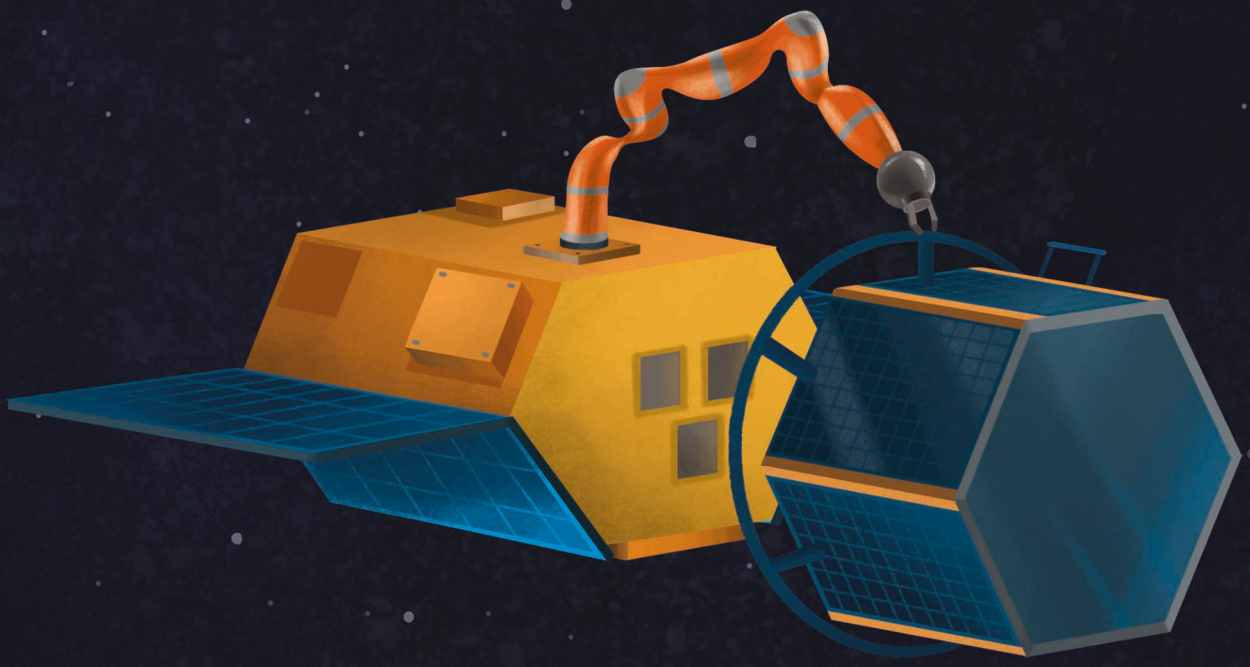
Combined Control of the Servicing Spacecraft with Robotic Manipulator

Model Predictive Control architecture

Paulina Maria Swiatek

Delft University of Technology



TU Delft

Combined Control of the Servicing Spacecraft and Robotic Manipulator with Model Predictive Control Architecture

MSc Thesis

by

Paulina Maria Swiatek

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 6, 2022 at 1:30 PM.

Student number: 4840445

Thesis committee:	Dr. ir. J. Guo,	TU Delft, Supervisor/Chair
	Ir. R. Krenn,	DLR, Daily Supervisor
	Dr. ir. E. Mooij,	TU Delft, Examiner
	Dr. A. Menicucci,	TU Delft, Examiner

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Front image designed by Sacha Berna

TU Delft

DLR

Abstract

The growing satellite congestion in the Earth orbits increases the risk of Kessler Syndrome, that could potentially hinder humanity's activities in space. One of the ways to tackle the problem is Active Debris Removal (ADR) and On-Orbit Servicing (OOS) missions with the capture phase of another spacecraft performed using a satellite equipped with robotic manipulator. Robotics solutions are good candidates for the application in on-orbit servicing and active debris removal missions. Due to the heritage in previous missions in which the robotic arms were used mainly operated by astronauts or in a semi-autonomous mode, and also given the possible technology transfer from terrestrial robotics autonomous systems, the space robotics are foreseen to have high potential in space for applications in close proximity operations between two spacecraft. The GNC system for such a mission is one of the main challenges due to the strict safety requirements and precision, moreover the complexity increases with any uncertainty of the target spacecraft state. Agility and precision in coupled position and attitude control are critical to ensure an operation free of collisions between the servicer spacecraft and the target. The challenge lies mainly in the close rendezvous, reach and capture phase, as all the mission phases prior to close rendezvous phase function are similar to single satellite mission without robotic subsystem.

This research project aims to propose a control system solution of a robotic spacecraft specifically for the reach phase of the OOS space mission, when the close rendezvous phase is successfully finished and the spacecraft starts the reach maneuver towards the target. Majority of proposed solutions for control system concerns free-floating mode in which the controller of the s/c base is turned off and only the manipulator's controller is active. This project aimed to investigate the combined controller approach for OOS mission reach phase in which both the base and robotic manipulator are actively controlled by a single control system that coordinates all sensor data and the actuation. The project boundaries were set such that the initial condition is assumed that the chaser spacecraft is few meters away from the target and the initial relative position and velocity with respect to the target is kept constant, the end of the reach phase is when the end-effector reaches the grasping point. In order to focus uniquely on control system design certain simplifying assumptions were taken with respect to the guidance and navigation capabilities. Moreover, the capture maneuver itself is out of scope of this work, for this reason contact dynamics were not included in the model.

The multi-body dynamics were defined with the SpaceDyn toolbox in Matlab and the advanced control strategy was chosen, Model Predictive Control (MPC). The objective of the project was to investigate the application of MPC for the design of the combined controller. For this purpose firstly a variety of MPC architectures was studied in order to preliminary choose the best candidate for the robotic s/c system. The final decision was also supported by a trade-off study. The advantage of MPC is the optimization-based strategy, a straightforward definition of the constraints and the dynamics prediction within the future horizon. A nonlinear model predictive control (NMPC) based on successive linearization approach was developed, the linear dynamics prediction model is obtained in every simulation step by linearizing the nonlinear dynamics around the current operating point. The optimization problem was constructed as quadratic problem with the primary goal of end-effector position reference tracking and other secondary goals and it was modelled with Yalmip toolbox interface. Finally, the NMPC controller and the plant model were put together in a feedback loop and tested for different scenario cases.

The final results show that the reach maneuver can be successfully accomplished as long as the weights tuning procedure is performed carefully. The final position error of end-effector is very small and remains in the acceptable performance limits. The constraints imposed by the definition of the MPC problem are well respected, and the optimization problem is converging in every iteration. The main improvement point of the designed control system is that it requires to be re-tuned for changing initial conditions, the s/c base position and attitude and manipulator configuration. Further extensions of the project would be very interesting, including complete integration with a proper guidance and navigation capabilities, adding low-level control level and extending the dynamics model such that it accounts for the contact dynamics and capture maneuver. All in all, the project results are a good starting point for the future development of the combined controlled strategies for OOS missions. This study was performed in collaboration with the German Aerospace Center DLR within RICADOS (Rendevous, Inspection, Capture and Detumbling by Orbital Servicing) project.

Preface

The way we function nowadays is highly dependent on satellites technology providing solutions in practically all day-to-day activities. Only imagining the potential scenario in which the utilization of the Earth orbits is limited and results in many disruptions in our daily lives is very frightening. There are many problems identified mainly being the increasing orbital congestion of space debris of different origin and the traditional design of spacecraft leading to lack of modularity and servicing capabilities. The already ongoing shift in the space industry will set up the frame in which the on orbit servicing will hopefully become the new normality, enabling the life extension services and hence allowing for the sustainable usage of space assets. The close proximity operations intrinsic to both the active debris removal and servicing missions, demand high level of autonomy and robustness which can be provided with the robotic manipulator solution, further addressed in this work. With a belief that developing the technology solutions relevant for enabling the sustainability of the Earth orbital environment, the work on this subject was very motivating and fulfilling, despite many challenges due to the global pandemic which had impact on my work organization. I feel privileged have been able to work on the project which is highly aligned with my personal interests and possibly contribute to technology development relevant for this area.

I am very grateful to my supervisors for their support towards the execution of my work. To my daily supervisor Rainer Krenn, for your time, valuable advices and e-being together in this research journey. You were one of the very few people who really understood what is going on in my code and provided me with a lot of technical advices. To prof. Jian Guo for all the critical feedback during our review meetings and making me feel part of TUDelft community despite the long-distance presence. It would be very difficult to find inner strength necessary for finalization of the work without help of my family, dearest friends and loved ones. I would like especially thank my Mum for tons of fresh fruits, dark chocolate, coffee and hug doses during the pandemic period which made me stay at home and execute my work from there. To all my friends from graduate studies, high school times and many other important periods of my life. For sharing together this learning experience in the Netherlands, observing the world changing and relieving pandemic related, war related and of course thesis related stress. For putting up with my moods, ideas and listening when support was really needed. For tons of beers, jokes and being the source of happiness in the past years and hopefully much more to come!

Last but not least and rather most importantly - big thank you to my grandparents for your wisdom, life lessons and eternal love that you always shared with me which made me brave enough to sign up for the challenges of the last years. Especially to my grandmother who clearly got bored of waiting for me to finish this work and decided not to wait anymore. For your smile, empathy and always showing me how to be strong no matter the circumstances. Without you I would never make it, dziękuję!

Paulina Maria Świątek
June 2022

Contents

List of Figures

List of Tables

List of Abbreviations

ADR	Active Debris Removal
AOCS	Attitude Orbit Control System
AOGNC	Attitude, Orbit, Guidance, Navigation, Control
CAM	Collision Avoidance Manoeuvre
CDMU	Central Data Management Unit
COM	Centre of Mass
CPO	Close Proximity Operations
CSA	Canadian Space Agency
DARE	Discrete Algebraic Riccati Equation
DEOS	Deutsche Orbitale Servicing (Mission)
DOF	Degree of Freedom
EE	End-effector
EOL	End-of-Life
EPOS	European Proximity Operations Simulator
ESA	European Space Agency
FDIR	Fault, Detection, Isolation and Recovery
FTS	Force-Torque Sensor
GEO	Geostationary Earth Orbit
GNC	Guidance, Navigation, Control
GSOC	German Space Operational Center
HDL	Hardware Description Language
HPTM	High Power Transmission Mode (Data)
ISS	International Space Station
KKT	Karush-Kuhn-Tucker
LEO	Low Earth Orbit
LEOP	Launch and Early Orbit Phase
LTI	Linear Time-Invariant
LTV	Linear Time-Varying
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
NMPC	Non-linear Model Predictive Control
NVM	Non-volatile memory
PUS	Packet Utilization Standard
OBDH	On-Board Data Handling
OCP	Optimal Control Problem
OOS	On-Orbit Servicing
ORU	Orbit Replaceable Unit
RCS	Robotic Control System
RICADOS	Rendezvous, Inspection, Capturing and Detumbling by Orbital Servicing
RMC	Robotics and Mechatronics Center
RPO	Rendezvous and Proximity Operations
RSI	Robotic Systems Integration
RSO	Resident Space Object
SASI	Satellite Simulator
S/C	Spacecraft
SMS	Space Manipulator System
TC	Telecommand
TCP	Tool Center Point
TM	Telemetry

Dedicated to my Dear Grandparents - Babcia Basia i Dziadek Staś

1 Introduction

This thesis work aimed to contribute to the technology development applicable to on-orbit servicing and active debris removal mission or any other type of orbital activities which entail close proximity operations between two spacecraft where the control system for the autonomous manipulation of robotic arm is required. In order to formulate research objectives firstly the state-of-the-art was reviewed to gain extensive knowledge about the research topic and currently proposed solutions. The review of literature, the motivation supporting the choice of the research topic and finally the definition of the main research question and the sub-questions are presented in this Chapter 1. The research project was performed in collaboration with German Aerospace Center (Deutsches Zentrum fur Luft- und Raumfahrt / DLR).

Firstly, Section 1.1 describes the motivation for this research work and puts it into the context of the past and on-going projects. In Section 1.2 the more detailed look into the literature review in this research domain is presented, specifically focusing firstly on the definition of ADR and OOS missions and different approaches towards the control of the system consisting of a s/c base and the robotic manipulator depending on which control system is active. Next, it gives a more detailed look at the current GNC solutions for close proximity operations and robotic manipulator with major focus on the control capabilities. Further, the review of existing implementations of the Model Predictive Control method in Space field and Robotics is presented. Finally, it is underlined that this thesis project was performed in the framework of RICADOS project of DLR - Rendevous, Inspection, Capturing and Detumbling by Orbital Servicing. Thus this system was described and the general DLR heritage in space robotics and OOS was reviewed.

The literature review concluded in formulation of the main research question which is presented in Section 1.3, the sub-goals to be achieved were linked to the findings from the review phase and they allowed to orient the work process.

1.1. Motivation

The rendezvous and proximity operations (RPO) also referred to as close proximity operations (CPO), have always been an intrinsic part of space activities since the very beginning of space era, considering Apollo times with the Lunar orbit rendezvous, the generation of Soviet space stations with the crew flight, a transfer of astronauts to e.g. Mir station, and finally the successful on-orbit assembly of the International Space Station. Despite not being a new concept, RPO however has a quickly emerging categories of new applications, for which the technology must be re-assessed or developed from scratch given the very unique challenges.

One of the applications belongs to the domain of active debris removal (ADR) addressing the increasing need of limiting the number of non-functional objects orbiting the Earth. According to the newest space environment report released by ESA [1], the estimated number of objects orbiting our planet is 30025, considering both the traceable objects of a known nature and the unidentified, covering all the Earth orbits. The GEO and LEO orbits, identified as protected regions, are estimated to enclose 2210 objects according to the report. Since the beginning of space operations in late 50's, the number of objects in the Earth orbits had been constantly growing. The increasing orbital congestion with the man-made objects puts the operational satellites as well as the future missions at higher risk of collision - at the high orbital velocities, even a collision with small debris can cause a chain reaction, hence likely produce numerous debris, which is known under the name of Kessler syndrome.[2] It is clear, that non-functional satellites are one of the main potential sources of such a cascade effect, hence it is highlighted that every mission design shall account for the end-of-life/disposal phase. In case of the satellite malfunction during its operational life, telecommanding a disposal maneuver might be impossible, and the only way to dispose a satellite from its orbit is active removal by another satellite, which is addressed by the emerging ADR missions. Moreover, the sustainable aspects of the orbital environment are evaluated in the frame of on-orbit servicing (OOS)

missions, whose aim is to provide the life extension services, on-orbit recycling and generally perceived servicing of a satellite, which can extend the mission operational time of a traditionally designed satellite without a need of launching a new one. Their potential has been already proved in extensive studies [3, 4] with many agencies and researchers currently working on related technology development projects.

The expected sustainable transition of the space activities with increasing collective approach to safeguarding of space environment is highlighted by UNOOSA in the *Guidelines for the Long-term Sustainability of Outer Space Activities* [5], as well is a part of expected ESA's Technology Strategy [6]. It is identified as a target goal to increase Europe's contribution to space debris by 2030, including development of technologies necessary for the successful active removal of space debris by 2025, such as advanced GNC for CPO, in-space robotics and servicing.

Figure 1.1: Canadarm2 [7] on the International Space Station (Credits:CSA)

Since servicing another spacecraft demands dexterous ability of the system, the robotic manipulator payload of the servicing spacecraft is identified as the best candidate to perform such a task [8]. Although the orbital robotics have already been demonstrated in-flight they had a very low level of autonomy. The first demonstration of the robotic arm manipulation in space was performed by DLR in a project ROTEX in 1993 [9, 10]. The technology has been successfully used on-board Space Shuttle with its Canadarm and on-board ISS with its derivative Canadarm2 [7], and is being evaluated for the new implementations such as ADR, on-orbit servicing and assembly. [11, 12] Nevertheless, the mentioned manipulators had no autonomous capacity and had been controlled by astronauts on-board or form the ground.

The prime motivation of this work is to support the technology development in these mission domains, involving the robotic manipulations control, that are foreseen to play an important role in shaping the future of Earth orbital environment and space exploration.

1.2. Background

The relevance of the on-orbit servicing and active debris removal missions in ensuring sustainable access to space is very clear. In order to understand what is the current state of the art in this domain a literature study was performed. In this Section 1.2 the more detailed information regarding the definition of ADR and OOS missions is given. Firstly, in Section 1.2.1 the history of close proximity operations in space missions is presented, which is important to understand the existing CPO applications and technical solutions. Different definitions of OOS missions are explained along with the description of the mission phases with the main focus on the reach and capture phase. The major challenge slowing down the transition from theory to practice is development of advanced GNC capabilities that would guarantee the maneuver safety, precision and collision-free operation. The existing GNC solutions for well-known applications are briefly presented and the technology gaps identified are described. The main interest is in the control technology solutions applied to relevant missions or solutions being currently investigated. The advanced control technique based on the future prediction of the dynamic system evolution called Model Predictive Control (MPC), is of a high interest for potential application in robotic capture of OOS/ADR mission. The review of MPC applications in space and robotics research area is presented in Section 1.2.2. This research work was performed under supervision of German Aerospace Center (DLR) in the framework of RICADOS project and for this reason an overview of the DLR heritage in space robotics and OOS is briefly presented along with a bit closer look into RICADOS project in Section 1.2.3.

1.2.1. Proximity Operations in Space Missions

In next paragraphs the most important aspects of the proximity operations in space missions are presented in order to clearly define the context of the project and the possible relevant applications. Firstly, the historical background

of the on-orbit servicing missions is presented and different definitions of OOS mission types and service classes are explained. OOS/ADR space mission involves specific phases which are briefly introduced. Moreover, the methodology for capture and removal of the target satellite depends on the characteristics of a debris, such as the knowledge of its physical properties and docking interface. The classification of target spacecraft non-cooperativeness for capturing is described. Next, overview of past missions and on-going projects is presented with a special focus on missions incorporating the robotic arm manipulations. Differences between the control strategy for a robotic satellite system are further explained.

Definition of ADR/OOS

Initially, in 80-90s on-orbit servicing was considered as not having enough financial feasibility to prove advantageous over launching new mission [3], along with lacking the matured technology required to safely perform such missions. Since then the space sector has been evolving - an increasing number of involved agents, such as universities and private companies, the new types of space missions, e.g. flying formations, communication constellations, and also the changing paradigm to sustainability of an orbital environment contributed to positive re-evaluation of on-orbit servicing.

The OOS missions can be defined as a type of mission during which one spacecraft (usually referred to as a servicing satellite, or a chaser spacecraft in more general way) is giving service to another spacecraft (called a client or target spacecraft/satellite). These missions can be primarily divided into three different types: *observation, motion and manipulation* [13]. The first one, *observation*, concerns the inspection of another satellite or orbital structure performed by another spacecraft. The approaching satellite is equipped with the vision system, which enables taking the pictures representing the state of a possibly failed satellite (a missile, or another human-made space object) and down-linking this data for the assessment to be performed by the engineers on ground. The goal of the *motion* type of OOS mission is to assist another satellite enabling its re-location or station-keeping. Usually this type of mission is required if a spacecraft receiving service is not capable of performing maneuver on its own i.e. when its fuel has been depleted. The re-location from the operational orbit into the Earth atmosphere, is essentially the controlled de-orbiting, hence the active debris removal type of missions can be assigned to this category of OOS. The last service class, *manipulation*, is the most complex as it involves the close approach to the client spacecraft, capture and repair while maintaining a client e.g. with the use of a robotic arm.

Service class	Examples of services
Observation	Remote inspection
Motion	Station keeping
	Relocation
	Disposal and de-orbiting
Manipulation	Refueling
	Maintenance
	Repair and retrofit
	Docked inspection

Table 1.1: OOS service classes [13]

There are many motivations which prove feasibility of on-orbit servicing and its benefits. Firstly, repair of a malfunctioning spacecraft evidently enables the continuation of its mission, which otherwise would have to be aborted causing great financial loss. The repair can involve e.g. replacement of a mechanical part enabling the solar panel deployment in case of the drive mechanism malfunction, or a complex repair of a scientific instrument on-board the client. Secondly, the fuel depletion has a significant impact on satellite operations, hence refuelling offers the life extension service to the mission. Another potential use case is the orbital replacement of a unit, e.g. the atomic clock in navigation satellites, the lifetime of this type of mission is directly constrained by the decreasing precision of a clock, hence its replacement would allow the continuation of a service discarding the need of launching a new satellite. Moreover, servicing paradigm may potentially transform the traditional approach towards mission and satellite design encouraging reducing design redundancy, thus driving down the production cost, as a potential repair or unit replacement would be available in orbit [4].

In order to assess the applicability of OOS, not only the technical feasibility shall be confirmed, but also the economic viability shall be analysed [3]. This type of mission are justified if the satellite replacement cost outweighs the cost of the service, as was e.g. the case with the Hubble Space Telescope multiple repairs during which a costly scientific instrument was serviced by astronauts. Nevertheless, it is foreseen that when the OOS technologies will be well established, design of he future assets which includes the servicing as an intrinsic part of the mission, could be more cost-effective [3].

Lastly, ADR missions, which fall under category of OOS, gain an increasing recognition. Disposal of a spacecraft has two main motivations - decreasing ground casualty risk associated with uncontrolled reentry if the space-

craft is estimated not to fully demise upon an atmospheric reentry, secondly given the already existing problem of space debris it shall be ensured that a new debris is not generated. In the end-of-life, satellites in GEO are maneuvered into a higher graveyard orbit, where they do not pose any potential risk for the operational spacecraft. The standard practice for satellites end-of-life (EOL) in the lower parts of LEO is to leave it in its operational orbit, the orbit decay due to the atmospheric drag will naturally lower down the perigee and cause a re-entry of a satellite. If a time required to perform a natural re-entry is too long, de-orbiting maneuver is usually a preferred strategy which ensures the satellite is disposed in the end of life. The problem occurs when the satellite is not able to perform such a maneuver, due to malfunction, and continues orbiting the Earth remaining a potential source of debris generation, which currently is the case of ENVISAT, the European observational satellite whose mission ended following the unexpected loss of contact in 2012. It gave birth to many debris removal studies performed in the scope of e.Deorbit mission [14], and other ADR-related activities performed by Clean Space at ESA.

Figure 1.2: Artistic visualisation of the ENVISAT capture (Credits:ESA)

A space mission for OOS and ADR consists of similar phases. Firstly, the standard launch and early orbit phase (LEOP), far rendezvous phase, close rendezvous phase and inspection fly around usually followed by synchronisation phase in which the rotational axis of the chaser satellite shall match the rotational axis of the target spacecraft. Next, depending on the specificities of the mission, in ADR the following phases are: capturing, stabilization of a stack configuration and removal phase. In manipulation type of OOS mission it would be capturing, servicing and release phase. The entire mission is performed under ground control continuous supervision, with capacity of teleoperation. These phases can be performed either autonomously or remotely controlled by ground-based mission operations. Capturing phase plays a crucial role in the entire mission process. Conceptually, many methods for space debris capturing have been proposed. The ADR missions can be further classified in terms of different capture method, the investigated solutions such as net, harpoon, tentacles, robotic arm and others are presented in the review [15]. Due to the extensive heritage from robotics and automation industry, the robotic arm is identified as the most mature and plausible solution for the near future. Unlike OOS, a target satellite of ADR mission is usually uncooperative and unprepared, which is also the case of ENVISAT. It essentially means that an object is incapable of maintaining or transferring the attitude information and it does not have neither grappling fixture for capturing nor navigation aids. The paper [15] presents four different classes of non-cooperativeness depending on which information on the target is available, and if it is prepared or not for the capture. The proposed classification is presented in the table 1.2. Clearly, higher level of target satellite cooperativeness, hence more certainty in the information of its state vector and characteristics of the grasping interface, decreases technical complexity and risk of the mission. Also, the methodology for capture and removal depends on the characteristics of a debris, therefore the mission is usually designed having a specific debris (dysfunctional satellite, upper stage of the rocket etc) in mind.

Class	Physical properties	Docking interface	Characteristics
A	Known	Available	Fully known, satellite is prepared captured
B	Known	Non-existent	Restricted compatibility with the capture
C	Unknown	Available	Dockable, dynamical behaviour unpredictable prior to close rendezvous
D	Unknown	Non-existent	Undetermined

Table 1.2: Classification of non-cooperativeness for capturing [15]

Overview of ADR/OOS missions

The on-orbit servicing missions have been already performed in space, with the best example of Space Hubble Telescope which was serviced five times during manned missions. The first robotic systems used in space can be traced back to the first deployment of the Shuttle Remote Manipulator (SRMS) from the cargo bay of the Space Shuttle Columbia in 1981 [8]. The International Space Station (ISS) assembly was performed with the use of robotic arms, nevertheless they were controlled directly by astronauts or ground control centre without autonomous capabilities. Another successfully functioning robotic system is Canadarm2 on the ISS. The overall survey of on-orbit servicing concepts and investigated technologies is presented in [3], and also shows state of the art in assembly robotics such as the Ranger telerobotics system at University of Maryland (1995), Skyworker robot (2001) and first version of NASA's Robonaut (2002). The more up-to-date state of the orbital robotics tested on ground was analysed in [8]. A review of engineering developments throughout the human history in the domain of OOS is also presented by Wei-Jie Li et al. in [16]. The authors review more than 130 launched or proposed spacecraft missions in the area of OOS, which were divided onto regular scale and large scale spacecraft. All the major aspects are addressed being the historical view on spacecraft missions and space robotics programs developed by main actors in Europe, USA, Russia, China, Canada, Japan and others. The concepts of Large Space Systems such as Space Shuttle Program, Space Solar Power Station and Large Space Telescope are presented with different assembly methodologies for all these structures. The authors present a good overview of an analysis of the mission and technology architecture from the systems engineering point of view. The on-orbit studies are currently performed by majority of space agencies and relevant institutes - NASA [4], DARPA [17], DLR [18] further described in detail in Section ?? due to its high relevance. ESA with its Clean Space Office is working on studies such as Space Servicing Vehicle and recently launched OMAR study [19, 20].

The technology investigated under the OOS frame can be easily transferred into ADR missions, as there are many commonalities between both of them, such as the relative navigation for far and close rendezvous approach and robotic capture technology for grasping a target satellite. To this day, there have been many studies and in-orbit technology demonstrations, however none of them have ever been implemented in a real mission with uncooperative target yet. The most relevant to mention are RemoveDEBRIS [21], e.Deorbit mission [14] and Restore-L technology demonstration mission [22] which was recently announced to develop into an OSAM-1 mission scheduled in 2023 [23]. Astroscale is preparing for the upcoming mission Elsa-D [24] technology demonstration this year with the magnetic capture system. In October 2019, the European ClearSpace-1 mission was announced [25], led by a Swiss consortium in collaboration with ESA and industrial partners, ClearSpace-1 aims the capture and removal of a payload adapter, the VESPA, an ESA-owned object. Due to the initial phase of the mission, the capture solution has not been officially presented yet. The mission is foreseen to be launched in 2025-26 being the first European ADR mission. It is clear that the future of OOS/ADR missions is recognized worldwide.

The numerous studies and efforts done towards making OOS/ADR operational have been introduced, one of the most relevant lessons learnt are derived from the Japanese mission Engineering Test Satellite VII (ETS-VII) launched in 1997 by the National Space Development Agency of Japan (NASDA) and secondly the mission Orbital Express launched in 2007 by The Defense Advanced Research Projects Agency (DARPA)/NASA [26, 17]. ETS-VII spacecraft was equipped with a 6DOF manipulator arm and successfully completed multiple on-board experiments for the autonomous rendezvous/docking validating the robotics technology in space. The latter consisted of a variety of experiments, such as robotic servicing tasks, verification of free-flying space robot coupled dynamics with a coordinated control between the manipulator's reaction and the satellite's response and capture of a target satellite. The ***coordinated control*** (in different sources named also a ***collaborative control***) means that the on-board attitude control system of a chaser platform and a robotic arm control system are distinct and independent, however they collaborate by feeding each other with the anticipated motion and related disturbance. Both controllers are active during the arm operations, satellite base anticipates a disturbance due to the robotic arm dynamics, and reacts to it with the feed-forward compensation signal. The mission requirements on the attitude control were maintained by reaction wheels and the gas jet thrusters were the actuators compensating the robot arm's reaction. Teleoperation of the robotic arm was performed in two different modes - the telemanipulation and supervised control, the ETS-VII robot arm had three following control modes: arm tip position, joint angle and compliance control mode. All of them were successfully verified in orbit, during the mission [27].

The Orbital Express mission, consisting of a servicing spacecraft with the 6DOF rotary joint robotic arm and its manipulator control unit, demonstrated successfully on-orbit servicing of spacecraft including rendezvous, transfer of battery and CPU modules, and transfer of propellant. The arm operations were pre-scripted and fully autonomous. The control strategy for the robotic manipulation phase was different with respect to ETS-VII - the manipulator was operating on a ***free-floating*** base, once the target satellite has been acquired by the visual system of the servicing spacecraft, the satellite platform transitioned to free drift and the robotic arm performed a visual servo operation to track and capture the object. Hence, the attitude control system of a satellite was turned off during the manipulations and no compensation of the angular momentum due to the manipulator dynamics was performed. The platform's attitude was perturbed during the manipulator's operations, and only corrected after an

arm motion script had completed, using its reaction wheels [28]. This approach clearly has its advantages and disadvantages, indeed it makes the control strategy more simple to handle as only the motion of an arm and its actuators are considered in the manipulation phase. On the other hand, it does not depict fully the dynamics of a whole system as the resulting motion of the satellite's base remains a subject to disturbance. If mission constraints and requirements do not impose very strict demands on the spacecraft attitude during the robotic manipulations (due to e.g. pointing requirements for the communication antenna position, sun-pointing for power generation etc.) the free-drift of the base might be acceptable. However the question remains, to what extent a delayed actuation of the satellite's attitude control system can effectively counteract the produced disturbances meeting the mission requirements, upon the end of the manipulation phase. Depending on the physical characteristics of the spacecraft and the mission requirements the accumulated attitude error might be exceeding the allowable limits, in such a case the coordinated control strategy could be a preferable option. Another control strategy which is being investigated since recently is ***combined control***, as defined by ESA in [29]. It concerns the system in which the chaser platform and the robot system are controlled by a single control system that coordinates all sensor data and the actuation, both the satellite's and the arm, to obtain the desired chaser motion and robot system configuration. This strategy has been investigated already in [30] within the research project in DLR, RICADOS.

The study "Combined control for robotic spacecraft and manipulator in servicing missions", COMRADE, is the most recent work on the development of the combined controller strategy [31]. The authors designed a control system with the use of H_∞ and nonlinear compliance control for the synchronization phase, reach and capture maneuver, grappling and stabilization.The external input to their controller was both the relative reference trajectory for the gripper and the chaser setpoint. Furthermore, a very good overview of GNC solutions for robotics manipulator in space is presented in a survey [32].

1.2.2. Review of MPC Applications in Space & Robotics

This section presents the survey of existing applications of MPC in Space and Robotics research areas. The main objective of the literature review presented in this section was to identify the already existing solutions in order to firstly understand what is the current state of the art, if there are any proposed applications of MPC into the space robotic system and secondly to collect a relevant set of references for the more high-level applications in other spaceflight fields and robot manipulators to facilitate the further trade-off of the controller design choices during the research period.

General Space Applications

The versatility of the predictive technique and its constant spreading in industry due to advances in science, technology and market makes it a mature technology for guidance, navigation and control applications in the aerospace sector. Some of the applications for which MPC is considered are planetary rover path planning, LTV MPC for wheel momentum damping by thrust orientation mechanism, LTI MPC for stabilization, Hybrid MPC for navigation of small UAVs, decentralized LTV MPC for formation flying and others [33]. The predictive techniques are implemented both for the guidance problem, when the trajectory is a solution of an optimization constrained problem, and for the control problem itself for which the control input values are found based on the knowledge of the future dynamics evolution.

The work [34] proposes the MPC application for powered descent guidance and control for thrust vectoring control, the formulated LTI MPC problem is solved via QP optimizer with the use of an accelerated dual gradient projection, which is an algorithm suitable for embedded applications. The control framework is used to optimally steer the vehicle towards a desired state; the trajectory is a controller's outcome and not a flight reference or constraint. The controller performance was further evaluated via performance-in-the-loop simulations on the processor running at 1GHz, the C code was generated directly from Simulink. Due to CPU architecture/scheduler, there might be spikes in the recorded task execution time if the CPU resources are assigned to a different process, then the MPC task is suspended and resumed when the CPU is newly available; this wait time translates in delays in the execution of the MPC code that can lead to a violation of the maximum step time constraint. The possible time delays due to preemptive scheduling of CPU shall be taken into account when performing analysis of controller performance, as it might pose an important constraint in a real system.

Interestingly, ESA with DLR had a very detailed look into the possible MPC integration into Space projects. German Aerospace Center developed an extension called MPCTOOL within the ORCSAT project (2009-2011) [35] with large emphasis on real-time implementation capabilities of MPC. Some of the applications identified were orbit synchronization and impulsive hopping, both manuevers relevant for space mission far and close rendezvous phase. The main features of the toolbox are further explained in [35]; Another toolbox was developed for ESA within ROBMPC project (Robust MPC for Space Constrained Systems) for linear time-varying (LTV), MPCSoft.

The predictive control was also proposed for the spacecraft rendezvous in Mars Sample Return scenario in [36], the LTV controller based on linear programming optimizer was employed for autonomous capture. The time-varying model was chosen as the equations used for the trajectory prediction model describing the relative motion on an arbitrary elliptical orbit have time variations due to J2 effects. The integer decision variables were

avoided to prevent increasing complexity of the optimizer; it is handled by solving multiple instances of continuous optimizations at each control step. There are 4 different MPC controllers designed, such that the rendezvous is divided into 3 phases with an additional controller to perform CAM during the final moments of RDz; in such a way the MPC controller is able to function in the given finite computational resources. For the forced terminal translational guidance phase (FTTG), which is the guidance from the final holding point at 100m to a position 3m from the target where it can capture the target on a free drift trajectory, the MPC controller maintains target pointing and handles attitude regulation to an externally provided setpoint with the use of thrusters. The controller is implemented using the QP-based LTV-MPC controller block from MPCTOOL, a linearised quaternion-based prediction model is used for the relative attitude control. Furthermore the authors describe design of MPC controller for every phase, each of the designed MPC controllers has a common output function to convert the ΔV into finite-duration thrust pulses in the inertial frame. What is interesting, the authors present the possible avionic architecture which could be able to cope with the MPC needs, particularly the selection of the Central Processing Unit (CPU) is the key for the MPC embedded implementation. It is of high importance to take into account available space-qualified processor computational performances, as it is usually much more constrained than a performance of the workstation.

Predictive control solution is also proposed for application such as vision-based spacecraft landing for which the nonlinear MPC is tested and successfully achieves the real-time performance also with the noisy estimates of the state [37]. The nonlinear dynamics of the model were solved via nonlinear programming problem in this study. The work presented in [38], proposes a linear tube-based MPC for the spacecraft approach maneuver to a free-tumbling target, which allows to guarantee feasibility and stability of the system in the presence of the uncertainty in the chaser state, that normally could potentially remove guarantees of stability from classical model predictive methods. The tube-based MPC concerns the generation of a bundle or a tube of trajectories, each of them corresponding to a particular realization of the uncertainty. The authors of the paper propose the solution which allows robustness of the MPC controller, in contrast of the conventional MPC which is not capable of handling additive and multiplicative disturbances. They define the control problem as one of tracking, rather than regulation. Furthermore, the definition of the uncertainty in the chaser state is explained. The classical attitude control problem of a spacecraft can be also solved with MPC. The authors of [39] apply the explicit linear predictive controller to a micro-satellite attitude control system with the thrusters and reaction wheel actuators, that allows to decrease computation effort to a table-lookup. This approach is advantageous given limited power and computational resources and also allows to take into account constraints. The interesting point is that authors utilize the bang-bang modulation scheme with dead-zone in order to account for the on-off nature of the actuating thrusters. Another work presents a linear implicit MPC controller for a spacecraft attitude problem with reaction control system and reaction wheels [40]. The authors also derive in detail a stability condition to prove that the closed-loop system becomes input-to-state stable. As can be seen, there is a huge variety of the potential MPC solutions for applications in spaceflight and its advantages are increasingly recognized. From the presented work, the preliminary ideas on the controller formulation for this project can be created.

Robotic Manipulator Applications

The current state-of-the-art of the predictive control for terrestrial robotic manipulator in general, and specifically for the on-orbit servicing mission has been analysed. The predictive control of robotic manipulators is proposed in variety of studies as it allows for considering the restrictions on the manipulator performance such as limitation in the position, speed, acceleration of the motors and the maximum torque available, when finding the optimal control input. In study [41] a multivariable constrained predictive controller is designed and tested with the use of a linearized model of the general nonlinear dynamics of the robotic manipulator. Reference tracking control of a 2-DOF manipulator modelled with the use of the Lagrange-Euler method was achieved with constraints on control rate. The authors of [42] propose an efficient approach for nonlinear MPC of a m-link industrial robot manipulator for a set point trajectory, and compare its performance with the frequently used computed torque control. The inverse and direct dynamic robot models were obtained from the Lagrangian equation. In design, the predicted future output error was taken into account, such that the applied control variable u consists of three terms, due to the tracking position error, the term for disturbance rejection and the last term corresponding to a model compensation. The controller robustness was tested in the case of disturbance rejection and model mismatch with efficient results.

Close Proximity Operations and Robotic Capture Phase

The MPC architecture is identified as a good candidate for the close proximity operations involved in the on-orbit servicing mission. To start with, the experimental evaluation of MPC for approach and docking maneuvers is demonstrated in [43]. This control strategy deals well with constraints imposed on the maneuver for trajectory planning which include thrust constraints, a LoS constraint linearized through polyhedral approximation and an obstacle avoidance constraint linearized through a rotating hyperplane. The evaluated algorithms are used to control the vehicle position, the control problem concerns only translational motion fof the center of mass for the

close rendezvous phase, nevertheless the formulation of the cost function from this work and the performance metrics are of interest. A linear MPC with a QP solver was chosen for this application; it is defined as a setpoint MPC – it takes the difference of the x in the current step with x_t, the targeted final condition. The authors defined four performance metrics: control effort, time to complete the maneuver, constraint handling and computational cost.

The predictive control applied to a robotic manipulator mounted on a satellite has been proposed by research community in the course of past years. In one of the most recent studies, the control of free-flying system is proposed for a simultaneous capture and detumble of a space object by Josep Virgili-Llop and Marcello Romano [44]. The paper presents mainly the aspects related to design of the system guidance, the problem solution is obtained by solving a collection of convex programming problems, making the approach suitable for onboard implementation and real-time use. The manipulator configuration and base-spacecraft attitude at capture, as well as the manipulator motion during the final seconds of the maneuvre is pre-set and not subject to optimization. To reduce the control effort of the chaser, the base-spacecraft is operated, during the pre-set manipulator motion period, in a translation-flying/rotation-floating mode, where the base spacecraft attitude is left uncontrolled. In the paper the authors present in detail the derivation of the kinematics and dynamics equations, as well as formulations of linear and angular momenta of the chaser and combined system. The objectives of both the maneuvers are encoded with a unified set of terminal constraints, once the capture is finalized, the requirements to detumble the stack configuration system are already met when the manipulator's motion is gradually stopped. Both maneuvers are constrained in order to achieve the goal which is a zero post-capture angular momentum on the combined chaser-target system. The constraint on the chaser's attitude and manipulator's motion is a zero relative velocity between the chaser's end-effector (EE) and target grappling fixture, which implies simply that the position and velocity of the EE shall match the position and velocity of the grasping point on the target. When it comes to the optimization, the cost to be minimized is formulated as a quadratic expression; in optimization constraints the authors implement limit on the chaser's control, a maximum force constraint, and a keep-out zone constraint. The first step is optimization of the system-wide translation, the second step is optimization of the internal reconfiguration. The authors give a throughout overview of the mathematical formulation of the convex programming problem.

In the work [45], the capture maneuver itself (from the study introduced in the previous paragraph) is explained in more detail. A sequential convex programming procedure, overcoming the presence of non-convex constraints and nonlinear dynamics is presented. The definition of the system constraints, optimal control problem, the convergence proof and an explicitly convex line-of-sight formulation are clearly presented by the authors. The mathematical formulation of the problem convexification is also explained, which might be a good reference.

The author of the thesis [46] proposes two control algorithms based on the predictive control for the planar system composed of a satellite platform and 2-link manipulator with rotational joints, a generalized predictive controller and a dual-mode MPC. The system is assumed to be free-floating and the control input vector of the system consists only of two control torques of manipulator joints. The control problem was defined as a tracking problem with objective for the outputs y to follow a reference trajectory. The interesting point is the inclusion of the perturbation terms which are introduced to the first n_c control inputs, their values are determined by the MPC through an optimization process; as a result the prediction model for state and the formula for the control input take another form, called transient mode, for these terms; if the control horizon is larger than n_c the remaining terms are defined by the standard prediction model, they are further called terminal mode, hence the name dual-mode MPC.

Lastly, the recent work, performed by researchers from the Space Research Centre in Warsaw Poland, proposes a control system for free-floating space manipulator based on NMPC [47, 48]. The authors develop nonlinear model of the system with a manipulator (with n rotational joints) mounted on a satellite; in this free-floating system the position and orientation of the satellite is not controlled during the capture maneuver, the motion of the manipulator influences state of the satellite. The detailed derivation of equations of motion from the Lagrange equation is presented, in which the generalized coordinates vector q includes the position vector of the satellite, its orientation and the vector of joint angles (based on the models in [49]); the simplified planar case is considered, in which the satellite is equipped with a manipulator that has 2 DOF, the full system has 5DOF (three for s/c). According to the nature of a free-floating system the manipulator-equipped satellite is not using its thrusters and momentum wheels during the motion of the manipulator, the control vector is composed solely of driving torques in manipulator joints. The proposed control system consists of two blocks - trajectory planning module and model predictive controller. In the first module, the optimization criterion used was a quadratic norm connected with the power consumption of manipulator motors, the authors propose for later improvement to include the term allowing for reduction of reaction torques and forces induced by motion of the manipulator in order to reduce changes of satellite orientation. The objective of MPC is to compute velocities of manipulator joints that will minimize an error between the measured EE velocity and reference EE velocity; however the reference EE trajectory is given as a set of EE positions and the gain matrix is introduced in order to define the EE position error, instead of velocity error, into the formulated definition of the manipulator joints velocities. In the second module, the NMPC is used

for assuring realization of the selected manipulator trajectory; as the first module already performed the global trajectory optimization the control torques minimization in a short time scale is skipped during operation of NMPC to reduce the computational effort. The authors confirm the performance of the designed system via simulations, accounting for the limited knowledge of system parameters, and adding white noise to control torques applied at the manipulator joints to simulate disturbances that could be experienced by the real system.

1.2.3. DLR heritage in Space Robotics and OOS

DLR has a very strong expertise in technologies directly related to the development of OOS-robotic systems, which proves a successful execution of three pioneering space robots orbital experiments [50]. The first remarkable mission dating back to 1993 was the Robot Technology Experiment (ROTEX), was one of the important milestones of robot technology in space which was flown on the Space Shuttle Columbia. The robotic 6-axis manipulator, was located on board the Shuttle, during the mission it successfully proved operation under variety of operational modes: teleoperation on board (astronauts control), teleoperation from ground and sensor-based off-line programming controlled remotely from ground control, hence becoming the first remotely controlled space robot system [10]. The ROTEX objective was to validate the robotics technology under zero gravity, sensor-based control features and control operations on different autonomy levels- full autonomy till the man-machine cooperation. Its multi-sensory gripper worked faultlessly during the mission, performing also operations proving servicing capabilities such as assembly, ORU exchange and ground-controlled capture of the free-flying object. The challenges of the robotics operations controlled directly by ground are mainly the large and varying time delays were treated with the predictive simulation in the ground control, which included the robot's sensory behaviour, proved to compensate for this time delay. With this experiment, DLR remarkably gained the forefront of European space robotics expertise.

