POLITECNICO DI TORINO

Master's Degree in Mechatronics Engineering

Master's Degree Thesis

Development and Optimization of a Rendezvous Manoeuvre for Robotic Facility Testing & Validation

Supervisors

Prof. Elisa CAPELLO

Ing. Francesco CACCIATORE

Ing. Jesús F. RAMÍREZ SÁNCHEZ

Candidate

Lucrezia MARCOVALDI

December 2023

Ackonwledgements

Innanzitutto vorrei ringraziare caldamente la mia relatrice, la professoressa Elisa Capello, per avermi concesso la possibilità di svolgere questo lavoro, per i preziosi consigli dispensati e per il costante supporto durante tutto lo svolgimento della tesi.

Ringrazio i miei correlatori: l'ingegner Francesco Cacciatore, per avermi accolto alla SENER Aeroespacial e per essere stato un assoluto punto di riferimento a livello professionale e umano, e l'ingegner Jesus Fernando Ramírez Sánchez, per tutto ciò che mi ha insegnato e per avermi fatto appassionare ogni giorno di più al mondo che si cela dietro agli argomenti legati a questo progetto.

Ringrazio Ester, il professor Hodei Urrutxua Cereijo e i miei amici e colleghi in SENER Aeroespacial, lavorare e crescere fianco a fianco a voi è un privilegio.
Ringrazio Maria, che con la sua dolcezza e la sua forza è stata il mio punto fermo ogni volta che i dubbi prendevano il sopravvento.

Ringrazio mia mamma Laura e mio papà Roberto, per tutto l'amore che mi dimostrate in ogni occasione e per il vostro continuo sostengo, siete la mia ispirazione. Spero di avervi reso orgogliosi di me.

Infine, vorrei dedicare questo lavoro a mia nonna Elisabetta, che mi raccomanda sempre di mettere in tutto ciò che faccio tutta la determinazione e il coraggio di cui sono capace. Mi auguro di diventare almeno la metà della persona che sei.

Abstract

Optimization of space manoeuvres is becoming a key requirement for the evolution of the space market and paves the way towards new investigation scenarios: the purpose of this work is to optimize rendezvous manoeuvres in the context of in-space logistics services, such as in-orbit maintenance, propellant depot and refuelling, considering the spacecraft trajectory and the dispatching of the limited available resources. The aim is to plant more cost-effective, sustainable and performing missions, with the implementation of the optimized guidance algorithms. The optimized manoeuvre has been simulated online through *Hardware-In-the-Loop* (HIL) testing, to investigate the feasibility and robustness of the optimization problem, replicating the 6 degree-of-freedom motion of the spacecraft (in the context of rendezvous manoeuvres, the so called *chaser*), through a cooperative robotic arm. In order to execute properly the simulations, the work envelope of the manipulator has been investigated, to overcome the issues related to the replication of a trajectory optimized for a spacecraft with a tool subjected to limitations over its maximum extension and to the risk of reaching singular configurations.

From the point of view of the guidance of the chaser, the optimization strategy that has been applied is based on a Model Predictive Control (MPC) scheme, due to its capability of adjusting control inputs in real-time to respond to variations in the system's behaviour, adaptability (making it suitable for real-time implementations) and sensitivity to system's dynamics. The optimization problem has been formulated and solved in Matlab® exploiting a toolbox developed in-house by SENER Aeroespacial, SENER Optimization Toolbox (SOTB), powered by an Interior-Point Method solver.

For what concerns the control allocation and the optimization of the resources available on the spacecraft, such as fuel consumption, the problem has been optimized formulating a Quadratic Programming (QP) and developing an *Active Set Method* (Active-Set Solver (ASM)) based solver in Matlab®. For the peculiar case of control allocation problems, the ASM provides rapid convergence when a warm start of the active set is provided and smooth transitions between consecutive optimizations, minimizing changing in control inputs and maintaining system stability. The designed solver doesn't recall any predefined Matlab® function, and has been robustified to handle ill-conditioned Hessian matrices of the QP, exploiting Singular Value Decomposition.

The accuracy of the ASM has been assessed with the applications to five different formulations of the problem under analysis, aiming at minimizing the error between the control actions generated by the actuators and the values of force and torque required to complete the optimized trajectory. In addition, the ASM has been evaluated on its capability to distribute homogeneously the control actions between the actuators. Finally, the optimal solution found with the ASM has been compared to the force and torque values commanded by the control loop of the robotic facility, while the correspondent execution time have been compared with SOTB, with the objective of verifying the applicability of the solver to online optimization of the control allocation problem and integration in the HIL simulation facility.

Contents

List of Figures

List of Acronyms

ASM	Active-Set Solver
DOF	Degree Of Freedom
GNC	Guidance, Navigation and Control
HIL	Hardware-In-the-Loop
IPM	Interior-Point Method
ISTV	In-Space Trasportation Vehicle
LEO	Low-Earth Orbit
LMPC	Linear Model Predictive Control
PWM	Pulse Width Modulation
QCQP	Quadratic Constrained Quadratic Programming
QP	Quadratic Programming
RCS	Reaction Control System
RvD	Rendezvous and Docking
SOTB	SENER Optimization Toolbox
SVD	Singular Value Decomposition
TCP	Tool Central Point
URJC	Universidad Rey Juan Carlos

List of Tables

Chapter 1

Introduction

The evolution of the space market questions for more innovative solutions in traditional space logistics and, basing on this purpose, automatic docking maneuvers have grown in prominence as a means of carrying out in-orbit servicing operations autonomously, particularly those involving physical contact (such as low relative-speed impact). Additionally, space logistics includes missions with the purpose of extending satellites' operational lifetime in orbit, rather than merely replacing them with new ones: sustainability is encouraged by the growing complexity and expense of satellite operations. All this elements lead to the necessity of reliable RvD operations carried out possibly autonomously in space between space vehicles.
The most demanding tasks to be managed in the context of space logistics include:

- In-orbit servicing such as refuelling, maintenance, propellant depot
- In-space assembly operations
- Debris removal
- Return to Earth

This work presents the design and verification of robotic-based HIL facility, with the aim of testing optimized Guidance algorithms developed for the early prototyping of Guidance, Navigation and Control (GNC) systems.
In particular, convex optimization has been adopted to plan the manoeuvres and carry-out the simulation of the capture in a cooperative scenario regarding a LEO, while in parallel an ASM has been designed to exploit again convex optimization to optimize the propellant-usage of the virtual spacecraft involved in the simulation. The RvD testing activities have been conducted with the support of the GNC Laboratory of the URJC, whose instrumentation will be presented in the next chapters to better describe and justify the choices behind the implementation of the control loop for the robotic facility and the design of the algorithms.
Performing Hardware-In-the-Loop (HIL) testings on-ground is part of the model-based design in aerospace missions planning. The models created represents aspects of systems behaviour, structure and interactions, providing to the designer a clear visual representation of the system behaviour. Design requirements are verified, to ensure that the system will perform as expected in critical aerospace environments; it gets also more intuitive to detect earlier issues, resolving before potential problems and reducing the chances of human error. Finally, model-based approaches helps in demonstrating compliance to regulations and makes it easier to modify the software over its lifecycle, ensuring that changes are made with minimal disruption to mission-critical systems.

Optimizing manoeuvres in space its a critical task as it powers many aspects of a mission,

starting from the significant reduction of the already high costs when it comes to save fuel and resources in general, often limited (minimizing fuel consumption, for example, extends the operational lifetime of the space vehicle). Optimization ensures a further degree of robustification against uncertainties during the planning of the manoeuvres, leading to more precise trajectories (for example when it comes to collision avoidance) and safer missions, and in many cases shorten the times needed for a mission, especially for human missions in which the minimization of the exposure to space radiation has to be taken into account. Last but not least, as space becomes more crowded, optimizing manoeuvres helps in manage space traffic, avoid orbital congestion and consequently reduces the risk of orbital debris generation, regulating the timing and trajectories of launches to avoid interference with existing space assets.
In this context, *convex* optimization is a powerful tool because it offers a systematic and efficient way to solve a wide range of optimization problems, characterized by the existence and uniqueness of a global optimal and a relatively fast convergence to the solution. Formulating a problem adopting convex optimization involves, in many cases, the possibility of handle it by polynomial-time complex algorithms in terms of problem size, making it computationally efficient when it comes to embedded applications, and a high level of robustness against noise and perturbation in problem data, crucial for real-world applications.
In order to optimize on-board the manoeuvre, numerical optimal control has been exploited , basing the trajectory optimization on a Linear Model Predictive Control (LMPC) scheme, powered by an Interior-Point Method (IPM) solver. In aerospace application LMPC is useful because it can provide precise control of complex systems, taking into account dynamics and perturbations to provide as outputs optimal control actions, handling constraints like fuel limitations, physical limitations of the actuators and stability requirements.
In parallel to the optimization of the trajectory, an Active Set Method based solver has been designed and implemented to carry out the optimization of the control allocation, based on thrust dispatching techniques and on a Quadratic Programming (QP) optimization scheme. QP results to be an interesting choice when it comes to resources allocation because it handles efficiently capacity limits without introducing, at the same time, an heavy computational burden in the entire optimization sequence.
This work is organized as follows. Chapter 1 presents the review of the main references from which information have been collected, together with the description of SENER Aeroespacial facilities used while developing of the HIL test. Chapter 2 focus on the main concepts behind space orbital dynamics and on the theory behind the adopted convex optimization algorithms ; the third Chapter presents the formulation of the translational state problem of the rendezvous manoeuvre under study and the implementation of MPC control scheme. Chapter 4 focuses on the Control Allocation problem, especially on thrust dispatching techniques, while Chapter 5 describes in depth what is an HIL test and how it has been planned in this work. Finally, Chapters 6&7 show testings results and present future developments for the improvement of the facility.

1.1 Mission description

In the context of in-space logistics, RvD manoeuvres play a crucial role, concerning a chaser spacecraft that safely and reliably approximates another orbiting vehicle/debris/probe in a non-inertial reference frame. The maneuver that has been optimized and simulated in this work entails a V-bar approach between two cooperative In-Space Trasportation Vehicle (ISTV) (the target is able to send information to the service satellite regarding its state), focusing on the last 2[m] of the trajectory between two holding points and located in a Low Earth Orbit (Low-Earth Orbit (LEO)) orbit (a circular or quasi-circular orbit that has an altitude of less than 2000 kilometers). LEO orbits are a common choice when it comes to transportation, telecommunication and Earth surface observation, due

to their close proximity to the Earth's surface and the comparatively brief orbital period; on the other side, carrying an in- orbit demonstration in a LEO presents disadvantages, such as the fast orbital decay due to the strong influence of air drag, and the small field of view from the surface, which affects visibility time windows between ground station and satellites.
When talking about ISTV, an orbital flight vehicle performing space logistics is indicated. Some examples include:

- *Motorised dispensers*: spacecraft intended to provide mixed tiny satellite payloads with maneuverable propulsion capabilities, so that they can be precisely delivered to the desired orbits
- *Space tugs*: spacecraft designed to transfer new payloads from one orbit to another, with integrated automatic GNC capabilities
- *Kick-Stages*: space vehicle to transport heavy payloads to complex orbits with high energy transfer, such as Moon or Mars missions
- *Capsules*: end-to-end transportation of cargo or humans

Basic elements of a RvD mission are the chaser and the target spacecrafts. With the term *chaser* is indicated a fully operational space vehicle that performs the RvD, while the *target* is the spacecraft with respect to which the RvD manoeuvre is planned. Targets usually are passive and they can be cooperative or uncooperative, depending if they provide their state information to the chaser and if they are capable to control their state. For what concerns the mission taken into account in this work, the chaser has been provided with 16 cold-gas thrusters, in order to be more precise in controlling the docking manouvre.
The whole manoeuvre is subdivided in phases: first, the **launch and orbit insertion** take place, launching the chaser in its own orbit. Then, the **phasing** regards the moment in which the chaser spacecraft adjusts its orbit by changing altitude or inclination; at this point, the **initial approach** starts, with the chaser beginning its approach towards the target at a safe distance. When the chaser gets closer to the target, generally firing smaller size thrusters for precision, the **rendezvous phase** starts, with the chaser that reaches the desired distance and relative position to dock with the target spacecraft. If the objective is to dock, the spacecrafts physically connect and post-rendezvous operations take place, such as crew transfer, cargo exchanges or data collection.

