DEGREE PROJECT

Ensuring safe docking maneuvers on floating platform using Nonlinear Model Predictive Control (NMPC)

Federico Gatti

Space Engineering, master's level (120 credits)
2024

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

[This page intentionally left blank]

Master's Thesis

Ensuring safe docking maneuvers on floating platform using Nonlinear Model Predictive Control (NMPC)

Department of Computer Science, Electrical and Space Engineering

Luleå University of Technology

Author: Federico Gatti
Supervisor: Sumeet Gajanan Sapute
Examiner: Soheil Sadeghi

21st February 2024

Abstract

Docking maneuvers are a relevant part of the modern space mission, requiring precision and safety to ensure the success of the overall mission.
This thesis proposes using a non-linear Model Predictive Control (MPC) as a controller with various constraints to ensure safe docking maneuvers for a satellite. This was done in MATLAB using as a model for the satellite the Sliders used by the Robotics Lab at Luleå University of Technology (LTU). The controller was tested first on the MATLAB model and then briefly on hardware.
The main objective of this thesis is to develop and implement an MPC-based control strategy to achieve safe docking maneuvers between two satellites. Great attention has been paid to implementing constraints, such as collision avoidance, and hardware constraints, such as thrust limits, to ensure the safety and reliability of the process.
Through the MATLAB simulations, it was possible to indicate that the introduced constraints contribute significantly to the safe execution of docking maneuvers, preventing collisions, and optimizing fuel usage. The controller successfully adapts to unforeseen disturbances and uncertainties in real-time, showcasing its robustness and reliability in dynamic space environments.
The hardware simulations have shown that the controller operates as expected but needs further tuning to adapt to the hardware uncertainties.
In conclusion, this thesis comprehensively explores MPC-based control strategies with constraints for space docking maneuvers. The positive results underscore this approach's potential to ensure the safety and reliability of future space missions, opening avenues for further research and application in autonomous space systems.

Contents

List of Figures

List of Tables

Acronyms

CAS Collision Avoidance System. 12, 15, 19, 21, 24, 36

DOF Degree of Freedom. 2, 3

GNC Guidance, Navigation and Control. 1, 2

IOS In-Orbit Servicing. 1

LQR Linear Quadratic Regulator. 1, 2

LTU Luleå University of Technology. i, iii, 2, 3

MPC Model Predictive Control. i, ii, 1, 2, 6, 7, 11, 36, 37, 42, 44, 46

NMPC Nonlinear Model Predictive Control. 1, 2, 4, 5, 7, 8, 10, 12

PD Proportional Derivative. 1, 2

PID Proportional Integrative Derivative. 2

PWM Pulse Width Modulation. 6, 35, 44

PWPF Pulse Width Pulse Frequency. 1

RDM Rendezvous and Docking Manoeuvres. 1

ROS Robot Operating System. iii, 35, 36

1 Introduction

Rendezvous and Docking Manoeuvres (RDM) are a set of operations that allow an active spacecraft (Chaser) to approach and capture a second one (Target). These operations are crucial for many space missions, such as In-Orbit Servicing (IOS). These space from refueling and maintenance of spacecraft to orbital debris removal and periodical missions to the ISS. The first RDM has been performed by the Gemini VIII mission on March 16^{th} of 1966 and was performed entirely manually [1]. Nowadays, these maneuvers are performed almost entirely by computers thanks to the advent of new technologies and due to the complexity of new spacecraft. A good example is the cargo spacecraft such as the latest SpaceX Dragon, HTV, and Progress that are able to perform autonomously the RDM with ISS thanks to advanced control strategies.
These kinds of maneuvers are vital for actual space missions and will be even more relevant in the future. For example, more complex structures, such as the ISS, are being developed, requiring modular design. This construction method requires an extensive use of docking maneuvers to connect the various modules one to another. Furthermore, the issue that is receiving more and more attention from the space industry is space debris. The increase of human activity in orbit has increased their amount to a level that could lead to a chain reaction that will fill the entire LEO region [2] with so many small particles and fragments that any further operations become unfeasible [3]. This scenario is called Kessler Syndrome [4]. There are two types of strategies to remove space debris: active and passive. While the latter is used for small-sized debris, the former is aimed at bigger debris and is more related to the work presented here. This is because they involve docking maneuvers to grab the debris and de-orbit it. A detailed explanation of the latest state-of-the-art strategies can be found at [5]. Therefore, the hardest challenge these strategies encounter is how to properly approach and dock to the debris, which demands developing highly reliable autonomous docking systems.

Performing RDM represents a highly intricate task for a spacecraft, demanding exceptional precision and the synchronization of various onboard systems. Every mission requires a different level of automation and precision, and its complexity varies based on the operational context and the spacecraft's environment. For example, missions such as manned expeditions or those involving non-cooperative targets demand a significant degree of precision and control effort. Therefore, different sets of Guidance, Navigation and Control (GNC) systems must be appropriately selected for each mission. This is challenging when selecting the control strategies. Different instruments are used to improve the robustness and reliability of the controllers, by providing more data and information to the controller. Some use vision-based navigation systems [6] and other radio-based ones.

1.1 Docking maneuvers control strategies

The most critical phase of the RDM is the last one when the Chaser and Target are just a few meters of distance from each other. In fact, the Chaser has to keep its motion synchronized with the Target until it docks to it. These maneuvers at such short distances demand high precision, and therefore, the margin of error is reduced. An effective control strategy is needed to ensure the success of the docking phase.
Different control laws are used to define a controller, and each of them can be tuned to optimize a certain aspect of the mission, from fuel consumption to the time allowed to perform the RDM. Controller based on Linear Quadratic Regulator (LQR) and Proportional Derivative (PD) have been studied and verified in [7] without time constraints. In [8], The authors compare different guidance trajectories using the control laws from [7], analyzing important parameters such as time and fuel consumption without investigating the optimality of the results. In [9], a solution is presented based on MPC without including attitude control. A similar approach has been studied in [10], but a Pulse Width Pulse Frequency (PWPF) modulator has been integrated into

the model. A different approach in which polynomial functions have generated the trajectory components has been used in [11]. Another type of controller that has been studied is the MPC. This controller has the ability to handle constraints on state vectors. The authors in [9] used a MPC to control the maneuvers of a spacecraft capturing a non-rotating and a rotating target in a planar environment.
The choice between these controllers depends on the specific application and system dynamics. Each one of them has their own strengths and weaknesses. PD are simple and effective for systems where quick response to changes is required but struggle with complex dynamics. Proportional Integrative Derivative (PID), thanks to the integral term, are suitable for a wide range of systems, especially those with unknown or variable disturbances. The downside is that they require the linearization of the system and are focused on reducing error rather than optimizing a cost function, which is the focus of this work. LQR are more suitable for nonlinear systems, and using a space state representation, they aim to minimize a cost function. This makes this type of controller more robust and efficient compared to the PID. MPC controllers predict future system behavior based on a dynamic model and optimize control actions over a finite prediction horizon. This is the main difference with LQR. They optimize the new solution for a finite time window rather than the whole simulation time. This approach is more robust and versatile and can handle both linear and nonlinear systems and constraints on inputs and states. MPC are commonly used in advanced process control, robotics, and autonomous systems. Given the nonlinear nature of the system used in this paper, the choice goes to the NMPC controller, a nonlinear version of MPC. According to previous research on similar case scenarios, the most stable and efficient controller for GNC systems in autonomous vehicles is the MPC. The works supporting this choice can be found in [12] and [13].

1.2 Testing Platforms

Ensuring the robustness and reliability of these systems requires extensive testing that has to be performed in a microgravity environment. This process, though, represents a challenge since it must be kept cost-effective and low-risk. In fact, recreating the frictionless environment found in space is critical and needs to be replicated in hardware in the loop test benches. This has led to the development of different techniques: from parabolic flight [14] to drop towers [15] and from underwater facilities [16] to counterweight ones [17]. All of these are affected by different constraints that make them suitable only for very restricted applications. For example, parabolic flights and drop-towers are limited by flight/drop time and are expensive. Underwater facilities are not suited for satellites and counterweight systems introduce disturbances in the system.
Therefore, the planar air-bearing platforms are the most suitable systems for cost-effective and low-cost simulations. Their only limitation is that only 3 Degree of Freedom (DOF) simulations can be conducted on them. The air-bearing platforms allow not only to validate GNC systems software but also their hardware implementation. Different space companies and research institutes have developed their own systems. NASA, for example, developed SPHERES that can be both used on floating platforms and directly in orbit on the ISS [18]. ESA has the ORBIT platform in which several tools can be used [19]. DLR developed the EPOS platform [20]. A comprehensive list of them can be found at [21].

1.3 Sliders and floating platform

In this thesis, it will be used the floating platform Figure 1 that is situated in Kiruna at the LTU Space Avionics Lab [22]. The spacecraft will be simulated using a slider, which is a robot developed by the Robotics Lab at LTU [23]. Hence, the work will be performed in a 2D environment with 3 DOF.

The advantage of using this kind of platform is that even though limited to only 3 DOF it can recreate the micro-gravity conditions present in the space environment. The planar and frictionless environment can be used to replicate the majority of scenarios for space operations and thus, rigorously validate software and relative hardware-based implementations.
The frictionless bench is an epoxy-topped table that has a 4 $\times$ 4 m area but considering the sliders' dimensions and safety constraints, it reduces to an effective working area of 3 020 mm $\times$ 3 015 mm in which the sliders can operate.

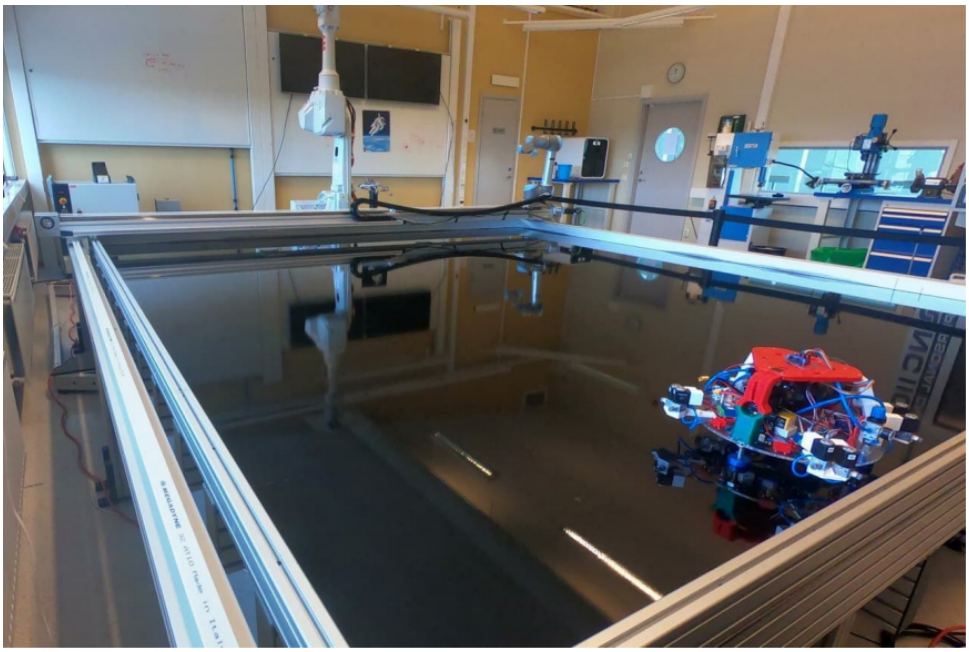


Figure 1: Floating platform at LTU Kiruna Space Campus. Sliders on the bottom right corner.

The model representing the satellite used both in the software and hardware simulations was based on the floating platform developed by the LTU Robotics Department called Slider (Figure 2). Its dynamics are represented by the Equation 1. This platform, integrated with sensors and actuator units, can accurately simulate the friction-less motion of a spacecraft in the space environment. It is supported by three air bearings that allow it to levitate on the table. It moves on the surface by using the 8 thrusters mounted on 4 different brackets. They operate in an on-off mode to simulate the mono-propellant thrusters which are commonly used for the satellites' attitude control corrections. In-depth information about the Slider can be found at [23]. Two of these sliders have been used and their parameters are the following:

Parameters	Values
Mass	4.436 kg
Moment of Inertia	1.092 $\frac{kg}{m^2}$
System Propagation time step	0.01 s
Minimum on time of thrusters	0.01 s
Thrust force interval	0-0.7 N

Table 1: Slider parameters

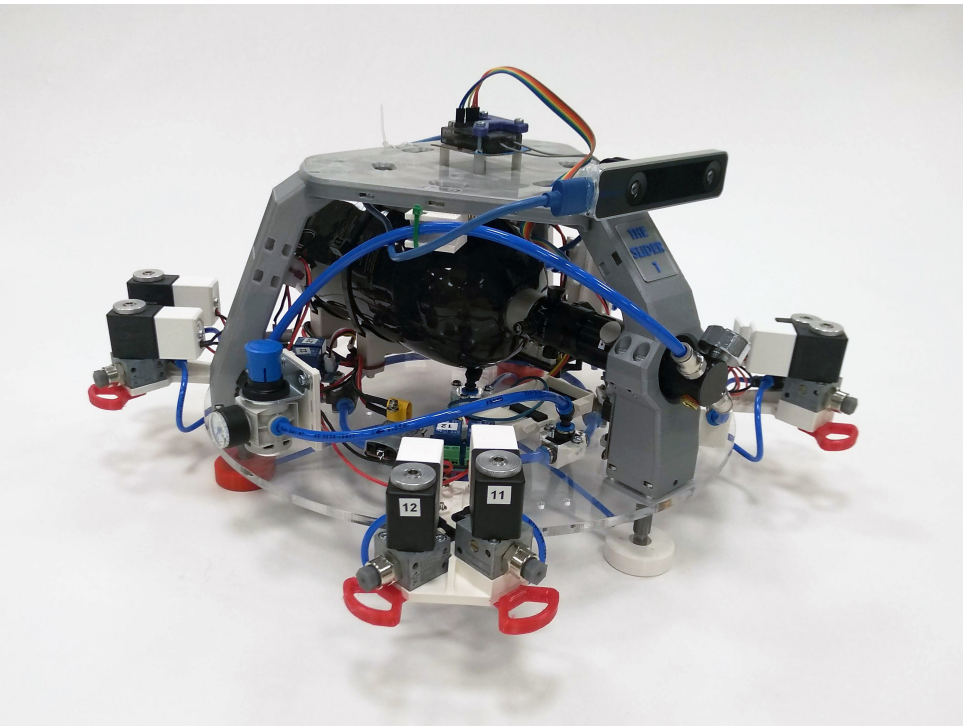


Figure 2: Slider platform

1.4 Thesis aim

The present thesis will investigate a control strategy based on a NMPC controller to optimize fuel consumption for a satellite approaching and docking with another one. The optimization problem will focus on reducing the amount of thruster activations by reducing a cost function. This is presented in 2.5. The problem has been divided into two main parts: one in which the Target was stationary and one in which it was moving.
The first part consisted of only one case in which the Target was fixed in one position, and the Chaser had to reach it and dock to it. The second part consisted of three cases where the Target moved and/or rotated. This has been done to simulate some of the possible scenarios in which a spacecraft might find itself when docking with another one.
Throughout the thesis, it will be described which challenges occurred while defining such a problem and how they have been overcome. The challenges found while defining the controller were to properly define the mathematical model of the Slider and state the proper constraints to make the maneuvers as safe as possible.
In fact, it is crucial to have a mathematical model of the system as accurate as possible to allow the NMPC to predict the behaviour of the system as accurately as possible. The other challenge that had to be faced was defining the correct constraints inside the controller to ensure the Chaser was safely docked to the Target.

1.5 Thesis structure

The thesis is structured as follows:
Section 2 will introduce the experiment platform and the problem the thesis aims to solve.
Section 3 will explain how the problem has been solved, i.e., how the controller has been designed.
Section 4 will show how the problem has been formulated on MATLAB and then executed both on software and hardware.
Finally, the results are discussed in section 5.

2 Modelling and Problem formulation

The thesis is based on two main things: the platform in which the simulations and tests are performed and the type of controller. As mentioned in 1.3, the model of the system has been based on the characteristics of the Sliders. The model will be described in detail in subsection 2.1. After reviewing many papers seen in subsection 1.1, the controller selected for this paper is the NMPC. Its functioning and how it has been set for this work will be described in subsection 2.2

2.1 Model of the system

Based on the work done in [23] the equations that describe the system dynamics are:

$$\begin{cases} \dot{x} & = V_x \cos\theta - V_y \sin\theta \\ \dot{y} & = V_x \sin\theta + V_y \cos\theta \\ \dot{\theta} & = r \\ \dot{V_x} & = V_y r + \frac{f_x}{m} \\ \dot{V_y} & = -V_x r + \frac{f_y}{m} \\ \dot{r} & = \frac{\tau}{I_{zz}} \end{cases} \tag{1}$$

where $\dot{x}$ and $\dot{y}$ are the velocities along the x and y axis, τ is the torque, f_x, f_y and m are the forces produced by the thrusters and the slider's mass. I_{zz} is the Slider's inertia along the z-axis.