GETEX, was a German Technology Experiment performed on board ETS-VII satellite, the DLR's telerobotic and programming system was implemented to control the robotic manipulator of the Japanese satellite in April 1999. The satellite was equipped with a robotic manipulator, as is explained in section 1.2.1. Several experiments were performed, which enabled successful verification of the telerobotic ground control station function during the control of the robot arm. The peg-in-hole experiment with the vision and force control scheme was performed, as well as the variety of robotic maneuvers verifying the dynamics models of the free-floating space robots. The manipulator performed maneuvers firstly with the attitude control system of the satellite switched off under assumption of no external forces acting on the free-floating robot. The results helped to realize that the external disturbances acting on a spacecraft in LEO must be accounted for in a correct modeling of the manoeuvres [51].

In order to verify the design of a light-weight torque-controlled robotic joints in real space environment, the experiment ROKVISS, standing for the Robotics Component Verification on the International Space Station, was performed in 2005 becoming the second space robot mission after ROTEX performed by DLR. The experiment mission was achieved with a successful installation of the robotic hardware components on the outside of ISS in 2005, which allowed the verification of different control modes on a full spectrum from high system autonomy to the telepresence mode by the ground control engineers via a direct radio link operations. In telepresence mode, the on-orbit manipulator was controlled as a slave, by the corresponding manipulator at the ground station via a force-feedback device in a real-time, generating force and position changes as control commands for the joints. The joint controller structure allows implementation of position, torque or impedance control. Not only the joint dynamics changes, but also the impact and contact dynamics upon the influence of harsh space environment (temperature variations, space radiation) were assessed [18]. Another relevant project is in-flight technology demonstrator DEOS, Deutsche Orbitale Servicing Mission, which was an OOS mission demonstrator of rendezvous, capture and de-orbiting techniques of an uncooperative satellite. The two satellites were to be launched together into a LEO orbit at 550 km altitude and perform the operations in the orbit, the servicing satellite is equipped with the light-weight manipulator. One of the challenges with respect to operations is the continuity of a communication link from ground to LEO, the DEOS servicer spacecraft was to be equipped with an inter-satellite link to a geo-relay as an alternative to direct space-to-ground communication. Nevertheless, the mission has never been launched and the studies were cancelled after definition phase [52]. The more detailed review of DLR's robotics technologies for on-orbit servicing are found in the [50].

The most recent project, successfully finished, is an On-Orbit Servicing End-to-End Simulation, which is a large distributed simulation environment for verification of rendezvous and docking/berthing robotics systems [53]. It meets the main objective of simulating the mission as close to real conditions as possible including the space segment, with software and hardware simulators, and ground segment with its communicational and operational infrastructure. The overview of the system is shown in the figure 1.3.

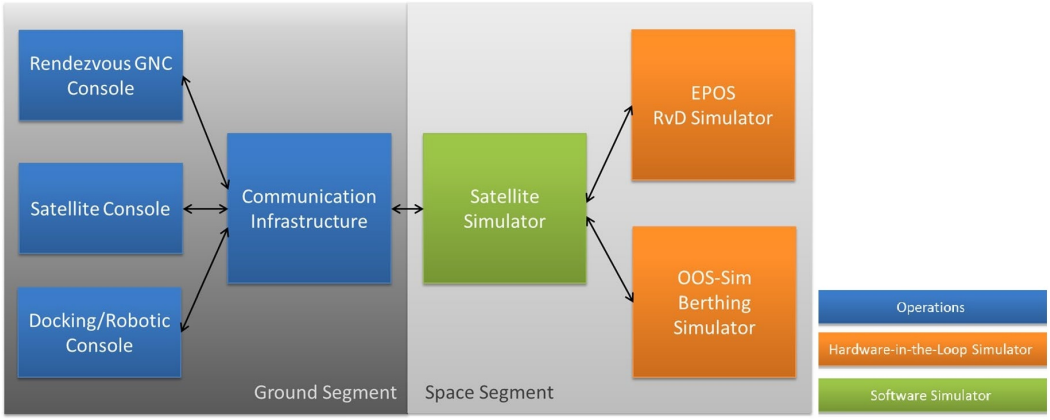


Figure 1.3: End-2-end simulation overview, derived from [54]

Software satellite simulator (SASI) is the first part of space segment, it consists firstly of simulation-specific components and mission-specific components. The first one is the multi-body satellite dynamics numerical simulation of the translational and rotational motion of servicer and client spacecraft. The latter includes the simulation of the servicer's subsystems: thermal, power and AOCS systems in the satellite bus as well as the TM/TC system compatible with the space standardization on data handling which allows communication link with the ground control center [54]. Next, the simulator for space segment consists of two hardware-in-the-loop simulators, which both have interfaces to the software satellite simulator. Firsty,the close range rendezvous phase, with distance between servicer and target satellite below 20m, is simulated in EPOS 2.0, the European Proximity Operations Simulator located at the German Space Operations Center (GSOC). EPOS test bed consists of two 6DOF KUKA robots simulating the motion of a servicer and a client satellite. The servicer is mounted on a rail system to simulate its translation motion towards the target, it is equipped with rendezvous sensors (cameras, LIDAR) which allow for the relative navigation implementation. The tumbling motion of the client is simulated with the mounted target mockup, as shown in the figure 1.4. At the hold point at 3.2m distance (target body origin to servicer body origin), the GNC control system transits into stand-by and since then the simulations are performed in OOS-Sim, the On-Orbiting Servicing Simulator located at the Robotics and Mechatronics Center [55]. The OOS-Sim similarly consists of two 6DOF KUKA robots, simulating a target and a chaser spacecraft, the chaser being equipped with an on-board robotic payload - a 7DOF robotic manipulator simulating motion to capture the grasping point on the target satellite. The detailed description of the facilities can be found in [53].

The ground segment simulator consists of three consoles in the control room for the GNC rendezvous, satellite operations, and robotic systems, from which the operations (as for a real servicing mission) can be performed. During the simulations the operations are supervised from the facility control room where the engineers can directly send commands to the servicer satellite in the way they would be transferred in real mission conditions, such that the time delay and transmission limits are taken into account. In console, telemetry data is visualized, it also enables the generation of commands to be sent to the payload on-board, e.g. in the GNC console the image processing parameters can be changed via the telecommands. The robotics console supports two type of operations: teleoperation mode and telepresence mode, the first one enables the nominal preprogrammed control of the robotics payload, whereas the latter allows the direct control of the manipulator remotely from ground using haptic and visual feedback devices. A telepresence approach is necessary to allow the operator on-ground to control manipulator directly to handle the possible failures and not predefined actions. The robotic console communicates with the robotics on-board controller via the uplink- and downlink stream, the high-level commands sent by ground are decomposed into elementary operations for execution on the robotics real-time motion controller, according to the task-directed programming approach developed at DLR [56]. The more detailed communication between all the parts of the end-2-end simulator are described in the [53].

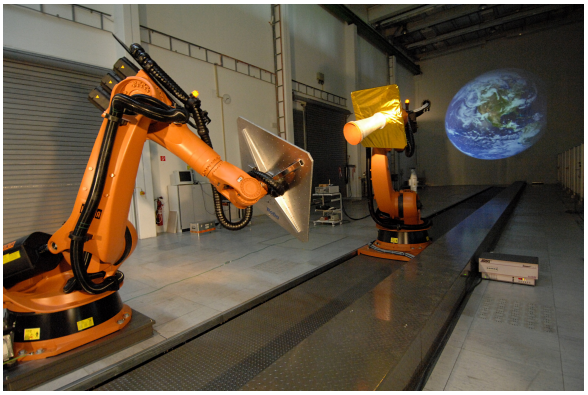


Figure 1.4: EPOS 2.0 testbed at GSOC.

Figure 1.5: OOS-Simulator at RMC.

The unique End-2-End simulation environment is the only known simulation which allows testing in conditions very similar to the real mission, allowing for two different modes of robotic manipulator control. In this environment, the project RICADOS is being developed by DLR which is a system providing robust and reliable operations for all the previously described phases, it is further described in next Section.

RICADOS

The RICADOS system (*Rendezvous, Inspection, Capturing and Detumbling by Orbital Servicing*) exploits the End-to-End simulation environment, it provides robust and reliable operations in all the phases included in E2E. The major goal of the project is to make the whole system as close to reality as possible to simulate and verify several scenarios in realistic manner, i.e. the operations are performed completely from ground. Moreover, for this project a new control strategy for the final robotic capture phase is being implemented, called a combined control. It refers to such a controller set-up that the satellite attitude control system and robotics controller are merged into one controller handling the sensors input from both systems and coordinating all actuators of the satellite platform and robot joints, to obtain the desired chaser motion and robot system configuration. It is expected to provide better performance with respect to the free-floating control strategy in which the AOCS system is switched off, especially for the capture of the uncooperative and tumbling target [11]. The main challenge of the controller is to ensure robustness to system uncertainties - plant parameters inaccuracies, deviations of nominal actuator performances, sensor feedback noise etc. For the controller implementation the solution is based on H_∞ control technique, the first design was already tested in the OOS-Sim and for the controller design is further described in the [30] thesis report.

1.3. Research Overview

The research undertaken for this work is aimed at extending the analysis and understanding of the model predictive control application for space robotic capture systems, currently available in the literature. The research is application-driven with the objective of implementing the MPC controller for the combined controller task in the frame of RICADOS project at DLR. Prior to research, a throughout review of the previous works was carried out during the literature study and critical knowledge gaps in the current research body were identified as presented in the previous section.

Research Questions

The main research objective of this thesis project is:
to increase the technology readiness level of the combined control strategy for the on-orbit servicing mission reach phase by designing and optimizing the system controller with the model predictive architecture and testing its performance via numerical simulations.

From the identified knowledge gaps in the literature, the central question towards which thesis project would be oriented is:

How can the performance of the reach phase of an on-orbit servicing mission be improved by implementing a combined controller with the model predictive architecture?

The main research question is further broken down into more traceable research questions and sub-questions to be answered directly during the project. These can be stated as sub-goals answering research sub-questions. The accomplishment of all sub-goals will imply that the main goal is met.

1. What model of the kinematics and dynamics of the system composed of the arm and of the spacecraft shall be developed and implemented in the controller design?
2. What constraints, relevant for the reach phase, shall be taken into account during the design of the controller?
3. Which type of the Model Predictive Control (MPC) strategy shall be implemented?
   (a) Which type of the MPC is the best candidate to account for different frequencies, performance specifications, admissible uncertainties of the both subsystems (AOCS and the robotic manipulator)?
   (b) Why is this architecture the best candidate for potential implementation?
   (c) Shall the optimization of the control effort be performed on-line or offline, what are the advantages and disadvantages of each of them?
   (d) How can the MPC architecture be implemented?
4. How should the combined controller of the system be designed with the chosen MPC architecture?
   (a) What simplifications shall be taken?
   (b) How can the implementation be verified?
   (c) What case studies shall be tested?
   (d) How the communication interface with ground can be accounted for?
   (e) Does the selected method and developed architecture meet the control performance requirements?

The research questions were set up based on the identified gaps during the literature study period. Question 1. concerns the model of the kinematics and dynamics of the system to be implemented in the controller design. In previous works it was seen that either the planar system was considered [46] or free-floating dynamics were modelled [47]. The only work proposing the combined control and full actuation of both the robotic arm and the s/c base was within COMRADE project [31], the authors used the internally available models (GNCDE, an integrated GNC development and verification environment [57]) considering fuel slosh modes and other parametric variations. The proposed control solution by them was H_∞ and nonlinear compliant control. It is not straightforward how such a complex dynamics model could be used for the prediction model required for MPC optimal control, for this reason the model of dynamics and kinematics of the robotic spacecraft system shall be defined. The plant architecture and dynamics definition are introduced in Section 3.1.

Next, one of the advantages of MPC is that in design process it is very straightforward to consider constraints. This can help the controller to find a solution while respecting the system limits. Question 2. relates to this, the overview of the constraints relevant for the reach phase will be presented in Chapter 2. The authors of [47] for example proposed a scheme of a nonlinear model predictive control, however their dynamic definition was different

than what is targeted for in this project and no constraints were considered. The work on convex optimization-based guidance, presented in [58], introduces constraints such as keep-out zone, line-of-sight and force input bounds. Their constraints definition is also reviewed for the context of this research work. Before a proper design of the controller, the type of the model to be used shall be concluded. In parallel the trade-off concerning choice of MPC type is performed which is addressed by Question 3. and it is presented in Chapter 4. Question 3a), 3b) and 3c) are leading to the finalization of the MPC choice trade-off table which will allow for the structured reasoning to support the choice. Question 3d) concerns the implementation of MPC architecture, the mathematical formulation as well as finalization of the software choice, the proposed software is briefly introduced in Section 4.9.

Furthermore, Question 4. with its sub-questions concerns the controller design itself. As was proved in the literature study, only the cases of a free-floating spacecraft control were identified with some MPC applications. The control of a free-flying system with MPC was not found to be present in the current body of knowledge, only with different control technique [31]. Therefore, it is interesting to investigate its design with the model predictive method to see if it could potentially improve the performance. Question 4a) concerns the simplifications of the combined controller. It was important to start with a simple version of the system and gradually add complexity to it, instead of starting off with too complex problem. The assumptions limiting the complexity of the controller for this project scope are set up. Next, the verification of the design is addressed by Question 4b), in this stage of the project it will be further specified and all the run simulations with objective of design verification will be described. Next, Question 4c) addresses the nominal simulations tests to prove the robustness of the controller to uncertainty. The results of the tested case scenarios are presented in Chapter 5.

Important aspect for the integration in RICADOS is communication with the ground segment. Thus, the controller design shall enable receiving of the telecommands from ground for the manual change of AOCS mode and also downlink the information for ground engineers who supervise the operation. The data encryption is out of scope of this study, however the simple identification of what data coming from controller could be sent to the ground console is expected to be identified. It could further allow testing in the RICADOS project context. It is addressed by Question 4d), and the brief discussion is presented in Section 5.3 aiming in proposing the solution to activate and tune the controller from the ground.

Finally, the conclusions from the work are presented. The tracking performance of the controller shall be critically analysed with respect to the expected performance for such a mission, it is addressed by Question 4e). The interpretation of the simulation results along with the comments on the design and computational complexity will be elaborated. The main conclusions on the system behaviour, and the applicability of MPC in this type of system are necessary for possible future developments.

2

Review of Control Solution Limitations In Robotic OOS Missions

This Chapter presents a brief overview of the most important aspects of a space mission which shall be kept in mind when designing the controller for reach phase with the robotic arm deployment. In order to design and test the controller in the way to bring it as close to the real conditions as possible, a variety of factors should be considered. Here they are divided into the spacecraft architecture constraints, operational mission constraints and environment constraints. Finally, the required performance criteria for reach maneuver are presented.

In general, to ensure the success of the performance of the control system for the chaser spacecraft, one must take into account several features that will affect its design. These can be subdivided into system and mission drivers. Among system drivers, one finds both the use of sensors and actuators. These will introduce errors into the GNC loop due to their implementation, operating frequency, mounting errors, etc. which should be modelled and their influence studied. In addition, the different performances which they can achieve will significantly influence the control and state estimation of the spacecraft equipped with a robotic arm. Regarding the software of the chaser, one finds that the operating rate of the complete subsystem will affect the performance of the GNC algorithms, as the dynamics may be changing at a faster rate than the one the controller can achieve. Additionally, the optimization algorithm to be used will have an influence on the outcome - different considered algorithms may show slight differences that could propagate with the trajectory itself. Finally, the environment and dynamic models developed for the design of control system will influence its actual performance, as insufficient modelling will lead to an incorrect design unable to perform correctly in real conditions [59] [60].

2.1. Spacecraft Architecture Constraints

There are certain limitations imposed on the design of a control system due to the characteristics of the system equipment, software and the interactions between both. This section is further divided into two subsections, the first one describing the aspects related to the hardware constraints and the second describing the limitations due to modelling uncertainties. During the thesis research, it was decided which constraints are important to be taken into account for the controller design in the reach phase by including them as constraints in the controller definition.

Hardware Constraints

The system performance is limited by the characteristics of its elements. The robotic OOS mission consists of a sensor suite for the relative navigation (camera and LIDAR system), the conventional sensors suite of the satellite platform (inertial measurement unit, star sensors, Earth sensors, GPS), the joint torque sensors of the robotic arm and touch sensors on the gripper fingers. None of the measurements provides the exact value, as the sensors hardware itself is not ideal. Therefore the signal cannot be treated by a controller with 100% trust and the noise components usually are added to the output signal. All the actuators, thrusters, reaction wheels and joint torque motors are characterised by a limited performance which should be also accounted for. For thrusters, it is impossible to produce an infinitely large thrust, the desired value of thrust is not immediately achieved upon the controller command and also the thrust profile is not a linear function. The rise time and the small variations from the nominal value are the sources of the error with respect to the desired behaviour. In the similar way, the reaction wheels and joint torquers have limited performance which is the case for all the electromechanical systems.

In order to achieve a desired performance and robustness of the control system, some computations must be performed online to ensure responsiveness to the uncertainties, such as external disturbances and the hardware malfunctions, to which the system is subject during the process. Another part of the required computations can be performed offline and then stored onboard keeping in mind though that data storage is limited by finite memory.

The real-time achievable computations are limited by two main factors - the onboard processing capabilities and the distribution of the computing power. The control of the plant during the reach phase is not the only process to be executed on board, all the other subsystems must remain functional such as e.g. thermal control, communication, power generation and distribution. Therefore the control system is dependent on a variety of factors imposed by the limitations of the OBDH subsystem, e.g. the computing power distribution between all the parallel processes, the computer architecture, floating-point operations and others. The available computational power also influences to what degree the plant model shall be simplified and therefore less accurate. Another limiting factor comes from the power subsystem, it allows only for the limited energy storage on-board, its allocation between all the subsystems might constrain the achievable performance of the actuators. Finally, the workspace definition of the robotic manipulator informs about the achievable configurations, which by definition is the constraint on the set of positions feasible to be achieved.

Modelling Uncertainties

In control system design, a mathematical representation of the physical system is required to map the inputs with the outputs according to the system dynamics. First of all, the parameters such as inertia mass distribution and the manipulator links' masses might entail some errors due to rounding and time-dependent values (e.g. it can be assumed that the system mass is invariant, however during thrusting the propellant is expelled and hence the mass is decreasing). The simplifications in dynamics description, e.g. linearization of nonlinearities, are all sources of uncertainties. The performance and stability of the control system may be severely affected by such modelling errors or uncertainties. Secondly, the system representation depends on the chosen model type. The authors of [61] summarize the main challenges of MPC control system design for aerospace systems. These are identified in system modeling and problem formulation, accounting for safety aspects, such as robustness, fault tolerance and time-delayed interactions and the implementation issues. They also introduce the common model types and explain the limitations due to system modelling approaches with extensive literature review on already existing aerospace applications. Depending on the model definition, the controller will be able to respond in a robust way to disturbance signals and dynamic perturbations in more or less effective way. Furthermore, the complexity inherent to MIMO systems also puts limitations on the achievable performance, these aspects can be further investigated from the book [62]. The more complex system is, and theoretically closer to reality, the more computationally burdensome it is, it must be kept in mind that the limited processing capabilities do not allow for development of an overcomplicated system. The trade-off between the detailed description of the complex dynamics vs the required simplifications was performed while working on first iterations of the controller design.

2.2. Operational Mission Constraints

Another aspect to be considered when designing the control subsystem, are the on-board data-handling (OBDH) subsystem operations and the role of the Ground Segment in all the mission phases. The book on Spacecraft Operations written by Thomas Uhlig, Florian Sellmaier and Michael Schmidhuber, from German Space Operations Center (GSOC) and other contributions from ESOC, is referred to [13].

During the reach phase operations, the ground station engineers supervise the operations on the basis of received information about the current spacecraft status. The downlinked data is usually processed on-ground in order to get a physical understanding of the situation in which the spacecraft is. In the same way, this communication is implemented in the OOS-Sim simulator. From the initialization of the robotic arm deployment until the capture of the target, the information about the spacecraft shall be constantly transmitted to ground. Besides the regular healthy status parameters, which are transmitted during all the phases of the mission, the specific information might be required depending on the characteristics of the phase. This data is transmitted to ground via the telemetry parameters, that can contain status information (such as on/off flags), numerical data or binary data; usually the telemetry information shall be coded into a binary format [13]. When designing the combined controller for the reach phase, the interface with the ground shall be taken into account in terms of identifying what type of information could be indicative of the current maneuver status. Another aspect to be considered is the uplink data, the AOCS shall be able to react to the instructions included in telecommands. The typical commands that are sent to the AOCS include e.g. setting the nominal attitude, commanding orbit control manoeuvre, power up or down of a unit (sensor or actuator), reconfiguration into the redundant unit or change of the operational mode [63]. In order to understand the TM/TC protocols and how the data should be decrypted and then fed to the controller, the books [13, 63] are referred to.

2.3. Environment Constraints

The orbital environment itself poses challenges that a terrestrial robotic manipulator and other systems do not have to cope with. In general, the spacecraft operational environment has effect on space flight hardware due to the vacuum, low gravity, radiation, solar pressure and temperature variations. These elements are intrinsic to every spacecraft - for a control engineer they should not have such a big importance, as all the hardware

components can be assumed to have been already space-qualified. The constraints of the spaceflight hardware design due to the environmental aspects are detailed in [64], and can be referred to if necessary. Nevertheless, the more important focus for a control engineer is on the disturbance forces, their estimated order of magnitude and the constraining factors due to the mission geometry. The disturbances that act on the spacecraft in OOS mission depend on the orbit type, e.g. force due to solar pressure is the predominant disturbance on a rendezvous trajectory of a satellite in GEO, whereas the drag of the residual atmosphere is the predominant disturbance in LEO [65]. The disturbance torques also origin from the gravity perturbations due to J2 effect and micro-gravity effects. For the reach phase only, these external disturbances are often assumed to be negligible, because the relative motion with respect to the target spacecraft is considered. Both chaser and target are in the same orbit few meters away from each other. Therefore it can be assumed that the same disturbances act on both s/c and thus they ”cancel” each other out.

2.4. Performance Requirements

In the available literature on the GNC for close proximity operations, the performance criteria were found specifically for the reach/capture maneuver. The two studies are taken as an example, first one is e.Deorbit mission, and second one is the combined controller project COMRADE. The values are presented in tables below, they are a good indicator of the final performance of the designed control system. Clearly, the performance requirements considered in both projects are almost equivalent, the only difference is in the relative velocity. The final results achieved after simulating the designed controller in a feedback loop (see Section 5) are compared to these values.

The GNC performance requirements for the rendezvous, synchronisation and capture phase were found in the report from e.Deorbit mission (2017) [60]. The considered performance criteria for the chaser spacecraft are the following:

Position [m]	Velocity [m/s]	Attitude [deg]	Angular rate [deg/s]
0.05	0.01	2	0.5

Table 2.1: The GNC performance requirements; derived from [60]

Another study results, specifically concerning the combined controller of the robotic spacecraft (COMRADE), were published recently (2021) [31]. The considered performance requirements are almost entirely identical with the ones above, they are depicted in table below for clarity. The shown values are specifically for the reach/capture phase only.

Relative position [m]	Relative velocity [m/s]	Relative attitude [deg]	Relative attitude rate [deg/s]
0.05	0.005	2	0.5

Table 2.2: The control system performance requirements; derived from [31]

3

Free-Flying Dynamics Model

In this Chapter the description of the dynamical multi-body system is given. The modelling of the such a complex system is not a trivial task, for this task the SpaceDyn toolbox was used. All the modelling choices, nomenclature and valid assumptions are presented in this chapter.

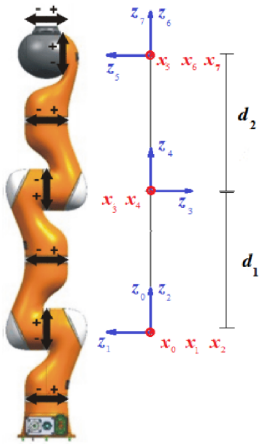
Firstly, the plant architecture is described in Section 3.1. The robotic system being the manipulator is presented in subsection 3.1.1 with the description of the joint axes, torques limits and the admissible range of rotational joints angles. Next, the servicing spacecraft is presented in subsection 3.1.2. The modelling simplifications of its attitude control system are explained and the other important aspects of the architecture. The modelling library SpaceDyn is presented in Section 3.2 with detailed explanation on the implemented attitude convention and coordinate systems. The final equations of motion of the system and the description of inputs and outputs are given in Section 3.3. Last but not least, the state space model definition is used in the prediction model of the MPC problem, therefore its formulation is given in Section 3.4 for clarity.

3.1. Plant Architecture

In this Section the definition of the system of interest is given. Having defined the required theoretical background in previous sections, the technical aspects of the system composed of a spacecraft-base with a mounted robotic arm can be presented. A robotic OOS mission is composed of two spacecraft - the servicer and the target, both of which are discussed herein with a more detailed definition of the servicer. The section is divided into three parts, firstly the aspects particular solely for the robotic system are given, next the spacecraft platform characteristics are presented and last but not least the description of the most important aspects of the target satellite are presented.

3.1.1. Robotic System

The robotic arm of interest is a lightweight manipulator, the KUKA LWR robot, characterized by a light anthropomorphic structure with 7 DOF, all rotational. The revolute joints are driven by compact brushless motors via harmonic drives, the joints are equipped with position sensors on the motor and link sides, and with a joint torque sensor [66]. For its control, desired joint torques are given as inputs. The manipulator, in its fully deployed configuration, is shown in the Figure 3.1, in which the Denavit-Hartenberg frames of the robot are presented.



Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	q_1
2	0	$-\pi/2$	0	q_2
3	0	$-\pi/2$	d_1	q_3
4	0	$\pi/2$	0	q_4
5	0	$\pi/2$	d_2	q_5
6	0	$-\pi/2$	0	q_6
7	0	0	0	q_7

Figure 3.1: The KUKA LWR manipulator; derived from [66]

The robot controller unit (KR C2), as implemented in OOS-SIM, with the Fast Research Interface (FRI) software provides the measurements at a 1 msec sampling rate of the links position q and joint torques τ and accepts the user input command as a desired joint position q_D, a desired joint velocity $\dot{q}_D$ or a user specified joint torque τ_{user}. Via the FRI library the robot is supplied with the set of the appropriate torque τ to be generated by the motors, for the commanded task by the user. The values of the maximum torques are given in [67], ranging from 30Nm for the joint 7,6 and 100Nm for joints 5,4,3 and 200Nm for joints 2 and 1. The admissible range of rotational joints is $\pm 170^o$ for joints 1,3,5 and 7 and $\pm 120^o$ for the remaining joints [66]. Definitions of the system, the link and the motor equations with identification of inertia matrices based on the numerical values provided by KR C2 are presented in [66]. The good overview of the robotic arm architecture with the background on the development process is presented in [68].

3.1.2. Servicing Spacecraft

The servicing spacecraft as implemented in RICADOS can be simplified to a cubic structure controlled by actuators suite consisting of 24 thrusters and reaction wheels.The 3 thrusters positioned at each corner of the cubic base in the way to enable thrust force in all three orthogonal directions, each of them providing max thrust of 10N yielding max thrust in one direction of 40N.

The chaser satellite control and subsequent coupled control of the satellite platform with its robotic arm requires the estimates of the system states based on the data from sensors. First of all, the relative navigation is performed during all the phase based on the LIDAR 3D camera which provides the Line of Sight and range measurements for pose estimation. During the capture phase itself, the target spacecraft can

The authors of [60] considered in chaser design, the navigation sensors suite comprising of the Inertial Measurement Unit (IMU), start tracker, sun sensor and GPS. In the OOS-SIM the chaser is equipped with the camera at the EE which provides the relative navigation with respect to a target based on images, if only the satellite simulator is implemented simulating the real hardware system, the measurement of the camera is not available and therefore the joint angles cannot be computed anymore. In such a situation the servicer s/c pose is obtained directly by considering the states of the model available in the satellite simulator.

The values of the typical frequency for the attitude control cycle are important when considering the sampling frequency of the controller so as the real-machine can handle. The typical frequency for the attitude control cycle is about 2 Hz but higher frequencies up to 20 Hz are possible [AOCS˙02]. The behaviour of the controller will also depend on the chosen sampling frequency, with which the controller is discretized. Too low sampling frequency could induce oscillations in the system, aspects such as noise and sensors bias are also sources of the uncertainty and shall be accounted for.

3.2. SpaceDyn Library

For the definition of the dynamics model the open source available toolbox SpaceDyn was used, it is a Matlab based library for the kinematic and dynamic analysis and simulation of articulated multi-body systems with a moving base. The system can be described in an environment with or without gravity (zero gravity approximation of the motion in an orbit). The library was developed by Prof. Kazuya Yoshida affiliated with Tohoku University, Japan. It consists of the set of Matlab S-functions (system-functions) that allow for the modelling of the system. The toolbox-specific description of the system consisting of multiple bodies and the definition of the dynamics with the relevant frames is briefly given in the following sections [69].

Model of the System

The system is composed of $n+1$ bodies and connected by n joints. Let the body 0 be a reference body, further referred to also as a 'base'. Multiple branches can attach on any single body, a far as the system keeps a topological configuration. There must be a single joint between two bodies. A terminal point or a point of interest such as manipulator hand is called an endpoint. Each body, except body 0, can have one endpoint at maximum. This framework allows to model a system such as e.g. a satellite equipped with multiple manipulator (branches), however in this work the system considered has only one branch.

Force and Torque Inputs

The toolbox allows the force and control inputs to be applied on the centroid of the reference body as $(\mathbf{f_b}, \tau_b)$, on each end-point as $\mathcal{F}_h$, and each joint as τ_m. The definition of these torques and forces are open to user programming, arbitrarily each joint can be either active or passive. The result of the computation of the forward solution of dynamics with numerical integration are the position, velocity and acceleration of the centroid of the reference body, each joint and each endpoint.

Attitude Representation

For the representation of attitude or orientation, the direction cosine matrices are used, coded with a symbol **A**, such that e.g. **A0** is the direction cosine matrix to represent the attitude of the body 0 with respect to the inertial reference frame, and the orientation of the other bodies are assigned to the struct AA containing the respective matrices for each of the remaining bodies. The advantages of direction cosine is singularity free, straightforward derivation of the Roll-Pitch-Yaw angles, Euler angles, or quaternions and clear mathematical relationship with angular velocity.

The Roll-Pitch-Yaw representation is also regularly used in the code flow, the respective angles are defined with a symbol **Q**. For example, in order to express the twisting angles between two coordinate systems, the α (roll) angle is considered around the x axis, β (pitch) around y axis, then γ (yaw) around z axis. The RPY angles are used for the computation of the coordinate transformation matrices in the script 'rpy2dc'.

Direction Cosine and Coordinate Transformation Matrices

The direction cosine matrices $\mathbf{C_i}$ are commonly used to represent attitude or orientation of a body in the field of aerospace engineering. On the other hand, the coordinate transformation matrices with the notation of $\mathbf{^I A_i}$, are commonly used in the field of robotics. These two are eventually the same, under the following transformation:

$$\mathbf{C_i} = \mathbf{^i A_I} \tag{3.1}$$

For the above definition of the link coordinate system, the three axis rotations are required in order to coincide from the the frame of the joint $i-1$ $\{\Sigma_{i-1}\}$) to the frame of the next joint i (Σ_{i-1}). The following equation allows to obtain the necessary transformation between two frames:

$$\begin{aligned} \{\Sigma_i\} &= \mathbf{^i C_{i-1}} \{\Sigma_{i-1}\} \\ &= [C_3(q_i) C_3(\gamma_i) C_2(\beta_i) C_1(\alpha_i)]^T \{\Sigma_{i-1}\} \end{aligned} \tag{3.2}$$

where $C_1(\alpha_i), C_2(\beta_i), C_3(\gamma_i)$ are direction cosine matrices - the coordinate transformations around each principle axis and $C_3(q_i)$ represents the coordinate transformation by angle q_i around joint i; the γ_i angle corresponds to an offset angle and is separated from a net rotation angle q_i. The RPY representation of the attitude of the link 0 is described in the same way, with the roll, pitch, yaw angles only. The direction cosines are redundant way to represent attitude, but its advantage is that the relationship between attitude and angular velocity can be expressed by a simple equation, such that:

$$\dot{\mathbf{C}}_\mathbf{i} = \omega_i \times \mathbf{C_i} \tag{3.3}$$

where $\dot{\mathbf{C}}_\mathbf{i}$ is a time derivative of $\mathbf{C_i}$. This relationship is used for the routine of singularity-free integration from angular velocity to attitude.

Coordinate System

The inertial reference coordinate frame $\{\Sigma_I\}$ considered shall be stationary or linearly moving with constant velocity in the inertial space. It is not physically precise, but in practice the orbital fixed frame is considered as the inertial frame when describing the relative motion of the two spacecraft during the close proximity operations. The relative states of the chaser spacecraft are described in the LVLH frame co-ordinates, centered at the target. The LVLH (Local Vertical Local Horizontal) frame is oriented such that z-axis is Earth-pointing, y-axis is negative to the orbit normal and x-axis is forming a right-handed coordinate system. The orbital motion of the considered satellite is such that the velocity vector is co-linear with the frame x-axis.

Moreover, the moving coordinate frames fixed on each link of the articulated body and on the base are defined. The orientation of principle axes is arbitrary, but it is recommended to orient these axes being parallel to the principle axes of the body inertia. For the assignment of moving coordinate frames on other links, one of the way commonly used in the field of manipulator kinematics is the Denavit-Hartenberg convention. This is known as advantageous in unique assignment of coordinate systems with minimum link parameters. However this convention locates sometimes the coordinate origin away from the location of the actual joint and for the dynamic analysis this is not reasonable. Therefore another set of rules was used, considering both the revolute and prismatic joints:

1. if joint i is revolute:
    - locate the origin of the coordinate frame on joint i and fixed to the link i,
    - set its $z-$axis to coincide with the joint rotation axis,
    - orient its x-axis toward joint $i+1$;
2. if joint i is prismatic:
    - locate the origin of the coordinate frame $\{\Sigma_i\}$ on the point when joint i has zero displacement and fixed to the link $i-1$,
    - set its z-axis to coincide with the joint displacement axis, with the positive direction,
    - orient its x-axis toward joint $i+1$

The system considered in this research work contains only the revolute joints, but the explanation of the system for the prismatic joint was also kept for clarity.

Connection Graph Representation

In order to make practical use of the SpaceDyn functionalities the system consisting of the interconnected bodies must be described in the way that is compatible with the toolbox convention. The toolbox adopts a method from mathematical graph theory, with the simplifications regarding the additional rules on the assignment of link and joint indices in order to allow the unique construction of two matrices - a connection index B and incidence matrices S. The numbering of the bodies is relevant for the correct assignment of the data to the rows of the matrices. In essence, the index number of a link i in the physical connection between the base (link 0) and link j, must be $0 < i < j$. As for the indices numbers of the joints, they begin from 1, and the numbering is such that the joint i that connects the joint j ($i < j$) is numbered j. There is always one single joint interconnecting two links. In this way, for example the joint interconnecting link 0 and link 1 is joint 1, and the joint interconnecting the link 3 and link 4 is joint 4. For further details, the [69] can be referred to.

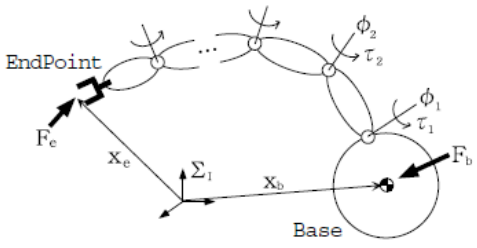


Figure 3.2: Offset-free MPC controller structure;

System Specific Inputs

The following variables are defined with the values specific for the architecture. Initially the system consisting of the s/c body and the 3 links manipulator (2 joints active) was tested, next the system was extended with one more joint and link (4 links, 3 joints active). In the future work the system could be further extended in order to achieve the full architecture, 7 links manipulator, however in this project it was assumed that activating three joints out of seven will already allow to see how the designed controller is functioning, what is its performance and the limitations. Below presented matrices and variables are representing the system consisting of the spacecraft platform body and three links manipulator. With respect to the fully actuated 7 dof manipulator, in this system the three links which are active are the 1st, 2nd and 4th link. The dimensions of the full system are considered, in the way that the first link is as in the full system and its output first joint is active, the second link is as in the full system and its output second joint is active, the third link is effectively the connection of the third and fourth link of the full system, with the original third joint kept inactive and the output fourth joint active, which here is considered as

the third joint. Next the end link consists of rigidly connected 5th, 6th, 7th links with all the respective joints kept inactive.

The data of the RICADOS-specific parameters has been adjusted for the SpaceDyn use. It contains the *Robot* struct of the size of the total number of the bodies - 1 satellite base and 7 manipulator joints. Each struct contains fields with information about the mass, inertia matrix, center of mass and dimensions of the body expressed in the link body reference frame. Therefore the required data to build the model using the SpaceDyn Multibody Dynamics toolbox consists of the following inputs for each body (8):

- Mass - mass of the link
- J - the inertia of a link with respect to the link's center of mass, expressed in the link's body reference frame.
- Com - the location of the center of the mass with respect to the link's body reference frame, expressed in link body reference frame.
- x - the position vector of the following joint with respect to the link's body reference frame, expressed in link body reference frame.
- A - the cosine direction matrix, the attitude of the following joint with respect to the link's body reference frame, expressed in link body reference frame. The z-axis is the revolute joint's axis.

3.3. Dynamics

Dynamics of a rigid body space robot equipped with a n-DOF manipulator:

$$\begin{bmatrix} M_t & M_{tr} & M_{tm} \\ M_{tr}^T & M_r & M_{rm} \\ M_{tm}^T & M_{rm}^T & M_m \end{bmatrix} \begin{pmatrix} \dot{v}_b \\ \dot{\omega}_b \\ \ddot{q} \end{pmatrix} + \begin{bmatrix} c_b \\ c_\omega \\ c_m \end{bmatrix} = \begin{pmatrix} f_b \\ \tau_b \\ \tau_m \end{pmatrix} + \begin{pmatrix} J_b^T \\ J_m^T \end{pmatrix} \mathcal{F}_h \tag{3.4}$$

with the components:

$$M(q) \in \mathbb{R}^{(6+n)\times(6+n)} : \text{inertia matrix,} \tag{3.5}$$

$$c(q, v_b, \omega_b, \dot{q}) \in \mathbb{R}^{(6+n)\times 1} : \text{coriolis/centrigual terms,} \tag{3.6}$$

$$J_b \in \mathbb{R}^{6\times 6}, J_m \in \mathbb{R}^{6\times n} : \text{base and manipulator jacobian} \tag{3.7}$$

and the states and inputs:

$$v_b, \omega_b \in \mathbb{R}^3 : \text{base linear and angular speed} \tag{3.8}$$

$$q \in \mathbb{R}^n : \text{angular position of manipulator joints} \tag{3.9}$$

$$f_b, \tau_B \in \mathbb{R}^3 : \text{base thrust force and torque} \tag{3.10}$$

$$\tau_m \in \mathbb{R}^n : \text{manipulator joints torque} \tag{3.11}$$

$$\mathcal{F}_h \in \mathbb{R}^6 : \text{force/moment on the end effector hand} \tag{3.12}$$

In the frame of this research, one of the project boundaries assumed in order to reduce the complexity is that the end effector joint is not active, hence the forces and torques of the end effector hand, F_h, are not the active inputs of the system. Moreover, the main goal of the controller, as later explained in section 4.4, is to bring the end effector position to the capture point on the target spacecraft. With achieving this goal, the considered phase is finished and the next phase of the capture and stabilization of the stack configuration can begin. These phases are not in the frame of this project, hence the contact dynamics between the end effector and the target spacecraft are not considered and as a result it is assumed that the external forces and torques F_h acting on the end effector are negligible.

3.4. State Space Model

Given all the dynamics formulations and the implemented attitude representation, the state space can be formulated by:

$$\dot{x} = f(x, u) = \begin{pmatrix} \dot{v}_b \\ \dot{\omega}_b \\ \ddot{q} \end{pmatrix} \tag{3.13}$$

$$\mathbf{x} = (x_b, Q0, q, v_b, \omega_b, \dot{q})^T : \text{state vector} \tag{3.14}$$

$$\mathbf{u} = (f_b, \tau_b, \tau_m)^T \in \mathbb{R}^{(6+n)} : \text{input vector} \tag{3.15}$$

where every DoF is specified by a double integrator, except attitude dynamics. The $Q0$ in the state vector is the vector of RPY attitude angles, it is obtained from the direction cosines matrix $A0$ representing the orientation of link 0, with respect to the chosen inertia reference frame. The forward dynamics of the system are computed with integration by Runge-Kutta method with a constant time step. For the update of the attitude, from $A0$ to $A0_n$, the Rodrigues formula for infinitesimal rotation is used. It is noted that $v_b, \omega_b, \dot{v}_b, \dot{\omega}_b$ are defined in the inertia frame.

4

MPC Controller Detailed Design

The following Chapter describes the main core of the project - the design aspects of the MPC controller with the implementation of the free-flying robotic spacecraft dynamics. The theoretical background presented in the previous chapters was the initial baseline for performing an informed trade-off between different model predictive control architectures. Next, the modelling phase was performed. In order to provide the prediction model for the solver, and describe the objective function as a convex programming problem, the general procedure of successive linearization was chosen. It consists of performing linearization of the dynamics around an operating point in each iteration of the simulation, hence effectively the linear time varying model is obtained which is used for the controller. After validation of the nonlinear continuous dynamics the jacobian linearization procedure was applied and validated with respect to the results achieved with the nonlinear definition. In the frame of this work the model is used twice - the original nonlinear model is used as plant for closed-loop testing and validation purposes. The linearized version of the model is integrated as prediction model in the controller. The details of this procedure are described in this Chapter.

Other design choices are further presented. They include the definition of the control objective and the cost function with its primary and secondary goals, constraints handling - the definition and chosen numerical values, choice of the prediction and control horizon, and last but not least the procedure of tuning the weights and scale factors. All these aspects are presented in details in this chapter with the proper explanation and reasoning standing behind each modelling/design decision.

Firstly, in Section 4.1 the MPC design trade-off is presented with the explanation of the six factors considered, the chosen weighting factors and the evaluation of each of the considered option. Following the description of the dynamical system from Chapter 3, the system dimensions i.e. state, control vectors, mpc parameters and the time constants are clearly given in section 4.2. In order to define the MPC optimal control problem the solver must have an insight into the ”future” which is done with the means of the prediction model, its definition is given in Section 4.3. Firstly the procedure of linearization of the nonlinear dynamics model is explained, than the discretization method and finally the generation of the prediction matrices. The control objective of this type of the mission, specifically the capture phase, is explained in Section 4.6. In order to define properly the cost function to achieve the control objective, the reference position of the end effector must be supplied to the solver. The generation of the reference trajectory used in this project is described in Section 4.5. Next, the detailed definition of the cost function is presented in Section 4.6. Each term of the function is given with its mathematical definition and explanation of the method of computation. In Section 4.7 the constraints introduced to the solver are explained and their formulation is given. One of the most important aspects of the design process was proper tuning of penalty weights. The scale factors, weights and overall tuning approach are presented in Section 4.8. Finally, the overview of the optimization problem modelling interface used in the project - YALMIP, is given in Section 4.9 and the graphical representation of the complete feedback loop and necessary pre-computations is shown in Section 4.10.

