulated that $u \in U \subseteq \mathbf{R}^{n_u}$ and $w \in W \subseteq \mathbf{R}^{n_w}$, with the assumption that W is defined as the hyperrectangle:

$$W = \{w : \forall i, |w_i| \leq \bar{w}_i\} \tag{2.73}$$

Unlike the dynamics outlined in 2.10, the present dynamic system is expressed as:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) + l(x(t))\mu(t) \tag{2.74}$$

This represents the "ideal" dynamic. We separate it from the "nominal" dynamic, which contains the disturbance, and it is represented as follows:

$$\dot{\hat{x}}(t) = f(\hat{x}(t)) + g(\hat{x}(t))u(t) + l(\hat{x}(t))\hat{\mu}(t) \tag{2.75}$$

It is crucial to note that in this scenario, μ serves a different purpose and is not a Lagrange multiplier, as denoted in the subsequent assumption.

Assumption 5. *In the specific, μ and $\hat{\mu}$ are time-invariant parameters vector of the system. And $w := \hat{\mu} - \mu$ is an unknown unbounded vector.*

Remark 9. *In subsection 2.2.1, in the NMPC with Pontryagin, the "prediction" dynamic coincides with the "real" dynamic.*

Given an arbitrary $\gamma > 0$, the constrained optimization problem is formulated as follows:

$$(u^*, w^*) = \arg\min_{u \in U} \max_{w \in W} \int_{t_k}^{t_k + T_p} (\|\tilde{x}(t)\|_R^2 - \gamma \|w(t)\|_2^2 + \|u(t)\|_Q^2) dt + \|\tilde{x}(t_k + T_p)\|_P^2. \tag{2.76}$$

s.t.

$$\begin{cases} \dot{\hat{x}}(t) = f(\hat{x}(t)) + g(\hat{x}(t))u(t) + l(\hat{x}(t))\mu(t) \\ \hat{x}(t_k) = x(t_k) \\ \hat{x}(t) \in X, u(t) \in U, \forall t \in [t_k, t_k + T_p] \\ u(t) \in KC([t_k, t_k + T_p]). \end{cases}$$

Here, $\tilde{x}(t) = x_r - \hat{x}(t)$.

Remark 10. *All assumptions in 2.2.1 remain valid and U, W and X are compact subsets.*

Remark 11. *In this context, also $l(x(t))$ is supposed to be C^1.*

Remark 12. *The notation used in 2.76 for the quadratic forms is the same as in 2.14-2.15, as follows*

$$\| \cdot \|_A^2 = (\cdot)^T A(\cdot) \tag{2.77}$$

with A positive-definite matrix.

Remark 13. *Notice that if the parameter γ is properly "low", the action of disturbance is more perceptible and so the robust algorithm can dampen it better.*

Moreover:

Lemma 2. *The pair (u^*, w^*) is the solution of the MIN-MAX problem if and only if the following conditions are simultaneously verified:*

$$u^* = \arg \min_{u(t) \in U} J_w(u, w), \tag{2.78}$$

$$w^* = \arg \min_{w(t) \in W} -J_w(u, w) = \arg \max_{w(t) \in W} J_w(u, w), \tag{2.79}$$

for all $t \in [t_k, t_k + T_p]$.

Definition 7. *Consider 2.75, 2.76, 2.78 and 2.79. The pair of Hamiltonians $H^{(u)}$, $H^{(w)} : C^1(X \times X \times U \times W) \to \mathbf{R}$ are defined as:*

- $H^{(u)} = \|\tilde{x}(t)\|_R^2 - \gamma \|w(t)\|_2^2 + \|u(t)\|_Q^2$
  $+ \lambda^{(u)T}[f(x(t)) + g(x(t))u(t) + l(x(t))(\hat{\mu} + w(t))]$
- $H^{(w)} = -\|\tilde{x}(t)\|_R^2 + \gamma \|w(t)\|_2^2 - \|u(t)\|_Q^2$
  $+ \lambda^{(u)T}[f(x(t)) + g(x(t))u(t) + l(x(t))(\hat{\mu} + w(t))]$

using $\mu = \hat{\mu} + w(t)$ *and* $\lambda^{(u)}, \lambda^{(w)} \in \mathbf{R}^{n_x}$ *covectors of the minimization and maximization problem, respectively.*

Fundamental Note 1. *Upon synthesizing lemma 2, theorem 5, and remark 7, a significant observation emerges: the existence of a Nash equilibrium, denoted as a saddle-node point, is contingent upon the condition that* $\min_u \max_w J(u, w) = \max_w \min_u J(u, w)$*. However, attaining this equilibrium is not universally achievable. Such equilibrium is more likely in cases where the cost function* J *exhibits separability, but this condition cannot be guaranteed in the majority of instances.*

The focus of this section is directed towards the utilization of Hamiltonians. Hamiltonians serve as a robust approximation for the cost function J*. Consequently, the objective is to identify a saddle-node point for Hamiltonians, representing an unidentified equilibrium in the context of the initial problem 2.76. This equilibrium, once found, will serve as an optimal solution for the optimization problem at hand.*

You can observe in 2.6 an example of possible convex Hamiltonian (dashed curve) which approximates a non linear cost function. You can imagine to find an equilibrium for the cost function which can be not optimal globally, using the Hamiltonian.

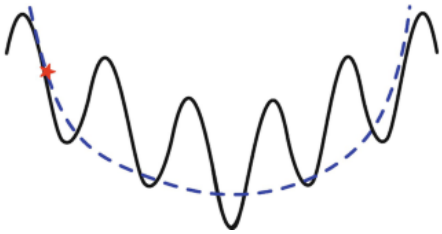

Figure 2.6: Approximation with Hamiltonian

Necessary condition for Optimality

Applying what is described from theorem 1 and similarly what we did in subsection 2.2.1, the necessary conditions for optimizaton are given by:

$$H^{(u)}(x^*, u^*, w^*, \lambda^{*(u)}, \lambda^{*(w)}) = \min_{u \in U} H^{(u)}(x^*, u, w^*, \lambda^{*(u)}, \lambda^{*(w)}), \tag{2.80}$$

$$H^{(w)}(x^*, u^*, w^*, \lambda^{*(u)}, \lambda^{*(w)}) = \min_{w \in W} H^{(w)}(x^*, u^*, w, \lambda^{*(u)}, \lambda^{*(w)}) \tag{2.81}$$

$$\dot{\lambda}^{(u)}(t) = -\nabla_x H^{(u)} \tag{2.82}$$

$$\dot{\lambda}^{(w)}(t) = -\nabla_x H^{(w)} \tag{2.83}$$

$$\lambda^{(u)}(t_k + T_p) = 2\tilde{x}^T(t_k + T_p)P \tag{2.84}$$

$$\lambda^{(w)}(t_k + T_p) = -2\tilde{x}^T(t_k + T_p)P \tag{2.85}$$

Theorem 6. ***Pagone et al. [15].*** *Considering min-max necessary conditions described from 2.80 to 2.85, then* $\lambda^{(u)}$ *and* $\lambda^{(w)}$ *are related as follows:*

$$\lambda^{(w)}(t) = -\lambda^{(u)}(t), \qquad \forall t \in [t_k, t_k + T_p]. \tag{2.86}$$

Proof. For simplicity, we consider the scalar case, when $x, \lambda^{(u)}, \lambda^{(w)} \in \mathbf{R}$. Anyway, it remains all valid also in a multidimensional case.
From 2.84 and 2.85, we have that $\lambda^{(w)}(t_k + T_p) = -\lambda^{(u)}(t_k + T_p)$.
If we put 2.75 into 2.82 and 2.83, we obtain:

$$\dot{\lambda}^{(u)}(t) = -(\frac{\partial f}{\partial x}(x(t)) + \frac{\partial g}{\partial x}(x(t))u(t) + \frac{\partial l}{\partial x}(x(t)\mu(t)))\lambda^{(u)}(t) \tag{2.87}$$

and

$$\dot{\lambda}^{(w)}(t) = -(\frac{\partial f}{\partial x}(x(t)) + \frac{\partial g}{\partial x}(x(t))u(t) + \frac{\partial l}{\partial x}(x(t)\mu(t)))\lambda^{(w)}(t). \tag{2.88}$$

They represent a set of backward first-order differential equation integration within the TPVBP.
Let

$$\rho(t) = \frac{\lambda^{(u)}(t)}{\lambda^{(w)}(t)} \tag{2.89}$$

a new auxiliary variable, whose final condition, from 2.84 and 2.84, is $\rho(t_k+T_p) = -1$. Taking the time derivative of 2.89, we have

$$\dot{\rho}(t) = \frac{\dot{\lambda}^{(u)}(t)}{\lambda^{(w)}(t)} - \frac{\lambda^{(u)}(t)\dot{\lambda}^{(w)}(t)}{\lambda^{(w)^2}(t)}. \tag{2.90}$$

Upon substituting 2.87-2.88 into 2.90, and defining

$$\xi(t) = -(\frac{\partial f}{\partial x}(x(t)) + \frac{\partial g}{\partial x}(x(t))u(t) + \frac{\partial l}{\partial x}(x(t)\mu(t))) \tag{2.91}$$

one has that:

$$\dot{\rho}(t) = \frac{\xi(t)\lambda^{(u)}(t) - \xi(t)\lambda^{(u)}(t)}{\lambda^{(w)}(t)} = 0. \tag{2.92}$$

Using remark 10, from solution properties of Lipschitz-continuous differential equations, we have that

$$\rho(t) = \rho(t_k + T_p) - \int_{t_k+T_p}^{\tau} \dot{\rho}(\tau)d\tau. \tag{2.93}$$

Now, taking account 2.92 and the fact that $\rho(t_k+T_p) = -1$, we lead to the following expression

$$\rho(t) = \frac{\lambda^{(u)}(t)}{\lambda^{(w)}(t)} = -1, \qquad \forall t \in [t_k, t_k + T_p] \tag{2.94}$$

which yields the statement. □

Remark 14. *The result obtained in theorem 6 yields the following observations:*

- *a joint Hamiltonian H and a common covector λ of the MIN-MAX problem can be defined. We can pick H as $H^{(u)}$ and $\lambda(t) = \lambda^{(u)}(t) = -\lambda^{(w)}(t)$ (or taking H as $H^{(w)}$ and inverting the sign of covectors). Hence, we have*

$$u^*(t) = \arg\min_{u(t)\in U} H(x(t), \lambda(t), u(t), w(t)) \tag{2.95}$$

and

$$w^*(t) = \arg\min_{w(t)\in W} H(x(t), -\lambda(t), u(t), w(t)) \tag{2.96}$$

$\forall t \in [t_k, t_k + T_p]$*;*

- *There exists an equilibrium which coincides with the joint Hamiltonian saddle point. So there is a pair (u^*, w^*) such that*

$$H(u^*, w) \leq H(u^*, w^*) \leq H(u, w^*); \tag{2.97}$$

- *the previous facts imply that it is sufficient to find only the solution of one TPBVP. Covectors of both problems are negative of each other and they can be obtained from the first-order necessary conditions from the Hamiltonian.*

2.2.6 Application of the robust NMPC to the UGV model

In this study, we explore the application of robust Nonlinear Model Predictive Control (NMPC) to the Unmanned Ground Vehicle (UGV) model, similar to the approach discussed in subsection 2.2.3. However, we consider two scenarios: an ideal scenario without disturbances and a nominal scenario with disturbances.
The ideal scenario is described by the following system dynamics:

$$\dot{x} = v \tag{2.98}$$
$$\dot{v} = u_1 \cos\theta \tag{2.99}$$
$$\dot{\theta} = u_2 \tag{2.100}$$

where $u = (u_1, u_2)$ represents the control input, and $\vec{x} = (x, v, \theta)$ denotes the state variables extracted from the system 2.98.
The nominal scenario introduces disturbances:

$$\dot{x} = v \tag{2.101}$$
$$\dot{v} = u_1 \cos\theta + w_1 \tag{2.102}$$
$$\dot{\theta} = u_2 + w_2 \tag{2.103}$$

where $w = (w_1, w_2)$ represents the disturbance vector.
The Hamiltonian function is given by:

$$H = \tilde{\vec{x}}^T Q \tilde{\vec{x}} + u^T R u - \gamma w^T w + \lambda_1(v + w_1) + \lambda_2(u_1 cos\theta + w_2) + \lambda_3(u_2 + w_3). \tag{2.104}$$

where $\tilde{\vec{x}} = (x - x_{ref}, v - v_{ref}, \theta - \theta_{ref})$, and γ is a positive constant.

The resulting Two-Point Boundary Value Problem (TPBVP) is then solved.
The **law of optimal control** for the input is provided by:

$$\begin{cases} u_1^* = -(\dfrac{\lambda_2 cos\theta}{2R_{11}}) \\ u_2^* = -(\dfrac{\lambda_3}{2R_{22}}). \end{cases}$$

The **law of optimal control** for disturbances is given by:

$$\begin{cases} w_1^* = -\lambda_2^{(w)}/(2\gamma) \\ w_2^* = -\lambda_3^{(w)}/(2\gamma) \end{cases}$$

Next, we determine the **Euler-Lagrange equation** for the control input since $\lambda^{(u)} = -\lambda^{(w)}$:

$$\begin{cases} \dot{\lambda_1}(t) = -2(x - x_{ref})Q_{11} \\ \dot{\lambda_2}(t) = -2(v - v_{ref})Q_{22} - \lambda_1 \\ \dot{\lambda_3}(t) = -2(\theta - \theta_{ref})Q_{33} + \lambda_2 u_1 sen\theta \end{cases}$$

Now we can simulate numerically the robust control, fixing "boundary" conditions.

Chapter 3

Simulations and Results

In this chapter, an in-depth exploration of the simulations conducted for this thesis is presented.
The software tools utilized for these simulations were MATLAB and Simulink.

3.1 Choice of Topology

This section focuses on the selection of network topology for the study.

As expounded in Chapter 2, specific motion equations tailored for various Unmanned Ground Vehicles (UGVs) are at our disposal. The six UGVs considered in this study collectively form a dynamic network. Each UGV, as previously outlined, is characterized by its motion equation and has been implemented within a subsystem, as depicted in Figure 3.1, where the "interpreted MATLAB function" encapsulates the pertinent motion equations. Consequently, each subsystem serves to represent an individual drone.

Here, the UGVs exhibit motion with random trajectories.
Our initial concept involved creating a topology that could vary over time. Subsequently, in another subsystem positioned hierarchically above the one illustrated in Figure 3.1 and represented in Figure 3.2, the time-varying coordinates of UGVs were extracted.

As evident in Figure 3.2, each rectangle contains the content depicted in Figure 3.1. The output of this subsystem consists of the extracted coordinates, which are combined with the coordinates of other UGVs. It is noteworthy that the transition from the coordinates of motion equations to Cartesian coordinates for simplicity occurred earlier.