In order to analyze the scenario, three different reference frames have been adopted:

- **Earth Centered Inertial (ECI):** The fundamental plane of this system corresponds to the equator of the planet and places its origin in the planet's center. The right-handed triad is made up by the I axis, which points toward the vernal equinox, the K axis, which runs across the North Pole, and J that completes the triad. The aforementioned definition does not constitute an inertial frame, because the equinox and the equatorial plane both gradually move with time as a result of precession, nutation and other secular motions. By referencing the axes directions at a specific epoch and defining the transformations to go from the present epoch to the reference one and viceversa, a *pseudo* Newtonian inertial system can be achieved.
  ECI spherical coordinates are commonly denoted as:
  - **Right ascension** α**:** the angle between the vernal equinox and the celestial meridian containing the object, measured eastward on the equator.
  - **Declination** δ**:** the equatorial plane's angle with the object, expressed as a positive angle above the equator in the meridian plane in which the object travels.

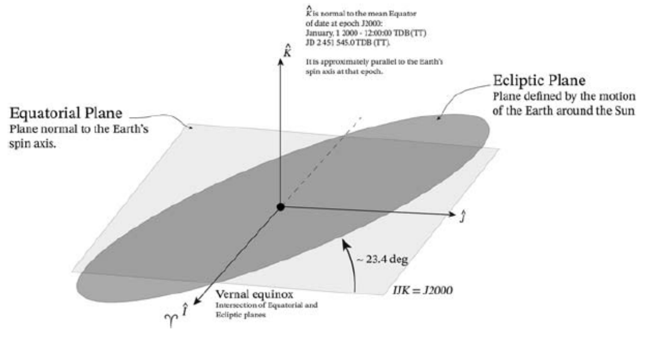

$\hat{K}$ is normal to the mean Equator of date at epoch J2000:
January, 1 2000 - 12:00:00 TDB(TT)
JD 2 451 545.0 TDB (TT).
It is approximately parallel to the Earth's spin axis at that epoch.
$\hat{K}$
Ecliptic Plane
Plane defined by the motion of the Earth around the Sun
Equatorial Plane
Plane normal to the Earth's spin axis.
$\hat{J}$
~ 23.4 deg
$IJK = J2000$
Vernal equinox
Intersection of Equatorial and Ecliptic planes
Υ $\hat{I}$

Figure 1.1: ECI J2000 reference system, picture taken from [42]

– **Radial distance:** the distance between the item and the Earth's center.

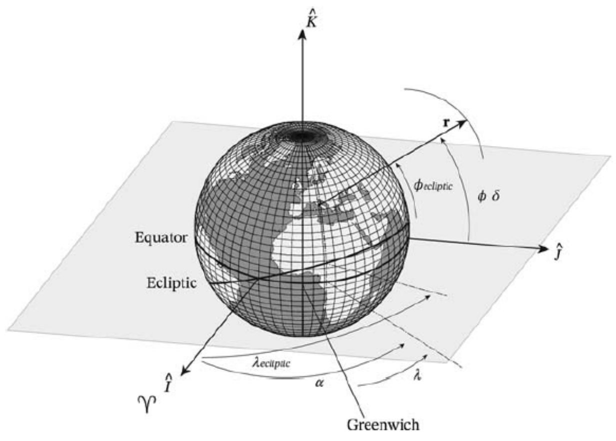


Figure 1.2: Right ascension, α, declination, δ, longitude, λ, latitude, ϕ, as spherical coordinates, to define position; picture taken from [42]

- **The orbital frame or LVLH frame:**The satellite is shown as the origin in the LVLH (Local Vertical, Local Horizontal) coordinate system. As the satellite moves through the orbit (Local Vertical), the R axis constantly points away from it along the Earth's radius vector. The orbital plane is perpendicular to the W axis. The right-handed triad is completed by the S axis. The S axis is pointing in the direction of the velocity vector and is perpendicular to the local horizontal (horizontal) radius vector.

- **The body frame:**The COM or geometrical center of the satellite is where the satellite body coordinate system is placed. The axes are in line with prominent body-defined directions, like the primary inertia axes or the geometric axes.

The scenario under test can be resumed in the above illustration, where P_1 stands for the phasing end, P_2 closes the initial approach phase, P_3 the rendezvous phase. From point P_3 on, the chaser spacecraft receives data about the target attitude and docking port orientation at point and the docking begins.
If the chaser arrives at the docking location or approaches target's lateral surface with a relative velocity less than a user-specified threshold, the docking maneuver is deemed

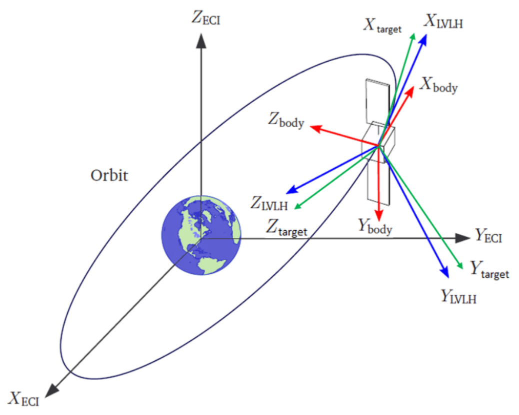


Figure 1.3: Illustration showing the ECI reference frame (black), the body-fixed reference frame (red), the LVLH reference frame (blue). To obtain the target orinentation, is commonly defined a target reference frame (green) too, with one axis oriented in the direction of the projection of the target on Earth surface (Z_{Target}), Y_{Target} given by the cross product between Z_{Target} and the spacecraft velocity vector and X_{Target} orthogonal to the previous ones. The picture is taken from [21]

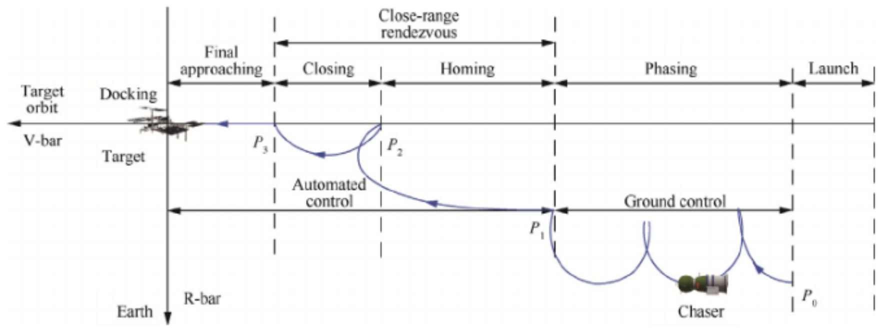


Figure 1.4: Illustration of spacecraft rendezvous and docking process, the picture is taken from [12]

successful, ensuring structural integrity and enabling a soft coupling between the chaser and the target.

1.2 Literature review

The optimization of spacecraft guidance has been long researched, with a variety of approaches to handle the high complexity and in many cases non-convexity of the problem under study.
It's interesting to notice that in the majority of studies the target spacecraft isn't maneuvered during the rendezvous by firing thrusters, and that the disturbance forces on the chaser and target spacecraft are not taken into account. Although topics range from 2 Degree Of Freedom (DOF) and 3 DOF position-only rendezvous in circular orbits to 6 DOF docking to tumbling targets in elliptical orbits, guidance and control of rendezvous to a stationary target in near circular orbit is likely the subject of the most research.
The goal of optimum control problems is typically either minimizing fuel usage, time consumption, or a combination of the two. On-line a priori trajectory planning and real-time

online computations are also investigated, along with formal proofs for more scholarly applications.
In order to better enfocus the requirements of the problem under analysis, [22] has been studied to investigate (GNC) systems for autonomous proximity operations and docking of two spacecrafts; in this case, the guidance was implemented by integrating state-dependent Riccati equations derived from relative motion dynamics and relative navigation employing vision sensor systems. The guidance strategy for the capturing phase is based on the Clohessy-Wiltshire state transition matrix and a V-bar hopping approach reference trajectory is defined, as it is assumed also in this thesis.
[15] has been taken has reference for what concerns online trajectory planning, even if in this case it is done for a tumbling target involved in a rendezvous manoeuvre, modeling the dynamics through linearization of the HCW equations. The target experiences a translational change due to an active maneuver or debris impact, and uses previously planned trajectory and control as a starting approximation to solve the newer planning faster, with constraints such as a cuboid no-fly zone around the target. It has been assumed that the essential information for the algorithm is provided correctly enough by a visual navigation system installed on a service spacecraft, as it has been assumed during the implementation of the control loop in this work.
Also in [30] guidance is optimized to control position (and attitude) for cooperatively docking of several small satellites. It is presented a 6 DOF position and attitude guidance regarding the approach between a chaser and a target in a LEO orbit, that must go to a specified relative location while consuming the least amount of fuel and making a soft approach (velocity is zero at the intended target point). The trajectory control method is based on relative motion formulation by Hill's equations, while *E-guidance* is adopted to optimize fuel consumption: thrust commands for the guidance of the vehicle are optimized given initial and final conditions, building the so called *E-matrix*, that relates the desired final state with the current state through guidance coefficients.
[33] has been studied for what concerns control optimization for CubeSats docking in a LEO orbit, with the actuation system regarding cold-gas propulsion operated in Pulse Width Modulation (PWM) (as it has been planned for the chaser spacecraft presented in this work). Also [26] presents the optimization of a rendezvous where the chaser is controlled by Reaction Control System (RCS) thrusters, and each thruster can deliver a constant thrust for a varied amount of time (PWM implementation). However, in this case the study describes an alternative optimization solution to the classical optimization approaches for nonconvex control problems with discrete logic constraints, that involve binary variables and mixed-integer programming that are generally slow and computationally expensive. [32] resulted to be very interesting from the point of view of investigating further goals and requirements when it comes to fuel-optimal paths accounting for system dynamics and constraints on operation. Lossless convexification has been presented to enforce non-convex thrust limits and to prevent from plume impingement on the target during docking maneuvers, leading to the definition of a Second-Order Cone Programming framework. It has been shown that the problem can be solved reliability and in real time on low-powered embedded processors.