4.1. Initial Design Trade-off

As presented in the previous chapters, the model predictive control is a very vast topic and hence it comprises a variety of architecture types. In order to assess which is the most promising for the application in this project, the initial trade-off was performed taking into the account six factors with different weighting factors:

- **Simplicity** of the description - there is no direct correlation between ”more complex - worse choice”, however every different aspect of mpc comes with its challenges which might be too complicated for this application while not improving the overall performance.
- **Model fidelity** - all the potential simplifications of the model shall be done in such a way that the representation is accurate enough with respect to the real representation of the plant.
- **Experience** - a previous working experience with any specific type of mpc structure could be potentially beneficial for progressing the work and understanding the complexities in depth. It is the least relevant factor, therefore its weighting factor is the lowest.
- **Optimization** - depending on the definition of the problem one can get different types of optimization problems (convex and non-convex) which can require specific solvers use.
- **Applicability** to RICADOS and free-flying system - the dynamics of the system of concern are very complex, yet the real-time solution requirement necessitates the model to be reduced enough to allow for the application in real mission.
- **Literature review** - the revision of the existing research allowed to have clear view of the challenges identified by other researchers, and have the idea about how the similar problems are already (if) tackled.

while assessing the following three main aspects of the design:

- **Dynamics prediction model** - Linear Time Invariant, Linear Time Varying, Non-Linear
- **Online/Offline computations** - Explicit, Implicit
- **Robust variants** - Tube-based, Min-max

accounting also for the hybrid model for the sake of completeness. In figure 4.1 the Trade-off Table is presented. The table shows an assessment of the aforementioned factors for each of the design aspects as described above. The general comments regarding the awarded scores are briefly summarized and the final scores are presented. The identified best design choice is to have the linear time varying dynamics prediction model, perform computations online - hence, the implicit mpc model is preferred, and for the eventual robust variant the tube-based mpc was identified as more suitable for this application in comparison with min-max architecture. Nevertheless, the robust aspects can be taken into account in many different ways and not only by these two specific design architectures. Therefore less attention is paid to it at this point.

The hybrid model was discarded mainly due to its complexity of the mixed logical and continuous values in the optimization problem that can be solved via mixed integer linear or quadratic programming, thus increases complexity of the optimizer.

overall good, score 3
overall average, score 2
overall bad, score 1
+ / - advantage / disadvantage

		Simplicity/Complexity *weighting factor=1*	Model fidelity *weighting factor=1*	Experience (own/of the team) *weighting factor=0.4*	Optimization *weighting factor=1*	Applicability to RICADOS and free-flying system *weighting factor=1*	Literature *weighting factor=0.6*	Final score
Dynamics prediction model:	**Linear Time Invariant**	+ compared to the nonlinear, the complexity can be overcome via linearization along the prediction trajectory + simplifies the description of the prediction model + more simple than LTV due to the lack of update of the model description in every time step - requires linearization procedure	- linearization around operating point will decrease the system accuracy wrt the real representation of the plant - one configuration should be chosen, such as e.g. the final configuration, as expected when EE reaches the handle on target s/c. Then the matrices values in eom are as for this configuration applied in the dynamics definition for all the control process		+ usually leads to convex optimization problem			14
	Linear Time Varying	+ can accommodate time-varying prediction dynamics, time-varying objectives and constraints - the problem needs to be re-formed at each time step - requires linearizaiton procedure	+ for the linear system, the matrices from state-space representation will vary throughout the prediction horizon hence it allows modelling the dynamics more accurately - linearization around operating point will decrease the system accuracy wrt the real representation of the plant		+ the same optimization procedure as for LTI + available solvers for optimization of LTV	+ the EOM for free-flying space robots, specifically the mass matrix and vector of non-linear velocity terms, are dependent on the configuration (s/c orientation + manipulator joint angles) in current steps, hence the time-varying system could allow increase of the system description accuracy		14.6
	Non-Linear	+ the nonlinear dynamics description as can be directly applied to the NMPC problem - the nonlinear dynamics description is less straightforward than a linear	+ it is the most powerful as it uses the most accurate representation of the plant, the predictions are more accurate compared to linearized		- can lead to non-convex optimization - the resulting computational complexity/demand can be high - challenging to solve in real time + there are available solvers for optimization of nonlinear programs	+ the free-flying system is described by nonlinear dynamics so NMPC depicts the real behaviour of what we have in RICADOS - the system seems to be too complex, it should be greatly simplified for NMPC design	+ proposed in e.g. for free-floating manipulator by Seweryn et al. (2019) however applied to simplified planar system of 5DOF (2 for rob + 3 for s/c) - no NMPC identified for such a complex system as in ricados	11
Online/offline computations:	**Explicit**	- the complexity lies in proper division into critical regions for construction of sub-domains	- for such a complex system, the explicit description of the control problem could result in inadequate reperesentation		+ good if real-time optimization is not possible due to computational limitations - based on parametric programming, increases complexity of optimization solver	- the offline construction of the feedback law scales badly with increasing dimensionality of the problem - required memory for explicit solutions storage	- the majority of explicit MPC applications identified were for smaller systems	9
	Implicit				- the on-line optimization is more computationally demanding than explicit		+ vastly covered in the relevant literature with majority of examples proposing implicit solution	14
Robust variants:	**Tube-based MPC**	- the proper definition of robustly positive invariant sets and tightened constraints are required	+ allows for taking into account additive disturbances contained within a known boundary		+ the regular definition of an optimization problem via e.g. quadratic program		+ vastly covered in the relevant literature + tube-based robust MPC to real-time scenarios was confirmed	14
	Min-max MPC	+ bounded additive disturbances are inside the cost function which objective is to minimize control effor for the maximum disturbance value within the boundaries	+ allows for taking into account additive disturbances contained within a known boundary with their max and min values		- computationally burdening - min-max MPC is rather solved via explicit implementation--> multi parametric programming		+ applications e.g. for fast tracking on a robot manipulator were found - very scarce applications in space systems	11
Others:	**Hybrid**	- mixed logical and continuous values increase complexity of the control problem description			- the resulting optimization prblem is solved through mixed integer linear or quadratic programming, which increases complexity of the optimiser + hybrid toolbox available in matlab and also Gurobi MILP solver		- not many examples related (at least partially) to ricados system were found + implemented e.g. for on/off thrusters attitude control of the upper stage	7.6

Figure 4.1: The MPC choice trade-off table.

4.2. System Dimensions

The dynamics equations 3.4-3.12 described in Chapter 3.3 are implemented for the system consisting of the satellite platform equipped with three link manipulator. In total the system has 9 degrees of freedom - 6 DoFs of the satellite, 3 DoFs of the manipulator joints. In order to describe the system as a state space model the state vector and input vector are defined. They are presented in Section 3.4 and recalled below for clarity:

$$\mathbf{x} = (x_b, Q0, q, v_b, \omega_b, \dot{q})^T \text{ : state vector} \tag{3.14}$$

$$\mathbf{u} = (f_b, \tau_b, \tau_m)^T \in \mathbb{R}^{(6+n)} \text{ : input vector} \tag{3.15}$$

The dimensions of the state, control vectors, and other parameter required for the definition of the optimal control problem in the MPC framework are defined in the struct 'dim' and their values are as follows:

- $nx = 18$ - state vector dimension (*dim.nx*)
- $nu = 9$ - control input vector dimension (*dim.nu*)
- $num\_q = 3$ - number of active manipulator joints (*dim.num_q*)
- $N = 20$ - prediction horizon (*dim.N*)
- $m = 1$ - control horizon (*dim.m*)
- $t = 18001$ - total number of iterations in a simulation loop (*dim.t*);it is equal to length of the *tparam.span* vector

The values nx, nu, num_q are dependent only on the chosen dynamics model. The state vector (eq. 3.14) consists of the positions and velocities of the s/c base, the attitude and angular velocity and manipulator joint angular positions and joint rates. It is clear, that when we consider a three dimensional system, x_b is a vector of length 3, $Q0$ is the vector of length 3, and the same holds for v_b and ω_b. Therefore, the minimum length of the state vector $nx = 12$. When the robotic subsystem is considered, angular position of each joint and the joint rates become part of state vector, and their size depends on the number of active joints. The final model considered in this project has 3 active manipulator joints, therefore q is a vector of length 3 and $\dot{q}$ length is also equal to 3. All these values combined clearly sum up to 18.

The prediction horizon N and control horizon m were chosen such that the solver has enough insight into the plant dynamics - the sampling time of the prediction model is 0.01 sec, therefore with the $N = 20$, the solver has insight into the first 0.2 sec of dynamics evolution which is sufficient considering the MPC frequency of 100 Hz (updates every 0.01s). Similar values were found in the literature e.g. $N = 30$ in [70].

In the list above, some of the values are only exemplary and are the subject of tuning during the controller design process. For example, in the first development phase of the model the satellite platform with a 2 link manipulator was implemented and tested. Thus, the number of active manipulator joints $num\_q = 2$ and the length of state vector $nx = 16$. In the final simulations results presented in this report the dimensions of the dynamics model given here were used. With regards to the prediction and control horizon, the different values were tested. The longer prediction horizon could potentially increase the accuracy of the reference tracking performance. However as the prediction model implemented is a linearized model around an operating point (see Section 4.3) the further away we get from this operating point, in terms of the dynamics propagation, the less accurate is the dynamics description. Therefore, a balance between these two aspects must be found, and it is concluded that much longer prediction horizons do not necessarily improve the controller performance.

Furthermore, the dimensions of the time parameters are defined in the struct 'tparam'. They are setting up the total time of the simulation and the value of the sampling time which affects the quality of the prediction. All the time constants are given in the seconds unit [sec]:

- $T_{sample} = 0.01$ - sampling time (*tparam.sample*)
- $T_{end} = 180$ - total time of the simulation (*tparam.end*)
- $T_{span} = [0 : T_{sample} : T_{end}]$ - vector of all the time steps (*tparam.span*)
- $T_{simsteps} = T_{end}/T_{sample} = 18000$ - total number of simulation steps (*tparam.simsteps*)

The sampling time T_{sample} is used for the discretization of the continuous dynamics. The total time of the simulation T_{end} is setting the total length of the maneuver such that the control loop simulation is run for this specified time and the final position of the end effector shall be achieved by the end of the simulation. The reference set of end-effector positions are computed for each simulation step from the span T_{span}, hence the look-up table with the reference values to be tracked has the length of the total number of simulation steps $T_{simsteps}$. The size of the time vector is also used to pre-assign dimensions of the matrices that will store the values of the state vector and control input sequence from all the simulation.

4.3. Prediction Model

The proper definition of the prediction model is necessary for the construction of the model predictive control problem. The optimization solver has knowledge of the system coming only from the prediction model, therefore

it is important to ensure that the valid information about the plant dynamics is passed to the solver. Depending on the characteristics of dynamics that are captured by the prediction model, the definition of the optimal control problem results convex or non-convex. For example if the prediction model is nonlinear, the overall definition of an optimal control problem will be non-convex, hence the non-convex optimization problem then would need to be addressed by the solver. For this reason it is important to keep in mind that the plant dynamics can be adequately simplified in order to provide a prediction model that could lead to the definition of a convex optimization problem.

In general, the dynamics model used for prediction inside MPC has to fulfill two basic requirements:

1. It shall represent the spacecraft dynamics (including wheel-soil interaction with sufficient accuracy.
2. It shall provide a sufficient computational performance in order to open up a prospective for use within real-time applications like the control of an on-orbit servicing spacecraft in real mission.

Accordingly, the goal was to provide a robotic spacecraft model with the minimum number of required states and with representative dynamics. The nonlinear dynamics model of the plant described in the previous chapter idealistically shall be equal to the model used in the prediction step in the controller. However with the use of non-linear prediction model the cost function becomes much more complex and possibly non-convex. The solution to a non-convex optimization problem is not a global solution, therefore defining the optimization problem is non-convex should be rather avoided. Moreover the introduction of nonlinearities in the optimization problem increases the computational burden. For these reasons the chosen strategy is to introduce the Nonlinear MPC (NMPC) with the procedure of successive linearization around an operating point. Certainly this procedure allows to decrease the computational burden related to solving an optimization problem, as the linear model allows to introduce it as a linear or quadratic program, which has a single global solution. This approach has also disadvantages which clearly is the mismatch between the real nonlinear model and its linearized version. Using only one linear model for all the capture phase is not a good design choice as with every motion of the spacecraft the real system is less and less represented by the linearized model and hence using it as prediction model would not yield the required performance of the MPC controller. The idea is to perform linearization of the nonlinear model in every control loop around the operating point being the current numerical value of the state vector, as determined by the spacecraft sensors. This approach is further described in next sections. The linearization procedure is explained in Section 4.3.1, next the formula to discretize the continuous model is given in Section 4.3.2 and finally, the generation of prediction matrices used in the OCP definition is given in Section 4.3.3.

4.3.1. Linearization of the nonlinear dynamics model

The linearization of the nonlinear dynamics model is done with the script *'dynmodel_linearization.m'*. The Jacobian linearization is performed around the current operating point (x_0, u_0) with the method of small perturbations. The matrices that are computed with the script are state space matrix A_{ct} and output matrix B_{ct} as in the linear state space model formalism. The corresponding linear, continuous-time state space model (index *ct*) can be defined at any operating point by:

$$\dot{\tilde{x}} = A_{ct}\tilde{x} + B_{ct}\tilde{u} \tag{4.1}$$

with

- $\tilde{x} =$ small state deviation from the operating point x_0,
- $\tilde{u} =$ small input vector deviation from the operating point u_0,
- $A_{ct} =$ state matrix of linear continuous-time state space model at operating point (x_0, u_0),
- $B_{ct} =$ input matrix of linear continuous-time state space model at operating point (x_0, u_0).

Firstly the nominal value of the operating point is used - it is injected into the nonlinear dynamics model (script *'f_dyn'*) which is described in the Chapter 3.

$$[vd01, wd01, qdd1] = f\_dyn(R0, A0, v0, w0, q, qd, F0, T0, Fe, Te, tau) \tag{4.2}$$

For the implementation of this procedure the operating point at time step *k* was selected as the latest configuration at time step *k-1*. Thus, we apply:

$$\begin{aligned} u_{op}(k) &= u(k-1) \\ x_{op}(k) &= x(k-1) \\ \tilde{u}(k) &= zeros(nu, 1) \\ \tilde{x}(k) &= zeros(nx, 1) \end{aligned} \tag{4.3}$$

The matrices A_{ct} and B_{ct} are numerically computed Jacobians:

$$A_{ct} = \frac{\partial f\left(x_{op}, u_{op}\right)}{\partial x}; \qquad B_{ct} = \frac{\partial f\left(x_{op}, u_{op}\right)}{\partial u} \tag{4.4}$$

Firstly the more conservative approach towards linearization was applied by performing the differentiation of the dynamics function with the use of matlab symbolic toolbox. Due to the computational burden this solution was compared with a more simple approach - the Jacobian matrices are obtained using perturbations only. The perturbation, of the value of 1e-4, is added successively to every term from the state vector and control input vector of the operating point, next the values are injected into the nonlinear dynamics model and as a result the vector of the acceleration $f_{perturb}$ is obtained. The state matrix A_{ct} is obtained

$$f_{op} = [vd01; wd01; qdd1]; \qquad f_{perturb} = [vd02; wd02; qdd2];$$

$$A_{ct}(\tfrac{nx}{2} + 1 : end, ix) = \tfrac{f_{perturb} - f_{op}}{\Delta x(ix)};$$

where $ix = 1 : nx$ and the $\Delta x(ix) = 1e - 4$ is the value of the perturbation of the ix^{th} term of the state vector. The output matrix B_{ct} is obtained in the similar manner, considering the dimensions of the control input vector $iu = 1 : nu$ and the perturbations of the control input values with respect to the value of an operating point, $\Delta u(iu) = 1e - 4$.

$$B_{ct}(\tfrac{nx}{2} + 1 : end, iu) = \tfrac{f_{perturb} - f_{op}}{\Delta u(iu)};$$

It must be underlined that the continuous linear state space model obtained with this linearization procedure is valid only in the vicinity of the operating point.

4.3.2. Discretization of the linear continuous model

The linear discrete-time state space model represented by the state matrix A_{dt} and input matrix B_{dt} is derived from the respective matrices of continuous-time state space model A_{ct} and B_{ct}. The transformation of a continuous-time system to a discrete time system is performed by the following operations:

$$A_{dt} = e^{A_{ct}T_s}; \qquad B_{dt} = A_{ct}^{-1}(e^{A_{ct}T_s} - I)B_{ct} \tag{4.5}$$

Both operations apply the matrix exponential $e^{A_{ct}T_s}$ with the sampling time T_s. The discrete state space form is finally:

$$\tilde{x}(k+1) = A_{dt}\tilde{x}(k) + B_{dt}\tilde{u}(k) \tag{4.6}$$

This discrete model of the system is implemented as a prediction model, the state vector is propagated for all the prediction horizon using the values of the matrices A_{dt} and B_{dt} in current simulation step.

4.3.3. Prediction matrices generation

The update of the prediction matrices used in the optimizer is generated with the script *'predmodgen.m'*. The input to the function is the linearized discretized LTI model and the struct containing system dimensions (*dim*). The below matrix A, B are referring to the matrices A_{dt}, B_{dt} obtained in the previous subsection. The state solution is as follows:

$$\begin{aligned} \tilde{x}(k) &= A^k x_0 + \sum_{j=0}^{k-1} A^{k-j-1} B u(j) \\ &= A^k x_0 + \begin{bmatrix} B & AB & \dots & A^{k-1}B \end{bmatrix} \begin{bmatrix} u(k-1) \\ u(k-2) \\ \dots \\ u(0) \end{bmatrix} \end{aligned} \tag{4.7}$$

In order to find a control solution with the model predictive control architecture, the evolution of state vector within all the prediction horizon must be available when the optimizer is being run. The size of the prediction matrices depends on the dimensions of the problem, the state space dimension, prediction horizon *N* and control horizon *m*. The sequence of the state vectors and control inputs in further notation is marked with the bold sign.

$$\tilde{\mathbf{x}}_{\mathbf{N}} = T \cdot x_0 + S \cdot \mathbf{u}_{\mathbf{m}} \tag{4.8}$$

where

- $\tilde{\mathbf{x}}_{\mathbf{N}}$ is a sequence of *N* state solutions deviations from the operating point $x_o p$ starting from initial state x_0, subject to sequence of control inputs,
- $\mathbf{u}_{\mathbf{m}}$ is a sequence of *m* control inputs;
- T is a prediction matrix from initial state, it includes state matrices A_{dt}
- S is a prediction matrix from control input sequence, it includes state matrices A_{dt} and input matrices B_{dt}

and the sequence of the vector is:

$$\tilde{\mathbf{x}}_{\mathbf{N}} = \begin{bmatrix} \tilde{x}(1) \\ \vdots \\ \tilde{x}(N) \end{bmatrix} \qquad \mathbf{u}_{\mathbf{m}} = \begin{bmatrix} u(1) \\ \vdots \\ u(m) \end{bmatrix},$$

Often in the literature, the size of the prediction horizon (N) and the control horizon (m) is equal, and then the u_m is signed as u_N. In this project it is considered advantageous to have shorter control horizon with respect to the prediction horizon, hence these variables are represented by different notation. In the script for the prediction model generation, the following structures of the matrices and vectors is applied:

$$T = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} \qquad S = \begin{bmatrix} B & 0 & 0 & 0 \\ BA & B & 0 & 0 \\ \vdots & & \ddots & \\ BA^{N-1} & BA^{N-2} & \dots & BA^{N-m} \end{bmatrix}$$

In the script the matrices are saved in the struct *predmodgen* and recalled respectively *predmod.T* and *predmod.S*. It must be again underlined that this prediction model operates not on the global values, but the local 'tilde' values which are deviations from the operating point.

4.4. Control Objective

The objective of the controller is to perform a safe capture of the target. This objective is broken down into sub-objectives, each having relevant meaning that shall be introduced to the solver. The components are mathematically described and put together in the optimization objective function, referred to as cost function.

The trajectory of the end-effector shall follow the reference trajectory provided from the guidance block. The end-effector shall achieve a very good final performance in order to ensure safety capture of the target. The role of the control system is to find the appropriate sequence of control inputs for the available actuators. It shall be done in a way that the constraints on the upper and lower input bounds are considered, the safety aspects of the mission are respected and the tracking error between the reference and current end-effector position is minimized. Moreover, as in every space mission, it is desirable to minimize the fuel expenditure and power consumption. The system shall be controlled in the most optimal way, such that all these aspects are considered and the required change of its kinetic energy is minimal.

The detailed definition of the mathematical description of the cost function is given in the next sections. The design process of this function consists of variety of aspects such as validation of the definition of each term, validation of the linearization procedures, tuning of the weights and scaling factors. The detailed definition of the cost function is presented in the subsection 4.6.

4.5. Reference Trajectory

Considering the whole GNC architecture, the reference trajectory definition is a task of the guidance block, it is doing the determination of the path from the satellite's current position to a target position. Usually determination of the path and its evolution in time, hence the trajectory, is computed as a solution of an optimization problem considering the system dynamics and constraints [71]. Next, the information is passed to the control block, which shall track an optimal reference trajectory decreasing the error between the current position of the vehicle (as determined by the control block) and the reference value, generated by the guidance subsystem a priori.

In this work, the focus was solely on the controller development, the scope of the work does not include a full GNC loop design, therefore the generation of the trajectory is simplified. The reference trajectory that shall be tracked is the end-effector trajectory, the evolution of its position is defined in the task space. It is assumed that the given position of the EE is specifically the position of its TCP and no orientation is considered. Initially, the

reference position was defined in the joint space, such that the controller objective was to follow the joint positions. This is a very straightforward approach, which can be valid for a system with small number of DoFs, however it takes away freedom of the configuration choice and hence is limiting the solution space of the optimization problem (the controller cost function). Once the tracking problem described in the joint-space was successfully solved the current definition of the problem was validated. Then, in next step, the reference trajectory was described in the cartesian space, as follows.

1. The initial position of the end-effector EE_0 in the LVLH frame is derived from the initial state vector of the chaser s/c and its known geometry via forward kinematics algorithm.
2. The coordinates of the capture point are known, they are defined in the LVLH frame of the target spacecraft. It is assumed that the relative navigation of the chaser s/c during duration of all the maneuver allows the estimation of the position of this point with perfect precision. Therefore, the relative distance between the current end-effector position and the target position is known. This final position of the end effector, EE_f, is defined offline and is kept constant during the simulation.
3. The reference trajectory is generated prior to the control feedback loop as a look-up table of the size equal to the number of the simulation steps ($T_{simsteps}$), such that there is a reference value of the EE position for each step of the simulation. The interpolation between the initial and final EE position can be computed in different ways. The two algorithms used are the linear and polynomial interpolation.

Linear

The linear interpolation between the initial end effector position EE_0 and the final position EE_f to be achieved by the end of the maneuver is determined in the following way. The difference between the initial and the final end-effector position is divided by the total number of simulation steps in order to obtain the value of a constant increment ΔEE. The look-up table containing the reference positions in every simulation time step is constructed by adding the increment value to the reference position from the previous step, starting with the initial position EE_0, as in the equation below.

$$\begin{aligned} \Delta EE &= \frac{EE_f - EE_0}{T_{simsteps}} \\ EE(k) &= EE(k-1) + \Delta EE \end{aligned} \tag{4.9}$$

This procedure is performed separately for each coordinate x, y, z of the end-effector position. It is clear that in this way only geometry of the trajectory is taken into account, the equidistant separation results theoretically in a constant velocity.

Polynomial function of 5^{th} degree

The 5th order polynomial interpolation of a trajectory is performed based on initial conditions and final conditions of time, position, velocity and acceleration. The input to the function is the sampling time interval $T_{simsteps}$, the initial and final position of end effector EE_0 and EE_f, both the initial and final velocity and acceleration are equal to zero. This is due to the fact that it is assumed that the maneuver starts with relative velocity between two spacecraft equal to zero and it is expected that by the end of the maneuver the motion should be fully decelerated and the final position kept stationary. These six boundary conditions are used to construct the coefficients of a 5th order polynomial function in every time step of the objective look-up table.

$$EE(k) = EE(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{4.10}$$

The coefficients are computed in a loop depending on the time position $T = t(k) - t_0$ and its fraction of the total maneuver time. This method is applied separately for each coordinate x, y, z of the end-effector position.

4.6. Cost Function

The definition of the cost function is presented in this section. For the clarity we can distinguish between primary goals and secondary goals, they are introduced in the objective function as separate terms. The function consists of the summation of all the terms. In order to define the convex objective function for the present nonlinear problem, the successive convexification approach is applied. It treats of linearizing all the nonlinear terms in every iteration step in order to enable a construction of the convex quadratic programming problem. The linearization of the dynamics is performed in the run time, in each simulation iteration, as explained in Section 4.3. The linearization is performed around the operating point (x_{op}, u_{op}) which is updated in every iteration with the value of the state vector and the applied control inputs vector from the previous step. Therefore the optimization problem originally

nonlinear due to the nonlinear dynamics, can be expressed as a quadratic program thanks to the successive linearization.

In terms of reference and cost function we can distinguish between primary goals which are related to the trajectory of the manipulator's end effector and chaser's rotational rate and secondary goals, which take the system's energy minimization and geometrical constraints into account.

Primary Goals

The primary optimization goal of the controller objective function of the robotic chaser s/c is optimal reference position tracking of the end effector. The reference data input is the trajectory output form the guidance block, as defined in the section 4.5. As the dynamics are defined in the joint space, the state vector includes the angular position of the manipulator joints (q), the mapping between the joint space and the manipulator cartesian space must be done such that the end effector position can be defined in (x, y, z) coordinates in LVLH frame during the all motion evolution. To perform this mapping the forward kinematics (FK) algorithm is implemented. The FK algorithm takes as input the current state vector - the position vector of the spacecraft base x_b with respect to LVLH frame, its orientation Q_0, and the angular positions of the mainpulator joints q. With the known geometry of the manipulator, these inputs are sufficient to compute the current position of the end effector in cartesian space. The FK function is nonlinear, in the objective function its linear mapping was implemented, such that the jacobian linearization procedure around an operating point is computed in every simulation iteration. The obtained Jacobian matrix $C_{FK} = J_f(x_{op})$ enables the linear mapping between the state vector x ($x = x_{op} + \Delta x$), and the function $f_{FK}(x)$:

$$EE_{current\_position} = [EE_x, EE_y, EE_z] = f_{FK}(x) \approx EE_{op} + C_{FK} \cdot \Delta x \tag{4.11}$$

The validation of the linearization of the forward kinematics algorithm is presented in the appendix B. The quadratic term representing this goal is finally formulated, as:

$$\|Q_{EE_{position}}(\Delta EE_{ref} - C_{FK} \cdot \Delta x_i)\|_2 \tag{4.12}$$

where $Q_{ee_{position}}$ is the weight matrix, the penalty applied to the non-zero value of the term. The explanation of the weight matrices construction and tuning is given in Section 4.8.2.

Secondary Goals

In order to drive the system in a desired direction, ensure the feasible configuration and attitude such that the maneuver is performed safely with the navigation aid whereas minimizing the effort - fuel and power, other aspects must be included in the cost function. The terms addressing these aspects are added to the definition of the function, they are presented in the next subsections.

Energy minimization

The minimization of the energy of the system, such that the goal is achieved with the minimum energy expenditure. It is simply the penalty on the velocity squared, hence representing the minimization of the kinetic energy, defined as the quadratic term The v_i as in the equation below, is the vector of velocities of each of the body of the system. It includes the linear velocity of the s/c platform, the angular velocity of it, and angular velocity of the manipulator joints. The weighting matrix Q_{energy} is applied, it has the weight values on its diagonal with all the other terms equal to zero. The size of the matrix depends on the size of the problem, in case of the considered multi-body system consisting of 9 degrees of freedom (3 translational of the base, 3 rotational of the base, 3 rotational of the manipulator joints), the matrix will have size of 9×9 and the penalty is applied on every velocity (linear or angular).

$$\|Q_{energy} v_i\|_2 = v_i' \cdot Q_{energy} \cdot v_i \tag{4.13}$$

Control inputs cost minimization

The minimization of the control effort is included in the cost function as the term penalizing control action of every active joint. The penalty matrix R is a matrix with weight values on its diagonal and the other elements equal to zero. As there are 9 degrees of freedom in the considered system, the matrix R has size of 9×9.

$$\|Ru_i\|_2 = u_i' \cdot Ru_i \tag{4.14}$$

Field of View (FoV)

The goal is to ensure that the navigation camera shall be pointing always into the marker aid on the target spacecraft. If the maneuver starts with a non-zero attitude of the chaser s/c with respect to the target, its angular position

shall be reached such that the navigation marker aid is in the field of view of the camera and can be continuously used as a reference for the spacecraft navigation. In order to define the FoV term in the cost function, the following procedure is applied. Firstly two vectors are defined:

- V1: Vector passing through the navigation camera of the chaser and passing through the c.o.m of the base, co-linear with the x-axis of the chaser base as defined in the body coordinate frame.
- V2: Vector passing through the marker aid, defined as the point in the target spacecraft co-linear with the x-axis of the LVLH frame, and passing through the navigation camera of the chaser.

In Figure 4.2 these vectors are clearly shown. The vectors are normalized and then the dot product between them is computed:

$$\vec{V1} \cdot \vec{V2} = |V1||V2|cos(\alpha) \tag{4.15}$$

which for the unit vectors becomes $\vec{V1} \cdot \vec{V2} = cos(\alpha)$.

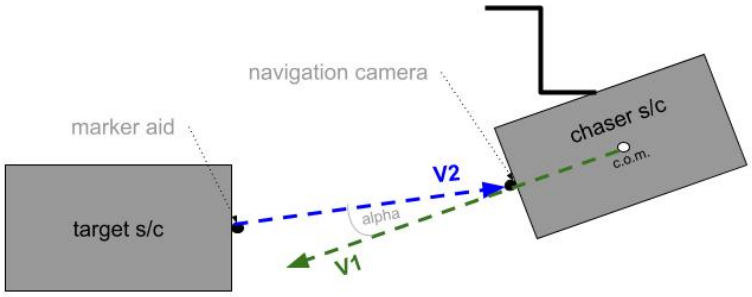


Figure 4.2: Definition of the FoV penalty term. Depiction of the system in 2D for clarity purposes only; the angle alpha considered in the system is defined between two vectors in 3D.

The goal is to make these two vectors co-linear, which would mean that the camera is perfectly pointed to the aid marker. However, in reality it is enough that the marker aid remains in the field of view of the camera, therefore the goal is to minimize the difference: $cos(0) - cos(\alpha) = 1 - cos(\alpha) = 1 - f(x)$. The nonlinear function is dependent on the current state vector, specifically the base position and orientation. In order to apply it into the objective cost function, it is linearized around an operating point in every simulation iteration, such that an output matrix C is obtained which allows for the linear mapping between the state vector ($x = x_{op} + \Delta x$), and the function f(x):

$$f(x) \approx FoVcos_{op} + C_{Fov} \cdot \Delta x \tag{4.16}$$

where $FoVcos_{op}$ is the value of the $cos(\alpha)$ at the oeprating point x_{op}.

$$1 - f(x) \approx 1 - (FoVcos_{op} + C_{FoV} \cdot \Delta x) \tag{4.17}$$

Finally the quadratic term representing this goal is formulated, as:

$$\|Q_{FoV}(1 - cos(\alpha))\|_2 \approx \|Q_{FoV}(1 - (FoVcos_{op} + C_{FoV} \cdot \Delta x))\|_2 \tag{4.18}$$

where Q_{FoV} is the weight matrix, the penalty applied to the non-zero value of the term. The explanation of the weight matrices construction and tuning is given in Section 4.8.2.

Complete definition of the cost function

Finally, all the terms are brought together and the complete definition of the cost function as a quadratic function is obtained, as shown below.

$$\min_{x_i, u_i} \sum_{i=1}^{N-1} \|Q_{EE_{position}}(\Delta EE_{ref} - C_{FK} \cdot \Delta x_i)\|_2 + \|Q_{energy} v_i\|_2 + \|Q_{FoV}(1 - (FoVcos_{op} + C_{FoV} \cdot \Delta x_i))\|_2 + \|Ru_i\|_2 \tag{4.19a}$$

s.t. $x_1 = x_{init}$ (4.19b)

$x_{i+1} = A(i)x_i + B(i)u_i$, for i = 1...N (4.19c)

$u_{min} \leq u_i \leq u_{max}$, for i = 1...N (4.19d)

4.7. Constraints Handling

The most advantageous quality of MPC compared to other control architecture is the implementation of the constraints definition. They allow to ensure that the control solution found by an optimizer is within the real limits. In design of this controller two main constraints are taken into account:

1. The limit on the magnitude of the control signals' value - the value of the maximum and minimum thrust force, reaction wheel torques and manipulator joints torques.
2. The range of the joints motion - the maximum and minimum allowable position angle of each joint is related to the mechanical limits of the configuration.
3. The rate of change of the control input - what is the maximum value in both positive and negative direction, by which the control signal can change in one step.
4. Approach cone - the geometrical limit on the position of the spacecraft base while approaching the target.

The mathematical definition of both constraints is described in next section. The global values of the magnitude of control signals $[N, Nm]$ and the range of joints position angles $[deg]$ are based on the literature of the similar systems.

As all the terms from the cost function are described based on the linearized model, the control input which is the optimization variable to be found, is effectively the $\tilde{u}$ - deviation from the operating point u_{op}. Therefore all the constraints must be updated in order to be limiting the $tilda$ values, and not the global values. The procedure to do it is presented in the next subsections.

All the constraints are translated into an inequality equation that is injected into the optimization software:

$$G * x_{op} + H * \tilde{u} + F * [u_{op}; \tilde{u}] + \Psi \leq 0 \tag{4.20}$$

such that the matrix G corresponds to the operating point x_{op}, matrix H corresponds to the optimization variable $\tilde{u}$, matrix F corresponds to the rate of change of the control input and Φ is the matrix of constants related to the numerical values of the constraints.

Input Magnitude

The servicing spacecraft as implemented in RICADOS can be simplified to a cubic structure controlled by actuators suite consisting of 24 thrusters and reaction wheels.The 3 thrusters positioned at each corner of the cubic base in the way to enable thrust force in all three orthogonal directions, each of them providing max thrust of 10N yielding max thrust in one direction of 40N. The values of the maximum torques of the manipulator revolute joints are given in [67], ranging from 30Nm for the joint 7,6 and 100Nm for joints 5,4,3 and 200Nm for joints 2 and 1.

- F_{ub} = $\begin{bmatrix}40 & 40 & 40\end{bmatrix}$ N
- T_{ub} = $\begin{bmatrix}40 & 40 & 40\end{bmatrix}$ Nm
- τ_{ub} = $\begin{bmatrix}200 & 200 & 100\end{bmatrix}$ Nm

The upper (ub) and lower (lb) global bound of the control signals are represented by the following vectors:

$$u_{ub} = \begin{bmatrix}40 & 40 & 40 & 1 & 1 & 1 & 200 & 200 & 100\end{bmatrix}';$$
$$u_{lb} = \begin{bmatrix}-40 & -40 & -40 & -1 & -1 & -1 & -200 & -200 & -100\end{bmatrix}';$$

In general these vectors can be represented as the inequality equation as:

$$\begin{cases} u \leq u_{ub} \\ u \geq u_{lb} \end{cases} \tag{4.21}$$

which is translated into:

$$\begin{cases} u \leq u_{ub} \\ -u \leq -u_{lb} \end{cases} \tag{4.22}$$

The equations 4.21 and 4.22 are still expressed on the global control inputs and its values, they need to be translated in the domain of deviation values as follow:

$$\begin{cases} \tilde{u}_{ub} = u_{ub} - u_{op} \\ \tilde{u}_{lb} = u_{lb} - u_{op} \end{cases} \tag{4.23}$$

$$\begin{cases} \tilde{u} \leq u_{ub} - u_{op} \\ -\tilde{u} \leq -u_{lb} + u_{op} \end{cases} \tag{4.24}$$

The matrix H corresponding to the $\tilde{u}$ depends on the dimension of the problem - the control horizon of the MPC (m) and the size of the control input vector (nu). The matrix H is constructed as follows:

$$E_{(nu\cdot m)\times(nu\cdot m)} = \begin{bmatrix} I_{nu} & 0 & \dots & 0 \\ 0 & I_{nu} & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & I_{nu} \end{bmatrix} \tag{4.25}$$

The identity matrix, with ones on its diagonal, is of the same size as the control vector nu, the E matrix size is the diagonal matrix with the

$$H_{(2\cdot nu\cdot m)\times(nu\cdot m)} = \begin{bmatrix} E \\ -E \end{bmatrix} \tag{4.26}$$

The matrix Ψ contains the numerical values of the upper and lower bound of the $\tilde{u}$ which are equal to the right side of the inequalities from the 4.24. As the 4.20 is inequality equation with the zeros on the right side, and the matrices on the left side, it is clear that Ψ matrix equals to the values from the right side of the 4.24 multiplied by -1.

$$\Psi_{(2\cdot nu\cdot m)\times(1)} = \begin{bmatrix} -(u_{ub} - u_{op}) \\ \vdots \\ -(u_{ub} - u_{op}) \\ -(-u_{lb} + u_{op}) \\ \vdots \\ -(-u_{lb} + u_{op}) \end{bmatrix} \tag{4.27}$$

It is clear that in this constraint the matrices G and F are not needed, therefore they are matrices containing only zero components of the dimensions depending on the problem dimensions.

$$G = \mathbf{0}_{(2\cdot nu\cdot m)\times(nx)} \tag{4.28}$$

$$F = \mathbf{0}_{(2\cdot nu\cdot m)\times((m+1)\cdot nu)} \tag{4.29}$$

Joints Range

The admissible range of rotational joints is $\pm 170^o$ for joints 1,3,5 and 7 and $\pm 120^o$ for the remaining joints [66]. Depending on how the multi-body system is modelled, according to the description in the Chapter 3, the numerical values of the joints range are respectively extracted from these vectors. As the manipulator was modelled with the joint 1,2 and 4 being active, hence the upper and lower bound on the range of the possible joint motion for all the joints in the KUKA LWR robot consisting of 7 revolute joints is represented by the following vectors:

$$q_{ub} = \begin{bmatrix} 170 & 120 & 120 \end{bmatrix} \text{ [deg]};$$
$$q_{lb} = \begin{bmatrix} -170 & -120 & -120 \end{bmatrix} \text{ [deg]};$$

In general these vectors can be represented as the inequality equation as:

$$\begin{cases} q \leq q_{ub} \\ q \geq q_{lb} \end{cases} \tag{4.30}$$

which is translated into:

$$\begin{cases} q \leq q_{ub} \\ -q \leq -q_{lb} \end{cases} \tag{4.31}$$

The equations 4.30 and 4.31 are still expressed on the global motion ranges and its values, they need to be translated in the domain of deviation values as follow:

$$\begin{cases} \tilde{q}_{ub} = q_{ub} - q_{op} \\ \tilde{q}_{lb} = q_{lb} - q_{op} \end{cases} \tag{4.32}$$

$$\begin{cases} \tilde{q} \leq q_{ub} - q_{op} \\ -\tilde{q} \leq -q_{lb} + q_{op} \end{cases} \tag{4.33}$$

The angular position of the joints $q = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}$ is contained in the state vector x. It is clear that $q = x(7:9)$ remains valid for all the control loop execution and therefore the constraint on the joints motion range can be expressed as the function of the prediction model, state vector in the operating point x_0 and the sequence of control inputs $\mathbf{u_m}$, as in the equation 4.8. In order to extract the q from the state vector, the following matrix M is constructed:

$$M_{(num\_q)\times(nx)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.34}$$

$$q = M \cdot x \tag{4.35}$$

Accounting for all the joint positions within the prediction horizon N, the matrix M is extended as follows:

$$MN_{(N\cdot num\_q)\times(N\cdot nx)} = \begin{bmatrix} M & 0 & ... & 0 \\ 0 & M & ... & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & M \end{bmatrix} \tag{4.36}$$

which allows writing the sequence of the joints positions for all the prediction horizon as the function of the sequence of the state vectors:

$$\tilde{\mathbf{q}} = MN \cdot \tilde{\mathbf{x}}_{\mathbf{N}} \tag{4.37}$$

Injecting the Eqn. 4.8 into Eqn. 4.35 one gets:

$$\tilde{\mathbf{q}} = MN \cdot (T \cdot x_0 + S \cdot \mathbf{u_m}) \tag{4.38}$$

In order to write the joints range constraint the equation 4.38 is combined with the equation 4.33, such that:

$$\begin{cases} MN \cdot (T \cdot x_0 + S \cdot \mathbf{u_m}) \leq \begin{bmatrix} q_{ub} - q_{op} \\ \vdots \\ q_{ub} - q_{op} \end{bmatrix}_{(N\cdot num\_q)} \\ -MN \cdot (T \cdot x_0 + S \cdot \mathbf{u_m}) \leq \begin{bmatrix} -q_{lb} + q_{op} \\ \vdots \\ -q_{lb} + q_{op} \end{bmatrix}_{(N\cdot num\_q)} \end{cases} \tag{4.39}$$

Now it is clear that the joints range constraint can be defined according to 4.20, where matrix G, H and Ψ correspond respectively to:

$$G_{(2\cdot N\cdot num\_q)\times(2\cdot nx)} = \begin{bmatrix} MN \cdot T \\ -MN \cdot T \end{bmatrix} \tag{4.40}$$

$$H_{(2\cdot N\cdot num\_q)\times(2\cdot nu\cdot m)} = \begin{bmatrix} MN \cdot S \\ -MN \cdot S \end{bmatrix} \tag{4.41}$$

$$\Psi_{(2\cdot N\cdot num\_q)\times(1)} = \begin{bmatrix} -(q_{ub} - q_{op}) \\ \vdots \\ -(q_{ub} - q_{op}) \\ -(-q_{lb} + q_{op}) \\ \vdots \\ -(-q_{lb} + q_{op}) \end{bmatrix} \tag{4.42}$$

It is clear that the matrix F remains equal to zero:

$$F = \mathbf{0}_{(2\cdot N\cdot num\_q)\times((m+1)\cdot nu)} \tag{4.43}$$

Approach Corridor

In order to approach the target satellite in a safe way, the approach corridor is defined as the geometrical constraint on the position of the chaser spacecraft base center of mass. The approach corridor is defined with respect to the center of mass of the target satellite, that is considered as the tip of the cone. In this point the target satellite body coordinate frame has its apex point. The cone axis is co-linear with the x-axis of the body reference frame, and the cross-section of the cone is comprised fully in the y-z plane. The cone angle α is a parameter that can be set, it is assumed that the value $\alpha = 15\,\mathrm{deg}$ is a good choice to create the approach cone for the close proximity approach and capture. The radius of the cone section is a function of the distance from the beginning of the coordinate frame as shown in figure 4.3. In mathematical terms it is defined as the linear function of x with the slope of the function equal to the tangent of the cone angle:

$$R_{cone} = tg(10^o) \cdot x \tag{4.44}$$

The definition of the constraint in the most straightforward way is the the quadratic inequality. The radius of the cone section is a function of the x position in current step, and the position of the chaser coordinates in the current step shall be inside the circle.

$$r_k^2 \geq y_k^2 + z_k^2 \longrightarrow (tg(\alpha) \cdot x_k)^2 \geq y_k^2 + z_k^2 \tag{4.45}$$

As it is clearly seen, this equation results in the nonlinear inequality constraint. In order to comply with the chosen approach, to define the OCP as a quadratic programming problem, there can be only the linear inequalities. For this reason, the inequality in equation 4.45 is approximated with the set of linear inequalities. This constraint has been expressed by a polyhedral approximation of the three-dimensional cone shape into a decahedron. [72] The cone is divided into 10 planes such that the inequality constraint can be expressed for all planes of a tilt angle in the y-z plane equal to $(k \cdot \beta)$ with $k \in \{0, 9\}$ and $\beta = 36\,\mathrm{deg}$ and a tilt angle α in the x-z plane. Effectively, the spacecraft is not penalized if its position is within the decahedron figure with the cross sections shown in figure 4.3.