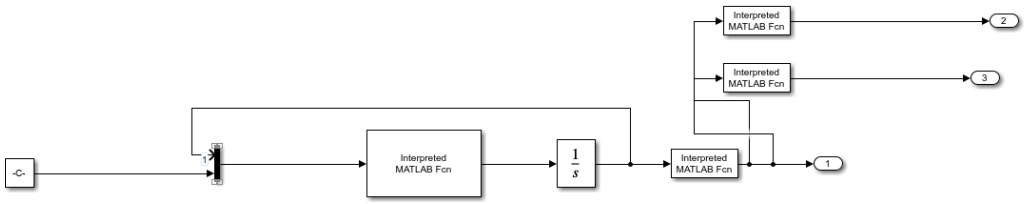

Figure 3.1: Simulink - single UGV

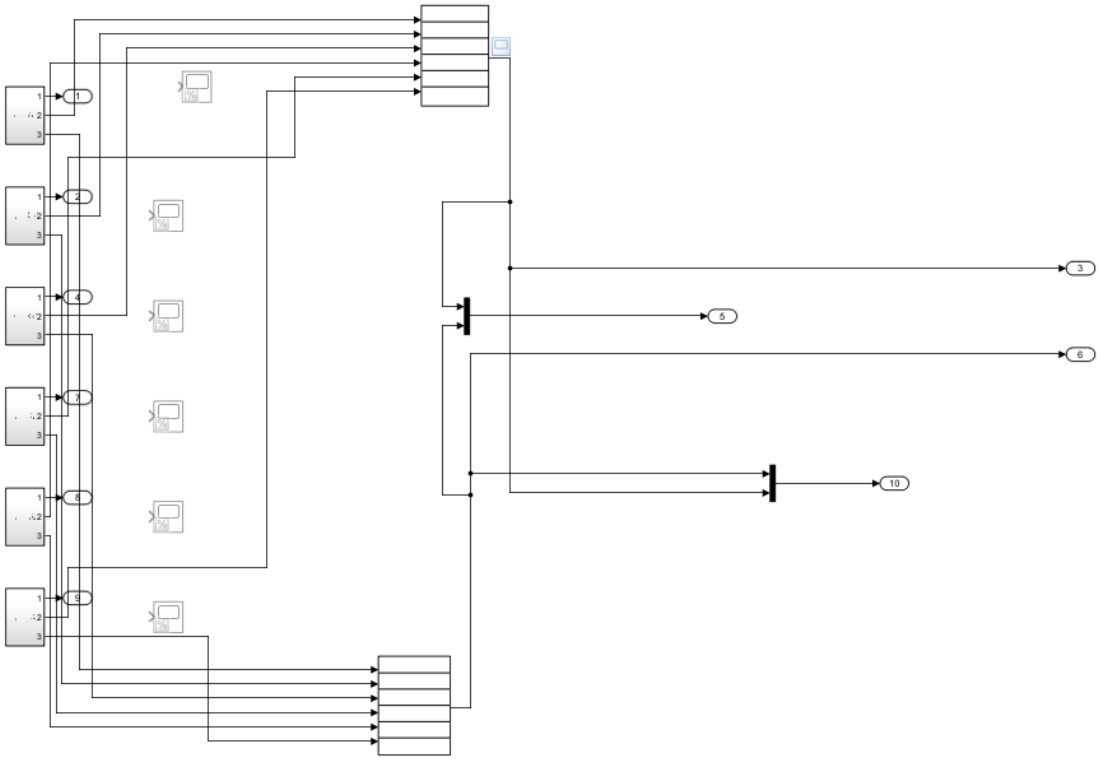

Figure 3.2: Simulink - Topology

The concept underlying the creation of the topology was implemented through the following code:

```
function [A_f] = adj_matrix(x_in,y_in)
n=length(x_in);
A=zeros(n,n);
radius=10;
for i=1:n
    for j=i+1:n
        if sqrt((x_in(i)-x_in(j))^2 +(y_in(i)-y_in(j))^2) <= radius
            A(i,j)=1;
            A(j,i)=1;
        end
    end
end




dist_matrix = zeros(n, n);
for i=1:n
    for j=1:n
        dist_matrix(i, j) = sqrt((x_in(i) - x_in(j))^2 + (y_in(i) -
    y_in(j))^2);
    end
end

g = graph(A);
isolatedNodes = find(degree(g) == 0)


if numel(isolatedNodes)==1
   for i=1:n
        A=single_node_fixing(i,A,dist_matrix,isolatedNodes);
    end

end

A_f=A;


end
```

The code establishes connections between Unmanned Ground Vehicles (UGVs) based on a specified distance criterion. If the distance between two UGVs is less than or equal to the defined radius, a connection is established in the adjacency

matrix. The code also addresses the scenario of isolated nodes, ensuring connectivity by connecting a single isolated node to its nearest neighbour. The resulting adjacency matrix, denoted as A_f, reflects the established connections in the network topology.

Examples of the varying topology in time of UGVs are illustrated in Figure 3.3, 3.4, 3.5.

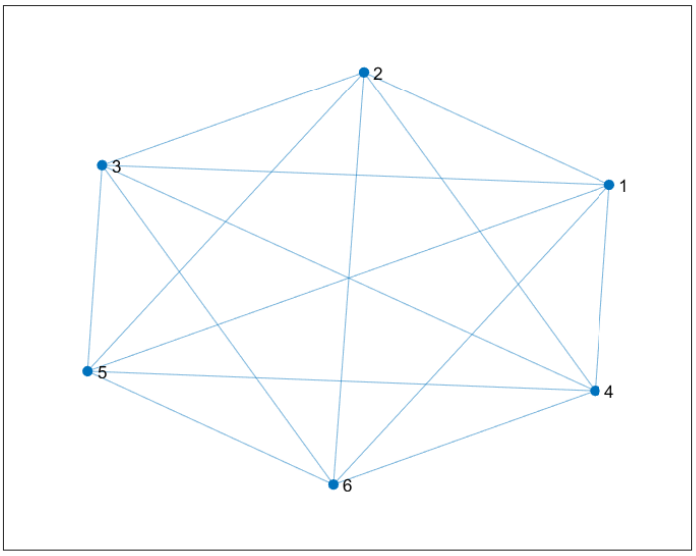

Figure 3.3: Example of topology 1

The creation of a topology for drones using this approach is not simple and demands substantial resources. Therefore, we have opted for a more streamlined methodology, involving the utilization of a time-invariant matrix as adjacency matrix.

As described in Chapter 1, a formation master-slave has been considered. A UGV serves as the master and interconnects with all other UGVs, as illustrated in Figure 3.6.

3.2 Implementation and Application of NMPC

In this section, we aim to delineate the implementation of the Nonlinear Model Predictive Control (NMPC) algorithm employing the Pontryagin approach. Our focus

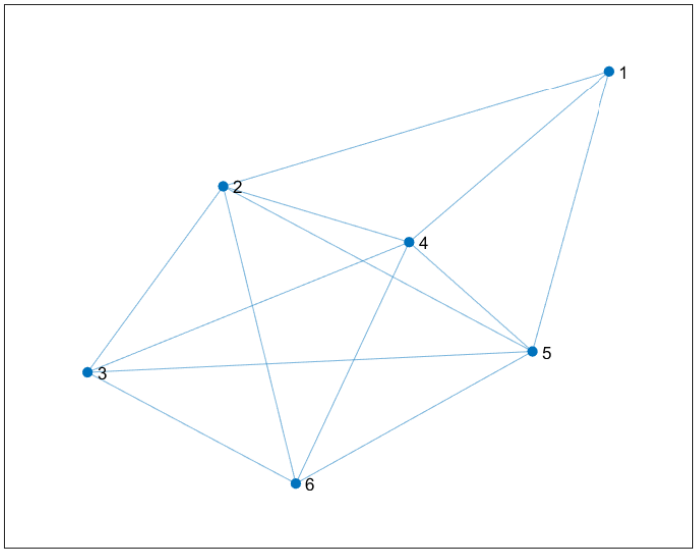

Figure 3.4: Example of topology 2

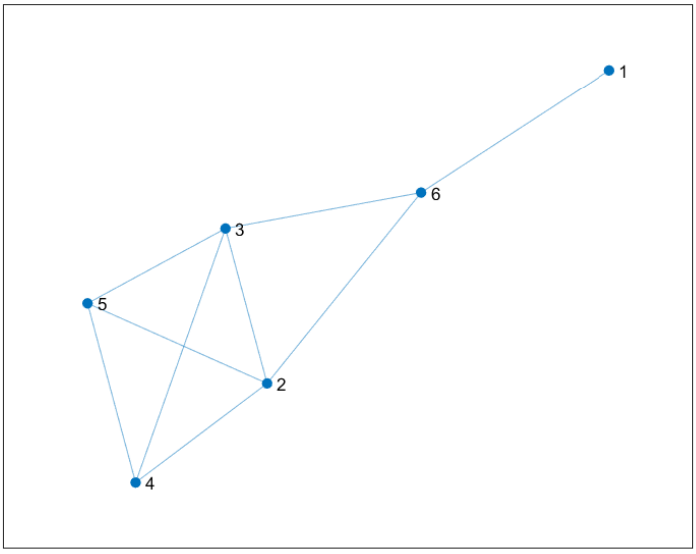

Figure 3.5: Example of topology 3

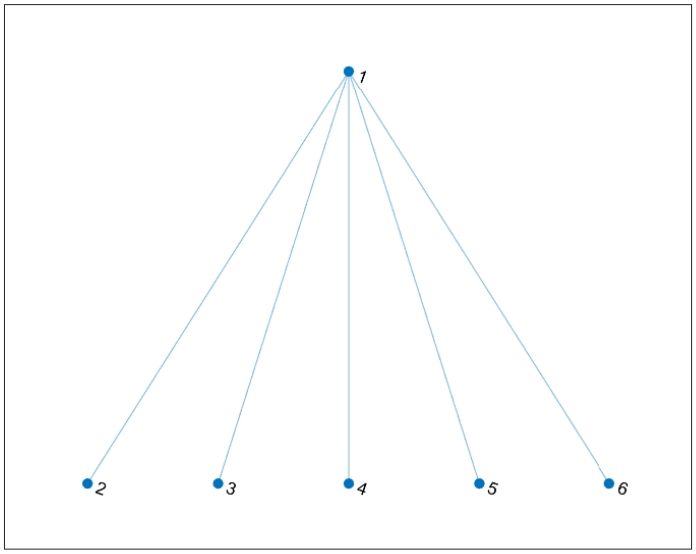

Figure 3.6: Topology for this work

lies in elucidating its application, commencing with its deployment on a singular Unmanned Ground Vehicle (UGV), and subsequently extending its utilization to multiple UGVs within a formation context.

3.2.1 Implementation and Simulation of NMPC with Pontryagin approach

Implementation on a single UGV

In the initial phase, the selection of parameters for developing the NMPC simulation on a single UGV is a crucial step. The primary function, called *main* function, encapsulated in the provided MATLAB code snippet, outlines key parameters and configurations.

```
1
2 %Initialization of parameters
3 Tstart = 0;
4 Tstop  = 40;
5 Ts     = 0.01;
6
7 par.Ts    = Ts;
```

```
par.model = @Rover_dyn_2;
par.n     = 3;
par.nu    = 2;
par.ny    = 3;
par.tol   = [0.02; 0.02; 0.02];
par.Tctrl = 0;
par.Tp    = 0.1;
par.NS    = 100;
par.MS    = 10;
par.R     = [15 0;0 15];
par.P     = [30 0 0; 0 50 0; 0 0 50];
par.Q     = [10000 0 0;0 500 0; 0 0 200];


%Number of drones considered
num_componenti=1;
for i = 1:num_componenti
    parextractor{i}.Ts = par.Ts{i};
    parextractor{i}.model = par.model{i};
    parextractor{i}.n = par.n{i};
    parextractor{i}.nu = par.nu{i};
    parextractor{i}.ny = par.ny{i};
    parextractor{i}.tol = par.tol{i};
    parextractor{i}.Tctrl = par.Tctrl{i};
    parextractor{i}.Tp = par.Tp{i};
    parextractor{i}.NS = par.NS{i};
    parextractor{i}.MS = par.MS{i};
    parextractor{i}.R = par.R{i};
    parextractor{i}.P = par.P{i};
    parextractor{i}.Q = par.Q{i};
end

Q = diag(parextractor{1}.Q);
K = nmpc_design0(parextractor{1});
```

The simulation duration spans from $T_{\text{start}} = 0$ to $T_{\text{stop}} = 40$ with a sampling time $T_s = 0.01$ and a prediction horizon $T_p = 0.1$. The tolerance for the system estimation is uniformly set to 0.02. Key matrices R, P, and Q are defined as:

- $Q = \begin{pmatrix} 10000 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 200 \end{pmatrix}$,

- $R = \begin{pmatrix} 15 & 0 \\ 0 & 15 \end{pmatrix}$,

- $P = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{pmatrix}$.

All parameters prefixed with *par.* are organized into a MATLAB structure, crucial for subsequent optimal control input calculations.

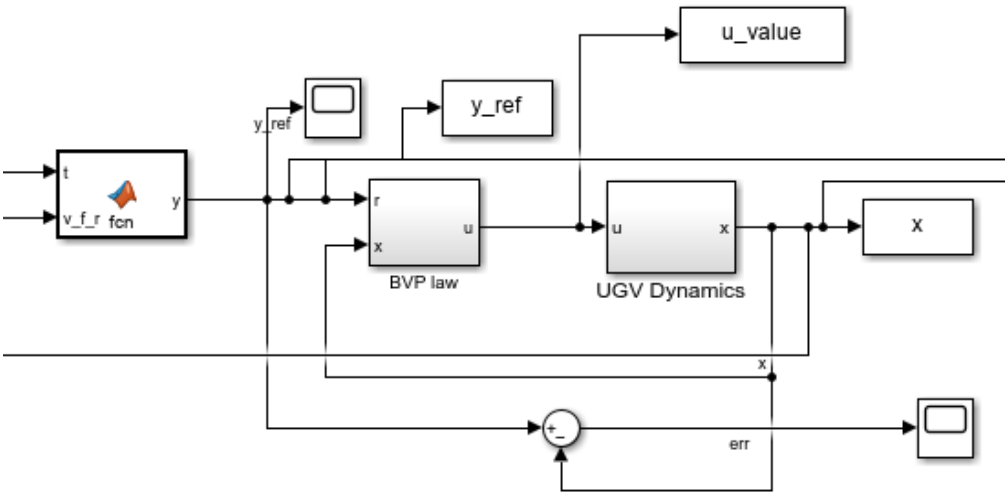

Figure 3.7: Simulink - single UGV

The simulation is executed using Simulink, integrating functions such as reference point determination, optimal control input calculation, and UGV dynamics. The key components include:

- **fcn** function box for the reference point;
- **BVP law** box for the *ustar* function (view also figure 3.8);
- **UGV dynamics** box for motion equations. (view also figure 3.9).

Simulation results, extracted from Simulink, include information about the reference point, control input values, and error between the reference and current UGV coordinates.