In order to clarify the design requirements to be fulfilled when planning a rendezvous manoeuvre, [40] has been taken as reference with respect to the performance of proximity and capture operations for on-orbit services. The document aims at drawing the attention to the fact that there are no defined and widely agreed technical and safety standards for on-orbit servicing manoeuvres planning, so it presents recommendations regarding how to improve the safety and sustainability of the missions, together with design principles applicable to all the phases of the manoeuvre.
In [38], it is addressed the problem of spacecraft rendezvous with a target orbiting in an eccentric trajectory, with the space vehicle equipped with reaction wheels and an arbitrary number of thrusters, in order to have a versatile configuration for a range of

mission types. A hybrid system model is employed, where propulsive actions are treated as impulses, while attitude control is continuously regulated over time. The proposed solution methodology leverages the translational state transition matrix and the attitude flatness property to transform the time-continous dynamics into algebric relations. This transformation enables the reformulation of the optimal control problem into an equivalent, parametrized and discretized form, yielding a finite static program suitable for computational analysis. The paper introduces a closed-loop MPC scheme, based on the linearization of the dynamcis, with the dual purpose of mitigating disturbances and accomodating unmodeled dynamics, enhancing the reliability and precision of the rendezvous manouvre.
In [19] spacecraft rendezvous is handled again using MPC, providing with HCW equations the model that characterized spacecraft relative motion. Within this framework, the paper explores two different scenarios: the first one emphasizes position control while considering fuel constraints, while the second scenario extends the challenge by integrating obstacle avoidance into the position control problem. To pursuit optimized rendezvous, fuel consumption and obstacle avoidance are incorporated into the cost function to force efficiency and safety, that have been further tested through a set of comprehensive simulations.
Autonomous docking through MPC is investigated in [24] always employing HCW model to predict proximity trajectory, utilizing a receding prediction horizon to ensure finite-time completion of the docking manoeuvre, while fuel consumption is minimized by incorporating a 2-norm cost index related to the control input. On the other hand, in [27] it is presented a comparison between three significant trajectory generation methods rooted in convex optimization, such as Lossless Convexification and Sequential Convex Programming. In the first case, Pontryagin's maximum principle is adopted to show that a convex relaxation of a nonconvex problem leads to the globally optimal solution to the original problem, concerning nonconvex control constraints, such as an input norm lower bound and a nonconvex pointing constraint (lossless relaxation of nonconvex state constraints remains under investigation). In the second case, exploiting the idea of iterative convex approximation, SCP is applied to show its performance when it comes to safety-critical applications in disciplines such as aerospace and automotive engineering, and to obtain theoretical guarantees on computational complexity, in opposition to general NLP optimization where the convergence guarantees are weaker.
[18] puts the focus on the enhancing of a good level of robustness while performing trajectory optimization, determining probabilistic optimal trajectories accounting for uncertainties in mission critical parameters. The manoeuvres are planned to be optimized on-board, improving fuel consumption and tracking performance.

Drawbacks of MPC have also been taken into account during the guidance planning: one of them is the heavy computational burden when it comes to the implementation of on-line guidance on the on-board computer of a spacecraft. In [17] it is shown how the computing cost is heavily influenced by how the optimal control problem is formulated, presenting a compact and sparse alternative formulation of the MPC based on a variable change that results in a block banded Hessian. In this situation, the problem can be solved in linear time in the horizon length using an IPM. In order to better understand the implementation of an MPC in SENER Optimization Toolbox (SOTB) and its range of usage, [35] provided a demonstration of MPC usage to regulate the re-entering phase of a parafoil in closed-loop, reducing the impact velocity. The optimization problem is solved by SOTB, and in addition various design elements that are critical for getting the desired results, such as robustifying the formulation against uncertainties through a constraint-tightening method, have been taken as reference for the development of the thesis.
For what concerns HIL testing of approach manoeuvres for in-orbit logistics, the European Proximity Operations Simulator (EPOS 2.0) [7] set a milestone in the field of rendezvous

operations on-ground, constituting a state-of-art facility. This large-scale simulator plays a crucial role in missions involving orbital maintenance and towing services, which demand highly complex Rendezvous and Docking (RvD) manoeuvres. One of the primary challenges addressed by EPOS 2.0 is the approach to autonomous, non-cooperative, passive client satellites that lack specialized rendezvous equipment. It serves as hub for designing, developing and verifying rendezvous sensors and systems, basing on two industrial robots mounted on a 25-meter-long rail, which enables real-time testing of satellite approach. In this set-up, one robot simulates the service satellite, while the other replicates the movement of the client satellite. EPOS 2.0 is characterized by exceptional precision, with sub-millimeter accuracy over a 25-meter span, and a commanding frequency of 250 Hertz. To ensure realistic testing conditions, the facility is equipped with a high-performance solar simulator, which replicated ambient lighting- a critical consideration when evaluating optical sensors.

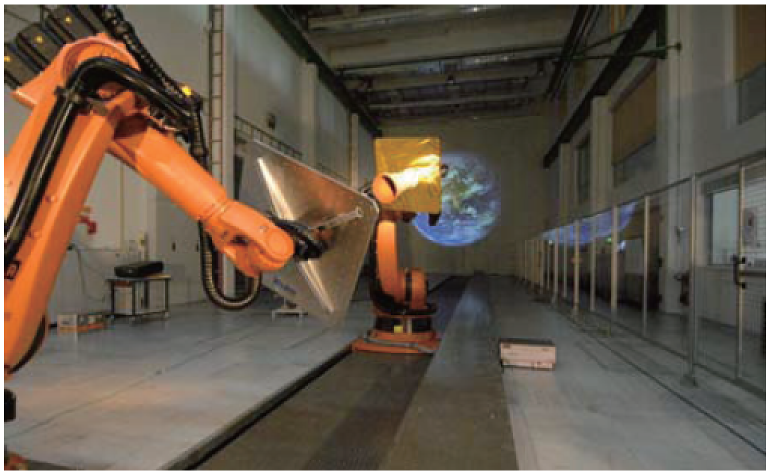


Figure 1.5: EPOS facility: the robotics-based testbed, picture taken from [7]

Also in [37] is presented a comprehensive framework for the autonomous capture and servicing of satellites, shedding light on the crucial aspects of autonomy and remote operations. The paper is based on laboratory experiments that underscore the practicality and applicability of autonomous servicing operations. The core problem tackled is the one of a satellite capture representative of a broad spectrum of on-orbit robotic manipulation tasks within a known and structured environment, target in free flight and with the quality of communication linkages influenced by bandwidth limitations and communication dropouts.
Docking experiments in-space are described in [11], accounting for the handling of anomalies with trajectory optimization through the usage of SPHERES, a test bed constituted by three satellites, each one provided with a GNC module to perform state estimations form sensors and executing control algorithms with actuators.

The solver that has been used to optimize the rendezvous problem under analysis in this work is powered by an IPM; however, well documented are also application of the alternative ASM to the MPC control scheme, as reported in [34] where a primal ASM for the efficient solution of the block-sparse QP that characterize the MPC is presented, called *PRESAS*. The solver is implemented in a standalone C code and presents good computational performance if compared to the present state of the art in the field of linear and nonlinear MPC case studies.
In [20] a comparison between the implementation of IPM and ASM is presented, when applied to MPC. The parameters under analysis are convergence speed, computational

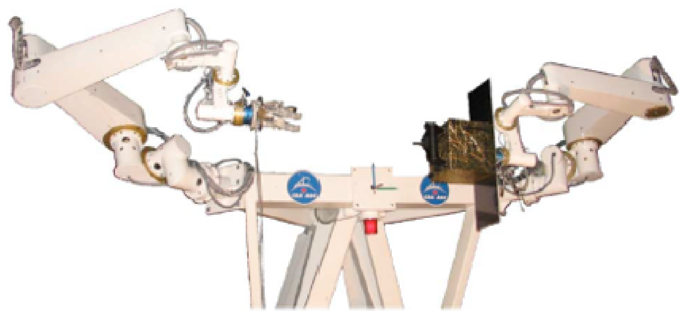


Figure 1.6: Simulation of a docking scenario with a robotic system: the manipulator on the left side of the figure docks through an hand gripper, while the one on the right presents a mock-up. The picture is taken from [37]

complexity, storage and practical implementation issues; from the analysis, it turns out that ASM converges faster if the number of optimization variables is small, otherwise IPM represents a better choice due to its scalability.
In order to apply the ASM to the control allocation problem and to formulate the problem as a QP, [6] has been investigated to compare the performance and computing requirements of control allocation optimization algorithms.Two different control allocation problems are presented: a direct allocation approach and a mixed optimization strategy, that minimizes the error between the desired and accomplished moments as well as the control effort. Constrained optimization problems are converted into linear programs, which can be solved using linear programming techniques like the simplex algorithm, but to speed up computations also a redistributed pseudoinverse methodology and a fixed-point solution with low processing requirements were built for comparison. In [16] it is presented an implementation of an ASM to the Control Allocation problem, to spread the control effort among the actuators. The problem is presented as a least-squares problem, to which it is applied an ASM algorithm that always identifies the optimal control distribution and it is shown through simulation that the computational timing requirements are met.
[14] also presents a comparison of the performance of three algorithms in order to solve a QP: null space method, ASM and gradient projection method, to optimize quadratic functions. In order to implement the ASM, [25] has been studied to understand the fundamental requirements behind the application of optimization theory to aerospace applications, while [28] has been taken as reference for the implementation of the algorithms itself and the initial feasible guess, together with [31], that has been investigated to clarify the requirements that an efficient ASM should have when it comes to MPC real-time embedded implementations, that is the final goal of the solver designed, both in sparse and dense setup.

1.3 Facility

These last three paragraph will introduce the facility adopted to plan the optimization of the manoeuvre, two of them developed in-house in SENER Aeroespacial.

1.3.1 SENER Optimization ToolBox

SOTB is a numerical optimization toolbox focusing on optimal control developed in Matlab® in-house in SENER Aeroespacial. It offers both linear and non-linear numerical optimal control and its optimization algorithms are powered by an IPM solver, enabling rapid development and testing in industrial applications (the entire facility is suitable for autocoding in C and C++). The tool presents a user-friendly interface that allows the formulation of complex optimization problems into more direct and easier implementations.
During the verification phases of the GNC facility design, SOTB has been used to build the optimization algorithms and it has been integrated with extra features, specific for the scenario under study, in particular the Shrinking Horizon algorithm for the MPC has been modified in order to exploit a dense formulation of the problem, save memory allocation and include customized constraints for the last stage of the prediction horizon.