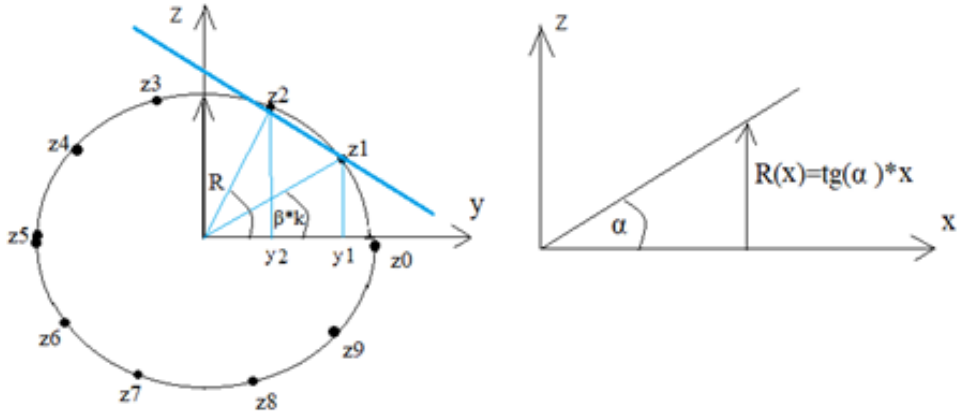


Figure 4.3: Depiction of the polyhedral approximation of a cone in the approach corridor constraint.

4.8. Tuning

The process of controller tuning concerns determining values of the controller parameters such that it is possible to achieve a desired output. Good balance between the numerical values of the parameters shall allow the optimization of a process, such that all the aspects of the objective function are well understood and treated by the solver. In order to achieve this balance, each term of the cost function, as explained in Section 4.6, preferably shall be unitless and in the same order of magnitude. This can be achieved by introducing scaling factors. They are chosen to represent a relevant physical value, such as e.g. the approximate span of a variable. The scale factors are introduced to the cost function as divider of each term. The process is explained more in details in the next subsection 4.8.1. Once the terms of the cost function are scaled and hence they have the same order of magnitude, next step is to tune the values of the weight matrices. The balance between the penalty values must be found such that the priority is given to minimizing the error of the reference tracking objective, and all the remaining secondary goals are adequately tuned in order to represent the importance of each of the term. This process is described in the subsection 4.8.2. Last but not least, it must be said, that the process of controller tuning is not trivial, and is a very variable process as it depends on many factors including the dynamic behaviour of the controlled process, the definition of the objective and the available knowledge of the system. Despite the fact there is a variety of techniques developed to facilitate tuning, it remains a rather difficult and time-consuming

process. By saying this, it is underlined that the numerical results and the described procedure presented in the following subsections might not be the best, but it was certainly adequate enough to be applied into the control system of interest. If there is more time to spend on further tuning it still could be continued.

4.8.1. Scale Factors

For the proper construction of the cost function, the scale factors for each plant input and output variable are specified. It is a recommended practice to perform scaling of the variables especially if the certain variables have much different magnitudes than others. The scale factor should be approximately equal to the span of the variable, being the difference between its maximum and minimum value in engineering units. Performing scaling allows to present the variables in the same order of magnitudes which facilitates the definition of weighting matrices for the cost function, as the correlation between the increase of the weight and its physical meaning on the system is more clear. The values of scale factors are based on the understanding of the system, its constraints and characteristics of environment - the maximum disturbance expected, maximum control inputs available and the maximum control error. A useful approach for scaling is to make the variables less than one in magnitude. This is done by dividing each variable by its maximum expected or allowed change [62]. The maximum deviation from a nominal value should be chosen by thinking of the maximum value one can expect, or allow, as a function of time. The scale factors for the inputs and outputs of the plant are depicted in table 4.1 below. The value of each vector of scale factors is tuned respective to each term of the cost function, as explained in Section 4.6.

EE_{scale} [m]	$[0.01, 0.01, 0.01]$
V_{scale} [m/s][rad/s]	$[0.01, 0.01, 0.01, 0.0175, 0.0175, 0.0175, 0.0175, 0.0175, 0.0175]$
FoV_{scale} [-]	$[0.00060917]$
U_{scales} [N][Nm]	$[20, 20, 20, 20, 20, 20, 100, 100, 50]$

Table 4.1: Scaling factor values related to each term of the cost function.

The values shown in the table have respective physical meaning. The EE_{scale} values equal to $1[cm]$ is the acceptable position error; The scale factor V_{scale} for the kinetic energy minimization term is equal to $1[cm/sec]$ for the linear velocity of the base, $1[deg/s] = 0.0175[rad/s]$ for the rotational velocity of the base and $1[deg/s] = 0.0175[rad/s]$ for the angular velocity of the manipulator joints. These values represent the assumed maximum allowable velocity of each of the degree of freedom. The scale factor FoV_{scale} for the field of view term is equal to the values of the following expression: $FoV_{scale} = 1 - cos(2^o)$, such that it is compatible with the definition of the term from cost function $(1 - C_{FoV} \cdot x_i)$. It is representing the preferred allowable error, such that the deviation of 2 degrees is acceptable. Last but not least, the scale factor U_{scales} applied on the control minimization term has value of $20[N]$ for the thrust control input, $20[Nm]$ for the torque of the base and $100[Nm]$ for the first and second manipulator joint and $50[Nm]$ for the third joint. These values were determined simply by taking the half of the upper bound of the control inputs values, in this way it can be interpreted as expected magnitude of each input signal. Clearly the value is bounded by the constraint upper and lower bound, not by the scale factor, however it is preferred not to reach the boundary values and any deviation further from the mean value is penalized more.

Finally, definition of the cost function as in equation 4.19 is augmented with the scale factors, such that they are dividers of each of the terms. The complete definition of equation, as implemented in the controller script, is given below in equation 4.46.

$$\min_{x_i,u_i} \sum_{i=1}^{N-1} \left|\left| Q_{EE_{position}} \frac{(\Delta EE_{ref} - C_{FK} \cdot \Delta x_i)}{EE_{scale}} \right|\right|_2 + \left|\left| Q_{energy} \frac{v_i}{V_{scale}} \right|\right|_2 + \left|\left| Q_{FoV} \frac{(1 - (FoVcos_{op} + C_{FoV} \cdot \Delta x_i)}{FoV_{scale}} \right|\right|_2 + \left|\left| R \frac{u_i}{U_{scales}} \right|\right|_2 \tag{4.46}$$

The values of the scale factors are kept constant during all the control loop simulation. Setting different numerical values would effectively result in different balance between cost function terms. In case of project extension, it would be recommended to run sensitivity study to see how different scale factor values affect the result of the optimization problem.

4.8.2. Weights

The priority of each term of the cost function is balanced with weighting factor, such that the higher the value the more important it is to achieve a certain sub-goal. It is directly affecting the optimization process, the term with higher penalty will have more drastic effect in the increasing the value of the objective function which shall be minimized. The weighting factors are translated into the weight matrices in order to match the size of the problem, such that the quadratic term including the multiplication of the weight matrix and an expression to be minimized in the equation 4.46 results in the scalar value. The chosen values of the weighting factors are put on the diagonal of the null matrix of an appropriate size. The values of the weights applied on each penalty term from the cost function are given below. The balance between these values was found in a tuning procedure in earlier simulations which is further described in subsection 4.8.3.

$$W_{EE_{position}} = [W_{EEx}, W_{EEy}, W_{EEz}] = [50, 50, 50], \tag{4.47}$$

$$W_{FoV} = 50, \tag{4.48}$$

$$W_{energy} = [W_{vb}, W_{\omega b}, W_{\omega m}] = [10, 10, 10], \tag{4.49}$$

$$W_u = [W_{fb}, W_{\tau b}, W_{\tau m}] = [50, 50, 50] \tag{4.50}$$

The vectors of diagonal values are used to create the matrices of appropriate size, e.g. the weighting matrix applied on the term penalizing the error between the current and reference position of end effector is the following:

$$Q_{EE_{position}} = diag(W_{EE_{position}}) = \begin{bmatrix} W_{EEx} & 0 & 0 \\ 0 & W_{EEy} & 0 \\ 0 & 0 & W_{EEz} \end{bmatrix} \tag{4.51}$$

In the same manner, the remaining weighting factors are applied to construct the penalty matrices of the following sizes:

$$Q_{EE_{position}} \in \mathbb{R}^{3\times3}$$
$$Q_{FoV} \in \mathbb{R}^{1}$$
$$Q_{energy} \in \mathbb{R}^{(6+n)\times(6+n)}$$
$$R \in \mathbb{R}^{(6+n)\times(6+n)}$$

4.8.3. Tuning approach

Initial simulations were run with ht first guess of the weights values such that the highest weight was related to the EE reference position tracking $W_{EE_{position}}$ differed by two order of magnitude with respect to other weights. This value was chosen, because EE term is the primary objective of the cost function, and it is required from the controller to have good tracking performance. The result was not satisfactory, indeed the tracking was very good, but it resulted in very high extension of the robotic arm and high attitude change of the s/c base. Many iterations of the simulations with ”guessed” weight values were run and it was observed that not only the magnitude of the weights is affecting the solver performance but also the relation between weights of each term - the ratio between them.

Next, in order to simplify the tuning procedure and make it more clear, the baseline simulation with all unit weights was run. It allowed to clearly see which term has the highest contribution into the overall value of the objective function. It was observed that the highest contribution has EE term, therefore there is no need of drastically increasing the weight value. The weights of remaining terms were increased and only the energy-related term was kept lower. After running few more simulations, the values presented in the section above were found and allowed to reach very satisfactory performance.

In general, the tuning procedure performed for this project was to a major extent based on trial and error. If the more powerful machine was used for the computations, some grid search technique or learning algorithms could be used to find the best numerical balance between all the weights in more automated way.

4.9. Optimization Modelling Yalmip

Once the mathematical definition of an optimization problem is formulated, in the way explained in section 4.6, it must be set up in the understandable way for the optimization solver. In order to do so, the optimization problem can be formulated with the use of the available modelling engines. Some of the available toolboxes enabling an interface between the mathematical definition of an optimization problem and the solver are *CasADi* - an open-source tool for nonlinear optimization and algorithmic differentiation, *CVX* - a Matlab-based modeling system for convex optimization, *ForcesPro* - a software enabling generation of a tailor-made solvers from a high-level mathematical description of an optimization problem, and *YALMIP* - a free Matlab toolbox for rapid prototyping of optimization problems [73]. Due to the open-source characteristics, the easiness of the optimization problem modelling and last but not least a prior experience with the toolbox, the latter one YALMIP was implemented in this research project.

The modelling approach and general use of the toolbox is very straightforward. The user defines the constraints and objective functions using intuitive and standard matlab code, and the Yalmip performs an automatic categorization of the optimization problem based on the provided mathematical definition. One of the core ideas in YALMIP is to rely on external solvers for the low-level numerical solution of optimization problem, whereas the main focus is on the efficient modelling and high-level algorithms. The yalmip-specific variable is 'sdpvar' and it is used to define the optimization variable, it is a symbolic decision variable. It is very convinient as all the matlab operators can be applied also on sdpvar objects, therefore the creation of e.g. diagonal matrices can be easily performed using the 'diag' matlab function. The user defines constraints, objectives and solver options. The solver choice can be defined by the user or left to YALMIP, then the problem is solved and the solution obtained.

4.10. Final Control Loop

All the components of the controller design as described in the previous sections make up the final control loop. In a space systems engineering usually we refer to a complete GNC system, which comprises the guidance, navigation and control block. In essence, the guidance block task is to determine the reference trajectory that shall be followed, it says 'where to go', the navigation block determines the current position of the spacecraft, it says 'where it is', and the control block determines the specific actions of the actuators system to be executed in order to meet the position goal. In the Figure 4.4, the control block is only shown, as during the project development it was assumed that the main focus is on the controller design, the reference trajectory is determined by the guidance block for an entire motion, hence there is no additional feedback loop updating the guidance online, and last but not least it was assumed that the knowledge about the spacecraft position and its manipulator joints is perfect, hence there was no focus on navigation part.

The diagram in the Figure 4.4 consists of two parts: 1) Data Initialization and Pre-computations, 2) NMPC Feedback Control Loop and the Dynamics Simulation. The first concerns the set-up of the problem - the input model data is used to construct the dynamical model compatible with the SpaceDyn architecture, the initial conditions and the values of parameters required for the construction of the MPC problem are set. The reference trajectory is computed (see Section 4.5), it is an input to the control system in the form of a look-up table. The second part of the diagram, the NMPC Feedback Control Loop and the Dynamics Simulation, depicts the feedback loop which consists of the three main parts:

- *NMPC - Successive Linearization*; in this block the operating point is updated, the dynamics model is linearized around this operating point and the functions - forward kinematics and field of view, are also linearized in order to obtain the matrices necessary for the objective function construction.
- *Optimization block*; all the parameters obtained are used to construct the optimal control problem. The cost function is updated, the inequality constraints are defined with respect to the values of the operating point. The quadratic programming problem is defined in a yalmip interface and solved with the use of quadprog solver.
- *Nonlinear Plant Model*; the nonlinear model of the plant dynamics is used, it is equal to the definition of the model that is linearized in every iteration step. The mis-match between the prediction model and the plant model comes from the linearization procedure only. The first control input sequence from the computed optimal solution is injected into the continuous dynamics model and integrated to obtain the state vector values at the next time step.

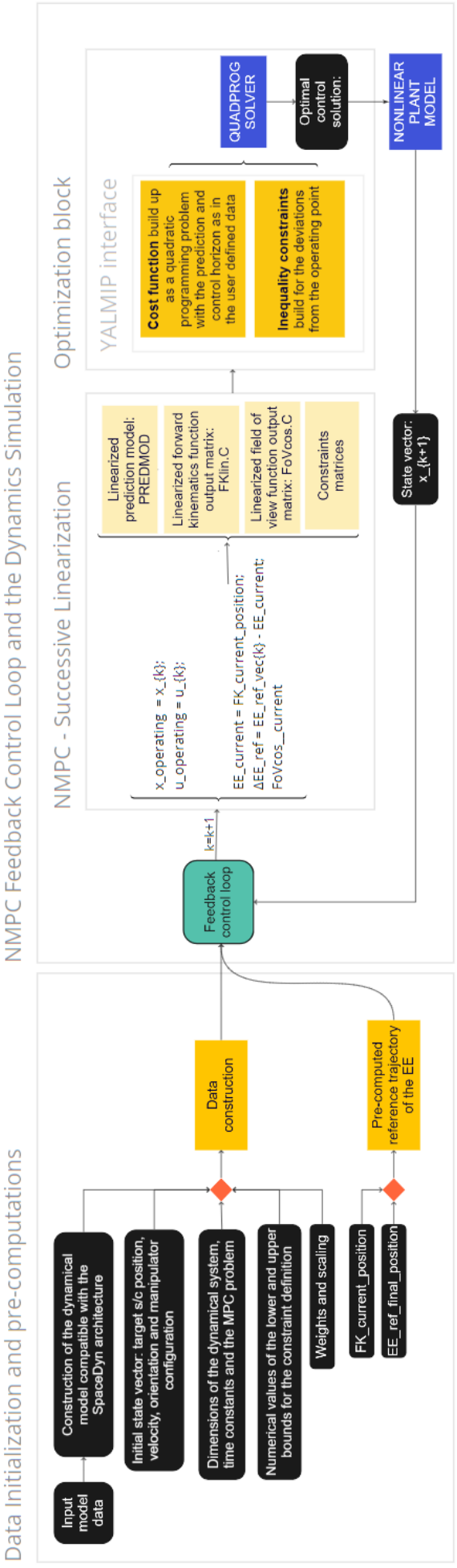


Figure 4.4: The final control loop.

5

Controller-in-the-loop Case Scenarios Results

In this Chapter, the results from the simulations of the designed controller will be presented. The modelled dynamics with the designed controller in the loop were tested for different scenarios depending on the initial conditions and values of the penalty matrices. In order to achieve a good reference tracking performance, the definition of the cost function and MPC parameters must be properly tuned. This is not a straightforward task, especially for such a complex system, however preliminary results of the tuning are presented here and conclusions from the effect of the change of weights values on the overall performance of the controller can be reached. Firstly the simulation with unit weight for each term of the cost function is presented, next the results from the simulation with the tuned weights are shown. The proper tuning of weights clearly allows to decrease the end-effector position error during all the simulated motion.

In Section 5.1 the table with the parameters values specific for each scenario is presented. Next, the plots of the position, velocity, attitude and angular rate during all the simulated maneuver for each body of the system are shown and described. The required input data to the simulation contains the parameters of the multi-body system derived from the RICADOS project data. The data includes the information for every link of the robotic satellite system. Therefore if only few joints are considered active, the non-actuated joints are assumed to be a rigid connection between two consecutive links. This input data was described in Section 3.2. In this Section, the data are pre-processed in order to be compatible with the format required for a proper definition of the system dynamics with SpaceDyn toolbox. Last but not least, the initial conditions and MPC specific parameters such as value of the weight matrices, scale factors etc are set by the user before running the simulation control loop. The values of input parameters for each presented scenario are given in a table 5.1.

In Section 5.2 the timing statistics of the simulations are given. It is clear that the solver takes very little time to find the solution of an optimization problem, however the overall control loop is time-consuming. The required time is an important constraint especially in real-time applications, and the consequences of designing too complex solver that results in longer computation times could be that it is not suitable to be used in real-time space mission.

Last but not least, in Section 5.4 the performance metrics of end-effector position tracking are given in terms of the root mean squared error and the mean absolute error. With these numerics performance analysis is very straightforward and allows to choose the best design candidate. Results of the simulations and comparison between them is addressed also in this Section as well.

5.1. Scenario cases

The input data of the presented scenarios is given in table 5.1 below. In next sections, the results of the simulations are shown in the figures. The most representative results were chosen that validate the functioning of the designed controller for the on-orbit servicing type of mission.

The scaling factors used for the cost function definition in all the scenarios are the same and are equal to the values in table 4.1. Therefore only the penalty weights are tuned with this approach. The initial condition is described with an initial value of a state vector $x0 = [R0; Q0; q; v0; w0; qd]$; It is assumed that the beginning of the reach maneuver phase follows the end of the close rendezvous phase. Therefore in order to make the transition as safe as possible, and change the required parameters of the controller, the robotic spacecraft decelerates to keep the position in the hold stop. In this moment, the data could be down-linked to the control center to e.g. allow the user-based activation of the maneuver phase or parameters up-link if necessary. For this reason it can be assumed that the relative velocity between the chaser and target spacecraft is zero. Therefore, the velocity-related terms in the initial value of a state vector are always equal to 0 and in table 5.1 the values of the base

position, attitude and joints angular position are only presented.

In the table the N stands for the length of a prediction model, m is the control horizon, Ts is sampling time used both for the discretization of the dynamics prediction model for the MPC formulation, and for the plant dynamics. The frequency of the MPC controller is also equal to Ts. The time during which the maneuver should be accomplished is another parameter, it is the simulation time given in the table. The generation method of the reference trajectory is also specified (refer to Section 4.5) as it has influence on the tracking performance.

Scenario 1 concerns the simulation with the unit values of all the penalty weights, **Scenario 2** is the simulation with tuned weights for the same initial conditions that shows how the performance is improved thanks to tuning. Next **Scenario 3** is a comparison between two simulations run for the same input data and parameters values but different method of the reference trajectory generation. First, the linear interpolation is used to generate the reference trajectory, as explained in Section 4.5. It is compared to the other method - the interpolation with the use of 5^{th} order polynomial function, as explained in Section 4.5. The plots showing results of every scenario are presented in next section.

Table 5.1: Data of the simulated scenario cases.

	Scenario 1	**Scenario 2**	**Scenario 3**
Ts [s]	0.01	0.01	0.01
N [ ]	10	10	10
m [ ]	1	1	1
Simulation time [sec]	180	180	90
Reference trajectory [ ]	poly5	poly5	linear/poly5
$x_0[]$	$\begin{bmatrix} -4.5 \\ 0 \\ 0 \\ 0 \\ -5^o \\ 0 \\ 0 \\ 45^o \\ 90^o \end{bmatrix}$	$\begin{bmatrix} -4.5 \\ 0 \\ 0 \\ 0 \\ -5^o \\ 0 \\ 0 \\ 45^o \\ 90^o \end{bmatrix}$	$\begin{bmatrix} -4.5 \\ 0 \\ 0 \\ 0 \\ -5^o \\ 0 \\ 0 \\ 45^o \\ 90^o \end{bmatrix}$
$EE_0[m]$	$\begin{bmatrix} -3.5570 \\ 0 \\ 2.6117 \end{bmatrix}$	$\begin{bmatrix} -3.5570 \\ 0 \\ 2.6117 \end{bmatrix}$	$\begin{bmatrix} -3.5570 \\ 0 \\ 2.6117 \end{bmatrix}$
$EE_{end}[m]$	$\begin{bmatrix} -0.4178 \\ 0 \\ 1.0770 \end{bmatrix}$	$\begin{bmatrix} -0.4178 \\ 0 \\ 1.0770 \end{bmatrix}$	$\begin{bmatrix} -0.4178 \\ 0 \\ 1.0770 \end{bmatrix}$
$Q_{EE_{position}}$diag[]	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 50 \\ 50 \\ 50 \end{bmatrix}$	$\begin{bmatrix} 50 \\ 50 \\ 50 \end{bmatrix}$
Q_{energy}diag[]	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}$
$Q_{FoV}[]$	1	50	50
R diag [ ]	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 50 \\ 50 \\ 50 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}$

The initial conditions of the scenarios were chosen such that the characteristics of the simulated motion are as close to the real space mission as possible. It was assumed that the last hold point before performing the reach and capture maneuver is 4.5 meters away (the distance between the s/c base c.o.m. and the target s/c c.o.m). This value was chosen as it is long enough to perform the maneuver with extension of the robotic arm, but not too short, therefore the translational motion of the base is expected and not only of the manipulator.

The simulation time of 3 minutes (180 seconds) is chosen, which should be physically achievable time for performing the maneuver. As an example, in the study [74] the authors consider proximity operations at geosynchronous orbit. They consider few hold points - one at 10 meters away from the target spacecraft, where s/c stays for 10 minutes and then approaches the last hold point along the capture axis 1 meter away from the grapple

point on the target. They considered 3 minutes to perform this maneuver, which effectively results in the approach rate equal to 5 cm/sec. Considering the same amount of time for shorter distance in this work will result in lower value of the approach rate which could be potentially preferable for safety reasons. When considering the distance between the x-coordinate of the initial end-effector position EE_0 and the x-coordinate of the grapple point EE_{end} (defined in the target reference frame), the average approach rate is 1.744 cm/sec, see below:

$$\frac{|\Delta EE_x|}{180} = \frac{|-3.557m - (-0.4178m)|}{180sec} = 1.744[cm/sec]$$

The initial angular position of the robotic manipulator joints is such that the arm is already extended in the beginning of the phase: the first joint is in its nominal position, while second joint is $q2 = 45^o$ and third joint $q3 = 90^o$. Following the definition of the plant model, see Section 3.1, the rotation axis of the first joint, if the s/c base is not rotated, is co-linear with the z axis of the body reference frame. The non-zero angular position of this joint results in the non-zero y-coordinate of the end-effector position. The grasping point positioned on the target spacecraft is co-linear with x-axis, therefore it is preferable to keep the position of the end-effector also co-linear with the x-axis. Moreover, the motion of the manipulator around the z-axis of the servicing s/c body frame would result in the non-zero y coordinate of the end-effector. It would cause additional disturbance on the satellite base, which potentially could induce the rotational motion of the base and hence require the counteracting action of the torquers.

Last but not least, the non-zero attitude of the servicing spacecraft with respect to the target spacecraft was chosen, the initial roll angle is equal to -5^o, whereas pitch an yaw are zero. In real space mission, before starting the reach and capture phase, the attitude synchronisation is performed such that the attitude and angular rate of the servicing s/c is matching the target spacecraft. However in order to test if the controller is able to compensate the attitude offset, the tested scenarios begin with this non-zero attitude. The initial position of the servicing spacecraft is presented in figure 5.1 below.

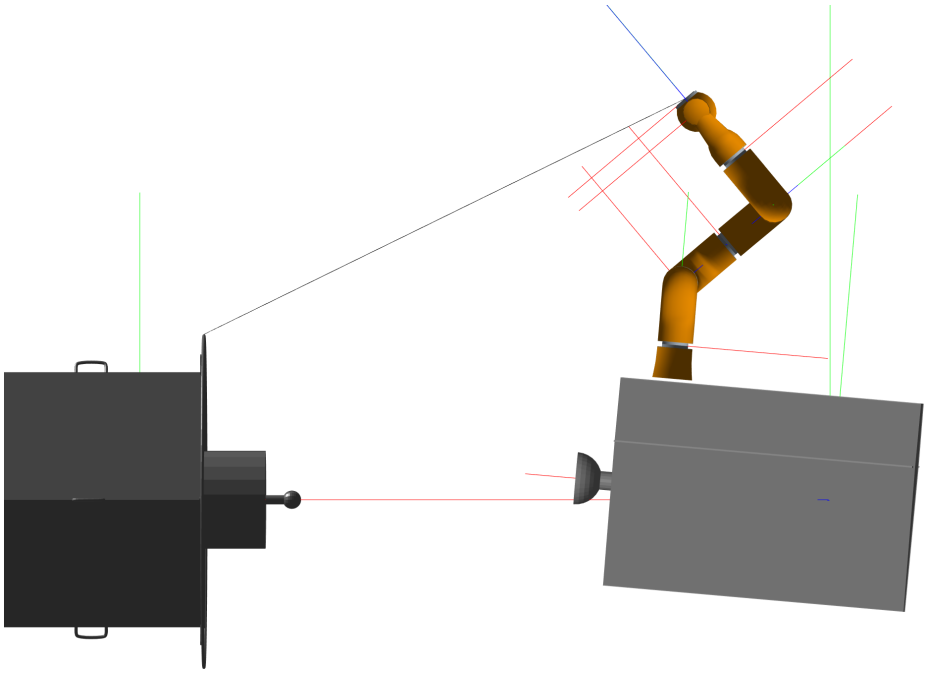


Figure 5.1: Initial condition for scenarios 1-3.

5.1.1. Scenario 1 - Baseline definition with unit weights

In order to get a good overview of how the change of the weight value for each cost function term affects the overall performance of the controller, firstly a scenario with all the unit weights is run. It is assumed that the choice of the scale factors values made during the controller design is kept constant. These values were chosen based on the informed guess what the value of the allowable position error and of the expected average value of velocity should be. This modelling choice is valid for every case scenario presented in this Section.

Figure 5.2 presents the evolution of the motion of the servicer spacecraft base, the center of mass is considered. The position vector $R0$ is defined in the LVLH frame centered at the target spacecraft, as explained in Section 3.2. The servicer is approaching from the negative x-axis side "behind" the target spacecraft, therefore the relative -x coordinate has negative value. The initial position in which the motion starts is 4.5 meters behind the target spacecraft in negative x direction, the y and z coordinates are equal to zero. The objective function of the optimal control problem includes the reference positions of the end-effector but not of the base c.o.m. Therefore the base motion is an available degree of freedom of the system, the evolution of the motion of the base is the choice of an optimizer. In figure below it can be observed that the motion is performed along all the three axes, with the longest distance along x-axis, which was expected because firstly the servicer needs to approach the target in order to achieve the final EE position. The linear velocity V_x is in the range $0 - 1.5[cm/s]$ which is a reasonable value ensuring the safety of the maneuver. The V_z reaches the highest amplitude of $6[mm/s]$, the change of position in the z direction is much slower compared to x direction. By the end of the maneuver the position of the base is $45[cm]$ in the z direction. Motion along y-axis is negligible. Linear velocity profile in each direction is such that the body decelerates while approaching the end of the maneuver.

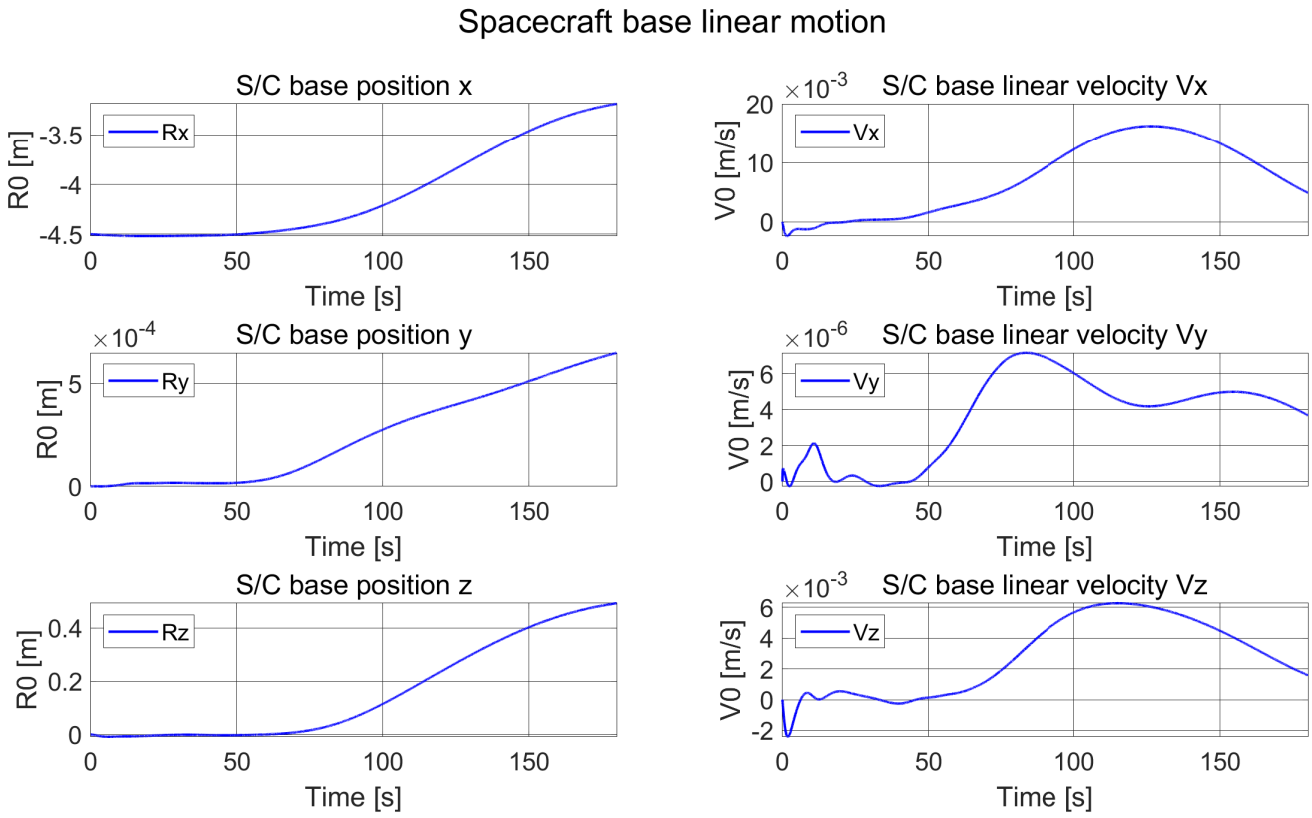


Figure 5.2: Scenario 1 - Spacecraft base position and linear velocity in LVLH frame.

Next, figure 5.3 below shows the evolution of the spacecraft base angular motion in RPY convention. The attitude is defined in the body reference frame of the spacecraft co-linear with LVLH. The initial attitude is non-zero only around y axis, the initial pitch $p = -5$deg, which is equal to -0.0873 rad. During the simulated maneuver, the pitch angle grows positive and achieves the largest value by the end of the motion equal to 0.22 rad = 12.61 deg. It can be seen in figure 5.8 how the term related to Field of View cost evolves during the simulation. The highest value is reached for the initial condition, during the second part of the motion there is another peak value and next the value of the FoV term decreases almost to zero. Despite the pitch angle is growing, it keeps a good pointing angle with respect to the target navigation aid markers due to the fact that the position of the base is

moving along the positive z axis. All together, the increase of the pitch angle is not linearly correlated with the value of the FoV term, as the position of the spacecraft base also influences the relative pointing between two satellites. The angular motion around x and y axis is negligible, the highest value achieved among these two axes is the roll angle in the 125th second equal to -0.003rad $\simeq -0.1719$deg.

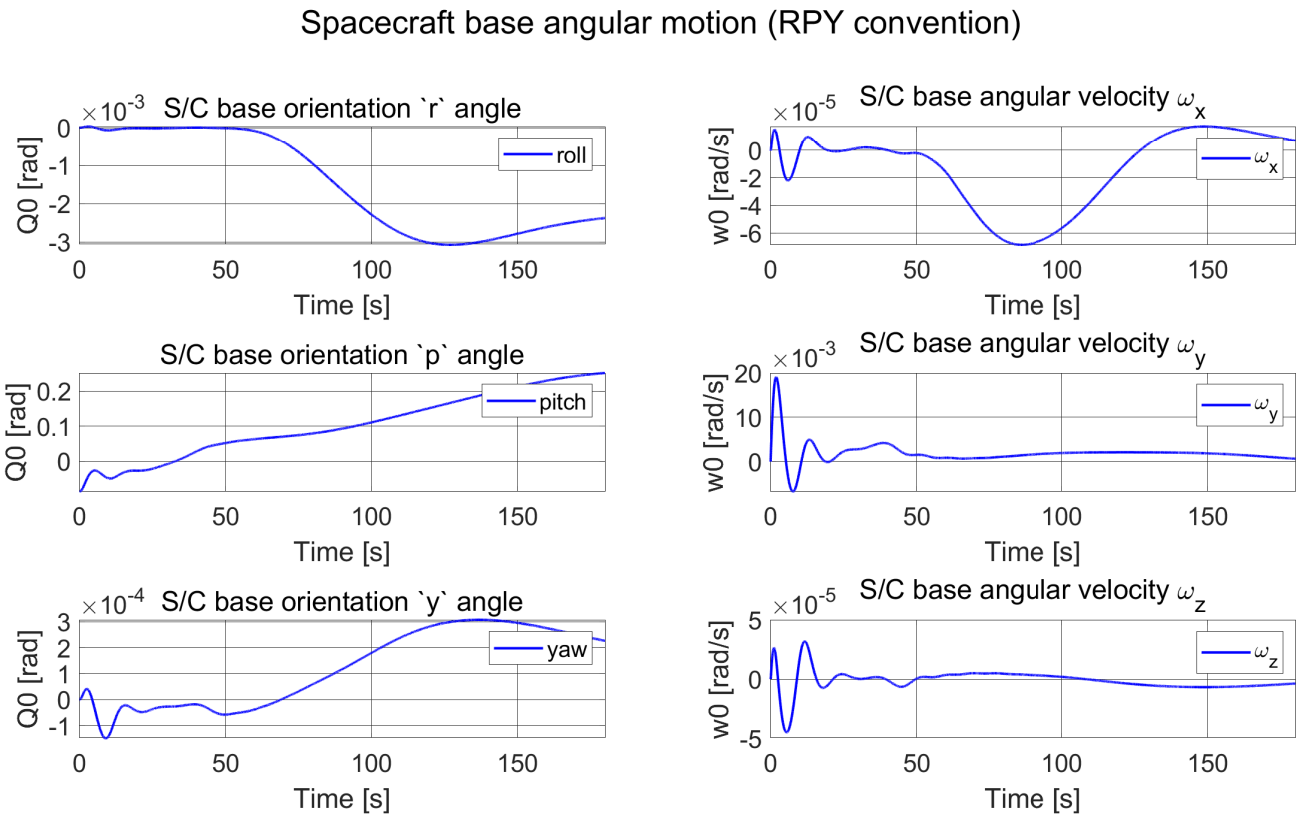


Figure 5.3: Scenario 1 - Spacecraft base attitude and angular velocity in body frame.

In figure 5.4 the arm motion is presented, which is the evolution of an angular position of the manipulator joints. The plant architecture was presented in Chapter 3, it is clear that each of the three joints is a revolute joint. The angular position is given in the moving coordinate frame fixed in each link. From the figure it is clear that the first joint is only slightly actuated, its motion is negligible. The main difference in the angular position is in the second joint q_2, with the initial angular position 45 deg, and the final position -45 deg. The overall change during all the simulated motion is around 90 degrees in this joint. The angular rate $\dot{q}_2$ highest amplitude is equal to 0.02rad/sec $= 1.15$deg/sec. The initial condition of the third joint q_3 is 90deg, during the simulated maneuver it keeps growing until it reaches the angular position of 1.82rad $= 104.23$deg in the 90^{th}. Then it rotates in the opposite direction and reaches the final angular position of 1.42rad $= 81.36$deg. The motion of the joints is very smooth, the initial oscillations clearly seen in the angular velocity profile are quickly dumped and the evolution of the velocity is a smooth function during the entire motion.

The evolution of the control inputs for each degree of freedom is presented in figure 5.5. The control input values are found by the solver as a solution of an optimization problem. The values of the control input from the first stage of an MPC problem are taken directly from the solver output and injected into the plant model. The first plot presents the evolution of the thrust force in each of the three directions, the second plot shows the evolution of the torques acting around each of the main body axis and last but not least, a plot of the value of the torque applied in each of the manipulator active joints is shown. From the plots it is clear that when initializing the capture phase and the motion of the system assuming the stationary initial conditions and null value of the initial control inputs, the beginning of the control response has oscillatory characteristics. This behaviour settles down very smoothly after the first 20 seconds of simulated motion and for the remaining simulation time the control input evolution is very smooth. We can see that the value of the force and torque applied on the s/c base remain in a proximity of zero for the majority of the motion whereas the highest values are reached only at the beginning of the motion. Interestingly, from the evolution of the manipulator joint torques, it is clear that the solver decides to apply an actuation mainly on the second and third joint with the value of torque in the first joint close to zero for the entire motion.

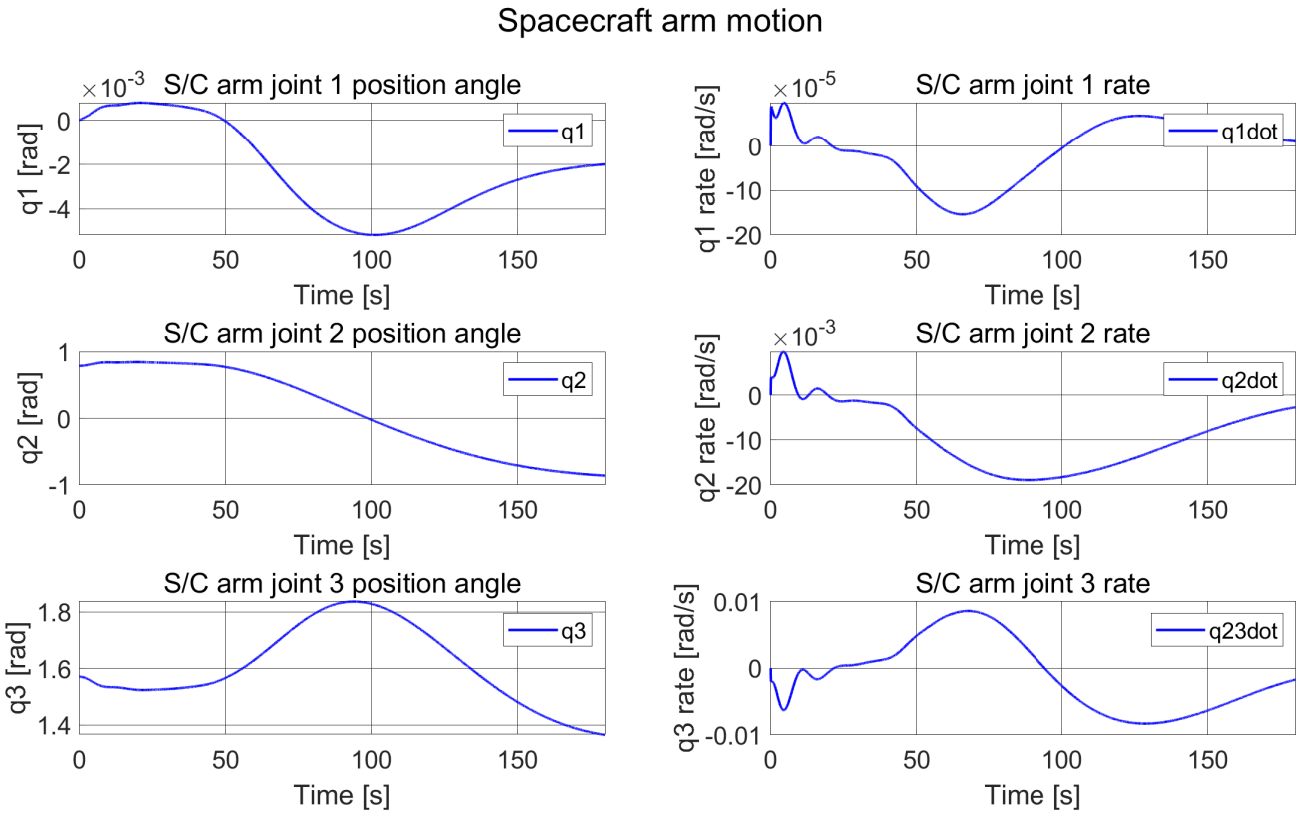


Figure 5.4: Scenario 1 - Spacecraft arm joints angular position and angular rate in joint frames.

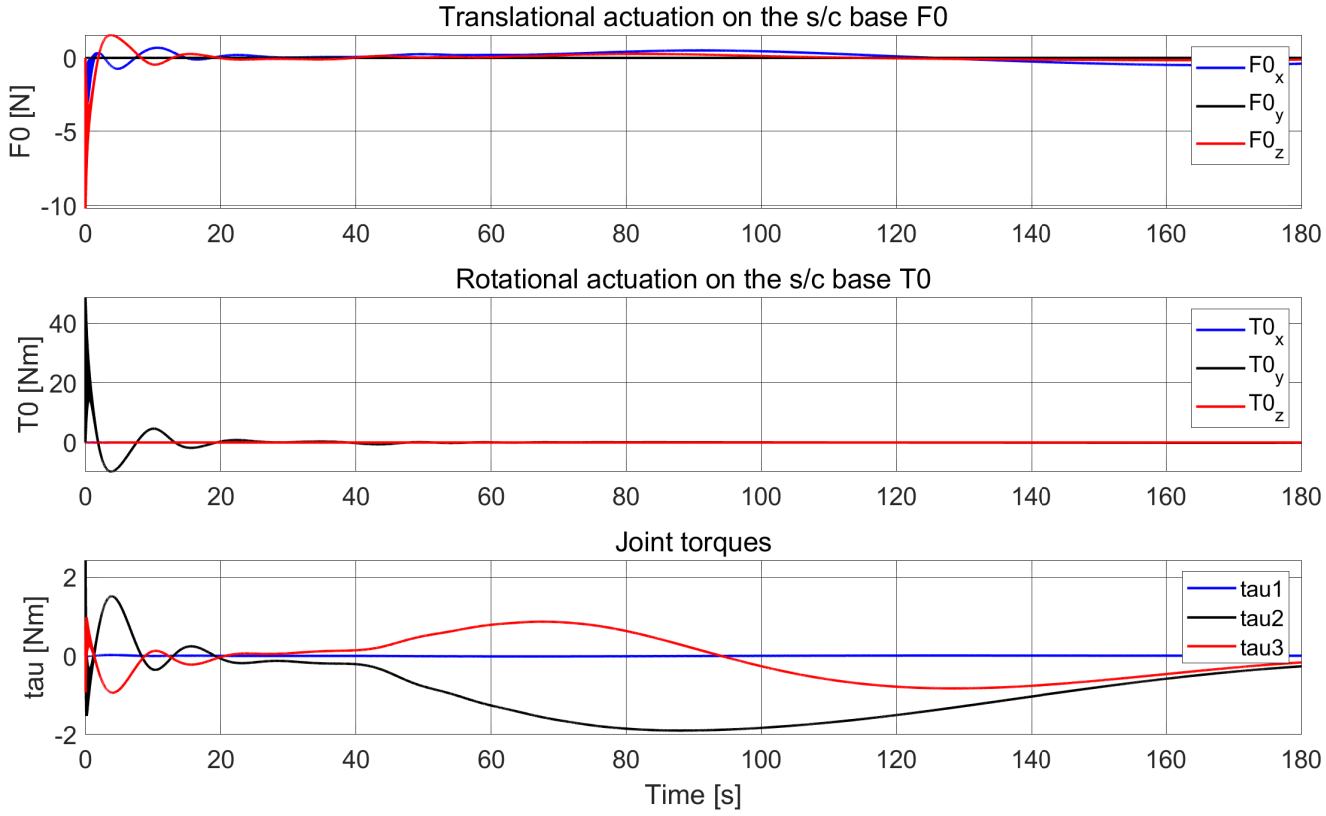


Figure 5.5: Scenario 1 - Control inputs.