Now, the simulation can be initiated exclusively through the *main* function (part of this function is described in the beginning of this subsection (3.2.1)), using the following code:

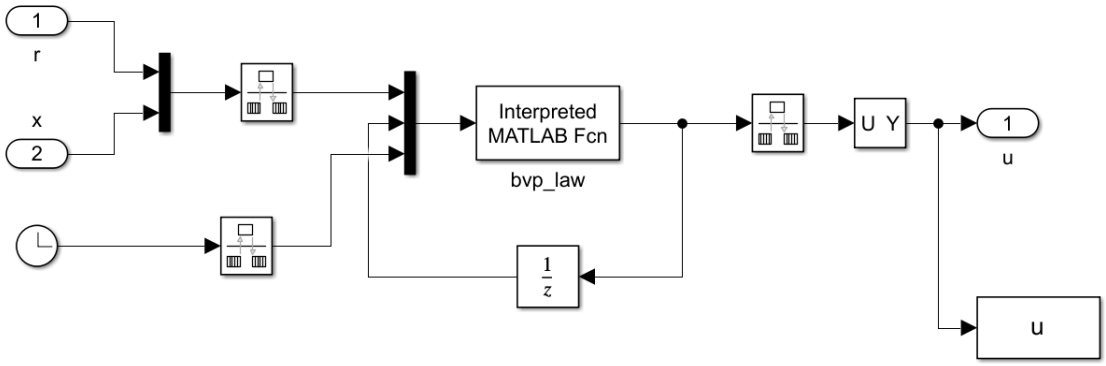

Figure 3.8: Simulink - BPV law

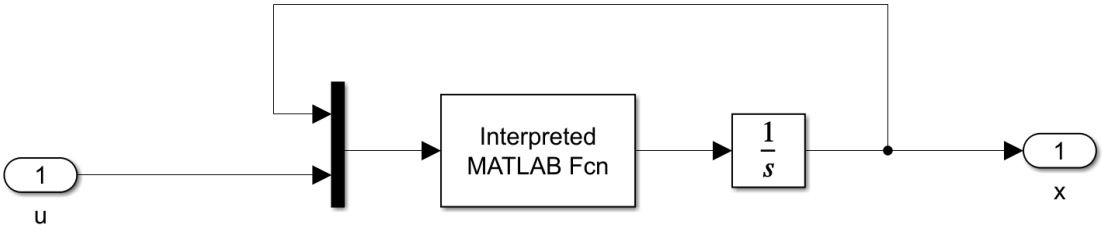

Figure 3.9: Simulink - UGV Dynamics

```
x0 = [5;0;0];
lambda0 = [0;0;0];
u0 = [0;  0];

% Here simulation starts
sim_model = sprintf('UGV_sim_1');
Decimation = 1;
options = simset('decimation',Decimation,'solver','ode4','FixedStep',
    Ts,'OutputVariables','ty');
open(sim_model);
sim(sim_model,[Tstart,Tstop],options);

% Here some plots after simulation

% UGV trjectory plot
figure
axis equal
xlabel('x')
ylabel('y')
hold on
grid on
```

```
plot(x(:,1), x(:,1).*tan(x(:,3)), 'b','linewidth', 2)

% Control input plot in function of time
figure
axis equal
xlabel('t')
ylabel('u')
hold on
grid on
plot(u_value(:,1), 'b','linewidth', 2)
plot(u_value(:,2), 'b','linewidth', 2)
```

It is noteworthy that the initial position for the Unmanned Ground Vehicle (UGV) has been set $x_0 = [5; 0; 0]$, which, in Cartesian coordinates, denotes the point (5,0). Subsequently, the control input and Euler-Lagrange equations were initialized with vectors comprising entirely of zero components. Notice that in **fcn** function box in Simulink (figure 3.7), we considered as reference value $y = [20; 0; \pi/6]$.

Following these specifications, the simulation was initiated employing the *simset* and *sim* commands, invoking the Simulink platform and encapsulated functions.

After the simulation, graphical representations were generated to illustrate the trajectory of the Unmanned Ground Vehicle (UGV) in Cartesian coordinates and the temporal evolution of the control input.

The minimized trajectory of the UGV, taking into account the initial conditions specified earlier, is depicted in Figure 3.10. Additionally, the time-dependent behavior of the control inputs is presented in Figures 3.11 and 3.12. Notably, it is evident from these figures that the control inputs exhibit a tendency to converge to zero over time.

Implementation of NMPC on more UGVs and formation of a network

Implementation and Formation

In this part of work, we aim to expand the application of Nonlinear Model Predictive Control (NMPC) to a fleet of six Unmanned Ground Vehicles (UGVs) operating in formation. The underlying concept aligns with previously discussed principles, and the MATLAB functions employed remain the same. However, a distinctive feature is introduced where only one drone, referred to as the master drone, serves as the reference point. All other drones derive their reference from the master drone and are tasked with forming a predefined shape relative to the master drone.
The system's structure is implemented in Simulink.

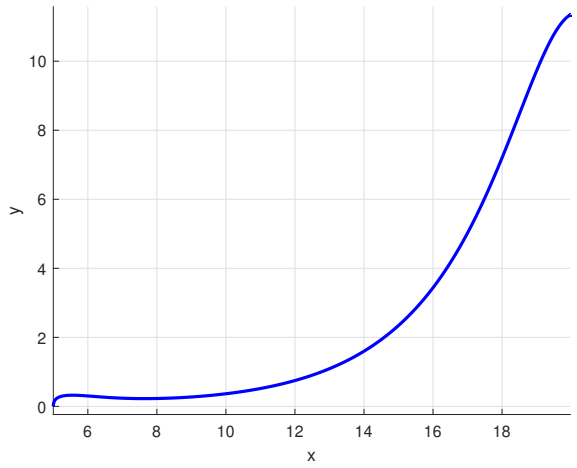

Figure 3.10: trajectory of UGV using NMPC with Pontryagin approach

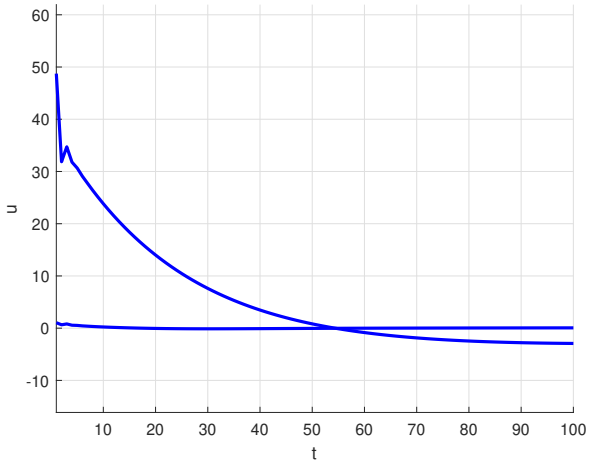

Figure 3.11: Behaviour of control inputs in time (1)

The methodology commences with the utilization of the adjacency matrix, illustrated in Figure 3.6, corresponding to the connectivity of the drones:

$$W = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.1}$$

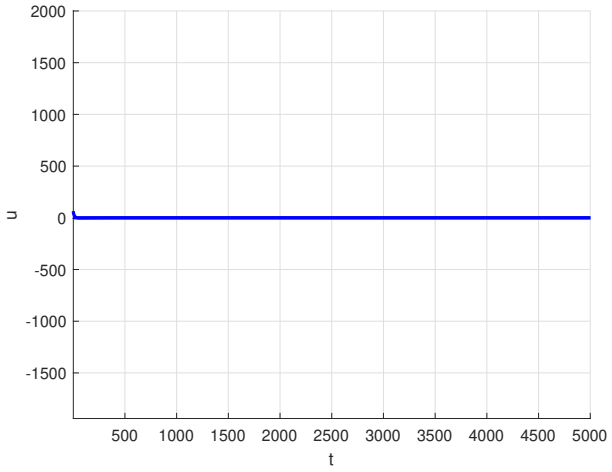

Figure 3.12: Behaviour of control inputs in time (2)

To streamline the simulation, assuming a connection in each row of the matrix, we intentionally disregard prior knowledge of the topology. Consequently, for each drone, excluding the master, dedicated functions are employed to extract connection information from the adjacency matrix, as demonstrated in Figure 3.13.

Figure 3.13 illustrates a function containing the adjacency matrix on the left, with two subsequent functions containing the following MATLAB codes:

```
function y = fcn(A)
% Number of drones
n=6;

y=0;

% I want to extract the information about the connection with master
    drone
for i=1:n
    % Value 2 means that this code is about the second UGV in the
    order of the UGVs on Simulink
     if A(2,i)==1
         y=i;
     end
end
end

function y = fcn(u1,u2,u3,u4,u5,par)
% Here, we want to pass the exact position for the reference of
    master drone
```

```
18
19    if par==1
20        y=u1;
21    else
22        y=u2;
23    end
24 end
```

The latter function, on the right, encompasses information about the position each UGV must assume concerning the master drone's position. The adopted formation law for our simulations is a regular polygon, specifically a pentagon surrounding the master drone. The general law using Cartesian and polar coordinates is expressed as:

$$(x, y) = (x_0 + rcos(\theta_0 + \theta_k), y_0 + rsin(\theta_0 + \theta_k)) \tag{3.2}$$

with a given radius r, and where (x_0, y_0) is the current position and θ_0 the current angle of the master drone, $\theta_k = (2\pi k)/n$ with $k = 1, ...,5$ and $n = 5$ (since we have 5 drones around the master).
In our simulation, following the previous law, we consider the following code (this one is about the second drone, in the order, on Simulink):

```
1 function y = fcn(x)
2
3 % We fix here the ray
4 ray=3;
5
6 y=[x(1,:)+ray*cos(x(3,:)+(2*pi/5)*2);0;atan((x(1,:).*tan(x(3,:))+ray*
      sin(x(3,:)+(2*pi/5)*2))./(x(1,:)+ray*cos(x(3,:)+(2*pi/5)*2)))];
7
8 end
```

where the fixed ray is 3.

Since the motion equations yield (x, v_x, θ), the transformation between Cartesian and polar coordinates is applied to determine the correct arrangement of individual UGVs around the master drone.

Notice that in this case the *main* function provides the initialization of the parameters for all the UGVs with the usage of cell arrays in MATLAB. The code snippet below showcases the initialization process:

```
1 Tstart = 0;
```

```
Tstop  = 40;
Ts     = 0.01;

% Value as flag to give information to master drone about reference
value_for_ref=0;

%% NMPC parameters

% Simulation parameters throught the usage of a structure with cell
    arrays

par.Ts     = {Ts,Ts,Ts,Ts,Ts,Ts};
par.model = {@Rover_dyn_2,@Rover_dyn_2,@Rover_dyn_2,@Rover_dyn_2,
    @Rover_dyn_2,@Rover_dyn_2};
par.n      = {3,3,3,3,3,3};
par.nu     = {2,2,2,2,2,2};
par.ny     = {3,3,3,3,3,3};
par.tol    = {[0.02; 0.02; 0.02],[0.02; 0.02; 0.02],[0.02; 0.02;
    0.02],[0.02; 0.02; 0.02],[0.02; 0.02; 0.02],[0.02; 0.02; 0.02]};
par.Tctrl = {0,0,0,0,0,0};
par.Tp     = {0.1,0.1,0.1,0.1,0.1,0.1};
par.NS     = {100,100,100,100,100,100};
par.MS     = {10,10,10,10,10,10};
par.R      = {[15 0;0 15],[15 0;0 15],[15 0;0 15],[15 0;0 15],[15 0;0
    15],[15 0;0 15]};
par.P      = {[30 0 0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0 50],[30 0
    0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0
    50],[30 0 0; 0 50 0; 0 0 50]};
par.Q      = {[10000 0 0;0 500 0; 0 0 200], [10000 0 0;0 500 0; 0 0
    200], [10000 0 0;0 500 0; 0 0 200],[10000 0 0;0 500 0; 0 0 200],
    [10000 0 0;0 500 0; 0 0 200], [10000 0 0;0 500 0; 0 0 200]};

%Number of drones considered

num_componenti=6;
for i = 1:num_componenti
     parextractor{i}.Ts = par.Ts{i};
     parextractor{i}.model = par.model{i};
     parextractor{i}.n = par.n{i};
     parextractor{i}.nu = par.nu{i};
     parextractor{i}.ny = par.ny{i};
     parextractor{i}.tol = par.tol{i};
     parextractor{i}.Tctrl = par.Tctrl{i};
     parextractor{i}.Tp = par.Tp{i};
     parextractor{i}.NS = par.NS{i};
     parextractor{i}.MS = par.MS{i};
     parextractor{i}.R = par.R{i};
     parextractor{i}.P = par.P{i};
     parextractor{i}.Q = par.Q{i};
```

```
end

% Extract the different parextractors

Q = diag(parextractor{1}.Q);
K1 = nmpc_design0(parextractor{1});
Q          = diag(parextractor{2}.Q);
K2 = nmpc_design0(parextractor{2});
Q          = diag(parextractor{3}.Q);
K3 = nmpc_design0(parextractor{3});
Q          = diag(parextractor{4}.Q);
K4 = nmpc_design0(parextractor{4});
Q          = diag(parextractor{5}.Q);
K5 = nmpc_design0(parextractor{5});
Q          = diag(parextractor{6}.Q);
K6 = nmpc_design0(parextractor{6});
```

It is noteworthy that the *value_for_ref* flag is incorporated in the Simulink environment. This flag is crucial for choosing the reference point for the master drone. The objective is to ensure a coherent formation and reformation after the initial formation, targeting a distinct reference point.

The remaining initialization for this specific simulation is provided as follows:

```
%% Initial conditions
x0 =[5;0;0];
x1 =[5.1;0;0];
x3=[5.2;0;0];
x4=[5.3;0;0];
x5=[5.4;0;0];
x6=[5.5;0;0];
lambda0 = [0;0;0];
u0 = [0; 0];
```

The simulation is executed using the subsequent code within the *main* function:

```
% Simulation
 sim_model = sprintf('UGV_sim_1');
 Decimation = 1;
 options = simset('decimation',Decimation,'solver','ode4','FixedStep'
    ,Ts,'OutputVariables','ty');
 open(sim_model);
 sim(sim_model,[Tstart,Tstop],options);
```

Subsequently, the trajectories of the drones in Cartesian coordinates and the behavior of control inputs over time are visualized through the following code:

```
%% Post Processing

% Trajectories figure

figure
axis equal
xlabel('x')
ylabel('y')
hold on
grid on
plot(x(:,1), x(:,1).*tan(x(:,3)), 'b','linewidth', 2)
plot(x11(:,1), x11(:,1).*tan(x11(:,3)), 'r','linewidth', 2)
plot(x22(:,1), x22(:,1).*tan(x22(:,3)), 'g','linewidth', 2)
plot(x33(:,1), x33(:,1).*tan(x33(:,3)), 'k','linewidth', 2)
plot(x44(:,1), x44(:,1).*tan(x44(:,3)), 'c','linewidth', 2)
plot(x55(:,1), x55(:,1).*tan(x55(:,3)), 'm','linewidth',  2)

% Control inputs in time

figure
axis equal
xlabel('t')
ylabel('u')
hold on
grid on
plot(u_value(:,1), 'b','linewidth', 2)
plot(u_value(:,2), 'b','linewidth', 2)
plot(u_out1(:,1), 'r','linewidth', 2)
plot(u_out1(:,2), 'r','linewidth', 2)
plot(u_out2(:,1), 'g','linewidth', 2)
plot(u_out2(:,2), 'g','linewidth', 2)
plot(u_out3(:,1), 'c','linewidth', 2)
plot(u_out3(:,2), 'c','linewidth', 2)
plot(u_out4(:,1), 'm','linewidth', 2)
plot(u_out4(:,2), 'm','linewidth', 2)
plot(u_out5(:,1), 'y','linewidth', 2)
plot(u_out5(:,2), 'y','linewidth', 2)
```

The resulting trajectories and control inputs are illustrated in Figure 3.14 and Figures 3.15-3.16.
Notably, the first reference point for the master drone corresponds to $y_{ref} = [20; 0; \pi/6]$ as in the case of a single UGV.