The Slider's actuation unit is equipped with eight small thrusters and its control action. i.e. forces (f_x, f_y) and torque τ components, is modeled as:

$$\begin{cases} f_x & = \sum_{k=1}^{8} T_k \cos\beta_k \\ f_y & = \sum_{k=1}^{8} T_k \sin\beta_k \\ \tau & = \left(\sum_{k=1}^{8} \left(T_k r^y_{T_k} \cos\beta_k - T_k r^x_{T_k} \cos\beta_k \right) \right) \end{cases} \tag{2}$$

where T_k is the constant thrust magnitude, $r^y_{T_k}$ and $r^x_{T_k}$ are the position of the k^{th} thrusters in the $X_B Y_B$ plane (Figure 4) and β_k is the thrusters orientation with respect to the X_B axis. The complete thrusters' logic and actuation are represented in Figure 3.

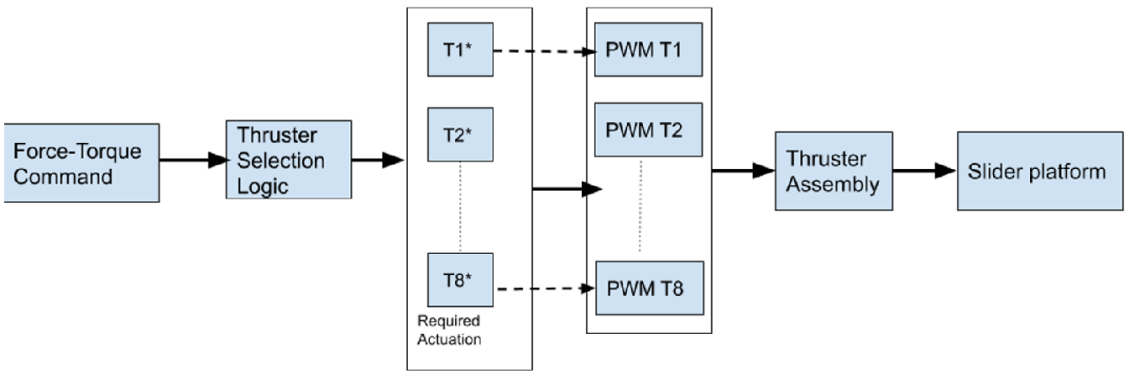


Figure 3: Representation of the thrusters' activation logic.

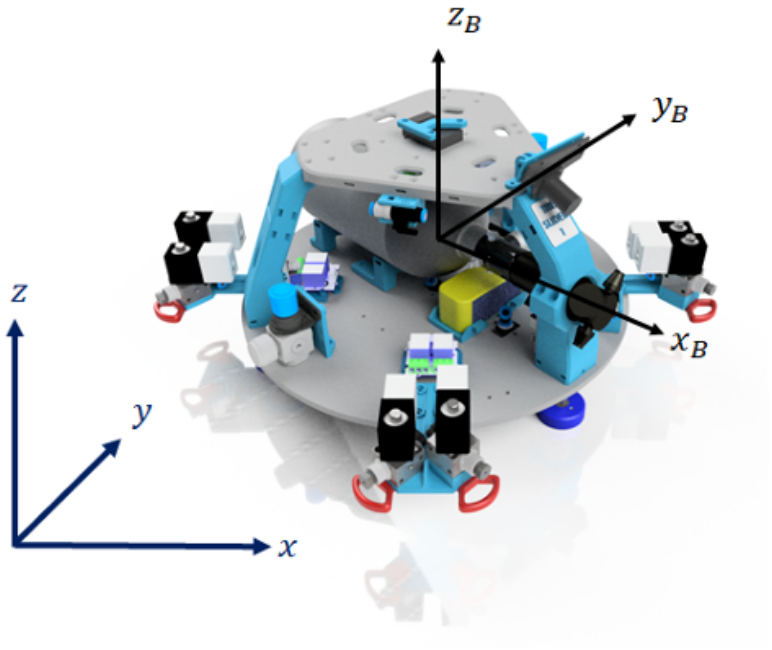


Figure 4: Sliders' reference frames

A thruster selection logic is necessary to determine which set of thrusters has to be activated in order to provide the necessary commanded thrust and torque. Due to the on-off nature of the thrusters, a Pulse Width Modulation (PWM) technique had to be incorporated. The PWM generates a sequence of pulses in such a way that the average thrust produced by each thruster closely matches the required magnitude T_k.
The optimization problem is shown in Chapter V of [23].

2.2 Nonlinear Model Predictive Controller (MPC)

The MPC is a class of algorithms that compute a sequence of manipulated variable adjustments to optimize the future behaviour of the plant. In fact, it is based on iterative, finite-horizon optimization of a plant model (Figure 5). At time t the current plant state is sampled and a cost-minimizing control strategy is computed (via a numerical minimization algorithm) for a relatively short time horizon in the future. This is defined as an open-loop optimal sequence of control moves. Only the first step of the control strategy is implemented, then the plant state is sampled again and the calculations are repeated starting from the new current state, yielding a new control and new predicted state path. The prediction horizon keeps being shifted forward and for this reason MPC is also called receding horizon control. Due to the repeated optimization, MPC is considered a closed-loop approach.

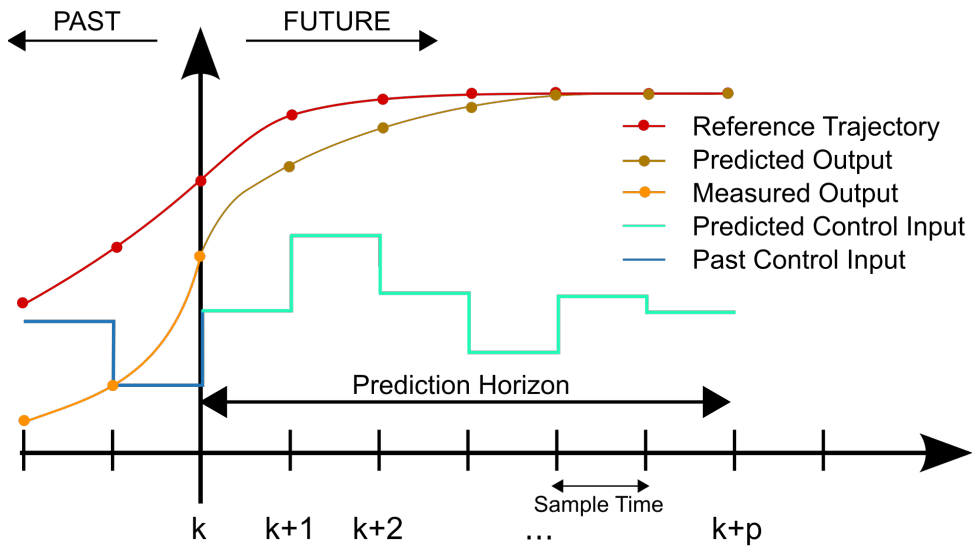


Figure 5: MPC Discrete scheme

A NMPC is a variant of the MPC that is used when the system dynamics are not linear, such as the one in this thesis, meaning that their behavior cannot be accurately represented by linear models. Due to the nonlinearities, it is necessary to use more complex numerical methods to solve the optimization problem. This makes the NMPC more computationally demanding than the MPC but, at the same time, the only controller capable of handling nonlinear inputs and constraints such as thruster saturation or collision avoidance limits.
In this thesis, due to the nonlinear nature of the system dynamics (1), the NMPC will be used to optimize the slider's thrusters' activation while reaching and docking to a second slider. In fact, the controller will find the most efficient way to move from point A to point B while considering several constraints.
The constraints that the controller has to take into account are several: obstacle positions, thruster activation rate, and collision avoidance. These will be explained in detail in the subsection 2.3
Following the notation used in Figure 6, the NMPC control loop has been set in the following way:

- Inputs:
  - Reference: Target docking door coordinates and orientation (x_T,y_T and θ_T).
  - Feedback: Chaser position and orientation after thruster activation (x_C,y_C and θ_C).
- Constraints:
  - Thrusters thrust and torque (u_b. l_b).
  - Collision avoidance (Cardioid coordinates).
- Output:
  - Control Moves: Thrusters' activation control input (f_x, f_y and τ).

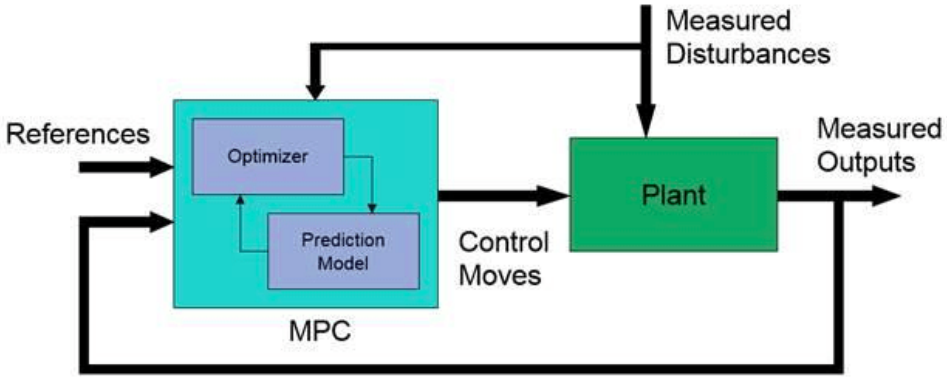


Figure 6: MPC Control Loop

2.3 Constraints

The role of the constraints inside the controller is to set rules that have to be followed while performing the optimization problem. This allows to have parameters optimized within certain limits and therefore, have more control on the optimization process. Sometimes they are necessary to just give the controller information that cannot be described simply by the model of the system. The constraints that has been taken into account in this paper can be grouped into 3 different areas:

- Platform and Slider's limitations
- Motion planning
- Collision avoidance

2.3.1 Platform and Slider's limitations

The controller does not know the characteristics of the platform in which the simulations happens and therefore, it is necessary to provide the physical limitations that the floating platform and the slider have. The first one is the limited space, as said in subsection 1.3, the area in which the slider can move is limited and therefore that has to be taken into account in the NMPC.
The second one is the slider performances, especially regarding the activation thruster rate and the amount of thrust and torque they can produce. Those have been included in the NMPC: the first one when selecting the sampling time dt and the second one by including a lower and upper bound (u_b. l_b) in the cost function J.
Finally, since the sliders can store a limited amount of propellant, there is a limitation of time in which the simulation can run. Hence, a limit of 2 minutes has been set for the simulation running time.

2.3.2 Motion planning

Two different motion planning strategies have been studied to ensure the Chaser properly approaches the Target. This meant having the Chaser always moving at a safe distance from the Target and, at the same time, arriving in front of the docking port in both the right direction and orientation. The strategies that have been investigated are the following:

- Polynomial trajectory
- Cardioid area

Polynomial trajectory

The first strategy consisted of creating a set of waypoints and then connecting them with a polynomial trajectory. This will generate a set of points that can be used as a reference trajectory for the controller. This ensured that the Chaser would follow a trajectory that brought it towards the Target docking port from the right direction and, at the same time avoid any collision with the Target.
This strategy is based on knowing beforehand the Chaser's initial and final position and the Target orientation. Given this information, the other value that had to be given to the algorithm was the duration of the simulation (Δt) and the Chaser's initial and final velocity. With that, the waypoints were generated.
The waypoints have been generated so that the final one would correspond to Chaser's final position and would bring it at $2\,cm$ from Target's docking port. This is to ensure a safe distance when approaching the target.
Once all the waypoints have been generated the trajectory has been generated using a quintic polynomial trajectory defined by the following equations:

$$\begin{cases} x(t) &= at^5 + bt^4 + ct^3 + dt^2 + et + f \\ \dot{x}(t) &= 5at^4 + 4bt^3 + 3ct^2 + 2dt + e \\ \ddot{x}(t) &= 20at^3 + 12bt^2 + 6ct + 2d \end{cases} \tag{3}$$

The difficulty encountered while using this strategy was that depending on the initial Target position and Chaser orientation the number of waypoints could change. In fact, by manually selecting the number of waypoints the shortest trajectory could have been ensured in each case. By design choice, the number could vary from 3 to 5. This strategy has shown consistent results, as can be seen in subsection 3.2. Still, after a few simulations, it was clear that it was unsuitable for a non-collaborative and moving target. In fact, if the Target was moving, a new set of waypoints and, consequently, a new trajectory had to be calculated at each iteration, making the computational load very high. Furthermore, with the introduction of obstacles on the trajectory, ensuring the free-from-collision approach with the moving Target was impossible.

Cardioid area

For the second strategy used in this paper, instead of generating a reference trajectory through waypoints and polynomial equations, the controller had assigned the final point that the Chaser had to reach and a designated area in which the Chaser was prohibited from entering. This area has been defined by a cardioid whose shape ensures that the slider keeps a safe distance from the slider and, at the same time, approaches the target's docking port in the correct direction [24]. In fact, the cardioid has been designed in a way where its origin will coincide with the final point that the Chaser has to reach. This will ensure that the Chaser follows the cardioid's edges and positions itself in the right spot. The following equation defines the cardioid area

$$(x^2 + y^2 + 2bx)^2 = 4b^2(x^2 + y^2) \tag{4}$$

where b is the cardioid radius and determines the area it will cover.
Therefore, the cardioid as a safe barrier function has been implemented in the controller by defining it as a non-linear constraint inside the *fmincon* function. The nonlinearity of the constraint is due to the time-dependent variable on which the constraint itself is defined. It has been implemented in the controller by defining the cardioid as a nonlinear constraint with the following equation,

$$4b^2\left((x - x_{0card})^2 + (y - y_{0card})^2\right) - \left((x - x_{0card})^2 + (y - y_{0card})^2 + 2b(x - x_{0card})\right)^2 \leq 0 \tag{5}$$

where x_{0card} and y_{0card} are the cardioid's origin coordinates and x and y define the Chaser position. As said in the previous subsection, for safety reasons the final point has to be a point at $2\,cm$ from Target's docking port. Hence, the cardioid's origin corresponds to that point.

This strategy has been found suitable for every case studied in this paper. It ensured that the Chaser always approached the Target in the right direction, especially when the Target was moving and rotating. Furthermore, it didn't interfere with the obstacle avoidance constraint.

2.3.3 Collision avoidance

A collision avoidance system is crucial to prevent the chaser from crashing into the target while moving towards it and avoid any obstacles the chaser could encounter. Since it is impossible for the controller to have any information regarding obstacles and how to avoid them from the model of the system, it is necessary to define a robust strategy that will work in any case. In this case, the proper strategy would be the one that ensures that the Chaser does not collide with both the Target and obstacles and that the Chaser approaches the Target in the right direction. The collision avoidance with obstacles has been defined separately from the one to approach and avoid the collision with the target correctly. In fact, for the first one, a simple constraint in which a safe distance has been introduced, while for the second one, the motion planning strategy shown in subsubsection 2.3.2 has been implemented. The collision avoidance constraint has been implemented in the *fmincon* function to prevent the Chaser from hitting any obstacles along the way. The constraint is simple and sets a certain distance at which the slider needs to keep from the obstacle. It is formulated in the following form

$$h(t) = ||\vec{x}_{Chaser} - \vec{x}_{obs}||^2 > d \tag{6}$$

$$d = r_{obs} + r_{slider} + d_{safe} \tag{7}$$

where, x_{Chaser} is the chaser position, $x_{obstacle}$ is the obstacle position, r_{obs} is the obstacle's radius, r_{slider} is the slider's radius and d_{safe} is a safety margin to avoid any contact between the slider and the obstacle.

2.4 Upper and Lower boundaries

Another constraint that had to be implemented in the optimization process is the maximum and minimum thrust (f_x, f_y) and torque (τ) that the onboard thrusters can produce. Therefore, the NMPC optimization loop will have to comply with the following constraint

$$l_b \leq u \leq u_b \tag{8}$$

The l_b and the u_b mean lower and upper bounds, respectively. Both contain the values of f_x, f_y and τ. These will help the controller in its model prediction and return input control values according to Sliders' physical capabilities. The values selected for these constraints can be seen in subsection 3.1.