The resulting position of the manipulator end-effector during the simulated motion is presented in figure 5.6 below. The end-effector remains in the same x-z plane, it is clear from the plot that the y coordinate is equal to zero during all the motion. It is quite a logical result consistent with the plots that were presented before. The only way to induce the change of y coordinate is the motion of the first joint $q1$ or the servicer base roll or yaw angle change. Each of these three angular positions were shown to be negligible, hence the end-effector y coordinate remains constant during all the simulated motion. In the plot, the x- and y-coordinates of the final position of the end-effector are marked with circles. It is clear that the desired position is not reached, the exact value of the offset is clearly seen in figure 5.7 which shows the end-effector position error evolution for every simulation step. The error for every axis is computed by subtracting the value of the reference position in k^{th} simulation step from the value of the position output from the simulation in this step:

$$EE_{error}(k) = EE_{sim}(k) - EE_{ref}(k) \tag{5.1}$$

The plots in figure 5.7 show the error evolution for each of the three coordinates, clearly the y coordinate error is equal to zero. The highest amplitude of an error is achieved in the 120 second in x coordinate, the error is equal to -45cm, which means that the position of end-effector was ”behind” the reference position for his simulation step by this distance. Such error is definitely too large for this type of a mission with high safety requirements. However, this high error happens one minute before the end of the maneuver, and the final position accuracy is much better. The final end-effector position error in x-coordinate is equal to -11cm and in z-coordinate 6cm. This is not acceptable error range and clearly leads to overshoot of the capture point by the end effector. As far as the x-coordinate error is negative, the poor tracking performance should not cause major risk, however a positive value of the error means that the end-effector position is closer to the target spacecraft than expected, and it could potentially lead to a collision. In order to increase the tracking performance it is clear that the weights shall be tuned, and with the current design and unit weights only, the tracking objective cannot be achieved with a desired performance.

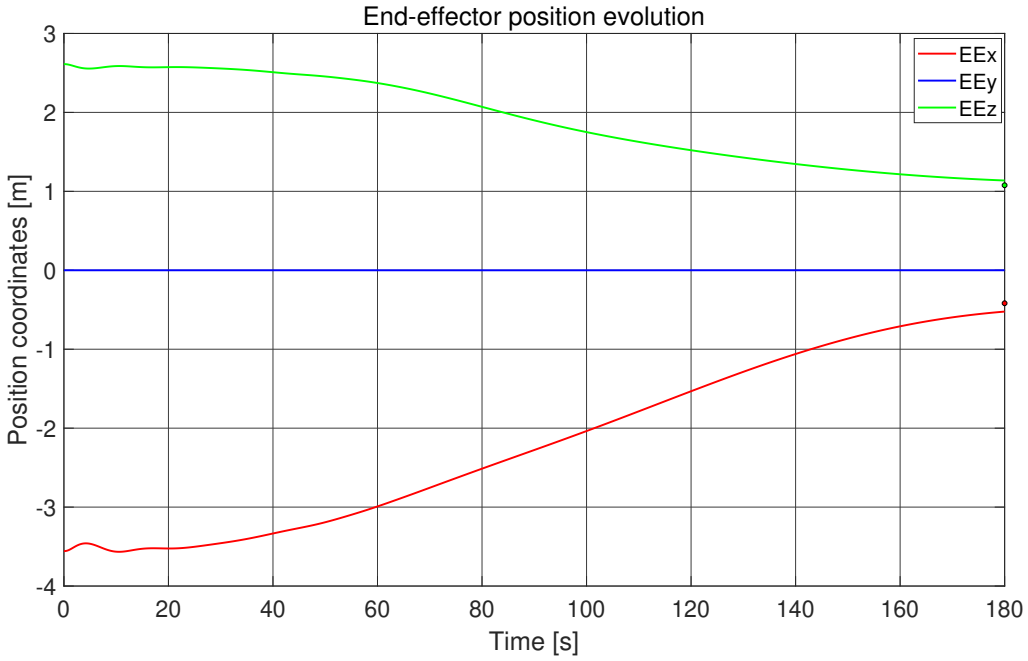


Figure 5.6: Scenario 1 - End-effector position coordinates in LVLH frame.

In figure 5.8, the contribution of each term from the cost function to the overall value of the function is presented in plots. It is clear that the highest contribution has the end-effector position error of x-coordinate at t = 120 s. When the position tracking performance is better, this cost function term decreases. Next, the field of view related cost has impactful value on the overall function, however the magnitude is slightly smaller than the EE cost. The energy cost and control inputs cost have the lowest contribution. It is very clear from the plots, that the designed cost function is correctly depicting the ongoing changes in the plant dynamics and reference position errors. The optimization solver is minimizing the overall value of the function while looking for the best solution, therefore it is important to validate that the cost function represents what the design intention was. In figure 5.9, the overall value of the cost function is presented.

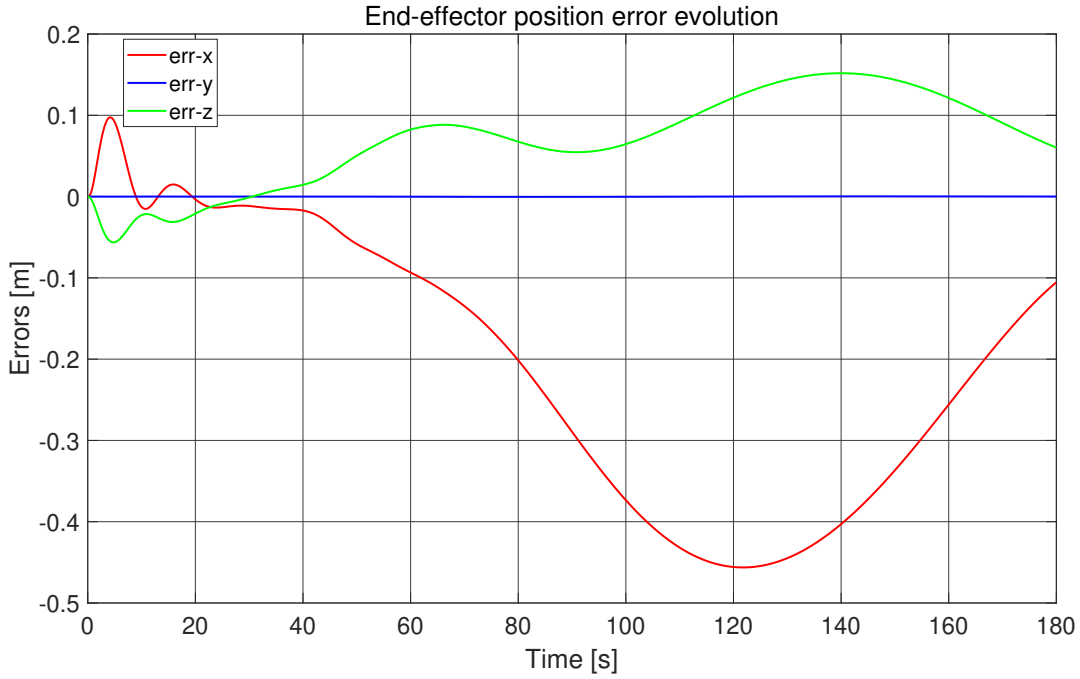


Figure 5.7: Scenario 1 - End-effector position error.

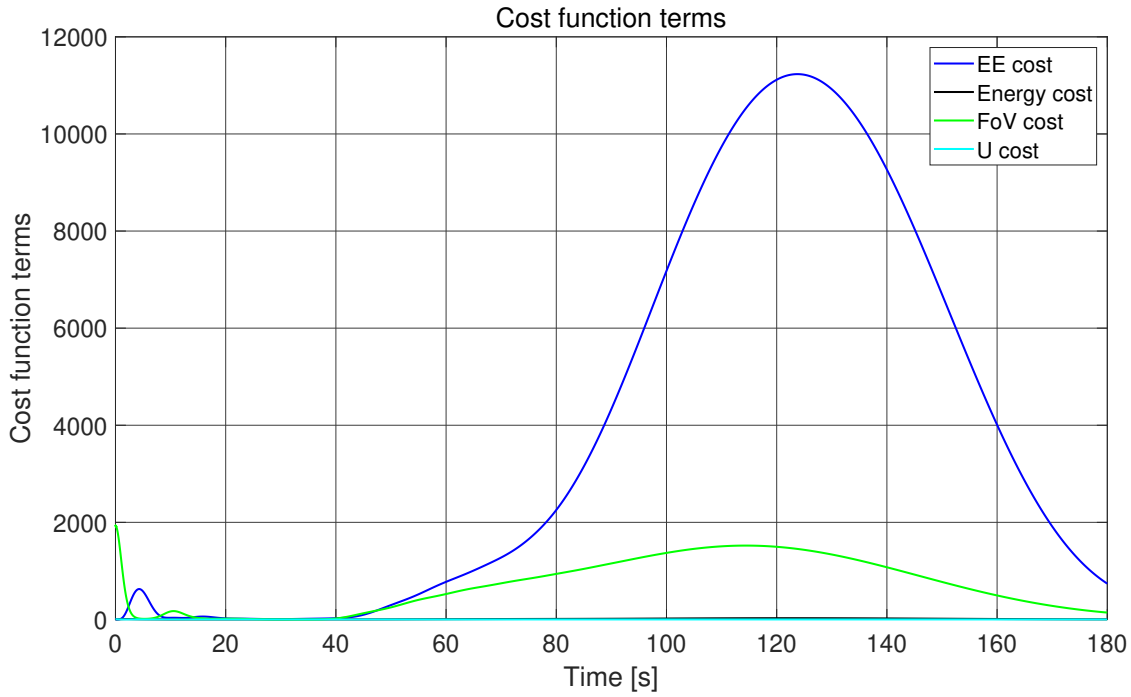


Figure 5.8: Scenario 1 - Cost function terms.

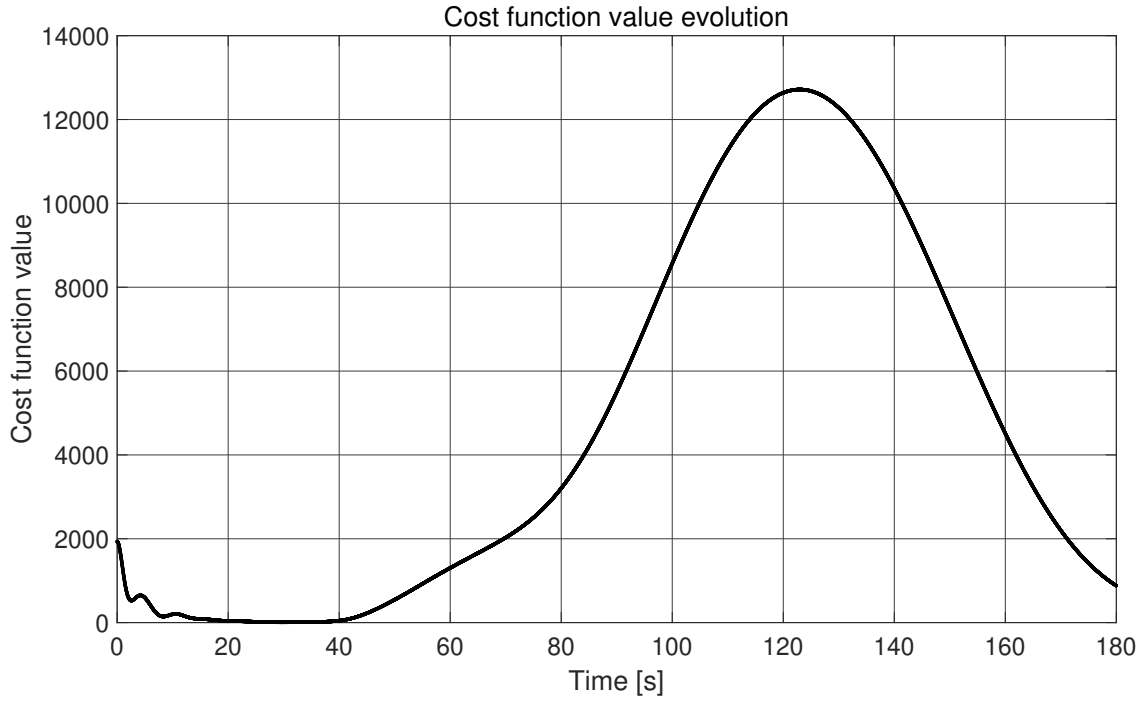


Figure 5.9: Scenario 1 - Cost function overall.

In order to visualize the motion of the controlled servicer, a 3D model provided by DLR is used. It represents the multi-body system as is available in the OOS-Sim on-orbit servicing simulator. The data from matlab simulation, specifically the state vector in current simulation step, is provided to the visualization function which updates with the same sampling time as the control loop. Hence, the motion of the spacecraft can be seen in real time during the simulation. In figure 5.10, the position of the servicer achieved by the end of the simulated motion is shown. On right we see the servicer spacecraft equipped with the robotic arm and on the left the target satellite to be captured. The black line is the reference trajectory starting from the initial EE position in the top right and finishing on the LAR ring of the target which is considered the capture point.

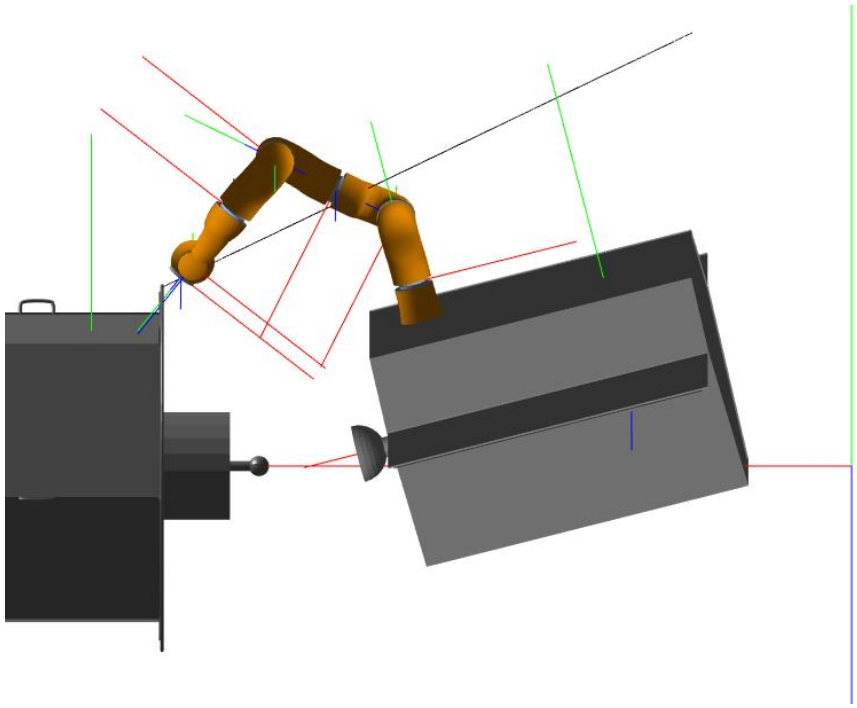


Figure 5.10: Scenario 1 - Final capture position from the simulation viewer.

5.1.2. Scenario 2 - Baseline definition with tuned weights

The results of the simulation for the weights values of the scenario 2 (see table 5.1) are presented in this section. As it was seen in the outcome from the simulation of scenario 1, the reference position tracking of an end-effector was very poor. For this reason the weight on the EE term from cost function is increased from 1 to 50. Next, the weight on the term related to the Field of View and the weight on the control inputs are increased to the value of 50 and term related to the energy - velocity, is increased to 10. The simulation was run with these weight values and the outcomes are shown below. Overall, the main control objective - the end-effector reference position tracking is achieved with much better performance. Therefore it is confirmed that proper weight tuning is the most essential part of the design required to achieve good performance of the controller.

Figure 5.11 presents the evolution of the motion of the servicer spacecraft base, with the center of mass being considered. It can be observed that the motion is performed along all three axes, with the longest distance along x-axis. The motion profile is very similar to scenario 1, with the main difference being in the y-direction: the total change of the position by the end of the motion is from $y_0 = 0$m to $y_f = 1$mm, whereas in the scenario 1 it was 0.6mm. The velocity V_y shows much more oscillatory behaviour compared to the results of scenario 1 in figure 5.2. It is not a desired characteristics of the motion, nevertheless the value of the velocity is very low, in the submilimiter range therefore it should not have a great impact on the overall system behaviour. The velocity profiles are very similar to scenario 1 and approaching the end of the maneuver the body decelerates in each direction.

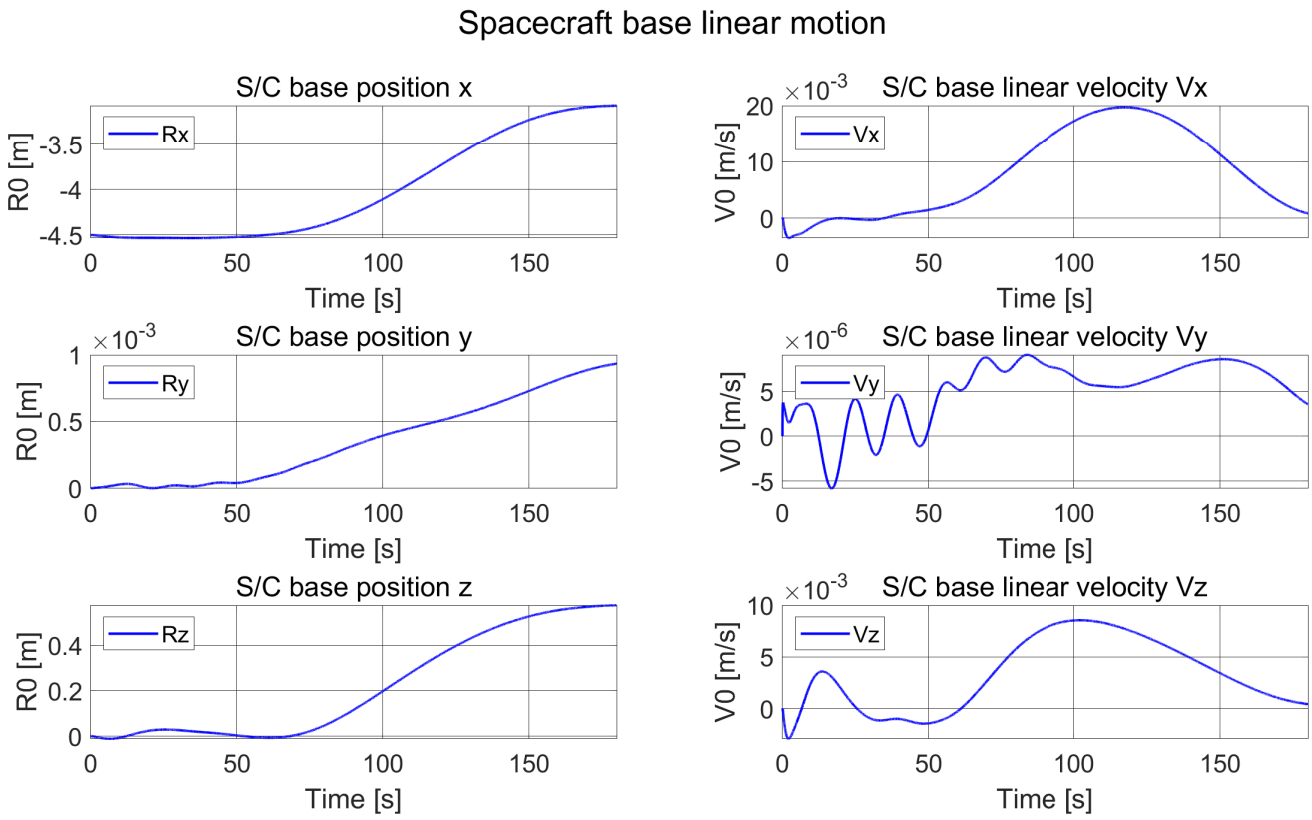


Figure 5.11: Scenario 2 - Spacecraft base position and linear velocity in LVLH frame.

Next, figure 5.12 shows the evolution of the spacecraft base angular motion in RPY convention. The attitude is defined in the body reference frame of the spacecraft collinear with LVLH. The initial attitude is non zero only around y axis, the initial pitch $p = -5$deg, which is equal to 0.0873 rad. During the simulated maneuver, the pitch angle grows positive and achieves the largest value by the end of the motion equal to 0.24rad $= 13.75$deg. The angular motion around x and y axis is negligible, the highest value achieved among these two axes is the roll angle in the 110^{th} second of the motion equal to -0.0032rad $\simeq -0.1833$deg. The angular positions evolution in each of the three axis is similar to the scenario 1, however in figure 5.12 we can see more oscillatory behaviour in the first half of the simulated motion especially comparing the plots of the angular velocity ω_x. When comparing the angular velocity ω_z with the outcome of the scenario 1 in figure 5.3 it is clear that the oscillatory behaviour in the velocity profile of the scenario 2 is much longer and is damped only at t = 100 s of the motion.

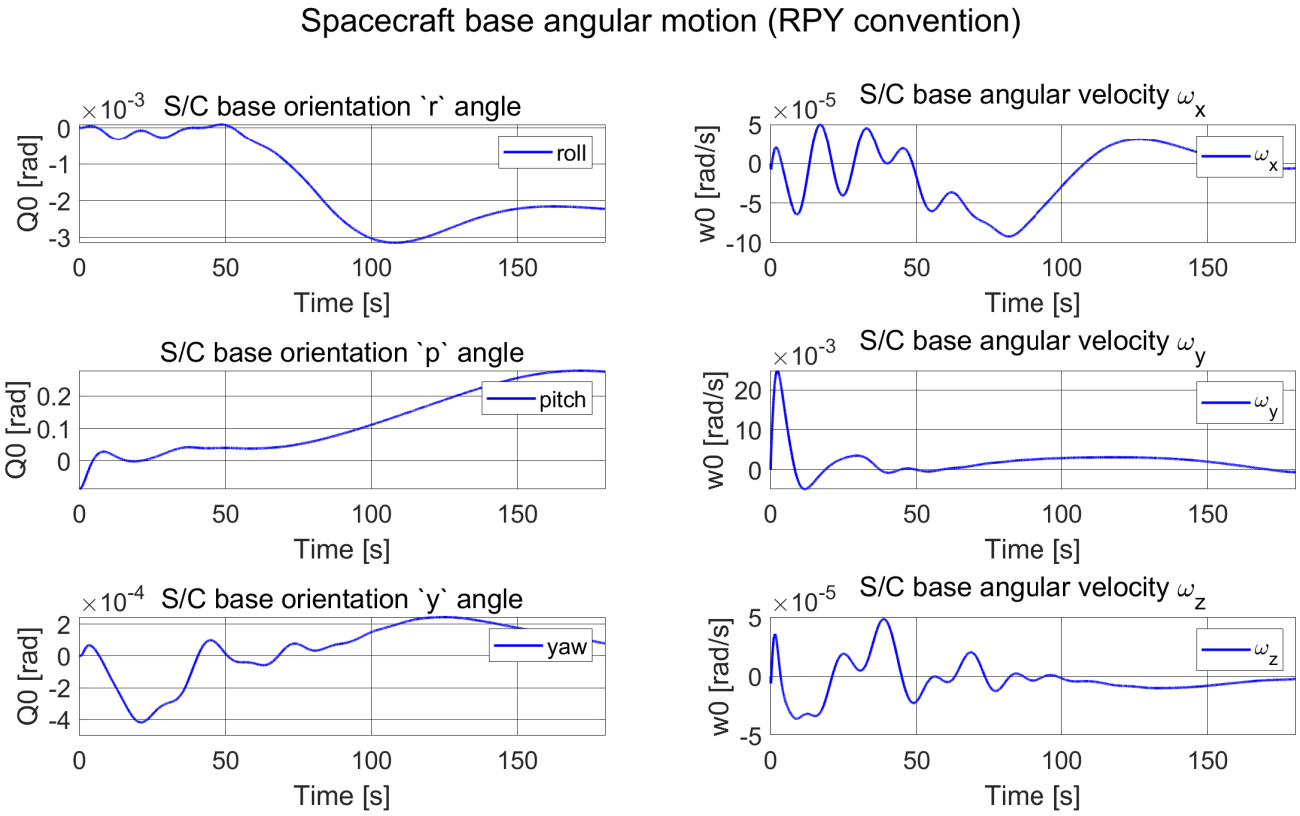


Figure 5.12: Scenario 2 - Spacecraft base attitude and angular velocity in body frame.

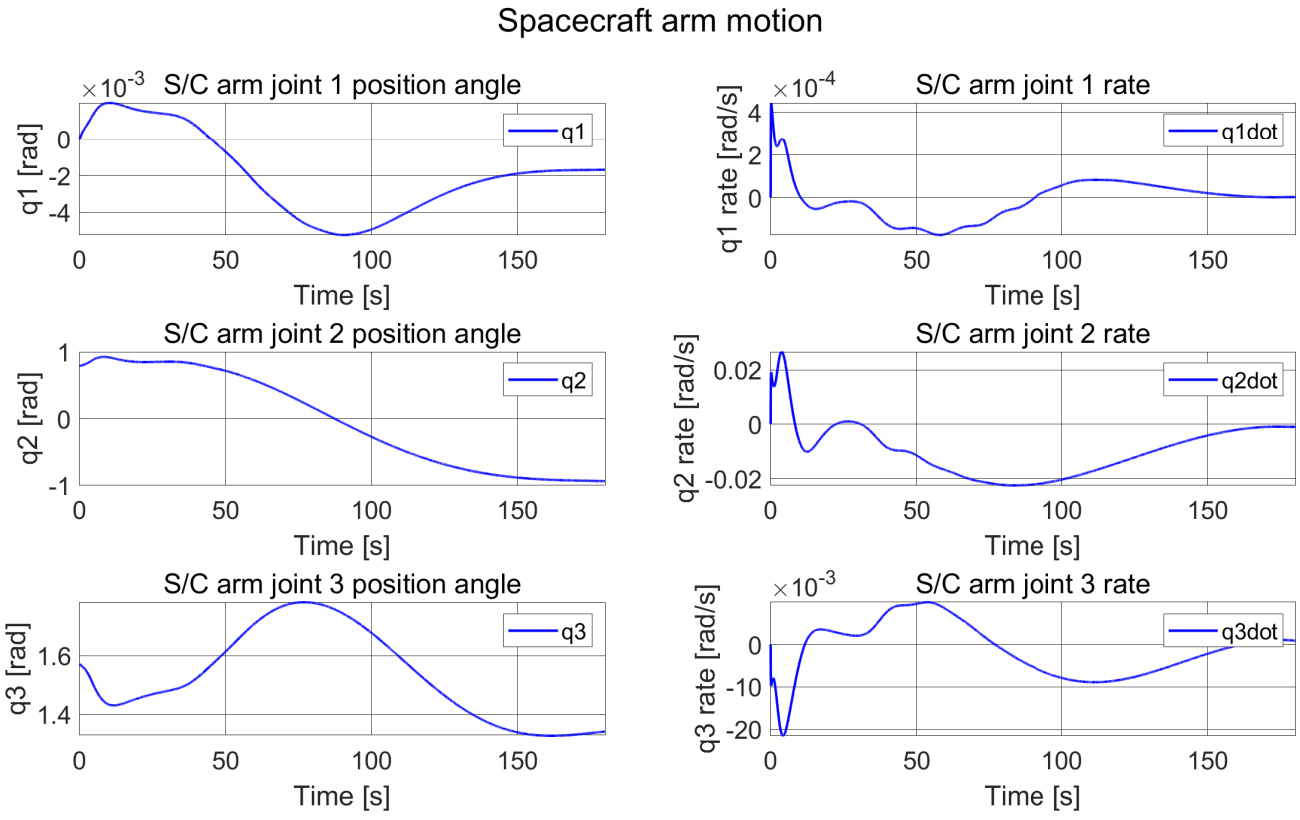


Figure 5.13: Scenario 2 - Spacecraft arm joints angular position and angular rate in joint frame.

In figure 5.13 the arm motion is presented, which is the evolution of an angular position of the manipulator joints. Similarly to scenario 1, the motion of the first joint q_1 is negligible as during the whole simulated maneuver the maximum angular position is only -0.004rad. The angular position of each joint during the motion has practically the same profile as in scenario 1, with the main difference being slight oscillatory motion in the very beginning of the maneuver. The largest change of the angular position is in the second joint q_2, with the initial angular position 45deg and the final position -45deg. The range of the angular velocity is the same as for the scenario 1, and the highest amplitude of $\dot{q}_2$ is equal to 0.002rad/sec. Despite the motion profile being so similar to scenario 1 with unit weights, it is clearly seen in figure 5.16 that the tuned weights enabled to reach much better performance of the end-effector position.

Next, the evolution of the control inputs for each degree of freedom is presented in figure 5.14. The control input values are found by the solver as a solution of an optimization problem. The first plot presents the evolution of the thrust force in each of the three directions, the second plot shows the evolution of the torques acting around each of the main body axis and the third presents the value of the torque applied in each of the manipulator active joints is shown. The profile of the control inputs is clearly similar to the scenario 1, with the main difference being much higher amplitude of the torque τ applied on the second joint of the manipulator q_2. It reaches the value of 12Nm in the very beginning of the motion, whereas the peak of the τ_2 in scenario 1 has value of 2.2Nm only. Each plot of the applied actuation on the base in terms of force and torque settles down very smoothly after the first 40 seonds of simulated motion. We can see that the value of the force and torque applied on the s/c base remain in a proximity of zero for the majority of the motion, whereas the highest values are reached only in the beginning of motion.

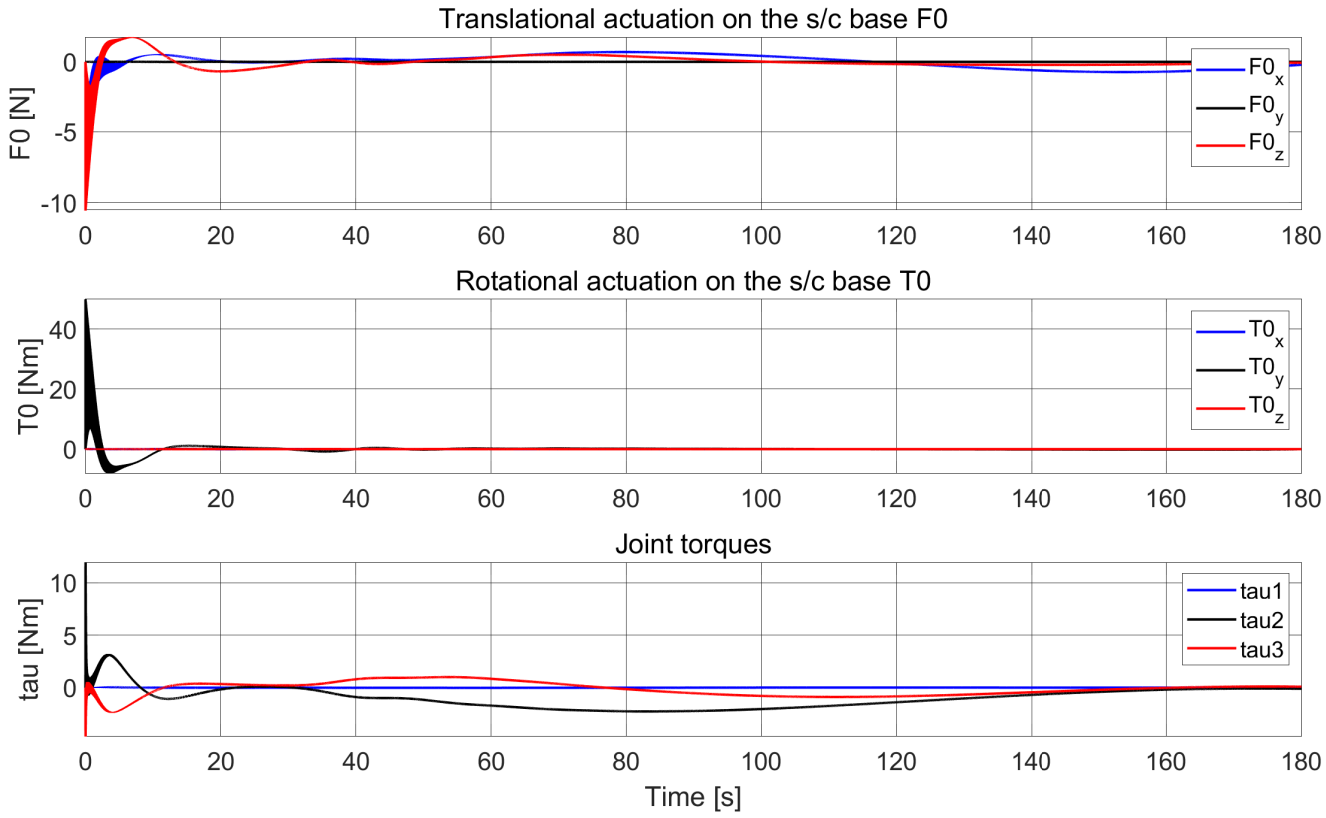


Figure 5.14: Scenario 2 - Control inputs.

The most interesting result achieved as a consequence of the tuned weights is an improved tracking performance of the reference end-effector position. The position of the manipulator end-effector during the simulated motion is presented in figure 5.15 below. The evolution of each coordinate is very similar to the scenario 1, however the main difference is seen in the second part of the motion in the value of the EE_x. The value of the x coordinate is growing slightly quicker, compared to scenario 1, and most importantly achieves the final required position very well. In the plot, the x- and y- coordinates of the reference final position of the end-effector are marked with circles. The improvement of reference position tracking performance can be better seen in figure 5.16 which shows the end-effector position error defined as in equation 5.1. The magnitude of the error for every axis is much smaller compared to scenario 1: the highest amplitude of an x-coordinate error is equal to -11.5cm and and of

an z-coordinate error is equal to 3.7cm. These values for the scenario 1 were equal to -45cm and 15cm respectively. Not only is the decrease of the error amplitude an improvement but more importantly, the final position is achieved with a very small error allowing a safe capture of the target point. The final end-effector position error in x-coordinate is equal to 0.37cm and in z-coordinate 0.008cm which is considered very low for this time of the precision maneuver.

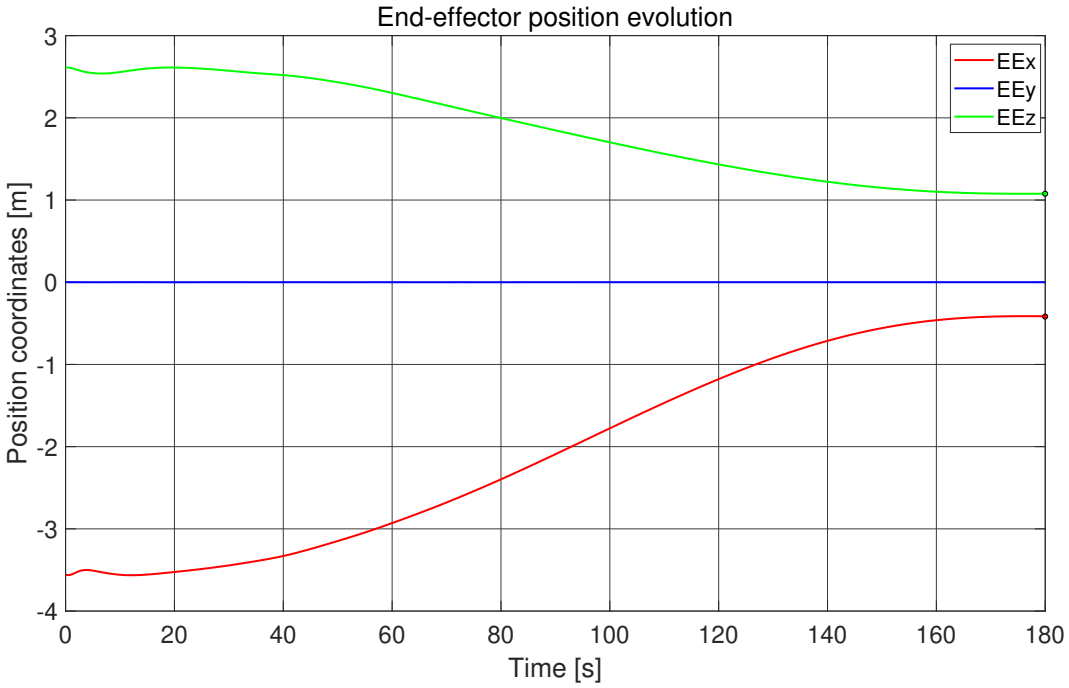


Figure 5.15: Scenario 2 - End-effector position coordinates in LVLH frame.

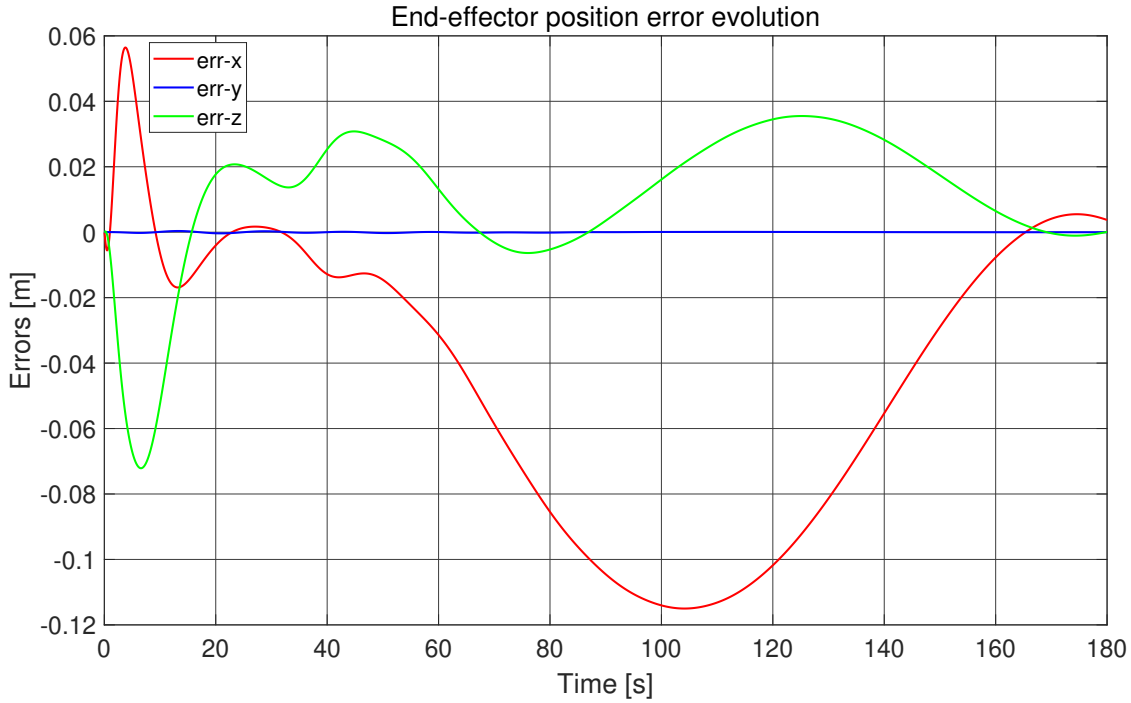


Figure 5.16: Scenario 2 - End-effector position error.

In figure 5.17, the contribution of each term from the cost function to the overall value of the function is presented in plots. With the increased weights, the absolute value of the FoV term in the initial condition is much higher than for the unit weight in scenario 1. The optimization solver tries to decrease this value as quick as possible, which is the cause of the initial oscillations in the motion seen in figures 5.11 and 5.12. In the second part of the motion, the highest value is of the EE cost due to the peak of the EEx position error. In figure 5.18 the overall value of the cost function is presented.

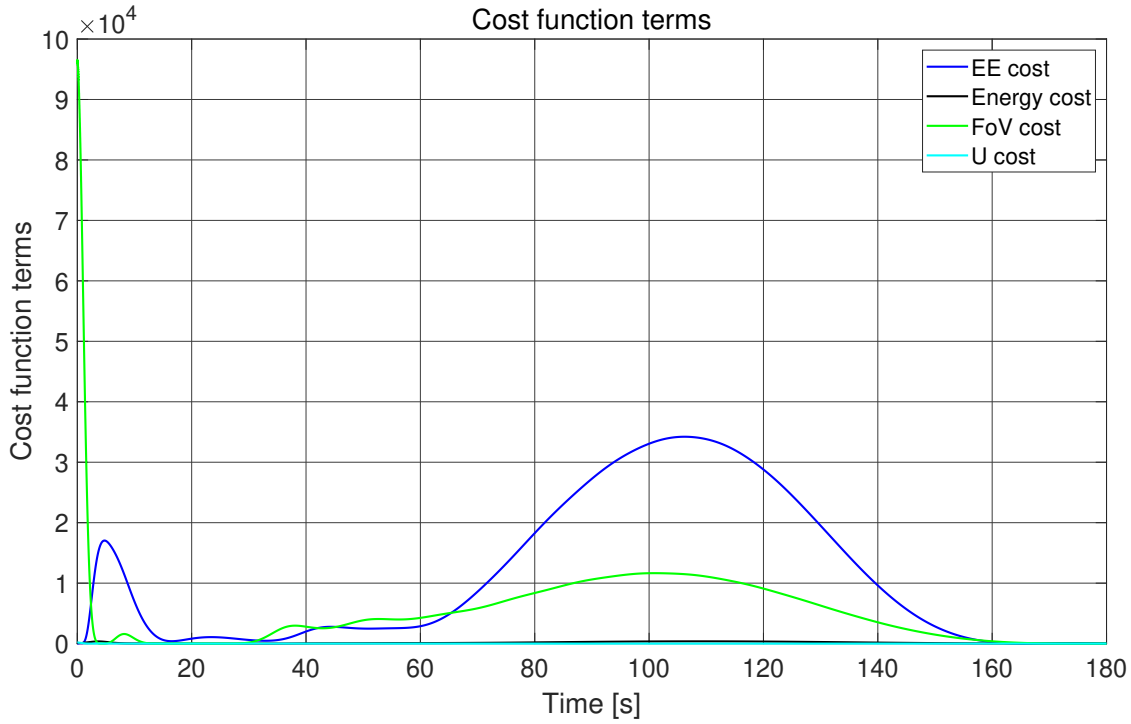


Figure 5.17: Scenario 2 - Cost function terms.