Notice that control inputs tend to zero properly, in time.

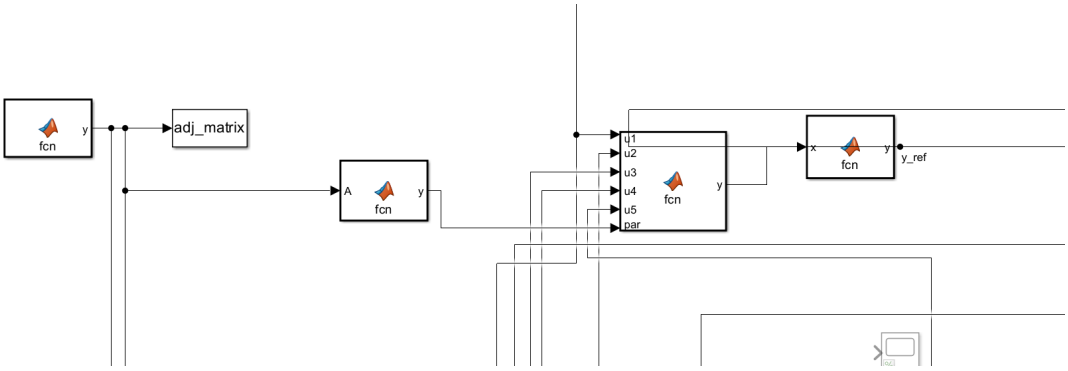

Figure 3.13: Information extraction from adj matrix

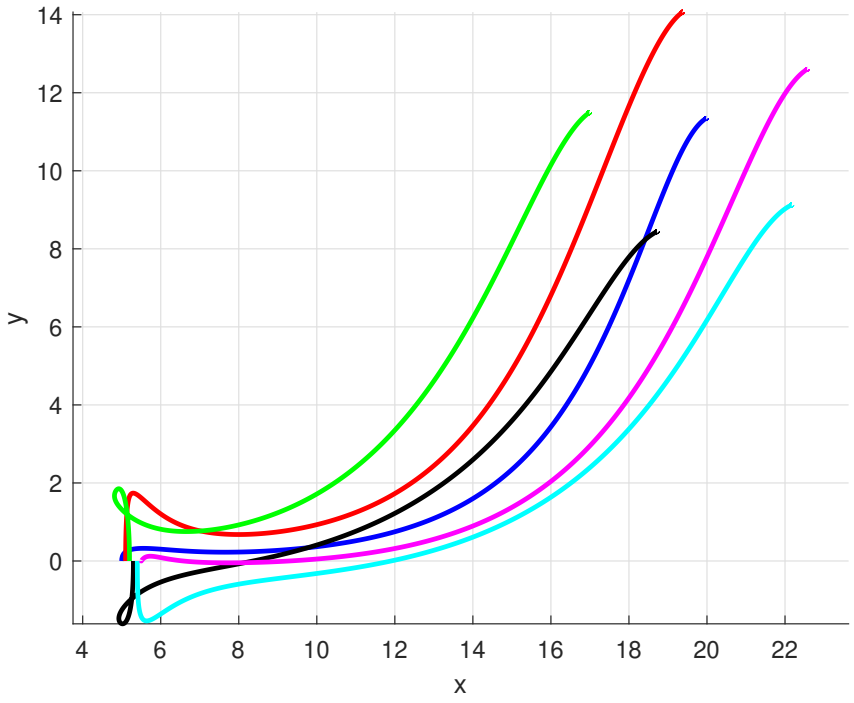

Figure 3.14: Trajectories - six UGVs

Implementation and Reformation

In the process of reforming drones, the following codes were implemented within the *main* function.

The initial piece of code serves the purpose of re-establishing the initial conditions

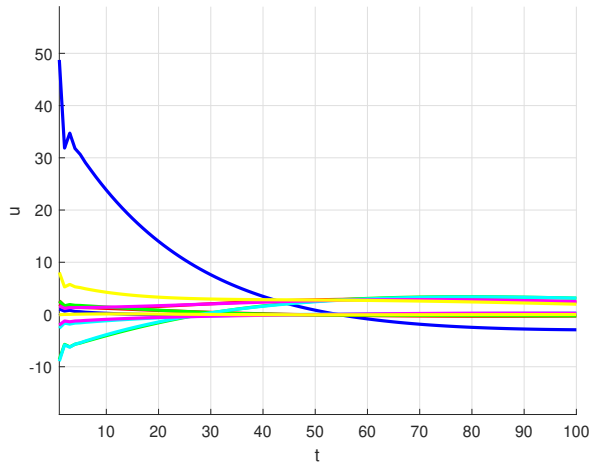

Figure 3.15: variation in time of control inputs - six UGVs [1]

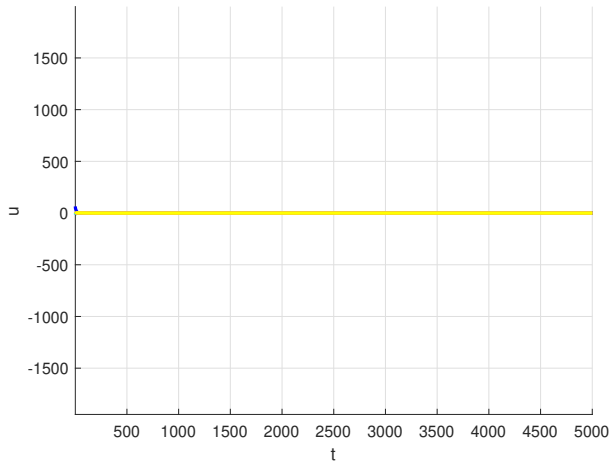

Figure 3.16: variation in time of control inputs - six UGVs [2]

for the Unmanned Ground Vehicles (UGVs), representing the conditions of the previously formed configuration. This occurs only if the previous formation exhibits a tolerance of 0.02 for each motion component:

```
1
2 %% Finding new initial condition after formation
3 toll=[0.02;0.02;0.02];
4 dimerror=size(simout);
5 dimarray=size(x);
```

```
dimerror=dimerror(1);
dimarray=dimarray(1);
if all(simout(dimerror)<toll) && all(simout1(dimerror)<toll) && all
    (simout2(dimerror)<toll) && all(simout3(dimerror)<toll) && all(
    simout4(dimerror)<toll) && all(simout5(dimerror)<toll)
     x0_new=x(dimarray,:);
     x1_new=x11(dimarray,:);
     x2_new=x22(dimarray,:);
     x3_new=x33(dimarray,:);
     x4_new=x44(dimarray,:);
     x5_new=x55(dimarray,:);
end


%% New Initial conditions after formation

x0 = x0_new;
x1 = x1_new;
x3=x2_new;
x4=x3_new;
x5=x4_new;
x6=x5_new;
lambda0 = [0;0;0];
u0 = [0; 0];
```

Subsequently, the simulation was restarted with the updated reference value for the master drone by modifying the *value_for_ref* flag:

```
%% Simulation after formation

value_for_ref=1;
sim_model = sprintf('UGV_sim_1');
Decimation = 1;
options = simset('decimation',Decimation,'solver','ode4','FixedStep',
    Ts,'OutputVariables','ty');
open(sim_model);
sim(sim_model,[Tstart,Tstop],options);
```

Additionally, the new trajectories of drones were plotted, along with the variation of the master-slave distance over time. Discrepancies were calculated as the difference between the maximum and minimum variations in the master-slave distance:

```
% Plot of trajectories

figure
```

```
axis equal
xlabel('x')
ylabel('y')
hold on
grid on
plot(x(:,1), x(:,1).*tan(x(:,3)), 'b','linewidth', 2)
plot(x11(:,1), x11(:,1).*tan(x11(:,3)), 'r','linewidth', 2)
plot(x22(:,1), x22(:,1).*tan(x22(:,3)), 'g','linewidth', 2)
plot(x33(:,1), x33(:,1).*tan(x33(:,3)), 'k','linewidth', 2)
plot(x44(:,1), x44(:,1).*tan(x44(:,3)), 'c','linewidth', 2)
plot(x55(:,1), x55(:,1).*tan(x55(:,3)), 'm','linewidth',  2)



%% Variation of distance slave-master in time during the riformation

figure
hold on
grid on

d1=sqrt((x(:,1)-x11(:,1)).^2+ (x(:,1).*tan(x(:,3))-x11(:,1).*tan(x11
    (:,3))).^2);
d2=sqrt((x(:,1)-x22(:,1)).^2+ (x(:,1).*tan(x(:,3))-x22(:,1).*tan(x22
    (:,3))).^2);
d3=sqrt((x(:,1)-x33(:,1)).^2+ (x(:,1).*tan(x(:,3))-x33(:,1).*tan(x33
    (:,3))).^2);
d4=sqrt((x(:,1)-x44(:,1)).^2+ (x(:,1).*tan(x(:,3))-x44(:,1).*tan(x44
    (:,3))).^2);
d5=sqrt((x(:,1)-x55(:,1)).^2+ (x(:,1).*tan(x(:,3))-x55(:,1).*tan(x55
    (:,3))).^2);
axis equal
plot(d1, 'b', 'Linewidth', 2)
plot(d2, 'r', 'Linewidth', 2)
plot(d3, 'g', 'Linewidth', 2)
plot(d4, 'k', 'Linewidth', 2)
plot(d5, 'm', 'Linewidth', 2)


xlabel('t')
ylabel('distance')


%% Values of discrepancies

maxdiscrepancy1=max(d1)-min(d1);
maxdiscrepancy2=max(d2)-min(d2);
maxdiscrepancy3=max(d3)-min(d3);
maxdiscrepancy4=max(d4)-min(d4);
maxdiscrepancy5=max(d5)-min(d5);
```

In this simulation, the new reference value for the master drone is specified as $y_{ref} = [10; 0; \pi/6]$.

The reformation of UGVs, along with the master-slave distance variations over time, is illustrated in Figures 3.17, 3.18, and 3.19:

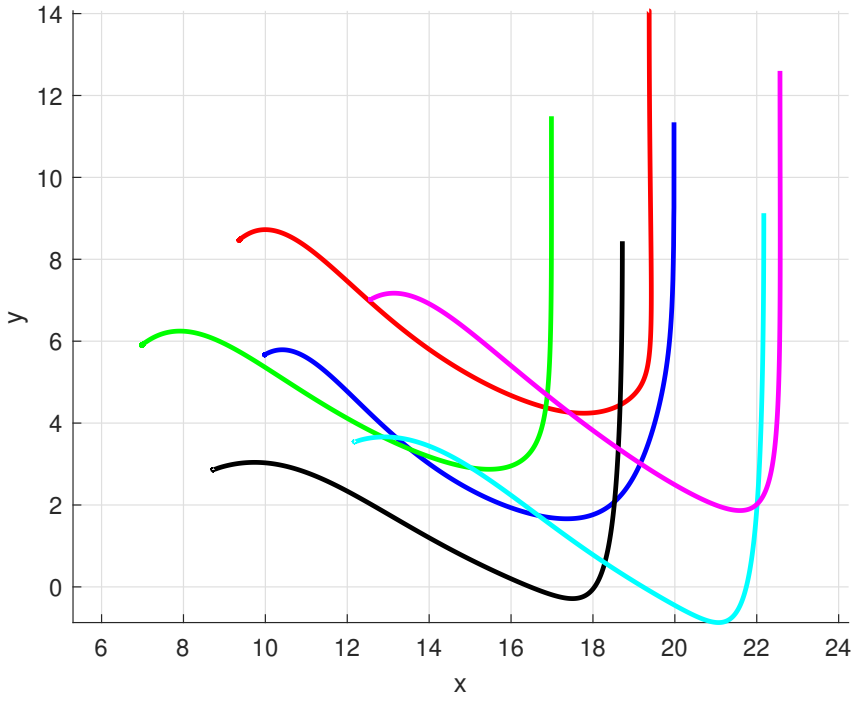

Figure 3.17: Reformation - six UGVs

From the analysis of Figures 3.17, 3.18, and 3.19, it is evident that the reformation of UGVs is effective, as the master-slave distance exhibits a coherent behavior from the initial to the reformed formation, because it is observable that, after the first formation, they move in formation. This observation is further supported by table 3.1, illustrating the discrepancies of the drones with respect to the master drone.

maxdiscrepancy1	4.1486
maxdiscrepancy2	3.7249
maxdiscrepancy3	2.8818
maxdiscrepancy4	3.4396
maxdiscrepancy5	3.1590

Table 3.1: Discrepancies

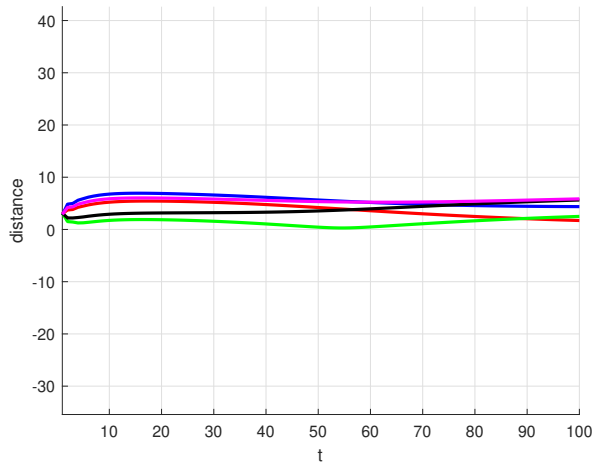

Figure 3.18: Variation distance master-slave in time [1]

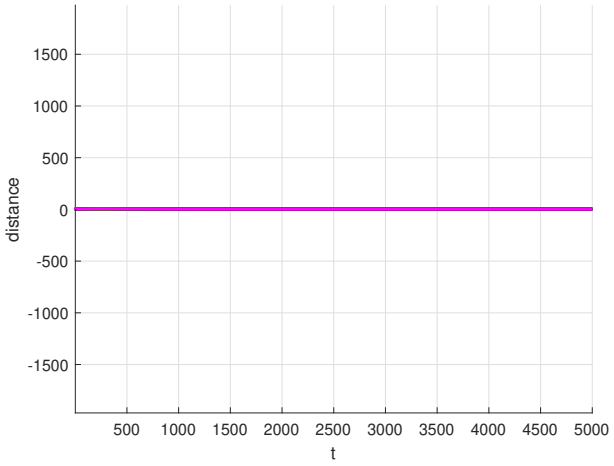

Figure 3.19: Variation distance master-slave in time [2]

3.2.2 Implementation and Simulation of robust NMPC with Pontryagin approach

In real-world scenarios, the motion of Unmanned Ground Vehicles (UGVs) is often perturbed by external noises, such as wind or friction with the terrain. To address this challenge, the objective is to make an algorithm capable of mitigating such disturbances, thereby enabling precise movement of individual drones along a predefined reference path. Furthermore, the algorithm should facilitate the coordination of multiple drones to form cohesive formations coherently. This way

entails the development of robust control strategies to ensure reliable and accurate navigation among external perturbations.

