TÉCNICO LISBOA

Model Predictive Control and Allocation with Nonconvex Constraints for Space Rendezvous

Pedro Alberto Ferreira Taborda

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Daniel de Matos Silvestre
Prof. Rita Maria Mendes de Almeida Correia da Cunha

Examination Committee

Chairperson: Prof. Célia Maria Santos Cardoso de Jesus
Supervisor: Prof. Daniel de Matos Silvestre
Member of the Committee: Prof. Rodrigo Martins de Matos Ventura

November 2023

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

This dissertation would not have come to fruition without the collaboration of several people, to whom I am deeply grateful.

Firstly, I would like to thank my supervisor, Professor Daniel Silvestre, for the guidance and support throughout this work, and Dr. Pedro Lourenço, for his patience, availability and invaluable expertise. This work would not have been possible without their help.

I would also like to thank all my friends and the amazing people I've met at IST, for making these years so memorable. Special thanks to Gonçalo Gomes, Hugo Matias, João Luzio and José Reis, for their friendship and continued support.

Lastly, I would like to thank my family, for their support and encouragement throughout my life, especially during the last few years.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through project FirePuma (https://doi.org/10.54499/PCIF/MPG/0156/2019), through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020, and through COPELABS, University Lusófona project UIDB/04111/2020.

Abstract

Space exploration and satellite-based services have become standard in our daily lives, whether it be through the use of navigation systems, meteorological forecasts, or even satellite television. One of the many problems that arise in the context of space exploration is the rendezvous manoeuvre, where a spacecraft approaches another and matches its velocity, allowing for essential tasks such as refuelling, resupplying, or even crew transfer.

This dissertation presents a rendezvous scenario between a chaser and a target satellite, focusing on using Model Predictive Control (MPC) to build a controller capable of generating firing durations for the chaser's thrusters that drive it to the target. MPC is a flexible control technique based on optimization that, unlike traditional methods, allows for the inclusion of constraints, such as obstacles in the chaser's path. This dissertation also addresses the computational complexity of the optimization problem posed by the MPC controller and presents two algorithms that approximate the solution to this problem, while taking much less time to do so.

The results show that these two algorithms generate trajectories very similar to the optimal solution, making them much more suitable for real-time applications, where the MPC controller is used to generate firing durations for the thrusters in real time. Further work is needed to ensure that the solution generated by these algorithms is close enough to the optimal solution to be useful in practice, as safety and reliability are of utmost importance in space exploration.

Keywords: Spacecraft Rendezvous; Model Predictive Control; Optimization; Obstacle Avoidance

Resumo

A exploração espacial e serviços baseados em satélites tornaram-se comuns no nosso dia-a-dia, quer seja através do uso de sistemas de navegação, previsões meteorológicas, ou até televisão por satélite. Um problema comum no contexto da exploração espacial é a manobra de rendezvous, onde uma nave espacial se aproxima de outra e iguala a sua velocidade, permitindo tarefas essenciais como reabastecimento, ou transferência de tripulação.

Esta dissertação apresenta um cenário de rendezvous entre um satélite perseguidor e um satélite alvo, com foco no uso de controlo preditivo para construir um controlador capaz de gerar durações de disparo para os propulsores do perseguidor que o conduzem ao alvo. Esta é uma técnica de controlo flexível baseada em otimização que, ao contrário de métodos tradicionais, permite a inclusão de restrições no problema de controlo, como obstáculos ao longo da trajetória do perseguidor. Esta dissertação aborda também a complexidade computacional do problema de otimização colocado pelo controlador preditivo, e apresenta dois algoritmos que aproximam a solução deste problema, com um custo computacional muito menor.

Os resultados obtidos mostram que estes algoritmos geram trajetórias semelhantes à solução ótima, tornando-os adequados para aplicações em tempo real, onde o controlador é usado para gerar durações de disparo para os propulsores em tempo real. É necessária investigação adicional para garantir que a solução gerada por estes algoritmos é suficientemente próxima da solução ótima para ser útil na prática, uma vez que a segurança e a fiabilidade são de extrema importância na indústria espacial.

Keywords: Rendezvous Espacial; Controlo Preditivo; Otimização; Desvio de Obstáculos

Contents

List of Tables

List of Figures

Acronyms

ADR active debris removal. 1

AOCS Attitude and Orbit Control System. 1

CW Clohessy-Wiltshire. 8

ECI Earth Centered Inertial. xiii, 5–8, 25

GNC Guidance, Navigation and Control. 1

LVLH Local Vertical Local Horizontal. xiii, 5–8, 13, 25–29, 32

MILP Mixed-Integer Linear Programming. 16, 17

MIP Mixed-Integer Programming. 4

MPC Model Predictive Control. v, xi, xiii, xiv, 1, 3, 4, 12–17, 23, 24, 31, 40–42, 45, 46

OSAM on-orbit servicing, assembly, and manufacturing. 1

Chapter 1

Introduction

1.1 Motivation and Spacecraft Rendezvous

Guidance, Navigation and Control (GNC) and Attitude and Orbit Control System (AOCS) systems handle the control of spacecraft and satellites, which are key components of some services used in our daily lives, such as communication, navigation, and meteorology.

Given the safety standards and high cost of space missions, this industry is very conservative in nature, leading to the use of outdated tools and theoretical methods with a significant focus on safety, robustness, and trust over performance. However, more and more space applications require high-precision and agile control systems that cannot be realized with traditional methods. In particular, on-orbit servicing, assembly, and manufacturing (OSAM) [4], or active debris removal (ADR) [5], are extremely complex up-and-coming proximity operations that require more advanced methods than those that traditional GNC offers.

Within the context of control systems, numerical optimization has emerged in the last decades in the form of the proliferation of MPC. MPC is a widely used control technique that involves solving an optimization problem within a set of actuator constraints, but the computational complexity of MPC can be a challenge in the context of space flight, as actuators are often bang-bang in nature. Moreover, the use of MPC is limited by the onboard computational power, which is usually constrained by the available power budget. Some research has been successful in reducing the computational load of MPC for some space-flight applications, such as [6] and [7], but not for the particular case of spacecraft rendezvous.

Spacecraft rendezvous is the process of bringing two spacecraft together. This is a fundamental problem in space exploration, as it is required for many space missions, such as docking, refueling, and the aforementioned OSAM operations. Rendezvous is a non-trivial problem. If the target spacecraft is in the same circular orbit, but ahead of the chasing spacecraft, simply accelerating towards the target spacecraft will result in a higher altitude, drifting away from the latter. Likewise, if the target spacecraft is behind the chasing spacecraft, accelerating towards the target spacecraft will result in a lower altitude, also drifting away from it. If the target spacecraft is in a different orbit, the problem is even more complex, as the spacecrafts will have different orbital periods.

Hohmann [1] showed that the most efficient way to transfer from one circular orbit to another is by

performing two impulsive manoeuvres, one to transfer to an elliptical orbit, and another to transfer from the elliptical orbit to the target circular orbit, as depicted in Figure 1.1 (presented by [1]) for the coplanar case. This is known as a Hohmann transfer. When both spacecraft are in the same orbit, there are several ways to perform a rendezvous maneuver such as those illustrated in Figure 1.2 (presented by [2]).

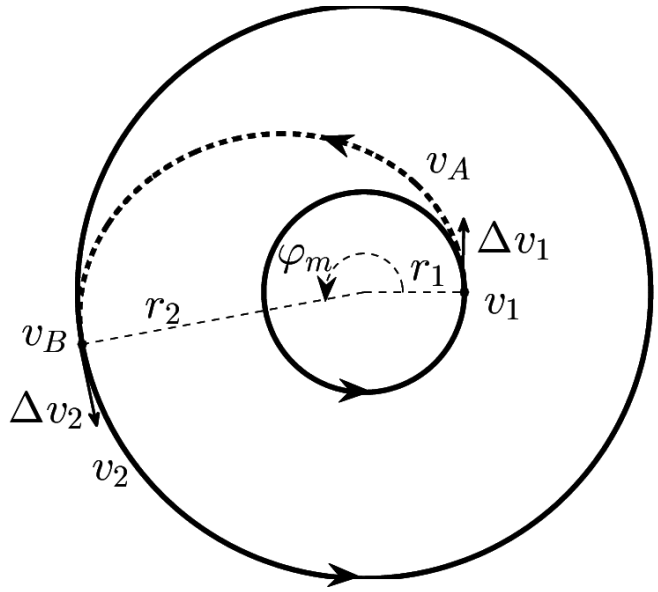


Figure 1.1: Example Hohmann transfer, from circular orbit to circular orbit (with radii r_1 and r_2, respectively), with the transfer orbit (dashed line) tangent to both orbits [1].

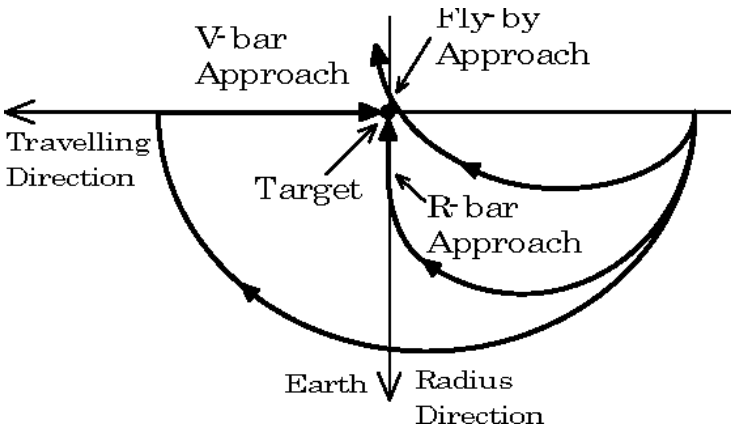


Figure 1.2: Example rendezvous maneuvers for spacecraft in the same orbit [2].

The first successful rendezvous was performed in December 1965, when the Gemini 6A and Gemini 7 spacecrafts met in orbit, keeping a distance below 100 meters for over 5 hours [8]. This was a big landmark in space exploration, achieved only 4 years after the first human spaceflight.

Currently, rendezvous is a standard procedure in space missions. The International Space Station is a good example of this, as it is composed of several modules that were launched separately and then docked together in orbit and requires regular resupply missions. The Hubble Space Telescope was also serviced several times [9]. However, existing rendezvous solutions rely on open-loop control during the initial stages [10, Appendix B], where errors may accumulate and are only rectified during scheduled

correction manoeuvres. An approach based on MPC able to perform rendezvous from a large distance offers the advantage of continuous error correction throughout the mission, possibly minimizing fuel consumption. This is extremely important in scenarios where recurrent rendezvous phases are foreseen, for example for in-orbit assembly of large structures where multiple rendezvous to fetch materials and deliver them to the assembly site must occur [11]. In-orbit servicing is another example, be it for refuelling, maintenance or disposal [5] – the chaser spacecraft has to recurrently rendezvous with different targets and orbits. In these examples, it is very relevant to reduce propellant consumption. Within the full rendezvous (and docking/berthing) phase, the far-range stages are the largest contributors to delta-V expenditure. Therefore, having methods to improve performance (in terms of state dispersion) and, most importantly, reduce consumption during such phases is an enabler for the future missions in these contexts.

1.2 Model Predictive Control

This section provides a brief overview of MPC, as well as of convex optimization techniques that have been successfully used in the past to improve the computational load of MPC, by avoiding solving non-convex problems directly.

1.2.1 Formulation

Suppose we have a discrete-time system that can be described by a state $\mathbf{x} \in \mathbb{R}^n$. The system evolves according to a dynamic model

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{1.1}$$

where $\mathbf{u}_k \in \mathbb{R}^m$ is the control input at time k, $\mathbf{x}_k \in \mathbb{R}^n$ is the state at time k.

The goal is to find a sequence of control inputs $\mathbf{U} \in \mathbb{R}^{m \times N}$ that minimizes a cost function $J(\mathbf{X}, \mathbf{U})$, $\mathbf{X} \in \mathbb{R}^{n \times (N+1)}$, subject to a set of constraints, over a finite time horizon N, given an initial state $\hat{\mathbf{x}}_0$.

The sequence of control inputs $\mathbf{U}$ and the sequence of states $\mathbf{X}$ are composed as

$$\begin{aligned} \mathbf{U} &= \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \dots & \mathbf{u}_{N-1} \end{bmatrix}, \\ \mathbf{X} &= \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \dots & \mathbf{x}_N \end{bmatrix}. \end{aligned} \tag{1.2}$$

The cost function is a function of $\mathbf{X}$ and $\mathbf{U}$ and is used to evaluate the performance of a given sequence of control inputs and is commonly defined as a sum of a quadratic cost on the state and control inputs and a terminal cost on the final state. MPC then solves the optimization problem, finding the sequence of control inputs with the best performance.

The problem can be formulated as

$$
\begin{aligned}
\underset{\mathbf{X},\,\mathbf{U}}{\text{minimize}} \quad & J(\mathbf{X},\mathbf{U}) \\
\text{subject to} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k,\mathbf{u}_k), \quad k = 0,...,N, \\
& \mathbf{x}_0 = \hat{\mathbf{x}}_0, \\
& \mathbf{x}_k \in \mathcal{X} \qquad\qquad k = 0,...,N, \\
& \mathbf{u}_k \in \mathcal{U} \qquad\qquad k = 0,...,N-1.
\end{aligned}
\tag{1.3}
$$

The set $\mathcal{X}$ is the state constraint set, and $\mathcal{U}$ is the control input constraint set. These can represent safety constraints, such as the maximum speed of a vehicle and obstacles to avoid, or physical constraints, such as the maximum steering angle and actuator limits.

After solving the optimization problem, the first control input, $\mathbf{u}_0 \in \mathbb{R}^m$, is applied and the process is repeated. This is the most explored control strategy used throughout this work, as it can deal with constraints, which is not the case for other control techniques such as Linear Quadratic Regulator (LQR).

1.2.2 MPC Optimization

Solving an MPC problem (defined in more detail in Section 1.2.1) involving actuation with discrete states (i.e. on or off), may be performed by using binary variables for the actuation states, and using a Mixed-Integer Programming (MIP) solver. This is the most direct and standard approach and is used in many applications, such as [12].

MIP is a class of optimization problems that contain discrete and (optionally) continuous variables. There exist efficient algorithms to solve some MIP problems, such as the branch-and-bound algorithm [13] (and optimizations of this algorithm for MPC applications [12]). MIP solvers such as CPLEX [14] and Gurobi [15] are widely used and employ this algorithm to solve a wide range of MIP problems. However, these algorithms are not suitable for real-time applications, as they present worst-case exponential complexity. This is because the solution of a MIP is such that the algorithm may need to explore a large portion of the search space to find a feasible solution. This will be the baseline for comparison of the algorithms presented in this work, as, despite its computational complexity, it is the only algorithm that guarantees the optimal solution to the MPC problem among the algorithms presented in this work.

A non-convex problem may have a convex relaxation which is globally optimal at the global optimum of its non-convex counterpart or for some subset of the feasible set of the non-convex problem. Some optimization problems related to the issue at hand, namely minimum-fuel [16], and later, minimum-error [17] rocket landing, have been shown to have convex relaxations. These methods have since been extended to handle generalized versions of the original problem, as presented in [6] and [7]. The issue at hand does not fall exactly into any of the classes of problems studied in the reviewed literature for which convex relaxations have been proven to exist. Still, even if global optimality is not guaranteed, we will use the relaxation proposed in [17] as a reference for other algorithms implemented in this work regarding the solution of an optimization problem with similarly structured constraints on the actuation.

Chapter 2

Methodology

2.1 Motion Dynamics

We will control a chaser spacecraft attempting to rendezvous with a target spacecraft assumed to be in a circular orbit around a central body. We assume the gravitational pull caused by the spacecraft to be negligible. Starting with a description of the reference frames to be used and the equations of motion for a spacecraft in orbit, this section describes the dynamics of the spacecraft and the simplifications made to the model used by the controller.

The chaser spacecraft, denoted as C, is assumed to be a point mass, with a mass m_{C}. The target T is orbiting the Earth, with mass m_{T} and at an altitude R_{T}. The Earth is assumed to be a perfect sphere, with mass M_{E} and radius R_{E}. The universal gravitational constant is denoted by G, and the gravitational parameter of the Earth is $\mu = GM_{\mathrm{E}}$. Note that the radius of the orbit of the target is then given by $R_{\mathrm{T}} + R_{\mathrm{E}}$.

2.1.1 Reference Frames

The simplest way to describe the dynamics of the spacecraft is to express the equations of motion (Newton's second law) on an inertial frame centered but not fixed on the central body, the Earth in this case. Moreover, an inertial frame may also be useful for other tasks, such as analyzing results. Assuming that the masses of the two spacecraft are negligible with respect to the Earth's, their gravity force is neglected. Furthermore, it is assumed that the differences in accelerations due to the gravity of the Sun are not significant, and the gravity of the Moon and other celestial bodies is neglected. Under these assumptions, this frame is the Earth Centered Inertial (ECI) frame. It is defined as having its origin at the centre of the Earth, and it is non-rotating. The z-axis points to the North pole, the x-axis points to Earth's vernal equinox, and the y-axis completes a right-handed reference frame.

On the other hand, since the control problem in this thesis is that of controlling the relative position of the chaser spacecraft with respect to the target spacecraft, a reference frame centered on the latter is useful for the controller, as the target is stationary in this frame and the dynamics can be simplified. This frame is called the Local Vertical Local Horizontal (LVLH) frame. The origin of the LVLH frame is the center of the target spacecraft, and the axes are defined as follows: the z axis – also denoted as

R-bar – points from the target to the center of the Earth (the central body); the y is perpendicular to the orbital plane, pointing in the opposite direction of the angular momentum of the target; and the x axis - also denoted as V-bar – is tangent to the orbit of the target (pointing in the direction of the velocity of the target for circular orbits), completing the right-handed coordinate system. Figure 2.1 shows both frames along with some of the variables used in transformations between the two frames.

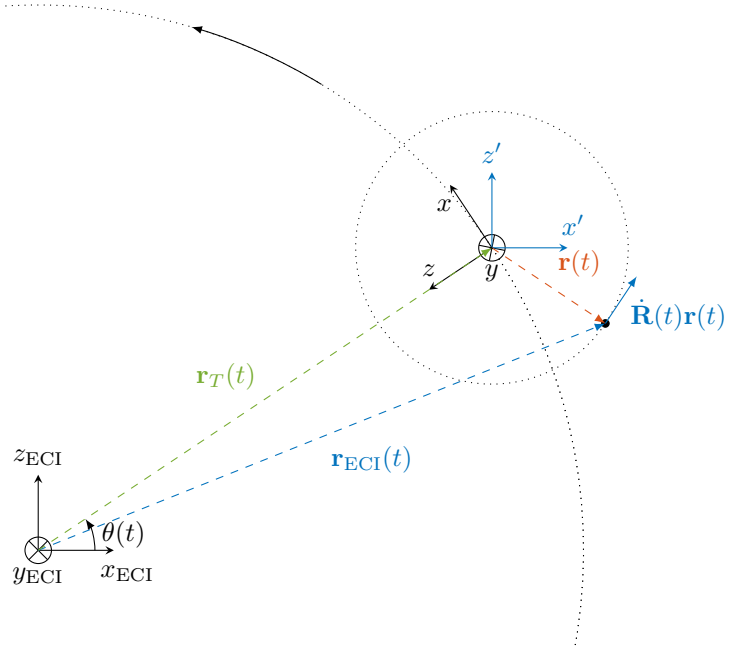


Figure 2.1: Relative motion of the chaser with respect to the target. Position of the chaser relative to the LVLH frame $r(t)$ and relative to the ECI frame $r_{\mathrm{ECI}}(t)$. LVLH frame rotated by $\theta(t)$ (blue axes x' and z'). Physical interpretation of $\dot{\mathbf{R}}(\theta(t))\mathbf{r}(t)$ (blue). Position of the target relative to the ECI frame $r_T(t)$ (green).