2.5 Optimization control problem

The aim of the optimization control problem is to minimize the thruster activation while the Chaser approaches and docks to the Target. This has been done by minimizing the following cost function

$$J = (x - x_{ref})^T Q\,(x - x_{ref}) + (u - u_{ref})^T R_u\,(u - u_{ref}) + (u - u_{old})^T R_d\,(u - u_{old}) \tag{9}$$

where

- x and x_{ref}: vector containing respectively the state prediction (x) and the state references for each prediction step (x_{ref})

- u and u_{ref} and u_{old}: vector containing respectively the system inputs of every prediction step (u), references for each prediction step(u_{ref}) and the previous command input (u_{old})
- Q: matrix containing state penalties
- R_u: matrix containing input penalties
- R_d: matrix containing delta input penalties

This has been implemented in the controller using the *fmincon* MATLAB function.

Matrices Q, R_u and R_d play a vital role in the optimization problem because they are essentially the tuning parameters for the control design.
Matrix Q quantifies the importance of minimizing the error between predicted and reference states. The higher the values, the more responsive the controller will be.
Matrix R_u quantifies the deviation between the system inputs and the desired ones. Hence, it ensures the correct amount of inputs are sent to the thrusters, minimizing the deviation.
Matrix R_d quantifies the cost incurred due to changes in the control inputs from one step to the next. It ensures smoothness in the control inputs, aiming to minimize abrupt changes or activations of thrusters.

2.6 MPC problem formulation

With all written above, the MPC non-linear problem can be mathematically defined in the following way

$$
\begin{aligned}
\underset{x,\,u}{\text{minimize}} \quad & J = \frac{1}{2}\sum_{k=0}^{N} \left(x_k - x_k^*\right)^T Q_k \left(x_k - x_k^*\right) \\
& \quad + \frac{1}{2}\sum_{k=0}^{N} \left(u_k - u_k^*\right)^T R u_k \left(u_k - u_k^*\right) \\
& \quad + \frac{1}{2}\sum_{k=0}^{N} \left(\Delta u_k - \Delta u_k^*\right)^T R d_k \left(\Delta u_k - \Delta u_k^*\right) \qquad (10) \\
\text{subject to} \quad & x_0 = x_k^{ref} && k = 0, \ldots, N-1, \\
& x_{k+1} = f(x_k, u_k) && k = 0, \ldots, N-1, \\
& g(x_k, u_k) < 0 && k = 0, \ldots, N-1, \\
& h(x_k, u_k) > d && k = 0, \ldots, N-1, \\
& u_{min} \leq u_k \leq u_{max} && k = 0, \ldots, N-1
\end{aligned}
$$

where J is cost function shown in subsection 2.5. The first two constraints ensure that the predicted state is consistent with the system equations 1. The third constraint refers to the cardioid area, which is shown in more detail in subsubsection 2.3.2. The fourth constraint is relative to the collision avoidance explained in subsubsection 2.3.3. The last one is relative to the range of values the input u can have and has been explained in subsection 2.4.

3 Simulations Results

The simulations were set on MATLAB and divided into two main parts; the first consisted of having the Target fixed in one position and the second one moving in a circular trajectory. The Target would be fixed or rotating along its axis in both parts. This brings the simulations to investigate four different scenarios. Before the simulations could be performed, the controller had to be set up. The process will be shown in subsection 3.1. The simulations with the fixed Target will be discussed in subsection 3.2, and those with the moving Target will be discussed in subsection 3.3.

3.1 Controller setup

As mentioned before, the controller used for this work is the NMPC, and there are different values and constraints that must be set to have the simulations running. These are the following:

- Positioning of the Chaser, Target and obstacles
- Δt simulation time
- Weight matrices Q, R_u and R_d
- Constraints
  - Upper and lower boundaries
  - Polynomial trajectory
  - Cardioid
- N prediction horizon
- dt controller sample time

Chaser, Target, and obstacles positioning

Throughout all the simulations, the initial position of both the Target and Chaser has been kept the same. Instead, the two obstacles' positions have been kept the same for all the simulations but moved randomly in some cases to show the validity of the controller's Collision Avoidance System (CAS). These cases can be seen in Figure 8, Figure 17 and Figure 21. The coordinates can be found at Table 2

	x (m)	y (m)	θ (rad)
Chaser	-1	-1.5	$\frac{\pi}{2}$
Target	0	0	$\frac{\pi}{6}$
Obstacle 1	-0.5	-1	-
Obstacle 2	-0.7	-0.5	-

Table 2: Chaser, Target and obstacles positioning

The coordinates for the final point that the Target has to reach have been calculated based on the orientation of the Target plus a safety margin d_{safe} of $0.02\,m$ with the following equations:

$$\begin{cases} x_f &= x_{Target} + d_{safe} \cdot \cos\left(\theta_{Target}\right) \\ y_f &= y_{Target} + d_{safe} \cdot \sin\left(\theta_{Target}\right) \end{cases} \tag{11}$$

Δt simulation time

The choice of the overall duration of each simulation has been set at 120 s. This was to give the maximum time that the actual Slider could run due to its limited amount of fuel that could be stored.

Weight matrices Q, R_u and R_d

The values of these matrices have been kept constant for all the MATLAB simulations and then modified for the practical test. The values for the MATLAB simulations can be found at Table 3 while the modifications done for the practical test can be found at Table 20.

Q	R_u	R_d
diag(0.5 0.5 1 0.5 0.5 1)	diag(0.03 0.03 0.15)	diag(0.02 0.02 0.2)

Table 3: Weight matrices Q, R_u and R_d values for MATLAB simulations

Constraints

As mentioned in subsection 2.3, the constraints used inside the controller are three:

1. Lower and Upper bounds
2. Polynomial trajectory
3. Cardioid area
4. Obstacle avoidance

The first constraint is related to the physical characteristics of the model used in the simulations, i.e., the Slider. The values have been provided following the schematics on [23] and are the following:

	f_x (N)	f_y (N)	τ (Nm)
lb	-1.4	-1.4	0.518
ub	1.4	1.4	0.518

Table 4: Upper and Lower bounds

The following two constraints are related to the objective of having a safe approach and docking in any scenario.

The polynomial trajectory has been used just for the fixed Target scenario, and it has been defined by the Equation 3. The method consisted of defining a reference trajectory using these equations. These were solved by defining some waypoints and solving the problem $Ax = b$. The number of waypoints could vary depending on the initial position of the Chaser and the Target orientation.

The cardioid area has been defined to solve the issues encountered with the polynomial trajectory and, hence, provide an alternative technique to ensure a safe approach and docking in every scenario. The cardioid area is defined by Equation 4, and its radius has been set to be $0.25\,m$. By implementing this barrier function, the controller automatically calculates the area's inclination according to the Target's rotation and its docking port orientation, making it suitable for each scenario used in this work. Then, at each iteration, it calculates the distance between the Chaser and the cardioid area and ensures that the next step won't bring the Chaser inside the area.

The last constraint has been implemented into the controller to avoid the obstacles. The principle behind it is to let the controller calculate the distance between the Chaser and the obstacle at every step (Equation 6) and avoid the Chaser getting closer than a certain margin to the obstacles. That margin has been named d_{safe} and its value is $0.02\,m$.

N and dt parameters

These parameters must be carefully tuned to work correctly in each scenario the Sliders will encounter. Several values have been considered and are shown in Table 5. The challenge here is to get the best values by balancing computational efficiency, controlling performance, and ensuring stability. In real-time applications, a longer prediction horizon N can increase computational complexity and enhance the controller's robustness. However, it can be a big concern for real-time implementation and responsiveness. On the other hand, a shorter sampling time dt allows the controller to be more reactive but can lead to an overly aggressive one. Furthermore, when adding constraints different parameter values can let the system run into minimums that let the system stuck in one point and invalidate the simulation.

N (s)	dt (s)
5	0.1
10	0.2
15	0.5
20	1.0

Table 5: N and dt parameters values that have been considered to set the controller

After several simulations two different sets of values have been chosen. The first one is for the first Fixed Target scenario while the second one is for the second Fixed Target scenario and the Moving Target ones. The final values that have been chosen are the following

	N (s)	dt (s)
Fixed Target (Case 1)	10	0.5
Moving Target/Fixed Target (Case 2)	5	0.5

Table 6: N and dt parameters values used for the simulations

3.2 Fixed Target

Two tests have been performed for the Fixed Target case. Both are point-to-point navigation cases where in the first one, the Target is fixed both in position and orientation, while in the second one, the Target is fixed in position but is rotating along its axis. For the first case, two approaches have been followed: one in which the reference trajectory is generated with the polynomial trajectory. In contrast, the second one has the final point as a reference value. Table 8 display the initial and expected final position of the Target. Note that the target's final orientation is rotated 180° so that the two docking ports face each other.

	Case 1	Case 2
Position	Fixed	Fixed
Rotation	None	Rotating
Reference trajectory	Polynomial trajectory and final point	Final point

Table 7: Characteristics of the two different cases for the Fixed Target simulations

	x (m)	y (m)	θ (rad)
Chaser	-1	-1.5	$\frac{\pi}{2}$
Target	0	0	$\frac{\pi}{6}$

(a) Chaser and Target Initial positions

	x (m)	y (m)	θ (rad)
Chaser	0.3204	0.1850	$\frac{7}{6}\pi$
Target	0	0	$\frac{\pi}{6}$

(b) Chaser and Target Final positions

Table 8: Chaser and Target Initial and Final positions for Case 1

Case 1

As mentioned, the first test used a trajectory generated by a polynomial function. To do so, 4 waypoints have been manually defined and can be seen at Table 9. Based on those, the trajectory has been generated. In Figure 7, the cardioid has been added to show that in both strategies, this one and the one with the cardioid, a safe approach is guaranteed. The simulations shown in Figure 8 show that the CAS inside the controller works when the Slider encounters an obstacle along its path and quickly sticks back to the reference trajectory.

	x (m)	y (m)	θ (rad)
Waypoint 1	-1	-1.5	$\frac{1}{2}\pi$
Waypoint 2	0.375	-0.6495	$\frac{7}{6}\pi$
Waypoint 3	0.7048	0.2565	$\frac{7}{6}\pi$
Waypoint 4	0.3204	0.1850	$\frac{7}{6}\pi$

Table 9: Selected waypoints for Case 1

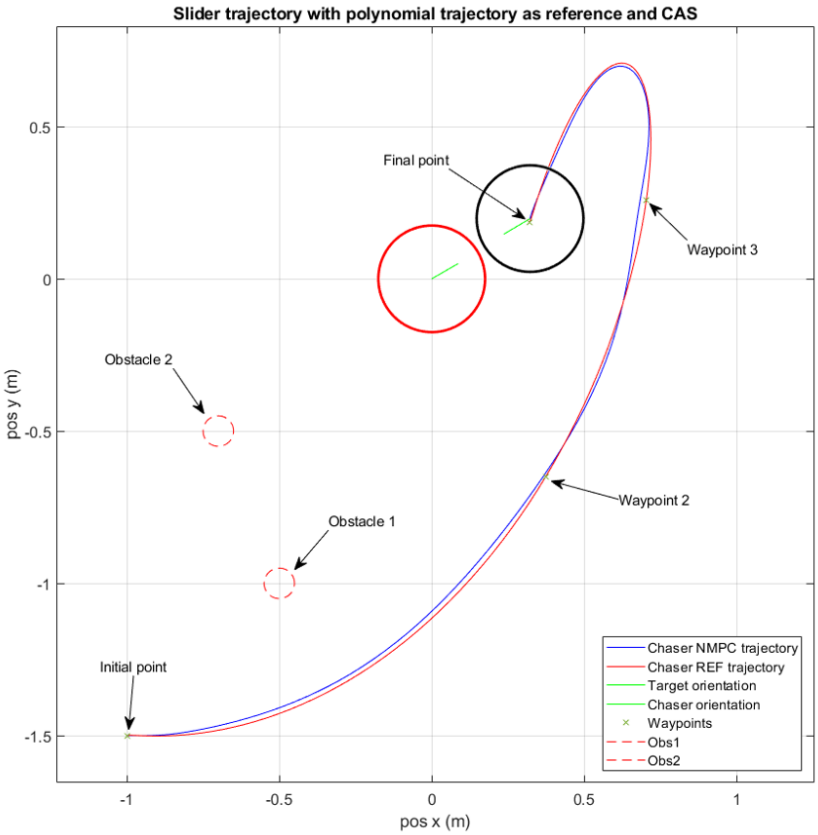


Figure 7: Slider trajectory when using the polynomial trajectory as reference

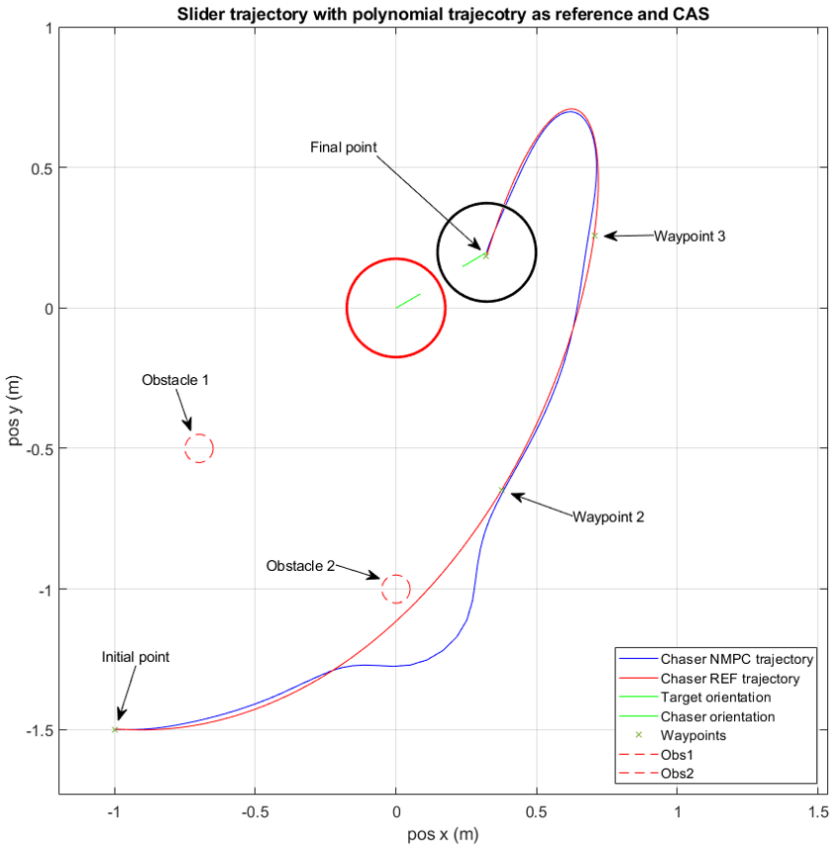
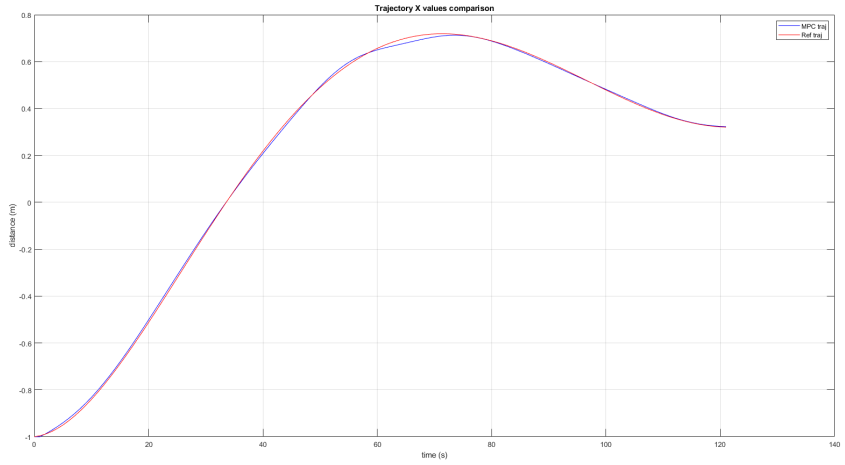
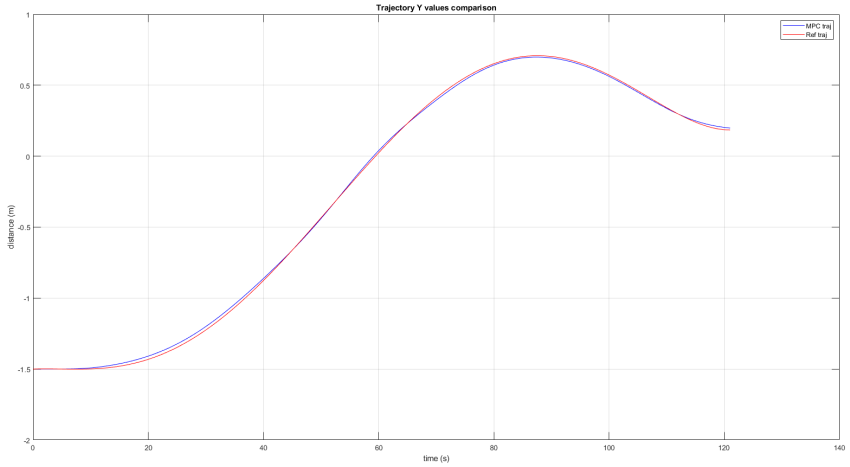


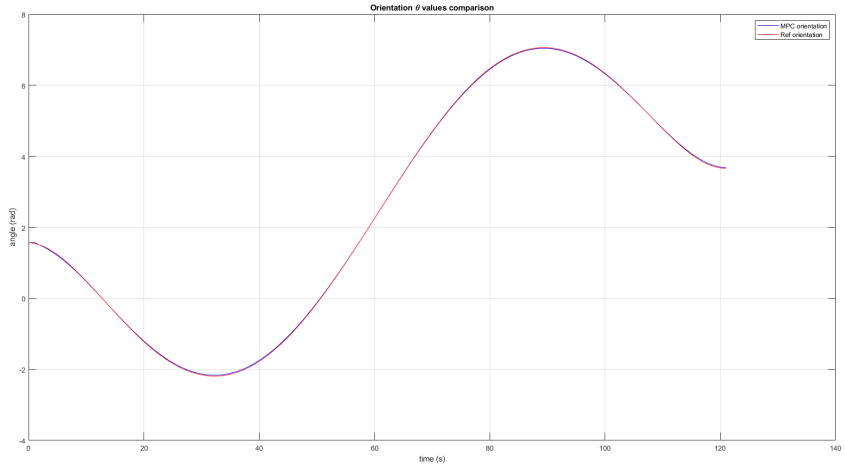
Figure 8: Slider trajectory when using the polynomial trajectory as a reference with an obstacle on the trajectory



(a) X component of the Slider trajectory



(b) Y component of the Slider trajectory



(c) Slider's orientation θ

Figure 9: Trajectory and orientation analysis of the case shown in Figure 7

In the second test, the reference was just the final point, and the collision avoidance integrated the cardioid and the obstacles. This results in having the Slider that has to reach the reference point as quickly as possible, and as we can see in Figure 10, that ends with a straight line between the initial and final points. The CAS and the controller manage to bring the Slider to the desired position, respecting all the constraints given.