Perturbation on motion equations

In our analysis of perturbations affecting the Unmanned Ground Vehicle (UGV) model, we have considered the following perturbed dynamics:

$$\dot{x} = v \tag{3.3}$$

$$\dot{v} = u_1 \cos\theta + w_1 \tag{3.4}$$

$$\dot{\theta} = u_2 + w_2 \tag{3.5}$$

Here, w_1 and w_2 represent disturbances extracted from a uniform distribution within the interval $(-0.2, 0.2)$.

Given our decision, as outlined in Subsubsection 3.2.1, to maintain a fixed distance ray of 3 units from the master drone in the formation component, perturbations of maximum 0.2 or minimum -0.2 on the acceleration and angular velocity equations can significantly impact our system.

Implementation of robust NMPC

In Simulink, the environment resembles what depicted in previous figures. The key modifications occur within the *bvp_law* (figure 3.8), and the main function.

The idea of the code structure mirrors the previously discussed NMPC in section 3.2.1. And optimal values for noise and control inputs are determined, as outlined in section 2.2.6. Notably, perturbations are introduced in the second and third equations, impacting both motion and Euler-Lagrange equations.

It's worth noting that we treat the Euler-Lagrange equations as related to the control input. As demonstrated in section 6, we utilize the three solutions from the Euler-Lagrange equations of control input, with altered signs, for optimal disturbance calculation.

In this implementation, we have also introduced saturation for the optimal control inputs. This ensures that if the optimal control inputs exceed a predetermined threshold (it has to be defined in the main function), they are saturated, thus preventing the emergence of anomalous control inputs that are excessively high or low.

Here we find the MATLAB function for the nominal scenario:

```
function [s_dot, y] = UGV_dyn(s, u)
global delta

s_dot(1,:) = s(2,:)+delta(1,:);
s_dot(2,:) = u(1).*cos(s(3,:))+delta(2,:);
s_dot(3,:) = u(2)+delta(3,:);

y = s;
```

Here we find the MATLAB function for the ideal scenario:

```
function [s_dot, y] = UGV_dyn_pred(s, u)

s_dot(1,:) = s(2,:);
s_dot(2,:) = u(1).*cos(s(3,:));
s_dot(3,:) = u(2);

y = s;
```

Moving forward, let us delve into the changes made in the main function directly for the six Unmanned Ground Vehicles (UGVs):

```
clear all
close all
clc

%% Parameters definition

global delta gamma x0 x1 x3 x4 x5 x6 lambda0 umax Q flag

%gamma parameter for the disturbance
gamma=1;

%choice of flag to simulate: Ideal case (0) - Nominal case (1) -
    Robust case (2)
flag =2;

%fix seed for the disturbance
rng(42);

%noise extracted from uniform distribution U(-0.2,0.2)
intensity1=0.2;
intensity2=0.2;
val1=-1+2*rand;
val2=-1+2*rand;
```

```
noise1 = intensity1*val1;
noise2 = intensity2*val2;

if flag == 0
    disp('Ideal scenario -> No uncertainty in the system')
    delta = [0;0;0];
elseif flag == 1
    disp('Nominal scenario -> Uncertainty in the system')
    delta = [0;noise1;noise2];
elseif flag == 2
    disp ('Robust scenario')
    delta = [0;noise1;noise2];
else
    error('Invalid flag')
end

%% Simulation parameters
Tstart = 0;
Tstop  = 40;
Ts     = 0.01;

%% NMPC parameters
%simulation parameters throught the usage of a structure
par.Ts     = {Ts,Ts,Ts,Ts,Ts,Ts};
par.model = {@UGV_dyn,@UGV_dyn,@UGV_dyn,@UGV_dyn,@UGV_dyn,@UGV_dyn};
    %non c'è il tempo qui perchè è shift invariant (come prima
    componente)
par.pred ={@UGV_dyn_pred,@UGV_dyn_pred,@UGV_dyn_pred,@UGV_dyn_pred,
    @UGV_dyn_pred,@UGV_dyn_pred};
par.n      = {3,3,3,3,3,3}; %numero stati
par.nu     = {2,2,2,2,2,2}; %numero input
par.ny     = {3,3,3,3,3,3}; %numero output = 3
par.tol    = {[0.02; 0.02; 0.02],[0.02; 0.02; 0.02],[0.02; 0.02;
    0.02],[0.02; 0.02; 0.02],[0.02; 0.02; 0.02],[0.02; 0.02; 0.02]};
par.Tctrl = {0,0,0,0,0,0};
par.Tp     = {0.1,0.1,0.1,0.1,0.1,0.1}; %prediction horizon
par.NS     = {100,100,100,100,100,100};
par.MS     = {10,10,10,10,10,10};
par.R      = {[15 0;0 15],[15 0;0 15],[15 0;0 15],[15 0;0 15],[15 0;0
    15],[15 0;0 15]};
par.P      = {[30 0 0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0 50],[30 0
    0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0 50],[30 0 0; 0 50 0; 0 0
    50],[30 0 0; 0 50 0; 0 0 50]};
par.Q      = {[10000 0 0;0 500 0; 0 0 200], [10000 0 0;0 500 0; 0 0
    200], [10000 0 0;0 500 0; 0 0 200],[10000 0 0;0 500 0; 0 0 200],
    [10000 0 0;0 500 0; 0 0 200], [10000 0 0;0 500 0; 0 0 200]};
umax       = 20;

%number of drones considered
```

```
num_componenti=6;
for i = 1:num_componenti
    parextractor{i}.Ts = par.Ts{i};
    parextractor{i}.model = par.model{i};
    parextractor{i}.pred = par.pred{i};
    parextractor{i}.n = par.n{i};
    parextractor{i}.nu = par.nu{i};
    parextractor{i}.ny = par.ny{i};
    parextractor{i}.tol = par.tol{i};
    parextractor{i}.Tctrl = par.Tctrl{i};
    parextractor{i}.Tp = par.Tp{i};
    parextractor{i}.NS = par.NS{i};
    parextractor{i}.MS = par.MS{i};
    parextractor{i}.R = par.R{i};
    parextractor{i}.P = par.P{i};
    parextractor{i}.Q = par.Q{i};
end

% Extract the different parextractor
Q = diag(parextractor{1}.Q);
K1 = nmpc_design0(parextractor{1}); %crea una struttura tramite
    nmpc_design0 con i parametri definiti in precedenza
Q          = diag(parextractor{2}.Q);
K2 = nmpc_design0(parextractor{2});
Q          = diag(parextractor{3}.Q);
K3 = nmpc_design0(parextractor{3});
Q          = diag(parextractor{4}.Q);
K4 = nmpc_design0(parextractor{4});
Q          = diag(parextractor{5}.Q);
K5 = nmpc_design0(parextractor{5});
Q          = diag(parextractor{6}.Q);
K6 = nmpc_design0(parextractor{6});

%% Initial conditions
x0 = [5;0;0];
x1 = [5.1;0;0];
x3=[5.2;0;0];
x4=[5.3;0;0];
x5=[5.4;0;0];
x6=[5.5;0;0];
lambda0 = [0;0;0];
u0 = [0;  0];


%% Simulation

 sim_model = sprintf('UGV_sim_1');
 Decimation = 1;
```

```
110  options = simset('decimation',Decimation,'solver','ode4','FixedStep'
       ,Ts,'OutputVariables','ty');
111  open(sim_model);
112  sim(sim_model,[Tstart,Tstop],options);
113
114 %% Post Processing
115
116 %trajectory plot
117
118 figure
119 axis equal
120 xlabel('x')
121 ylabel('y')
122 hold on
123 grid on
124 plot(x(:,1), x(:,1).*tan(x(:,3)), 'b','linewidth', 2)
125 plot(x11(:,1), x11(:,1).*tan(x11(:,3)), 'r','linewidth', 2)
126 plot(x22(:,1), x22(:,1).*tan(x22(:,3)), 'g','linewidth', 2)
127 plot(x33(:,1), x33(:,1).*tan(x33(:,3)), 'k','linewidth', 2)
128 plot(x44(:,1), x44(:,1).*tan(x44(:,3)), 'c','linewidth', 2)
129 plot(x55(:,1), x55(:,1).*tan(x55(:,3)), 'm','linewidth',  2)
130
131 %plot of consumation for the control input
132
133 trapz(0:size(u_final_1(:,1))-1,abs(u_final_1(:,1)))
134 trapz(0:size(u_final_2(:,1))-1,abs(u_final_2(:,1)))
135 trapz(0:size(u_final_3(:,1))-1,abs(u_final_3(:,1)))
136 trapz(0:size(u_final_4(:,1))-1,abs(u_final_4(:,1)))
137 trapz(0:size(u_final_5(:,1))-1,abs(u_final_5(:,1)))
138 trapz(0:size(u_final_6(:,1))-1,abs(u_final_6(:,1)))
```

This function commences with the definition of simulation and NMPC parameters, adopting a structured approach to enhance comprehension. Notably, one of the parameters initialized here, differing from previous cases, is the value of gamma. Gamma serves the purpose of weighting the considered noise in the equations. Additionally, this is where disturbances are selected from a uniform distribution, as described before.

Furthermore, the distinction between the ideal, nominal, and robust scenarios is delineated through the "flag" parameter. In the ideal scenario, the application of NMPC occurs without any disturbances, essentially representing the absence of external influences. The nominal scenario employs the conventional NMPC with Pontryagin (see 2.2.1), which is non-robust, aimed at addressing disturbed scenarios. Conversely, the robust scenario utilizes both nominal and ideal models to iteratively adjust the nominal model, thereby minimizing disturbances and aligning it with the ideal scenario.

Following parameter initialization, the simulation is executed, and postprocessing procedures are considered. Trajectory plots and control input "consumptions" are visualized. Specifically, in the plots, we display, as control input consumptions, the integral of the absolute value of the first component of the control input, which represents a force. This allows us to quantify the "consumption" of the force over time. It's noteworthy that with robust control, we aim to demonstrate a reduced "consumption" of force, indicating improved efficiency and performance, than the usage of non robust NMPC on the distrubed UGV model.

Simulation of robust NMPC and comparison with ideal and nominal scenarios

Simulations have been conducted using a reference of $[10; 0; \pi/6]$ for the master drone.

Robust simulations have been assumed varying values for γ. Specifically, γ values of 2, 4, 10, and 25 have been employed.
It has been observed that when $\gamma = 2$, the algorithm encounters computational failure. This failure happens from the necessity, as elucidated in Remark 13, for γ to assume an appropriately "low" value. When γ is excessively low, the noise becomes excessively perceptible to the system, resulting in algorithmic failure. In this simulation, an appropriately low value has been determined to be 4.

Remark 15. *If γ is too low, we encounter the failure of concavity with respect to w and convexity with respect to u of the Hamiltonian functions.*
For this reason, there exists a critical value for γ for which $w \to \infty$.

In Figures 3.20, 3.21, and 3.22, simulations can be observed in ideal, nominal, and robust (with $\gamma = 4$) scenarios, accompanied by errors post-simulation of the three coordinates extracted by motion equations, as presented in Table 3.2.

You can see also the errors post-simulation in robust cases with $\gamma = 10$ and $\gamma = 25$ in table 3.3 and moreover the behavior of the disturbance in time using robust algorithm (on master UGV, for simplicity) with $\gamma = 4$,10,25 in figures: 3.25 - 3.28 - 3.31.
Notice that in these last cited figures, it is evident that if γ is lower, the system perceives more the disturbance.

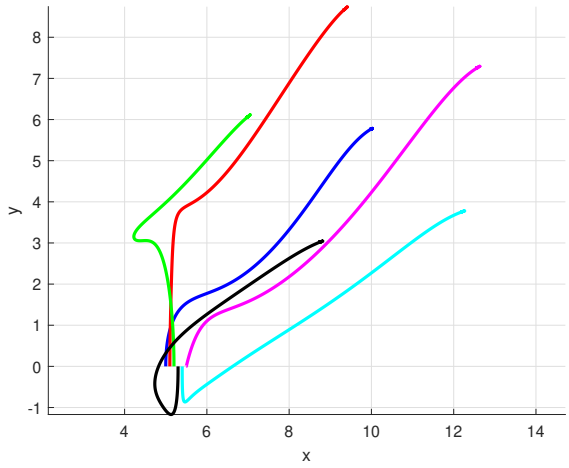

Figure 3.20: ideal scenario

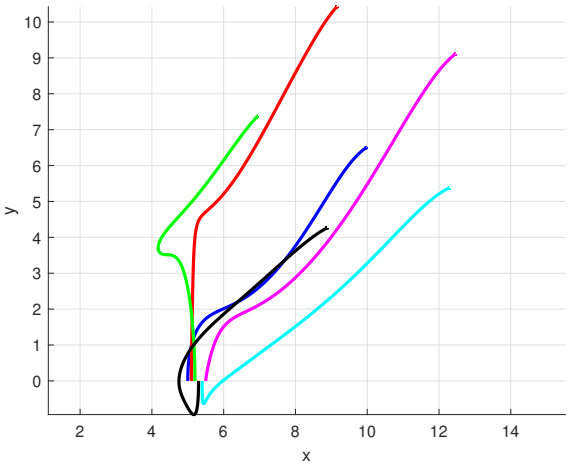

Figure 3.21: nominal scenario

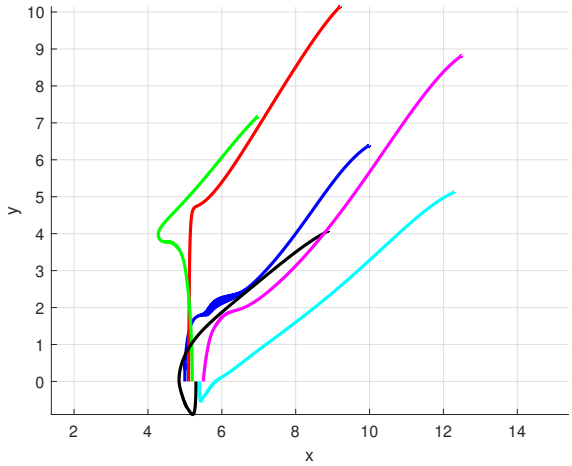

Figure 3.22: Robust scenario with $\gamma = 4$

Ideal Scenario	Nominal Scenario	Robust Scenario
(0.0051,-0.0036,0.0000)	(0.0258,-0.0051,-0.0357)	(0.0197,-0.0051,-0.0273)
(0.0116,-0.0060,-0.0001)	(0.0383,-0.0051,-0.0357)	(0.0267,-0.0051,-0.0273)
(0.0119,-0.0057,-0.0000)	(0.0359,-0.0051,-0.0357)	(0.0254,-0.0051,-0.0273)
(0.0156,-0.0058,-0.0000)	(0.0229,-0.0051,-0.0357)	(0.0180,-0.0051,-0.0273)
(0.0159,-0.0057,-0.0000)	(0.0224,-0.0051,-0.0357)	(0.0177,-0.0051,-0.0273)
(0.0123,-0.0062,0.0000)	(0.0274,-0.0051,-0.0357)	(0.0205,-0.0051,-0.0273)

Table 3.2: Errors of the 3 components from motion equations for the 6 UGVs in the different scenarios