Let $\mathbf{r}(t) = [x(t), y(t), z(t)]^\intercal$ be the position of the chaser in the LVLH frame and $\mathbf{r}_T(t)$ the position of the target (by definition equal to the center of the LVLH frame), then the position of the chaser in the ECI frame is given by

$$\mathbf{r}_{\mathrm{ECI}}(t) = \mathbf{R}(t)\mathbf{r}(t) + \mathbf{r}_{\mathrm{T}}(t), \tag{2.1}$$

where $\mathbf{r}_{\mathrm{ECI}}(t) = [x_{\mathrm{ECI}}(t), y_{\mathrm{ECI}}(t), z_{\mathrm{ECI}}(t)]^\intercal$ and $\mathbf{R}(t)$ is the rotation matrix that transforms vectors from the LVLH frame to ECI frame, and that depends on the target's position in the orbit, as well as the orbit itself (in particular its eccentricity and inclination). The sequel will address the dynamics of $\mathbf{r}_{\mathrm{ECI}}(t)$ and $\mathbf{r}_{\mathrm{T}}(t)$.

2.1.2 Orbital Dynamics

In the ECI frame, the dynamics of any celestial object S according to Newton's Second Law are given by

$$m_{\mathrm{S}}\ddot{\mathbf{r}}_{\mathrm{S}}(t) = m_{\mathrm{S}}\ddot{\mathbf{r}}_{\mathrm{S}_g}(t) + \mathbf{u}_{\mathrm{S}}(t), \tag{2.2}$$

where dot notation to denote the derivative with respect to time was used, $\mathbf{u}_{\mathrm{S}}(t)$ represents the forces applied on S, e.g. a control input for a spacecraft, and $\ddot{\mathbf{r}}_{\mathrm{S}_g}(t)$ is the acceleration due to gravity, itself given

by Newton's Law of Universal Gravitation as

$$\ddot{\mathbf{r}}_{\mathrm{S}_g}(t) = -\frac{\mu}{\|\mathbf{r}_{\mathrm{S}}(t)\|^3}\mathbf{r}_{\mathrm{S}}(t). \tag{2.3}$$

Not considering external control forces, all acting forces are central-forces. Therefore, there is conservation of orbital angular momentum of the object. Hence, the orbital motion is planar and it is the same as Kepler's second law, which states that the swept out area during constant time intervals is constant. The solution is then a conic curve: Keplerian orbits are circular, elliptical, parabolic, or hyperbolic. For Earth-orbiting satellites, the first two categories are the most relevant. Assuming, circular orbits, the motion of a celestial object (with $\mathbf{u}_{\mathrm{S}}(t) = 0$) can be described as a circle, with its position entirely defined by the true anomaly angle $\theta(t)$ and the orbital plane (see Figure 2.1).

Therefore, if the target spacecraft is in such conditions, it is possible to achieve a closed form solution for (2.2). In particular, considering without loss of generality that the orbit is circular and equatorial (i.e., inclination is zero), the rotation matrix $\mathbf{R}(T)$ becomes

$$\mathbf{R}(\theta(t)) = \begin{bmatrix} -\sin\theta(t) & 0 & -\cos\theta(t) \\ 0 & 1 & 0 \\ \cos\theta(t) & 0 & -\sin\theta(t) \end{bmatrix}, \tag{2.4}$$

the target's motion is described by

$$\mathbf{r}_T(t) = \begin{bmatrix} R_T\cos\theta(t) \\ 0 \\ R_T\sin\theta(t) \end{bmatrix}, \tag{2.5}$$

and $\theta(t) = \omega t$ is the true anomaly in the Keplerian orbit, i.e., the supplementary of the angle between the z axis of the LVLH frame and the x axis of the ECI frame, as shown in Figure 2.1.

For the inverse transformation, from ECI to LVLH, it is only necessary to invert 2.1, as

$$\mathbf{r}(t) = \mathbf{R}(\theta(t))^{\intercal}(\mathbf{r}_{\mathrm{ECI}}(t) - \mathbf{r}_T(t)). \tag{2.6}$$

Note that $\mathbf{R}(\theta(t))$ is an orthogonal matrix, and therefore $\mathbf{R}(\theta(t))^{\intercal} = \mathbf{R}(\theta(t))^{-1}$. The transformation of the velocity from ECI to LVLH is again given by the derivative of 2.6.

2.1.3 Relative Dynamics

Considering our scenario of a chaser approaching a non-actuating target spacecraft, it is possible to write the relative dynamics as

$$\ddot{\mathbf{r}}_{\mathrm{T}}(t) - \ddot{\mathbf{r}}_{\mathrm{C}}(t) = \ddot{\mathbf{r}}_{\mathrm{T}_g}(t) - \ddot{\mathbf{r}}_{\mathrm{C}_g}(t) - \frac{1}{m_{\mathrm{C}}}\mathbf{u}_{\mathrm{C}}(t), \tag{2.7}$$

where the control input $\mathbf{u}_{\mathrm{C}}(t)$ will be described in Section 2.1.5, As described above, it is important to obtain the equations of motion in the LVLH frame, as the relative dynamics can be further simplified. First of all, recall (2.6), and take its first derivative

$$\dot{\mathbf{r}}(t) = \dot{\mathbf{R}}^{\intercal}(t)\left(\mathbf{r}_{\mathrm{C}}(t) - \mathbf{r}_T(t)\right) + \mathbf{R}^{\intercal}(t)\left(\dot{\mathbf{r}}_{\mathrm{C}}(t) - \dot{\mathbf{r}}_T(t)\right), \tag{2.8}$$

which, calling the last parcel $\mathbf{v}(t)$ and noting that the derivative of $\mathbf{Ra}$ is $\mathbf{R}\omega \times \mathbf{a}$, with ω being the angular velocity of the LVLH frame with respect to the ECI frame, can be rewritten as

$$\dot{\mathbf{r}}(t) = -\omega \times \mathbf{r}(t) + \mathbf{v}(t). \tag{2.9}$$

Taking now the derivative of $\mathbf{v}(t)$ and following similar steps as above yields

$$\dot{\mathbf{v}}(t) = -\omega \times \mathbf{v}(t) + \mathbf{R}^\intercal(t)\left(\ddot{\mathbf{r}}_{\mathrm{C}}(t) - \ddot{\mathbf{r}}_T(t)\right). \tag{2.10}$$

Recalling (2.7) and replacing it in the above expression, allows to write

$$\dot{\mathbf{v}}(t) = -\omega \times \mathbf{v}(t) - \frac{\mu}{\|\mathbf{r}_{\mathrm{T}}(t)\|^3}\mathbf{R}^\intercal(t)\mathbf{r}_{\mathrm{T}}(t) + \frac{\mu}{\|\mathbf{r}_{\mathrm{T}}(t) + \mathbf{R}(t)\mathbf{r}(t)\|^3}\mathbf{R}^\intercal(t)\left(\mathbf{r}_{\mathrm{T}}(t) + \mathbf{R}(t)\mathbf{r}(t)\right) - \frac{1}{m_{\mathrm{C}}}\mathbf{R}^\intercal(t)\mathbf{u}_{\mathrm{C}}(t), \tag{2.11}$$

where the chaser position is replaced using (2.1). Since this is a nonlinear expression, it is difficult to remove the explicit influence of the target position. However, these equations of relative motion in the LVLH frame can be simplified noting that the distance between the two spacecraft is much smaller than the orbital radius of either spacecraft, i.e., $\|\mathbf{R}(t)\mathbf{r}(t)\| \ll \|\mathbf{r}_{\mathrm{T}}(t)\|$. This is the basic assumption behind the model derived in the 60's by Clohessy and Wiltshire [18], which will be further detailed in the next subsection. The next step is then to simplify the above expression as follows

$$\dot{\mathbf{v}}(t) \approx -\omega \times \mathbf{v}(t) - \frac{\mu}{\|\mathbf{r}_{\mathrm{T}}(t)\|^3}\mathbf{R}^\intercal(t)\mathbf{r}_{\mathrm{T}}(t) + \frac{\mu}{\|\mathbf{r}_{\mathrm{T}}(t)\|^3}\mathbf{R}^\intercal(t)\left(\mathbf{r}_{\mathrm{T}}(t) + \mathbf{R}(t)\mathbf{r}(t)\right) - \frac{1}{m_{\mathrm{C}}}\mathbf{R}^\intercal(t)\mathbf{u}_{\mathrm{C}}(t) \tag{2.12}$$

$$= -\omega \times \mathbf{v}(t) + \frac{\mu}{\|\mathbf{r}_{\mathrm{T}}(t)\|^3}\mathbf{r}(t) - \frac{1}{m_{\mathrm{C}}}\mathbf{R}^\intercal(t)\mathbf{u}_{\mathrm{C}}(t). \tag{2.13}$$

If the target is in a circular orbit, these can be further simplified resulting in the Clohessy-Wiltshire (CW) equations as explained next.

Having a model for the relative motion in the LVLH frame, it is relevant to also compute the velocity of the chaser in the ECI frame for the particular case of our equatorial circular orbit. From 2.1,

$$\dot{\mathbf{r}}_{\mathrm{ECI}}(t) = \dot{\mathbf{R}}(\theta(t))\mathbf{r}(t) + \mathbf{R}(\theta(t))\dot{\mathbf{r}}(t) + \dot{\mathbf{r}}_T(t), \tag{2.14}$$

where

$$\dot{\mathbf{R}}(\theta(t)) = \begin{bmatrix} -\omega\cos\theta(t) & 0 & \omega\sin\theta(t) \\ 0 & 0 & 0 \\ -\omega\sin\theta(t) & 0 & -\omega\cos\theta(t) \end{bmatrix}. \tag{2.15}$$

2.1.4 The Clohessy-Wiltshire Equations

For the controller, it is useful to describe the dynamics of the spacecraft through a set of equations that are simpler than the full dynamics, as an optimization problem (constrained by the dynamics) will be solved at each time step. The Clohessy-Wiltshire (CW) equations [18] describe the relative motion of a spacecraft in orbit around a central body. The equations are derived by linearizing the full dynamics around a circular orbit. The equations, in the LVLH frame centered on the target described in Section 2.1.1, are given by

$$\begin{cases} \ddot{x}(t) - 2\omega\dot{z}(t) & = \frac{F_x(t)}{m_c} \\ \ddot{y}(t) + \omega^2 y(t) & = \frac{F_y(t)}{m_c} \\ \ddot{z}(t) + 2\omega\dot{x}(t) - 3\omega^2 z & = \frac{F_z(t)}{m_c} \end{cases}, \tag{2.16}$$

where ω is the angular velocity of the target around the Earth, m_c is the mass of the chaser, and $\mathbf{F}(t) = [F_x(t), F_y(t), F_z(t)]^\intercal$ is the force applied to the chaser. (2.16) can be rewritten in a linear state-space form

$$\dot{\mathbf{x}}(t) = \mathbf{A_c}\mathbf{x}(t) + \mathbf{B_c}\mathbf{u}(t), \tag{2.17}$$

where

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix}, \ \mathbf{u}(t) = \begin{bmatrix} F_x(t) \\ F_y(t) \\ F_z(t) \end{bmatrix}, \ \mathbf{A_c} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix}, \ \mathbf{B_c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_c} & 0 & 0 \\ 0 & \frac{1}{m_c} & 0 \\ 0 & 0 & \frac{1}{m_c} \end{bmatrix}. \tag{2.18}$$

This system is linear, so it can be discretized exactly for constant amplitude pulses as we will see in Section 2.1.6, which is useful because it correctly models the actuators at hand, as we will also see in Section 2.1.5.

2.1.5 Actuation

Propulsion systems used in spacecraft vary in complexity and capabilities. Currently, many propulsion technologies are available [19]. There are chemical and electric propulsion systems, with different types of thrusters. Within chemical propulsion, there are several types of thrusters, such as monopropellant, bipropellant, and cold gas thrusters. These thrusters all share the same principle of operation: a propellant is heated and expelled through a nozzle, generating thrust.

Ideally, a thruster would be able to instantly supply one hundred percent of its rated thrust for any amount of time, and instantly stop when commanded to do so. In reality, this is not the case. Thrusters have some response delay and a minimum amount of time they can be on for. Moreover, they take some time to reach their maximum thrust, and some time to stop. In this work, we will consider a simplified model of a thruster, discarding the delay and rise and fall times, effectively applying a constant force in a single direction for a given amount of time. Figure 2.2 illustrates a typical thruster pulse, and a rectangular pulse approximation.

For a spacecraft with M actuators, assuming each actuator is allowed at most a single pulse, the control input $\mathbf{u}$ is given by

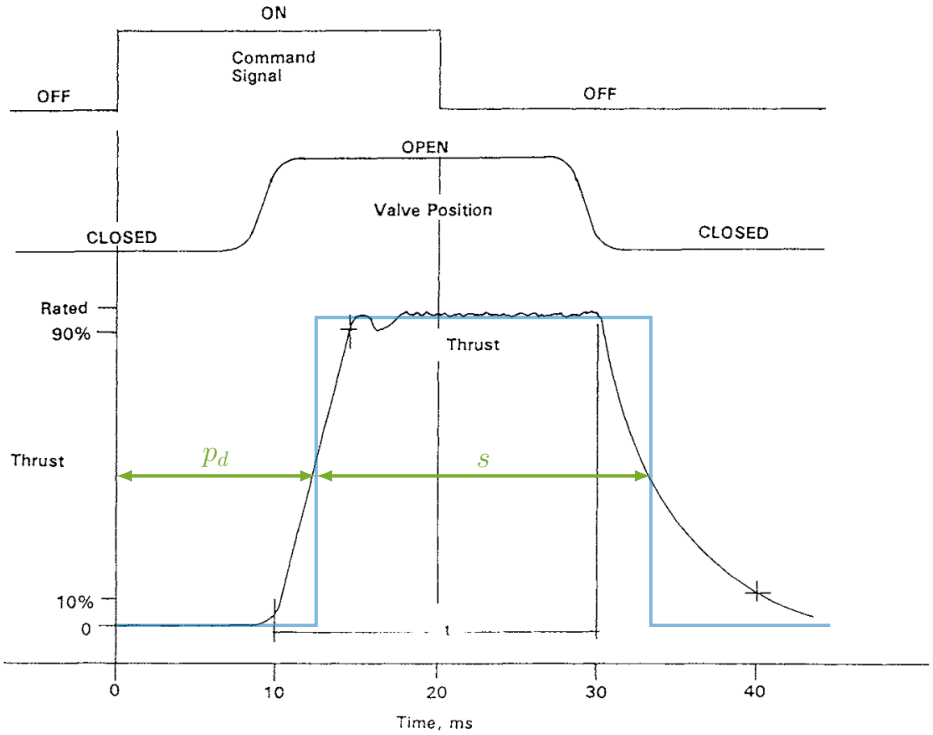


Figure 2.2: Realistic thruster pulse. The thruster takes some time to reach its maximum thrust, and some time to stop. Approximation of a thruster pulse with a rectangular pulse (blue). Pulse delay p_d and duration s. The delay is not considered. Adapted from [3].

$$\mathbf{u}(t) = \sum_{i=1}^{M} F_i \mathbf{w}_i (H(t - \tau_i - s_i) - H(t - \tau_i)) \tag{2.19}$$

$$H(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t \leq 0 \end{cases}, \tag{2.20}$$

where, for actuator i, $t_i^{\min} \in \mathbb{R}^+$ is its minimum pulse duration, $F_i \in \mathbb{R}$ is the maximum magnitude of the force applied by itself, $\mathbf{w}_i \in \mathbb{R}^3$ is a unit vector in the direction of the force, and $H(t - \tau_i - s_i) - H(t - \tau_i)$ is a unit pulse denoting its state, which is 1 (on) starting at $\tau_i \in \mathbb{R}$ for the duration $s_i \in \mathbb{R}^+$, and 0 (off) the rest of the time.

In general, several pulses may be applied, such that

$$\mathbf{u}(t) = \sum_{i=1}^{M} F_i \mathbf{w}_i \sum_{p=1}^{P} H(t - \tau_i^p - s_i^p) - H(t - \tau_i^p), \tag{2.21}$$

where each actuator is turned on at most P times, with the p^{th} pulse for actuator i starting at τ_i^p and lasting for s_i^p.

It is also necessary to ensure that the pulses do not overlap, which can be done by ensuring that the start of the next pulse occurs after the end of the previous pulse, and the minimum pulse duration is respected, i.e.

$$\tau_i^{p+1} \geq \tau_i^p + s_i^p + t_i^{\min}. \tag{2.22}$$

2.1.6 Discretizing the Dynamics

In the notation used for discrete time, the time variable is represented as an index k, representing the time at $t = kT$ (e.g. x_k represents $x(kT)$).

The discrete dynamics can be obtained from the continuous dynamics as

$$\mathbf{x}_{k+1} = e^{\mathbf{A_c}T}\mathbf{x}_k + \sum_{i=1}^{M}\int_0^T e^{\mathbf{A_c}(T-\tau)}\mathbf{B_c}\mathbf{u}_i(\tau)d\tau, \tag{2.23}$$

where $T \in \mathbb{R}^+$ is the sampling time, and $e^{\mathbf{A_c}t}$ is the matrix exponential of $\mathbf{A_c}$, given by

$$e^{\mathbf{A_c}t} = \begin{bmatrix} 1 & 0 & 6(\omega t - \sin(\omega t)) & \frac{4}{\omega}\sin(\omega t) - 3t & 0 & \frac{2}{\omega}(1-\cos(\omega t)) \\ 0 & \cos(\omega t) & 0 & 0 & \frac{1}{\omega}\sin(\omega t) & 0 \\ 0 & 0 & 4 - 3\cos(\omega t) & \frac{2}{\omega}(\cos(\omega t) - 1) & 0 & \frac{1}{\omega}\sin(\omega t) \\ 0 & 0 & 6\omega(1-\cos(\omega t)) & 4\cos(\omega t) - 3 & 0 & 2\sin(\omega t) \\ 0 & -\omega\sin(\omega t) & 0 & 0 & \cos(\omega t) & 0 \\ 0 & 0 & 3\omega\sin(\omega t) & -2\sin(\omega t) & 0 & \cos(\omega t) \end{bmatrix}. \tag{2.24}$$

For actuator i, assuming the control input is applied at the beginning of each time step (applied for $s_i \in [0, T)$), we can calculate the input part of the dynamics as

$$\int_0^T e^{\mathbf{A_c}(T-\tau)}\mathbf{B_c}\mathbf{u}_i(\tau)d\tau = e^{\mathbf{A_c}T}\int_0^{s_i} e^{-\mathbf{A_c}\tau}\mathbf{B_c}F_i\mathbf{w}_i\, d\tau \tag{2.25}$$

$$= e^{\mathbf{A_c}T}\int_0^{s_i} e^{-\mathbf{A_c}\tau}d\tau\mathbf{B_c}F_i\mathbf{w}_i \tag{2.26}$$

$$= e^{\mathbf{A_c}T}\mathbf{G}(s_i)\mathbf{B_c}F_i\mathbf{w}_i, \tag{2.27}$$

where

$$\mathbf{G}(s) = \int_0^s e^{-\mathbf{A_c}\tau}d\tau = \begin{bmatrix} s & 0 & -3s^2\omega + \frac{6(1-\cos(\omega s))}{\omega} & \frac{3s^2}{2} + \frac{4(\cos(\omega s)-1)}{\omega^2} & 0 & \frac{2s}{\omega} - \frac{2\sin(\omega s)}{\omega^2} \\ 0 & \frac{\sin(\omega s)}{\omega} & 0 & 0 & \frac{\cos(\omega s)-1}{\omega^2} & 0 \\ 0 & 0 & 4s - \frac{3\sin(\omega s)}{\omega} & \frac{2\sin(\omega s)}{\omega^2} - \frac{2s}{\omega} & 0 & \frac{\cos(\omega s)-1}{\omega^2} \\ 0 & 0 & 6\omega s - 6\sin(\omega s) & \frac{4\sin(\omega s)}{\omega} - 3s & 0 & 2\frac{\cos(\omega s)-1}{\omega} \\ 0 & 1-\cos(\omega s) & 0 & 0 & \frac{\sin(\omega s)}{\omega} & 0 \\ 0 & 0 & 3(\cos(\omega s) - 1) & 2\frac{1-\cos(\omega s)}{\omega} & 0 & \frac{\sin(\omega s)}{\omega} \end{bmatrix}. \tag{2.28}$$

The matrix describing the dynamics of the actuator $\mathbf{G}$ is a non-linear function of the time the actuator is on, s. $e^{\mathbf{A_c}t}$ and $\mathbf{G}(s)$ were computed using symbolic computation software.