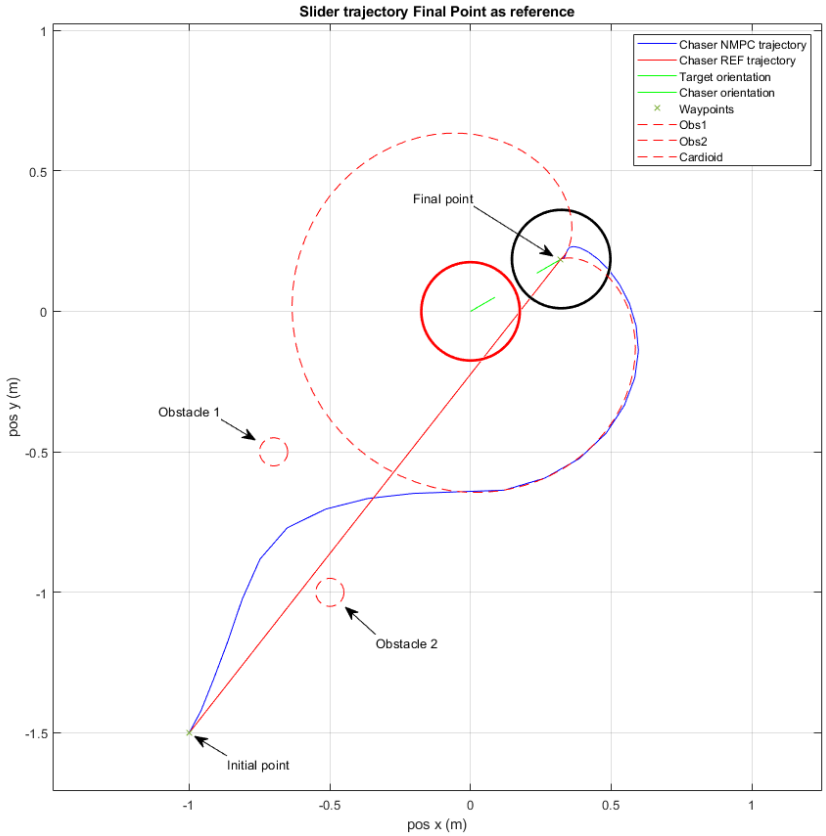
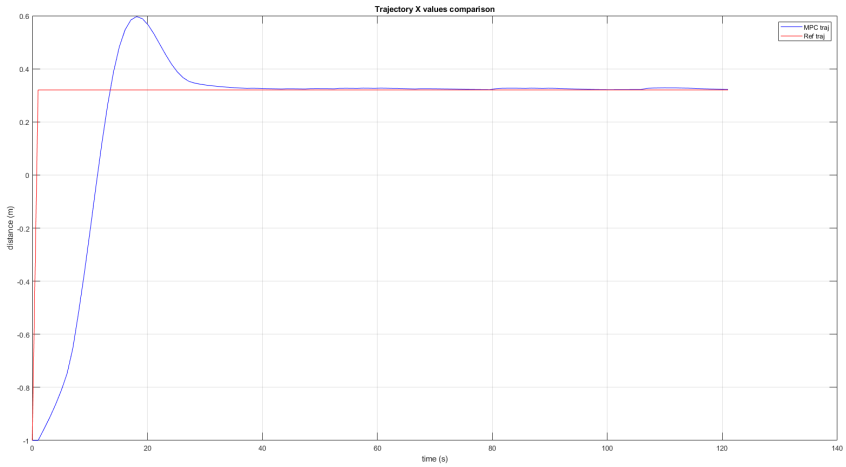
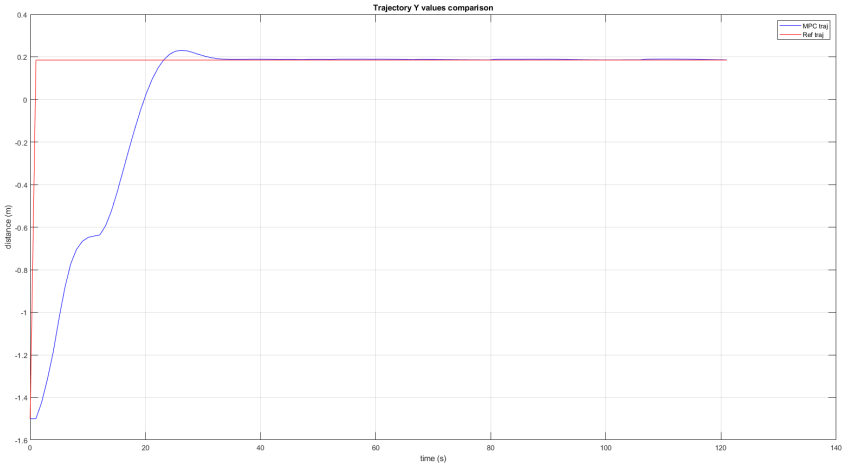


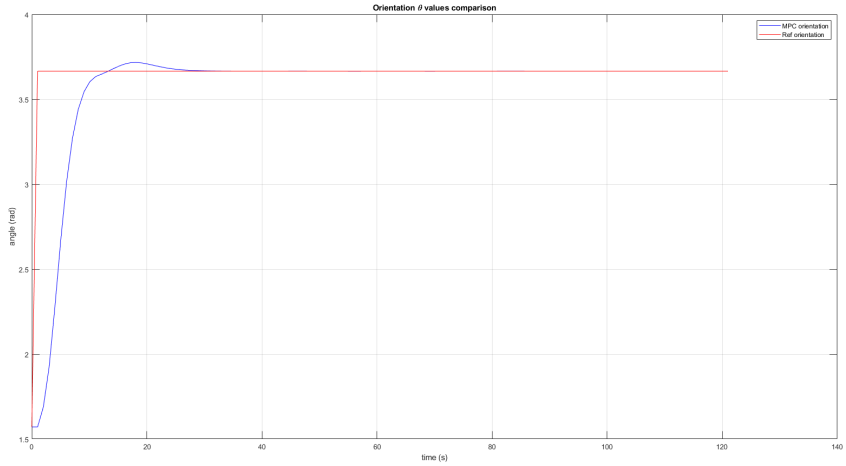
Figure 10: Slider trajectory when using the final point as reference



(a) X component of the Slider trajectory



(b) Y component of the Slider trajectory



(c) Slider's orientation θ

Figure 11: Trajectory and orientation analysis of the case shown in Figure 10

Case 2

In this case, the Target is rotating along its axis at a constant speed of $0.01\,rad/s$. The CAS has been implemented, and since the target is rotating, the reference final point is updated accordingly. Table 10 display the Target's initial and expected final position. Note that the Target's final orientation is rotated 180° so that the two docking ports face each other.

	x (m)	y (m)	θ (rad)
Chaser	-1	-1.5	$\frac{\pi}{2}$
Target	0	0	$\frac{\pi}{6}$

(a) Chaser and Target Initial positions

	x (m)	y (m)	θ (rad)
Chaser	-0.0563	0.3656	4.8652
Target	0	0	1.7236

(b) Chaser and Target Final positions

Table 10: Chaser and Target Initial and Final positions for Case 2

Here are the results. In Figure 12, the first settings for N and dt as shown in Table 6 have been used, while the second ones have been used in Figure 13. It is noticeable that a shorter prediction horizon removes all the oscillations when the Slider is close to the final point. In fact, as we can see in Figure 13, the Slider's trajectory is way smoother than the one in Figure 12.

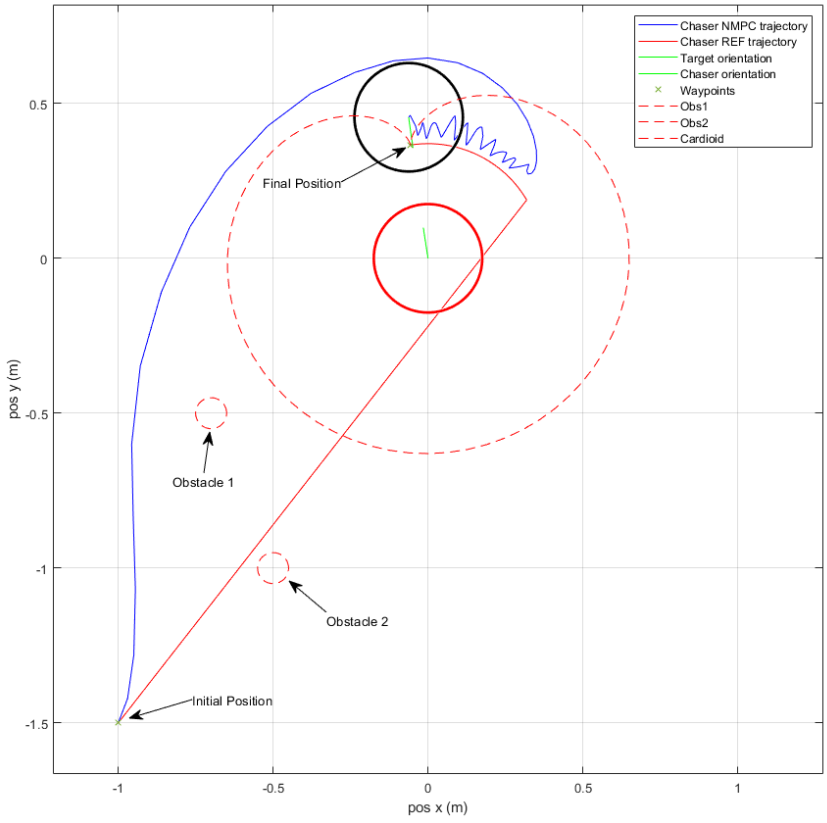


Figure 12: Slider trajectory when the target is rotating on its axis in a fixed position

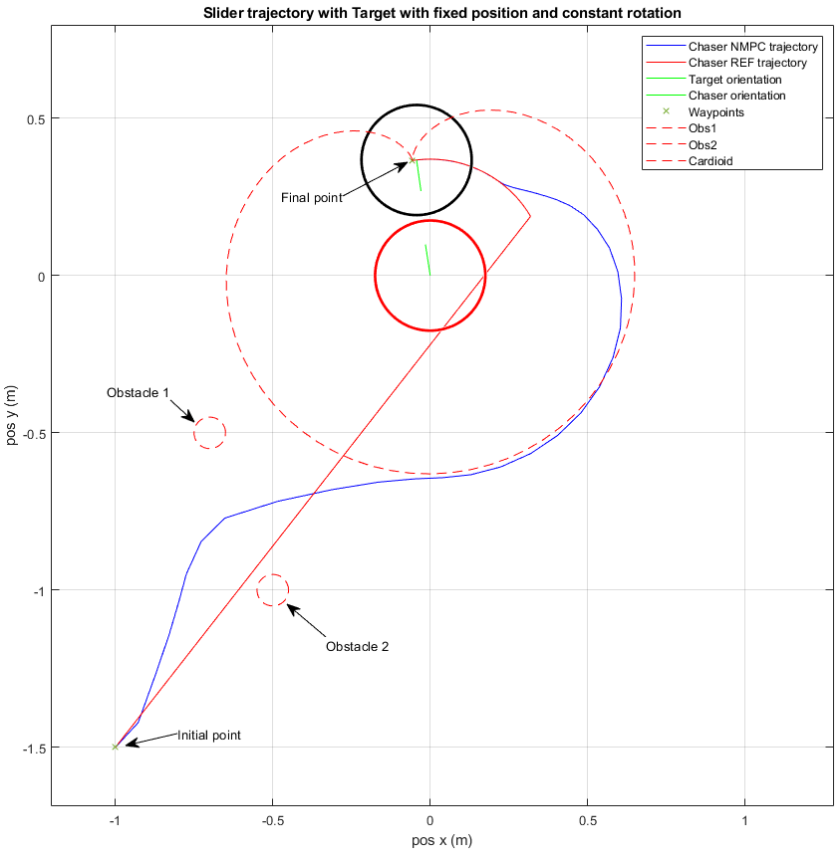
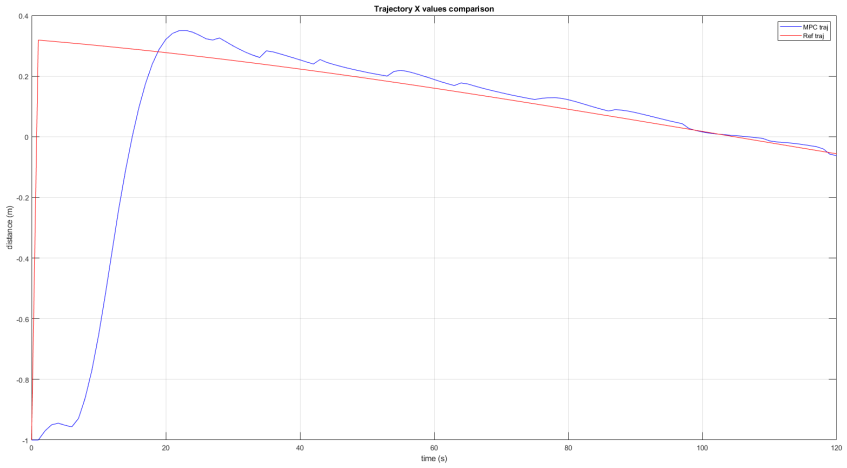
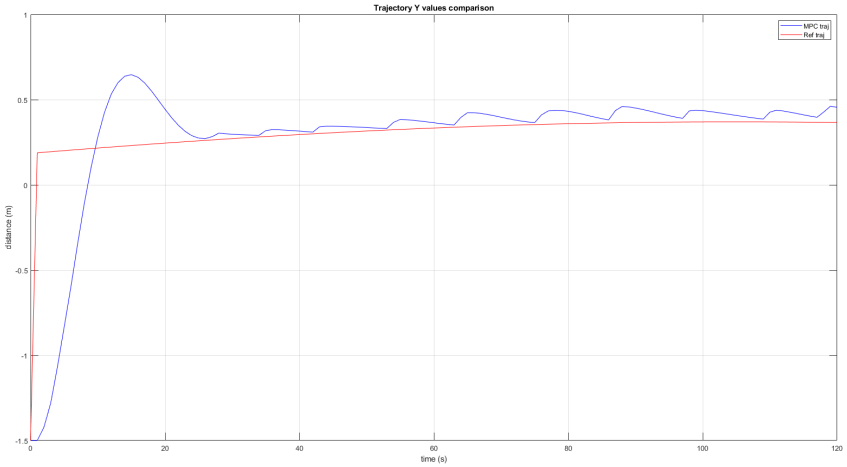


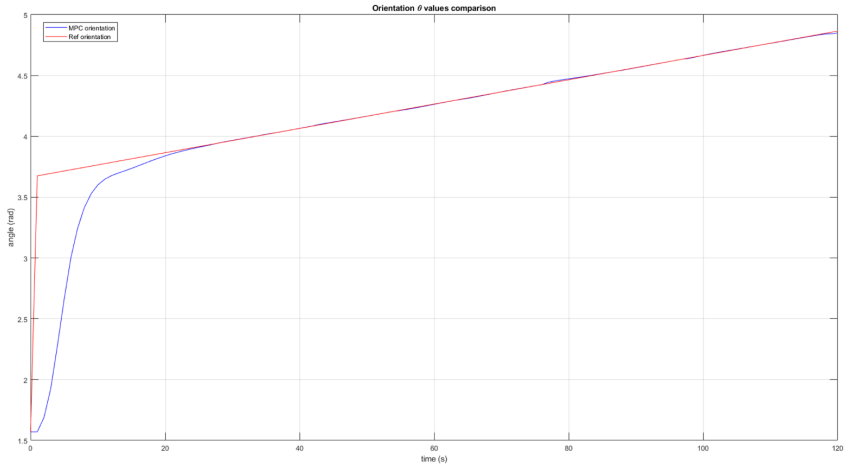
Figure 13: Slider trajectory when using $N = 5\,s$



(a) X component of the Slider trajectory



(b) Y component of the Slider trajectory



(c) Slider's orientation θ

Figure 14: Trajectory and orientation analysis of the case shown in Figure 12

3.3 Moving Target

The second part of the simulation focused on having the Target moving in a controlled behavior across the plane. As subsection 3.2, this part has been divided into two cases: one having the Target fixed in its orientation while moving and the other rotating along its axis. In both cases, the Target follows a circular trajectory centered in (0,0) with a radius of $0.6\,m$ at a constant speed of $0.02\,rad/s$.