Robust Scenario ($\gamma = 10$)	Robust Scenario ($\gamma = 25$)
(0.0236,-0.0051,-0.0323)	(0.0249,-0.0051,-0.0343)
(0.0340,-0.0051,-0.0323)	(0.0366,-0.0051,-0.0343)
(0.0320,-0.0051,-0.0323)	(0.0343,-0.0051,-0.0343)
(0.0211,-0.0051,-0.0323)	(0.0222,-0.0051,-0.0343)
(0.0206,-0.0051,-0.0323)	(0.0217,-0.0051,-0.0343)
(0.0248,-0.0051,-0.0323)	(0.0264,-0.0051,-0.0343)

Table 3.3: Errors of the 3 components from motion equations for the 6 UGVs in robust cases with $\gamma = 10$ and $\gamma = 25$

It is evident that classical NMPC applied to the disturbed model yields inferior performance compared to the case of robust NMPC applied to the disturbed

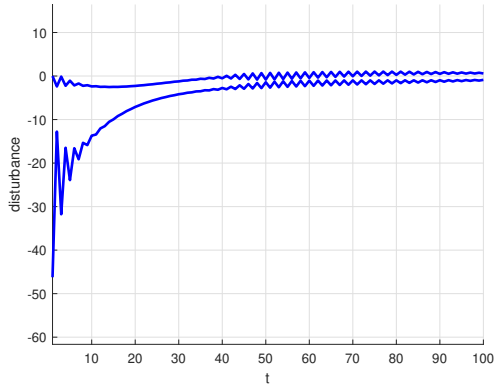

Figure 3.23

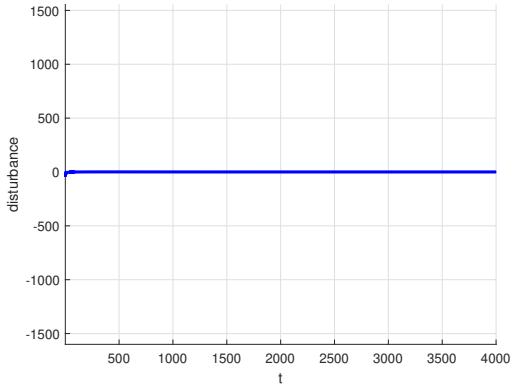

Figure 3.24

Figure 3.25: Behaviour of disturbance in time with $\gamma = 4$

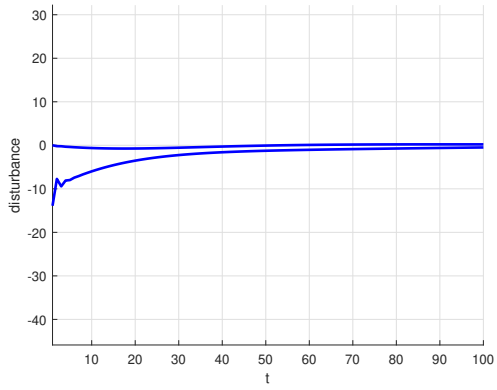

Figure 3.26

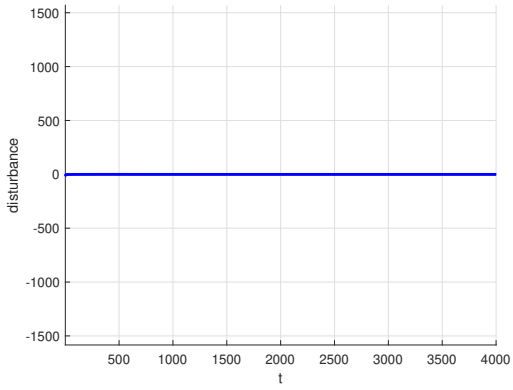

Figure 3.27

Figure 3.28: Behaviour of disturbance in time with $\gamma = 10$

model. Furthermore, it is discernible that the perturbed case, mitigated through the utilization of robust NMPC, provides a better approximation than the nominal case to the ideal case. This comparison is further elucidated by the overlapping depiction of the three aforementioned cases in Figure 3.32.

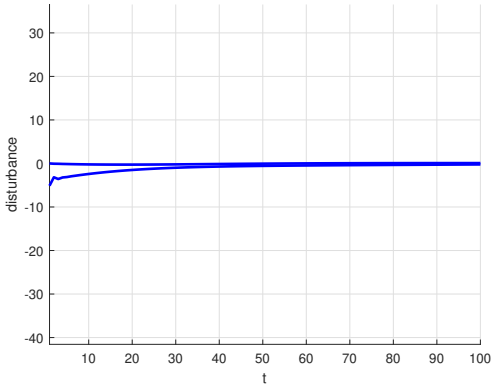

Figure 3.29

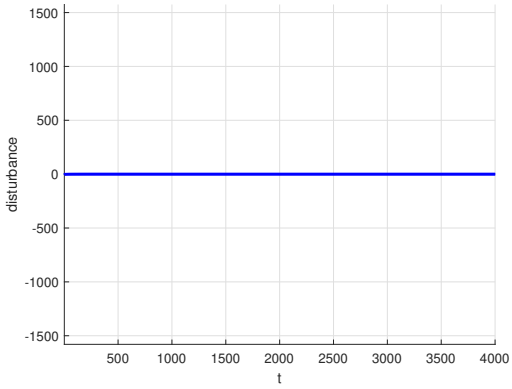

Figure 3.30

Figure 3.31: Behaviour of disturbance in time with $\gamma = 25$

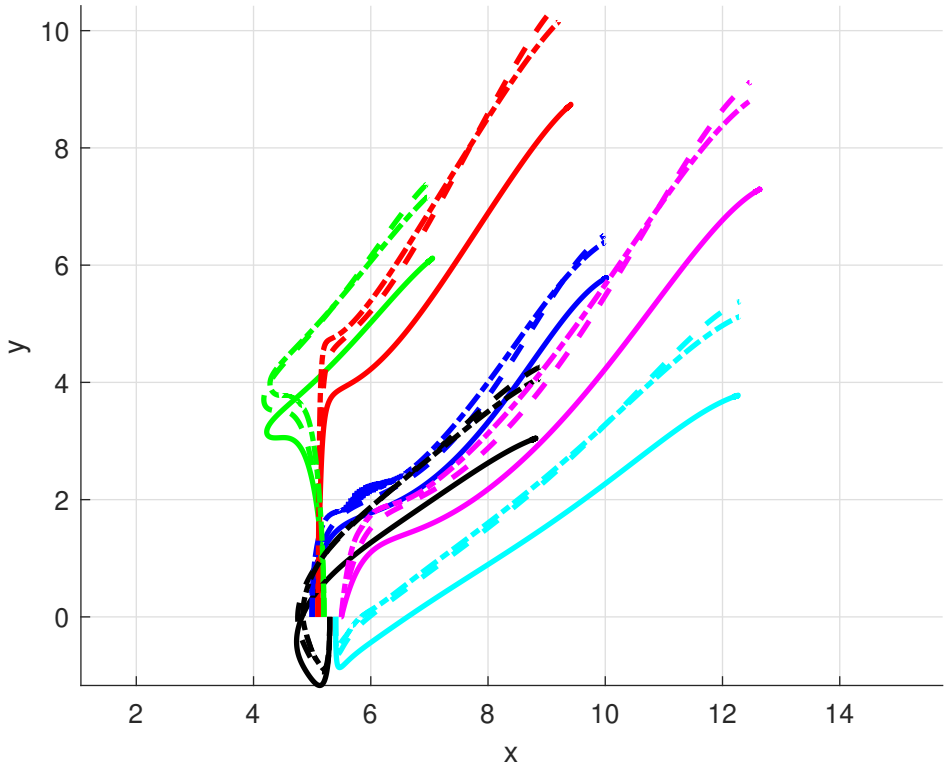

Figure 3.32: Ideal (continuous line), Nominal (dashed line), Robust (dashed line with dot) together

Investigation was also conducted, as delineated in the preceding subsubsection, into the "consumption" metrics employed by robust algorithms across varying degrees of γ. As depicted in Figure 3.33 and detailed in Table 3.4, we examined instances where γ took values of 4, 10, and 25. Notably, the optimal configuration emerged with γ set to 4, wherein a reduced module of force input was necessitated for the maneuvering of UGVs (see also table 3.5 which contains information of the very bad "consumption" with $\gamma = 2$).

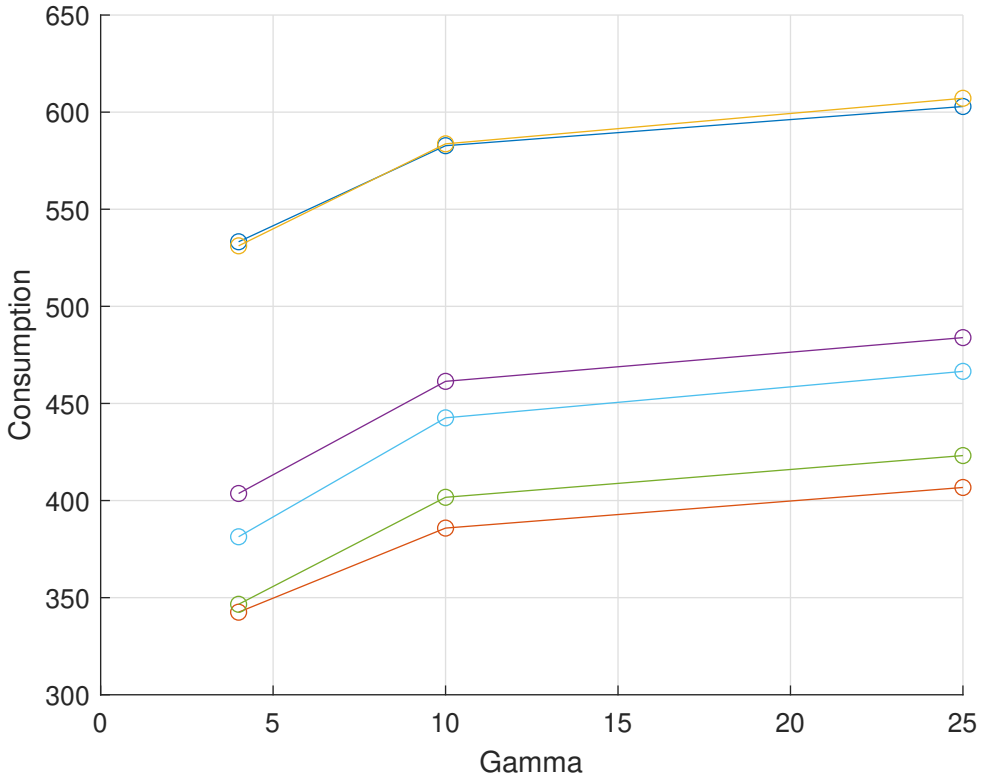

Figure 3.33: Comparison with consumption in function of γ

Robust $\gamma = 4$	Robust $\gamma = 10$	Robust $\gamma = 25$
533.2519	582.7184	602.8995
342.5116	385.8224	406.7327
531.1568	583.6731	607.1400
403.6320	461.3818	483.8935
346.5876	401.6685	423.1607
381.3436	442.5844	466.5228

Table 3.4: "Consumption" considered in robust case with different values of γ for the 6 UGVs

Robust $\gamma = 2$
5.8957e+06
5.5427e+05
1.7197e+07
4.0951e+06
1.9892e+06
5.5326e+05

Table 3.5: Very bad "consumption" in the robust case with $\gamma = 2$

Furthermore, an analysis encompassing "consumption" metrics was undertaken, encompassing ideal, robust scenarios with γ values of 4, 10, and 25, alongside nominal scenarios, with respect to the six UGVs. It was observed (figure 3.34) that the "consumption", ordered according to the aforementioned scenarios, exhibited an escalating trend (see also table 3.6 which contains "consumption" in ideal and Nominal scenarios, for a better comparison with cases just described).

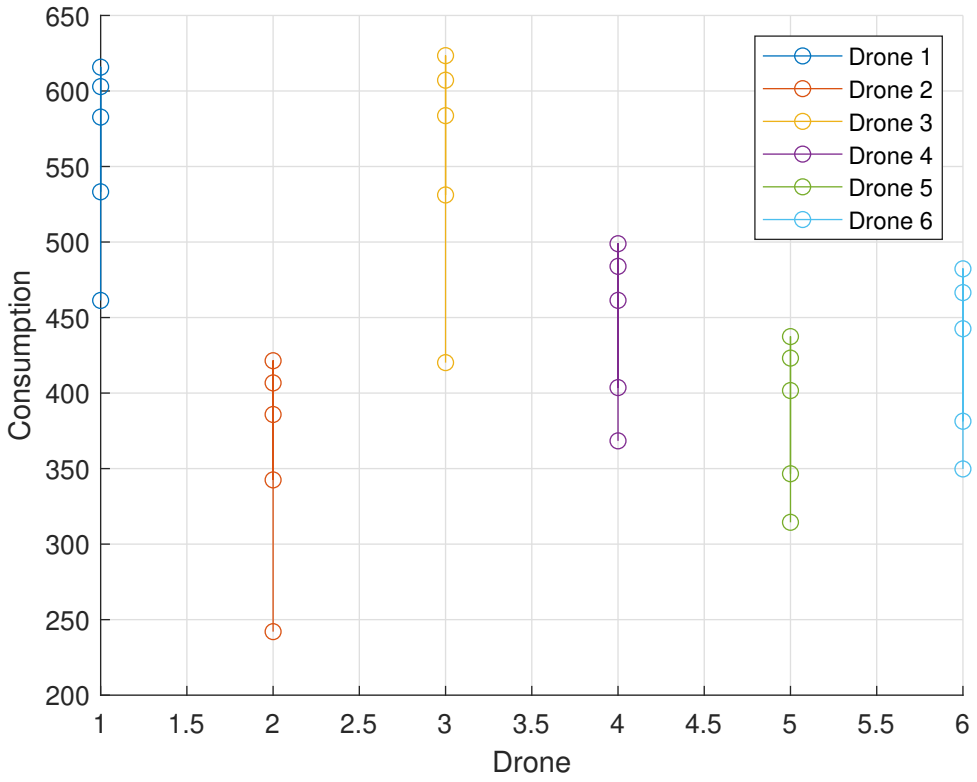

Figure 3.34: Comparison with consumption in function of the i_{th} UGV

Ideal	Nominal
461.3220	615.7744
242.0500	421.4187
420.1609	623.4009
368.3623	498.8878
314.4419	437.3979
349.7583	482.2566

Table 3.6: "Consumption" considered in Ideal and Nominal Scenarios

The figures 3.33 - 3.34 are extracted by the following codes:

```
figure;
hold on;

```

```
n=6;%number of drones
for i = 1:n
    plot(gamma_values, [u_int_4(i), u_int_10(i), u_int_25(i)], '-o');
end

xlabel('Gamma');
ylabel('Consumption');
grid on;
```

and

```
n = 6; %number of UGVs
figure;
hold on;
grid on;

for i = 1:n
    plot(i*ones(1,5) , [u_int_4(i), u_int_10(i), u_int_25(i),
    u_int_nominal(i), u_int_ideal(i)], 'o-');
end

xlabel('Drone');
ylabel('Consumption');
legend('Drone 1', 'Drone 2', 'Drone 3', 'Drone 4', 'Drone 5', 'Drone
    6');
```

For all other data and figures it is all described in the main function explained in the previous subsubsection, with some little changes of the considered variables.