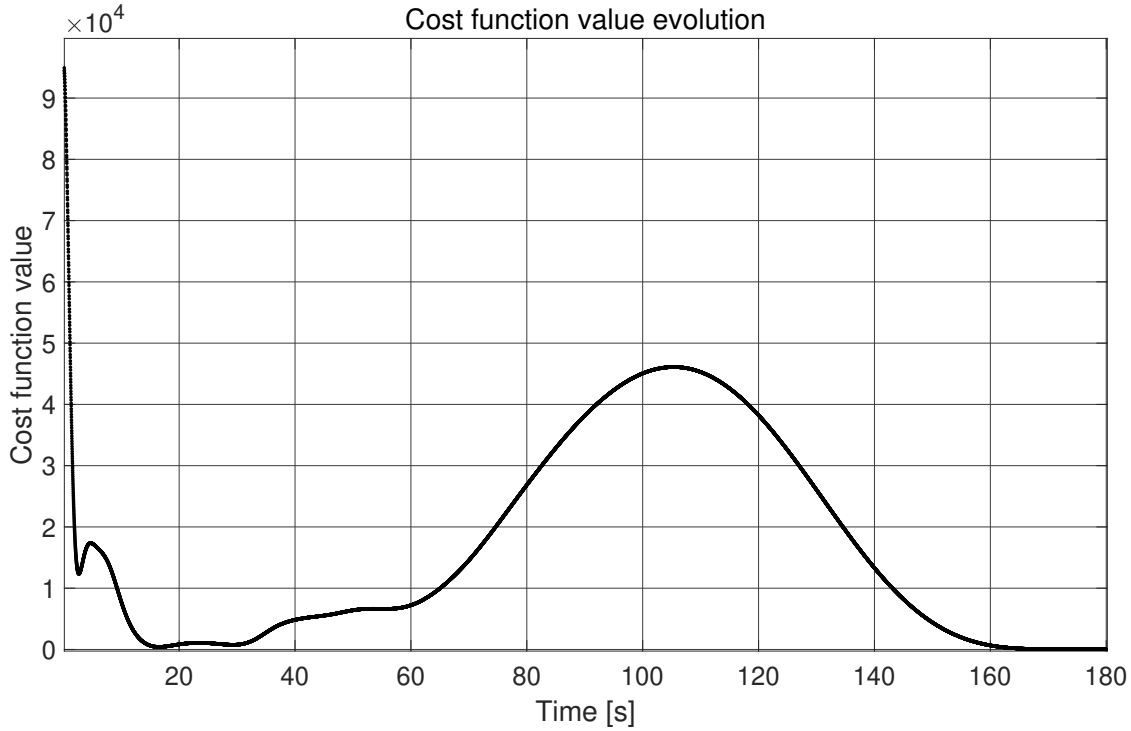


Figure 5.18: Scenario 2 - Cost function overall.

Last but not least, the final position can be visualized with the use of the aforementioned 3D model. Similarl to figure 5.10, the final position of the servicer spacecraft with respect to the target is presented below in figure 5.19. When comparing to the visualization of scenario 1, it is very clear that the target point is captured in case of scenario 2.

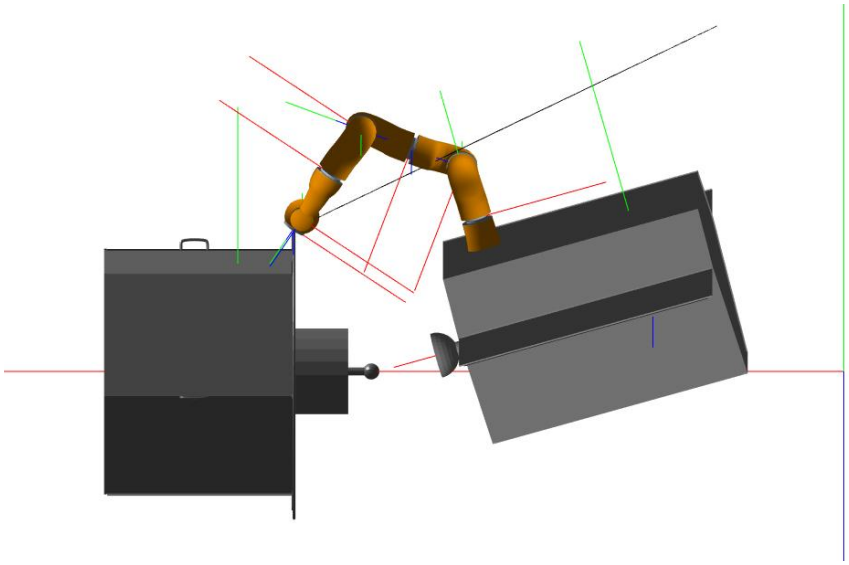


Figure 5.19: Scenario 2 - Final capture position from the simulation viewer.

5.1.3. Scenario 3 - Comparison of reference trajectories output

The performance of the controller and the achieved results depend not only on the tuned weights, but also on the characteristics of the reference trajectory supplied to the controller. A proper design and definition of the reference function is a task of the guidance block. It is usually defined as an optimization problem updated online based on the current measurements from the navigation. In this work however, the only focus is on the controller, therefore the reference trajectory passed to the control system is a synthetic data generated offline - before running the simulation with MPC controller in the loop. The reference trajectory in this work was generated in two distinct ways - performing interpolation with a linear function or polynomial function of the 5^{th} degree, as explained in the section 4.5. The difference between the outcome of the simulation run for these two reference trajectories with the input data of the scenario 3 (see table 5.1) are presented in this section.

Figure 5.20 presents the evolution of the motion of the servicer spacecraft base. The data related to the simulation run with the reference trajectory from the linear interpolation is shown in the purple plot, and the data from the interpolation with polynomial function is shown in the dark blue plot. In both simulations the weights have the same values and all the other parameters are equal. The difference between the two is very apparent. The linear velocity V_x of the 'poly5 traj' data reaches much higher amplitude and then decreases until it achieves zero value by the end of the simulation. It is clear that by the end of the maneuver the base fully decelerated in x direction. The V_x of the 'linear traj' keeps on growing during all the simulated maneuver. It doesn't reach as high amplitude, but by the end of the motion it doesn't change its growing behaviour and the simulation ends with the $V_x = 3$cm/s. It is a very high value and the failure to decrease the relative velocity of the servicer s/c base could potentially lead to collision. The evolution of V_y for both versions has small values in submiliter range and oscillatory behaviour with higher amplitude for the 'poly5 traj'. The linear velocity of the base in z direction V_z has similar evolution as the velocity along x direction. The 'poly5 traj' V_z reaches the highest amplitude of 2cm/s and then decreases until it achieves zero value by the end of motion. Clearly highest point of the 'linear traj' output data is two times smaller and it has a value of 1cm/s. The velocity settles around this value with a slight decrease by the end of the motion finishing the maneuver with the value of 0.95cm/s.

Comparing the plots for these two reference trajectories it is already clear that the main advantage of the trajectory generated with the polynomial function is that it forces the solver to find such solution of the optimization problem that by the end of the maneuver the motion is decelerated. It is very important from the point of view of the mission safety.

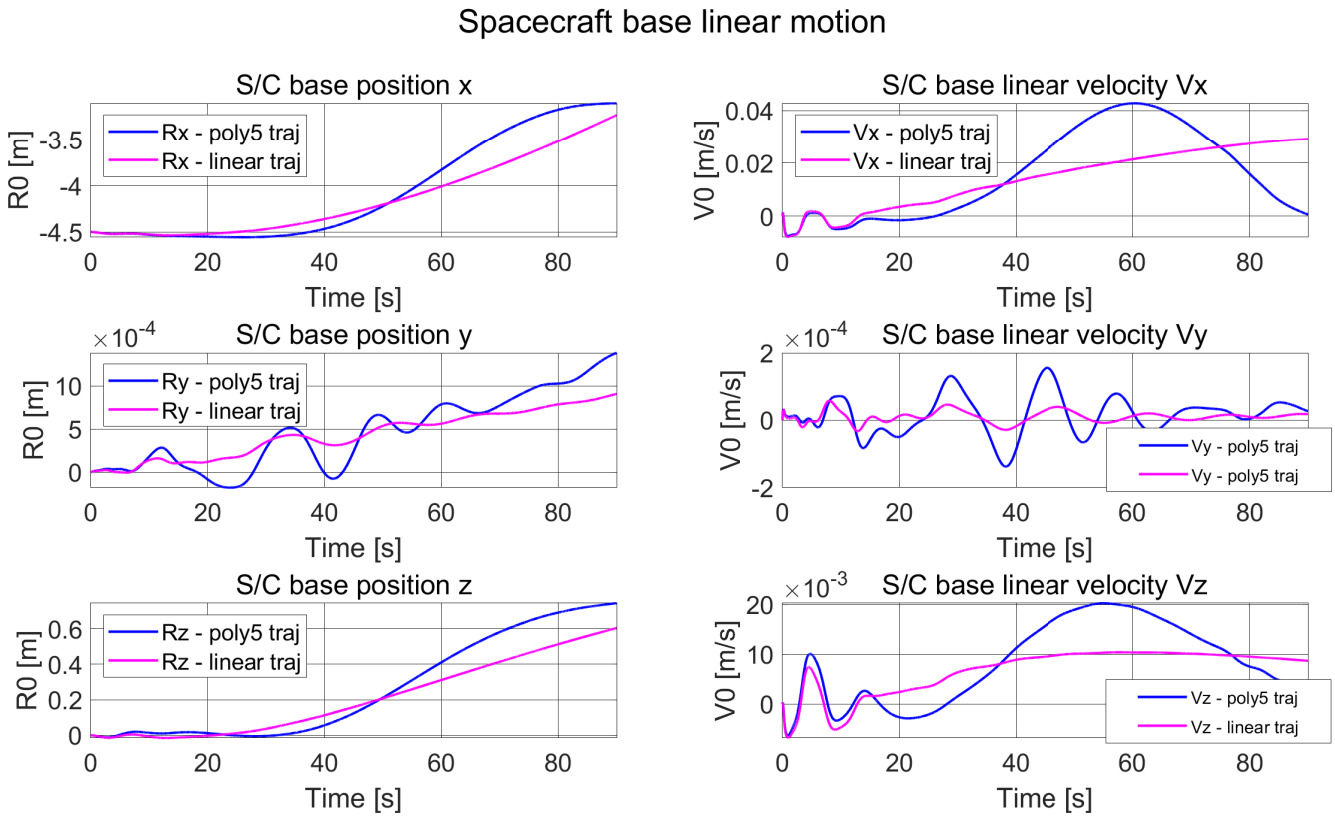


Figure 5.20: Scenario 3 - Spacecraft base position and linear velocity in LVLH frame.

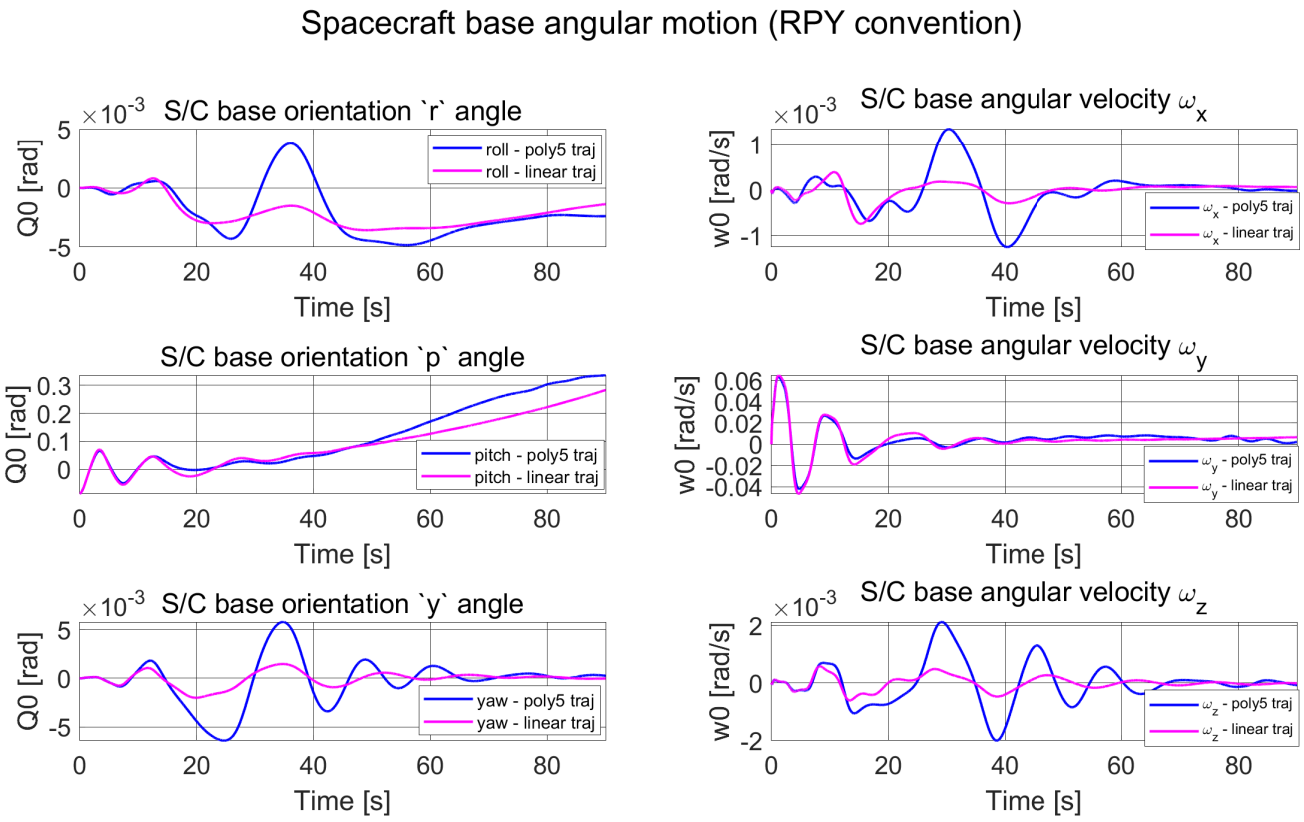


Figure 5.21: Scenario 3 - Spacecraft base attitude and angular velocity in body frame.

In figure 5.21 the angular motion of the s/c base is shown. Clearly the 'poly5 traj' data of the angular velocity in all three directions has more oscillatory characteristics with the largest amplitude differences in motion around x and z axes. The higher peaks compared to 'linear traj' however are not worsening the overall performance and stability as they still operate in very small range - the highest angular position peak of the 'poly5 traj' is for the yaw angle and is hardly equal to 0.005rad $\simeq 0.29$deg. For both types of trajectory data, the pitch angle (y axis) achieves the highest values compared to x and z axes with the final angular position of about 0.3rad $\simeq 17.1$deg.

Next, the motion of every active joint of the spacecraft robotic arm is shown in figure 5.22 below. The difference between the 'poly5 traj' and 'linear traj' data is very clear. The angular motion of the first joint q_1 is very small and remains in the subdegree range, with higher angular position amplitude for 'poly5 traj' simulation. The second joint q_2 angular position evolves from 45deg to about 50deg, the 'linear traj' angular position data evolution is roughly a linearly decreasing function while the 'poly5 traj' has largest angular position change in the middle phase of the simulated motion and in the initial and final phase it has more settled behaviour - for the first 35 seconds the position oscillates about the initial value, and for the 65 - 90 seconds it has the constant value of -1rad $= -57.8$deg. Last but not least, the third joint q_3 motion is clearly different for each of the generated reference trajectories. The $\dot{q}_3$ of the 'linear traj' has higher positive amplitude which makes the position increase faster compared to 'poly5 traj'. Next, it settles down and the angular rate value is about -0.01rad/s $= -0.57$deg/s beginning in 30^{th} second until the end of the motion. This angular rate evolution results in constantly decreasing function of the angular position from the 20^{th} second until the end of the motion. The 'poly5 traj' data of the third joint has different behaviour. The angular rate profile has a bit lower amplitude in the beginning, which makes the angular position increase more slow. In the second half of the simulated motion it has negative value, while in the last 20 seconds it has positive value. This behaviour results in an non-monotonic function, the angular position is decreasing in the interval of 35 to 65 seconds, and increasing in the interval of 65 to 90 seconds.

In figure 5.23 the plots of the end-effector position errors for evey coordinate in each simulation step is shown. The error was computed with the equation 5.1. Despite the larger oscillations of the motion of the coordinates respective to the 'poly5 traj' data, the final EE position is such, that the reference tracking error is much smaller than for the 'linear traj' scenario. The x error is equal to 5.4mm and z error equal to -2.8mm. The respective data of the 'linear traj' is x error equal to -15.7mm and z error equal to 7.8mm.

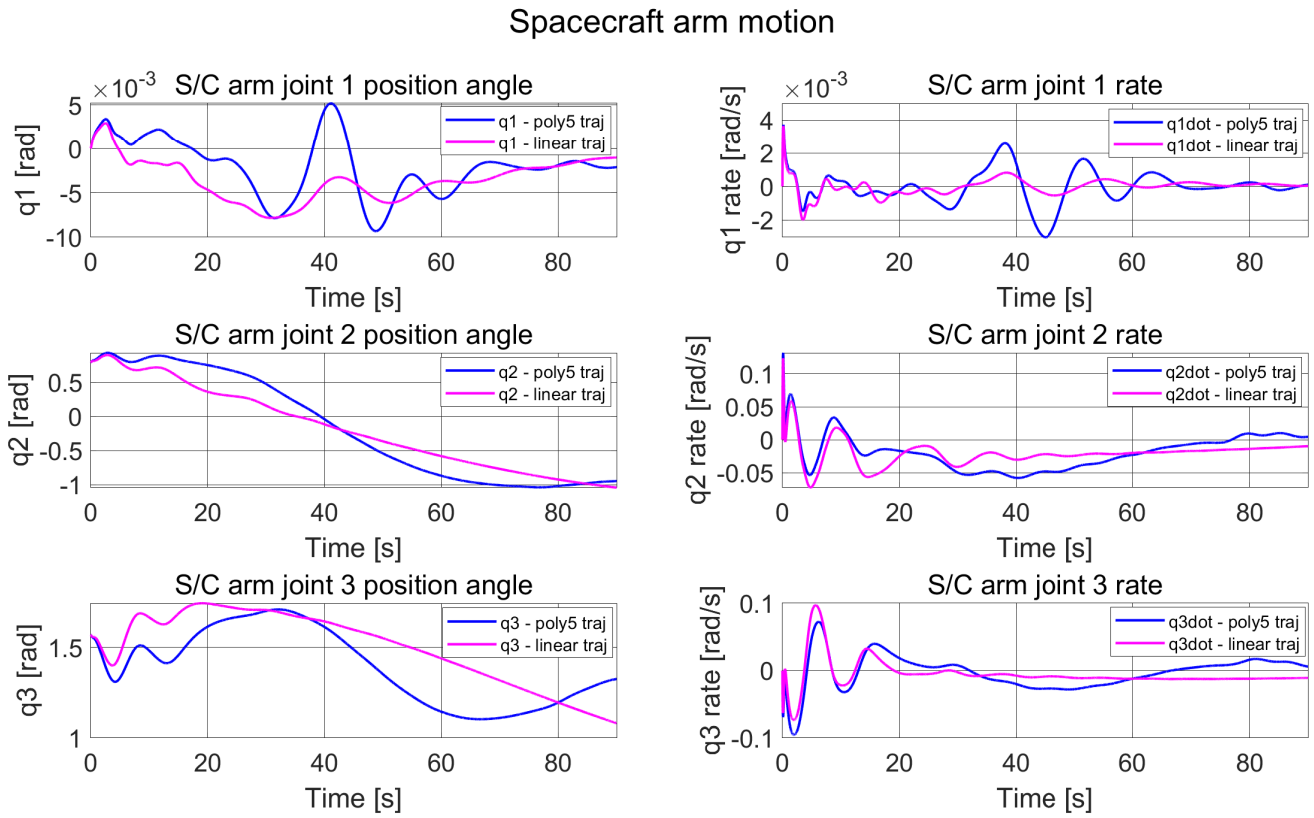


Figure 5.22: Scenario 3 - Spacecraft arm joints angular position and angular rate.

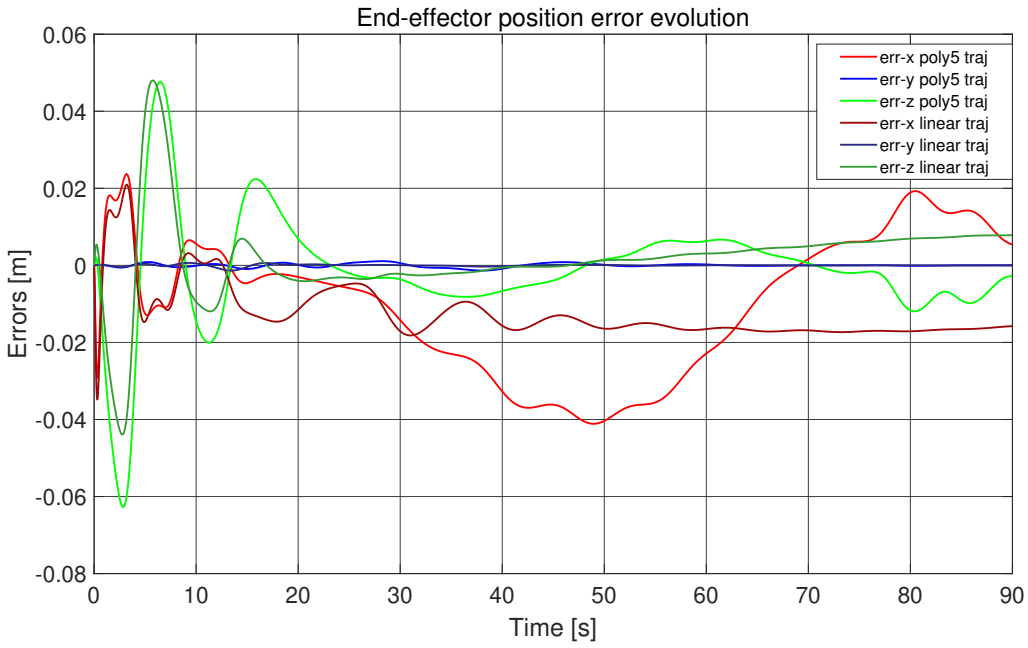


Figure 5.23: Scenario 3 - End-effector position error.

From the presented figures it is very clear how different definition of the reference trajectory can influence the overall behaviour of the dynamical system and the final reference position tracking performance. Due to the decelerating behaviour by the end of the motion, and better reference tracking performance, the 'poly5 traj' is advantageous and for this reason it was implemented for the other simulated scenario shown in this report.

5.2. Timing statistics

In order to analyse the designed MPC control system and simulation loop with the implementation of the quadratic programming solver, the timing statistics are considered. It is important to ensure that the required time by the solver, to perform the on-line computations and find the solution of an optimization problem, is lower that the sampling time. The timing statistics are crucial when considering the applicability of the designed system in the real space mission, in which the software is run on the onboard processor and it must be ensured that the optimization output is fed to the AOCS and Robotic management system with the required frequency.

This work fully simulation-based, which was coded with high level language, matlab. It cannot be directly implemented in real machine as it is, but there are few ways in which the designed controller could be implemented as an embedded code with the hardware-in-the-loop. Automatic generation of C code from matlab code, or the direct design of the simulation in C/C++ code are the two options. The low-level languages have another quality - the compilation time is faster than in high-level languages. For this reason, the analysis of the computation time for the control system designed and tested in matlab will allow to reach valid conclusions, if the computation time of the simulation run in matlab is acceptable, it can be concluded that the respective system run with C code will have similar or better performance.

In table 5.2 the time it took to perform specific computations is shown for each presented scenario. The solver time and yalmip time, is the extracted information that the solver returns in matlab. 'Solver time' concerns the total time it took the solver to find a solution of an optimization problem, therefore it is specifically the time required to find this solution by quadprog solver. 'Yalmip time' refers to time necessary to construct an optimization problem. As explained in Section 4.9, the YALMIP toolbox provides interface to facilitate the modelling, the yalmip-specific variable 'sdpvar' is used and all the components of the MPC problem, constraints, objective function, upper and lower bounds, are modelled. All the equations and mathematical formulas are defined before calling the solver, however some time is needed for yalmip to understand the matlab-based problem description and translate it in an understandable way for the optimization solver. Last, but not least, the value 'sim loop time' refers to the total computation time of the simulation loop with the MPC re-construction in every iteration, input of the control vector into the plant model, dynamics propagation and successive linearization around an operating point (see fig. 4.4).

Table 5.2: Solver computation time.

	Scenario 1	**Scenario 2**	**Scenario 3 (linear)**	**Scenario 3 (poly5)**
solver time [sec]	73.737	75.044	26.810	41.431
yalmip time [min]	34.131	34.269	7.815	11.724
sim loop time [h]	10.000	9.947	2.195	3.357

Firstly, it must be underlined that the units of each of the three time values are different, solver time is in seconds, yalmip time in minutes, and sim loop time in hours. Logically, the longest time is the 'sim loop time', however it was not expected to take matlab as long as it did to perform all the operations. The time compared to yalmip time is almost 18 times longer than the respective yalmip time for the scenario 1, which means that this computation time difference was required by the linearization of the dynamics nonlinear algorithm, FoV algorithm and FK algorithm, creation of the predictive model and the propagation of the dynamics with the use of the nonlinear plant model. It is difficult to say which aspect required the most computational time, the reason could be matlab, any limitations of the hardware system it is ran on (16 GB of RAM of the computer which was used for the simulations) or the dynamics model is too heavy for this application. If more time would be available, all these aspects could be investigated starting firstly with the transition from matlab to low-level language and then comparing the time statistics.

When it comes to the solver time values, they are very well below the maximum allowable value for all the scenarios. In scenario 1 the average solver time per simulation step is 0.0041 sec, in scenario 2 it is equal to 0.0042 sec, in scenario 3 (linear) it is 0.003 sec and scenario 3 (poly5) it is 0.0046 sec. It is clear that each of the values is less than the assumed sampling time between the simulation steps which is equal to $Ts = 0.01$sec. The respective average values of the 'yalmip time' are by one magnitude larger than the acceptable maximum for the scenario 1 and 2, 0.1138 sec for scenario 1 and 0.1142 sec for scenario 2. Scenario 3 'yalmip time' is shorter due to the shorter simulated maneuver time, however when looked into the average yalmip time for every iteration it is also lower - 0.0521 sec for the 'linear' case and 0.0782 sec for the 'poly5' case.

Moreover, the required number of solver iterations of every simulation step was looked at. If the optimization is performed on board, it is expected that the solver is able to reach the solution within finite number of iterations. In order to limit this number, the solver settings were set such that the maximum number of iterations is equal to 50. In figure 5.24 below, it is clear that the number of iterations solver took in every step was much smaller than the given threshold.

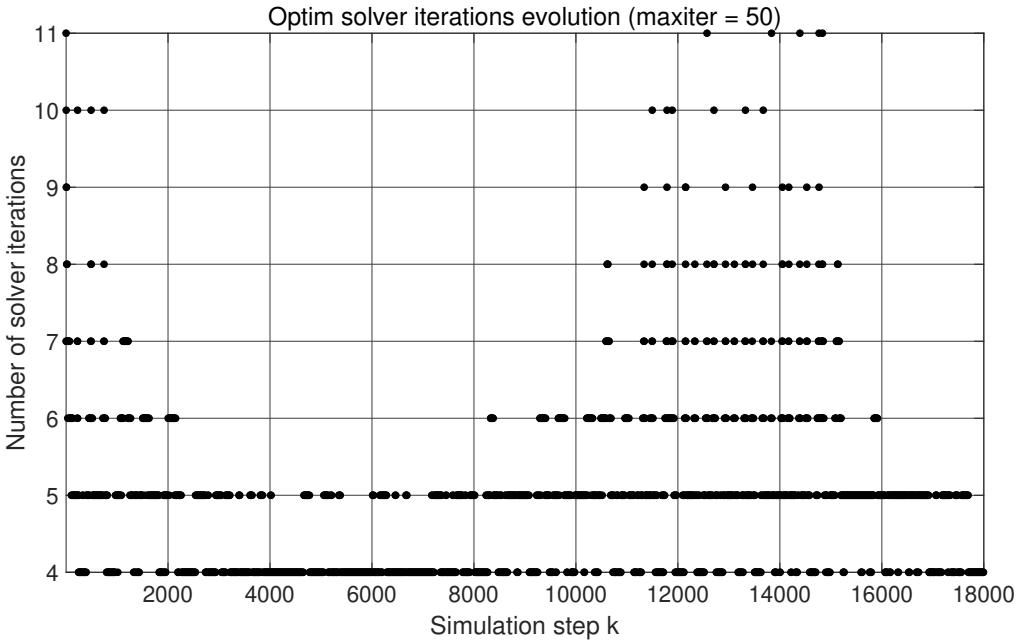


Figure 5.24: Solver iterations for simulated scenario 2.

In conclusion, it is very promising to see that the quadprog requires such little time to solve the optimization problem for every analysed scenario. It re-assures that the definition of the objective function and constraints is all-together well defined as a quadratic programming problem for being solved by the chosen solver. Another interesting point to investigate could be to compare other modelling toolboxes such as CVXGEN or Forcespro, they both support the quadprog solver solution, so the solver time should be rather expected to be similar however maybe it could resuls in shorter time required for setting up the optimization problem.

5.3. Communication with Ground

Last but not least, as it was explained in the introduction to the research (see Section 1) that in real on-orbit servicing space mission clearly the ground segment plays an active role during the reach phase. Although the maneuver is fully autonomous, the system still should provide collision avoidance maneuver (CAM) capability and contingency modes in case of any hardware or software failure. This could be done with designing an FDIR system - Fault, Detection, Identification and Recovery. These aspects are related to the development of on-board software and monitoring of the system. During all the mission phases, the ground engineers are supervising the mission progress. Firstly shortly after orbit insertion, LEOP phase takes places. During this phase operation engineers monitor and control various satellite subsystems to ensure all the modules are activated correctly, satellite appendages are fully deployed and orbit and attitude control maneuvers are tested. This is part of every space mission, and similar activities are carried out when transitioning to another mission phase.

Specifically in the context of on-orbit servicing mission, when the servicing spacecraft is already in very close proximity of another the collision risk is very high. This is the main reason why all the close proximity operations must be carried out carefully and a continuous communication with the ground must be kept established. During reach maneuver, the presence of reception antennas must be ensured such that the contact with the ground is continuous during all the required time. Ground engineers should be able to up-link the data and command the satellite to stop the maneuver and retract in any moment if such a situation would be required.

For this purpose, it is proposed to down-link the following information that could allow ground engineers to have a good insight into what is going on during the reach maneuver and interrupt it in case it is needed.

- Value of the cost function, with specific values for each term. Knowing the values of the weights and scaling factors and also the state vector from the sensors, the information about the error of each cost function term can be derived.
- Information from the rotary joints encoders about the measured position change, the estimated relative position of the chaser obtained by filtering the data from the visual sensors and the navigation sensors on board.
- Diagnostic data from the on board computer, telemetry data as in a regular other phase of a space mission. It allows to understand if not only the control system, but also all the other subsystems are working correctly during this phase.
- Low-level data from the optimization solver, such as e.g. the number of solver iterations required to get the solution.

Based on this data, and the position, attitude coming from the sensors it is possible to understand from ground what is going on with the system. It should be possible for ground engineers to activate these modes from ground:

- Collision Avoidance Maneuver,
- Retract from the reach maneuver,
- Update the reference trajectory and provide it to the controller,
- Communicate with other s/c subsystems if needed,

The communication link with ground segment could potentially be tested in a DLR facility, GSOC. Due to covid and limitations of the physical presence in the facility, this phase of the research project was discarded.

5.4. Simulation results and analysis

The designed control system was tested and run for many different scenarios. In this report only few of them were presented, which best serve the purpose of illustrating the tuning procedure and the final performance achieved. As is clearly seen in the plots showing the results of scenario 1, the achieved performance was not satisfactory and much better performance was achieved thanks to tuning process.

The final position of end-effector reached during the simulation of the scenario 2 data has very small error with respect to the reference position of less than 0.5 cm, which is a very good result. Nevertheless, it can be seen that the overall performance of the end-effector position tracking during the simulation could still be improved as e.g. in the figure 5.16. In this figure we can see that the largest error is achieved around t = 105 s and it is equal to -11.5 cm. It coincides with the highest amplitude of the Vz linear velocity of the s/c base and also the local minimum of the angular velocity of the third joint $\dot{q3}$. Clearly, the negative error indicates, that the current EE position is "behind" the reference position, therefore the overall motion of the end effector in x direction was slower than the synthetic reference velocity needed to precisely achieve the reference position. It would be interesting to further analyse this behaviour to understand if there is any correlation between the weight on the energy term from cost function applied on the velocity of the q3 and the EEx position error.

In order to compare the performance of different simulations, it is useful to define the erorrs indicating the accumulated reference tracking error throughout all the simulation. For this reason, the performance metrics are defined which are presented in table 5.3 below. There are two values computed - MAE, mean absolute error and RMSE, root mean squared error, which allow to better analyze the overall tracking performance of the controller during all the simulation. These values are computed using the error of the end-effector position with respect to the reference position of every k^{th} step of the simulation. The errors are computed separately for each of the three axes.

Table 5.3: End Effector position error metrics.

	Scenario 1	**Scenario 2**	**Scenario 3 (linear)**	**Scenario 3 (poly5)**
RMSE$_x[m]$	0.265422	0.062823	0.014377	0.020479
RMSE$_y[m]$	0.000173	0.000185	0.000262	0.000487
RMSE$_z[m]$	0.089285	0.023897	0.010753	0.014062
MAE$_x[m]$	0.211229	0.047481	0.013583	0.016279
MAE$_y[m]$	0.000139	0.000008	0.000144	0.000343
MAE$_z[m]$	0.077181	0.018744	0.006138	0.008733

Analysis of these metrics make it easier to formulate final conclusions. It is clear now, that not only the final end-effector position error is much better in scenario 2 compare to scenario 1, but also each of the two metrics for every axis have lower values for scenario 2. Especially the difference is seen when comparing the RMSE$_x$ and MAE$_x$ values. It is clear that the tuning allowed to drastically improve the tracking performance.

The comparison between scenario 3 for the reference trajectory generated with linear function and polynomial function was done in subsection 5.1.3, where it was clearly shown that the advantage of the polynomialy obtained reference trajectory is the decelerating capability by the end of the motion. However, from the table 5.3 above we can see that the reference tracking performance metrics are slightly worse compared to linearly obtained reference trajectory. The difference is is very small and, what is more, the end-effector position reached in the end of the simulation is very precise for both scenarios. For this case, the difference is very small and it is definitely advisable to already account for the required deceleration of the maneuver in the reference trajectory. It is clear now, that it comes with a certain cost of decreasing the reference tracking performance during all the simulated motion, however the achieved performance is still within the acceptable limits.

The results of the three presented scenarios can be analysed with respect to the performance requirements from the table 2.2 and table 2.1. In figure 5.11 we can see that the velocity is in the range of the values presented in the tables. Only V_x is exceeding the limit in the second half of the maneuver at t = 80-160 s. The angular rate of the spacecraft base during the maneuver is within limits for all three directions with exception of the ω_y in the first 10 seconds of the motion. In figure 5.12, in the plot "S/C base angular velocity ω_y" it is seen that in the very beginning of the maneuver there is some oscillatory motion and the highest peak is achieved at t = 3 s, with the value $\omega_y = 23 \times 10^{-3}$rad/s. The value of the angular rate in this time step is exceeding the limit 0.023rad/s $= 1.15$deg/s > 0.5deg/s however the peak quickly decreases and the value settles down within the range ± 0.3deg/s which is within the considered performance requirements. Regarding the tracking precision, the error of the end-effector can be considered. The performance criteria from the table 2.2 limit the allowable position error to 0.05m. The end-effector position error is shown in figure 5.16, it is clear that the x-coordinate

and z-coordinate have the highest contribution to the overall position error. This figure shows the position error in every time step with respect to the value from the reference trajectory look-up table. The error of x-coordinate reaches two maxima, at t = 5 s, it is equal to $err_x = 0.055$m and at t = 105 s, it is equal to $err_x = -0.12$m. The error of z-coordinate reaches two maxima, at t = 8 s, it is equal to $err_z = -0.071$m and at t = 125 s, it is equal to $err_z = 0.037$m. Clearly these values are exceeding the limits, nevertheless these two peaks occur more than a minute before the end of the maneuver and the final position accuracy is very good. In the end of the maneuver the respective errors are equal to $err_x = 0.005$m and $err_y \approx 0$m only.

Moreover, according to the table 5.3 and presented performance metrics, it is clear that the mean absolute error (MAE) for the scenario 2 is kept within the required limits. The highest MAE is for the x axis: $MAE_x = 0.0475$m and is only slightly below the accepted error value. With these values it can be concluded that the developed architecture in this tested scenario meets the control performance requirements in major part of the maneuver, however not entirely. The final position of end-effector is achieved with a very good accuracy.

6

Conclusions and recommendations for future

Final conclusions form the work are presented in this Chapter based on the methodology and results described in the report. The scope of the performed work was limited due to the thesis framework and its requirements. Therefore the critical overview of the achieved objectives, limitations and possible extensions of the project is presented. It is concluded that the major research objective was achieved with this work, and the recommendations for future are presented.

Firstly in Section 6.1, the Research Questions presented in Chapter 1 are analysed and answered based on the performed work and the results. The final conclusions are formulated based on the results of simulations presented in Chapter 5, experience with modelling of the system dynamics and definition of the control problem, and the throughout understanding of the available optimization and modelling software as well as the current state-of-the-art of the relevant literature. Finally in Section 6.2 the recommendations for the future work and potential extensions of the project are presented, which include majorly an improvement of the modelling, full GNC loop integration and testing in hardware facility.

6.1. Research conclusions

In order to formulate the conclusions, the research questions are repeated in this section with the critical overview of the achieved results and how they are linked to the objectives. The central research question steering the direction of the project was formulated as:

How can the performance of the reach phase of an on-orbit servicing mission be improved by implementing a combined controller with model predictive architecture?

The relevant current status of the worldwide research addressing the design of the optimal control problem of the robotic spacecraft was analyzed during the literature study period and it was concluded that the focus of the available work was mainly on the free-flying system (when the s/c base is not actuated) or on the formulation of guidance problem with MPC architecture, not control problem. Therefore, the research begun with the assumption that this work could potentially contribute to the current state of the knowledge related to the MPC control problem of the robotic spacecraft during proximity operations with the active actuation of the satellite base. In order to understand how the MPC control system shall be designed, the workflow was broken down into sub-goals to be achieved. Subsequent list of follow-up sub-goals was proposed that allowed to break down the research problem into the list of objectives to be achieved to finally lead to the main research question. The accomplishment of all sub-goals implies that the main goal is met.

In order to provide the control methodology to perform on-orbit servicing maneuver, first, the system of interest (to be controlled) must be properly defined. In case of the application of the model predictive control algorithm, it concerns the definition of both, the dynamics of the plant and of the predictive model. Both of them are not necessarily equivalent, which is also the case of this research project - the plant model is a nonlinear continuous dynamics system, whereas the implemented prediction model is a linearized and discretized system. With respect to this task, the first sub-goal was formulated as:

1: *”What model of kinematics and dynamics of the system composed of the arm and of the spacecraft shall be developed and implemented in the controller design?”*

The answer to this question is not trivial. First of all, proper description of the multi-body dynamics system in general is a complicated task. Moreover when the body is in the orbital motion with respect to another satellite body, the dynamics description becomes complicated. The approach taken in this project included certain simplifications.

First of all, the question of how to approach the modelling of the free-flying system had to be answered. The control system was developed in Matlab (due to prior personal experience), therefore the plant dynamics definition had to be modelled such that it is compatible with the control loop presumed to be defined in matlab. In the modelling phase many approaches were targeted before the final model was achieved. The model could be either designed in Simulink or scripted in Matlab. In order to maintain clarity of the design process, firstly the simplifed two dimensional model was defined. It consisted of the point mass with 2-link manipulator. The system's dynamics were described directly in matlab script with equations of motion derived from the Langrangian of the system. The equations of motion derived in this way take a very complicated form with every additional degree of freedom, which also makes it very prone to human error and further increases the challenge of proper validation and verification. For this reason, the available toolboxes that could enable modelling of the orbital free-flying robotic system were looked at. The SpaceDyn was chosen, which is a free available toolbox consisting of the matlab scripts published by Japanese research group from Tohoku University [69]. The advantages of using the toolbox are: a very straightforward definition of the multibody system which is the baseline of the dynamics definition, clear unproblematic integration with matlab and optimization solvers due to its easy structure consisting purely of matlab scripts and . The detailed description of the toolbox features were analysed, as described in Section 3.2, and it was concluded that SpaceDyn capabilities shall meet the requirements of the project. Thanks to these features the toolbox eases the modelling process - there is no need to re-invent the wheel if the part of the work can be based on the available toolboxes. As the project boundaries were clearly defined, and the designed controller is to be applied specifically for the autonomous control of the robotic s/c during reach phase, some simplifications were applied. The far rendezvous approach phase was not considered, and all the reach phase is performed in a very close proximity of the target spacecraft. It was assumed that both chaser and target spacecraft in the beginning of the phase are moving in the same orbit with only a very small difference of true anomaly. The main simplifying assumption was to consider the target body reference frame as an inertial frame in which the relative motion of the chaser spacecraft was defined. This assumption can be made for the very short maneuver only, because otherwise the orbital dynamics must be considered. In order to verify the definition of the dynamics and kinematic (forward and inverse) the multibody system consisting of the main base and two links with revolute joints were firstly defined and tested. Next, the model was extended with third link, and again validated. The application of SpaceDyn toolbox in the modelling process in general is very clear, as all the scripts are well defined and it is very straightforward to trace back dependencies of the functions and variables.

Another assumption applied during the modelling followed from the design choice that the combined control system, subject of this research work, will operate on high level only. It means that the objective of the combined control system is to perform control allocation within all the available actuators - thrusters, reaction wheels and joint actuators without specifying which thruster of all 24 available is actuated and what precisely is the input (value of the voltage or current signal). The force acting on the spacecraft base was assumed to be applied on the c.o.m. of this body link, the same applied for the torques inducing rotational motion of the base. This clear definition of the system boundaries determines the requirements of the level of the detail of the dynamics description of the plant model. Given the fact that the designed controller would operate on high level only, there was no need to consider dynamics of the actuators itself. In case of the full GNC design and detailed controller design, the lower level system solving optimization problem concerning thrusters allocation and reaction wheels allocation could be designed. Moreover, the multibody system considered in the final simulations consisted of 3 manipulator links. In further phase of the project it would be interesting to see how the system consisting of more active manipulator joints, and hence more links, is behaving. Last but not least, it is important to mention that the considered reach phase does not include the final hard capture itself. For this reason, the contact dynamics were not considered.

Next, in order to bring the designed control system as close to the real space mission conditions as possible, the constraints of the system had to be accounted for. In order to identify them, the second sub-goal was formulated as:

2: *"What constraints, relevant for the reach phase, shall be taken into account during the design of the controller?"*

These constraints are described in Section 2 in terms of spacecraft architecture constraints, operational mission constraints, environment and the reach/capture task control limitations. Then the constraints related to the s/c architecture itself, such as upper and lower bounds on the control signals were formulated mathematically in order to apply them in the MPC controller. They were described in Section 4.7. The most important constraint is the upper and lower boundary on the allowable control input values. Moreover, due to geometry of the plant, and especially the robotic arm, it must be ensured that the manipulator motion will never be such that a self-collision will occur. In other words, we do not want to crash with the target spacecraft and also must ensure that there will be no collision between bodies of the multi-body dynamic system. For this reason the constraints limiting the allowable joint angular positions were introduced, such that in case the joint position have either positive or negative extreme value from the possible boundary range, they will neither collide one with another nor with the s/c base.