	Case 3	Case 4
Position	Moving	Moving
Rotation	None	Rotating
Reference trajectory	Final point	Final point

Table 11: Characteristics of the two different cases for the Moving Target simulations

	Case 3 (rad/s)	Case 4 (rad/s)
Circular trajectory	0.02	0.02
Rotation	None	0.02

Table 12: Speed values of the Target in the Moving Target simulations

Case 3

As mentioned above, this test was performed by having the Target move in a circular trajectory while keeping its orientation fixed all the time. Table 13 shows the Chaser's initial and expected final positions.

	x (m)	y (m)	θ (rad)
Chaser	-1	-1.5	$\frac{\pi}{2}$
Target	0	0	$\frac{\pi}{6}$

(a) Chaser and Target Initial positions

	x (m)	y (m)	θ (rad)
Chaser	-0.7153	0.6652	$\frac{7}{6}\pi$
Target	-0.4424	0.4053	$\frac{\pi}{6}$

(b) Chaser and Target Final positions

Table 13: Chaser and Target Initial and Final positions for Case 3

Figure 15 shows that using a value of $10\,s$ for the N makes the Target's position oscillate close to the final point. This is more detailed in Figure 18 where the trajectory components have been separated. X and Y values oscillate close to the reference values without reaching them, while the orientation θ stays constantly fixed to the reference value. As it can be seen in Figure 16, by reducing the value of N from 10 to $5\,s$ a more stable and closer to the reference values trajectory can be obtained. Figure 17 shows that the CAS implemented into the controller allows the Slider to avoid the obstacles in its way even in this scenario.

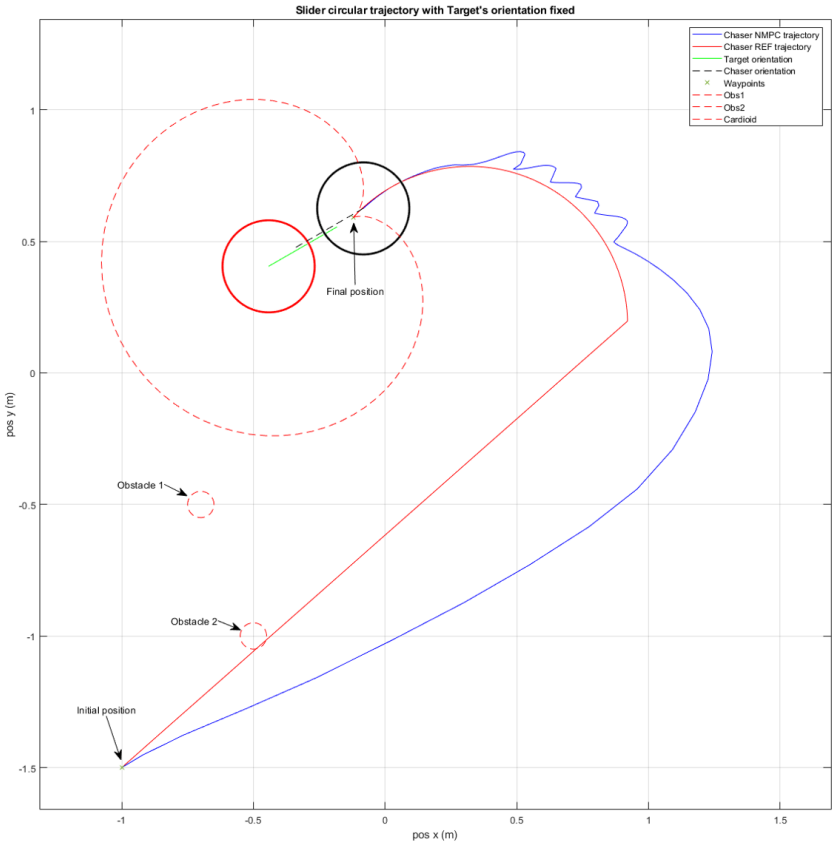


Figure 15: Slider trajectory when the Target is moving in a circular trajectory with fixed rotation

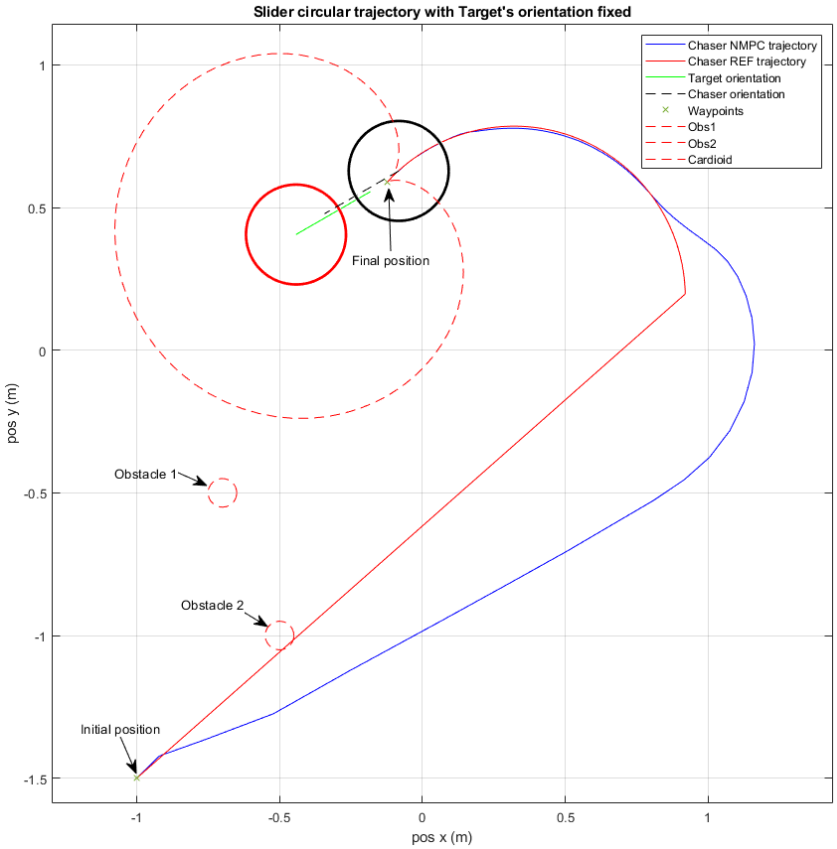


Figure 16: Slider trajectory when using $N = 5\,s$

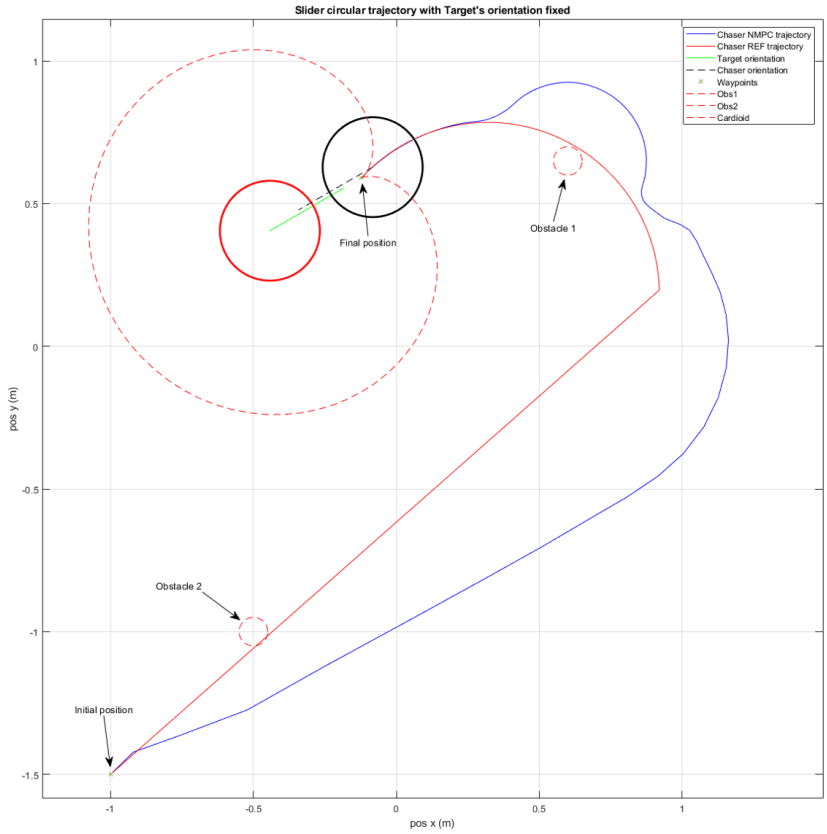
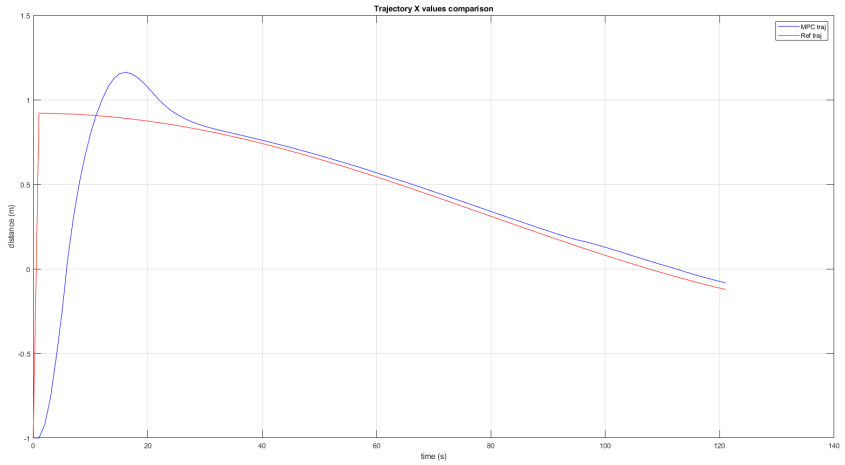
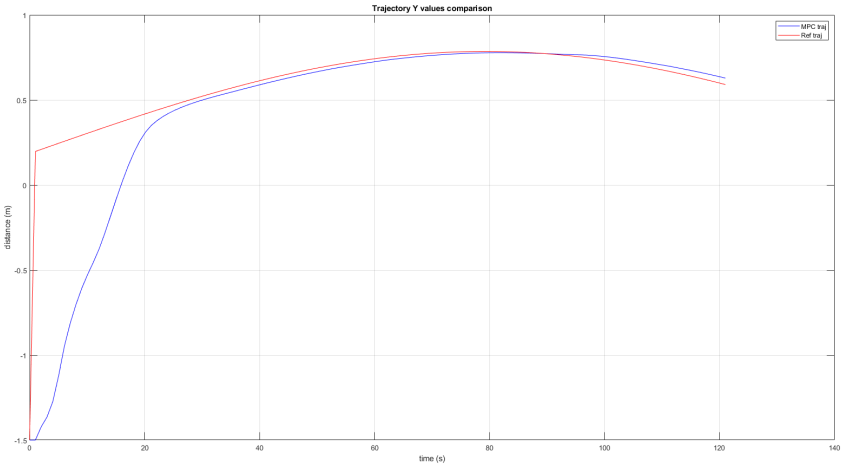


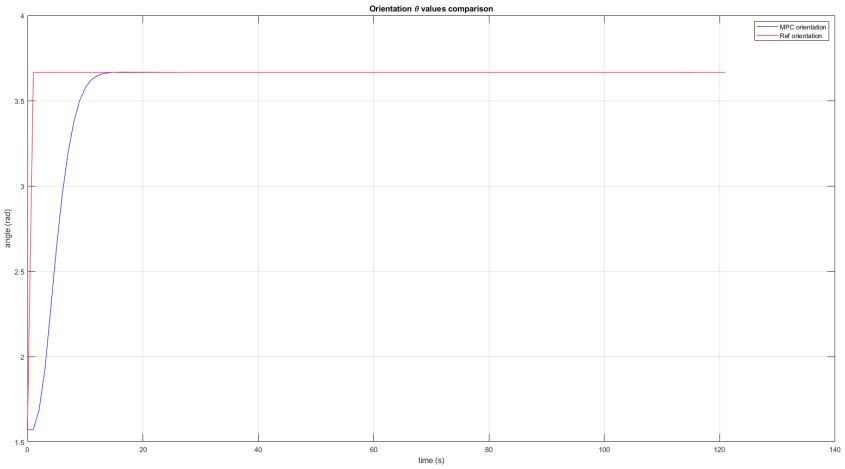
Figure 17: Slider trajectory when using $N = 5\,s$ and obstacles on it way



(a) X component of the Slider trajectory



(b) Y component of the Slider trajectory



(c) Slider's orientation θ

Figure 18: Trajectory and orientation analysis of the case shown in Figure 16

Case 4

In this simulation, the Target moves in a circular trajectory as in Case 3.3 and rotates on its axis. The Target's revolution around (0,0) and rotation speeds are matched so the docking port always faces the outside of the circular trajectory. This has been done to ease the controller's effort when controlling the Slider in this scenario by reducing the computational effort.