The non-linear discrete-time dynamics are then given by

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{A}\sum_{i=1}^{M}\mathbf{G}(s_i)\mathbf{B_c}F_i\mathbf{w}_i, \tag{2.29}$$

where $\mathbf{A}$ is the state transition matrix, $\mathbf{A} = e^{\mathbf{A_c}T}$.

2.1.7 Linearization of the Discrete Actuator Dynamics

The discrete actuator dynamics $\mathbf{G}(s)$ are non-linear with respect to the control variables. As we want to build an MPC controller which solves an optimization problem constrained by these dynamics at each time step, it is beneficial to linearize the dynamics around a point s_0, so that we can use linear solvers to solve the optimization problem. That is

$$\mathbf{G}(s) \approx \mathbf{G}(s_0) + \left.\frac{\partial \mathbf{G}}{\partial s}\right|_{s=s_0} (s - s_0) := \bar{\mathbf{G}}(s, s_0) \tag{2.30}$$

where $\left.\frac{\partial \mathbf{G}}{\partial s}\right|_{s=s_0}$ is the element-wise derivative of $\mathbf{G}(s)$ with respect to s evaluated at s_0.

$$\left.\frac{\partial \mathbf{G}}{\partial s}\right|_{s=s_0} = \begin{bmatrix} 1 & 0 & 6s_0\omega - 6\sin(\omega s_0) & \frac{4}{\omega}\sin(\omega s_0) - 3s_0 & 0 & \frac{2}{\omega}(1-\cos(\omega s_0)) \\ 0 & \cos(\omega s_0) & 0 & 0 & \frac{1}{\omega}\sin(\omega s_0) & 0 \\ 0 & 0 & 4 - 3\cos(\omega s_0) & \frac{2}{\omega}(\cos(\omega s_0) - 1) & 0 & \frac{1}{\omega}\sin(\omega s_0) \\ 0 & 0 & 6\omega(1-\cos(\omega s_0)) & 4\cos(\omega s_0) - 3 & 0 & 2\sin(\omega s_0) \\ 0 & -\omega\sin(\omega s_0) & 0 & 0 & \cos(\omega s_0) & 0 \\ 0 & 0 & 3\omega\sin(\omega s_0) & -2\sin(\omega s_0) & 0 & \cos(\omega s_0) \end{bmatrix}. \tag{2.31}$$

Note that

$$\bar{\mathbf{G}}(s, 0) = \mathbf{0}_{6\times 6} + s\mathbf{I}_{6\times 6} = s\mathbf{I}_{6\times 6}, \tag{2.32}$$

which is equivalent to fully disregarding the actuator dynamics over the pulse duration, and directly applying force proportional to the pulse duration at the beginning of the pulse.

To demonstrate this, it is only necessary to recalculate the discrete dynamics calculation from 2.23, but instead of $\mathbf{u}_i$, we use $\bar{\mathbf{u}}_i(\tau) = \delta(\tau) s_i F_i \mathbf{w}_i$, where $\delta(\tau)$ is the Dirac delta function. Indeed,

$$\int_0^T e^{\mathbf{A_c}(T-\tau)} \mathbf{B_c} \bar{\mathbf{u}}_i(\tau) d\tau = e^{\mathbf{A_c}T} \int_0^T e^{\mathbf{A_c}\tau} \delta(\tau) d\tau \mathbf{B_c} s_i F_i \mathbf{w}_i \tag{2.33}$$

$$= e^{\mathbf{A_c}T} \mathbf{B_c} s_i F_i \mathbf{w}_i \tag{2.34}$$

$$= e^{\mathbf{A_c}T} (\mathbf{0}_{6\times 6} + s\mathbf{I}_{6\times 6}) \mathbf{B_c} F_i \mathbf{w}_i \tag{2.35}$$

$$= e^{\mathbf{A_c}T} \bar{\mathbf{G}}(s, 0) \mathbf{B_c} F_i \mathbf{w}_i. \tag{2.36}$$

2.2 MPC Controller

2.2.1 The Cost Function

In the formulation of the MPC controller, it is required to define a cost function that guides the behaviour of the controlled system by penalizing certain behaviours. In our case, we want the spacecraft to reach the target and stay there, so the cost function should penalize deviations from the target. Moreover, we want the control effort to be as small as possible, so the cost function should also contain a term penal-

izing the control effort. A convex quadratic cost on the state and control variables will be considered, as it allows for position and velocity control, as well as balancing the control effort. This is enough to control a simple version of the spacecraft rendezvous problem.

In the problem at hand, the control variables are s_i^k, the time for which the i^{th} actuator is active between kT and $(k+1)T$. It is useful to define a variable $\mathbf{S} \in \mathbb{R}^{M \times N}$ containing all the control variables

$$\mathbf{S} = \begin{bmatrix} s_1^1 & s_1^2 & \dots & s_1^N \\ s_2^1 & s_2^2 & \dots & s_2^N \\ \vdots & \vdots & \ddots & \vdots \\ s_M^1 & s_M^2 & \dots & s_M^N \end{bmatrix}. \tag{2.37}$$

Two different cost functions will be used and compared. The first one, defined as follows, penalizes the control variables and the state error over the entire time horizon.

$$J_1(\mathbf{X}, \mathbf{S}) = \sum_{k=1}^{N} \Big((\mathbf{x}_k - \mathbf{x}_{\text{ref}})^\intercal \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_{\text{ref}}) + \mathbf{s^k}^\intercal \mathbf{R} \mathbf{s^k} \Big), \tag{2.38}$$

where $\mathbf{x}_{\text{ref}}$ denotes the reference state, $\mathbf{s}^k$ is the kth column of $\mathbf{S}$, the positive definite matrices $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{R} \in \mathbb{R}^{m \times m}$ penalizes the state error and control variables, respectively. As a reminder, there are $N+1$ states and N control inputs (the initial state $\mathbf{x}_0$ is not controlled).

The second cost function to be considered also penalizes the control variables over the entire horizon, but penalizes the state error only at the final time step. It may be written as

$$J_2(\mathbf{X}, \mathbf{S}) = (\mathbf{x}_N - \mathbf{x}_{\text{ref}})^\intercal \mathbf{Q} (\mathbf{x}_N - \mathbf{x}_{\text{ref}}) + \sum_{k=1}^{N} \mathbf{s^k}^\intercal \mathbf{R} \mathbf{s^k}, \tag{2.39}$$

As $\mathbf{x}$ is the relative position between the chaser and the target spacecraft expressed in the LVLH frame centred in the target, the reference desired position is the origin, i.e., $\mathbf{x}_{\text{ref}} = 0$. Simplifying both cost functions, we obtain

$$J_1(\mathbf{X}, \mathbf{S}) = \sum_{k=1}^{N} \Big(\mathbf{x}_k^\intercal \mathbf{Q} \mathbf{x}_k + \mathbf{s^k}^\intercal \mathbf{R} \mathbf{s^k} \Big), \tag{2.40}$$

and

$$J_2(\mathbf{X}, \mathbf{S}) = \mathbf{x}_N^\intercal \mathbf{Q} \mathbf{x}_N + \sum_{k=1}^{N} \mathbf{s^k}^\intercal \mathbf{R} \mathbf{s^k}. \tag{2.41}$$

2.2.2 Obstacle Avoidance

Using MPC for the rendezvous problem comes with the advantage of being able to easily add constraints to the optimization problem. Obstacles may be expressed as constraints on the state variables, and the MPC controller will avoid them so long as there is a feasible solution - that is, it is possible to avoid them.

The obstacles are assumed to be circles on the orbital plane, each described by a radius $R \in \mathbb{R}^+$ and a center at $\mathbf{C} \in \mathbb{R}^2$. The obstacles are assumed to be static in the LVLH frame and can be imagined as debris fields or other spacecraft.

For a given obstacle i, the constraint on the state variables $\mathbf{x}_k$ is given by

$$\|\mathbf{D}\mathbf{x}_k - \mathbf{C}\|_2 \geq R, \tag{2.42}$$

where $\mathbf{D} \in \mathbb{R}^{2\times 6}$ is a matrix that selects the position components of $\mathbf{x}_k$ on the orbital plane. That is,

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{2.43}$$

The constraint (2.42) is highly non-convex, and cannot be used directly in the optimization problem without resorting to a non-linear solver, which is too computationally expensive for real-time applications and is not guaranteed to find a feasible solution within some specified amount of time. Instead, we use an algorithm very similar to the one described in [20] and [21] to find a feasible solution to the non-convex problem using only linear constraints to approximate the non-convex constraint.

To describe this algorithm, consider the following generic MPC problem with (2.42)

$$\begin{aligned}
\underset{\mathbf{X},\,\mathbf{U}}{\text{minimize}} \quad & J(\mathbf{X},\mathbf{U}) && && \text{(2.44a)}\\
\text{subject to} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k,\mathbf{u}_k), && k = 0,...,N, && \text{(2.44b)}\\
& \mathbf{x}_0 = \hat{\mathbf{x}}_0, && && \text{(2.44c)}\\
& \|\mathbf{D}\mathbf{x}_k - \mathbf{C}\|_2 \geq R && k = 0,...,N. && \text{(2.44d)}
\end{aligned}$$

The main idea of the algorithm is to first solve (2.44) without the obstacle constraint (2.44d), and if the solution is infeasible, add affine constraints tangent to the obstacle boundary, at the points nearest to each of the infeasible states. A pseudo-code description of the algorithm is presented in Algorithm 1.

Algorithm 1 Sequential generation of affine constraints for avoiding a circular obstacle.

Input: $\hat{\mathbf{x}}_0$, $\mathbf{C}$, $\mathbf{R}$
Output: $\mathbf{U}$
$\mathbf{X}, \mathbf{U}$ **=** solve (2.44) without the obstacle constraint (2.44d)
if $\mathbf{X}$ is feasible **then**
    Return $\mathbf{U}$
end if
while $\mathbf{X}$ not feasible or $\mathbf{U}$ has not converged **do**
    newConstraints = $\emptyset$
    for each currently or previously infeasible state $\mathbf{x}_k$ **do**
        add to **newConstraints** a plane tangent to the obstacle boundary at the nearest point to $\mathbf{D}\mathbf{x}_k$
    end for
    $\mathbf{X}, \mathbf{U}$ **:=** solve (2.44) with the constraints **newConstraints**, without the obstacle constraint (2.44d)
end while
Return $\mathbf{U}$

A visual representation of the constraint boundaries generated by the algorithm for a trajectory with

infeasible points is shown in Figure 2.3.

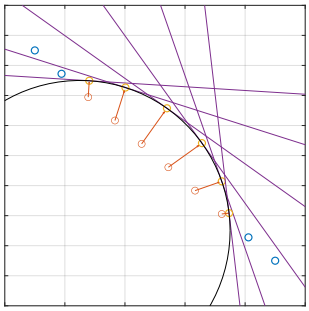


Figure 2.3: Example trajectory (blue - feasible - and orange - infeasible), constraint boundaries generated by the algorithm for the infeasible point (purple) and the nearest points on the circle to each of the infeasible points of the trajectory (yellow).

2.2.3 MPC Optimization Problem

The optimization problem solved by the MPC can be explicitly written as

$$\underset{\mathbf{X},\,\mathbf{S}}{\text{minimize}} \quad J(\mathbf{X},\mathbf{S}) \tag{2.45a}$$

$$\text{subject to} \quad \mathbf{x}_{k+1} = e^{\mathbf{A_c}T}\mathbf{x}_k + \sum_{i=1}^{M} e^{\mathbf{A_c}T}\bar{\mathbf{G}}(s_i^k, s_0)\mathbf{B_c}F_i\mathbf{w}_i, \tag{2.45b}$$

$$\mathbf{x}_0 = \hat{\mathbf{x}}_0, \tag{2.45c}$$

$$s_i^k \in 0 \cup [t_i^{\min}, T), \qquad i = 1,\dots,M, \quad k = 0,\dots,N, \tag{2.45d}$$

$$\|\mathbf{D}_j\mathbf{x}_k - \mathbf{C}_j\|_2 \geq R \qquad j = 1,\dots,O, \quad k = 0,\dots,N, \tag{2.45e}$$

where O is the number of obstacles.

Note that the non-linear actuator dynamics $\mathbf{G}(s)$ (2.28), if used directly, would be a source of non-convexity, requiring the use of non-linear solvers to solve directly. In order to enable the use linear solvers, these dynamics are replaced by the linearized actuator dynamics $\bar{\mathbf{G}}(s, t^{\min})$ (2.30).

2.3 Simulator Layout

To produce results, a basic simulator was implemented. There are two main components to the simulator: the dynamics model and the controller. The dynamics model is responsible for simulating the motion of the spacecraft, given a control signal. The controller generates the control signal. The dynamics model and the controller are described in Sections 2.1.2 and 2.2, respectively.

The simulator has fixed time steps of T seconds. The controller matches the time step of the simulator. It is assumed the chaser spacecraft has ideal sensors capable of measuring its correct position and velocity. The controller generates the control signal based on the position and velocity at the end of the previous time step. The control signal is then applied to the dynamics model, which simulates the motion of the spacecraft for the next time step. The process is repeated until the end of the simulation time is

reached. The dynamics model is described in Section 2.1.2. A diagram of the simulator is shown in Figure 2.4. The simulator is used to test the controller and to generate the results presented in Chapter 3.

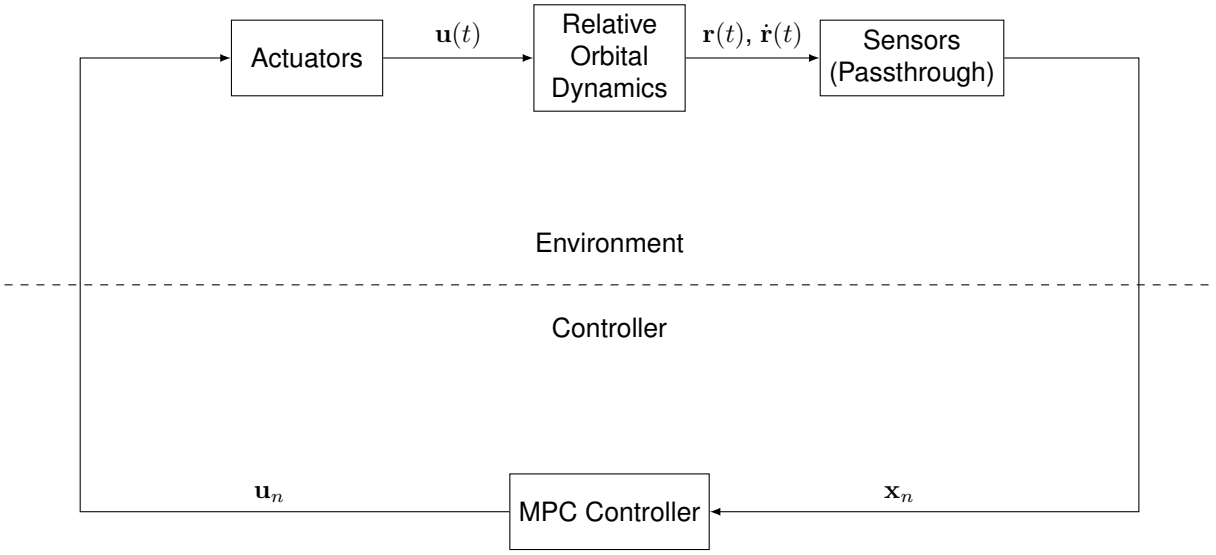


Figure 2.4: One time step of the simulator for testing the MPC controller. The controller receives the state $\mathbf{x}_n$ and generates a control signal $\mathbf{u}_n$. The control signal is converted to a force vector over time $\mathbf{u}(t)$ by the actuators, which is applied to the continuous time dynamics model. The position and velocity are measured by the sensors - in this case, the sensors are assumed to be perfect - and passed to the controller.

2.4 Solver Algorithms

The optimization problem becomes a mixed-integer problem due to the constraints on each actuator's time of activation, $s_i^k \in 0 \cup [t^{\min}, T)$. Several approaches will be tested to solve the optimization problem, exploring the trade-off between computational cost and the optimality of the solution. The following algorithms (name in bold) will be tested:

1. **Standard**: Use of a Mixed-Integer Linear Programming (MILP) solver, in this case, `GUROBI` in MATLAB, to directly (and optimally) solve the simplified optimization problem 2.45.
2. **Relaxed**: Current state-of-the-art algorithm for convexification of similar mixed-integer problems to convex problems.
3. **Projected**: Proposed algorithm for approximating the mixed-integer solution using only convex problems as intermediate steps.

An important note is that using a mixed-integer nonlinear programming (MINLP) solver, such as `fmincon` in MATLAB, would be the most direct approach to solving the full problem with non linear dynamics. However, not only is this approach not feasible for real-time applications due to its high computational cost, but it is also prohibitively expensive to run even simple simulations in a reasonable amount of time. Therefore, this approach is not considered in this thesis.

Obstacle avoidance and exploring efficient solutions for the mixed-integer problem are problems considered separately in this thesis. In practice, this means obstacle avoidance is only implemented for the **Standard** algorithm, serving mostly as a proof of concept for the flexibility of the MPC formulation, and the remaining algorithms are only tested on the problem without obstacles. A joint analysis of the effect of the obstacle avoidance algorithm and the relaxations introduced by the **Relaxed** and **Projected** algorithms falls beyond the scope of this work.

2.4.1 Algorithm 1, Standard

The simplified optimization problem 2.45 without the obstacle constraint is a MILP problem, which can be solved using a mixed-integer solver. In this case, the `GUROBI` solver in MATLAB is used.

Without obstacles, this approach is optimal in the sense that it guarantees the optimal solution to the simplified optimization problem 2.45, therefore it is used as a baseline for comparing the performance of the remaining algorithms.

However, this approach is not feasible for real-time applications, as it presents worst-case exponential complexity on the number of integer constraints. The fact that the number of integer constraints is equal to the prediction horizon N multiplied by the number of actuators M presents a significant limitation to the prediction horizon that can be used while keeping the worst-case computation time within reasonable limits. This is the main motivation for the remaining algorithms.

2.4.2 Algorithm 2, Relaxed

This algorithm is a simple relaxation of the integer constraints of the problem 2.45, leading to a convex problem. This relaxation was motivated by the fact that similarly structured problems have been shown to have convex relaxations that are globally optimal at the global optimum of the non-convex problem [6]. Even though the problem at hand does not fall exactly into any of the classes of problems studied in the reviewed literature for which convex relaxations have been proven to exist, we will use the relaxation proposed in [17] both to analyse the performance of this algorithm despite the lack of theoretical guarantees, and as a reference for the other algorithms implemented in this work.

Because there is no guarantee that the solution to the relaxed problem is feasible for the original problem, solutions to the relaxed problem that are not feasible for the original problem are projected onto the nearest feasible solution space of the original problem. The actuator times are unidimensional, so this projection is trivial.