Once the plant dynamics and the system constraints were well understood, the first approaches towards the

MPC controller design were made. The general design of the controller and choice of the most suitable architecture were addressed within the third sub-goals formulated as the following question: *"Which type of the Model Predictive Control (MPC) strategy shall be implemented?"*. This question was further broken down into sub-questions to facilitate the direction of the research and ensure all the most important aspects of the controller design are considered. The questions related to this sub-goal are repeated below and each of them is commented based on the presented research work.

3 (a): *"Which type of the MPC is the best candidate to account for different frequencies, performance specifications, admissible uncertainties of the both subsystems (AOCS and the robotic manipulator)?"*

Based on the literature study of the characteristics of on-orbit servicing missions and requirements, the critical overview of the possible MPC types was made in order to identify their advantages and disadvantages with respect to the specified application of combined controller. The MPC choice trade-off table presented in figure 4.1 identified the major types of MPC considered for the project. Firstly, different types of dynamics prediction model were looked at, there are three different possible definitions LTI, LTV or NL. The nonlinear dynamics (NL) definition uses the most accurate representation of the plant model, the predictions should be the most accurate compared to linearized description. On the other hand, it comes with a lot of challenges when it comes to the MPC design, its validation and overall the system could result too complex. Moreover, the use of NL dynamics is directly affecting the possible type of an optimization problem, such that usually with NL dynamics a non-convex optimization problem is achieved which is not very desirable. Non-convex problem can find solution in local minima instead of global, and can have higher resulting computational complexity. In general, always convex optimization problems are much more desirable, for this reason linear models LTI and LTV were also considered. The LTI is the easiest most handy definition, it simplifies the description of the prediction model and usually leads to convex optimization problem. However there are certain limitations coming from it, such that if only one model is used during all the reach phase, the accuracy of the prediction model with respect to the real system is decreasing because the equations of motion of free-flying space robots are dependent on the configuration, s/c orientation and manipulator joint angles, in current steps. This poses a very big limitation on the possible achievable performance of the system. To counteract it, another option is to update a linear model by performing linearization of the nonlinear dynamics around an operating point of the current position. Such approach effectively results in a linear time-varying model (LTV), therefore the advantages of the linear prediction model with respect to the convex optimization problem type are kept, whereas the model accuracy is much better compared to LTI.

When designing the simulation with controller in the loop, if we want to account for different operating frequencies of the AOCS and robotic system, it can be done by setting different frequency of the update of the current position of the manipulator joints, and of the s/c base attitude to the block performing linearization of the dynamics. The nominal design of the control in the loop in this project considers sampling time equal to 0.01 sec and is equal for both subsystems. It was assumed that we have the perfect knowledge of the state vector in every simulation step, and the possible disturbances and signal noise that lead to inaccuracies of the model are somehow accounted for when doing linearization procedure.

3 (b): *"Why is this architecture the best candidate for potential implementation?"*

During the trade-off of the MPC type, factors taken into account were: simplicity/complexity, model fidelity, optimization aspects, applicability to RICADOS free-flying system and the coverage in literature, see section 4.1. For reasons explained in that section and also in the paragraph above, it was concluded that the MPC problem shall have linear prediction dynamics updated in every simulation step based on the current state vector. It results in a linear time-varying model, which allows to keep a good level of model fidelity and description accuracy despite a time-varying configuration of the system. Moreover, the safety concerns related to close proximity operations rather require the design of a control system such that its behaviour is possible to be verified, validated, and the proposed control signal is a unique solution. For these reasons it is important to construct a convex optimization problem. This is achieved in the easiest way by constructing a quadratic programming problem.

3 (c): *"Shall the optimization of the control effort be performed on-line or offline, what are the advantages and disadvantages of each of them?"*

This question was addressed following the trade-off, as presented in Section 4.1. We can differentiate between the explicit and implicit MPC definition. The first concerns the division of the optimization problem into critical regions, construction of sub-domains and solution of an optimization problem offline, whereas the latter concerns online solution of an optimization problem. If the computing power is very limited, the explicit MPC is very advantageous for some systems as it allows to solve optimization problem a priori, and during the controlled motion only the reference input signal values are read form the look-up table based on the feedback law. However, for very

complex dynamics of free-flying space robot, the proper division into sub-domains while keeping required level of model accuracy is very difficult, the complexity of this task increases for higher dimensionality of the problem. Moreover, the larger number of the critical regions, the higher is memory requirement. Due to the fact that there are many solvers available allowing for the real-time computations and due to the complexity related to proper design of explicit MPC, it was concluded that much more interesting option is an implicit MPC.

3 (d): *"How can the MPC architecture be implemented?"*

When the conclusions were reached about the type of MPC problem, the implementation method had to be identified. The overall simulation with control in the loop is graphically presented in figure 4.4. As mentioned earlier, the work presented is fully based on the matlab scripts. The MPC architecture was designed as an NMPC problem with successive linerization. In this way the linear dynamics model is constructed in every time step based upon the received feedback information of the current state of the plant. The Jacobian linearization procedure was applied, as explained in section 4.3. In order to construct a quadratic programming problem, the cost function terms were defined as quadratic terms and constraints as linear functions, see Section 4.6 . The construction of the optimization problem was formulated with the use of a modelling engine YALMIP. It is a matlab toolbox for rapid prototyping of optimization problems making the optimization modelling much easier, as explained in Section 4.9.

In this stage, the plant dynamics were defined, the MPC design choices were throughoutly analysed and finally all the aspects were combined together in order to provide a closed-loop system with the controller and plant model and simulate the OOS reach maneuver. This objective is entailed in the fourth research sub-goal defined as follows: *"How should the combined controller of the system be designed with the chosen MPC architecture?"*. It refers not only to MPC design choices but also to the proper integration of the controller with the simulated dynamics, definition of the reference data and performance assessment necessary for the validation of the system applicability for this scenario. In order to identify these aspects, the fourth sub-goal is broken down into sub-questions commented below.

4 (a) *"What simplifications shall be taken?"*

Some of the assumptions had already been taken when considering the best MPC type and its further implementation methods. Moreover, the dynamics modelling process was also based on assumptions. When it comes to the general approach towards the integrated simulation with controller-in-the-loop, other aspects had to be simplified. Firstly, the input data constructed before running the simulation (graphically presented in figure 4.4) includes parameters that were assumed to be constant during all the simulation motion. They include the following: the weights and scaling factors are pre-defined and kept constant, the time parameters are also defined during data initialization phase and they include the total maneuver time, constant sampling time equal, constant frequency of the state vector feedback updates. If the project was to be continued, these aspects could potentially be changed such that e.g. the total maneuver time is not pre-defined but depends on the achieved accuracy of the end-effector position, if the final position is not close enough to the reference position more iterations of actuating the control system shall be possible. Also, if the project would have longer timeline, the simulations could be run with different sampling times. Clearly, the lower the sampling time, the higher the number of iterations required for the same total maneuver time. It could significantly affect the computation time though.

Next, in order to simulate the maneuver with the designed controller in the loop, the reference trajectory of an end effector must be available. The main objective of the MPC cost function is to follow the reference trajectory which is normally an output from the guidance block. As the full GNC subsystem design was not within the scope of this project, but control system only, the generation of the reference trajectory was simplified - based on the known initial position of end-effector and desired final position, a simple interpolation function between two points was applied in order to obtain look-up table with cartesian coordinates of EE position for every time sample of the simulation, see Section 4.5. It is clear that with this approach, the reference trajectory is fully defined offline. It has significant limitations, as there is no update capability of the reference trajectory based on the real position. If the full GNC system was simulated, there should be a feedback loop not only between the navigation and control block, but also guidance which would allow for the real-time updates of the trajectory based on the knowledge of the position and attitude from the navigation system. Also, the definition of the trajectory in GNC systems is usually defined as an optimization problem, and not a simple interpolation function.

Last but not least, it is assumed that the output of the propagated dynamics which is a state vector, is directly supplied to the NMPC successive linearization block in every simulation step. It is based on the assumption that we have the perfect knowledge of the plant dynamics and the perfect sensors, which in reality is very rarely the case. However, this assumption was made here and no additional disturbances are added explicitly, because already the linearization procedure introduces some mismatch between the plant model and prediction model which brings the controller closer to real conditions.

4 (b) *"How can the implementation be verified?"*

Firstly, all the components of the MPC optimal control problem were verified: both the dynamics definition and the quadratic optimization problem. Some of the quadratic terms contain nonlinear expressions which were linearized - it was the case of the end-effector reference position term and field-of-view term. The validation of the Jacobian linearization procedure was performed and the results of the validation were presented in Appendix B and C. The controller and feedback loop design was an iterative process. Whenever a new functionality was added or some numerical values were changed, the simple scenarios were run to check if the overall system still behaves in an expected way after the introduction of a new feature. To give an example, when reference trajectory term within the objective function was defined firstly it was constructed in joint space. The joint reference position for every active joint was supplied to the controller. In this way the reference tracking capability of the system was validated. The reference trajectory in joint space is very straightforward to follow, moreover it does not allow the solver to choose the best manipulator configuration itself to meet the required position of end-effector. In order to give more freedom to the solver, in next step the reference tracking task was formulated in task space. The forward kinematics algorithm which maps the positions in joint space (as in the state vector) to cartesian positions of EE in task space was linearized. It was verified, running both the linearized and original non-linear algorithm for the same input data.

Next, upon running first simulations, it was observed how change of the cost function value is steering the behaviour of the system in order to validate that the objective function was defined correctly and it is doing what it is expected to. The output data from the solver such as the number of solver iterations, the value of cost function and the solver time were looked at and helped to assess if this definition of the MPC could potentially be applied in real-time systems (see Section 5.2). Last but not least, the motion was visualized in real time by injecting the updated state vectors into the 3D visualisation of RICADOS system, which is a very convenient way to understand behaviour of the simulated motion next to data plots analysis.

4 (c) *"What case studies shall be tested?"*

There was a variety of case scenarios run during the design process, as explained in the paragraph above. It was the most time-consuming part of the research work - to critically look at the designed system, gradually add functionalities and test the implemented changes. Since these simulations contributed to verification and validation of design choices but not to the final evaluation of the performance of the designed control system, they were not included in the final report.

The presented results of the case studies in Chapter 5 aimed to show:

- how the final performance depends on the right definition and tuning of the optimal control problem,
- what is the reference tracking error to understand if it stays within acceptable limits for reach phase of OOS mission or not,
- whether the controller is able to compensate the initial attitude offset of the chaser s/c base,
- whether the evolution of the proposed inputs values trajectories is smooth and allows getting satisfactory results within the limits.
- to what extent the controller performance depends on the quality of the reference trajectory (normally outcome of the guidance block)

Scenarios described in this report concern three cases. Firstly the unit weights were applied for each term from the objective function to show the performance of the nominal design before the tuning procedure. Next, the multi-iterative tuning process was carried out in order to find the numerical values of the weights to be used that allow to get better performance. Last but not least, two different methods to get a synthetic reference trajectory were applied, both of them are very simplified approaches. As the proper design of the guidance capability is yet another vast research topic, the supplied reference trajectory of the end-effector was computed as a point-to-point trajectory from the initial position of end-effector to the final grasping point using two different interpolation functions. The goal was to see to what extent the overall performance of the controller depends on the quality of the reference trajectory data.

In all these scenarios the initial conditions were the same, the initial attitude pitch angle offset was considered so as to see if the controller is able to compensate it during the simulated maneuver. The relative position of the chaser s/c and the target in the beginning of the simulated maneuver was chosen to be 4.5m away, such that it closely resembles the initial conditions for the beginning of the reach maneuver in a real On-Orbit Servicing mission.

The results presented in Chapter 5 show that indeed the performance of the controller is vastly dependent on the values of weights and hence tuning procedure. The proposed definition of an optimal control problem is very sensitive to these numerics, and therefore also not very robust for the much different initial conditions. With every

meaningful change of the initial state vector, or change of the upper and lower boundaries of the allowable control inputs, the tuning procedures must be repeated. It is not strictly an indicator that the proposed control system is not applicable for OOS mission. Quite opposite, if the testing campaign would be further continued, it would be a good idea to identify the subsets of all possible initial conditions and the respective values of the weights. Moreover, in a real mission, while the chaser spacecraft is in the hold point and it is in stationary position relative to the target, the communication with the ground can occur for as much time as needed. The only constraint is the orbital position due to the required sun illumination for performing maneuver. In such a way, the exact position of the chaser is known thanks to the sensors data and the weights can be tuned on the ground by engineers and up-linked to the satellite on-board computer. This aspect is elaborated on in the next research sub-question.

All in all, the chosen case studies showed that the designed control system, if tuned properly can give a very good tracking performance with the end-effector final position error not higher than 4 mm.

4 (d) *"How the communication interface with ground can be accounted for?"*

The proposal of the way to do it was described in Section 5.3. It concerns specifically the information about the progress of the maneuver and solver low-level data that could be down-linked to the ground in order to provide information in real-time about the situation to operation engineers. Unfortunately due to a limited time and also limited access to facilities the developed software was not implemented in the machine. Therefore, it could not be tested in the end-2-end simulator, in the framework of RICADOS project where the link between ground segment and space segment is artificially established such that the communication can be simulated (see figure 1.3).

4 (e) *"Does the selected method and developed architecture meet the control performance requirements?"*

The design process of the control system for a robotic spacecraft was quite complex and involved many design choices, assumptions and simplifications. The developed method was majorly based on the design choice to construct the objective function as a quadratic program in order to ensure that there is only one optimal solution. All the nonlinear algorithms were linearized for this reason. It would be interesting to compare the results with another model predictive control architecture for the similar control system.

The results of the three presented scenarios were analysed with respect to the performance requirements from the table 2.2 and table 2.1. In Section 5.4, the achieved results were compared with the values of position, velocity, attitude and angular rate allowable errors from the literature. It is clear that overall, the developed architecture meets the control performance requirements in major part of the maneuver, however not entirely. The final position of end-effector is achieved with a very good accuracy, which is the most important criteria, only then the capture maneuver can begin.

This research work would benefit form further extensions and running more simulations of different scenarios. All in all, the presented work proved it has a very good potential for further improvements. The results achieved are very satisfactory as for the current development status.

6.2. Further recommendations

The technology development of Guidance, Navigation & Control system capabilities tailored for the On-Orbit Servicing and Active Debris Removal missions will certainly continue to grow. This research project aimed to make a contribution to the technology for autonomous control of a robotic spacecraft mission in the close proximity phase. Achieving the results presented herein required a thorough understanding of the theoretical system based on the literature review, many attempts at the iterative design process and appropriately tuning the weighting matrices. Still, the presented work was based on simplifying assumptions and would require more detailed analysis to achieve more accurate results. It would be interesting to continue this project and e.g. compare it with the controller developed within COMRADE project where the H_∞ architecture was implemented [31]. Also, the full GNC loop could be developed such that the guidance capability is not as highly approximated as it was done in this project. A proper feedback loop between all the three components could be established, such that during the maneuver there is a possibility of updating a reference trajectory online if needed. First of all, a variety of case scenarios should be further tested to identify if the proposed control system is performing well with much different initial conditions, e.g. when the chaser initial position is not perfectly aligned with the x-axis of the target. Next, the software integration in the OOS-SIM facility for the hardware-in-the-loop tests could provide a valuable information of how the controller behaves in this simulated environment. Due to limited accessibility to these facilities this phase of the work was not conducted.

In summary, the most interesting aspects for further development could include:

- Integration in RICADOS simulator, performing hardware-in-the-loop testing campaign and simulating the link with ground segment.
- Improvement of the modelling - sloshing fuel, specific geometry of the base, sensors uncertainty. Also model disturbances due to sensors noise, measurement uncertainties and changing mass of the system due to the fuel consumption.
- Add more detailed control allocation feature, such that the force and torque acting on the base is defined for each actuator instead of considering the generalized force/torque acting around the center of mass.
- Add contact dynamics formulation and consider capture maneuver. It could be required to design a second controller for the hard capture and switch between them when the reach phase is finished and the hard capture begins.
- Design a low level control, such that the manipulator joint dynamics are modelled and the specific current signal to be applied to each joint is computed.
- Development of a complete GNC loop and integrating this predictive combined controller into it. Allowing the online update of the reference trajectory and communication with ground.

References

[1] ESA Space Debris Office. *ESA's Annual Space Environment Report*. Tech. rep. GEN-DB-LOG-00288-OPS-SD. ESA, 2022.

[2] Donald J. Kessler et al. "The Kessler Syndrome: Implications to Future Space Operations (AAS 10-016)". In: *Rocky Mountain Guidance and Control Conference; 47-62*. 2010.

[3] Andrew Tatsch, Norman Fitz-Coy, and Svetlana Gladun. "On-Orbit Servicing: A Brief Survey". In: *Proceedings of the 2006 Performance Metrics for Intelligent Systems Workshop*. 2006, pp. 276–281.

[4] Goddard Flight Space Center. *On-Orbit Satellite Servicing Study Project Report*. Tech. rep. NP-2010-08-162-GSFC. NASA, 2010.

[5] Chair of the Working Group on the Long-term Sustainability of Outer Space Activities. "Guidelines for the Long-term Sustainability of Outer Space Activities". In: *United Nations Committee on the Peaceful Uses of Outer Space*. 2018.

[6] ESA General Director and Executive Board. *ESA's Technology Strategy, version 1.1*. Tech. rep. ESA, 2019.

[7] *Canadarm, Canadarm2, Canadarm3 - A comparative table*. URL: https://www.asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp (visited on 04/29/2020).

[8] Angel Flores-Abad et al. "A review of space robotics technologies for on-orbit servicing". In: vol. 68. Elsevier, Mar. 2014, pp. 1–26. DOI: http://dx.doi.org/10.1016/j.paerosci.2014.03.002.

[9] G. Hirzinger. "ROTEX — The first space robot technology experiment". In: *Experimental Robotics III*. Ed. by Tsuneo Yoshikawa and Fumio Miyazaki. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 579–598. ISBN: 978-3-540-39355-9.

[10] G. Hirzinger et al. "ROTEX-the first remotely controlled robot in space". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. Vol. 3. 1994, pp. 2604–2611.

[11] Heike Benninghof et al. "RICADOS - Rendezvous, Inspection, Capturing and Detumbling by Orbital Servicing". In: *7th International Conference on Astrodynamics Tools and Techniques*. 2018.

[12] Maximo A. Roa et al. "Robotic Technologies for In-Space Assembly Operations". In: *14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*. 2017.

[13] Thomas Uhlig, Florian Sellmaier, and Michael Schmidhuber. *Spacecraft Operations*. Cambridge Aerospace Series. Springer, 2015. ISBN: 978-3-7091-1802-3.

[14] G. Hausmann et al. "E.Deorbit Mission: OHB Debris Removal Concepts". In: *ASTRA 2015 - 13th Symposium on Advanced Space Technologies in Robotics and Automation*. 2015.

[15] M Shan, J Guo, and E.K.A. Gill. "Review and comparison of active space debris capturing and removal methods". In: *Progress in Aerospace Sciences* 80.January (2016), pp. 18–32. ISSN: 0376-0421. DOI: 10.1016/j.paerosci.2015.11.001.

[16] Wei-Jie Li et al. "On-Orbit Service (OOS) of spacecraft: A review of engineering developments". In: *Progress in Aerospace Sciences* 108 (2019), pp. 32–120. ISSN: 0376-0421. DOI: https://doi.org/10.1016/J.PAEROSCI.2019.01.004.

[17] C.J. Dennehy and J.R. Carpenter. *A Summary of the Rendezvous, Proximity Operations, Docking, and Undocking (RPODU) Lessons Learned from the Defence Advanced Research Project Agency (DARPA) Orbital Express (OE) Demonstration System Mission*. Tech. rep. NASA, 2011.

[18] K. Landzettel et al. "Robotic On-Orbit Servicing - DLR's Experience and Perspective". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 4587–4594.

[19] *On-Orbit Servicing ESA*. URL: https://blogs.esa.int/cleanspace/category/in-orbit-servicing/ (visited on 04/29/2020).

[20] *ESA is looking into futuristic in-orbit servicing: recycling satellites*. URL: https://blogs.esa.int/cleanspace/2019/09/09/esa-is-looking-into-futuristic-in-orbit-services-recycling-satellites/ (visited on 04/29/2020).

[21] J.L. Forshaw et al. "RemoveDEBRIS: An in-orbit active debris removal demonstration mission". In: *Acta Astronautica 127* 80 (2016), pp. 448–463.

[22] B.B. Reed et al. “The Restore-L Servicing Mission”. In: *AIAA SPACE Forum*. 2016.

[23] *OSAM-1: On-Orbit Servicing, Assembly and Manufacturing mission 1*. URL: https://nexis.gsfc.nasa.gov/osam-1.html (visited on 05/14/2020).

[24] C. Blackerby et al. “ELSA-D: An in-orbit end-of-life demonstration mission”. In: *Proceedings of the 69th International Astronautical Congress. Bremen, Germany*. 2016.

[25] *ESA commissions world’s first space debris removal*. URL: https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal (visited on 04/29/2020).

[26] Kazuya Yoshida. “Engineering Test Satellite VII Flight Experiments for Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit”. In: *The International Journal of Robotics Research* 22.5 (2003), pp. 321–335. DOI: 10.1177/0278364903022005003. eprint: https://doi.org/10.1177/0278364903022005003. URL: https://doi.org/10.1177/0278364903022005003.

[27] M. Oda. “Experiences and lessons learned from the ETS-VII robot satellite”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. 2000, pp. 914–919.

[28] Andrew Ogilvie et al. “Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System”. In: *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space, iSAIRAS* (Jan. 2008).

[29] *Statement of Work for the Control and Management of Robotics Active Debris Removal (COMRADE)*. SoW. ESA, 2015.

[30] Jean-Daniel Crepy-Marglais. “Development of a Robust and Combined Controller for On-Orbit Servicing Missions”. Master’s Thesis. Delft University of Technology, 2018.

[31] Pablo Colmenarejo et al. “Results of the COMRADE project: combined control for robotic spacecraft and manipulator in servicing missions: Active Debris Removal and re-fuelling”. In: June 2021.

[32] Evangelos Papadopoulos et al. “Robotic Manipulation and Capture in Space: A Survey”. In: *Frontiers in Robotics and AI* 8 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2021.686723. URL: https://www.frontiersin.org/article/10.3389/frobt.2021.686723.

[33] Alberto Bemporad and C. Rocchi. “Decentralized Hybrid Model Predictive Control of a Formation of Unmanned Aerial Vehicles”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 11900–11906. ISSN: 1474-6670. DOI: 10.3182/20110828-6-IT-1002.00942. URL: http://dx.doi.org/10.3182/20110828-6-IT-1002.00942.

[34] Carlo Alberto Pascucci, Samir Bennani, and Alberto Bemporad. “Model predictive control for powered descent guidance and control”. In: *2015 European Control Conference* (2015), pp. 1388–1393. DOI: 10.1109/ECC.2015.7330732.

[35] A. Bemporad. “Is MPC a Mature Technology for Guidance and Navigation?” In: (2011).

[36] M. Saponara et al. “Model Predictive Control application to spacecraft rendezvous in Mars Sample Return scenario”. In: *4th European conference for Aerospace Sciences (EUCASS)* (2011).

[37] D. Izzo and G.C.H.E. de Croon. “Nonlinear Model Predictive Control Applied to Vision-Based Spacecraft Landing”. In: *EuroGNC 2013* (2013), pp. 1–17.

[38] Caroline Buckner and Roberto Lampariello. “Tube-Based Model Predictive Control for the Approach Maneuver of a Spacecraft to a Free-Tumbling Target Satellite”. In: *Proceedings of the American Control Conference* June (2018), pp. 5690–5697. ISSN: 07431619. DOI: 10.23919/ACC.2018.8431558.

[39] Oyvind Hegrenaes, Jan Tommy Gravdahl, and Petter Tondel. “Spacecraft attitude control using explicit model predictive control”. In: *Automatica* 41.12 (2005), pp. 2107–2114. ISSN: 00051098. DOI: 10.1016/j.automatica.2005.06.015.

[40] Takeya Asakawa and Takashi Kida. “Application of MPC to Spacecraft Attitude Maneuver using RCS and/or RW”. In: *Journal of Space Engineering* 6.1 (2013), pp. 1–14. ISSN: 1881-736X. DOI: 10.1299/spacee.6.1.

[41] F. Valle, Fernando Tadeo, and Teresa Alvarez. “Predictive control of robotic manipulators”. In: *IEEE Conference on Control Applications - Proceedings* 1 (2002), pp. 203–208. DOI: 10.1109/cca.2002.1040186.

[42] Ph Poignet and M. Gautier. “Nonlinear model predictive control of a robot manipulator”. In: *International Workshop on Advanced Motion Control (AMC)* 2 (2000), pp. 401–406. DOI: 10.1109/amc.2000.862901.

[43] Josep Virgili-Llop et al. “Experimental evaluation of model predictive control and inverse dynamics control for spacecraft proximity and docking maneuvers”. In: *Naval Postgraduate School* March (2016). DOI: 10.13140/RG.2.1.4901.1444.

[44] Josep Virgili-Llop and Marcello Romano. “Simultaneous Capture and Detumble of a Resident Space Object by a Free-Flying Spacecraft-Manipulator System”. In: *Frontiers in Robotics and AI* 6 (2019).

[45] Josep Virgili-Llop et al. “A convex-programming-based guidance algorithm to capture a tumbling object on orbit using a spacecraft equipped with a robotic manipulator”. In: *International Journal of Robotics Research* 38.1 (2019), pp. 40–72. ISSN: 17413176. DOI: 10.1177/0278364918804660.

[46] Avgoustinos D. Saravanos. “Nonlinear Model Predictive Control for Space Robotic Systems (Diploma Thesis)”. Master’s Thesis. University of Patras, 2019.

[47] Tomasz Rybus, Karol Seweryn, and Jurek Z. Sasiadek. “Control System for Free-Floating Space Manipulator Based on Nonlinear Model Predictive Control (NMPC)”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 85.3-4 (2017), pp. 491–509. ISSN: 15730409. DOI: 10.1007/s10846-016-0396-2.

[48] Tomasz Rybus, Karol Seweryn, and Jurek Z. Sasiadek. “Application of predictive control for manipulator mounted on a satellite”. In: *Archives of Control Sciences* 28.1 (2018), pp. 105–118. ISSN: 23002611. DOI: 10.24425/119079.

[49] Steven Dubowsky and Evangelos Papadopoulos. “The Kinematics, Dynamics, and Control of Free-Flying and Free-Floating Space Robotic Systems”. In: *IEEE Transactions on Robotics and Automation* 9.5 (1993), pp. 531–543. ISSN: 1042296X. DOI: 10.1109/70.258046.

[50] Detlef Reintsema, Klaus Landzettel, and Gerd Hirzinger. “DLR’s Advanced Telerobotic Concepts and Experiments for On-Orbit Servicing”. In: *Advances in Telerobotics*. Ed. by Manuel Ferre et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 323–345. ISBN: 978-3-540-71364-7. DOI: 10.1007/978-3-540-71364-7_20. URL: https://doi.org/10.1007/978-3-540-71364-7_20.

[51] G. Hirzinger et al. “A Unified Ground Control and Programming Methodology for Space Robotics Applications - Demonstrations on ETS-VII”. In: Jan. 2000, pp. 422–427.

[52] Thomas Wolf. “Deutsche Orbitale Servicing Mission: The in-flight technology demonstration of German’s robotics approach to dispose malfunctioned satellites”. In: *ASTRA*. 2011.

[53] Heike Benninghof et al. “End-to-End Simulation and Verification of GNC and robotic systems considering both space segment and ground segment”. In: *CEAS Space Journal* (2018).

[54] Heike Benninghof et al. “End-to-End Simulation and Verification of Rendezvous and Docking Berthing Systems using Robotics”. In: *5th International Workshop on Verification and Testing of Space Systems* (2016).

[55] J. Artigas et al. “The OOS-SIM: An On-ground Simulation Facility For On-Orbit Servicing Robotic Operations”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015.

[56] B. Brunner et al. “Tele Sensor Programming - A task-directed programming approach for sensor-based space robots”. In: Jan. 1995.

[57] Luigi Strippoli, Nuno Gomes Paulino, and Julien Peyrard. “GNCDE as DD&VV Environment for ADR Missios GNC”. In: *6th ICATT* (2016).

[58] Josep Virgili-Llop et al. “Convex Optimization for Proximity Maneuvering of a Spacecraft with a Robotic Manipulator”. In: Feb. 2017.

[59] Richard E. Korf. *Space Robotics*. August. The Robotics Institute at Carnegie-Mellon University, 1982.

[60] Juergen Telaar et al. “GNC Architecture for the e.Deorbit Mission”. In: Jan. 2017.

[61] U. Eren et al. “Model Predictive Control in Aerospace Systems: Current State and Opportunities”. In: *Journal of Guidance, Control and Dynamics* 40 (2017), pp. 1541–1566.

[62] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control - Analysis and Design*. John Wiley Sons, 2005. ISBN: 978-0-470-01167-6.

[63] Alessandro Pasetti. *Software Frameworks and Embedded Control Systems*. Springer, 2002. ISBN: 3-540-43189-6.

[64] Glover Daniel. *Design Considerations for Space Flight Hardware*. Tech. rep. Lewis Research Center: NASA, Jan. 1990.

[65] Wigbert Fehse. “Disturbance by Solar Pressure of Rendezvous Trajectories in GEO”. In: May 2012.

[66] C. Gaz, F. Flacco, and A. De Luca. “Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1386–1392.

[67] Parag Tarwadi, Arockia A, and Juan Antonio Corrales Ramon. “Manipulation and path planning for KUKA (LWR/LBR 4+) robot in a simulated and real environment”. In: *Journal of Automation, Mobile Robotics Intelligent Systems* 11 (Nov. 2017), pp. 15–21. DOI: 10.14313/JAMRIS_3-2017/24.

[68] R. Bischoff et al. “The KUKA-DLR Lightweight Robot arm - a new reference platform for robotics research and manufacturing”. In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. 2010, pp. 1–8.

[69] Kazuya Yoshida. “The Spacedyn, a MATLAB Toolbox for Space and Mobile Robots”. In: *The Space Robotics Lab, Department of Aeronautics and Space Engineering, Tohoku University, Japan* (1999).

[70] Avishai Weiss et al. “Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies”. In: *IEEE Transactions on Control Systems Technology* 23 (2015).

[71] Mihir Bhagat. “Convex Guidance for Envisat Rendezvous”. PhD thesis. Feb. 2016.

[72] Richard Scott Erwin Avishai Weiss Morgan Baldwin. “Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies”. In: *IEEE Transactions on control systems technology, vol.23, no.4* (2015).

[73] J. Löfberg. “YALMIP : A Toolbox for Modeling and Optimization in MATLAB”. In: *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[74] Brent Barbee et al. “Guidance and Navigation for Rendezvous and Proximity Operations with a Non-cooperative Spacecraft at Geosynchronous Orbit”. In: *The Journal of the Astronautical Sciences* 58 (2010), pp. 389–408. URL: https://doi.org/10.1007/BF03321176.

[75] M. Morari A. Bemporad. *Robust Model Predictive Control: A Survey*.

[76] Sergio Grammatico. “Model Predictive Control (SC42125)”. In: *Course Notes* TU Delft (2020).

[77] M.M. Diehl J.B. Rawlings D.Q. Mayne. *Model Predictive Control: Theory, Computation and Design*. Second Edition. Nob Hill Publishing, 2019. ISBN: 9780975937730.

[78] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009. ISBN: 9780521833783.

[79] A. Bemporad and M. Morari. “Control of systems integrating logic, dynamics, and constraints”. In: *Automatica 35* (1999), pp. 407–427.

[80] Alberto Bemporad. “Model Predictive Control of Hybrid Systems”. In: *2nd HYCON PhD School on Hybrid Systems* (2007).

[81] R.R. Negenborn and J.M. Maestre. “Distributed Model Predictive Control: An overview of features and research opportunities”. In: Apr. 2014, pp. 530–535. ISBN: 978-1-4799-3106-4. DOI: 10.1109/ICNSC.2014.6819682.

[82] Brett T. Stewart et al. “Cooperative distributed model predictive control”. In: *Systems and Control Letters* 59.8 (2010), pp. 460–469. ISSN: 01676911. DOI: 10.1016/j.sysconle.2010.06.005. URL: http://dx.doi.org/10.1016/j.sysconle.2010.06.005.

[83] M. Morari A. Bemporad F. Borrelli. “Model Predictive Control Based on Linear Programming - The Explicit Solution”. In: *IEEE Transactions on Automatic Control* 47.12 (2002), pp. 1974–1985.

[84] M. Kvasnica. “Implicit vs explicit MPC — Similarities, differences, and a path owards a unified method”. In: *2016 European Control Conference (ECC)*. 2016, pp. 603–603.

A

Model Predictive Control - theoretical background

This chapter presents the overview of the control theory to be implemented in the project. The Model Predictive Control (MPC) has been chosen as a potentially good candidate for the optimized solution of spacecraft robotic manipulator control problem given its extensive use in other industries since 90's, as well as emerging applications in space industry. In the next section A.1 the motivation standing behind this choice is further outlined, with the standard formulation of the MPC problem in section A.1.1 and introduction to optimization in MPC in section A.1.2. The section A.2 presents different MPC structures: offset-free, hybrid and distributed, subsequently the comparison of implicit and explicit control laws is given.

A.1. Introduction and Formulation

The main purpose of this section is to provide the theoretical background of model predictive control (MPC) that was implemented in the project, which is an advanced control technique enabling finding an optimal control solution to the problem while satisfying a given set of constraints. This control technique was chosen as a candidate to investigate its possible application onto design of a combined controller in the frame of RICADOS project, to see how can it be implemented and how effectively it deals with the system and control task.

The first ideas of this technique can be traced back to the 1960s with an interest in the field starting in 1980s mainly in the process industries [75]. By the end of 90s, the MPC had already industrial applications mainly in the field of refining, petrochemicals as well as first implementations in automotive industry e.g. active steering of a car. The MPC technique proved to be effective in the above-mentioned contexts due to its numerous advantages - it allows optimizing the process within given set of constraints simultaneously predicting the evolution of its dynamics and penalizing the undesired behaviour. It is clear, that optimizing a production process while minimizing other parameters such as e.g. cost, employees working time and buy-to-fly ratio can tremendously increase the profit which is of high importance in majority of industries nowadays. The dynamics prediction with the active control have significant role in automatic control and hence, MPC is frequently implemented in autonomous vehicle online path planning. Moreover, there is an increasingly growing interest in aeronautics and aerospace sector with many already existing applications such as e.g. navigation of UAVs, formation flying control, spacecraft attitude control or wheel momentum damping by thrust orientation mechanism. Especially in the last decade, the interest into space applications is growing such as e.g. planetary rovers or orbital guidance and control [35]. Review of existing MPC solutions in space sector is not covered in this section, it was presented in the Chapter 1.2.2 along with the MPC applications for the robotic manipulators.

MPC is a feedback control strategy, which exploits a plant model to predict its likely future evolution and to compute the (an almost) best control input. It is an optimal control technique which enables minimization of the cost function while meeting the required objective. The main distinct feature of MPC is constraint handling, which allows finding a control solution while respecting the hard and soft constraints enforced by the system such as e.g. the limited performance of the actuators, limited process time, obstacle avoidance and others. It can be considered already reasonably mature technology given more than 20 years of research, the vast scientific literature providing overview on implementation of different approaches both off-line design and real-time code, further presented in the Section A.2. However, the efficiency of MPC is a function of accuracy of the dynamical model used, sometimes there might be no model available, or it is highly approximated, which might produce inconsistencies between the real dynamics of a plant and modelled ones. It must be also pointed, that finding solution to an optimization problem demands high memory storage of the processor and computation time which is increasing along with the complexity of the model and its optimization variables. Nowadays, it is of less concern as the optimization solvers

are getting faster, nevertheless the on-line computation will always remain limited by the processing power of the computer, which in many cases has very limited performance as in satellite on-board computer.

The underlying MPC concept is to choose the ”best” control action based on the prediction of the process future evolution by using its dynamical model, which is depicted in Figure A.1. The model-based optimizer, which is in essence an MPC controller, contains two main items inside - a prediction model and an optimization algorithm. The mathematical formulation of an MPC problem is given in next Section A.1.1.

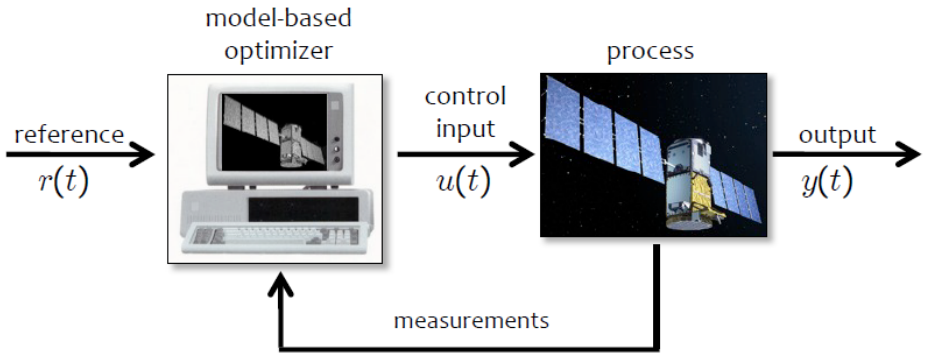


Figure A.1: Controller structure; derived from [35]

A.1.1. MPC Formulation

In this section, firstly the receding horizon policy is presented which is an underlying principle of the considered control technique. Next, the general formulation of MPC problem is given, with definition of variables, their dimensions and algorithm sequence, as well as the cost function.

Receding Horizon Policy

The main idea standing behind the receding horizon policy, is presented in the figure A.2, where N is prediction/control horizon length and k is a current time step in a discrete model; k is a nonnegative integer denoting the sample number, which is related to time by $t = k\Delta t$ in which Δt is the sampling time. The optimization algorithm computes the possible control input sequences consisting of N control commands, among all possible control sequences the optimal one is selected. The evolution of the output is predicted for the next N steps, parametrically on the initial condition and the optimal control sequence. At time step k only the first control input of the computed optimal command sequence is applied to the system, $u^*(k)$. The remaining optimal inputs are discarded, and a new optimal control problem is solved at time step $k + 1$. The same procedure is repeated, the initial state is considered to be the state from the previous step k, the control horizon N length remains unchanged, which subsequently results in $k + N + 1$ predicted control inputs with respect to the previous step k. New measurements are collected from the plant at each time step k, the receding horizon mechanism provides the controller with the desired feedback characteristics.

In some variants prediction horizon length (or output horizon) and control horizon length (or input horizon) are not equal, they are referred to N_p and N_m respectively. If the $N_p = \infty$ it is referred to as the infinite horizon problem, and similarly, if N_p is finite, it is a finite horizon problem [75].

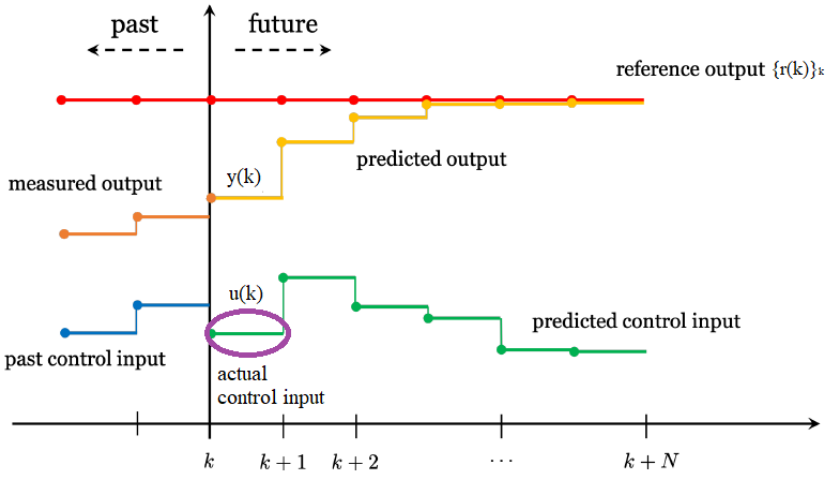


Figure A.2: Receding horizon policy; derived from [76]

The basic MPC law is described by the following algorithm:

1. Get the new state x(k)
2. Solve the optimization problem
3. Apply only $u(k) = u(k+0|k)$
4. k= k+1. Go to (a).

The formulation of a discrete-time dynamical systems of an arbitrary MPC problem is shown below in the essence. Discrete time models are usually favorable if the system of interest is sampled at discrete times. If the sampling rate is chosen appropriately, the behaviour between the samples can be safely ignored and the model describes exclusively the behaviour at the samples times. In further parts of the report, the formulations for discrete time models are given, as they are the most convenient way of depicting the underlying principles commonly found in the literature and textbooks.

State Space Form
In the research literature MPC is formulated mainly in the state space form [75]. The general state space form of a discrete-time dynamical system is presented, in which $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ denote the state, control input, and output respectively, x_0 is the initial state and $k \in \mathbb{I}_{\geq 0}$ is a time-step of a discrete system. $\mathbb{R}^n$ denotes the set of real-valued $n-$vectors. The equations on the right, in A.1, are a compact notation to be further implemented in the document and are commonly used by MPC designers.

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) & \qquad x^+ &= f(x,u) \\ y(k) &= h(x(k), u(k)) & \qquad y &= h(x,u) \\ x(0) &= x_0 & \qquad x(0) &= x_0 \end{aligned} \tag{A.1}$$

State Solution
$x(k)$ is a state at time k, starting from initial state x_0, subject to control sequence $\mathbf{u}_k$, where ϕ is a state solution function. It is noted that bold notation means a ”sequence” which will be frequently used in further parts of the report.

$$x(k) = \phi(k; x_0; \mathbf{u}_k) \tag{A.2}$$

Control Input Sequence
A control input sequence is a vector of the computed optimal control commands, the subscript k is the sequence length. Hence, to find the state solution $x(k)$ from equation A.2, one needs to apply all the sequence into the formula.

$$\mathbf{u}_k = (u(0), u(1), ..., u(k-1)) = \begin{bmatrix} u(0) \\ \vdots \\ u(k-1) \end{bmatrix} \in \mathbb{R}^{m \cdot k} \tag{A.3}$$

Discrete-Time Linear Dynamical Systems
For a linear dynamical system, the state space form and state solution are formulated in the following way:

$$\begin{aligned} x^+ &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{A.4}$$

$$\begin{aligned} x(k) &= A^k x_0 + \Sigma_{j=0}^{k-1} A^{k-j-1} B u(j) \\ &= A^k x_0 + \begin{bmatrix} A^{k-1}B & \dots & AB & B \end{bmatrix} \begin{bmatrix} u(0) \\ \vdots \\ u(k-2) \\ u(k-1) \end{bmatrix} \\ &= A^k x_0 + \mathcal{C}_k \mathbf{u}_k = \phi(k; x_0; \mathbf{u}_k) \end{aligned} \tag{A.5}$$

where $\mathcal{C}_k$ is a controllability matrix; the formulation A.5 for the linear system is called a prediction model of the system.