Figure 19 shows that using a value of 10 for the N makes the Target's position oscillate close to the final point. Looking at Figure 22, it can be seen more in detail by analysing the single components of the trajectory. X and Y values oscillate close to the reference values without reaching them while the orientation θ stays constantly fixed to the reference value.
In Figure 20, it can be seen that reducing the value of N to 5 allows the controller to stabilize the Slider to the reference final point.
Figure 21 demonstrates that the controller can also make the Slider avoid the obstacles in the Slider's way.

	x (m)	y (m)	θ (rad)
Chaser	-1	-1.5	$\frac{\pi}{2}$
Target	0	0	$\frac{\pi}{6}$

(a) Chaser and Target Initial positions

	x (m)	y (m)	θ (rad)
Chaser	-0.7153	0.6652	5.5416
Target	-0.4424	0.4053	2.9236

(b) Chaser and Target Final positions

Table 14: Chaser and Target Initial and Final positions for Case 4

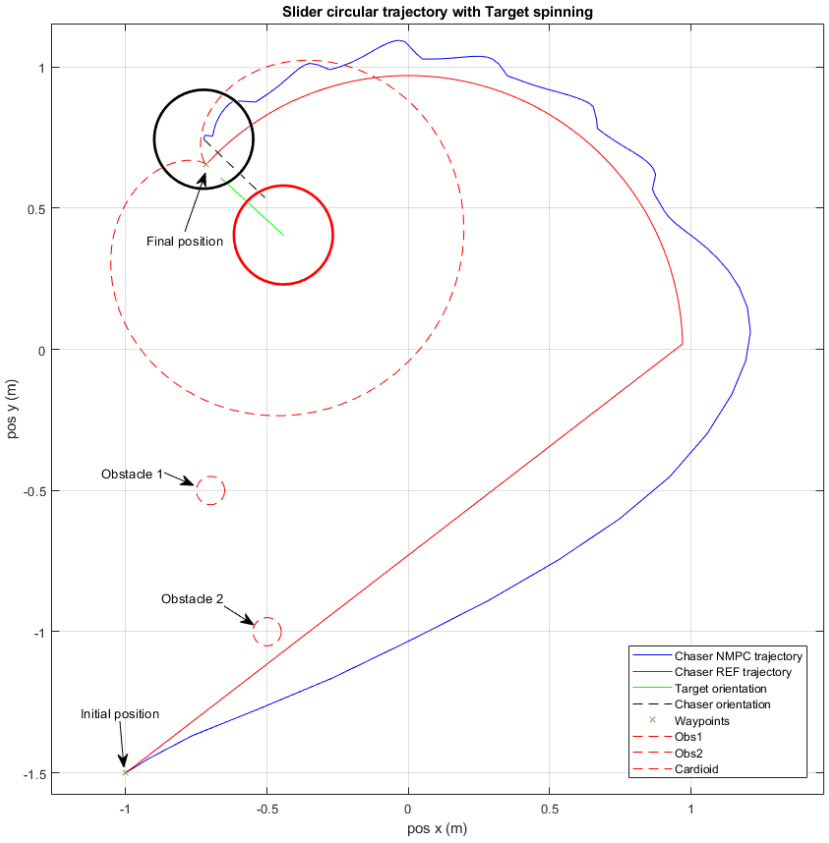


Figure 19: Slider trajectory when the Target is moving in a circular trajectory and rotating

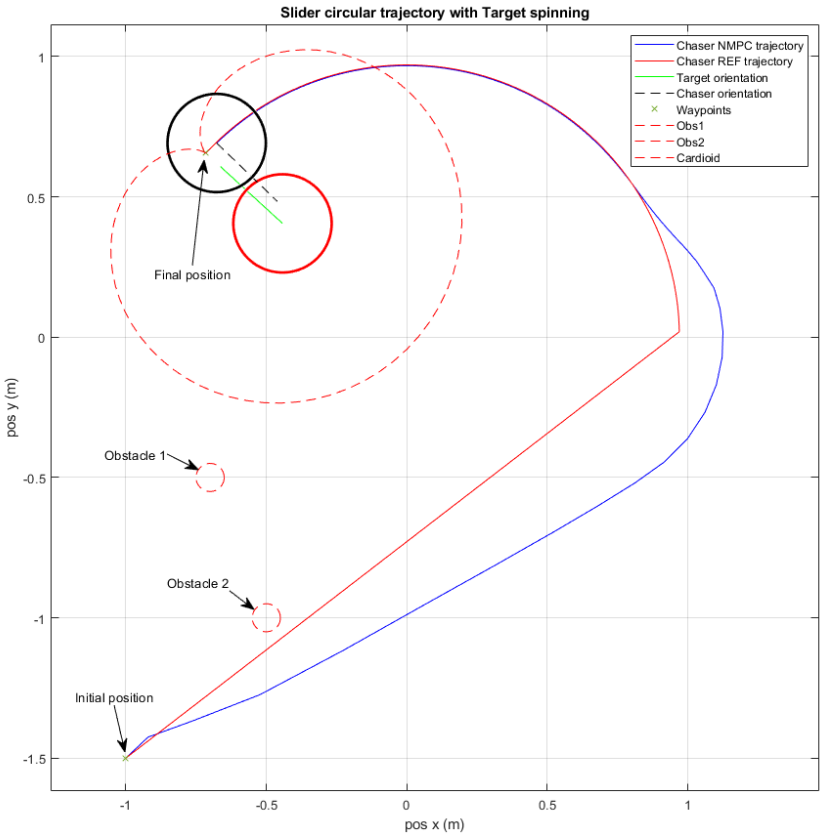


Figure 20: Slider trajectory when using $N = 5\,s$

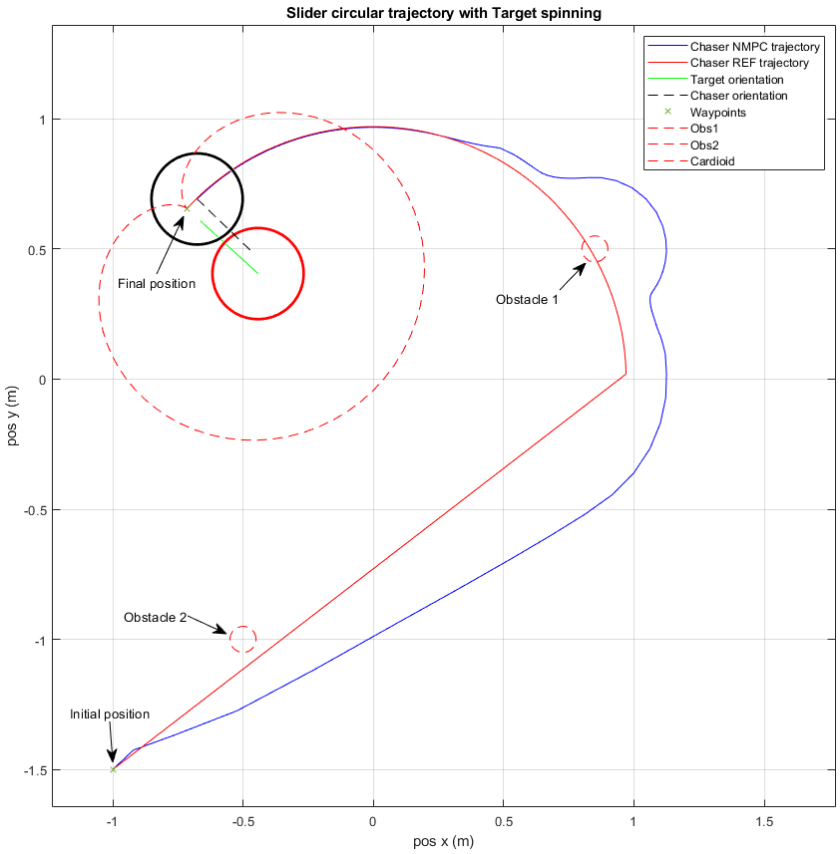
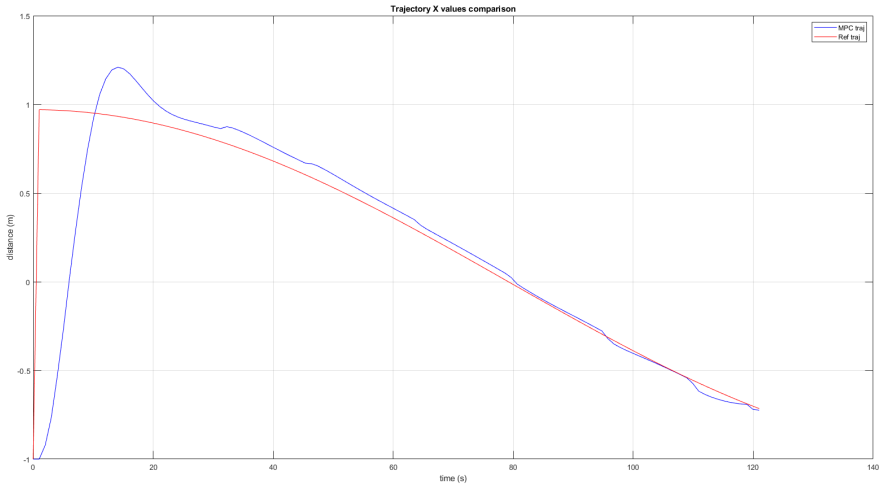
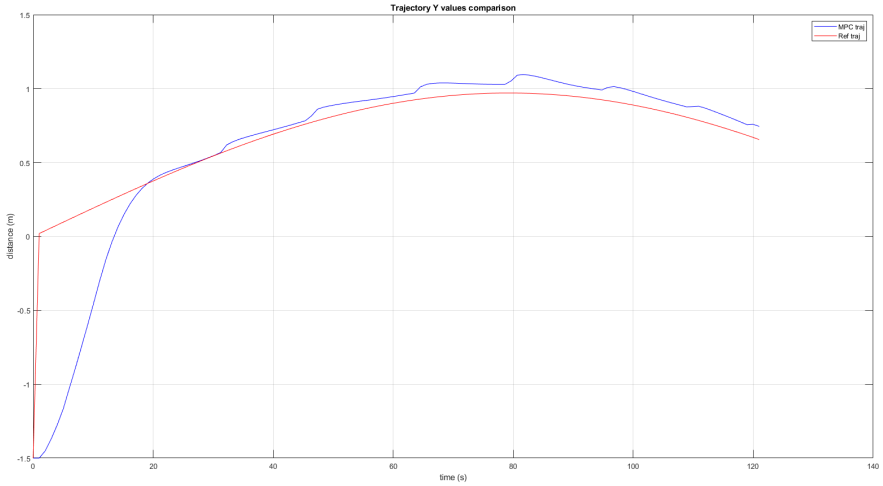


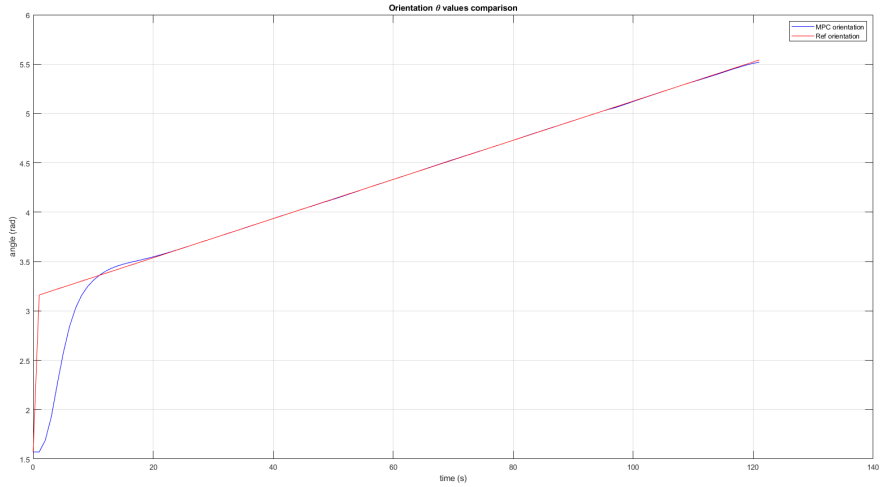
Figure 21: Slider trajectory when using $N = 5\,s$ and obstacles on its way



(a) X component of the Slider trajectory



(b) Y component of the Slider trajectory



(c) Slider's orientation θ

Figure 22: Trajectory and orientation analysis of the case shown in Figure 19

3.4 Results' analysis

Looking at the graphs above, the controller and the safety constraints handle the task very well. The Slider never crushes into any obstacle and follows the trajectory with little to no deviation. In the fixed Target cases, the controller allows the Chaser to follow the reference trajectory, avoid obstacles, and avoid entering the cardioid area whether the Target is rotating or not. it is worth mentioning that a shorter prediction horizon (N) makes the controller more precise in following the reference trajectories. The simulations for the moving Target cases showed similar behavior and, therefore, good results. The oscillations in the trajectory for the cases in which the Target is rotating and/or moving have been removed by simply lowering the N values.

All of this can be seen in the trajectory analysis graphs shown in Figure 9, 11,18,14,22.
Another way to demonstrate the controller's accuracy is to show the difference between the Chaser's expected final position and orientation and the actual ones achieved in the simulations. This is shown in the following tables, where the error has been calculated as

$$Error = \| x - x^* \| \tag{12}$$

where x represented the value obtained from the simulation and x^* the expected value.

Case 1

As we can see in Table 15, the controller is accurate enough to keep the error below the cm scale. Noticeably, the choice of using just the final point as a reference reduces the error. In fact, as it can be seen in Table 15b, the Target reaches the final position with just a deviation of just one millimeter and with the correct orientation.

	x (m)	y (m)	θ (rad)
Expected	0.3204	0.1850	3.6652
Result	0.3225	0.1978	3.6763
Error	0.0021	0.0128	0.0111

(a) Case 1 error in final positioning

	x (m)	y (m)	θ (rad)
Expected	0.3204	0.1850	3.6652
Result	0.3221	0.1860	3.6652
Error	0.0017	0.001	0

(b) Case 1 error in final positioning with final point

Table 15: Case 1 error in final positioning

Case 2

In this case, the results in Table 16 show similar errors between two cases, but it is noticeable from Figure 12 and Figure 13 that the overall precision in following the path and reaching the final position is higher in the case with $N = 5$. This is why the error on the y coordinates is high for Table 16a than for Table 16b.

	x (m)	y (m)	θ (rad)
Expected	-0.0563	0.3657	4.8652
Result	-0.0625	0.4552	4.8493
Error	0.0062	0.0895	0.0159

(a) Case 2 error in final positioning with $N = 10$

	x (m)	y (m)	θ (rad)
Expected	-0.0563	0.3656	4.8652
Result	-0.0433	0.3672	4.8525
Error	0.013	0.0016	0.0127

(b) Case 2 error in final positioning with $N = 5$

Table 16: Case 2 error in final positioning

Case 3

Case 3 shows almost identical behavior of the Chaser and similar positioning errors. The controller manages to achieve the exact orientation needed for this case, but there is a small error in achieving the exact final position.

	x (m)	y (m)	θ (rad)
Expected	-0.1220	0.5903	3.6652
Result	-0.0788	0.6285	3.6652
Error	0.0432	0.0382	0

(a) Case 3 error in final positioning with $N = 10$

	x (m)	y (m)	θ (rad)
Expected	-0.1220	0.5903	3.6652
Result	-0.0828	0.6281	3.6652
Error	0.0392	0.0378	0

(b) Case 3 error in final positioning with $N = 5$

Table 17: Case 3 error in final positioning

Case 4

In the last case, the controller does not achieve the same precision in positioning the Chaser in the right final position. This is due to the higher complexity of the scenario, but nonetheless, the error is really small, lower than $4\,cm$. As seen in the previous cases, the lower value of N increases the smoothness of the Chaser trajectory.

	x (m)	y (m)	θ (rad)
Expected	-0.7153	0.6552	5.5416
Result	-0.7461	0.7416	5.5247
Error	0.0308	0.0864	0.0142

(a) Case 4 error in final positioning with $N = 10$

	x (m)	y (m)	θ (rad)
Expected	-0.7153	0.6552	5.5416
Result	-0.6764	0.6920	5.5161
Error	0.0389	0.0368	0.0255

(b) Case 4 error in final positioning with $N = 5$

Table 18: Case 4 error in final positioning

4 Experimental Results

To demonstrate the effectiveness of the controller with its constraints, four tests were conducted in the Kiruna test bed. In this section, two sets of tests have been performed using one Slider. Firstly, three tests with point-to-point navigation were used to test and verify that the Slider and controller could perform well inside the Kiruna test bed environment. Finally, a single test was performed to test that the controller could handle the constraint of the cardioid. The control model was implemented into MATLAB SIMULINK, and the communications between PC and Sliders were performed through Wi-Fi and ROS systems. The detailed communication process can be found at [25].
It is worth mentioning that the results are very limited and lack precision due to the limited time in which the laboratory was available and the hardware available during the time of tests. In fact, the real-time feedback necessary for the control algorithms could be provided only by the Intel real sense tracking camera T265. Its sensor technology allows fairly accurate localization and velocity (transnational and rotational) of the Slider but not enough to provide smooth linear trajectories expected for this kind of test. Nonetheless, it was possible to perform some tests.

4.1 SIMULINK Model

As mentioned before, the MATLAB model had to be implemented into SIMULINK. First, a software simulation has been run with the PWM implemented. A schematic representation of it can be seen in Figure 23. Secondly, the ROS was implemented in it to make it work on Slider hardware.