1.3.2 Robotic contact dynamics test facilities

In order to carry out the simulations of RvD processes, the GNC Laboratory of the URJC in Madrid made available the hardware equipment. Due to its effectiveness in performing

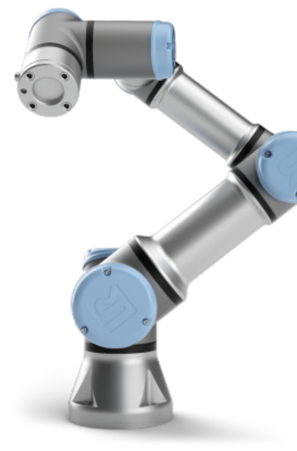


Figure 1.7: UR3e robotic arm model, picture taken from [41]

motions in limited workspaces, an UR3e industrial cooperative robotic arm has been adopted to simulate the chaser motion; the robot is equipped with 6 rotating joints, each of which allows movements in the range of 360°.
The robotic arm is integrated with a computer-based monitoring and control system that the user access through a control box connected to a teach pendant, to modify manually robot joints and monitor the variables in the data pool. The teach pendant gives the user the possibility of controlling the robotic arms in free-drift and preventing serious harms due to singularities, which are workspace configurations that cause the loss of one or more degrees of freedom. Alternatively to the teach pendant, each robot is controlled in real-time by its own local control unit, provided by the robot manufacturer.
The robotic arm processes every commanded movements with respect to its own reference systems:

- **Single joints reference system:** Six separate servo-controlled axes on each robot allow the end-effector to be moved in relation to the base of the robot.

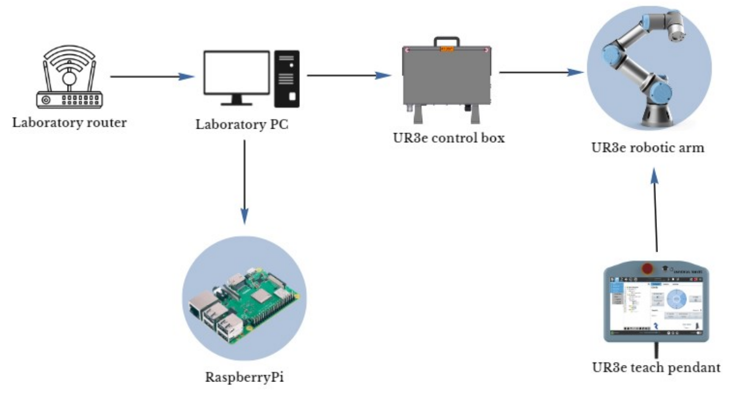


Figure 1.8: Robotic arm control options: the robot presents a control box that communicates with the teach pendant and with the PC of the GNC Laboratory. In order to command the robot from PC, they must be connected to the same router.
On the end-effector of the robotic arm, a RaspberryPi board controls a microcamera, that takes pictures feeding Visual Processing Navigation algorithms. The camera receives commands by the PC of the GNC Laboratory by the connection to the router.

- **Tool reference frame:** By specifying the location and orientation of robot's Tool Central Point (TCP) with respect to its base, commands may be referred directly to the TCP reference frame. The breadboard mounting face serves as the origin of the tool coordinate system, which is a Cartesian coordinate system with the z-axis perpendicular to the breadboard and the x-axis oriented toward the electrical interface block on the breadboard's reverse.

- **Base reference frame:**The z-axis is pointed toward the lab ceiling, the x-axis is oriented to the opposite side of the cable plugs at the back of the robot, while the origin of the Cartesian coordinate system is located at the center of the mounting face (the base) of the robot.

A Raspberry Pi 3 camera Module 3 has been mounted on the TCP of the robotic arm to replicate the on-board optical sensors of the real spacecraft, in order to feed back the HIL facility. To enhance a more precise scenario simulation, the laboratory facility is outfitted with professional studio lighting that replicates solar light (a spotlight is used to mimic Sun light) and Earth Albedo (replicated with diffuse light panels), to reproduce the illumination conditions to which the ISTV would be subject on-orbit.

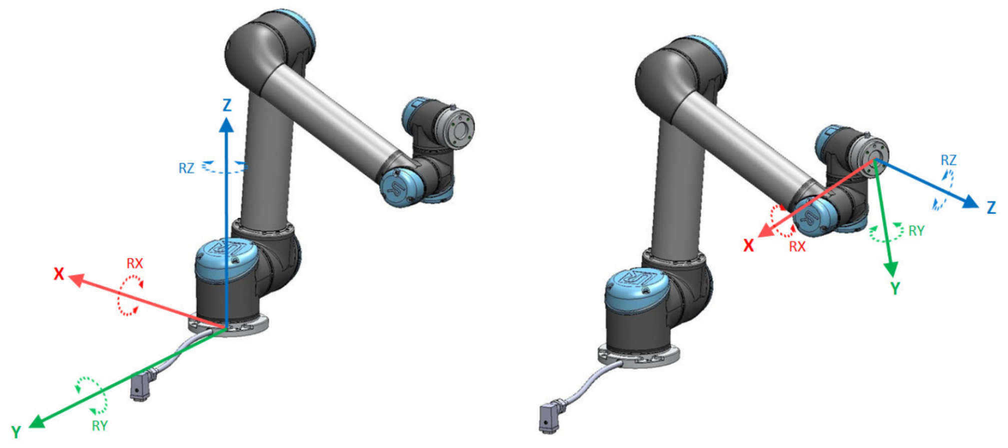


Figure 1.9: UR3e basis refrence frame and TCP reference frame, picture modified from [4]

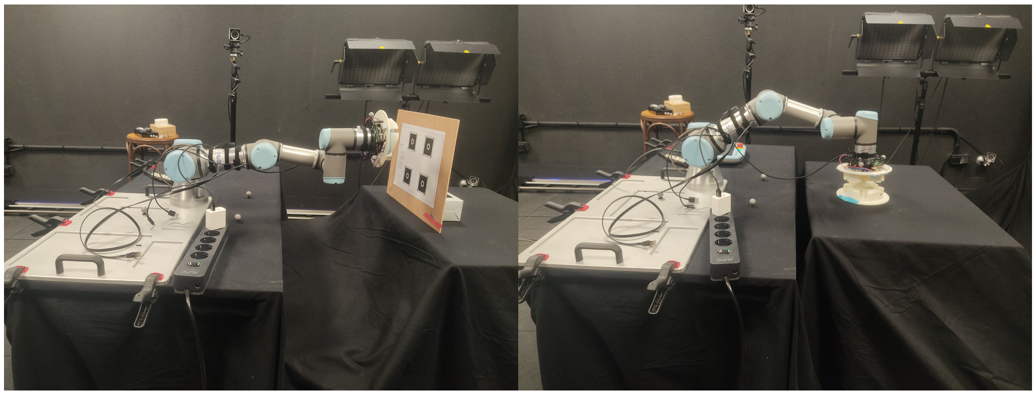


Figure 1.10: SIROM coupling interfaces installed on the robotic arm

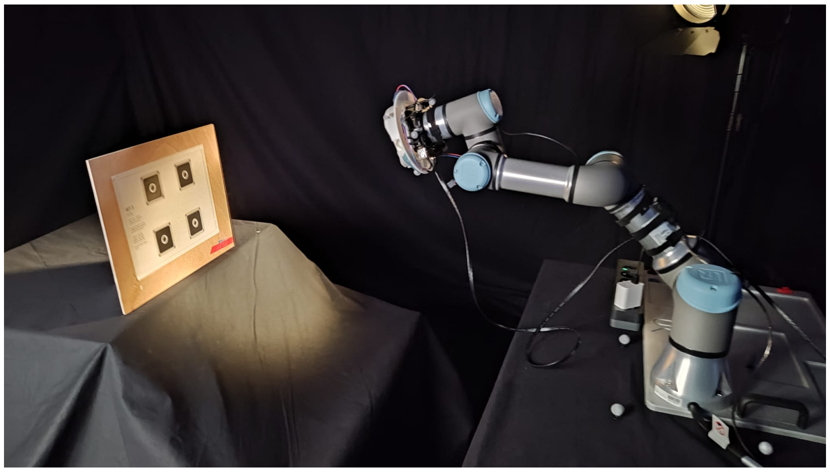


Figure 1.11: TCP pointing four markers on a wooden table, whose middle point is identified as the target of the manoeuvre

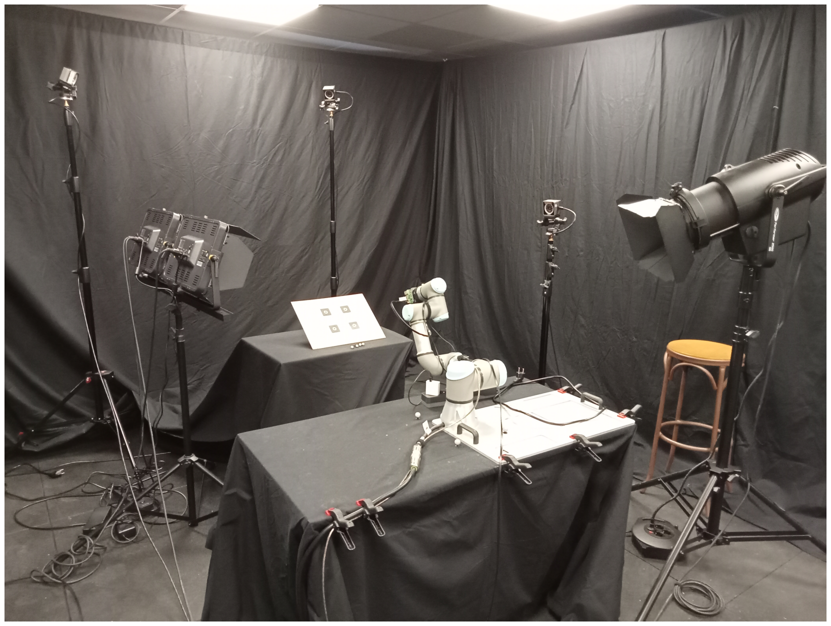


Figure 1.12: GNC laboratory set-up

Chapter 2

State of art

The following chapter provides an overview of the fundamental theoretical notions exploited to investigate GNC systems and to optimize rendezvous guidance algorithms. In the first part, an overview of the main concepts regarding standardized reference systems commonly adopted in GNC design is presented, together with the dynamic models most commonly adopted; following, the focus is put on optimization, with special concern for convex optimization algorithms, regarding their capabilities and limitations. In conclusion, a comparison has been made between the algorithms at the basis of the solvers adopted to optimize the rendezvous trajectories, to provide a complete overview of the state-of-art in the field of convex optimization.