About figures: 3.25-3.28-3.31 we list here:

```
figure
axis equal
xlabel('t')
ylabel('disturbance')
hold on
grid on
%plot of disturbance
plot(Lambda(:,1), 'b','linewidth', 2)
plot(Lambda(:,2), 'b','linewidth', 2)
```

the code for the plot;

```
        dstar2 = ((sol.y(5,1))./(2.*gamma));
```

```
        dstar3 = (sol.y(6,1))./(2.*gamma);
        dstar=[dstar2,dstar3];
```

the code where we extracted the values for the disturbance, inserted into *bvp_law* after the calculus of *ustar*, considering *dstar* as a global variable;

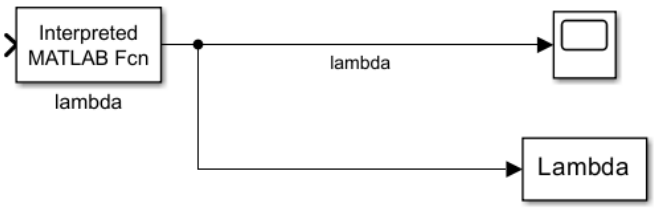

Figure 3.35: Box for disturbance - Simulink

the figures of boxes added inside *bvp_law* box in Simulink, where the MATLAB interpreted function is given by:

```
function d_star = lambda_fun()
global dstar flag
if flag == 2
    d_star = dstar;
else
    d_star = 0;
end
end
```

3.3 Application of APF together with robust NMPC

In this final section of the work, we aim to introduce a technique known as Artificial Potential Fields (APF) method. As previously outlined in the Introduction 1, APF offers a viable approach to facilitating the formation of Unmanned Ground Vehicles (UGVs) while reducing the risk of collision.

3.3.1 Implementation and Simulation of APF with robust NMPC

The idea of implementing and simulating Artificial Potential Fields (APF) method lies in its union with robust Nonlinear Model Predictive Control (NMPC), given the global presence of disturbances in real-world scenarios.

Implementation of APF

First of all, we delineated the modifications made in Simulink. In contrast to the configurations depicted in figure 3.7, an augmentation has been introduced within the *bvp_law* box. Specifically, the coordinates of neighboring drones have been integrated, as depicted in figure 3.36.

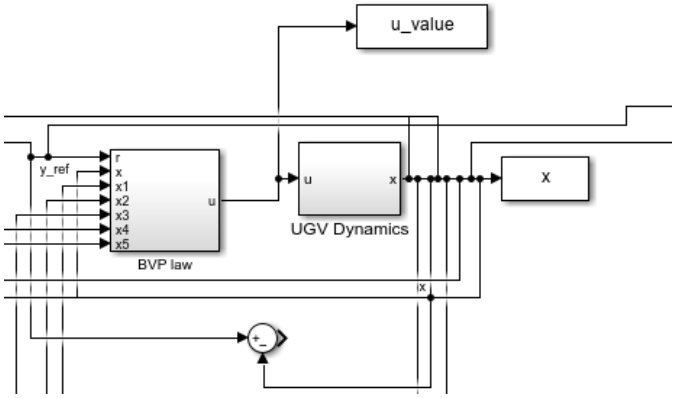

Figure 3.36: Simulink addition for APF

Within the *bvp_law* box, the transition occurs from the representation depicted in Figure 3.8 to that shown in Figure 3.37.

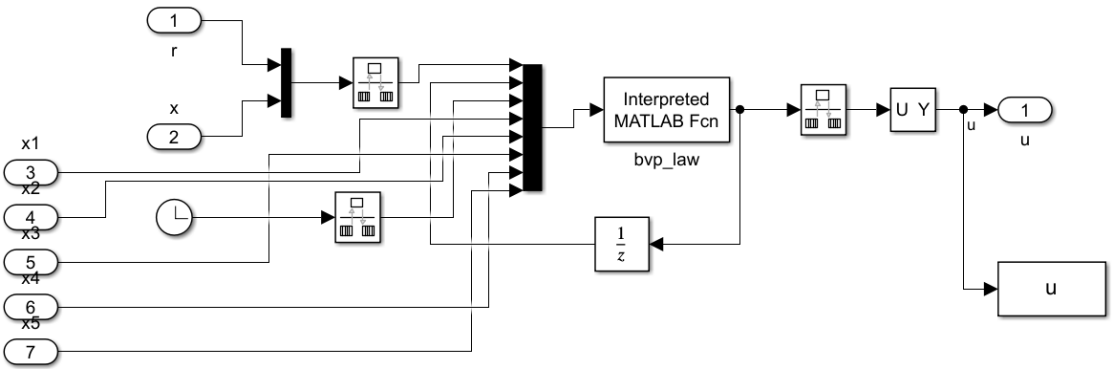

Figure 3.37: Simulink - modified BVP law

Subsequently, we have introduced a function within the function inside the *bvp_law* box to generate a repulsive force to be applied to the coordinates of a specified drone when a connection exists between this drone and another one. This function is described as follows:

```
function repulsiveForce= apf(dronePos1,dronePos2)
 radius=1;
 repulsionIntensity=5;
 distance=norm(dronePos1-dronePos2);
 if distance<radius
      forceMagnitude=repulsionIntensity*exp(-distance);
      forceDirection=(dronePos1-dronePos2)/distance;
      repulsiveForce=forceMagnitude*forceDirection;
 else
      repulsiveForce=[0,0];

 end
end
```

The repulsion force is determined by the following function:

$$F_{repulsion} = \begin{cases} me^{(-d)}(\frac{x1-x2}{d}) & \text{if } d \leq r \\ 0 \quad \text{otherwise} \end{cases}$$

where $x1$ and $x2$ represent the positions of the considered drones, d denotes the distance between the drones, calculated as $||x1 - x2||_2$, m represents the intensity of repulsion and r is the ray fixed.

Finally, within the function *UGV_dyn_aug*, the calculation of the repulsion force is implemented. This calculation incorporates the previously described function and inserts the component along the x-axis of the force into the motion equations.

Simulation of APF with robust NMPC

Simulations were conducted with the same reference as in subsubsection 3.2.2.

The robust case was considered with $\gamma = 4$, determined to be the optimal parameter in subsubsection 3.2.2.

Various scenarios were explored by altering the parameters of the system, specifically the ray r and intensity m previously described.

Cases were tested with $r = 1,2,3.5$, with the assumption that the scenario with $r = 3.5$ represents the worst condition, as it corresponds to a formation ray of 3 units. However, it is useful to determine the positive work of the method.

Furthermore, different intensity levels were assessed with $m = 5,10,17.5,25$. It is evident that higher intensity fosters improved perception among Unmanned Ground Vehicles (UGVs).

The ensuing figures, tables detailing errors, and tables detailing "consumption" delineate these scenarios:

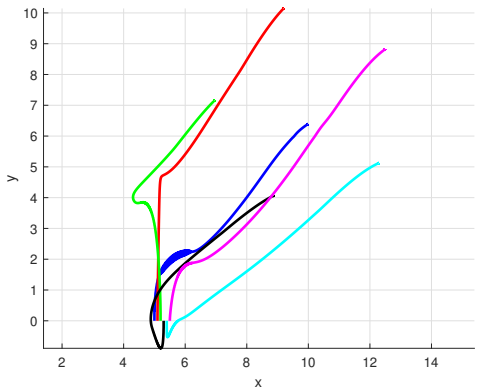

Figure 3.38: Intensity $m = 5$

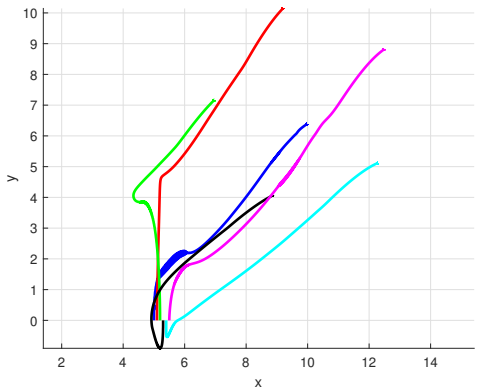

Figure 3.39: Intensity $m = 10$

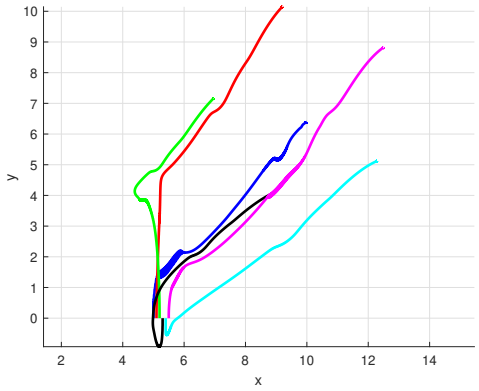

Figure 3.40: Intensity $m = 17.5$

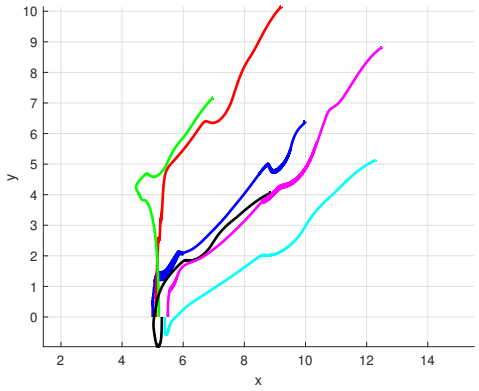

Figure 3.41: Intensity $m = 25$

Figure 3.42: Case APF with $r = 1$

Summing up, the usage of the APF method is evident from graphics and tables. As mentioned earlier, it is noticeable that the case of $r = 3.5$ does not perform well:

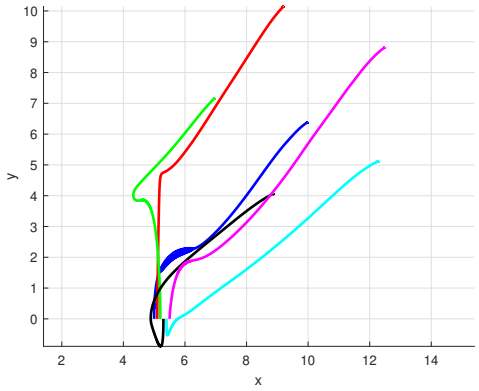

Figure 3.43: Intensity $m = 5$

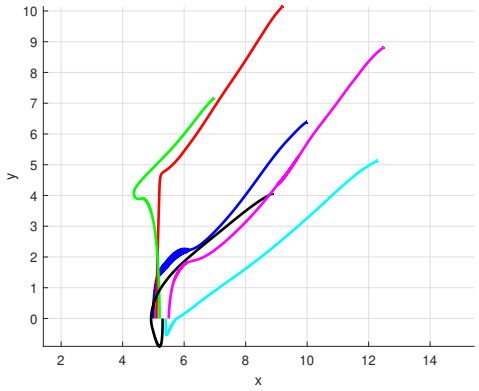

Figure 3.44: Intensity $m = 10$

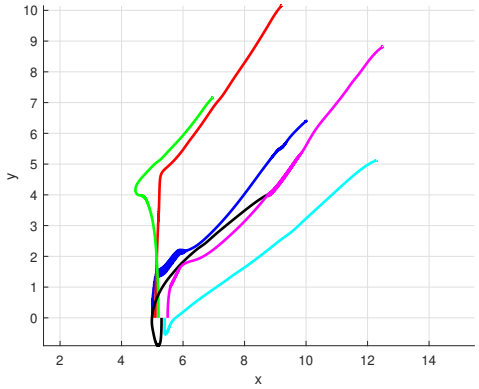

Figure 3.45: Intensity $m = 17.5$

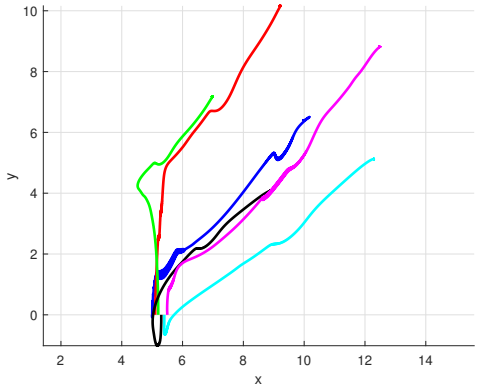

Figure 3.46: Intensity $m = 25$

Figure 3.47: Case APF with $r = 2$

m=5	m=10	m=17.5	m=25
(0.0197,-0.0051,-0.0273)	(0.0180,-0.0051,-0.0273)	(0.0180,-0.0051,-0.0273)	(0.0180,-0.0051,-0.0273)
(0.0267,-0.0051,-0.0273)	(0.0177,-0.0051,-0.0273)	(0.0177,-0.0051,-0.0273)	(0.0177,-0.0051,-0.0273)
(0.0254,-0.0051,-0.0273)	(0.0206,-0.0051,-0.0273)	(0.0206,-0.0051,-0.0273)	(0.0205,-0.0051,-0.0273)
(0.0180,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)
(0.0177,-0.0051,-0.0273)	(0.0268,-0.0051,-0.0273)	(0.0267,-0.0051,-0.0273)	(0.0267,-0.0051,-0.0273)
(0.0205,-0.0051,-0.0273)	(0.0254,-0.0051,-0.0273)	(0.0254,-0.0051,-0.0273)	(0.0254,-0.0051,-0.0273)

Table 3.7: Errors - APF with $r = 1$

there are high errors as results, although the consumption is not correspondingly high. However, in the other cases, with different rays and different intensity levels, the performance is quite good. It is also evident that an intermediate repulsion does not necessarily imply an amplification in "consumption"; instead, it can also