For completeness, this section presents the full optimization problem for which [17] presents a convex relaxation, along with the conditions under which the relaxation is globally optimal at the global optimum of the non-convex problem.

Proposed by [6], and generalized on [7], this algorithm is one of the current methods for solving a subset of mixed-integer MPC problems. The setup for this algorithm requires introducing a great deal of new notation. For it not to become too convoluted, the notation within this section is self-contained and does not follow the notation used in the rest of the thesis.

Consider the following optimization problem:

$$\underset{u_i, \gamma_i, t_f}{\text{minimize}} \quad m(t_f, x(t_f)) \tag{2.46a}$$

$$\text{subject to} \quad \dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}\sum_{i=1}^{M} u_i(t) + w, x(0) = x_0, \tag{2.46b}$$

$$\gamma_i(t)\rho_1 \leq \|u_i(t)\|_2 \leq \gamma_i(t)\rho_2, \qquad i = 1, \ldots, M, \tag{2.46c}$$

$$\gamma_i(t) \in \{0, 1\}, \qquad i = 1, \ldots, M, \tag{2.46d}$$

$$\sum_{i=1}^{M} \gamma_i(t) \leq K, \qquad i = 1, \ldots, M, \tag{2.46e}$$

$$\mathbf{C}_i u_i(t) \leq 0, \qquad i = 1, \ldots, M, \tag{2.46f}$$

$$b(t_f, x(t_f)) = 0, \tag{2.46g}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u_i(t) \in \mathbb{R}^p$ is the control input of actuator i, $w \in \mathbb{R}^n$ is a known external input, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the state transition matrix and $\mathbf{B} \in \mathbb{R}^{n \times p}$ is the control input matrix. Matrices $\mathbf{C}_i \in \mathbb{R}^{q_i \times p}$ define polytopic cones constraining the input vectors (the j-th row of $\mathbf{C}_i$, $C_{i,j}$ defines the outward-facing normal of the j-th facet of the polytopic cone). $b : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^q$ defines the terminal manifold. $m : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ is the cost function (only a terminal cost is considered). $\rho_1, \rho_2 \in \mathbb{R}$ bound the norm of the control input when the corresponding $\gamma_i(t)$ is active. K is the maximum number of actuators that can be active at any given time.

A few more definitions are needed to describe the conditions under which this algorithm can be used. The *input pointing sets* are the cones constraining the control inputs:

$$\mathcal{U}_i = \{u_i \in \mathbb{R}^p \mid \mathbf{C}_i u_i \leq 0\}. \tag{2.47}$$

The Euclidean projection of $z \in \mathbb{R}^p$ onto a set $\mathcal{S} \subseteq \mathbb{R}^n$ is $\mathcal{P}_{\mathcal{U}_i}(z) \triangleq \|\text{argmin}_{s \in \mathcal{S}} \|s - z\|_2\|_2$. The normal cone at z to a set $\mathcal{S} \subseteq \mathbb{R}^n$ is denoted $\mathcal{N}_{\mathcal{S}}(z) \subseteq \mathbb{R}^n$. The *adjoint system* is defined as

$$\dot{\lambda}(t) = -\mathbf{A}^T \lambda(t), \quad y(t) = \mathbf{B}^T \lambda(t), \tag{2.48}$$

where $y(t) \in \mathbb{R}^m$ is its output. The *input gain measure* is $\Gamma_i(t) \triangleq \mathcal{P}_{\mathcal{U}_i}(y(t))$.

According to [7], given that

1. the cones defined by the matrices $\mathbf{C}_i$ do not overlap,
2. matrices $\mathbf{C}_i$ are full row rank and the terminal cost is non-trivial ($\nabla m(t_f, x(t_f)) \neq 0$),
3. the control norm bounds ρ_1 and ρ_2 are distinct,
4. the adjoint system (2.48) is observable,
5. either (a) or (b) hold:
   (a) $\Gamma_i(t) \neq 0$ a.e. $t \in [0, t_f]$ for all $i \neq j$, s.t. $y(t) \notin \mathcal{N}_{\mathcal{U}_i}(0)$

(b) on any interval where $\Gamma_i(t) = 0$, $\Gamma_j(t) > 0$ for at least K other inputs.

6. either (a) or (b) hold:

   (a) $\Gamma_i(t) \neq \Gamma_j(t)$ a.e. $t \in [0, t_f]$ for all $i \neq j$, s.t. $y(t) \notin \mathcal{N}_{\mathcal{U}_i}(0)$

   (b) on any interval where $\Gamma_i(t) = \Gamma_j(t)$, there exist K inputs with $\Gamma_k(t) > \Gamma_i(t)$ or $M - K$ inputs with $\Gamma_k(t) < \Gamma_i(t)$.

7. the following intersection holds:

$$\operatorname{range} \begin{bmatrix} \nabla_x b[t_f]^\intercal \\ \nabla_t b[t_f]^\intercal \end{bmatrix} \cap \operatorname{cone} \begin{bmatrix} \nabla_x m[t_f]^\intercal \\ \nabla_t m[t_f]^\intercal \end{bmatrix} = \{0\}, \tag{2.49}$$

   where $\operatorname{cone}(\mathbf{W}), \mathbf{W} \in \mathbb{R}^{n_1 \times n_2}$ denotes the conical hull of the columns of $\mathbf{W}$ and $\operatorname{range}(\mathbf{W}), \mathbf{W} \in \mathbb{R}^{n_1 \times n_2}$ denotes the vector space of all possible linear combinations of the columns of $\mathbf{W}$.

then the following relaxed version of the problem (2.46) may be constructed, and its solution is globally optimal a.e. $t \in [0, t_f]$ for problem (2.46):

$$\underset{u_i, \gamma_i, \sigma_i, t_f}{\text{minimize}} \quad m(t_f, x(t_f)) \tag{2.50a}$$

$$\text{subject to} \quad \dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}\sum_{i=1}^{M} u_i(t) + w, x(0) = x_0, \tag{2.50b}$$

$$\gamma_i(t)\rho_1 \leq \sigma_i(t) \leq \gamma_i(t)\rho_2, \qquad i = 1, \ldots, M, \tag{2.50c}$$

$$\|u_i(t)\|_2 \leq \sigma_i(t), \qquad i = 1, \ldots, M, \tag{2.50d}$$

$$0 \leq \gamma_i(t) \leq 1, \qquad i = 1, \ldots, M, \tag{2.50e}$$

$$\sum_{i=1}^{M} \gamma_i(t) \leq K, \qquad i = 1, \ldots, M, \tag{2.50f}$$

$$\mathbf{C}_i u_i(t) \leq 0, \qquad i = 1, \ldots, M, \tag{2.50g}$$

$$b(t_f, x(t_f)) = 0, \tag{2.50h}$$

As mentioned, the structure of the problem for which the algorithm is proposed in this thesis is different from the one presented in [7], leading to no guarantees of applicability of the algorithm. However, it is the closest to the problem considered in this thesis, and thus it is used for comparison.

In particular, one important part of the cost function selected in Section (2.2.1) is the control effort, which is not present in the cost function of [7]. Moreover, the actuator configuration considered (number of actuators and thrust directions, specified in Table 3.1) would make it so conditions 5 and 6 are not satisfied.

2.4.3 Algorithm 3, Projected

This algorithm, proposed in this thesis, is based on the idea of leveraging information from a relaxed version of the problem to make decisions regarding which actuators to activate during the first time step.

Consider a set of binary variables $\alpha \in \{0,1\}^M$ and $\beta \in \{0,1\}^M$, which will be used to constrain the actuator activation times during the first time step of the prediction horizon $s_i^0,\ i = 1,\dots,M$. If $\alpha_i = 1$ then $s_i^0 \in [t_{\min_i}, T)$, and if $\beta_i = 1$ then $s_i^0 = 0$. α_i and β_i are mutually exclusive, i.e. $\alpha_i\beta_i = 0$ for all $i = 1,\dots,M$.

A relaxed version of the optimization problem 2.45, where the constraints on the actuator activation times are relaxed to be convex and include the binary variables α and β, is then solved. This optimization problem may be written as

$$\underset{\mathbf{X},\,\mathbf{S}}{\text{minimize}} \quad J(\mathbf{X},\mathbf{S}) \tag{2.51a}$$

$$\text{subject to} \quad \mathbf{x}_{k+1} = e^{\mathbf{A_c}T}\mathbf{x}_k + \sum_{i=1}^{M} e^{\mathbf{A_c}T}\bar{\mathbf{G}}(s_i^k, s_0)\mathbf{B_c}F_i\mathbf{w}_i, \tag{2.51b}$$

$$\mathbf{x}_0 = \hat{\mathbf{x}}_0, \tag{2.51c}$$

$$s_i^0 \in [\alpha_i t_{\min}, T(1-\beta_i)), \qquad i = 1,\dots,M,, \tag{2.51d}$$

$$s_i^k \in [0, T), \qquad i = 1,\dots,M, \quad k = 1,\dots,N \tag{2.51e}$$

By changing (2.45d) to (2.51d) and (2.51e), the problem becomes convex and therefore can be solved efficiently, but may generate unfeasible solutions.

The algorithm consists of solving the relaxed problem (2.51) with $\alpha = \mathbf{0}$ and $\beta = \mathbf{0}$, and checking if the solution is feasible for the original problem (2.45). If it is, the algorithm returns the solution. If it is not, for each infeasible actuation within the first time step, s_i^0, the algorithm sets $\alpha_i = 1$ if s_i^0 is nearer to $t_{\min_i}$ than to 0, and $\beta_i = 1$ otherwise. The relaxed problem is then solved again and the solution is checked for feasibility, repeating the aforementioned process until a feasible solution is found.

In essence, the first relaxed problem solved by the algorithm fully ignores the set of unfeasible values for the actuator activation durations, $(0, t_{\min})$, within the first step of the prediction horizon, $s_i^0,\ i = 1,\dots,M$. Setting some α and β elements to 1 locks the corresponding actuators' states, constraining their activation durations to either $\{0\}$ or $[t_{\min}, T)$. Solving the relaxed problem with these values for α and β retrieves a solution that is both guaranteed to be feasible for the locked actuators and optimal for the relaxed problem (2.51). The activation times of the still fully relaxed actuators may now be unfeasible for the original problem (2.45) due to these new constraints, making it necessary to repeat the process of locking actuator states and finding a new solution.

The algorithm is summarized in Algorithm 2. Note that the algorithm is guaranteed to terminate (and find a feasible solution) after at most M iterations, as in each iteration at least one of the binary variables α_i or β_i is set to 1, and any combination of α_i and β_i where $\sum_{i=1}^{M} \alpha_i + \beta_i = M$ is feasible for the original problem (2.45) (noting that $\alpha_i\beta_i = 0$ for all $i = 1,\dots,M$ throughout all iterations).

Algorithm 2 Description of algorithm **Projected**.

Input: $\mathbf{x}_0$
Output: $\mathbf{S}$
Initialize α and β to all zeros
do
    Solve the relaxed problem (2.51) to obtain $\mathbf{S}$
    s^0 is the first column of $\mathbf{S}$, corresponding to the first time step
    if s^0 is feasible under 2.45d **then**
        Return $\mathbf{S}$
    else
        for each $i \in \{1, \ldots, M\}$ **do**
            if $0 < s_i^0 < t_{\min_i}$ **then**
                if $s_i^0 < t_{\min_i}/2$ **then**
                    $\beta_i \leftarrow 1$
                else
                    $\alpha_i \leftarrow 1$
                end if
            end if
        end for
    end if
while true

Chapter 3

Results

3.1 Simulation Setup

The simulator described in Section 2.3 was implemented in Matlab [22]. The optimization problems were solved using Gurobi [15].

3.1.1 Parameters Used

The default values for each parameter are presented in 3.2 and Table 3.1.

For simplicity, we will consider only cases where all actuators share the same minimum actuation time $t^{\min}$. That is,

$$t_i^{\min} = t^{\min}, \quad i = 1, \dots, M. \tag{3.1}$$

Table 3.1: Default simulation parameters.

Parameter	Value	Description
G	$6.674 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$	Gravitational constant
M_{E}	$5.972 \times 10^{24}\,\mathrm{kg}$	Earth mass
R_{E}	$6371\,\mathrm{km}$	Earth radius
R_{T}	$800\,\mathrm{km}$	Target orbit altitude
R_{C}	$700\,\mathrm{km}$	Chaser orbit altitude
m_{C}	$2000\,\mathrm{kg}$	Chaser mass
M	6	Number of thrusters
$\mathbf{w}_i$	(3.2)	Actuator directions
F_i	$1000\,\mathrm{N} \qquad i = 1, \dots, 6$	Actuator thrust
$\mathbf{Q}$	$\mathbf{1}_{6\times 6}$	State error weight matrix
$\mathbf{R}$	$\mathbf{1}_{6\times 6}$	Control effort weight matrix
T_{sim}	$3600\,\mathrm{s}$	Total simulation time
$t_{\min}$	$5\,\mathrm{s}$	Minimum actuation time
$t_{\max}$	$10\,\mathrm{s}$	Maximum actuation time
s_0	$t_{\min}$	Actuator dynamics linearization point[1]
T	$10\,\mathrm{s}$	Step size
N	10	MPC prediction horizon
J	J_2	MPC cost function
$\mathbf{x}_0$	$\begin{bmatrix} R_{\mathrm{T}} - R_{\mathrm{C}} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$	Initial state

[1] This means by default, a change in $t_{\min}$ implies a corresponding update in s_0.

$$\mathbf{w}_1 = -\mathbf{w}_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2 = -\mathbf{w}_5 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_3 = -\mathbf{w}_6 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.2}$$

For the tested parameters, the initial position and velocity along the y axis (perpendicular to the orbital plane, xz) are always zero. Because there is no interaction between the dynamics along the y axis and the dynamics along the xz plane, the position and velocity maintain zero throughout the simulations. For this reason, the trajectories are only shown on the xz plane. The total simulation time, T_{sim}, is set to $3600\,\mathrm{s}$ such that the full approach is completed within most configurations of the other parameters.

3.1.2 Mission Time and Fuel Spent

Considering a few quantitative metrics to evaluate the performance of the MPC controller in the rendezvous scenario may be useful to draw conclusions about the performance of the controller for different parameters, outside of the qualitative analysis possible by observing the trajectories and actuator activations. Two such metrics are the mission time and the fuel spent.

1. **Mission Time:** The mission time is the time t_m such that the distance between the chaser and the target does not exceed $1\,\mathrm{km}$ for all $t \geq t_m$.

2. **Fuel Spent:** The fuel spent is the total amount of fuel spent by the chaser during the mission. As all actuators are assumed to be identical except for their direction, the fuel spent by each actuator is measured by the total amount of time it is activated during the mission. The fuel spent is then the sum of the fuel spent by each actuator, measured in seconds.

There are limitations to these metrics. The end of mission does not take into account the velocity of the chaser relative to the target, which may eventually need to reach zero, nor does it take into account the fact that the chaser may be moving away from the $1\,\mathrm{km}$ boundary before the simulation ends. Even with these limitations, these metrics were considered useful to compare the performance of the MPC controller for different parameters, as in practice, for the simulations performed, the chaser was not to exceed the $1\,\mathrm{km}$ boundary after reaching it.

3.2 Model Validity

This section showcases the different results obtained by each model described in Chapter 2. Four models are compared: the full dynamics directly using Newton's Universal Law of Gravitation (2.3), solved numerically using MATLAB's `ode45`; the CW equations (2.16), also solved numerically using MATLAB's `ode45`; the CW equations discretized for pulses (2.29); and finally the discrete CW equations with linearized actuator dynamics (2.29 using 2.30). Throughout this section, each of these models will be referred to as 'full dynamics', 'continuous CW', 'discrete CW', and 'linearized CW', respectively.

Even though there should be no difference between the continuous and discrete CW models, except for integration errors introduced by the numerical solver, both are shown to validate the implementation of the discrete CW model.

3.2.1 Experiment 1 - Free Motion

This simulation ran for $T_{\text{sim}} = 10\,000\,\text{s}$, so that more than one orbit around the Earth occurred and the error in the CW models became apparent. No actuators fired during this time. This experiment mostly serves to highlight the differences between the full dynamics and the rest of the models, as the linearized CW model is a simplification of the discrete CW model only on the actuator dynamics. This is what we observe, as in Figures 3.1 and 3.2, showing the trajectories generated by each model in the LVLH and ECI frames, respectively, we see only two lines, one for the full dynamics and one for the rest of the models.

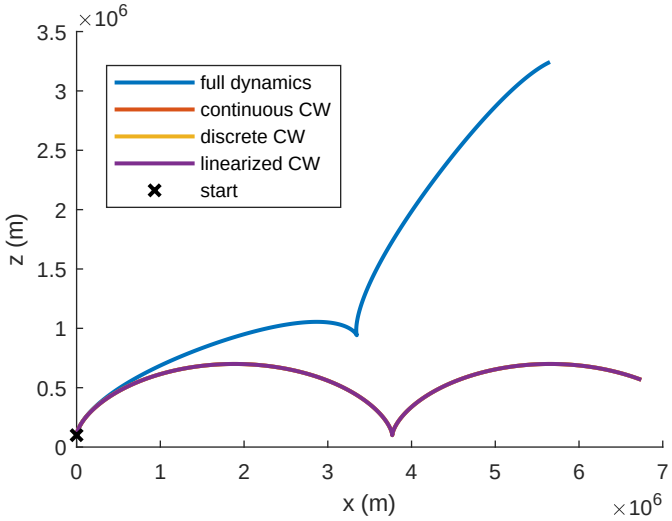


Figure 3.1: In-plane trajectory (on the LVLH frame) generated by the different models, without any actuation.

Figure 3.2 makes it evident how unrealistic the CW models' trajectories become the further away the chaser is from the target. This is clear by observing that the chaser does not orbit the Earth in the CW models, but instead spirals outwards.

The error plots in Figure 3.3 contain the Euclidean distance between corresponding positions (w.r.t. time) from the trajectories generated by each of the listed models to the trajectory of a reference model.

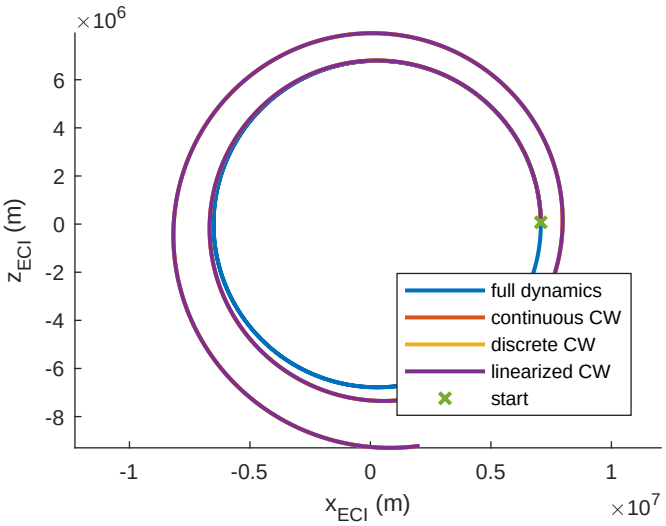
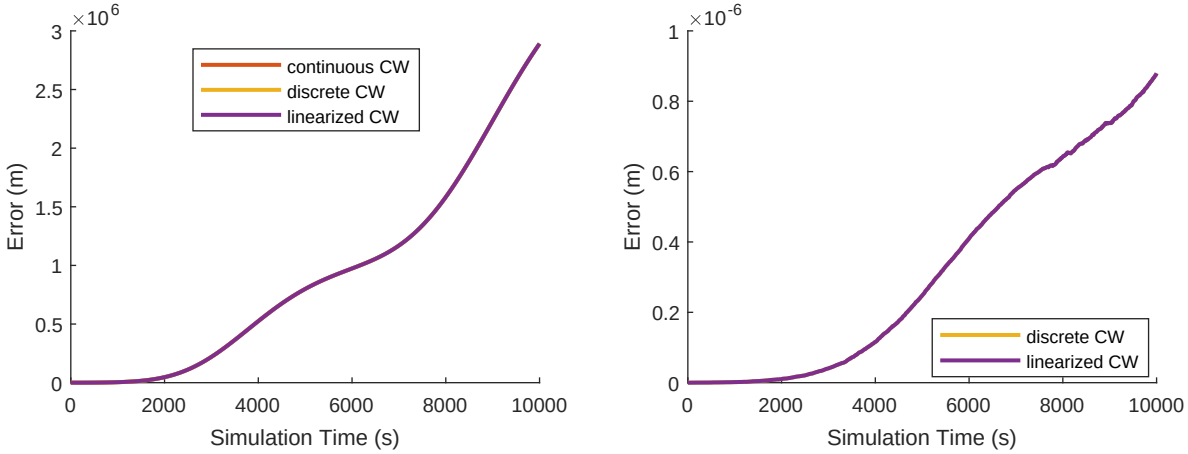


Figure 3.2: In-plane trajectory (on the ECI frame) generated by the different models, without any actuation.