2.1 Problem overview: the two-body problem

The term *orbital dynamics* refers to the description of the translational and rotational motion of an object traveling in space, taking into account both the perturbing effects of the space environment and the gravitational pull of the celestial bodies. In the two-body problem (2BP) formulation, two masses are taken into account, with the smaller mass being negligible in comparison to the bigger one (primary attractor). Due to the preminent role of gravitational perturbation when it comes to the dynamics of two bodies in space, the Newton's law of universal gravitation constitutes the basis of the description of the 2BP, when applied to a general distribution of N point masses. It is possible to express the gravitational acceleration acting on the i-th body as:

$$\ddot{\mathbf{r}}_i = -\sum_{j=1, j\neq i}^{N} \frac{Gm_j}{\|\mathbf{r}_{ij}\|^3}\mathbf{r}_{ij} \tag{2.1}$$

where $\ddot{\mathbf{r}}_i$ is the ith mass's acceleration vector with respect to the system's barycenter, G is the gravitational constant, m_j is the group's jth body's mass, and r_{ji} is the ith body's position vector with regard to the jth body. Although it's common to reduce the perturbation effect to the summation of the mutual attraction of two close bodies, the above Newton's inverse square law predicts that the attraction decreases based on the reciprocal of the square of the increasing distance of the ith body from the jth point mass. Therefore:

$$\ddot{\mathbf{r}}_1 = -\frac{Gm_2}{\|\mathbf{r}_{21}\|^3}\mathbf{r}_{21} \tag{2.2}$$

$$\ddot{\mathbf{r}}_2 = -\frac{Gm_1}{\|\mathbf{r}_{12}\|^3}\mathbf{r}_{12} \tag{2.3}$$

The motion of the object of interest happens often far from all other bodies in the cluster and close to a main attractor, therefore it is common to attribute the perturbation effects in this situation to the mutual attraction of the two closest bodies, leading to:

$$m_1\ddot{\mathbf{r}}_1 + m_2\ddot{\mathbf{r}}_2 = 0 \tag{2.4}$$

Given that $\mathbf{r}_{12} = \mathbf{r}_2 - \mathbf{r}_1$ by definition, it follows that the second body's relative acceleration relative to the first body is:

$$\ddot{\mathbf{r}}_2 = \frac{m_1}{m_1 + m_2}\ddot{\mathbf{r}}_{12} \tag{2.5}$$

The relative motion is calculated by substituting Eq. (2.5) to the second equation of the binary system dynamics in Eq. (2.3):

$$\ddot{\mathbf{r}}_{12} = -G\frac{m_1 + m_2}{\|\mathbf{r}_{12}\|^3}\mathbf{r}_{12} \tag{2.6}$$

The motion of the two isolated masses with regard to their shared COM is described by the previous equation. The body with mass m_2, for which the dynamics is defined, is typically much smaller than the attractor, whose mass is regarded as m_1. This allows to define the so called *Restricted Two-Body Problem* (R2BP):

$$\ddot{\mathbf{r}} = -G\frac{m_1}{\|\mathbf{r}\|^3}\mathbf{r} = -\frac{\mu}{\mathbf{r}^3}\mathbf{r} \tag{2.7}$$

where μ is regarded as the *standard gravitational parameter*, $\mathbf{r} = \mathbf{r}_{12}$ and $\mathbf{r} = |\mathbf{r}|$.

Relative orbital dynamics refers to the orbital dynamics that describes the motion of a spacecraft with respect to a free-falling unattractive point in orbit, a situation pretty common when it comes to GNC system design. To describe the relative motion of two space-

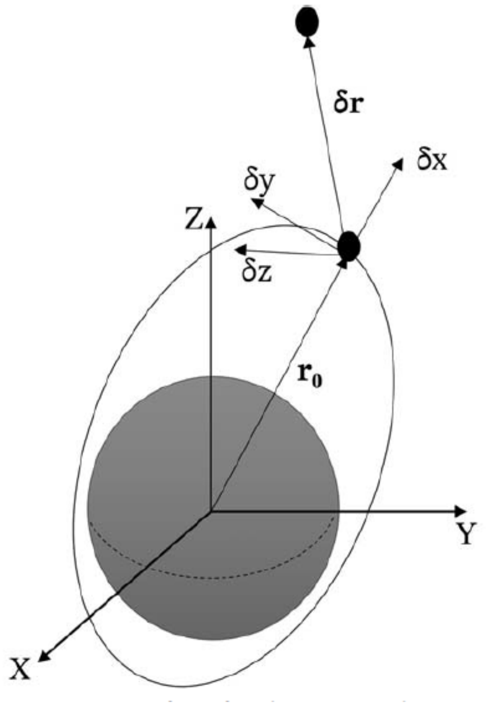


Figure 2.1: Relative motion in description reference frame, picture taken from [42]

crafts, the **Local-Vertical-Local-Horizontal** (LVLH) reference frame $\Delta_{i,j,k}$ is mainly adopted, with the center located in the COM of the target, which is following an orbital

trajectory with angular velocity $\Omega(t)$. The inertial reference frame coincides with the ECI reference frame denoted as $I_{i,j,k}$.
The angular velocity of an orbital point is described by the following relationship:

$$\Omega = \frac{\mathbf{h}}{r_0^2} = \frac{\mathbf{r}_0 \times \mathbf{v}_0}{r_0^2}$$

$$\dot{\Omega} = -2\frac{h}{r_0^3}\dot{\mathbf{r}}_0 = -2\frac{\mathbf{r}_0 \cdot \mathbf{v}_0}{r_0^2}\Omega$$

The inertial position of the spacecraft in the inertial frame can be described as:

$$\mathbf{r}_I = \mathbf{r}_{0,I} + \delta\mathbf{r}_I \tag{2.8}$$

The unit vectors in $\Delta_{i,j,k}$ are referred to as:

$$\hat{\mathbf{i}} = \frac{\mathbf{r}_0}{r_0}, \hat{\mathbf{k}} = \frac{\mathbf{h}}{h}, \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{i}}$$

where $\mathbf{h}$ stands for the specific angular momentum.
The relative position and velocity are translated into the comoving frame as:

$$\delta\mathbf{r}_\Delta = \delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}} + \delta z\hat{\mathbf{k}}$$

$$\delta\mathbf{v}_\Delta = \dot{\delta x}\hat{\mathbf{i}} + \dot{\delta y}\hat{\mathbf{j}} + \dot{\delta z}\hat{\mathbf{k}}$$

Taking into account the characteristic R2BP equation:

$$\ddot{\mathbf{r}}_I = -G\frac{m_1}{\mathbf{r}^3}\mathbf{r}_I = -\frac{\mu}{r^3}\mathbf{r}_I \tag{2.9}$$

The relative position can be expressed as:

$$\delta\ddot{\mathbf{r}}_I = -\ddot{\mathbf{r}}_{0,I} - \frac{\mu}{r^3}(\mathbf{r}_{0,I} + \delta\mathbf{r}_I) \tag{2.10}$$

And the relative acceleration in the comoving frame:

$$\delta\mathbf{a}_\Delta = \ddot{\delta x}\hat{\mathbf{i}} + \ddot{\delta y}\hat{\mathbf{j}} + \ddot{\delta z}\hat{\mathbf{k}}$$

Related to the absolute relative acceleration in the following equation:

$$\delta\ddot{\mathbf{r}_\Delta} = \delta\mathbf{a}_\Delta + \dot{\Omega} \times \delta\mathbf{r}_\Delta + \Omega \times (\Omega \times \delta\mathbf{r}_\Delta) + 2\Omega \times \delta\mathbf{v}_\Delta \tag{2.11}$$

The main contributions in the equation above come from the Euler acceleration $\dot{\Omega} \times \delta\mathbf{r}_\Delta$, centrifugal acceleration $\Omega \times (\Omega \times \delta\mathbf{r}_\Delta)$ and Coriolis acceleration $2\Omega \times \delta\mathbf{v}_\Delta$.
Expressing Eq.(2.10) in the LVLH frame and substituting Eq.(2.11), the relative dynamics in the moving frame is described by:

$$\begin{cases} \delta\ddot{x} - 2\Omega\delta\dot{y} - \dot{\Omega}\delta y - \Omega^2\delta x = -\mu\frac{r_0+\delta x}{[(r_0+\delta x)^2+\delta y^2+\delta z^2]^{\frac{3}{2}}} + \frac{\mu}{r_0^2} \\ \delta\ddot{y} + 2\Omega\delta\dot{x} + \dot{\Omega}\delta x - \Omega^2\delta y = -\mu\frac{\delta y}{[(r_0+\delta x)^2+\delta y^2+\delta z^2]^{\frac{3}{2}}} \\ \delta\ddot{z} = -\mu\frac{\delta z}{[(r_0+\delta x)^2+\delta y^2+\delta z^2]^{\frac{3}{2}}} \end{cases} \tag{2.12}$$

2.1.1 Relative orbital dynamics: HCW equations

There is no analytical way to obtain a solution for the dynamics description presented in the nonlinear equations Eq.(2.12). Analytical solutions are essential for investigating system behaviour when dealing with a large number of numerical simulations, which are a critical feature during GNC design.
The first step to define this problem more clearly is to linearize the system of ordinary differential equations. Close-range relative motion is assumed, so it can be set that:

$$\frac{\delta r}{r_0} \ll 1$$

The aforementioned equation represents the relative speed of two objects in orbit whose distance from one another is significantly less than their distance from the object that is pushing them towards. Exploiting first-order Taylor expansion:

$$\delta \ddot{\mathbf{r}}_I \approx -\frac{\mu}{r_0^3}[\delta \mathbf{r}_I - \frac{3}{r_0^2}(\mathbf{r}_{0,I} \cdot \delta \mathbf{r}_I)\mathbf{r}_0] \tag{2.13}$$

where the terms in $\frac{r}{R}$ of order greater than one have been neglected. Substituting Eq.(2.11) in Eq.(2.12), it's possible to find the set of linear second-order differential equations that describe the comoving frame's relative motion.

$$\begin{cases} \delta\ddot{x} - (\frac{2\mu}{r_0^3} + \frac{h^2}{r_0^4})\delta x + \frac{2(\mathbf{v}_0 \cdot \mathbf{r}_0)h}{r_0^4}\delta y - \frac{2h}{r_0^2}\delta\dot{y} = 0 \\ \delta\ddot{y} + (\frac{\mu}{r_0^3} - \frac{h^2}{r_0^4})\delta y - \frac{2(\mathbf{v}_0 \cdot \mathbf{r}_0)h}{r_0^4}\delta x + \frac{2h}{r_0^2}\delta\dot{x} = 0 \\ \delta\ddot{z} + \frac{\mu}{r_0^3}\delta\dot{z} = 0 \end{cases} \tag{2.14}$$

Some insights into the relative motion in close proximity are offered by the linearized equations of motion:

- The cross-track component is independent from the other two, whereas the in-plane components δx and δy are coupled.
- The vectors r_0 and v_0, which represent the position and speed of the comoving reference frame center in the nonlinear version of the equations of motion, are functions of time

Due to the time dependence of the reference frame position and velocity in the linear differential equations, the linearized equations likewise do not offer a straightforward analytical solution. The current derivation is the broadest one may use with unperturbed relative motion. The **Hill-Clohessy-Wiltshire** (HCW) model instead constitutes a particular case, in which the reference orbit can be assumed to be circular (or nearly circular) and it is one of most frequently adopted reference frames in GNC design. The circularity of the shape of the orbits can be expressed as:

$$e = 0 \rightarrow \mathbf{v}_0 \cdot \mathbf{r}_0 = 0 \rightarrow h = \sqrt{\mu r_0} \rightarrow n = \sqrt{\frac{\mu}{r_0^3}}$$