Constraints
In order to consider the physical limitations of the control input and desired states or outputs for reasons of safety, operability, time criticality and others, the contraints are imposed on the system. The constraints are the main feature that distinguishes MPC from the standard linear quadratic (LQ) control. The below given dimensions will be kept consistent in the report. For further background in constraints definition, the textbook [76] is referred to. The control u is subject to the constraint $u \in \mathbb{U} \subseteq \mathbb{R}^m$, where $\mathbb{U}$ is compact and contains the origin in its interior, the state x is subject to the constraint $x \in \mathbb{X} \subseteq \mathbb{R}^n$, where $\mathbb{X}$ is closed and contains the origin in its interior, and output y is subject to the constraint $y \in \mathbb{Y} \subseteq \mathbb{R}^p$. The constraints usually are included by linear inequalities as follows:

$$Fx(k) \leq e, \qquad \forall k \geq 0 \tag{A.6}$$

$$Eu(k) \leq f, \qquad \forall k \geq 0 \tag{A.7}$$

where the matrix F and e represents the upper/lower boundaries on the state, and the matrix E and f the boundaries on the input. If one wants to consider fairly general form for a linear system, the following state-input constraint formulation is chosen:

$$Fx(k) + Eu(k) \leq h, \qquad \forall k \geq 0 \tag{A.8}$$

The described previously input and state constraints are considered a set of hard constraints, which cannot be violated during a whole evolution of the system. In case the constraints in some cases do not necessarily need to be satisfied by some feasible solutions, they can be softened with the introduction of a slack variable $\epsilon(k)$. The constraints are relaxed in the following way, here an example of a state constraint:

$$Fx(k) \leq e + \epsilon(k) \qquad \text{, where } \epsilon(k) \geq 0 \tag{A.9}$$

As is discussed further, one then formulates a stage-cost penalty that weights how much one 'cares' about the state x, the input y and the violation of the hard state constraint, which is given by ϵ. The benefit of this reformulation is that the state constraint cannot cause an infeasibility in the control problem because it can be relaxed by choosing ϵ; larger values of ϵ may be undesirable as measured by the stage-cost function but they are not infeasible.

Cost function
In every optimization step, an objective function is minimized within given constraints and dynamics of the plant in order to find an optimal control sequence. The cost function $V(\cdot)$, in its essence, consists of two terms - a stage-cost function $l(x(k), u(k))$ and a terminal cost term $V_f(x(N))$ which penalizes all predicted states and required inputs to achieve these states, and the final state respectively.

$$V_N(x_0, \mathbf{u}_N) = \Sigma_{k=0}^{N-1} \{l(x(k), u(k))\} + V_f(x(N)) \tag{A.10}$$

where $x(k)$ is as defined in prediction model (eqn. A.5). One of the most common formulation of the cost function is a quadratic cost function similar to the LQR control problem.

$$V_N(x_0, \mathbf{u}_N) = \Sigma_{k=0}^{N-1} \{\frac{1}{2} x(k)^T Q x(k) + \frac{1}{2} u(k)^T R u(k)\} + \frac{1}{2} x(N)^T P_f x(N) \tag{A.11}$$

in which the weighting matrices Q, R and P_f are the tuning parameters. The large values of matrix Q reflect the intent to drive the state to the origin quickly at the expense of large control action, large values of R matrix relative to Q penalize the control, which effectively reduces the control action and slows down the rate at which the state approaches the origin. The choice of the appropriate tuning parameters in the controller is not always obvious, this challenge is as well intrinsic to the LQR control tuning. In order to guarantee that the solution to the optimal control problem exists and is unique certain assumptions are made: the Q, P_f and R are real and symmetric; Q and P_f are positive semidefinite and R is positive definite. Very often the matrices Q, P_f and R are chosen to be diagonal. One of the frequent strategy is to define matrix P_f from the Discrete Algebraic Riccati Equation (DARE) for the infinite horizon control problem.

It must be kept in mind that not always the finite horizon optimality ensures stability of the system, unless the horizon is sufficiently large. However, the infinite horizon LQR is always stabilizing.

Optimal Control Problem Formulation

Given all the above definitions, finally the general formulation of an optimal control problem is presented, for a horizon length N and an optimization variable $\mathbf{u}_N$

$$\mathbb{P}_N(x_0, t): \begin{cases} \min\limits_{\mathbf{u}_N} & V_N(x_0, \mathbf{u}_N) \\ \text{s.t.} & \text{system dynamics} \\ & \text{system constraints} \\ & x(N) \in \mathbb{X}_f \end{cases} \text{, where } x(t) = x_0 \text{is the current state} \tag{A.12}$$

where $\mathbb{X}_f$ is a terminal constraint, a bound on the final state of the system. It is noted, that a terminal constraint is often omitted in the control problem definition and the actual computations, as it penalizes optimality and increases computational effort [76]. The final state can be included also as a weighting factor on the terminal cost in the following way: $\beta V_f(x(N))$, $\beta \geq 1$ large enough. The system dynamics come from the state solution prediction model, and system constraints are defined accordingly to the process specifications, physical limits, desired trajectory and others.

A.1.2. Optimization

The optimization problem is inherent to MPC, hence some focus must be given to the proper formulation of an optimization problem and approaches towards its solution. The optimization problem objective is to minimize the value of an objective function $f(u)$ with respect to the optimization variable u of a problem, such that the inequality constraints $g(u)$ and equality constraints $h(u)$ are met. The most generic form of an optimization problem is defined as follows:

$$\mathbb{P}: \begin{cases} \min\limits_{u \in \mathbb{R}^n} & f(u) \\ \text{s.t.} & g(u) \leq 0 \\ & h(u) = 0 \end{cases} \tag{A.13}$$

The $g(u)$ and $h(u)$ are in fact column matrices for which the (in)equality condition is applied element-wise for every row function $g_i(u)$ and $h_j(u)$. The feasible set of solutions for the optimization problem can be formulated as: $F = \{u \in \mathbb{R}^n | g(u) \leq 0, h(u) = 0\}$.

Convex optimization

In order to ensure that the found optimal solution is global, it is favorable to formulate a problem as a *convex optimization problem*, for which any locally optimal point is globally optimal (Proposition C.8 (*Global optimality for convex problems*), [77], p.741). The problem $\mathbb{P}$ is convex if:

1. f is convex, e.g. $f(u) = u^T P u - p^T u, P \succ 0$
2. g_i is convex, e.g. $g_i(u) = a_i^T u - b_i$
3. h_j is affine, i.e. $h_j(u) = c_j^T - d_j$

A vector u^* is called optimal, or a solution of the problem A.13 if it has the smallest objective value among all vectors that satisfy the constraints. Very often in MPC formulations zero subscript is used to indicate optimality, u^0. The textbook ([77], p.729-767), is further referred to for theoretical formulations of a variety of optimization problems. Moreover, the textbook [78] is extensively treating the *Convex Optimization* problems, which are a frequent formulation in the MPC.

A mathematical optimization problem is called a programming problem, and depending on the formulation of its structure, which in MPC is the cost function, in the literature it is referred to as a Quadratic, Linear, Nonlinear or Mixed-Integer Program which is shown in the Table A.1.

Prediction model, constraints, cost	MPC optimization problem
Linear model and constraints, quadratic costs	(convex) Quadratic Program (QP)
Linear model and constraints, linear costs (e.g. infinity norms)	Linear Program (LP)
Nonlinear models, constraints, cost	Nonlinear Program (NLP)
Hybrid dynamical models	Mixed-Integer Program (MIP)

Table A.1: Types of MPC optimization problems.

Optimality conditions
For any optimization problem with differentiable objective $f(u)$ and constraint functions $g(u)$, $h(u)$ the following theorem should hold to meet the optimality conditions of the problem.

[Theorem] u is the local minimizer $\implies \exists \lambda \in \mathbb{R}_{\geq 0}^{n_{ineq}}, \mu \in \mathbb{R}^{n_{eq}}$ such that the Karush-Kuhn-Tucker (KKT) conditions hold for the system:

$$\nabla f(u) + \sum_{i=1}^{n_{ineq}} \lambda_i \nabla g_i(u) + \sum_{j=1}^{n_{eq}} \mu_j \nabla h_j(u) = 0 \tag{A.14}$$

$$\lambda_i \nabla g_i(u) = 0, \forall i \in \{1, ..., n_{ineq}\} \tag{A.15}$$

in which the equation A.14 is a stationarity condition and the equation A.15 is a complementarity condition. Moreover, a constraint qualification is required for the Karush-Kuhn-Tucker condition to be a necessary condition of optimality for the optimization problem [76].

A.1.3. Feasibility

When setting up the optimization problem with the equality and inequality constraints, it is important to keep in mind the feasibility of the problem at at each step k. Typically ones assumes feasibility at time $t = 0$ and chooses cost function and the stability constraints such that feasibility is preserved at the following time steps [75]. It could be done e.g. by treating the constraints involving state components as soft constraints and the input constraints as hard; relaxing the state constraints removes the feasibility problem [75, p. 5]. Due to the presence of noise, disturbances, and numerical errors keeping state constraints as hard is impractical. As the inputs are generated by the optimization procedure, the input constraints can always be regarded as hard. The further assumptions for feasibility of robust control, can be found in the handbook [77].

A.1.4. Stability

Another aspect to be always considered is that the controlled system must be stable. In the literature many techniques used to enforce stability can be found, which are briefly summarized below. Frequently, in order to proof stability of the system the cost function (as defined in A.10) is considered as Lyapunov function and the stability is proved by showing the Control Lyapunov Function (CLF) decrease, defined in the figure A.3. The stability check can be done by continuously computing the value of $V_f(f(x,u)) - V_f(x) \leq -\ell(x,u)$ in each simulation step k of the system.

- $\mathcal{X}$, $\mathcal{A}$ control invariant, $\mathcal{A} \subset \mathcal{X}$

Control Lyapunov function (CLF): $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ s.t.

$$\alpha_1(|x|_{\mathcal{A}}) \leq V(x) \leq \alpha_2(|x|_{\mathcal{A}})$$
$$\inf_{v \in \mathbb{U}} \{V(f(x,v)) \mid f(x,v) \in \mathcal{X}\} \leq V(x) - \alpha_3(|x|_{\mathcal{A}})$$

for all $x \in \mathcal{X}$, for some $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$, $\alpha_3 \in \mathcal{PD}$

Theorem: $\exists$ CLF $\implies$ Global Stabilizability

Lyapunov control law: $\kappa(x) := \operatorname{argmin}_{v \in \mathbb{U}} \{V(f(x,v)) \mid f(x,v) \in \mathcal{X}\}$

Figure A.3: Controller structure; derived from [76]

The techniques for enforcing stability of the MPC system are presented in the survey by A.Bemporad and M.Morari [75], they include:

- *Terminal Constraint* - imposing the stability constraint such that the final state is steered to the origin $x(t + N_p|t) = 0$; such constraint could yield infeasibility issues as it might require a large control effort, the domain of attraction is also limited to the set of initial states that can be steered to 0 in N_p steps. The terminal constraint can also drastically decrease the performance.
- *Infinite Output Prediction Horizon* - in case the output horizon $N_p = +\infty$, no stability constraint is required for asymptotically stable systems.
- *Terminal Weighting Matrix* - putting a weighting factor P_f on the final state in eqn. A.10 such that it is solution of a Riccati equation can guarantee stability without the addition of stability constraints.
- *Invariant Terminal Set* - relaxing the terminal constraint into the terminal set such that $x(t + N_p|t) = \in \Omega$, the Ω set is invariant and such that the constraints are fulfilled inside this set.
- *Contraction Constraint* - decreasing of the state $x(t)$ in some norm $\| x(t + 1|t) \| \leq \alpha \| x(t) \|, \alpha < 1$, is required explicitly rather than relying on the optimal cost $V(t)$ as a Lyapunov function.

For further stability properties of discrete time systems the Appendix B from the book [77] will be referred to, which provides theoretical background of stability theory. The further discussion concerning which technique for stability proof is appropriate for the project will be done during the controller design in an iterative manner.

A.2. Overview of MPC structures

In this section the most important characteristics of different MPC structures are given. Firstly the general form of the Output, Hybrid and Distributed MPC are given in order to present the limitations of each of them with the main advantages and disadvantages, which is important to understand in order to perform the trade-off of them to be used in the project. Moreover, there is another distinction in the MPC design in terms of the method for solving an optimization problem. The implicit model predictive control performs optimization online, whereas the explicit solves the optimization problem offline. These two approaches are also compared in this section.

A.2.1. Hybrid MPC

This strategy considers the system to be controlled that constitutes of the differential equations typically derived from the physical laws governing the dynamics of the system, and also logic rules with operating constraints which is a distinct characteristic of a hybrid system. The main motivation for the hybrid system is that such a definition allows to describe a system's behaviour which is characterised by continuous dynamics, continuous and discrete inputs and the interconnection between logics and dynamics. The parts described by logic can be for example the on/off switches, valves, car gears, speed selector or evolutions dependent on if-the-else rules and others. Hybrid model shall be descriptive enough, but also simple enough for solving both analysis and synthesis problems [76]. This type of system is denoted as *mixed logical dynamical (MLD)* system, it involves logic rules which are described by the linear integer inequalities and also the continuous and logical variables which are part of the mixed-integer linear relations. Due to the presence of integer variables, the resulting optimization problem is solved through *mixed integer linear or quadratic programming (MILP/MIQP)* [79].

The major examples of the MLD systems are piece-wise linear systems which can approximate nonlinear or discontinuous dynamics arbitrarily well; switched affine systems, in which the affine dynamics depend on the current mode and during the dynamics evolution they are switched e.g. according to the state-update equation consisting of the difference equation and the if-then-else conditions; event generators in which event variables are generated by linear threshold conditions over continuous states, inputs and time; piece-wise linear output functions, finite state machines, mode selectors and others. The main characteristics of each of them are summarized in [80] and might be used for further reference in case such a design solution shall be chosen later. The most generic MLD hybrid model can be defined in compact form as:

$$\begin{cases} x^+ &= Ax + B_1 u + B_2 \delta + B_3 z \\ y &= Cx + D_1 u + D_2 \delta + D_3 z \end{cases} \tag{A.16}$$

subject to:

$$E_2 \delta + E_3 z \leq E_4 x + E_1 u + E_5,$$

where $x = \left[\begin{smallmatrix} x_c \\ x_d \end{smallmatrix}\right] \in \mathbb{R}^{n_c} \times \mathbb{B}^{n_d}, u = \left[\begin{smallmatrix} u_c \\ u_d \end{smallmatrix}\right] \in \mathbb{R}^{m_c} \times \mathbb{B}^{m_d}, y = \left[\begin{smallmatrix} y_c \\ y_d \end{smallmatrix}\right] \in \mathbb{R}^{p_c} \times \mathbb{B}^{p_d}$ and the discrete binary variable $\delta \in \mathbb{B}^{r_d}$, where $\mathbb{B} = \langle 0, 1 \rangle$ and the auxiliary real variable $z := x\delta \in \mathbb{R}^{r_c}$. Further, x_c is the continuous component of the system state and x_d is the discrete component, the same holds for the output y and the input control command$u$, collecting both continuous commands u_c and binary (on/off) commands u_d. Such a hybrid model can be the subject of structural constraints applied onto continuous variables (e.g. actuators saturation, safety conditions etc): $\mathcal{C} := \{(x_c, u_c) \in \mathbb{R}^{n_c} \times \mathbb{R}^{m_c} | Gx_c + Hu_c + \phi \leq 0\}$.

The MLD model can be converted into an equivalent piecewise affine form (PWA), which usually requires the enumeration of all possible combinations of binary states, inputs and δ variables. However there are efficient algorithms available for converting MLD models into PWA models avoiding such an enumeration.

$$\begin{cases} x^+ & = A_i x + B_i u + f_i \\ y & = C_i x + D_i u + g_i \\ i \text{ s.t.} & H_i x + J_i u \leq K_i \end{cases} \tag{A.17}$$

Other existing classes of hybrid models include the linear complementarity systems, extended linear complementarity systems and min-max-plus-scaling systems, which are furthered presented in [80]. According to the theorem ([80], p.51) all the above classes of discrete-time hybrid models are equivalent (possibly under some additional assumptions, such as boundedness of input and state variables).

Having defined the dynamical system, the MPC problem can be formulated. The MLD or PWA model of the plant is used to predict the future behaviour of the hybrid system. The hybrid MPC requires online solution of MILP/MIQP.

$$\mathbb{P}_N(x_0) : \begin{cases} \min\limits_{\mathbf{u}_N, \delta_N, z_N} & V_N(x_0, \mathbf{u}_N) = \sum\limits_{k=0}^{N-1} \{l(x(k), u(k))\} \\ \text{s.t.} & x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k), \forall k \\ & E_2 \delta(k) + E_3 z(k) \leq E_4 x(k) + E_1 u(k) + E_5, \forall k \\ & x_0 = x(0) \end{cases}, \tag{A.18}$$

where the cost function is formulated as a quadratic stage cost and the state/output prediction is as for LTI systems, with accounting for the discrete parts:

$$x(k) = \phi(k; x_0, \mathbf{u_k}, \delta_k, z_k) = A^k x_0 + \mathcal{C}_k \mathbf{u_k} + \mathcal{C}_k^\delta \delta_k + \mathcal{C}_k^z z_k \tag{A.19}$$

where:

$$\mathcal{C}_k^\delta := \left[B_2 \; AB_2 \; \dots \; A^{k-1}B_2 \right], \mathcal{C}_k^z := \left[B_3 \; AB_3 \; \dots \; A^{k-1}B_3 \right]$$

Hybrid MPC control problem can be written in a compact form, e.g. as a MIQP formulation of MPC:

$$\mathbb{P}_N(x_0) : \begin{cases} \min\limits_{\xi_N} & \frac{1}{2} \xi_N^T H \xi_N + x_0^T F \xi_N \\ \text{s.t.} & G\xi_N \leq W + Sx_0 \end{cases}, \tag{A.20}$$

where the optimization vector ξ_N is composed of the elements referring to the mixed-integer, binary and real part:

$$\xi_N = \left[u(0), \dots, u(N-1), \delta(0), \dots, \delta(N-1), z(0), \dots, z(N-1) \right]^T$$

The MPC problem can be also formulated as MILP, which is proposed in ([80], p.69). Mixed-integer programming despite being extensively covered in literature, still has some major drawbacks. First of all, the hybrid MPC problem can loose its original boolean structure, it might be advantageous to combine symbolic with numerical solver. Moreover, the computation time may be too long which makes it hardly applicable to the fast sampling problems. The high level of software complexity makes MIP solver code difficult to certify, hence it is rather not applicable for safety critical operations. The paper [79] gives a very good overview of the implementation of the Hybrid MPC strategy on the example of the complex gas supply system. Moreover it presents the four major types of MIQP which were applied successfully to medium and large application problems. For applications in which MIP does not provide with satisfactory results in terms of computing time, performance, realistic level of complexity etc the implementation of off-line solvers might be favoured, which is further elaborated in the section A.2.4.

The certain design strategies for the hybrid controller are further presented in the [80], such as augmentation of the hybrid prediction model with integrators of output errors in order to account for measured disturbances, time delays, formulation of an optimal control problem with prioritized constraints.

A.2.2. Distributed MPC

Another class of the MPC problem is distributed approach, in essence it is applied to multi-agent systems and the main idea is to distribute the computation into all the agents - to break the big optimization problem into few smaller ones in order to relief the computational burden. The overall plant control is accomplished by the combined behaviour of the interacting local controllers. In such a way all the agents cooperate and share the cost of computation, which is the main advantage of this approach. However such a formulation of the control problem is suboptimal and is characterised by a possible performance loss.

The optimal control problem is distributed in separable problems into each agent (subsystem), hence e.g. for the system consisting of two agents there will be two optimal control problems. Each subsystems state vector consists of $x_{i,j}$, which is state of subsystem i affected by input j. OCP of the agent 1 would be parametric in u_2 in such a way that the cost function is optimized for the optimal sequence u_1 given u_2 and the same holds for the agent 2 - the OCP would be parametric in u_1 and the cost function is optimized for the optimal sequence u_2 given u_1. The further mathematical descriptions, relevant assumptions and convergence properties can be found in [76, 77] and might be addressed further in research if found relevant.

In [77], three control approaches are presented: decentralized, noncooperative and cooperative. Decentralized approach assumes no communication between the agents, whereas noncooperative and cooperative control requires the input sequences and the current states or state estimates for all the other local subsystems. All these methods require the local controllers to optimize over only their local inputs, and the computational requirements are identical. The distributed MPC is found in the literature to be applied into e.g. UAV formation flying, control of swarms of spacecraft or cooperative satellite formation flying and also in some applications for cooperative manipulation of the vehicle manipulator systems. The further overview of features and research opportunities of the distributed MPC are presented in [81, 82]. Nevertheless, it is considered not to be useful for application into a single robot manipulator system.

A.2.3. Offset-free MPC

As in many feedback controllers, the objective is to move the measured outputs of a system to a specified and constant setpoint, which is known as setpoint tracking [77]. In case of any disturbances in the system, tracking control must compensate for them so their effect on the controlled variable is mitigated, which is simply called disturbance rejection. The MPC strategy presented in this section allows for compensation of nonzero disturbances such that the selected controlled variables asymptotically approach their setpoints without offset, such a property is known as zero offset.

A simple method to compensate for an unmeasured disturbance is to firstly account for the disturbance in the model of the plant, use the measurements and based on them determine an estimate of the disturbance. Finally find the inputs that minimize the effect of the disturbance on the controlled variables. The formulation of the problem depends on which states that can be measured:

State Feedback

The formulation of the state feedback problem in which the state x is directly measurable:

$$\begin{cases} x^+ &= Ax + Bu \\ y &= Cx + d + v \end{cases} \tag{A.21}$$

where d is constant, unknown disturbance and v is measurement noise, such that the expected value for the random variable $v(t)$ is zero, $\mathbb{E}[v(t)] = 0$. In order to solve for such a system, one of the approaches is to design an augmented system with the same dynamic structure as before, with an auxiliary immeasurable state $d^+ = d$.

$$\begin{cases} \begin{bmatrix} x^+ \\ d^+ \end{bmatrix} &= \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u \\ y &= \begin{bmatrix} C & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + v \end{cases} \tag{A.22}$$

In order to solve such a system, one must design a disturbance observer, since the state x is directly measurable, one can design an observer for the ”state” d only. The disturbance estimate $\hat{d}^+$ for the next step is defined with the Luenberger observer as:

$$\hat{d}^+ = \hat{d} + L(y - \hat{y}) = \hat{d} + L(y - Cx - \hat{d}) \tag{A.23}$$

The idea is to use the best future estimate of the disturbance in the current simulation step in order to find the value of y_{ref} to be injected into the optimization problem formulation, where y_{ref} is a desired output setpoint reference required for formulation of the reference tracking problem, such that $Cx_{ref} = y_{ref}$, the algorithm (A.24) implemented to find the values of (x_{ref}, y_{ref}) is called an optimal target selector (OTS).

$$(x_{ref}, u_{ref})(\hat{d}, y_{ref}) \in \begin{cases} \min\limits_{x_{ref}, u_{ref}} & J(x_r, u_r) \\ \text{s.t.} & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_r \\ u_r \end{bmatrix} = \begin{bmatrix} 0 \\ y_{ref} - \hat{d} \end{bmatrix} \\ & (x_r, u_r) \in \mathbb{Z} \\ & Cx_r + \hat{d} \in \mathbb{Y} \end{cases} \tag{A.24}$$

For the simple tracking problem without disturbance in the output neither the input, the OTS is solved offline because y_{ref} is a given constant vector. However for the offset-free state feedback controller which must reject the existing disturbance, the values of reference state and output vector are computed in every simulation step using the updates of the disturbance estimates - the OTS is solved online. The cost function for such a problem is defined as the steady-state target problem modified to account for the nonzero disturbance $\hat{d}$ in the following way:

$$V_N(x_0, \hat{d}, y_{ref}, \mathbf{u_N}) = \sum_{k=0}^{N-1} \{l(x(k) - x_{ref}(\hat{d}), u(k) - u_{ref}(\hat{d}))\} + V_f(x(N) - x_{ref}(\hat{d})) \tag{A.25}$$

The final definition of an optimal control problem has the same structure as in MPC regulation, with the main difference that now the OCP is parametric in one additional variable, $\hat{d}$.

Output Feedback

The system in which the state x cannot be measured directly, the output feedback strategy can be implemented which also enables disturbance rejection.

$$\begin{cases} x^+ &= Ax + Bu + B_d d + w \\ y &= Cx + C_d d + v \end{cases} \tag{A.26}$$

The matrices (A, B) are controllable, (A, C) are observable which is necessary to reconstruct the state from input and output. As in the definition of the state feedback, d is a constant unknown disturbance, w is a state disturbance with $\mathbb{E}[w(t)] = 0$, v is a measurement noise $\mathbb{E}[v(t)] = 0$.

$$\begin{cases} \begin{bmatrix} x^+ \\ d^+ \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} w \\ w_d \end{bmatrix} \\ y &= \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x \\ d \end{bmatrix} + v \end{cases} \tag{A.27}$$

In order to solve such a system, one must design an observer which will allow to reconstruct the full unknown state $[x\ d]'$. The Lueneberger observer which accounts for both the estimate of state and disturbance is defined in the following way:

$$\begin{bmatrix} \hat{x}^+ \\ \hat{d}^+ \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} (y - \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix}) \tag{A.28}$$

Next, the OTS for the output feedback MPC is designed, the problem is solved online and for every simulation step the algorithm takes the best current estimate of $\hat{d}$ and $\hat{x}$ to find (x_{ref}, u_{ref}) and feed the MPC regulator. The OTS is parametric in y_{ref}:

$$(x_{ref}, u_{ref})(\hat{d}, y_{ref}) \in \begin{cases} \min\limits_{x_{ref}, u_{ref}} & J(x_r, u_r) \\ \text{s.t.} & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_r \\ u_r \end{bmatrix} = \begin{bmatrix} B_d \hat{d} \\ y_{ref} - C_d \hat{d} \end{bmatrix} \\ & (x_r, u_r) \in \mathbb{Z} \\ & Cx_r + \hat{d} \in \mathbb{Y} \end{cases} \tag{A.29}$$

For the better depiction of the problem, based on the presented equations, the offset-free output MPC algorithm is presented in the Figure A.4.

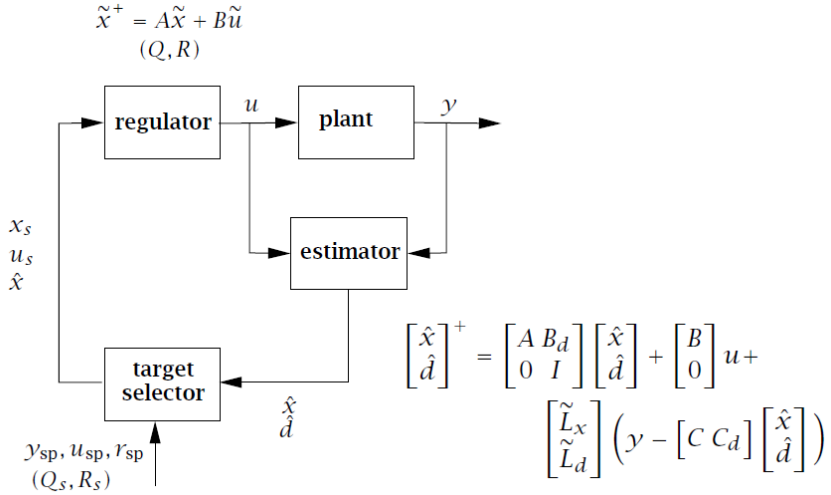


Figure A.4: Offset-free MPC controller structure; derived from [77]

The offset-free formulation of MPC seems to be interesting for the considered orbital robotic system. The disturbances of the plant, as described in the Chapter ??, are inevitable and their effect on the performance of the controller must be addressed.

A.2.4. Explicit MPC

In the already given definitions, the optimal control problem was solved in every time step, if such an iterative numerical procedure is implemented it is referred to as implicit MPC. Along with the increase of the modelled system complexity growths the demand for computational resources. This problem can be partially mitigated if the control law is defined offline, which is called explicit MPC.

The idea standing behind explicit MPC is to solve the optimization problem for all $x(t)$ within a given range of $\mathbb{R}^n$., the explicit representation of the MPC feedback law is obtained off-line using parametric programming. The state space is divided into multiple regions for which the optimization procedure is performed to find the optimal solution within the specified region. Number of regions depends on the possible combinations of active constraints whereas dependence on states is weaker. The explicit problem tends to have lower computation time compared to QP-based implicit problem [35], as no on-line solver is needed in the MPC implementation, which requires only the evaluation of a piecewise affine function. Determining explicitly the structure of MPC provides an insight into the behaviour of the MPC controller in different regions of the state space, highlighting regions where saturation occurs, where solver has multiple optima etc [83]. In order to solve the problem in the region the multiparametric linear programming must be solved (mp-LP). The mp-LP subdivides the space of parameters into the so-called critical regions (CRs), which are the set of all parameter vectors for which a certain combination of constraints is active at the optimizer of the problem. For further mathematical description the work of Alberto Bemporad [83] is referred to.

The main advantages of explicit MPC and the reasons for its appeal in the automotive and aerospace applications are:

- Simple execution code: explicit MPC is a lookup table of affine controllers, selected by linear inequalities.
- Basic operations: the controller implementation requires only sums, multiplications and comparisons.
- Predictability: the worst case number of operations is simple to compute.

On the other hand, explicit MPC requires less online computation, but the off-line construction of the feedback law scales badly with increasing dimensionality of the problem. Moreover, the memory footprint of the explicit solutions can easily violate limits of the available memory storage. Further comparison of the pros and cons of both methods are presented in the [84].

B

Validation of the FK linearization

The linearization procedure of the forward kinematics nonlinear algorithm is presented in this appendix. The validation of the linearization is showed on two data sets. The results are presented in table B.1.

The Jacobian linearization is performed around the current operating point - the state vector x_0, with the method of small perturbations (see section 4.3.1). The perturbation of the value $1e-4$ is added successively to every term from the state vector, next the values are injected into the forward kinematics algorithm. The output of the linearization function is the matrix C_{FK} which allows to map the state vector into the end-effector position. It is important to note, that the linearized definition of the nonlinear algorithm operates around the deviations from the operating point, and not the global value. To say precisely, the matrix C_{FK} is mapping the vector of deviation values from the state vector operating point values Δx to the vector of deviation values from the end-effector operating point value ΔEE_{LIN}. The linearization procedure is performed in order to include the EE reference position tracking goal term linear definition into the optimization cost function. It is underlined that the resulting definition of the goal term (as described in 4.46) is working in the deviation values domain, and not the global values.

The obtained linear equation B.1 output is ΔEE_{LIN}, the deviation from the operating point EE_{op}. In order to obtain the current position of the end-effector using the linearized algorithm, the deviation value is simply added to the operating point value, as in the equation B.2.

$$\Delta EE_{LIN} = C_{FK} \cdot \Delta x \tag{B.1}$$

$$EE_{LIN} = EE_{op} + \Delta EE_{LIN} \tag{B.2}$$

In order to validate the linearization procedure and assess how the magnitude of an error is changing with the increasing value of the deviation from the operating point around which the linearization procedure was performed, the error value is computed as shown below in equation B.3.

$$error = EE_{NL} - EE_{LIN} \tag{B.3}$$

In order to verify the linearization procedure, the validation tests have been performed. The results are presented in table B.1. The linearization is performed around an operating point state vector x_{op}, two values of the state vector were chosen for this test. Then, with the nonlinear forward kinematics algorithm, the end effector position is obtained EE_{op}. The errors are computed for three different conditions - three different values of the Δx. Firstly, the values equal to the perturbation values $1e-04$, as applied in the linearization procedure, are used. Next, higher values of the Δx are tested. It is clear from the table, that the value of the error between the end effector position vector obtained from a nonlinear algorithm and linearized algorithm is very small. For all the three components, the error value remains in the submilimeter domain - specifically the x-position error is equal to $0.297mm$, y-position error is equal to $0.079mm$ and the y-position error is equal to $-0.051mm$. These values are acceptable and they meet the required performance criteria for this type of the mission and precision manipulation. Next, the higher value of the state vector deviation Δx, by one magnitude, is applied. It is clear that it results in much higher error values, the x-position error is again the highest and is equal to $3.69mm$. The y-position error equals to $-0.11mm$ and the z-position equals to $-2.37mm$. The increase of the error is the consequence of applying the mapping matrix C_{FK} on the vector with higher values than the ones used as perturbation values in the linearization. Such increase of an error was expected, the correlation between the increase of the deviation value form the operating point with the increase of an end-effector position error is considered when designing an optimization control problem by avoiding very large prediction horizon, which would result in increasing too much the deviation vector. Last but not least, the deviation vector containing the values of $1[cm]$ for the base position

deviation and 1^o for the attitude deviation of the base and angular position deviation of the manipulator joints is applied. It is clear that the resulting error is in a centimeters domain - the x-position error is equal to $3.695cm$, y-position error is equal to $-0.123cm$ and z-position error is equal to $-2.382cm$. Such values of the error are quite high, and potentially could lead to the unsuccessful capture maneuver. For this reason it is important to keep in mind that if the linearized version of the algorithm is used, its definition remains valid only for the values in close proximity to the operating point.
The second value of the state vector implemented in the linearization validation tests gives very similar results. The end effector position error due to linearization, for the small deviation from an operating point remains in the submilimeter domain. The x-position error is equal to $0.448mm$, the y-position error is equal to $-0.233mm$ and the z-position error is equal to $-0.1mm$. For the higher values of the Δx the error keeps on increasing. When the deviation vector containing the values of $1cm$ for the base position deviation and 1^o for the attitude deviation of the base and angular position deviation of the manipulator joints is applied, the resulting error is in a centimeters domain in a similar way as for the first dataset. The x-position error is equal to $6.357cm$, y-position error is equal to $-5.549cm$ and z-position error is equal to $-3.247cm$. The same conclusions can be reached for this validation test - the linearized algorithm is valid in the very close proximity of the operating point. When the state vector deviation becomes higher the position error becomes bigger, its value of few centimeters is rather unacceptable for the precision operations of the capture maneuver. In order to ensure that the linearized description of the cost function is depicting well enough the real system, the balance between the value of a sampling time and the length of prediction horizon must be found. The linearization shall be done in every simulation iteration step using the updated value of an operating point in order to ensure that the linear algorithm is correctly describing the nonlinear behaviour.

x_{op}	EE_{op} [m]	Δx	ΔEE_{LIN} [m]	EE_{LIN} [m]	$EE_{NL}(x_{op}+\Delta x)$ [m]	$error$ [m]
$\begin{bmatrix} -3m \\ 0 \\ 0 \\ 0 \\ 10^o \\ 0 \\ 0 \\ 45^o \\ 90^o \end{bmatrix}$	$\begin{bmatrix} -1.41315 \\ 0 \\ 2.27871 \end{bmatrix}$	$\begin{bmatrix} 0.0001m \\ 0.0001m \\ 0.0001m \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \end{bmatrix}$	$\begin{bmatrix} -0.00026 \\ 0.00003 \\ 0.00011 \end{bmatrix}$	$\begin{bmatrix} -1.41342 \\ 0.00003 \\ 2.27881 \end{bmatrix}$	$\begin{bmatrix} -1.41312 \\ 0.00011 \\ 2.27876 \end{bmatrix}$	$\begin{bmatrix} 0.000297 \\ 0.000079 \\ -0.000051 \end{bmatrix}$
		$\begin{bmatrix} 0.001m \\ 0.001m \\ 0.001m \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \end{bmatrix}$	$\begin{bmatrix} -0.00383 \\ 0.00123 \\ 0.00257 \end{bmatrix}$	$\begin{bmatrix} -1.41699 \\ 0.00123 \\ 2.28128 \end{bmatrix}$	$\begin{bmatrix} -1.41330 \\ 0.00113 \\ 2.27890 \end{bmatrix}$	$\begin{bmatrix} 0.00369 \\ -0.00011 \\ -0.00237 \end{bmatrix}$
		$\begin{bmatrix} 0.01m \\ 0.01m \\ 0.01m \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \end{bmatrix}$	$\begin{bmatrix} -0.03833 \\ 0.01233 \\ 0.02571 \end{bmatrix}$	$\begin{bmatrix} -1.45148 \\ 0.01233 \\ 2.30442 \end{bmatrix}$	$\begin{bmatrix} -1.41453 \\ 0.01110 \\ 2.28060 \end{bmatrix}$	$\begin{bmatrix} 0.03695 \\ -0.00123 \\ -0.02382 \end{bmatrix}$
$\begin{bmatrix} -3m \\ 1m \\ 1m \\ 5^o \\ 10^o \\ -5^o \\ 30^o \\ 60^o \\ 45^o \end{bmatrix}$	$\begin{bmatrix} -2.30422 \\ 0.29659 \\ 3.45296 \end{bmatrix}$	$\begin{bmatrix} 0.0001m \\ 0.0001m \\ 0.0001m \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \end{bmatrix}$	$\begin{bmatrix} -0.00036 \\ 0.00024 \\ 0.00012 \end{bmatrix}$	$\begin{bmatrix} -2.30458 \\ 0.29683 \\ 3.45308 \end{bmatrix}$	$\begin{bmatrix} -2.30413 \\ 0.29659 \\ 3.45298 \end{bmatrix}$	$\begin{bmatrix} 0.000448 \\ -0.000233 \\ -0.000100 \end{bmatrix}$
		$\begin{bmatrix} 0.001m \\ 0.001m \\ 0.001m \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \end{bmatrix}$	$\begin{bmatrix} -0.00549 \\ 0.00490 \\ 0.00289 \end{bmatrix}$	$\begin{bmatrix} -2.30971 \\ 0.30149 \\ 3.45584 \end{bmatrix}$	$\begin{bmatrix} -2.30338 \\ 0.29594 \\ 3.45261 \end{bmatrix}$	$\begin{bmatrix} 0.00633 \\ -0.00555 \\ -0.00324 \end{bmatrix}$
		$\begin{bmatrix} 0.01m \\ 0.01m \\ 0.01m \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \end{bmatrix}$	$\begin{bmatrix} -0.05491 \\ 0.04900 \\ 0.02888 \end{bmatrix}$	$\begin{bmatrix} -2.35914 \\ 0.34559 \\ 3.48183 \end{bmatrix}$	$\begin{bmatrix} -2.29556 \\ 0.29010 \\ 3.44936 \end{bmatrix}$	$\begin{bmatrix} 0.06357 \\ -0.05549 \\ -0.03247 \end{bmatrix}$

Table B.1: Validation of the Forward Kinematics linearization algorithm

C

Validation of the FoV linearization

The linearization procedure of the field of view nonlinear algorithm is presented in this appendix. The validation of the linearization is showed on two data sets. The results are presented in table C.1.

The nonlinear definition of the FoV term is as described in Section 4.6. The Jacobian linearization is performed around the current operating point - the state vector x_0, with the method of small perturbations (see section 4.3.1). The perturbation of the value $1e-4$ is added successively to every term from the state vector, next the values are injected into the algorithm computing the cosine angle between the two vectors as explained in the section 4.6. The output of the linearization function is the matrix C_{FoV} which allows to map the state vector into the value of the cosine function. It is important to note, that the linearized definition of the nonlinear algorithm operates around the deviations from the operating point, and not the global value. To say precisely, the matrix C_{FoV} is mapping the vector of deviation values from the state vector operating point values Δx to the vector of deviation values from the cosine function operating point value $\Delta FoVcos$. The linearization procedure is performed in order to include the field of view term linear definition into the optimization cost function. It is underlined that the resulting definition of the goal term (as described in 4.46) is working in the deviation values domain, and not the global values.

The obtained linear equation C.1 output is $\Delta FoVcos_{LIN}$, the deviation from the operating point $FoVcos_{op}$. In order to obtain the current value of the cosine between two vectors using the linearized algorithm, the deviation value is simply added to the operating point value, as in the equation C.2.

$$\Delta FoVcos_{LIN} = C_{FoV} \cdot \Delta x \tag{C.1}$$

$$FoVcos_{LIN} = FoVcos_{op} + \Delta FoVcos_{LIN} \tag{C.2}$$

The performance metric is defined in order to assess the validity of the linearized definition, for this purpose an error is computed. It is defined as a numerical difference in a value of the cosine function of the FoV angle obtained with the nonlinear algorithm and the linear algorithm, as shown in equation C.3.

$$error[-] = FoVcos_{NL} - FoVcos_{LIN} \tag{C.3}$$

In order to better understand the magnitude of the error, it is translated into the degrees unit such that the difference between the angle obtained with the nonlinear algorithm and the angle obtained with the linear algorithm is computed as shown in equation C.4.

$$error[^o] = arccos(FoVcos_{NL}) - arccos(FoVcos_{LIN}) \tag{C.4}$$

Next, the validation tests have been performed in order to verify the linearization procedure. The input data and the outcomes are presented in table C.1 below. The angular position values are always input to the algorithm in radian unit, however in order to provide more reader-friendly description, some values are given in degrees [^{o}] if found appropriate. The linearization is performed around an operating point state vector x_{op}, two values of the state vector were chosen for this test. Then, with the nonlinear field of view algorithm, the value of the cosine is obtained $FoVcos_{op}$, and the value of the cosine $FoVcos_{LIN}$ with the linear algorithm. The errors are computed for three different conditions - three different values of the Δx. Firstly, the values equal to the perturbation values $1e-04$, as applied in the linearization procedure, are used. Next, higher values of the Δx are tested. It is clear from the table, that the error between the value of the cosine obtained from a nonlinear algorithm and linearized algorithm is very small. The error value in degrees is very small it is equal to 0.00988^o. Therefore it is concluded that for the value of the deviation from the state vector operating point equal to the value of the perturbation used in the linearization process, the linear algorithm performance is very good and the error is negligible. Next, the higher value of the state vector deviation Δx, by one magnitude, is applied. It is clear that it results in a higher

error value, which is equal to 0.23958^o. The value of the error is low enough to remain negligible. The increase of the error is the consequence of applying the mapping matrix C_{FK} on the vector with higher values than the ones used as perturbation values in the linearization. Last but not least, the deviation vector containing the values of $1[cm]$ for the base position deviation and 1^o for the attitude deviation of the base and angular position deviation of the manipulator joints is applied. It is clear that the resulting error is much higher and it rises up to the value of 2.47809^o. Nevertheless, this mismatch between the real value obtained from the nonlinear algorithm and the value obtained from the linearized algorithm should not cause any major issues, as the goal of the field of view term is to keep the target point within the visibility of the navigation camera and hence to minimize the angle between the two vectors (see fig. 4.2) any small deviation is acceptable due to the fact it is still expected to remain in the view cone of the navigation camera which usually for the close navigation camera has the aperture angle of between 15-40 deg.

x_{op}	$FoVcos_{op}$	Δx	$\Delta FoVcos_{LIN}$	$FoVcos_{LIN}$	$FoVcos_{NL}(x_{op}+\Delta x)$	$error[-]$	$error[^o]$
$\begin{bmatrix} -3m \\ 0 \\ 0 \\ 0 \\ 10^o \\ 0 \\ 0 \\ 45^o \\ 90^o \end{bmatrix}$	0.76979	$\begin{bmatrix} 0.0001m \\ 0.0001m \\ 0.0001m \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \end{bmatrix}$	0.00016	0.76995	0.76984	−0.000114	0.00988
		$\begin{bmatrix} 0.001m \\ 0.001m \\ 0.001m \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \end{bmatrix}$	0.00311	0.77290	0.77024	−0.002660	0.23958
		$\begin{bmatrix} 0.01m \\ 0.01m \\ 0.01m \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \end{bmatrix}$	0.03110	0.80089	0.77425	−0.02664	2.47809
$\begin{bmatrix} -3m \\ 1m \\ 1m \\ 5^o \\ 10^o \\ -5^o \\ 30^o \\ 60^o \\ 45^o \end{bmatrix}$	0.52557	$\begin{bmatrix} 0.0001m \\ 0.0001m \\ 0.0001m \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \\ 0.0001\text{rad} \end{bmatrix}$	0.00014	0.52571	0.52547	−0.000241	0.01616
		$\begin{bmatrix} 0.001m \\ 0.001m \\ 0.001m \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \\ 0.1^o \end{bmatrix}$	0.00177	0.52734	0.52459	−0.00275	0.18526
		$\begin{bmatrix} 0.01m \\ 0.01m \\ 0.01m \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \\ 1^o \end{bmatrix}$	0.01767	0.54324	0.51581	−0.02744	1.85286

Table C.1: Validation of the Field of View linearization algorithm