The reference model in Figure 3.3a is the full dynamics, and in Figure 3.3b it is the continuous CW model. We can verify the overlapping trajectory lines are indeed all the CW models by looking at these error plots. At the end of the simulation, the error between the full dynamics and the CW models is of the order of 10^6 meters, and the error between the CW models only about 10^{-6} meters.



(a) Error between the trajectory generated by the full dynamics model and the trajectory generated by each of the CW models.

(b) Error between the trajectory generated by the continuous CW model and the trajectory generated by the discrete CW and linearized CW models.

Figure 3.3: Experiment 1 error plots. The full dynamics model produces a trajectory that diverges from the CW models over time. The CW models produce similar trajectories.

Even though these absolute error plots are informative, we are more interested in the relative error between the models, as the relative error is independent of the scale of the trajectory - a $50\,\mathrm{m}$ drift is much more significant if the total distance travelled is $100\,\mathrm{m}$ than if it is $10\,000\,\mathrm{m}$. This is why we also include the relative error plots in Figure 3.4. The relative error is calculated as the absolute error divided by the total distance travelled by the reference model, in the LVLH frame, and shown as a percentage.

We can see that the relative error between the full dynamics and the CW models is well below $10\,\%$ for the first 2000 seconds (200 time steps) of the simulation, and sharply increases afterwards. In practice,

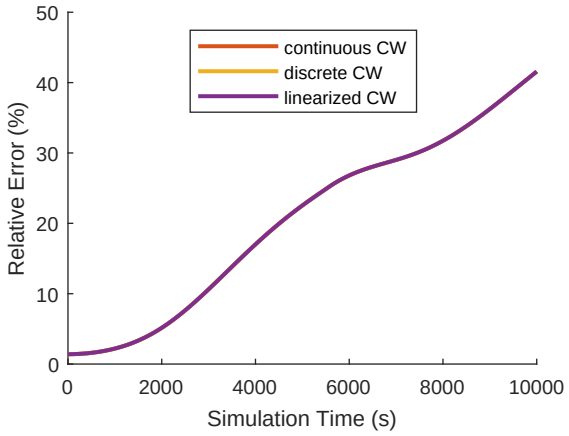


Figure 3.4: Relative error between the trajectory generated by the full dynamics model and the trajectory generated by each of the CW models.

for the simulations performed in this thesis, the prediction horizon used by the controller is of the order of 100 seconds, never reaching 250 seconds. Table 3.2 shows the absolute and relative errors for some key points in the simulation, especially those near to the prediction horizon used by the controller, and we can see that the relative error is below $2\,\%$ even at the $500\,\mathrm{s}$ mark.

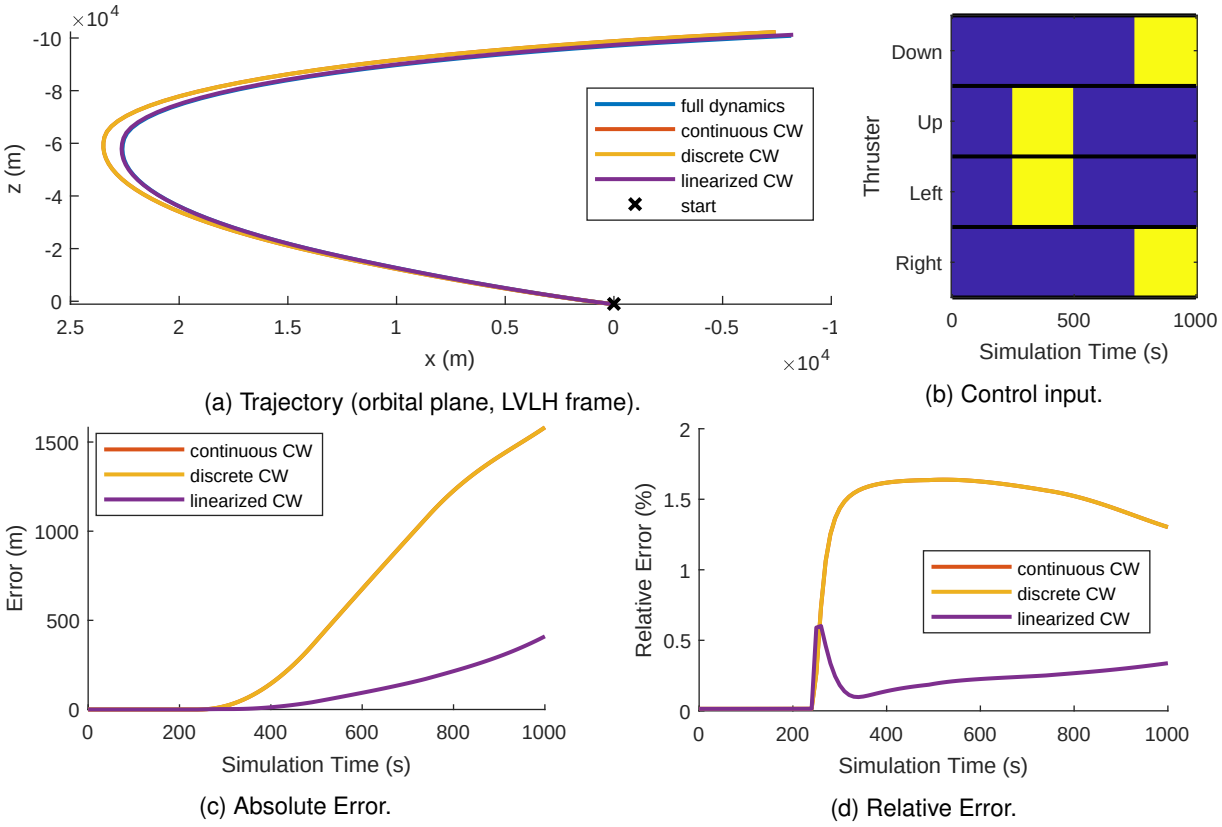
Table 3.2: Experiment 1. Distance travelled (full dynamics) and absolute and relative error between the full dynamics and the discrete CW model.

Time	Distance Travelled	Absolute Error	Relative Error (%)
$50\,\mathrm{s}$	$411\,\mathrm{m}$	$3.69\,\mathrm{m}$	$0.90\,\%$
$100\,\mathrm{s}$	$1.65\,\mathrm{km}$	$18.8\,\mathrm{m}$	$1.14\,\%$
$150\,\mathrm{s}$	$3.72\,\mathrm{km}$	$45.8\,\mathrm{m}$	$1.23\,\%$
$500\,\mathrm{s}$	$42.9\,\mathrm{km}$	$650\,\mathrm{m}$	$1.51\,\%$
$1000\,\mathrm{s}$	$189\,\mathrm{km}$	$4.02\,\mathrm{km}$	$2.12\,\%$

3.2.2 Experiment 2 - Actuated

A few simulations (this time lasting for $1000\,\mathrm{s}$) were performed to validate the actuator dynamics by applying different control inputs. The results are shown in Figures 3.5, 3.6, 3.7 and 3.8. Again, the discrete and continuous CW models produce similar trajectories. The linearized CW model seems to be a good approximation of the discrete CW model, with relative errors below $2.5\,\%$ for all simulations. In the next section, we will see that this performance is highly dependent on both the control input chosen and the linearization point s_0 used.

The control input is shown in each Figure as a function of time, yellow if the thruster is active (firing during the entire time step) and blue if it is not. The thrusters are labeled relative to the reader's viewpoint: 'Right', 'Left', 'Up' and 'Down' refer to the thrusters pointing to negative x, positive x, negative z and positive z, respectively. Note that the axes are inverted to more explicitly match the LVLH frame, with the z axis pointing into the Earth and the x axis pointing in the direction of motion.



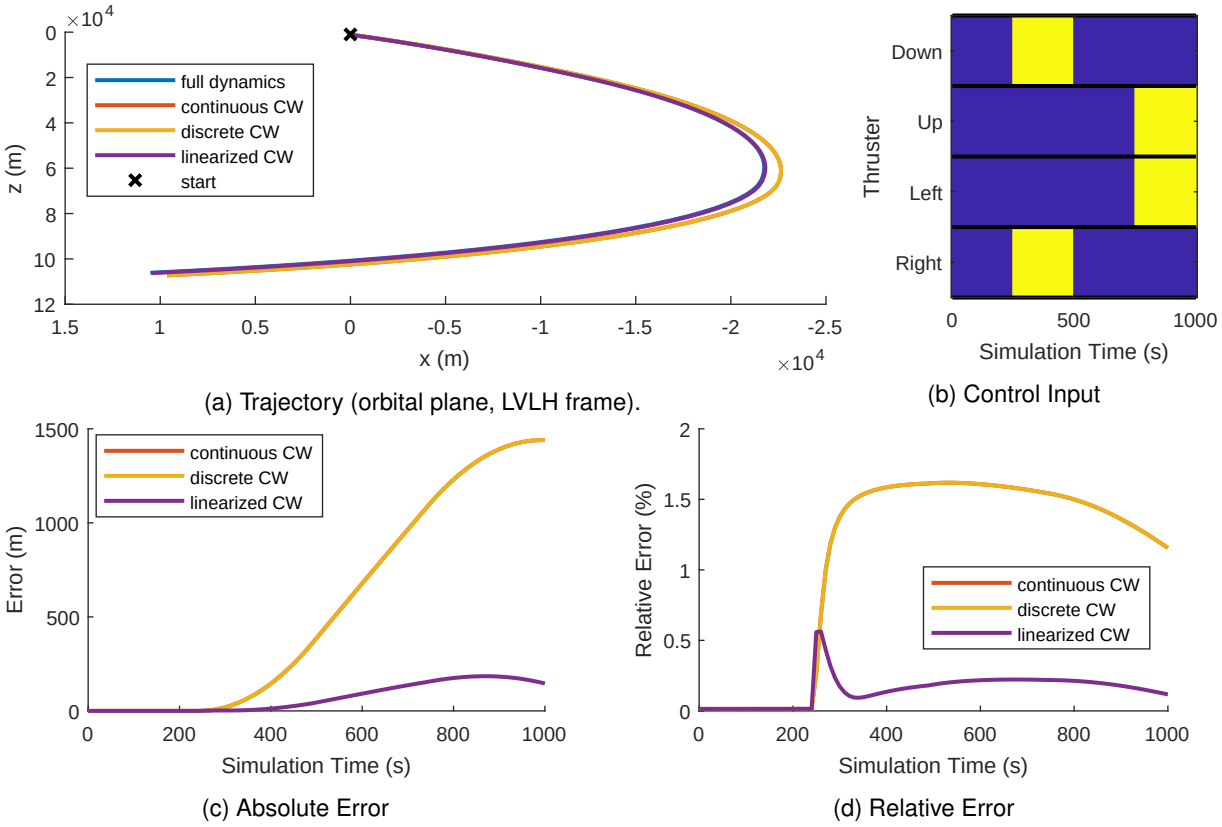
(a) Trajectory (orbital plane, LVLH frame).

(b) Control input.

(c) Absolute Error.

(d) Relative Error.

Figure 3.5: Trajectory, control input (blue - inactive, yellow - active) and errors for case 1.



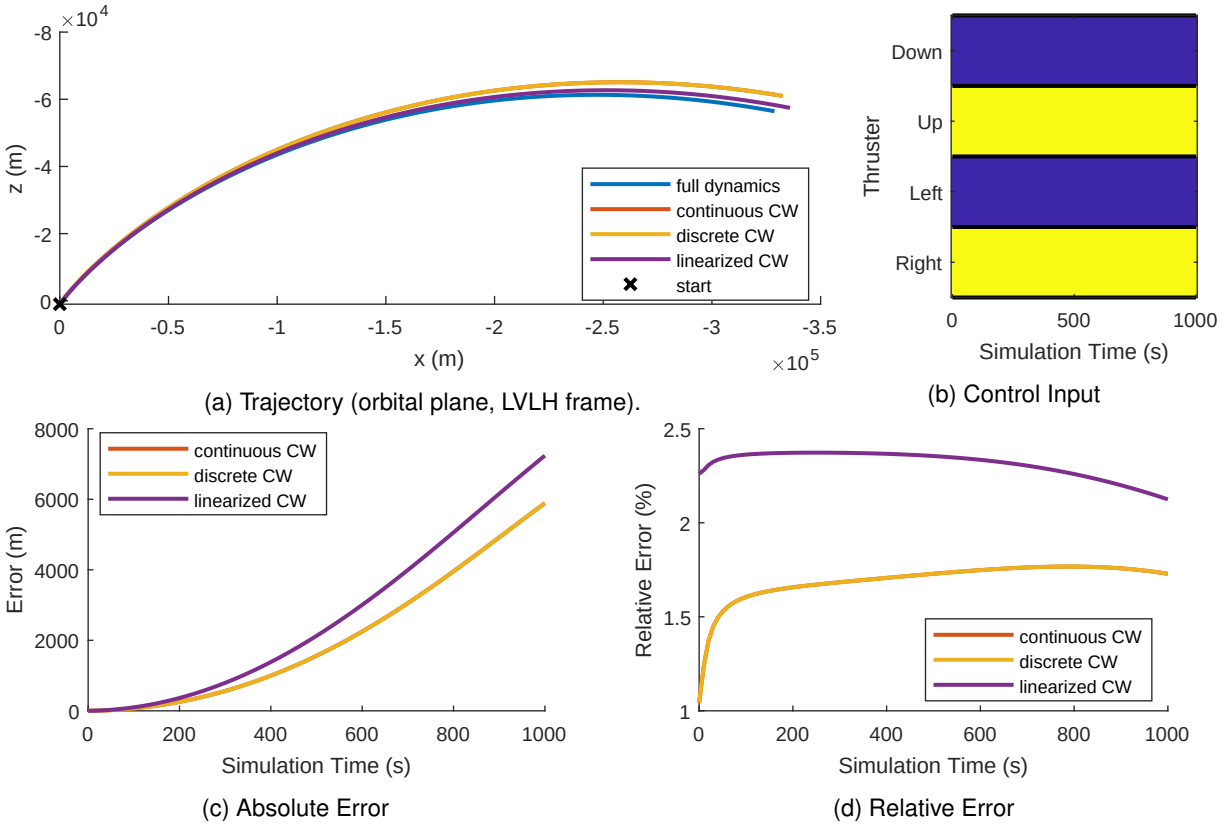
(a) Trajectory (orbital plane, LVLH frame).

(b) Control Input

(c) Absolute Error

(d) Relative Error

Figure 3.6: Trajectory, control input (blue - inactive, yellow - active) and errors for case 2.



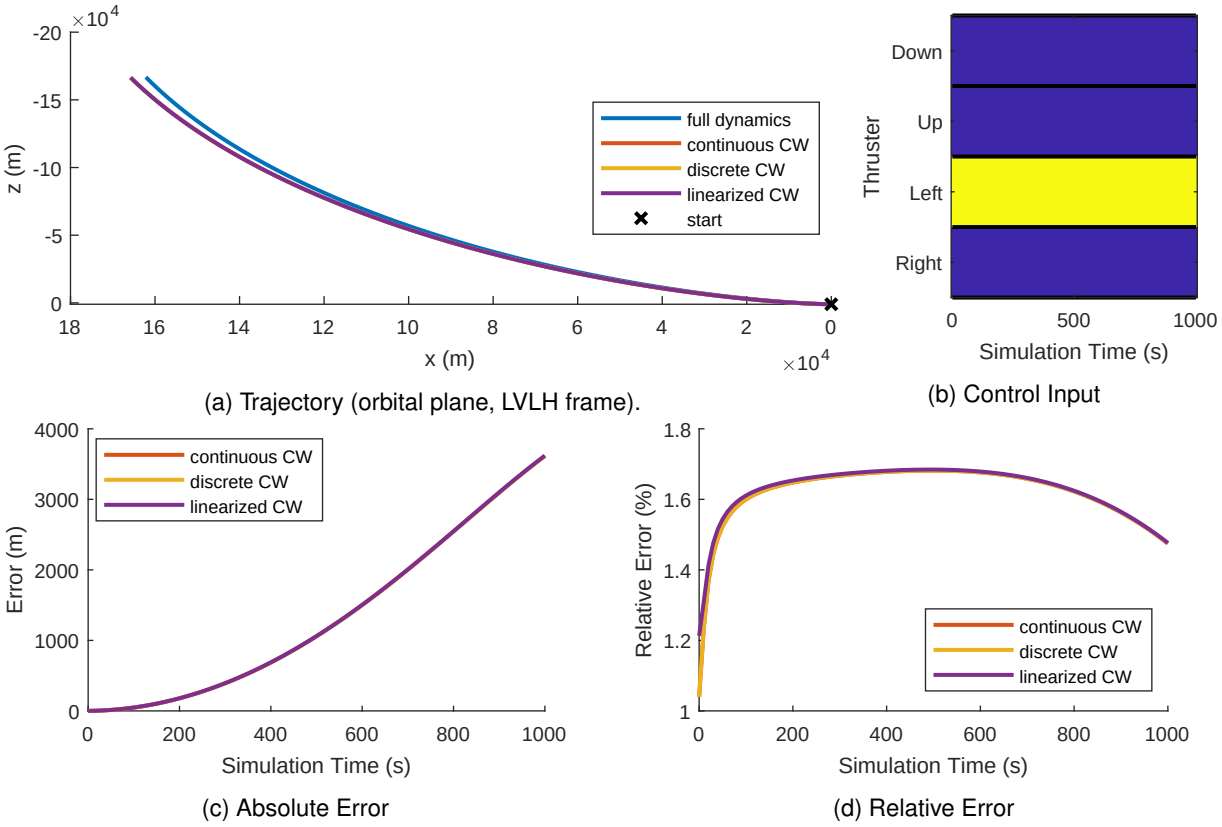
(a) Trajectory (orbital plane, LVLH frame).

(b) Control Input

(c) Absolute Error

(d) Relative Error

Figure 3.7: Trajectory, control input (blue - inactive, yellow - active) and errors for case 3.



(a) Trajectory (orbital plane, LVLH frame).