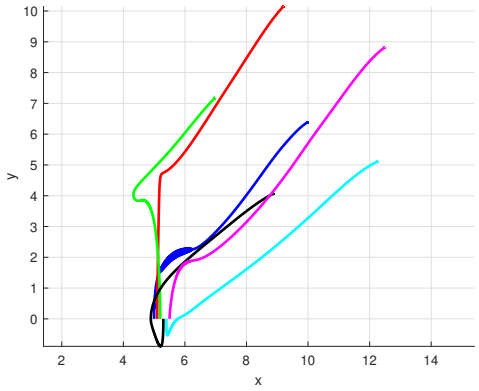

Figure 3.48: Intensity $m = 5$

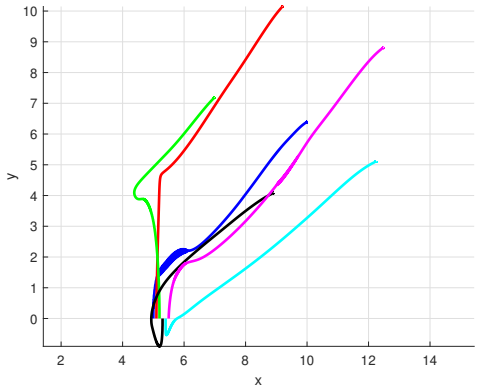

Figure 3.49: Intensity $m = 10$

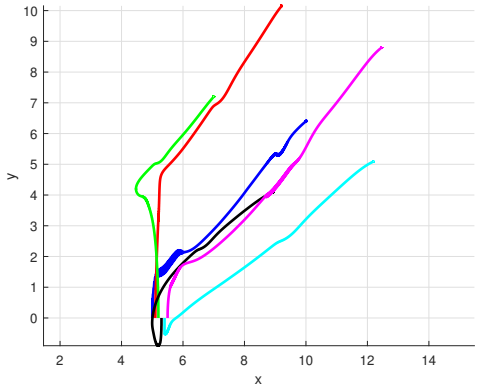

Figure 3.50: Intensity $m = 17.5$

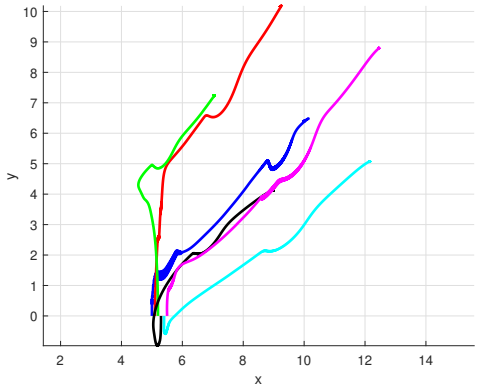

Figure 3.51: Intensity $m = 25$

Figure 3.52: Case APF with $r = 3.5$

m=5	m=10	m=17.5	m=25
(0.0180,-0.0051,-0.0273)	(0.0180,-0.0051,-0.0273)	(0.0180,-0.0051,-0.0273)	(0.0179,-0.0051,-0.0273)
(0.0177,-0.0051,-0.0273)	(0.0177,-0.0051,-0.0273)	(0.0177,-0.0051,-0.0273)	(0.0176,-0.0051,-0.0273)
(0.0206,-0.0051,-0.0273)	(0.0206,-0.0051,-0.0273)	(0.0205,-0.0051,-0.0273)	(0.0205,-0.0051,-0.0273)
(0.0197,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)	(0.0197,-0.0051,-0.0273)
(0.0268,-0.0051,-0.0273)	(0.0268,-0.0051,-0.0273)	(0.0267,-0.0051,-0.0273)	(0.0266,-0.0051,-0.0273)
(0.0254,-0.0051,-0.0273)	(0.0254,-0.0051,-0.0273)	(0.0254,-0.0051,-0.0273)	(0.0253,-0.0051,-0.0273)

Table 3.8: Errors - APF with $r = 2$

yield good performance.

m=5	m=10	m=17.5	m=25
(0.0048,-0.0051,-0.0273)	(-0.0163,-0.0051,-0.0273)	(0.0110,-0.0051,-0.0273)	(-0.0786,-0.0050,-0.0272)
(0.0373,-0.0051,-0.0273)	(0.0585,-0.0051,-0.0273)	(0.0264,-0.0050,-0.0273)	(0.1649,-0.0050,-0.0271)
(0.0265,-0.0051,-0.0273)	(0.0330,-0.0051,-0.0273)	(-0.0113,-0.0050,-0.0273)	(0.0574,-0.0051,-0.0273)
(0.0175,-0.0051,-0.0273)	(0.0151,-0.0051,-0.0273)	(-0.0457,-0.0050,-0.0273)	(0.0065,-0.0051,-0.0273)
(0.0267,-0.0051,-0.0273)	(0.0267,-0.0051,-0.0273)	(0.1075,-0.0050,-0.0273)	(0.0266,-0.0051,-0.0273)
(0.0156,-0.0051,-0.0273)	(0.0053,-0.0051,-0.0273)	(0.0443,-0.0051,-0.0273)	(-0.0287,-0.0051,-0.0272)

Table 3.9: Errors - APF with $r = 3.5$

m=5	m=10	m=17.5	m=25
479.0979	505.2509	699.6652	950.2264
268.8904	289.8247	357.9049	545.2180
445.3716	443.4987	437.3584	430.7935
335.6292	332.2154	331.3976	347.4342
290.2688	299.1804	314.6031	367.9954
324.6251	409.8196	552.6861	752.3440

Table 3.10: "Consumption" - APF with $r = 1$

m=5	m=10	m=17.5	m=25
484.0819	495.2283	583.0073	791.4292
274.1915	290.1431	353.9051	532.1420
447.6367	435.5248	420.3880	414.3952
338.0854	329.6243	323.0639	345.6368
296.3677	302.4533	316.7385	381.4621
326.3702	383.2978	501.7252	586.5063

Table 3.11: "Consumption" - APF with $r = 2$

m=5	m=10	m=17.5	m=25
479.8665	488.5016	630.4948	911.6613
272.1435	285.7205	338.3536	498.7317
445.1032	430.8560	404.3890	401.8973
335.8302	325.0471	309.7783	332.1906
292.2752	294.6156	297.2111	351.3405
323.2664	380.4826	484.4003	631.6077

Table 3.12: "Consumption" - APF with $r = 3.5$

Chapter 4

Conclusion and Future Works

In this thesis, the implementation of Nonlinear Model Predictive Control (NMPC) on a network of six Unmanned Ground Vehicles (UGVs) for formation control was considered. The selection of the network's topology started with deliberation on various structures, ultimately opting for a Master-Slave architecture. Although alternative topologies were explored, including non-shift invariant configurations, the inherent complexity led to the adoption of the aforementioned structure.

Subsequently, NMPC was employed to optimize trajectory minimization for individual UGVs in undisturbed conditions. Formation of the UGVs was constructed with the master drone serving as a reference point, with follower drones adjusting their coordinates relative to the master drone to form a regular polygonal arrangement.

Following this initial phase, robust NMPC was introduced to address disturbances, such as friction and wind, affecting the motion equations of the UGVs. The formation process was repeated under these perturbed conditions, contrasting the outcomes with the ideal scenario governed solely by NMPC and the disturbed scenario employing the initial NMPC formulation. The findings unequivocally supported the superiority of robust NMPC in mitigating disturbances.

Finally, trying to enhance realism, the Artificial Potential Field (APF) method was integrated to prevent collisions between drones. This addition augmented the fidelity of the simulations, knowing that there were not other spatial entities.

4.1 Future Works

Exploring diverse topologies to map the network of Unmanned Ground Vehicles (UGVs) presents a possible approach for further investigation. These topologies could extent structures similar to those detailed in subsection 3.1, where we conducted simulations to analyze dynamic topologies over time. Alternatively, searching into simpler network configurations differently from the conventional master-slave paradigm can offer another interesting approach. For instance, adopting a method like to the one outlined in **Sarras et al.** [**25**], which examines consensus dynamics within the network framework, holds potential for new explorations.

Another possible direction involves revising the structure of motion equations governing UGV behavior. In this thesis, we opted for a simplified point model for simplicity.
However, expanding this model to incorporate more complex representations, such as rigid body dynamics, could significantly enhance the sense of our simulations. For instance, **Khan et al.** [**26**] (figure 4.1) presents a detailed model of a UGV with two wheels, while **Guivant et al.** [**27**] (figure 4.2) offers insights into UGVs with four wheels, illustrating the range of possibilities for refining our modeling approaches.
By adopting such sophisticated models, we can attempt towards more realistic applications of formation control algorithms on UGVs, manipulating the theoretical considerations elucidated in preceding sections to drive practical advancements in autonomous vehicle technology.

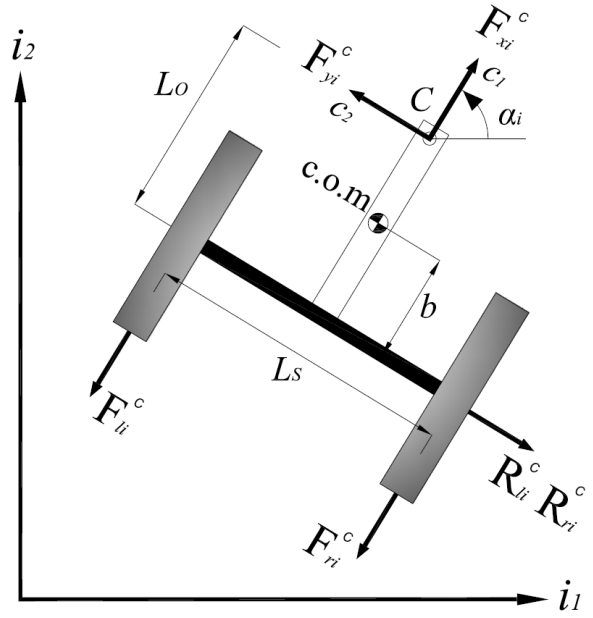

Figure 4.1: Example of future work - UGV with two wheels

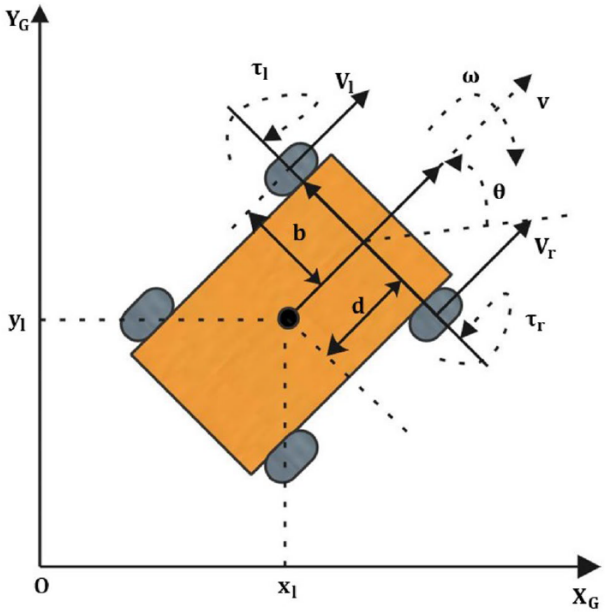

Figure 4.2: Example of future work - UGV with four wheels

Bibliography

[1] Yuanchang LIU Richard BUCKNALL. «A Survey of Formation Control and Motion Planning of Multiple Unmanned Veichles». In: (2013) (cit. on pp. 1, 2, 4).

[2] Yuanchang LIU Richard BUCKNALL. «Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment». In: (2015), pp. 126–144 (cit. on p. 3).

[3] Yehui SONG Guoru DING Jiachen SUN Jinghua LI Yitao XU. «Topology tracking of dynamic UAV wireless networks». In: *Chinese Journal of Aeronautics* (2021) (cit. on p. 3).

[4] Anna GUERRA Davide DARDARI Petar M. DJURI. «Dynamic Radar Network of UAVs: A Joint Navigation and Tracking Approach». In: (2020) (cit. on p. 3).

[5] Diego PASINI Charles JIANG Marie-Pierre JOLLY. «UAV and UGV Autonomous Cooperation for Wildfire Hotspot Surveillance». In: *IEEE* (2019) (cit. on p. 3).

[6] Minsoo PARK SeungGwan LEE Sungwon LEE. «Dynamic Topology Reconstruction Protocol for UAV Swarm Networking». In: (2020) (cit. on p. 4).

[7] Ankit A. RAVANKAR Abijheet RAVANKAR Takanori EMARU and Yukinori KOBAYASHI. «HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots». In: *IEEE* (2020) (cit. on p. 5).

[8] Yiqun DONG Youmin ZHANG Jianliang AI. «Experimental Test of Artificial Potential Field-Based Automobiles Automated Perpendicular Parking». In: *Abdelaziz Bensrhair* (2016) (cit. on p. 5).

[9] Rafael Monteiro Jorge Alves SOUZA Gabriela Vieira LIMA Aniel Silva MORAIS Luís Cláudio OLIVEIRA-LOPES Daniel Costa RAMOS Fernando Lessa TOFOLI. «Modified Artificial Potential Field for the Path Planning of Aircraft Swarms in Three-Dimensional Environments». In: *sensors* (2022) (cit. on p. 5).

[10] Mauro MANCINI Nicoletta BLOISE Elisa CAPELLO Elisabetta PUNTA. «Sliding Mode Control Techniques and Artificial Potential Field for Dynamic Collision Avoidance in Rendezvous Maneuvers». In: *IEEE CONTROL SYSTEMS LETTERS* (2020) (cit. on p. 5).

[11] James A. PRIMBS. «Handbook of Model Predictive Control». In: (2018) (cit. on p. 6).

[12] Olaf STURSBERG Zonglin LIU. «On the Use of MPC Techniques to Decide Intervention Policies against COVID-19». In: *Elsevier* (2021), pp. 476–481 (cit. on p. 6).

[13] Stefano DI CAIRANO Ilya V KOLMANOVSKY. «Automotive Applications of Model Predictive Control». In: (2019) (cit. on p. 6).

[14] Michele PAGONE Mattia BOGGIO Carlo NOVARA Anton PROSKURNIKOV Giuseppe C. CALAFIORE. «A barrier function approach to constrained Pontryagin-based Nonlinear Model Predictive Control». In: (2022) (cit. on p. 6).

[15] Michele PAGONE Lorenzo ZINO Carlo NOVARA. «A Pontryagin-based Game-theoretic Approach for Robust Nonlinear Model Predictive Control». In: (2023) (cit. on pp. 6, 29, 32).

[16] Alberto ZAFFARONI. «FUNZIONI VETTORIALI NON DIFFERENZIABILI». In: *Dottorato di Ricerca in Matematica Applicata ai Problemi Economici - UniTs* () (cit. on p. 16).

[17] Jr. ARTHUR E. BRYSON. «Applied Optimal Control». In: *Taylor Francis Group, New York - London* (1975) (cit. on p. 16).

[18] Alberto BRESSAN. «Noncooperative Differential Games. A Tutorial». In: (2012) (cit. on pp. 17–19, 21–28).

[19] OSBORNE MARTIN Ariel RUBINSTEIN. «A Course in Game Theory». In: *The MIT Press* (1994) (cit. on p. 23).

[20] Nash JHON. «Non-Cooperative Games». In: *Annals of Mathematics* (1951), pp. 286–295 (cit. on p. 25).

[21] Ibrahim KHALID. «Anti-Jamming Game to Combat Intelligent Jamming for Cognitive Radio Networks». In: *IEEE* (2021) (cit. on p. 25).

[22] BAGGS IVAN. «Functions with a closed graph». In: *Proceedings of the American Mathematical Society* (2020), pp. 439–442 (cit. on p. 25).

[23] URSESCU CORNELIU. «Multifunctions with convex closed graph». In: *Czechoslovak Mathematical Journal* (1975), pp. 438–441 (cit. on p. 25).

[24] Younggeun YOO. «KAKUTANI'S FIXED POINT THEOREM AND THE MINIMAX THEOREM IN GAME THEORY». In: (2016) (cit. on p. 25).

[25] Pelin SEKERCIOGLU Ioannis SARRAS Antonio LORIA Elena PANTELEY Julien MARZAT. «Bipartite Formation over Undirected Signed Networks with Collision Avoidance». In: *HAL open science* (2023) (cit. on p. 80).

[26] Imad KHAN Matthew SPENKO. «Dynamics and Control of an Omnidirectional Unmanned Ground Veichle». In: *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems October 11-15, 2009 St. Louis, USA* (2009) (cit. on p. 80).

[27] Subhan KHAN Jose GUIVANT. «Fast nonlinear model predictive planner and control for an unmanned ground vehicle in the presence of disturbances and dynamic obstacles». In: *nature - scientific reports* (2022) (cit. on p. 80).