The moving reference frame rotates around the attracting body with a constant angular velocity referred to as *mean motion n*, describing a circular orbit in which the position and velocity vectors are always orthogonal one to each other. The HCW model and the associated linearized equations of motion for approximately circular orbits are:

$$\begin{cases} \delta\ddot{x} - 3n_2\delta x - 2n\delta\dot{y} = 0 \\ \delta\ddot{y} + 2n\delta\dot{x} = 0 \\ \delta\ddot{z} + n^2\delta z = 0 \end{cases} \tag{2.15}$$

It's possible to determine the position as well as the velocity of the chaser at any time given its initial position and velocity by using the closed-form solutions of these coupled differential equations in matrix form:

$$\delta\vec{r}(t) = [\Phi_{rr}(t)]\delta\vec{r_0} + [\Phi_{rv}(t)]\delta\vec{v_0} \tag{2.16}$$

$$\delta\vec{v}(t) = [\Phi_{vr}(t)]\delta\vec{r_0} + [\Phi_{vv}(t)]\delta\vec{v_0} \tag{2.17}$$

$$\Phi_{rr}(t) = \begin{bmatrix} 4-3\cos(nt) & 0 & 0 \\ 6(\sin(nt)-nt) & 1 & 0 \\ 0 & 0 & \cos(nt) \end{bmatrix}$$

$$\Phi_{rv}(t) = \begin{bmatrix} \frac{1}{n}\sin(nt) & \frac{2}{n}(1-\cos(nt)) & 0 \\ \frac{2}{n}(\cos(nt)-1) & \frac{1}{n}(4\sin(nt)-3nt) & 0 \\ 0 & 0 & \frac{1}{n}\sin(nt) \end{bmatrix}$$

$$\Phi_{vr}(t) = \begin{bmatrix} 3n\sin(nt) & 0 & 0 \\ 6n(\cos(nt)-1) & 0 & 0 \\ 0 & 0 & -n\sin(nt) \end{bmatrix}$$

$$\Phi_{vv}(t) = \begin{bmatrix} \cos(nt) & 2n\sin(nt)) & 0 \\ -2\sin(nt) & 4n\cos(nt)-3 & 0 \\ 0 & 0 & \cos(nt) \end{bmatrix}$$

Given only the final conditions and characteristics of the target vehicle's orbit, it's possible to solve equations (2.16) and (2.17), due to the fact that the matrices involved are invertible.
The stringent assumptions that include limits on eccentricity and on the relative position are among the limitations of such a model. As a result, it is possible to rewrite the system that describes the relative motion of two objects as they travel through space in a linear time-invariant (LTI) state-space form as:

$$\mathbf{x} = (x, y, z, \dot{x}, \dot{y}, \dot{z}) \tag{2.18}$$

If the spacecraft has mass m and a force $F = (F_x, F_y, F_z)$ is applied, the force applied to unit mass $u = \frac{F}{m}$ lead to the definition:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{2.19}$$

where the state matrix A can be written as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}$$

and matrix B:

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The mean motion is defined as:

$$n = \sqrt{\frac{\mu}{a^3}} \tag{2.20}$$

where a is the radius of the target's body circular orbit and and μ is the standard gravitational parameter.

2.2 Optimization

In the following section, the theory behind the formulation of an optimization problem will be presented, in order to give a better understanding of the algorithms adopted in the implementation of the relative translational state problem and the control allocation problem, taking as reference the manual *J.Nocedal, S.J.Wright, Numerical Optimization, Springer (2006).*

In order to set an optimization problem appropriately, an *objective* has to be identified, indicating a numerical measure of the performance of the system under investigation, dependent from specific aspects of the system known as *optimization variables* or unknowns, which are generally constrained. Modeling is the process of determining the objective, variables, and constraints for a specific scenario; once the model has been created, the problem can be solved using an optimization algorithm. Since there is no single optimization algorithm that can be used for all applications, the user evaluates which is the correct algorithm to be adopted, determining how quickly the problem is solved as well as whether it is solved at all.
When determining if the current set of variables is the correct one to use as the solution of an optimization problem, optimality conditions have to be met, that provide helpful insight into how to raise the estimate of the solution from its current state. By using methods like sensitivity analysis, which indicates the sensitivity of the solution to changes in the model and data, the model can be improved.
According to mathematics, optimization is the minimization or maximization of a function when its variables are constrained. The standard notation adopted is:

- x as the vector of the unknown variables;
- f stands for the objective function
- c_i are the constraints of the problem, functions of the optimization variables that define the equalities and inequalities that the unknowns must satisfy.

$$\begin{array}{rll} \min\limits_{x \in R^n} & f(x) & \\ \text{s.t.} & c_i(x) = 0 & i \in E \\ & c_i(x) \geq 0 & i \in I \end{array} \tag{2.21}$$

where I and E stand for the indices for the inequality and equality constraints.

Discrete optimization problems are characterized by the fact that the unknown x is can assume values from a discrete set, such as in the case of *Integer Programming problems*, where the variables can only assume integer values, or cases in which the optimization variables involve permutations of an ordered set, in addition to integers and binary variables.
In contrast, for what concerns *continuous optimization problems*, the elements of x correspond to real values and the feasible set is uncountable infinite. The smoothness of the shape of the functions that model a continuous optimization problem makes it easy to exploit objective and constraint information at a specific point x, to infer information about the function's behavior in all other places close to x, which makes continuous optimization problems typically simpler to solve. In contrast, when dealing with discrete problems, the shape of the objective and the constraints may change a lot moving from one feasible point to another.

Many nonlinear optimization techniques only look for a local solution, or a location where the objective function is less than any other possible location nearby. Instead, the lowest function value among all possible points is the point that is considered to be the global

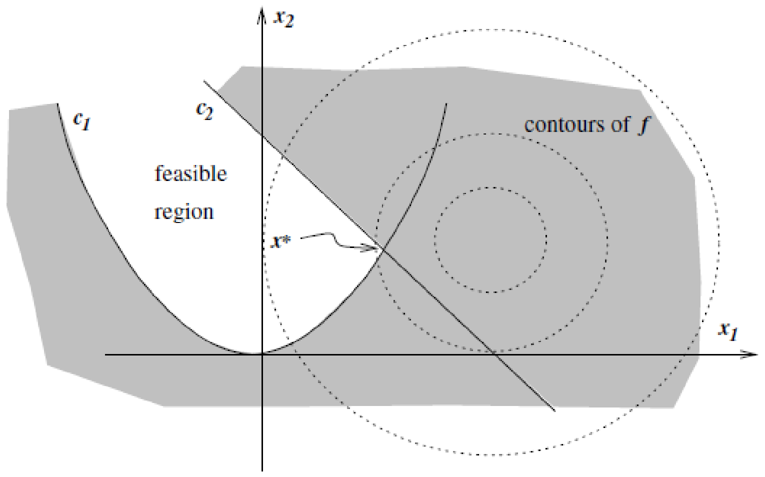


Figure 2.2: Geometrical interpretation of an optimization problem, picture taken from [28]

solution. Programming problems involving *convexity* are characterized by the property that local and global solutions are identical and that theoretical convergence is guaranteed.
Working with global minimizers of the objective functions or locations where the function reaches its minimum value, is typically simpler. An formal mathematical definition is provided below:

Definition 2.2.1 *A point x^* is called global minimizer if $f(x^*) \leq f(x)$ for all x*

Definition 2.2.2 *A point x^* is a local minimizer if there is a region called N around it such that $f(x^*) \leq f(x)$ for all $x \in N$.*

A point that satisfies the requirements above is referred to as a *weak local minimizer*, which is distinguished from the *strict local minimizer* (also known as *strong local minimizer*) when there is a neighborhood of x^* such that $f(x^*) \leq f(x)$ for all $x \in N$ with $x \neq x^*$. When the function f is smooth, there are more practical and efficient ways to find local minima. If f is twice continuously differentiable, a local minimizer (and possibly a strict local minimizer) can be found exploiting only the gradient $\nabla f(x^*)$ and the Hessian $\nabla^2 f(x^*)$.
To analyze the minima of a smooth function, Taylor's theorem has been exploited: suppose that $f : R \rightarrow^n R$ is continuously differentiable and that $p \in N$. Then it's possible to state that:

$$f(x+p) = f(x) + \nabla f(x+tp)^T p \tag{2.22}$$

for some $t \in (0,1)$. Moreover, if f is twice continuously differentiable:

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p\,dt \tag{2.23}$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp)p \tag{2.24}$$

for some $t \in (0,1)$.

Optimality conditions

Assuming x^* to be a local minimizer, it's possible to obtain *necessary conditions* for optimality.

Theorem 1 ***First-order necessary conditions****: If x^* is a local minimizer and f is continuously differentiable in an open neighborhood of x^*, then $\nabla f(x^*) = 0$.*

If the condition $\nabla f(x^*) = 0$ is met, we call x^* a *stationary point*, so it states from Theorem 1 that any local minimizer is regarded as stationary point.

Theorem 2 ***Second-order necessary conditions****: If x^* is a local minimizer of f and $\nabla^2 f$ exists and is continuous in an open neighborhood of x^*, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.*

Sufficient conditions, instead, guarantee that x^* is a local minimizer.

Theorem 3 ***Second-order sufficient conditions****: If $\nabla^2 f$ is assumed to be continuous in an open neighborhood of x^* and $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite, then x^* is a strict local minimizer of f.*

Theorem 3 guarantees a stronger condition than necessary conditions, stating that the minimizer is strict. It's interesting to highlight that if the objective is constituted by a convex function, than local and global minimizers are easily characterized.

2.2.1 Algorithms classification

The following section will provide a set of techniques for unbounded smooth functions optimization. Each method for unconstrained minimization needs the user to indicate a starting point, which is typically referred to as x_0, representing an accurate approximation of the solution.
Starting from x_0, optimization algorithms generate a sequence of iterations x_k that stops when either no further progress is possible or it appears that a solution has been approximated accurately enough, discovering a new iteration x_{k+1} that has a lower function value than x_k. The main strategies adopted to iterate are the *line search method* and the *trust region method.* In the first case, the algorithm selects a direction p_k and iterates from the current iteration x_k to a new iteration with a lower function value along this direction. In order to determine how much it is convenient to proceed in direction p_k, a minimization problem can be solved:

$$\min_{\alpha \geq 0} \quad f(x_k + \alpha p) \tag{2.25}$$

Solving (2.26) may be computationally heavy and is typically not necessary. The line search method generates a finite number of trial step lengths, until it finds one that roughly approximates the minimum of (2.26); thus the process starts again with a new search direction and step length, computed at the new starting point.
In the case of trust region methods, instead, a model function m_k that approximates f is created, whose behavior near the current point x_k is comparable to the one of the actual objective function. However, the model m_k might not be a suitable estimate of f when x is far from x_k. In order to obtain the candidate step direction p, an optimization problem can be solved:

$$\min_{p} \quad m_k(x_k + p) \tag{2.26}$$

where $x_k + p$ is part of the trust region.
If the potential solution does not result in a significant reduction in f, the trust region is

too wide and needs to be shrinked, while the problem (2.27) gets solved again. The trust region has often the shape of a ball defined by $\|p\|_2 \leq \Delta$,with radius Δ. On the other side, a quadratic function with the structure below is used to define the model m_k:

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B p \tag{2.27}$$

with f_k, ∇f_k and B that are a scalar, a vector and a matrix, defined to approximate f with m_k at the first order.
The sequence in which the line search and trust-region techniques select the direction and distance of the move to the next iterate is essentially where they diverge from one another. In order to find a suitable step length, line search fixes the direction p_k, while the trust-region radius k represents the maximum distance that it's possible to proceed before seeking the direction and step that will result in the greater improvement given the distance limit. If the results of this phase are not satisfactory, the distance measure k is decreased and a new iteration is planned. When it comes to line search methods,

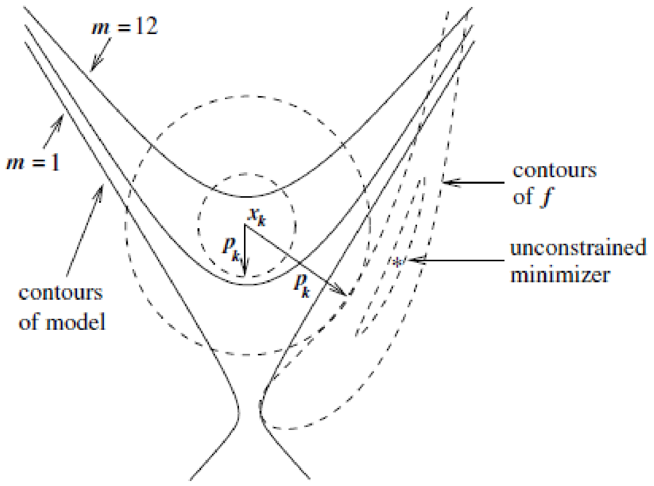


Figure 2.3: Trust regions (circles), corresponding steps p_k and m_k models of the objective function f; picture taken from [28]

the search direction is a crucial choice; the most common one is the steepest descent direction $p_k = -\nabla f_k$, the one that identifies where the function f decrease more quickly. The *steepest descent method* is thus based on the previous concept: at every step, the line search method implemented looks for the p_k along which move forward to. For each direction vector, there is a correspondence with a step length $_k$, that determine how big is the step implemented. However, any descent direction causes a decrease in f, so the steepest descent direction isn't the only convenient one to be implemented in an optimization algorithm. The *Newton direction*, for example, is obtained from the second-order Taylor expansion of $f(x_k + p)$:

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p = m_k(p) \tag{2.28}$$

The Newton direction is obtained finding the vector p that minimizes m_k, assumed that $\nabla^2 f_k$ is positive definite. The following equality holds, setting $m_k(p)$ derivative to zero:

$$p_k{}^N = -(\nabla^2 f_k)^{-1} \nabla f_k \tag{2.29}$$

The approximation $f(x_k + p) \approx m_k(p)$, however, holds when $\|p\|$ is small.
The vast majority of Newton's method line search implementations rely on the unit step

$\alpha = 1$ whenever possible and only modify this value when it does not result in an acceptable decrease in the value of f. Since $(\nabla^2 f_k)^{-1}$ might not exist, the Newton direction might not even be determined when $\nabla^2 f_k$ is not positive definite. Even if it is defined, it might not meet the descent property requirement, in which case it is incorrect to use as a search direction.
The rate of local convergence of Newton direction based techniques is typically quadratic, only a few number of iterations are needed. The necessity of computing the Hessian, on the other side, constitutes the main drawback, because the explicit computation of the second derivatives of this matrix can be laborious and computationally expensive.

2.3 Convexity

A set $S \in R^n$ is said to be *convex* if a straight line connecting any two points in the set lies fully within the set. Formally, $\alpha x + (1 - \alpha y) \in S$ for all $\alpha \in [0, 1]$ for any two points $x \in S$ and $y \in S$.

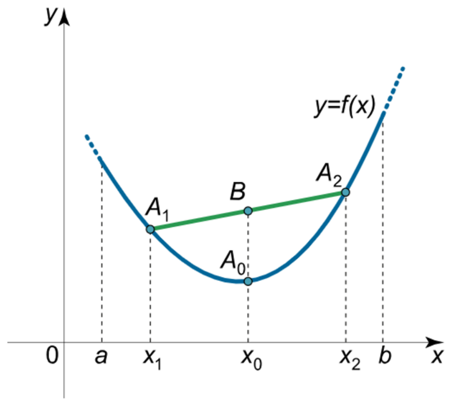


Figure 2.4: Convex function representation, picture modified from [13]

The *convex hull* denoted **conv**C is the set of all the convex combinations of points that lie in C:

$$\mathbf{conv}C = \{\theta_1 x_1 + ... + \theta_k x_k | x_i \in C, \theta_i \geq 0, i = 1, ..., k, \theta_1 + ... + \theta_k = 1\} \tag{2.30}$$

The convex hull is thus always convex and represents the smallest convex set that contains C.
A set C constitutes a *cone* if, for every $x \in C$ and $\theta \geq 0$, it holds that θx; a *convex cone* is a set that presents both the characteristics of convexity and the properties of a cone:

$$\theta_1 x_1 + \theta_2 x_2 \in C \tag{2.31}$$

for each $x_1, x_2 \in C$ and $\theta_1, \theta_2 \geq 0$.
A *polyhedron* is defined as the solution set of linear equalities and inequalities, so the intersection of a finite number of halfspaces and hyperplanes:

$$\mathcal{P} = \{x | {a_j}^T x \neq b_j, j = 1, ..., m, {c_j}^T x = d_j, j = 1, ..., p\} \tag{2.32}$$

That is summed up in compact form as:

$$\mathcal{P} = \{x | A^T x \preceq b, Cx = d\} \tag{2.33}$$

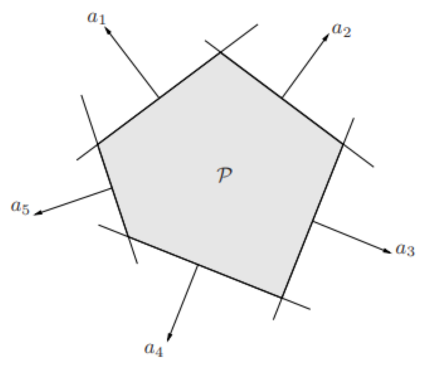


Figure 2.5: An example of polyhedron, as the interction of halfspaces, picture extracted from [2]

A bounded polyhedron is known as *polytope*.
When it comes to functions, f is defined as a *convex function* if its domain S is a convex set and if the following property is true for any two points x and y in S:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \tag{2.34}$$

for all $\alpha \in [0,1]$.

This inequality indicates that the chord from $(x, f(x))$ to $(y, f(y))$ falls above the graph of f (figure 2.5). If strict inequality holds in (2.38) whenever $x \geq y$ and $0 \geq \theta \geq 1$, then a function f is strictly convex. If -f is convex, then f is concave. Since the equality always holds for affine functions in (2.38), all affine functions (and by extension, all linear functions) are both convex and concave. On the other hand, any function that is both concave and convex is affine.

First-order convexity conditions

A differentiable function f is convex if and only if $\mathbf{dom}f$ and

$$f(y)(x) + \nabla f(x)^T(y-x) \tag{2.35}$$

and holds for all $x.y \in \mathbf{dom}f$. The above inequality represents the first-order Taylor expansion of f near x. That means that for a convex function the first-order expansion is a global underestimator of the function. This results leads to the conclusion that any local solution to the optimization problem (2.21) is in fact a global solution if the feasible region and the objective function are both convex, thus the major advantage offered by convex functions is the easiness in identifying local and global minimizers:

Theorem 4 *When f is convex, any local minimizer x^* is also a global minimizer of f. If f is also differentiable, than any stationary point x^* is also a global minimizer.*

Second-order convexity conditions

If f is twice differentiable, that is the Hessian $\nabla^2 f(x)$ exists for every point in $\mathbf{dom}f$, f is convex if and only if $\mathbf{dom}f$ is convex and the Hessian is positiv semidefinite:

$$\nabla^2 f(x) \succeq 0 \tag{2.36}$$

This condition can be traduced geometrically as the requirement that the graph of the function has an upward curvature at x.

Duality

Formulating an optimization problem in the form:

$$\begin{array}{cl} \min\limits_{x} & f_0(x) \\ \text{s.t.} & f_i(x) \leq 0, i = 1, ..., m \\ & h_i(x) = 0, i = 1, ..., p \end{array} \tag{2.37}$$

with optimization variable $x \in R^n$ and without assuming the problem as convex, if the objective function is summed up with weighted constraints functions, it's possible to define the *Lagrangian* $L : R^n \times R^m \times R^p \to R$ as:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x), \tag{2.38}$$

with $\mathbf{dom}L = \mathcal{D} \times R^m \times R^p$ and λ_i referred to as *Lagrangian multiplier* associated with the inequality constraint $f_i(x) \leq 0$, while ν_i are the Lagrangian multipliers associated with the equality constraint $h_i(x) \leq 0$; the multipliers are also called *dual variables*.
The *Lagrangian dual function* or *dual function* $g : R^m \times R^p \to R$ is the minimum of the Lagrangian over x:

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} (f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)) \tag{2.39}$$

The dual function is always concave, even if the problem (2.37) is not convex, due to the fact that the dual function is the pointwise infimum of affine functions of (λ, ν). Another important property of the dual function is that it yields lower bounds on the optimal value p^* for the problem (2.37) if $\lambda \succeq 0$ and ν:

$$g(\lambda, \nu) \leq p^* \tag{2.40}$$

The lower bound results as non-trivial only when $\lambda \succeq 0$ and $(\lambda, \nu) \in \mathbf{dom}g$, i.e. when $g(\lambda, \nu) \geq -\inf$.
The Lagrangian dual function thus provide a lower bounds on the optimal p^*; in order to understand which is the best possible lower bound, the following optimization problem can be formulated:

$$\begin{array}{cl} \max\limits_{x} & g(\lambda, \nu) \\ \text{s.t.} & \lambda \succeq 0 \end{array} \tag{2.41}$$

This problem is regarded as the *Lagrangian dual problem* associated with the problem (2.37), that is known as the *primal problem*; (λ, ν) is also called the *dual optimal* if optimal for the problem (2.41). The Lagrangian dual problem presents the characteristic to be a convex optimization problem, due to the fact that it present concave objective function to be maximized and convex constraints; this conclusion holds even if the primal problem is convex or not.

The best lower bound on p^* that can be derived from the Lagrange dual function is, by definition, the optimal value of the Lagrange dual problem, which is labeled d^*. In particular, it holds that:

$$d^* \leq p^* \tag{2.42}$$

The property above is regarded as *weak duality*. The value $p^* - d^*$ is known as *optimal duality gap*, always nonnegative. If the equality holds, instead, the optimality gap is zero and the *strong duality* holds. Through the exploitation of *constraint qualifications*, it's possible to derive the conditions under which the strong duality holds (if the primal problem is convex usually it holds); a fundamental result in this terms is *Slater's theorem*:

Theorem 5 *If there exist an $x \in \textbf{relint}\mathcal{D}$ such that:*

$$f_i(x) < 0, \forall i = 1, ..., m \tag{2.43}$$

with $Ax = b$ and the problem is convex, strong duality holds.