(b) Control Input

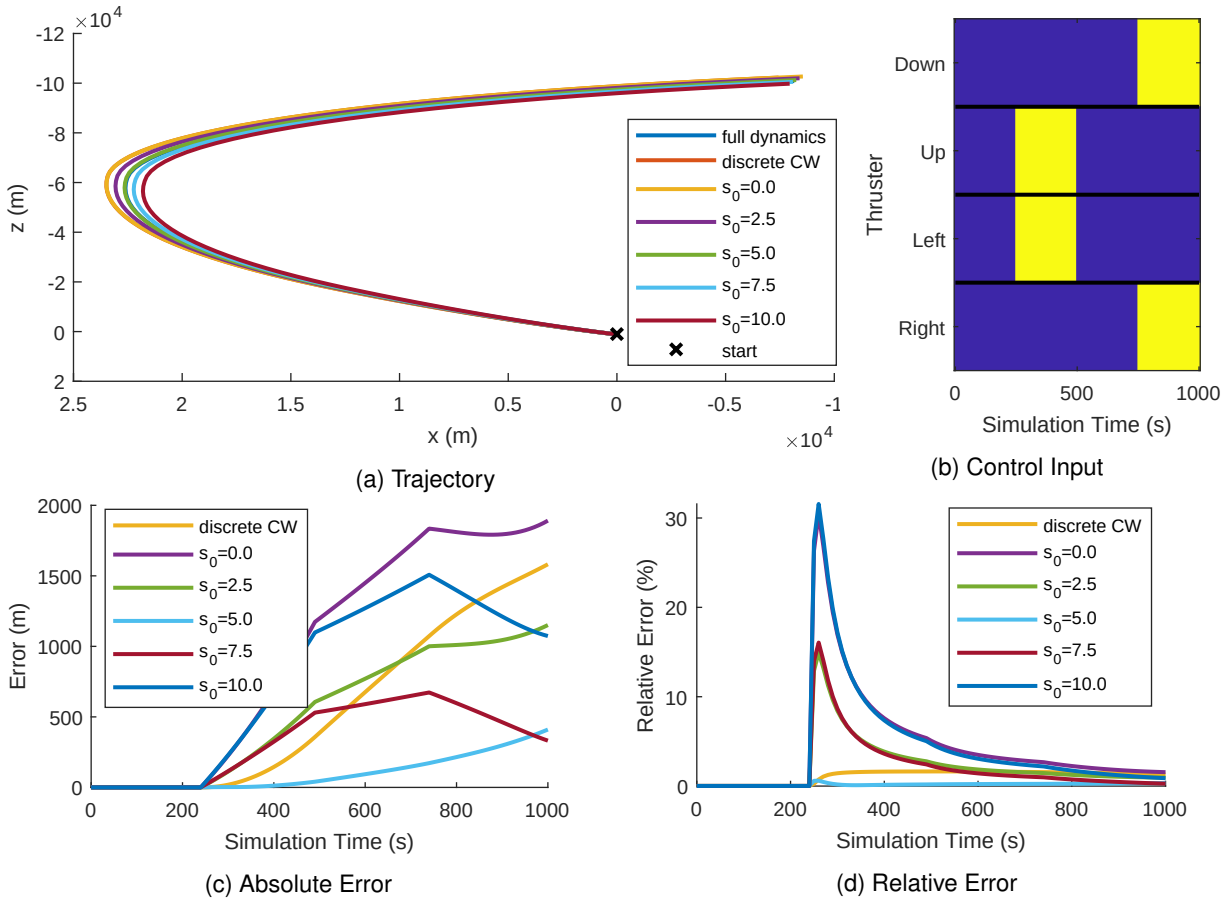
(c) Absolute Error

(d) Relative Error

Figure 3.8: Trajectory, control input (blue - inactive, yellow - active) and errors for case 4.

3.2.3 Experiment 3 - Linearization

This section serves to investigate the influence of the linearization point s_0 on the linearized CW model. In the first simulation, this model was tested using different values of s_0, and a control input such that each thruster is either inactive, or active during the full time step $T = 10\,\text{s}$. The results are shown in Figure 3.9.



(a) Trajectory

(b) Control Input

(c) Absolute Error

(d) Relative Error

Figure 3.9: Trajectory, control input (blue - $s = 0$, yellow - $s = T$) and errors for different values of s_0, expressed in seconds.

Despite similar looking trajectories, there is a clear difference on the results for different values of s_0. For the values of s_0 tested, error is clearly minimized for $s_0 = 5\,\text{s}$, which is the value used in the rest of the simulations. The further s_0 strays from $5\,\text{s}$, the further the trajectories appear in Figure 3.9a. The trajectories generated by the full dynamics, discrete CW and $s_0 = 5\,\text{s}$ overlap and are enclosed by the remaining. The reason for this may be related to the symmetry of the thrusters' directions and the control input used. For example, consider a case where there only exist two thrusters with the same thrust ($F_1 = F_2 = F$) pointing in opposite directions ($\mathbf{w}_1 = -\mathbf{w}_2$). In this case, we can write the discrete-time linearized dynamics as

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{A}(\bar{\mathbf{G}}(s_1, s_0)\mathbf{B_c}F\mathbf{w}_1 + \bar{\mathbf{G}}(s_2, s_0)\mathbf{B_c}F\mathbf{w}_2) \tag{3.3}$$

$$= \mathbf{A}\mathbf{x}_k + \mathbf{A}\frac{\partial \mathbf{G}}{\partial s}\bigg|_{s=s_0}(s_1 - s_2)\mathbf{B_c}F\mathbf{w}_1. \tag{3.4}$$

On the other hand, the exact discrete-time dynamics may be expressed as

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{A}(\mathbf{G}(s_1)\mathbf{B_c}F\mathbf{w}_1 + \mathbf{G}(s_2)\mathbf{B_c}F\mathbf{w}_2) \tag{3.5}$$

$$= \mathbf{A}\mathbf{x}_k + \mathbf{A}(\mathbf{G}(s_1) - \mathbf{G}(s_2))\mathbf{B_c}F\mathbf{w}_1. \tag{3.6}$$

On either case, when the opposing thrusters are on for the same amount of time ($s_1 = s_2$), the result is free motion ($\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$). When one actuator is on and the other one isn't ($s_1 = s$, $s_2 = 0$), for the exact discrete-time dynamics we get

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{A}(\mathbf{G}(s) - \mathbf{G}(0))\mathbf{B_c}F\mathbf{w}_1, \tag{3.7}$$

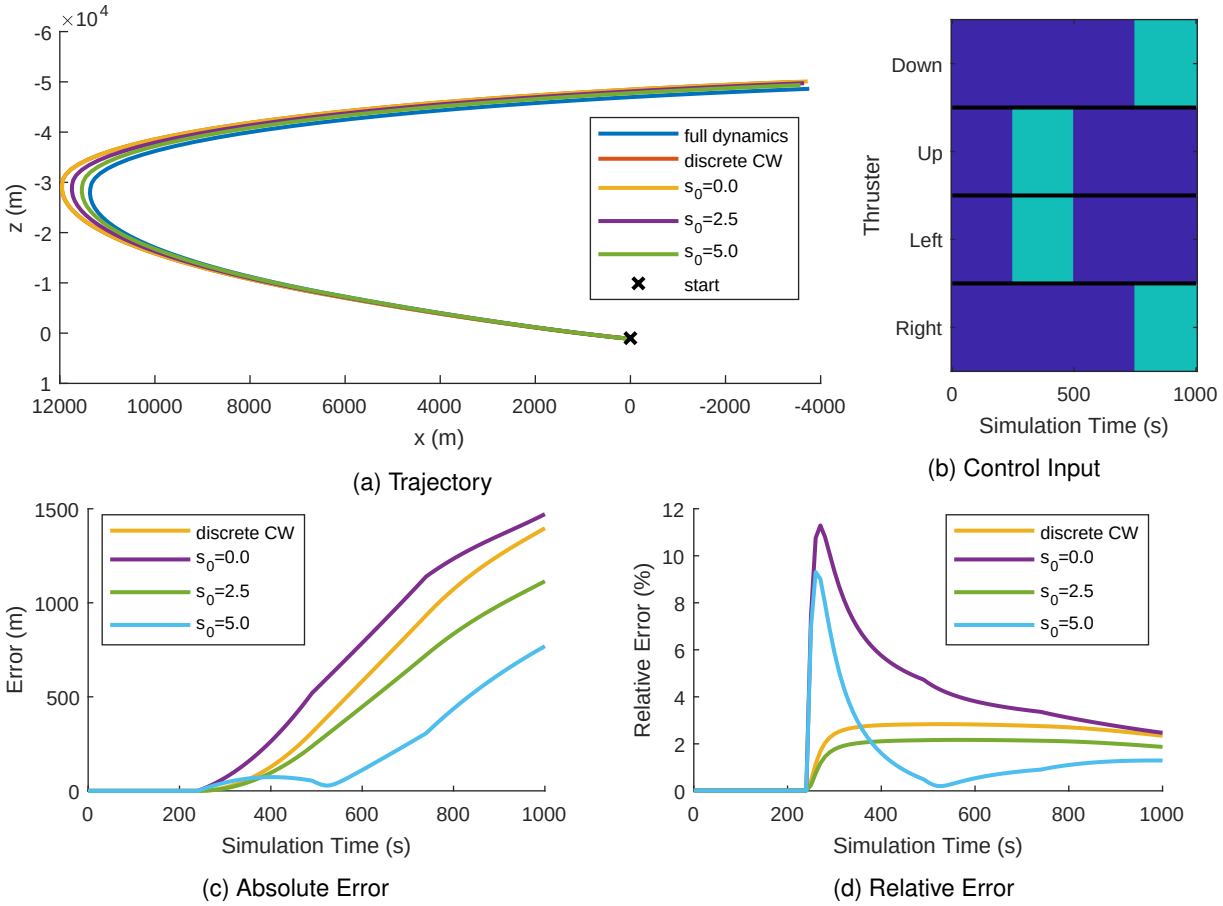
and the corresponding linearized dynamics are

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{A}\frac{\partial \mathbf{G}}{\partial s}\bigg|_{s=s_0}(s - 0)\mathbf{B_c}F\mathbf{w}_1. \tag{3.8}$$

Picking $s_0 = s/2$ places the linearization point in the middle of the interval $[0, s]$, which minimizes the furthest distance between s_0 and the extremes of the aforementioned interval. Moreover, we observe in Figure 3.9d that the relative error is minimized for $s_0 = 5\,\mathsf{s}$ (amongst the evaluated values for s_0), which is the middle of the interval $[0, 10\,\mathsf{s}]$. This suggests that for $s_1 = s$ and $s_2 = 0$, $s_0 = s/2$ is a good value of s_0 to use to get a good approximation of the discrete CW dynamics.

In order to solidify this hypothesis, a second simulation was performed, this time using a control input such that each thruster, when active, is active only for half of the time step ($s = T/2 = 5\,\mathsf{s}$). The results are shown in Figure 3.10. This time, the error is minimized for $s_0 = T/4 = 2.5\,\mathsf{s}$, and as expected, the trajectory generated by the full dynamics and discrete CW overlap for this value of s_0.

Of course, $\mathbf{G}(s)$ is highly non-linear, and these observations only apply because $\omega s \ll 2\pi$. In particular, throughout this thesis, $T = 10\,\mathsf{s}$ and $\omega = 1\,\mathsf{mrad\,s^{-1}}$, so $\omega s \leq \omega T = 0.01\,\mathsf{rad}$. These findings are merely suggestive, and a consequence of the control input chosen, used to aid in choosing a suitable value of s_0. We will see in the next section, where the MPC controller is employed, that many different control inputs are used, but $s = T$ seems to be the most common, which is the reason why $s_0 = T/2$ is used in the rest of this thesis.



(a) Trajectory

(b) Control Input

(c) Absolute Error

(d) Relative Error

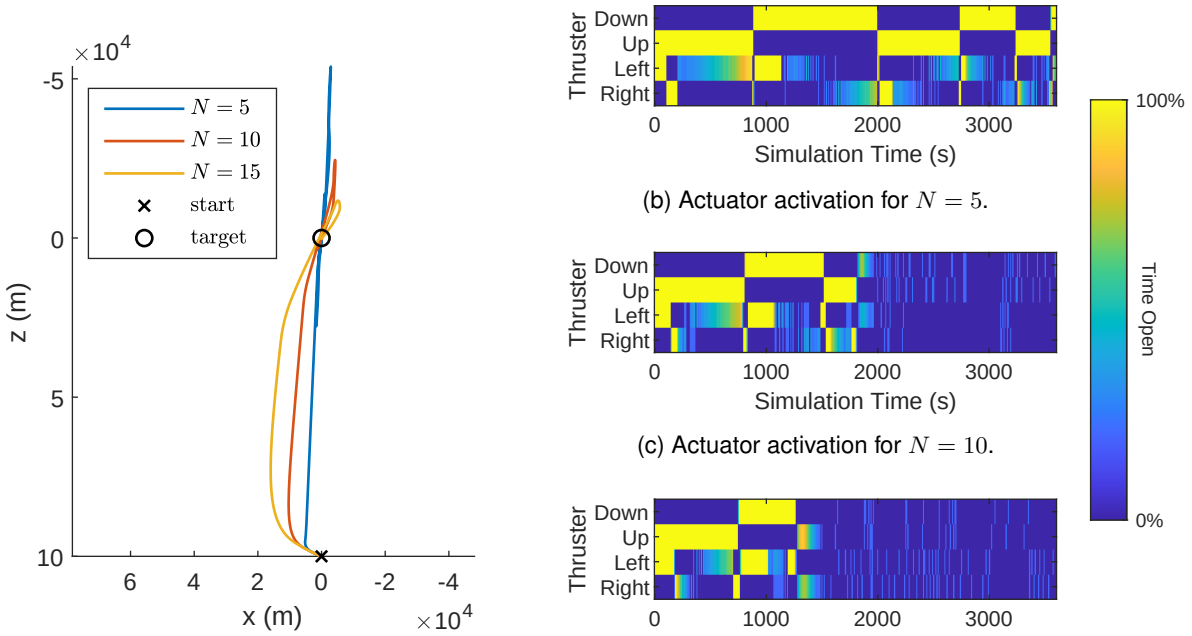
Figure 3.10: Trajectory, control input (blue - $s = 0$, light blue - $s = T/2$) and errors for different values of s_0, expressed in seconds.

3.3 Standard Algorithm

Firstly, a few trajectories using different simulation parameters are shown in Fig.3.11, all using the **Standard** algorithm, represented in the LVLH frame. Along with the trajectories, the control signal is also shown, making it possible to observe the manoeuvres performed by the satellite. The out-of-plane axis, y is not shown, nor are the thrusters which only apply force along this axis, as they never fire. The default parameter configuration is always shown, which can be seen as the red trajectory from Figure 3.11 to Figure 3.12 and the second row from Table 3.3 to 3.4.

3.3.1 Varying N

Along with the results from the three simulations shown in Figure 3.11, Table 3.3 contains the values of some quantitative metrics. We see that varying N has a significant impact on the manoeuvres performed by the satellite and consequently on each trajectory. A larger horizon seems to produce shorter, more efficient manoeuvres reaching the target sooner. However, increasing N comes with the cost of a nonlinear increase in computational time.



(a) Trajectories, varying N

(b) Actuator activation for $N = 5$.

(c) Actuator activation for $N = 10$.

(d) Actuator activation for $N = 15$.

Figure 3.11: Trajectories and actuator activations for the simulated rendezvous using **Standard** algorithm, varying N.

Table 3.3: Mission time and fuel spent for different values of N.

	Fuel Spent	Mission Time	Simulation Time
$N = 5$	5223.41 s	3540 s	5.06 s
$N = 10$	2931.49 s	1890 s	27.19 s
$N = 15$	2268.59 s	1380 s	141.87 s

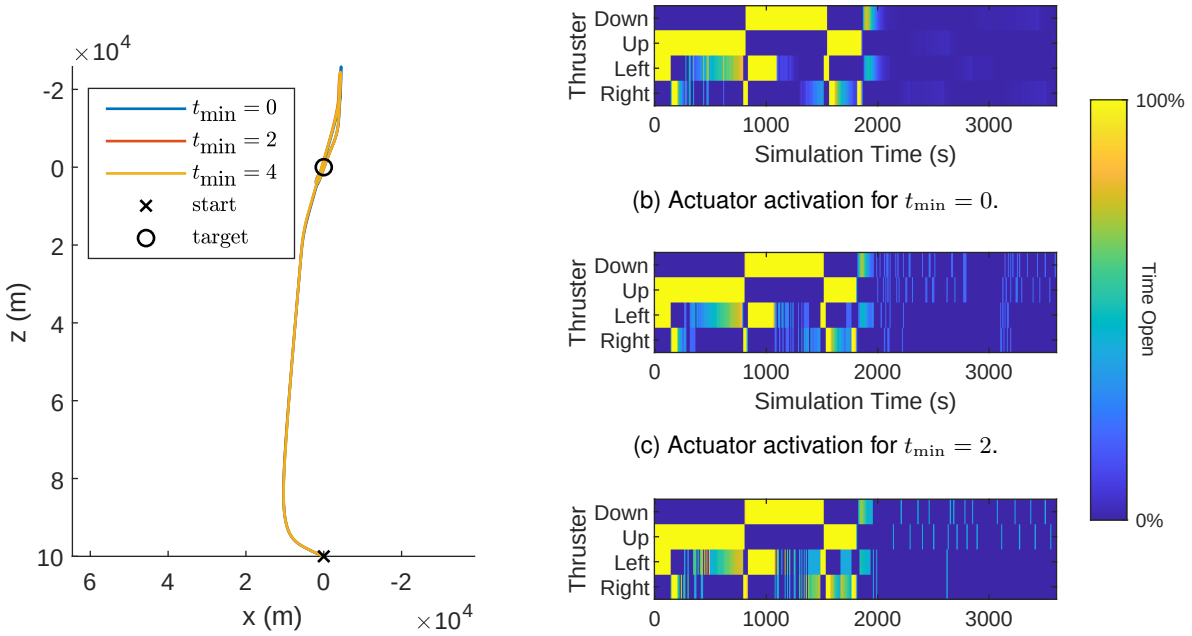
[1] Did Not Finish

3.3.2 Varying $t_{\min}$

As we can observe from Figure 3.12 and Table 3.4, changing the minimum time the actuators can be on doesn't seem to largely impact the overall performance of the controller - only small adjustments are affected by this constraint. The core of the manoeuvre requires some thrusters to be continuously on ($s = T$) for considerable amounts of time, and this is unaffected. Unlike all other positive values of $t_{\min} = 0$, for $t_{\min} = 0$, the algorithm is not solving a mixed integer problem, but a linear one, as there are no integer constraints. This is why the simulation time is so much lower than for the other values of $t_{\min}$. These results hint at the viability of using a linear solver for the rendezvous problem, which is what the **Relaxed** and **Projected** attempt to do.

3.3.3 Comparing the Cost Functions

In this section, both cost functions, J_1 and J_2 (described in Section 2.2.1), were used in rendezvous simulations. The results are shown in Figure 3.13 and Table 3.5. As a reminder, the only difference between them is that J_1 weighs the tracking error along the entire prediction horizon, while J_2 only weighs the tracking error at the end of the prediction horizon. In practice, the only advantage of using



(a) Trajectories, varying $t_{\min}$

(b) Actuator activation for $t_{\min} = 0$.

(c) Actuator activation for $t_{\min} = 2$.

(d) Actuator activation for $t_{\min} = 4$.

Figure 3.12: Trajectories and actuator activations for the simulated rendezvous using **Standard** algorithm, varying $t_{\min}$.

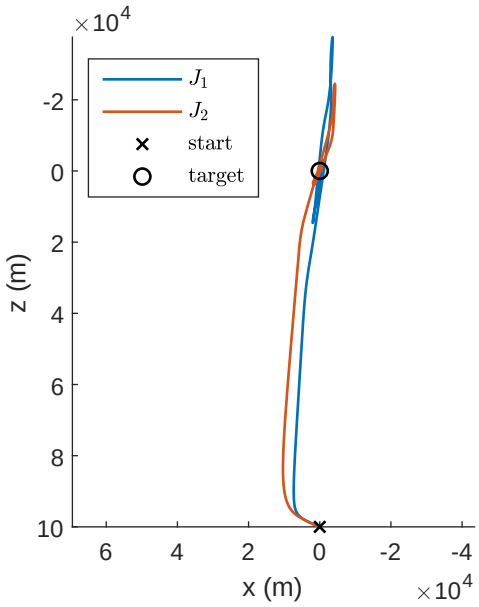
Table 3.4: Mission time and fuel spent for different values of $t_{\min}$.