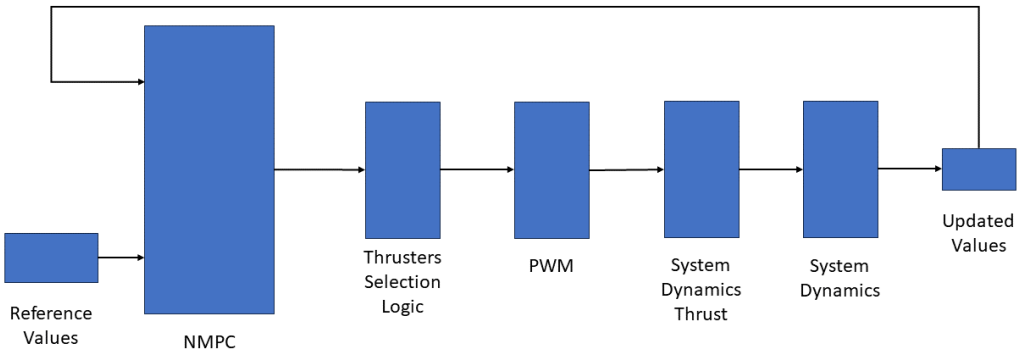


Figure 23: Simulink model scheme

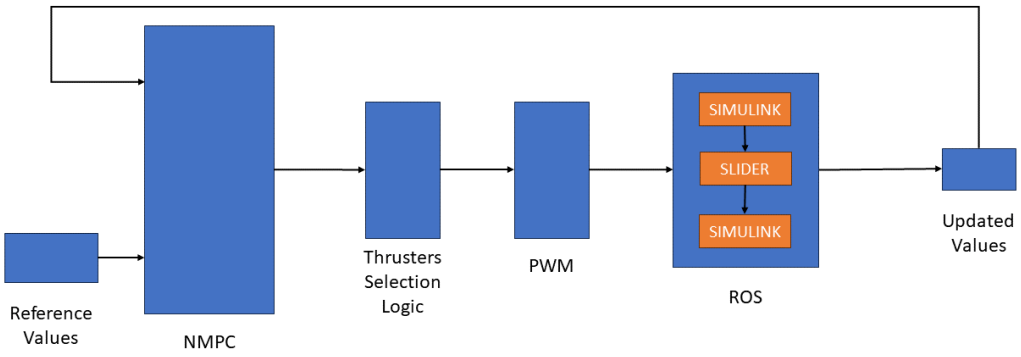


Figure 24: Simulink model scheme with ROS

4.2 Case 1: Point-to-Point Maneuver

As said above, the first part of the testing consisted of three small tests in which a point-to-point maneuver was performed. This maneuver simulates a docking scenario to test the controller and see the differences between the software and hardware simulation. The final position had to be intended as the final position the Chaser must reach to dock to the Target. Therefore, the focus was on position and orientation. In this set of tests, the CAS and the cardioid constraint were not implemented in the MPC.
In the simulation scenario, the Slider started resting and hence, its initial state was $X_0 = [0_{6\times1}]$ and then, it was sent to another position which was $X_f = \left[0.3204,\ 0.1850,\ \frac{pi}{6},\ 0_{1\times3}\right]^T$ for the first test and $X_f = [0.3,\ 0.3,\ 0,\ 0_{1\times3}]^T$ for the last two. Noticeably, the Slider's initial positioning was never exactly (0,0). This is because of the initialization process the Slider's hardware had to perform at the beginning of each simulation and the random placement of the Slider on the table. The values of matrices Q and R used in the three tests are shown in Table 20 while the final positions and orientations that the Slider had to reach in each test are displayed in Table 19. In the following sections, the trajectory of the Slider will be analyzed.

	x (m)	y (m)	θ (rad)
Test 01	0.3204	0.1850	$\frac{\pi}{2}$
Test 02	0.3204	0.1850	0
Test 03	0.3	0.3	0

Table 19: Chaser final positions and orientations

Parameters Symbol	Matrix Q Q_k	Matrix R R_k	Orientation θ
Test 1	$[5, 5, 10, 50, 50, 10] \times I_{6\times6}$	$[0.3, 0.3, 0.15] \times I_{3\times3}$	$\frac{\pi}{6}$
Test 2	$[5, 5, 0.005, 50, 50, 0.005] \times I_{6\times6}$	$[0.1, 0.1, 0.5] \times I_{3\times3}$	0
Test 3	$[5, 5, 0.005, 50, 50, 0.005] \times I_{6\times6}$	$[0.1, 0.1, 0.5] \times I_{3\times3}$	0

Table 20: Weight matrices Q and R values

4.2.1 Trajectory

In the following images, the overall trajectory of the Slider in each test is shown with a detailed comparison at the end in Figure 28. In Test 01 and 02, the Slider always starts near position $x_0 = (0, 0)$ and tries to reach the final position $x_f = (0.3204, 0.1850)$ while in Test 03 it tries to reach $x_f = (0.3, 0.3)$.
As can be seen in Figure 25 and Figure 28, in Test 01, the trajectory and orientation are not stable. This showed that the controller was not tuned correctly for hardware application. To solve this, the matrices Q and R have been modified, making the controller less responsive and reducing the thruster's activations. Their values can be found in Table 20.
Making significant changes in orientation is always challenging if the controller is not tuned perfectly. Therefore, in Test 02, it has been decided to keep the orientation constant at 0°. Figure 26 shows that the Slider has a way more stable behavior and oscillates close to the final point.
In Test 03, a different final position has been set to test the controller even more. Figure 27 shows that the MPC allows the Slider to reach a stable oscillation to a final point close to the reference one.
As mentioned before, the trajectories are not smooth due to several factors. These can be from the thrusters' ON-OFF mode nature, which always introduces some oscillations and the low accuracy of the sensors available at the time of the simulations. The latter explains the offset between the reference final point and the actual final point that the Slider tends to reach.

Analyzing in detail the trajectory and orientation data from Figure 28 and Figure 29 shows that Test 01 presents a highly unstable behavior. This confirms that the controller set up from the MATLAB simulations is unsuitable for the hardware. Meanwhile, the Slider has a way more stable behavior (Test 02 and 03) after the controller values have been tuned. In fact, it can be seen that the Slider reaches a stable oscillation close to the reference values.

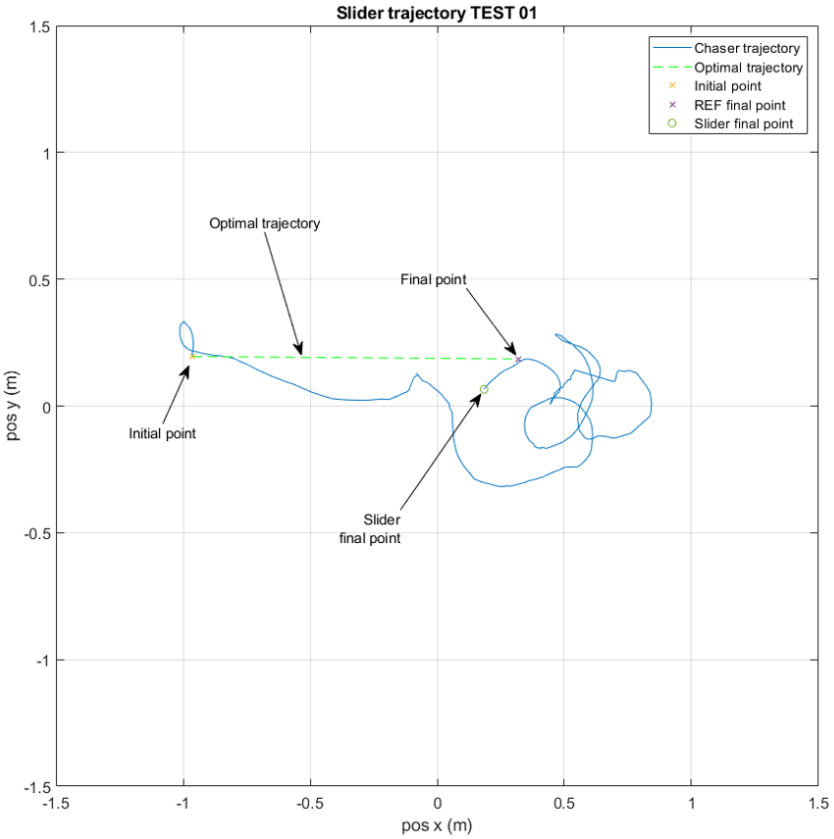


Figure 25: Slider trajectory test 01

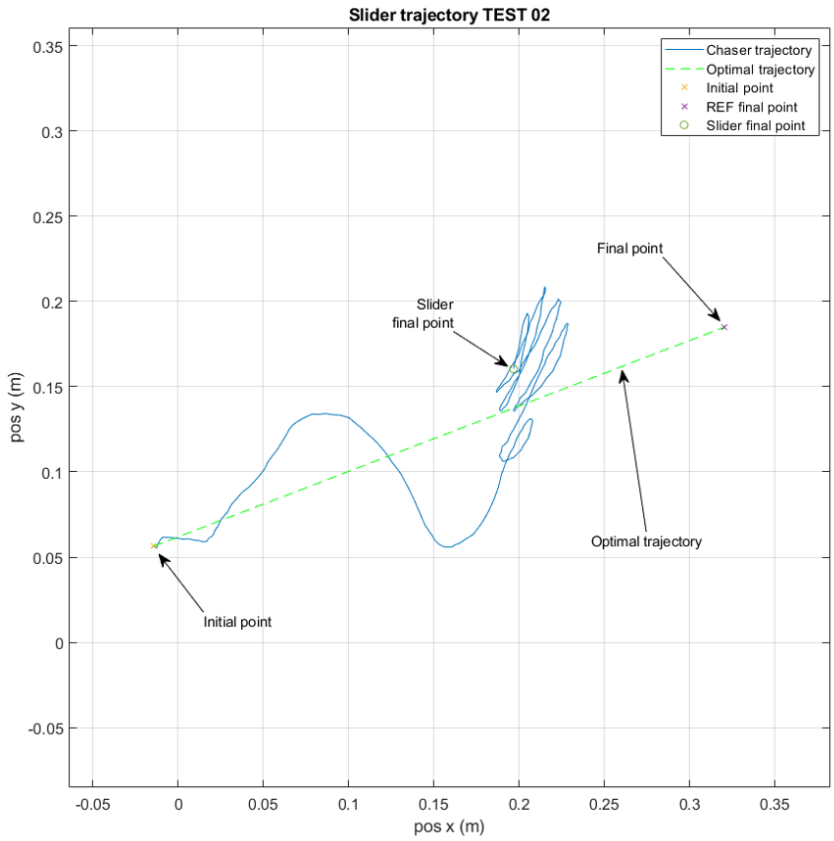


Figure 26: Slider trajectory test 02

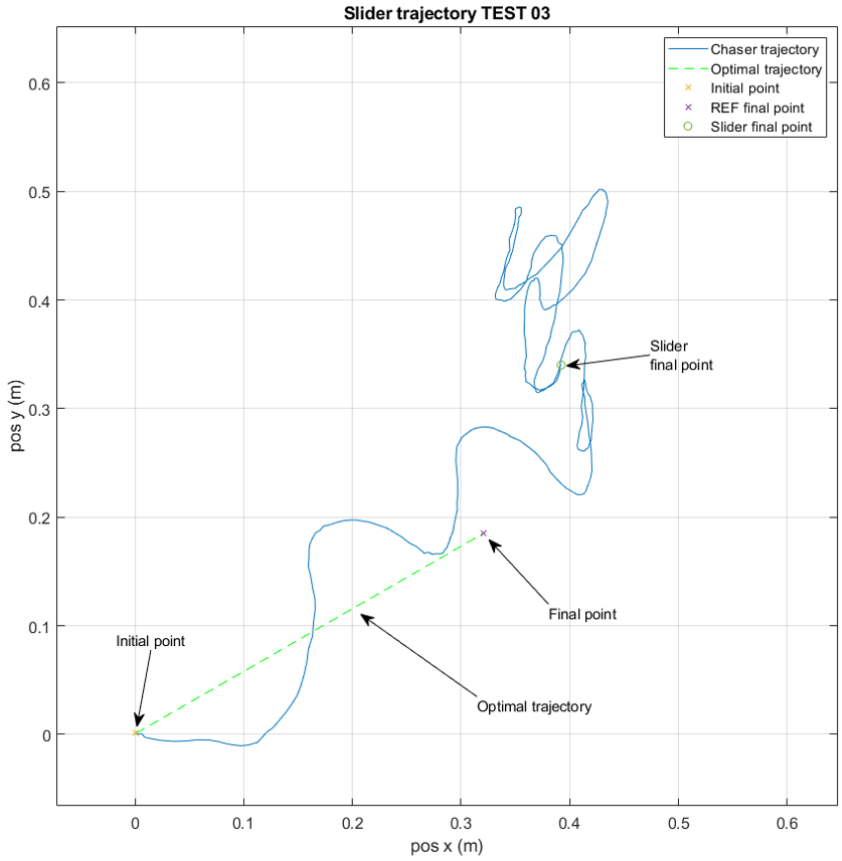


Figure 27: Slider trajectory test 03

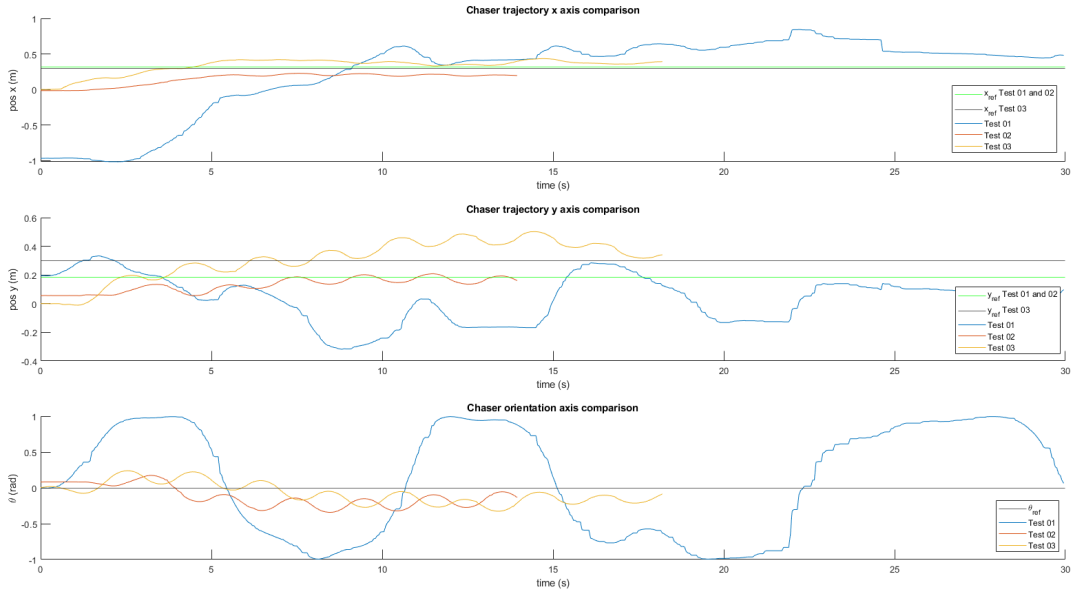


Figure 28: Slider trajectory components comparison Test 01, 02 and 03

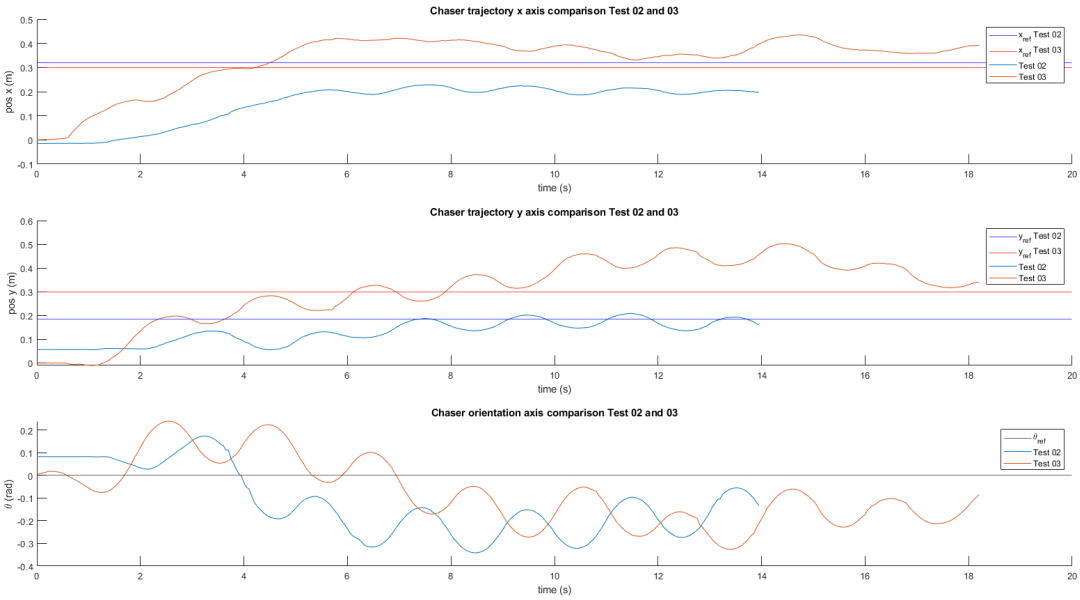


Figure 29: Slider trajectory components comparison Test 02 and 03

4.2.2 Thrust activation

The following graphs show the activation of each of the 8 thrusters in each test after the MPC sends the optimized signals to the Sliders. By comparing Figure 30 with the other two figures (31,32), it is noticeable that there is no correlation between a stable behavior and frequency of activation. In fact, even after tuning the MPC to make it less reactive towards the thrusters' activation, the number of activations didn't decrease noticeably.