	Fuel Spent	Mission Time	Simulation Time
$t_{\min} = 0$	3070.49 s	1930 s	4.50 s
$t_{\min} = 2$	2931.49 s	1890 s	27.19 s
$t_{\min} = 4$	3068.53 s	1890 s	21.62 s

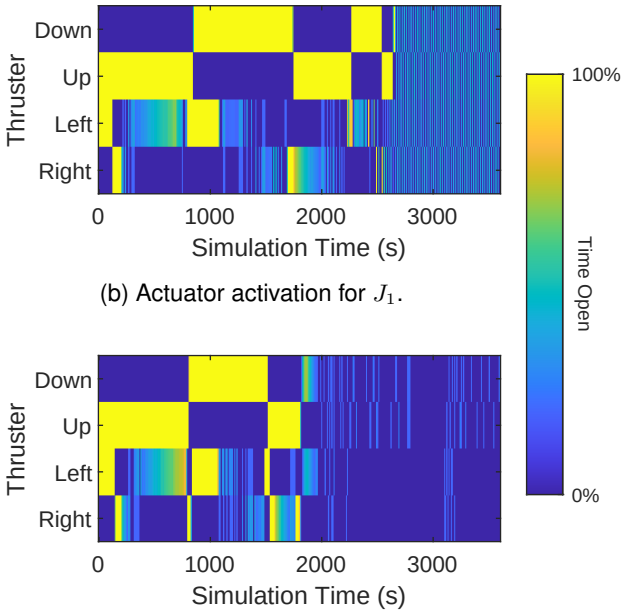
J_1 observable from this data is lower simulation times. Not only does the control effort increase but so does the mission time.

Table 3.5: Mission time and fuel spent for the two different cost functions.

	Fuel Spent	Mission Time	Simulation Time
J_1	3763.89 s	2550 s	11.63 s
J_2	2948.35 s	1860 s	25.55 s



(a) Trajectories, for the two different cost functions.



(b) Actuator activation for J_1.

(c) Actuator activation for J_2.

Figure 3.13: Trajectories and actuator activations for the simulated rendezvous using **Standard** algorithm, for each cost function.

3.4 Obstacle Avoidance

The obstacle avoidance algorithm described in Section 2.2.2 makes it possible for the chaser to avoid obstacles while still performing the rendezvous. The algorithm is tested in a simulated rendezvous with two obstacles, placed in the path of the chaser.

The two obstacles are described by

$$\mathbf{C}_1 = \begin{bmatrix} -10\,\text{km} & 70\,\text{km} \end{bmatrix}^T, \tag{3.9}$$

$$\mathbf{R}_1 = 20\,\text{km}, \tag{3.10}$$

and

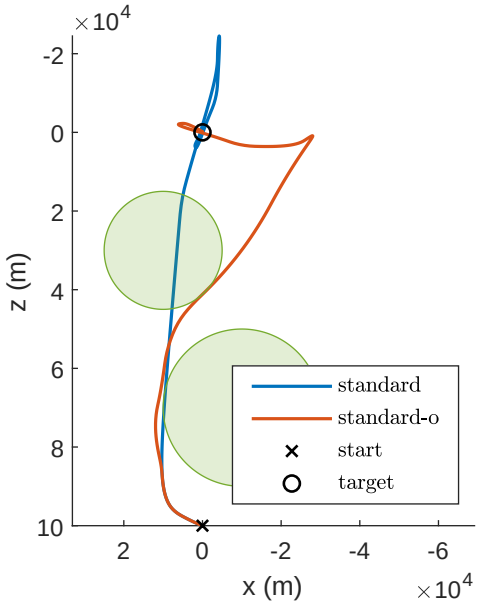
$$\mathbf{C}_2 = \begin{bmatrix} 10\,\text{km} & 30\,\text{km} \end{bmatrix}^T, \tag{3.11}$$

$$\mathbf{R}_2 = 15\,\text{km}, \tag{3.12}$$

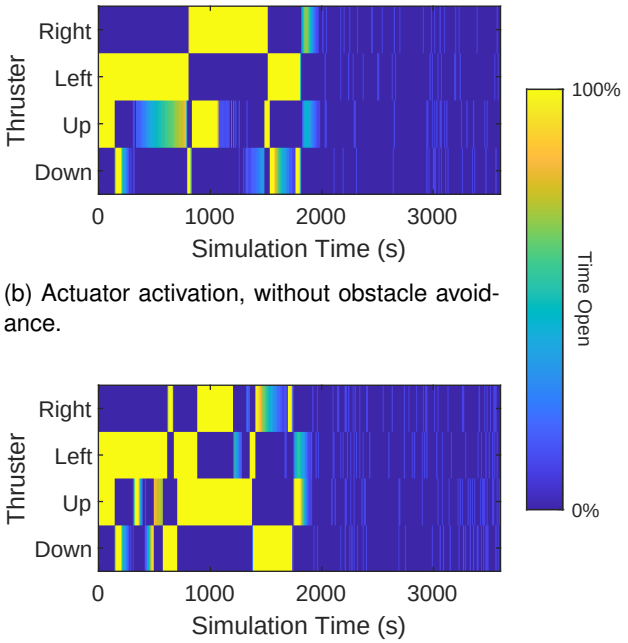
forming two constraints as described in 2.42. Algorithm 1 is then used to replace these nonconvex constraints with linear constraints.

Figure 3.14 shows the trajectories and actuator activations for the rendezvous with and without obstacle avoidance. We can see that without obstacle avoidance, the chaser's trajectory collides with the obstacles. However, with obstacle avoidance, the chaser is able to avoid the obstacles. Notably, the fuel spent for the rendezvous with obstacle avoidance is slightly less than when not using obstacle avoidance, even though the mission time slightly increases.

Figure 3.15 shows the trajectories and actuator activations for the rendezvous with obstacle avoid-



(a) Trajectories, **Standard** algorithm, with (standard-o) and without (standard) obstacle avoidance and obstacles (green).

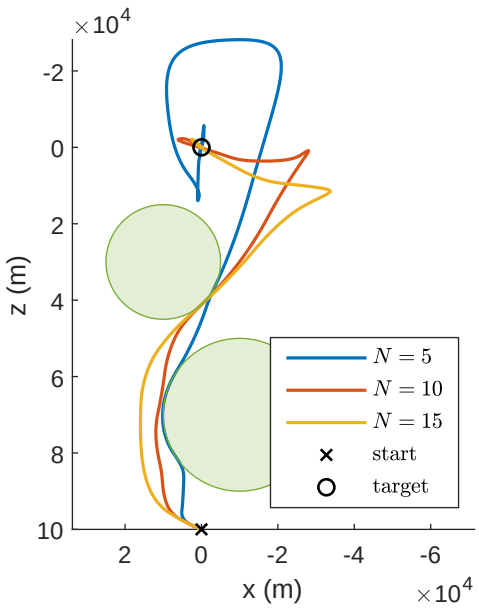


(b) Actuator activation, without obstacle avoidance.

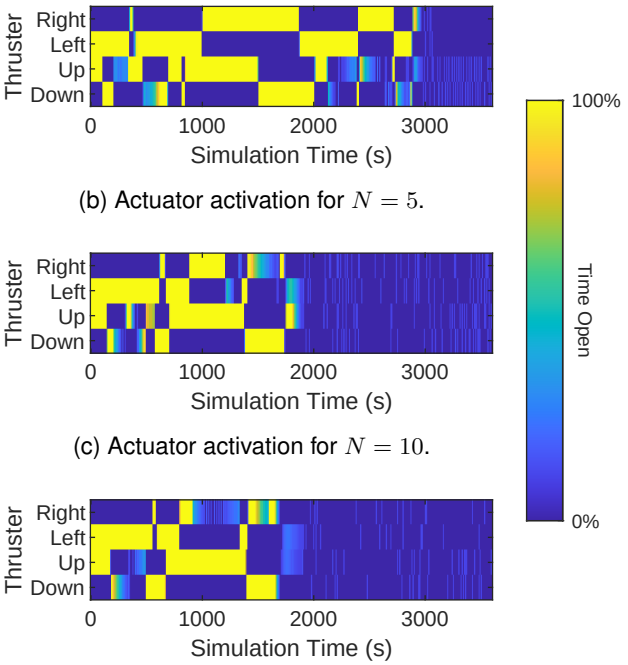
(c) Actuator activation, with obstacle avoidance.

Figure 3.14: Trajectories and actuator activations for the simulated rendezvous using **Standard** algorithm, with and without obstacle avoidance.

ance, varying N. The lower the prediction horizon N, the more fuel is spent, and the closer the chaser gets to the obstacles, due to having less time to react to them.



(a) Trajectories, **Standard** algorithm with obstacle avoidance, varying N, and obstacles (green).



(b) Actuator activation for $N = 5$.

(c) Actuator activation for $N = 10$.

(d) Actuator activation for $N = 15$.

Figure 3.15: Trajectories and actuator activations for the simulated rendezvous using **Standard** algorithm, varying N.

3.5 Algorithm Performance

This section shows the fundamental results this thesis aims to capture, presenting the performance of the algorithms described in Section 2.4 in the rendezvous scenario along different metrics. The mixed integer solution implemented as the **Standard** algorithm is used as the baseline for comparison, as it is optimal for the simplified problem 2.45.

3.5.1 Solution Quality

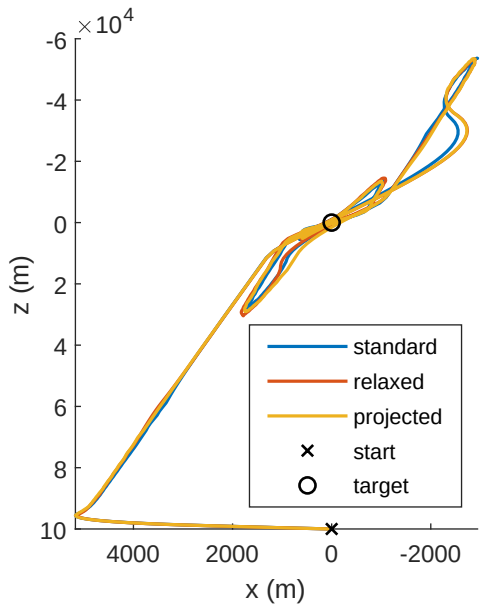
The three algorithms, **Standard**, **Relaxed** and **Projected**, were used in the rendezvous scenario with different values of N. The results are presented in Figures 3.16, 3.17 and 3.18, for $N = 5$, $N = 10$ and $N = 15$, respectively. We can see that, irrespective of the value of N, the trajectories generated by each of the algorithms are very similar. It is not obvious that this should be the case, as the **Standard** algorithm is optimal for the simplified problem, and it is also not clear how close to optimality the other two algorithms are.

These results seem to indicate that there is not much to gain from exploring the mixed integer solution space in these types of scenarios, as the **Relaxed** and **Projected** algorithms are able to generate trajectories that are very similar to the optimal solution. This is further confirmed by the results presented in Table 3.6, which shows the fuel spent and mission time for each of the algorithms, for different values of N, also being very similar for all three algorithms. Furthermore, the **Projected** algorithm, which does explore some small part of the mixed integer solution space, does not seem to be able to generate trajectories that are significantly better than the **Relaxed** algorithm, which does not explore the mixed integer solution space at all.

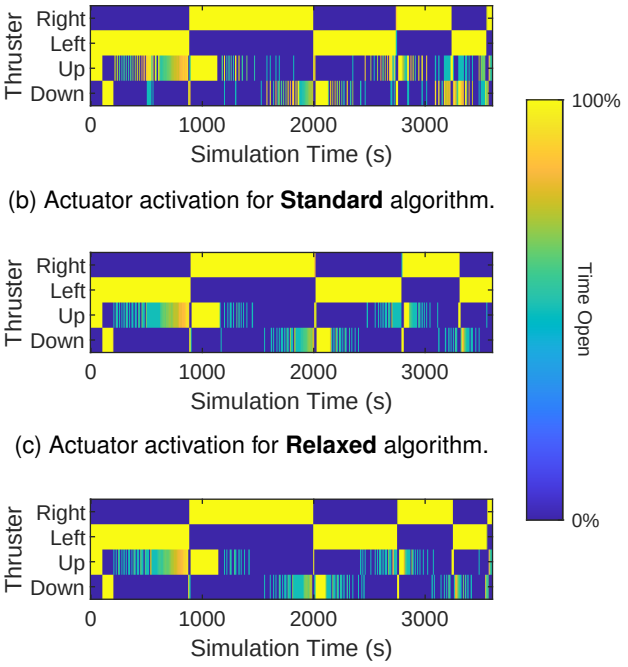
In Table 3.6, the worst-case exponential complexity of the **Standard** algorithm is hinted at, as the simulation time increases significantly with N. The **Relaxed** and **Projected** algorithms, on the other hand, do not suffer from this exponential complexity, and their simulation times are much lower. Even still, the solutions generated by these algorithms are very similar to the optimal solution, as seen in the actuator activations (and consequently in the trajectories themselves). The staggering difference in simulation time between the **Standard** algorithm and the other two algorithms makes it inviable not only to use the **Standard** algorithm in real-time, but also even in a simulation environment for higher prediction horizons, unlike the **Relaxed** and **Projected** algorithms.

3.5.2 Computation Time

For a deeper analysis of the computational cost of each of the algorithms, 100 simulations were performed per algorithm per value of N. Table 3.7 shows the mean, $95\,\%$ percentile and $99\,\%$ percentile of the computation time for each of the algorithms, for different values of N. We can see that on average, the **Standard** algorithm takes significantly longer to solve the optimization problem than the other two algorithms, accross all values of N. The **Relaxed** and **Projected** algorithms, on the other hand, differ by less than an order of magnitude in terms of computation time, with the **Projected** algorithm taking slightly longer than the **Relaxed** algorithm. This is expected, as the **Projected** solves several



(a) Trajectories generated by the chaser using each of the algorithms.

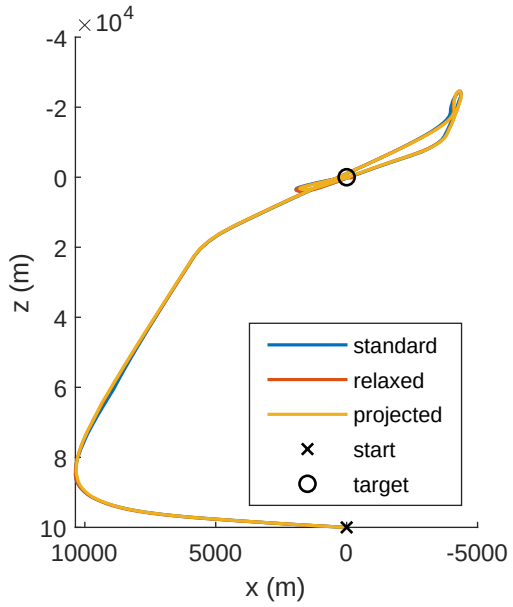


(b) Actuator activation for **Standard** algorithm.

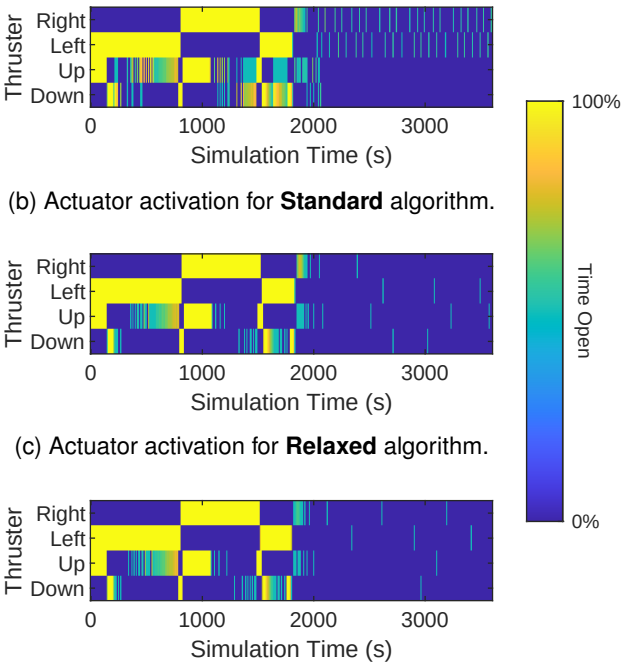
(c) Actuator activation for **Relaxed** algorithm.

(d) Actuator activation for **Projected** algorithm.

Figure 3.16: Trajectories and actuator activations for the simulated rendezvous using each of the algorithms, for $N = 5$.



(a) Trajectories generated by the chaser using each of the algorithms.



(b) Actuator activation for **Standard** algorithm.

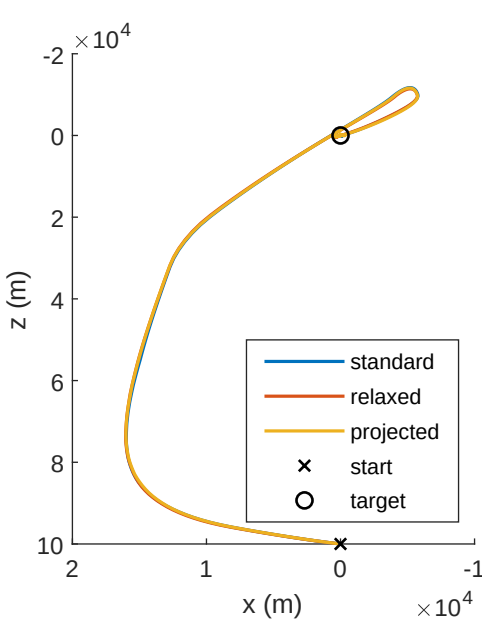
(c) Actuator activation for **Relaxed** algorithm.

(d) Actuator activation for **Projected** algorithm.

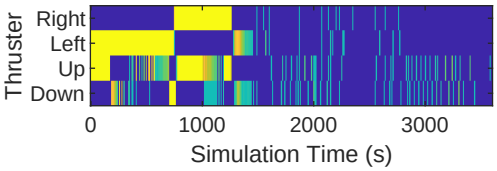
Figure 3.17: Trajectories and actuator activations for the simulated rendezvous using each of the algorithms, for $N = 10$.

optimization problems, one for each iteration of the algorithm, while the **Relaxed** algorithm only solves one optimization problem.

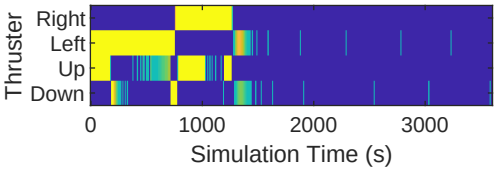
For a fuller picture regarding the solve times for each algorithm, for each prediction horizon N, the solve times for each of the 100 simulations were plotted in a histogram, shown in Figure 3.19. We



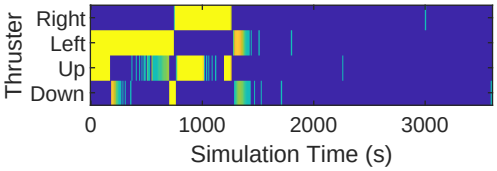
(a) Trajectories generated by the chaser using each of the algorithms.



(b) Actuator activation for **Standard** algorithm.



(c) Actuator activation for **Relaxed** algorithm.



(d) Actuator activation for **Projected** algorithm.

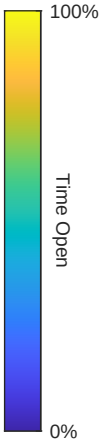


Figure 3.18: Trajectories and actuator activations for the simulated rendezvous using each of the algorithms, for $N = 15$.