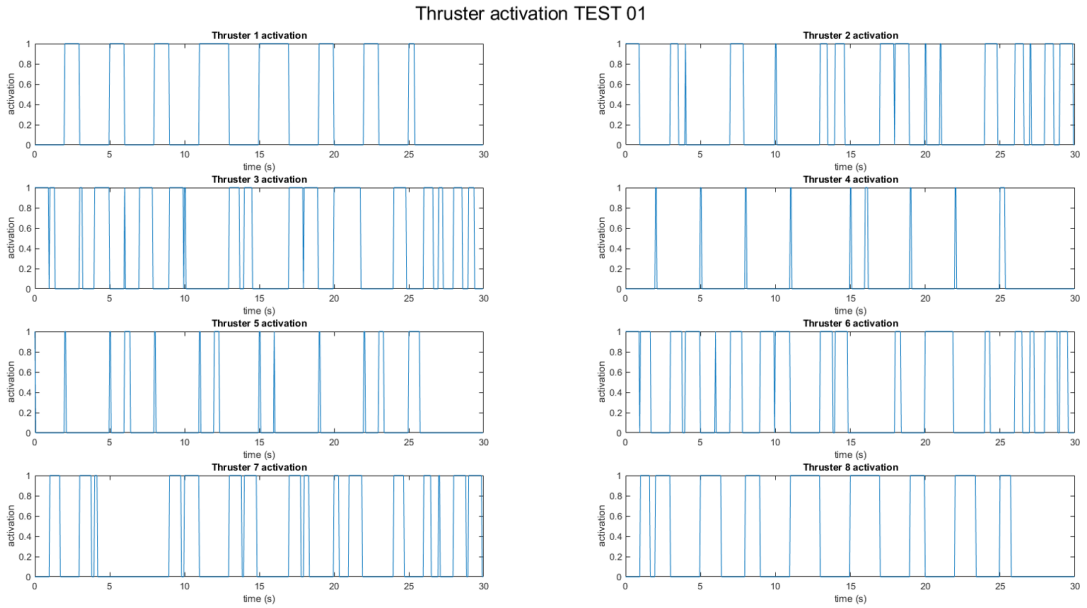


Figure 30: Slider thrust activation for test 01

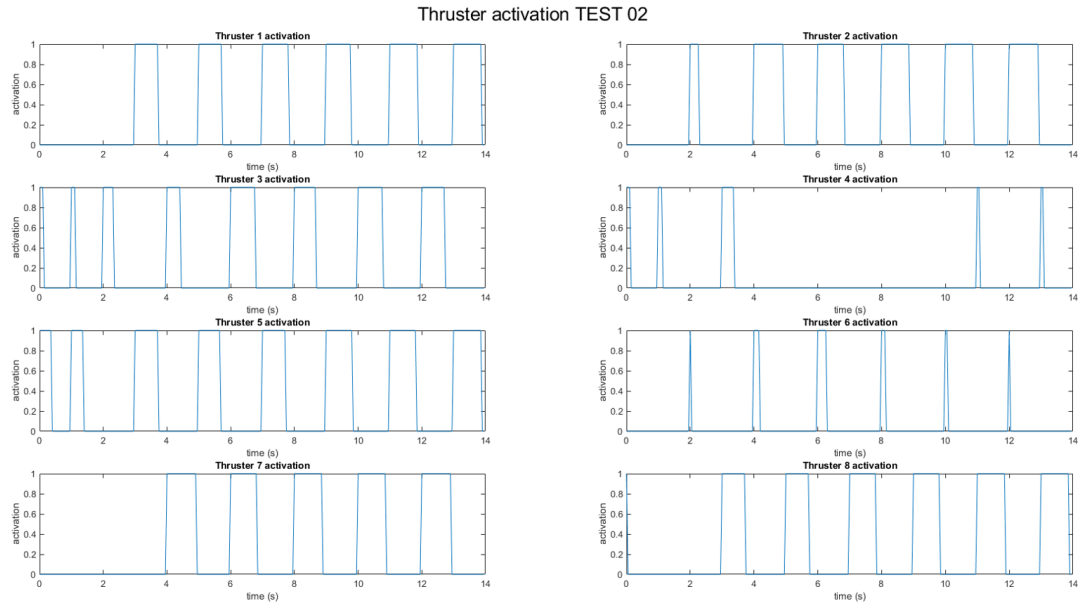


Figure 31: Slider thrust activation for test 02

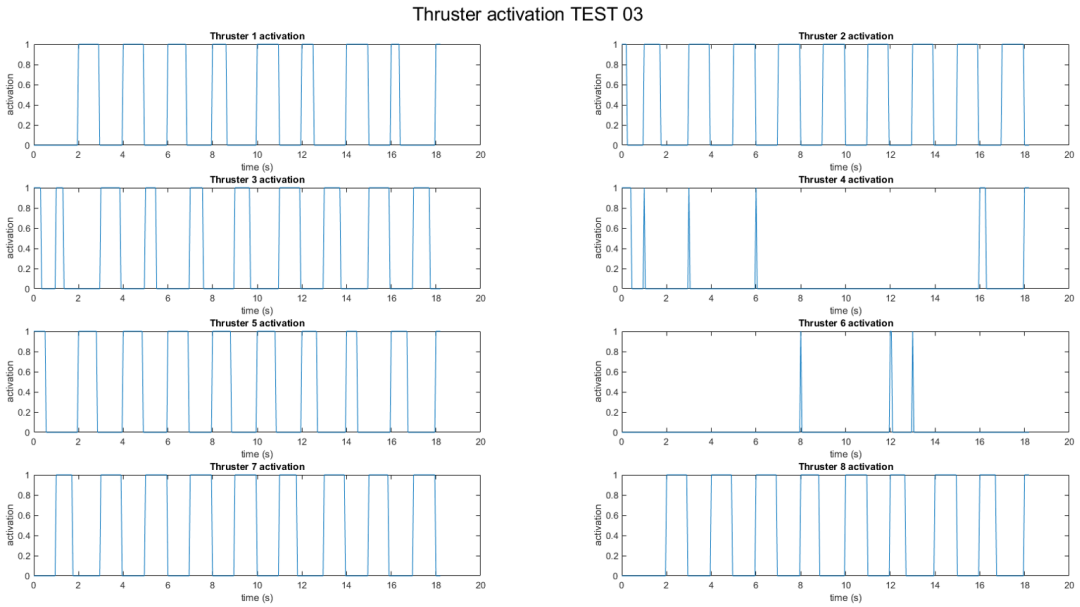


Figure 32: Slider thrust activation for test 03

4.3 Case 2: Cardioid Constraint

The last test involved applying the cardioid constraint to evaluate the Slider's behavior. The settings for this test were the same as the previous one, particularly test 03. The specific values can be found in Table 22. The Slider's initial state was $X_0 = [0_{6\times1}]$ and the final one was $X_f = \left[-0.3,\ 0.5,\ \frac{pi}{6},\ 0_{1\times3}\right]^T$ (Table 21).
The cardioid has been set to have inclination zero and with origin in (-0.3,0.5). The dimensions are the same as the ones used in the software simulations in section 3.

	x (m)	y (m)	θ (rad)
Test Cardioid	-0.3	0.5	0

Table 21: Chaser final position and orientation

Parameters Symbol	Matrix Q Q_k	Matrix R R_k	Orientation θ
Test Cardioid	$[5, 5, 0.005, 50, 50, 0.005] \times I_{6\times6}$	$[0.1, 0.1, 0.5] \times I_{3\times3}$	0

Table 22: Weight matrices Q and R values

The results are shown in Figure 33, and it can be seen that the Slider's trajectory follows a particular pattern. In fact, it show that the MPC can handle the cardioid constraint and keep the Slider close to the cardioid edges, but the trajectory is still not as smooth as the one obtained in the software simulations. As mentioned before, this is due to the low accuracy in the positioning system and the implementation of the PWM.

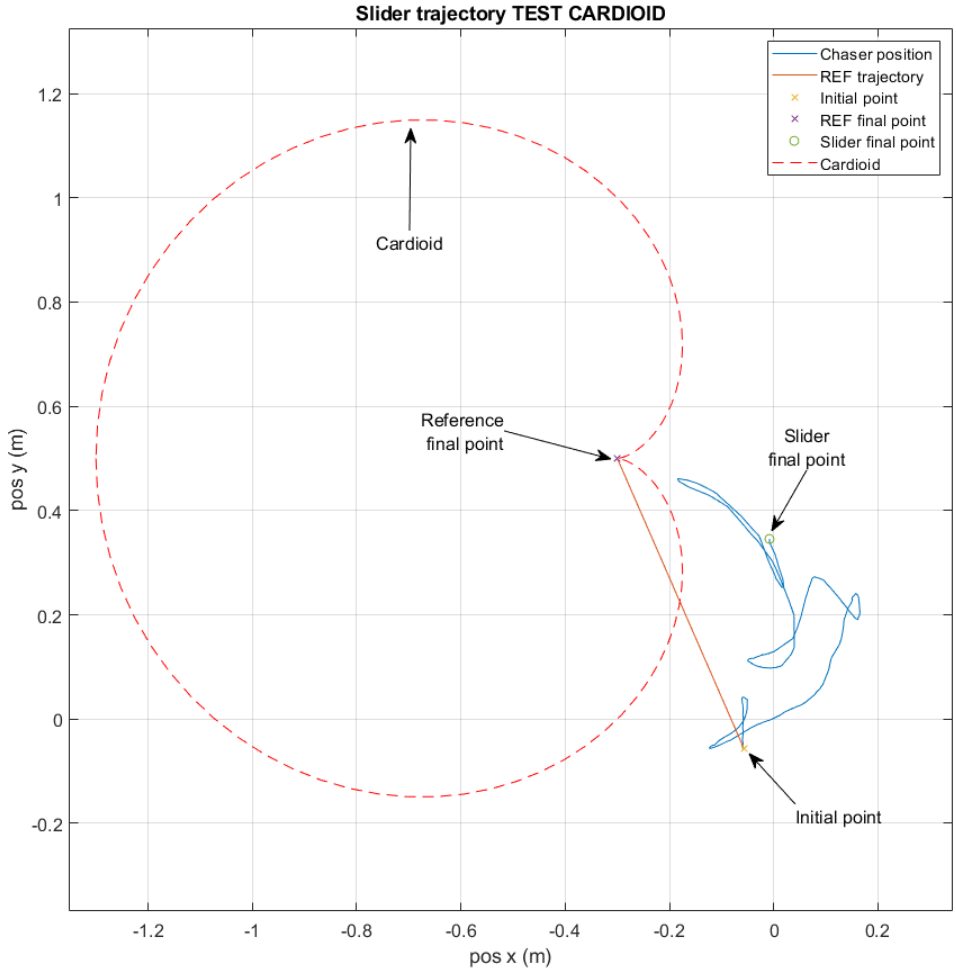


Figure 33: Slider trajectory with cardioid constraint

5 Conclusion and future work

This thesis has shown the feasibility of using a MPC controller to ensure both the reliability and safety of autonomous docking maneuvers applied to Sliders robots.
It has been found that by optimizing the MPC, it was possible to ensure that the controller could let the Chaser safely reach the Target. This was done by adjusting the N, dt, and weight matrices. It has also been found that the maneuvers could be achieved in safety by introducing the right constraint, i.e., cardioid limit function and obstacle avoidance.
Analyzing the results as shown in subsection 3.4 shows that for the fixed Target case, the MPC was very accurate with little error, while for the moving Target one, the error slightly increased. This difference is due to the controller settings requiring more tuning. The objective of the thesis has been achieved in both the fixed and moving Target cases of the software simulations. A small hardware simulation has shown that the strategy used for the software simulations is correct but needs further development and optimization. In fact, low accuracy affected the results of the hardware simulations. A more accurate positioning system can solve the problem and make the trajectory more accurate and similar to the ones obtained in the software simulations.

The work done in this thesis can be improved by developing further the hardware simulations by using a more accurate positioning system and tuning the controller accordingly. In fact, the first hardware tests demonstrated that the software simulations' settings were unsuitable for the hardware ones. This is caused by many factors and requires more time to find the best values for the perfect tuning of the controller. Furthermore, a visual or radar-based navigation system can be implemented in the software simulation to represent a real-case scenario better.

References

[1] J Uri. "55 Years Ago: Gemini VIII, the First Docking in Space". In: *NASA Johnson Space Center* ().

[2] NASA. *NASA - Low Earth Orbit (LEO) Economy Frequently Asked Questions*. 2024. URL: https://www.nasa.gov/humans-in-space/leo-economy-frequently-asked-questions/#:~:text=Low%20Earth%20orbit%20(often%20known,communication%2C%20observation%2C%20and%20resupply.

[3] URL: https://www.esa.int/Space_Safety/Space_Debris/Active_debris_removal.

[4] Donald J Kessler et al. "The kessler syndrome: implications to future space operations". In: *Advances in the Astronautical Sciences* 137.8 (2010), p. 2010.

[5] NASA Small Satellite Systems. *NASA Small Satellite Systems - Deorbit Systems*. 2024. URL: https://www.nasa.gov/smallsat-institute/sst-soa/deorbit-systems/.

[6] Camille Pirat et al. "Vision based navigation for autonomous cooperative docking of CubeSats". In: *Acta Astronautica* 146 (2018), pp. 418–434.

[7] Gilberto Arantes, Luiz S Martins-Filho, et al. "Guidance and control of position and attitude for rendezvous and dock/berthing with a noncooperative/target spacecraft". In: *Mathematical Problems in Engineering* 2014 (2014).

[8] Gilberto Arantes Jr, Ananth Komanduri, and Luiz S Martins Filho. "Guidance for rendezvous maneuvers involving non-cooperative spacecraft using a fly-by method". In: *Proceedings of 21st International Congress of Mechanical Engineering (COBEM'11)*. 2011.

[9] Fabrizio Stesina. "Tracking model predictive control for docking maneuvers of a CubeSat with a big spacecraft". In: *Aerospace* 8.8 (2021), p. 197.

[10] Peng Li, Zheng H Zhu, and SA Meguid. "State dependent model predictive control for orbital rendezvous using pulse-width pulse-frequency modulated thrusters". In: *Advances in Space Research* 58.1 (2016), pp. 64–73.

[11] Jacopo Ventura et al. "Fast and near-optimal guidance for docking to uncontrolled spacecraft". In: *Journal of Guidance, Control, and Dynamics* 40.12 (2017), pp. 3138–3154.

[12] Maidul Islam and Mohamed Okasha. "A comparative study of PD, LQR and MPC on quadrotor using quaternion approach". In: *2019 7th international conference on mechatronics engineering (icom)*. IEEE. 2019, pp. 1–6.

[13] Mohamed Okasha, Jordan Kralev, and Maidul Islam. "Design and Experimental Comparison of PID, LQR and MPC Stabilizing Controllers for Parrot Mambo Mini-Drone". In: *Aerospace* 9.6 (2022), p. 298.

[14] Vladimir Pletser. "Short duration microgravity experiments in physical and life sciences during parabolic flights: the first 30 ESA campaigns". In: *Acta Astronautica* 55.10 (2004), pp. 829–854.

[15] Peter Von Kampen, Ulrich Kaczmarczik, and Hans J Rath. "The new drop tower catapult system". In: *Acta Astronautica* 59.1-5 (2006), pp. 278–283.

[16] William Whitacre. "An autonomous underwater vehicle as a spacecraft attitude control simulator". In: *43rd AIAA Aerospace Sciences Meeting and Exhibit*. 2005, p. 137.

[17] H Benjamin Brown and John M Dolan. "A novel gravity compensation system for space robots". In: *Proceedings of the ASCE specialty conference on robotics for challenging environments*. 1994, pp. 250–258.

[18] Simon Nolet, Edmund Kong, and David W Miller. "Autonomous docking algorithm development and experimentation using the SPHERES testbed". In: *Spacecraft Platforms and Infrastructure*. Vol. 5419. SPIE. 2004, pp. 1–15.

[19] Martin Zwick et al. "ORGL–ESA'S test facility for approach and contact operations in orbital and planetary environments". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Madrid, Spain.* Vol. 6. 2018.

[20] Heike Benninghoff, Toralf Boge, and Tristan Tzschichholz. "Hardware-in-the-loop rendezvous simulation involving an autonomous guidance, navigation and control system". In: (2012).

[21] Wenfu Xu, Bin Liang, and Yangsheng Xu. "Survey of modeling, planning, and ground verification of space robotic systems". In: *Acta Astronautica* 68.11-12 (2011), pp. 1629–1649.

[22] Cristóbal Nieto-Peroy et al. "Simulation of spacecraft formation maneuvers by means of floating platforms". In: *2021 IEEE Aerospace Conference (50100)*. IEEE. 2021, pp. 1–10.

[23] Avijit Banerjee et al. "Slider: On the design and modeling of a 2D floating satellite platform". In: *arXiv preprint arXiv:2101.06335* (2021).

[24] Akshit Saradagi et al. "Safe Autonomous Docking Maneuvers for a Floating Platform based on Input Sharing Control Barrier Functions". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 3746–3753.

[25] Avijit Banerjee et al. "On the design, modeling and experimental verification of a floating satellite platform". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1364–1371.