Table 3.6: Mission time and fuel spent for the different algorithms, for different values of N.

	method	Fuel Spent	Mission Time	Simulation Time
$N = 5$	**Standard**	5633.04 s	DNF[1]	4.45 s
	Relaxed	5256.60 s	DNF[1]	1.88 s
	Projected	5187.47 s	DNF[1]	3.27 s
$N = 10$	**Standard**	3269.26 s	1860 s	18.28 s
	Relaxed	2887.34 s	1880 s	2.79 s
	Projected	2856.11 s	1860 s	5.17 s
$N = 15$	**Standard**	2405.63 s	1390 s	116.89 s
	Relaxed	2231.39 s	1390 s	3.21 s
	Projected	2203.84 s	1390 s	4.76 s

[1] Did Not Finish

can see that the **Standard** algorithm has a much wider distribution of solve times than the other two algorithms. Not only is it wider, but it is also shifted to the right, meaning that the **Standard** algorithm takes longer to solve the optimization problem than the other two algorithms, and the larger the value of N, the longer it takes. The **Relaxed** and **Projected** algorithms, on the other hand, have very similar distributions, with that of the **Projected** algorithm being slightly shifted to the right.

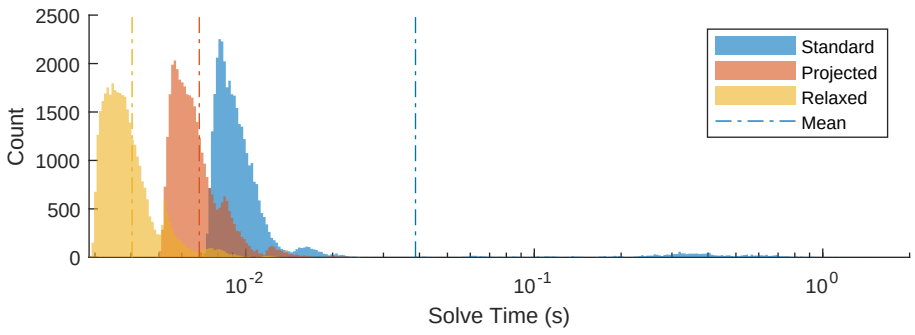
Distributions in Figure 3.19 seem to be bimodal, a characteristic that becomes more pronounced as N increases, even moreso for the **Standard** algorithm. The reason for this becomes clearer when we look at the solve times for each time step of the simulation, for each algorithm, for different values of N, shown in Figure 3.20. We can see that the solve times for each algorithm are not constant throughout the simulation, but rather vary significantly, especially for the **Standard** algorithm. Interestingly, the closer the chaser is to the target (as the simulation progresses), the higher the solve time for the **Standard**

Table 3.7: MPC solve time for the different algorithms (total time taken to generate one control signal), for different values of N, over 100 simulations.

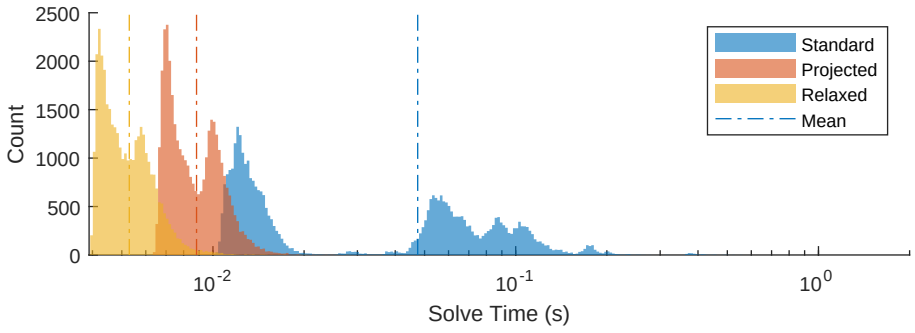
	method	Mean	$95\,\%$ Percentile	$99\,\%$ Percentile
$N = 5$	**Standard**	37.3 ms	109 ms	610 ms
	Relaxed	6.80 ms	9.67 ms	13.8 ms
	Projected	3.98 ms	5.86 ms	8.40 ms
$N = 10$	**Standard**	47.0 ms	115 ms	181 ms
	Relaxed	8.74 ms	11.9 ms	14.1 ms
	Projected	5.24 ms	7.11 ms	8.84 ms
$N = 15$	**Standard**	345 ms	953 ms	3.04 s
	Relaxed	10.7 ms	15.9 ms	22.1 ms
	Projected	6.50 ms	9.25 ms	12.4 ms

algorithm, while the solve times for the other two algorithms seem to vary less throughout the simulation and actually tend to decrease as the simulation progresses.

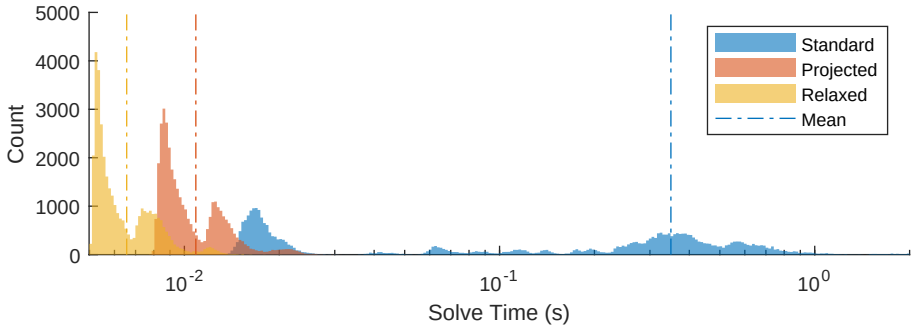
Despite taking longer to solve the optimization problem, the **Standard** algorithm also produces control signals requiring more control effort, as can be seen in Figure 3.21, showing the cumulative control effort for each algorithm, for different values of N. It is evident there is a large tradeoff between the marginally better tracking offered by the **Standard** algorithm and the higher control effort and computational cost. Figure 3.21 also highlights two distinct phases of the simulation: the first phase, where the chaser is far from the target, the control effort is higher, and a second phase, where the chaser is closer to the target, the control effort is lower.



(a) $N = 5$

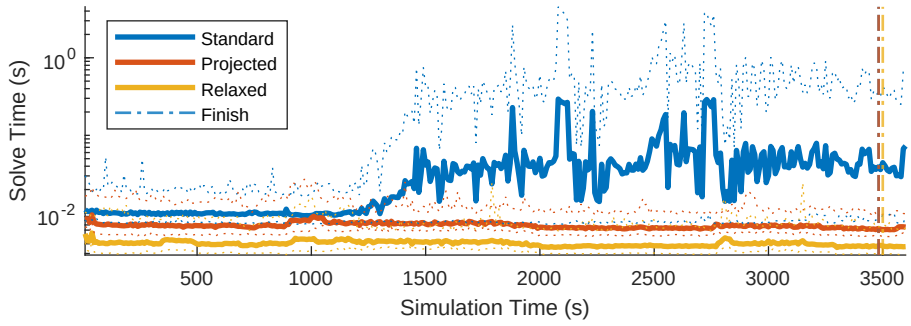


(b) $N = 10$

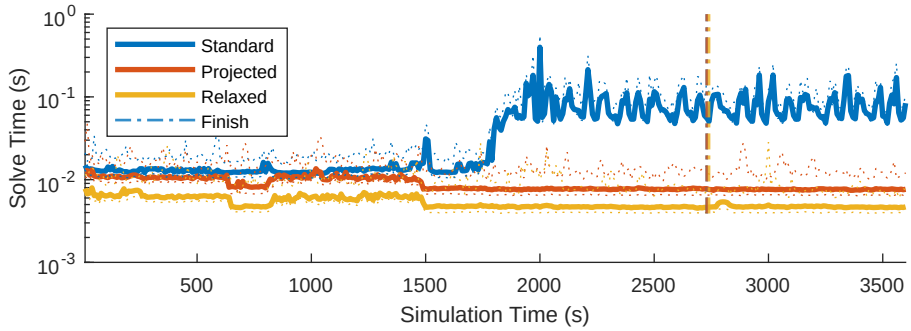


(c) $N = 15$

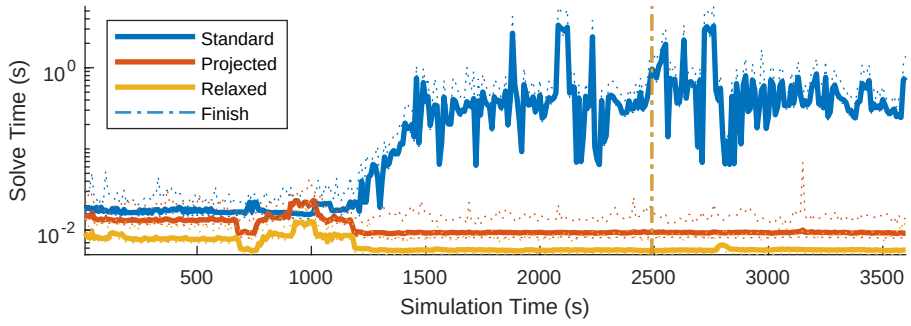
Figure 3.19: Histograms of the MPC solve time for the different algorithms, for different values of N, over 100 simulations.



(a) $N = 5$

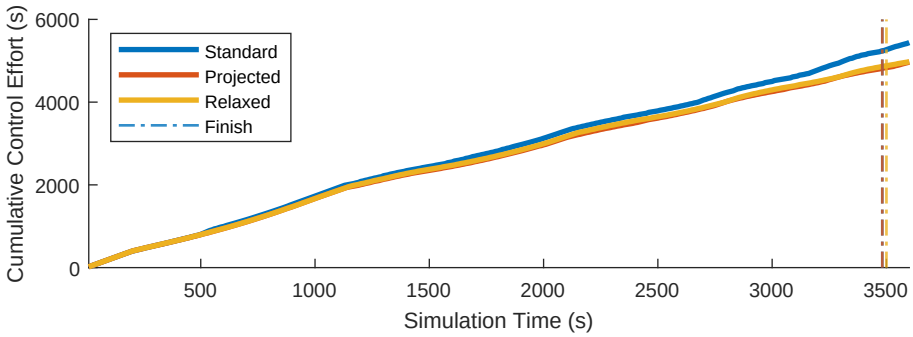


(b) $N = 10$

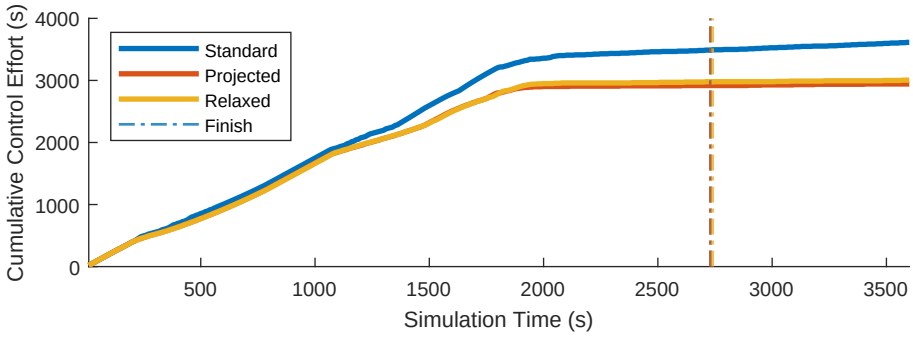


(c) $N = 15$

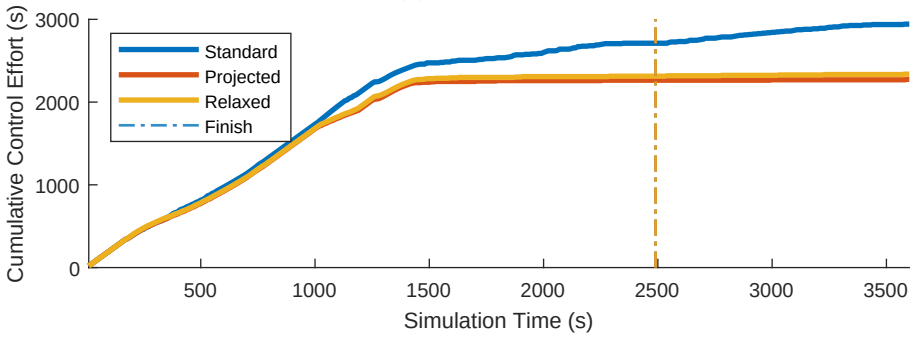
Figure 3.20: Mean of MPC solve times for each time step (solid line) and bounds (dotted lines), for the different algorithms, for different values of N, over 100 simulations. The vertical lines indicate the moment the mission is over.



(a) $N = 5$



(b) $N = 10$



(c) $N = 15$

Figure 3.21: Accumulated control effort, for the different algorithms, for different values of N.

Chapter 4

Conclusion

4.1 Summary

This dissertation presented a rendezvous scenario between a chaser and a target satellite, where the chaser, equipped with a set of thrusters, is tasked with approaching the target. The chaser's motion is described by Newton's Second Law of Motion and Newton's Law of Universal Gravitation. A set of equations known as the Clohessy-Wiltshire equations was used to linearize the chaser's motion relative to the target, which is assumed to be in a circular orbit. We modelled the chaser's thrusters as directional pulses with a minimum duration and discretized the resulting system in order to build an MPC controller capable of generating firing durations for the thrusters that drive the chaser to the target. Taking advantage of the flexibility of MPC, we were able to extend the rendezvous problem to include obstacles in the chaser's path. These obstacles were modelled as circular regions in the orbit plane, and an algorithm was implemented to substitute the nonlinear constraints they impose with linear constraints.

A simulator was built to test the MPC controller, using the full nonlinear model of the chaser's motion. In order to solve the optimization problem posed by the MPC controller, three algorithms were implemented: a mixed integer solution (optimal, but worst case exponential computational complexity on the number of integer constraints), and two other algorithms that approximate the solution to the optimization problem (not optimal, but linear complexity) - one that simply relaxes the integer constraints, and another that iteratively projects the solution of the relaxed problem onto the feasible solution space.

Before testing the MPC controller in the rendezvous scenario, the linear model used by the MPC controller was validated against the full nonlinear model both in the absence and in the presence of actuation. Then, the trajectories and actuator activations generated by the MPC controller using the mixed integer solution were analyzed for different parameters such as the prediction horizon. The rendezvous problem with obstacle constraints was then demonstrated.

The performance of these algorithms was compared in terms of solution quality and computation time. The results showed that the two algorithms that approximate the solution to the optimization problem are able to generate trajectories that are very similar to the optimal solution while taking much less time to do so. This makes them much more suitable for real-time applications, where the MPC controller is used to generate firing durations for the thrusters in real time.

4.2 Future Work

The work presented in this dissertation allows for a number of possible extensions and improvements.

1. **Theoretical Error Bounds:** A significant portion of this dissertation is the implementation and analysis of two algorithms that approximate the solution to the optimization problem posed by the MPC controller. However, the theoretical error bounds of these algorithms are not known. It would be relevant to explore these bounds, in order to understand how far from the optimal solution they can be. This would allow for a better understanding of how much of the mixed integer solution space needs to be explored in order to get a solution that is close enough to the optimal solution, according to some metric.

2. **Non-Circular Target Orbit:** The rendezvous scenario presented in this dissertation assumes that the target satellite is in a circular orbit. An opportunity for future work would be to extend the scenario to include a target satellite that is not in a circular orbit, but in an elliptical orbit, or even a target satellite that is not in a Keplerian orbit, such as an active satellite that is able to manoeuvre.

3. **Attitude Control and Docking:** The chaser is assumed to be a point mass, and its motion is described by its position and velocity. The chaser could be extended to include attitude control, enabling it to rotate and point its thrusters in different directions. This would allow for more complex maneuvers, and eventually allow the chaser to dock with the target.

4. **Obstacle Detection:** The obstacle avoidance algorithm presented in this dissertation assumes that the obstacles are known beforehand and are static. Detecting obstacles in real time and avoiding them would be a valuable addition to the obstacle avoidance algorithm.

Bibliography

[1] W. Hohmann, "The attainability of celestial bodies," *NASA Technical Translation F*, vol. 44, 1925.

[2] S. Matsumoto, S. Dubowsky, S. Jacobsen, and Y. Ohkami, "Fly-by approach and guidance for uncontrolled rotating satellite capture," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 5745, 2003.

[3] C. D. Brown, *Spacecraft propulsion*, p. 59. AIAA, 1996.

[4] D. Arney, R. Sutherland, J. Mulvaney, D. Steinkoenig, C. Stockdale, and M. Farley, "On-orbit Servicing, Assembly, and Manufacturing (OSAM) State of Play, 2021 Edition," tech. rep., Oct. 2021.

[5] J. Branco, P. Colmenarejo, P. Serra, P. Lourenço, T. V. Peters, and M. Mammarella, "Critical GNC Aspects for ADR missions," in *Proceedings of the 11th International ESA Conference on Guidance, Navigation & Control Systems*, (Sopot, Poland (Virtual)), ESA, June 2021.

[6] D. Malyuta, M. Szmuk, and B. Acikmese, "Lossless convexification of non-convex optimal control problems with disjoint semi-continuous inputs," *arXiv e-prints*, p. arXiv:1902.02726, Feb. 2019.

[7] D. Malyuta and B. Açikmeşe, "Lossless convexification of optimal control problems with semi-continuous inputs," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6843–6850, 2020. 21st IFAC World Congress.

[8] J. L. Goodman, "History of space shuttle rendezvous and proximity operations," *Journal of Spacecraft and Rockets*, vol. 43, no. 5, pp. 944–959, 2006.

[9] D. E. Hastings and C. Joppin, "On-orbit upgrade and repair: The hubble space telescope example," *Journal of spacecraft and rockets*, vol. 43, no. 3, pp. 614–625, 2006.

[10] W. Fehse, *Automated Rendezvous and Docking of Spacecraft*. Cambridge Aerospace Series, Cambridge University Press, 2003.

[11] J. Briz, N. Paulino, P. Lourenço, P. Cachim, E. Forgues-Mayet, L. Ferreira, A. Groth, E. Papadopoulos, G. Rekleitis, K. Nanos, and V. Preda, "Guidance, Navigation, and Control of In-Orbit Assembly of Large Antennas – technologies and approach for IOANT," in *Proceedings of the 41st ESA Antenna Workshop on Large Deployable Antennas*, (Noordwijk, The Netherlands), ESA, Sept. 2023.

[12] P. Hespanhol, R. Quirynen, and S. Di Cairano, "A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control," in *2019 18th European Control Conference (ECC)*, pp. 2763–2768, 2019.

[13] R. E. Davis, D. A. Kendrick, and M. Weitzman, "A branch-and-bound algorithm for zero-one mixed integer programming problems," *Operations Research*, vol. 19, no. 4, pp. 1036–1044, 1971.

[14] Cplex, IBM ILOG, "V12. 1: User's manual for cplex," 2009.

[15] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022.

[16] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1353–1366, Sept. 2007.

[17] B. Acikmese, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 2104–2113, Nov. 2013.

[18] W. Clohessy and R. Wiltshire, "Terminal guidance system for satellite rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, 1960.

[19] D. Krejci and P. Lozano, "Space propulsion technology for small spacecraft," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 362–378, 2018.

[20] A. Botelho, B. Parreira, P. N. Rosa, and J. M. Lemos, *Predictive Control for Spacecraft Rendezvous*, pp. 72–73. Springer, 2021.

[21] D. Silvestre and G. Ramos, "Model predictive control with collision avoidance for unknown environment," *IEEE Control Systems Letters*, vol. 7, pp. 2821–2826, 2023.

[22] MATLAB, *9.11.0 (R2021b)*. Natick, Massachusetts: The MathWorks Inc., 2021.