2.4 Convex optimization: overview

In the following section, a review of some of the main optimization problem classes adopted during the development of this work is presented: least-square problems, linear programming and QP

2.4.1 Least-Square Problems

In data-fitting issues, many models are linear functions of x. Since the residuals $r_j(x)$ in these situations are also linear, the challenge of reducing the residuals is known as a *linear least-squares problem.* For a matrix J and a vector y that are both independent of x, it's possible to rewrite the residual vector as $r(x) = Jx - y$ and to formulate the problem of reducing the value of the magnitude of the residuals as a optimization problem with objective function:

$$f(x) = \frac{1}{2}\|Jx - y\|^2 \tag{2.44}$$

with $y = r(0)$. The point x^* that verify $\nabla f(x^*) = 0$ is the unique global minimizer of f. Thus, the system of equations,

$$J^T J x^* = J^T y \tag{2.45}$$

known as *normal equations*, provide the solution of the least-square problem (2.44); however, this choice can lead to a heavy computational burden, due to the fact that a matrix inversion is involved. Thus alternative solutions are taken into account, such as the *QR factorization* of the matrix J (not recommendable if more information about the sensitivity of the solution to data uncertainties) or the Singular Value Decomposition (SVD) (more recommended in case of call for great robustness). If the problem has a large dimension, an iterative approach may be more recommendable, such as the conjugate gradient method (however, in this work such techniques won't be discussed; the objective was just giving to the reader an overview of the definition of least-square problem and which are the most common solving techniques).

2.4.2 Linear programming

Another class of optimization problems are *linear programming problems* (LP), constituted by a linear objective function and linear constraints (both equalities and inequalities), with as feasible set a polytope, which is a convex, linked set with flat, polygonal faces. Figure (2.6) shows a linear program geometrical interpretation in two dimensions, with dotted lines denoting the outlines of the objective function, with only one vertex that is a singular solution.
The optimal value $c^T x$ might take on the same value over an entire edge if the polytope is rotated, making the solution non-unique.
The standard form of a LP is formulated as:

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & {a_i}^T \leq b_i \quad i = 1, ..., m \end{aligned} \tag{2.46}$$

The *simplex methods* are the most widely adopted optimization algorithms to solve LP, together with their variants such as the *dual simplex methods*; in this work, however, the IPM (the name is due to the fact that the inequality constraints have to be verified

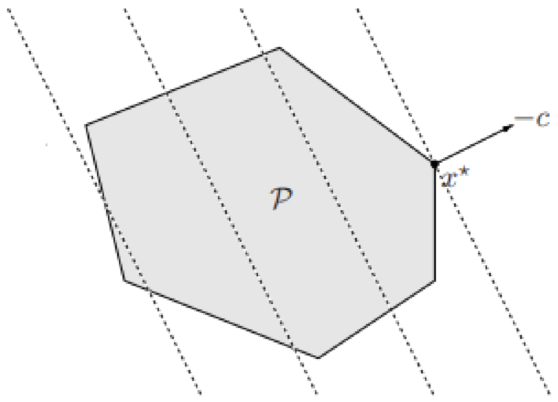


Figure 2.6: Geometric representation of a Linear Programming problem; the dashed lines stand for the objective function representation. The picture is taken from [28]

strictly) are discussed more in details. Interior-point techniques differ from the simplex technique in several ways: the interior-point approach typically requires a smaller number of iterations, whereas the simplex method typically requires a higher number of expensive iterations. In terms of geometry, the simplex approach tests each set of vertices of the feasible set until it determines the best one, working its way around the feasible polytope's perimeter, while IPM procedures get close to the reachable set's boundary only in the limit.

2.4.3 Quadratic optimization problems

The *quadratic programs* (QP) are a class of optimization problems that present convex quadratic objective function and affine constraints (the objective function is minimized over a polyhedron):

$$\begin{aligned} \min_x \quad & \tfrac{1}{2}x^T P x + q^T x + r \\ \text{s.t.} \quad & Gx \preceq h \\ & Ax = b \end{aligned} \tag{2.47}$$

where $P \in {S_+}^n$,$G \in R^{m \times n}$ and $A \in R^{ptimesn}$.
If, on the other side, the inequality constraints are quadratic convex functions too, the problem is denoted as *Quadratic Constrained Quadratic Program* (Quadratic Constrained Quadratic Programming (QCQP)):

$$\begin{aligned} \min_x \quad & \tfrac{1}{2}x^T P_0 x + {q_0}^T x + r_0 \\ \text{s.t.} \quad & \tfrac{1}{2}x^T P_i x + {q_i}^T x + r_i \leq 0 \\ & Ax = b \end{aligned} \tag{2.48}$$

In order to solve the system (2.47), a triangular factorization of the KKT matrix can be performed, or it is possible to implement the *Schur-complement method.* Alternative to the direct factorization presented above, iterative approaches are performed such as the *Conjugate-Gradient method* (CG method) or the *Active-Set Method*, adopted in the development of this work and described in the following section.

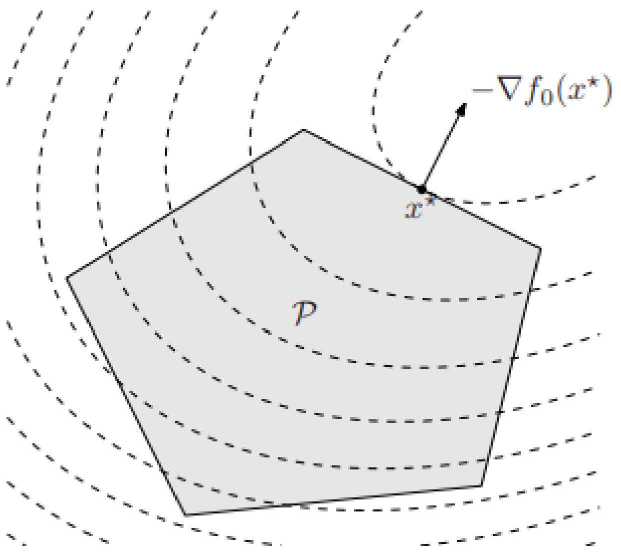


Figure 2.7: Geometric interpretation of a QP; picture taken from [28]

2.5 KKT optimality conditions

2.5.1 Equality-constrained quadratic programs

When discussing QP, the first scenario taken into account regards an optimization problem with only equality requirements. Since some generic QP algorithms necessitate the resolution of an equality-constrained QP at each iteration, techniques for this special situation also apply to problems with inequality constraints.

The equality constrained problem:

$$\begin{array}{ll} \min\limits_{x} & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} & Ax = b \end{array} \tag{2.49}$$

where A stands for the Jacoubian of the constraints with dimension $m \times n$, whose rows are ${a_i}^T$, while $b \in R^m$ is the vector of components b_i.
If x^* is assumed to be a solution of the system (2.48), the same optimization problem can be rewritten in the form below, based on the first-order necessary conditions in Theorem 4:

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \tag{2.50}$$

By rewriting x^* as $x^* = x + p$, where x represents an estimate of the solution and p is the optimization step, the system (2.50) can be expressed in a manner that is helpful for computing:

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix} \tag{2.51}$$

with

$$h = Ax - b, g = c + Gx, p = x^* - x \tag{2.52}$$

The matrix in Eq. (16.5) is known as the *Karush-Kuhn-Tucker* matrix (KKT). Denoting Z as the matrix whose columns are a basis for the null space of A, the following Lemma holds:

Lemma 6 *If A have is a full row rank matrix and the Hessian matrix Z^tGZ is positive definite, then the KKT matrix*

$$K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \tag{2.53}$$

is nonsingular, hence there is a unique vector pair (x^*, λ^*) that satisfies (2.50).
As the following Lemma demonstrates, if the conditions of Lemma (6) are met, x^* not only satisfies second-order sufficient conditions but it follows that x^* is a strict local minimizer of (2.49), so a global solution for the optimization problem.

Theorem 7 *If A have is a full row rank matrix and the Hessian matrix Z^tGZ is positive definite, then vector x^* satisfying (2.50) is a unique global solution of (2.49).*

2.5.2 Inequality-constrained problems

This chapter discusses two techniques for handling inequality and equality constraints in convex quadratic programs: *active-set techniques* and *interior-point techniques.*

In order to formulate an algorithm for the iterative optimization of a problem of the type (2.49), it's crucial to state the correspondent Lagrangian dual function to derive the corresponding KKT conditions:

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^TGx + x^Tc - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i({a_i}^Tx - b_i) \tag{2.54}$$

The active set $\mathcal{A}(x^*) = \{i \in \mathcal{I} \cup \mathcal{E} | {a_i}^Tx = b_i\}$ is composed by the set of the indices of the constraints for which the equalities in (2.49) are verified.
Any solution x^* of the problem (2.49) satisfies the KKT conditions below:

$$\begin{cases} Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} {\lambda_i}^* a_i = 0, & \\ {a_i}^Tx^* = b_i, & \text{for} \quad i \in \mathcal{A}(x^*) \\ {a_i}^Tx^* \geq b_i, & \text{for} \quad i \in \mathcal{IA}(x^*) \\ {\lambda_i}^* \geq 0, & \text{for} \quad i \in \mathcal{I} \cap \mathcal{A}(x^*) \end{cases} \tag{2.55}$$

Assuming that the active set associated with the system above is constituted by linear independent constraints at the solution and that G is positive semidefinite, the conditions (2.58) result to be sufficient.

Active-set methods

There are three types of *active-set* approaches for QP: *primal*, *dual*, and *primal-dual.* The ones presented in this work are only primal approaches, which produce iterations that continuously reduce the objective function $q(x)$ while maintaining feasibility with respect to the primary problem (2.49). By solving a quadratic subproblem in which all of the equality requirements and some of the inequality constraints are imposed as equalities, primal active-set methods can determine the step from one iteration to the next. The *working set* is the subset that is indicated at the kth iteration x_k by $\mathcal{W}_{||}$, with gradients that are linearly dependent.
An iterate x_k minimizes the quadratic q in the subspace denoted by the working set, given an iterate x_k and the working set $\mathcal{W}_{||}$. If not, a step p is calculated by resolving a QP subproblem with equality constraints, where the constraints pertaining to the working set are treated as equalities and all other constraints are momentarily ignored. In order to define this subproblem in terms of the step p,

$$p = x - x_k \tag{2.56}$$

$$g_k = Gx_k + c \tag{2.57}$$

Substituing x into the objective function in (2.49), it's possible to obtain:

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k{}^T p + \rho_k \tag{2.58}$$

with $\rho_k = \frac{1}{2}x_k^T Gx_k + c^T x_k$. This term doesn't depend from p, so it's possible to reformulate the problem as:

$$\begin{aligned} \min_p \quad & \tfrac{1}{2}p^T Gp + g_k^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k \end{aligned} \tag{2.59}$$

with p_k as the solution of this subproblem, that has been calculated exploiting the Schur complement method. It's important to highlight that, since the constraints in $\mathrm{W}_k aresatisfiedin \mathrm{x}_k$, they are also satisfied in $x_k + \alpha_k p_k$, for all α values.
If p_k is nonzero, it's possible to set $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k$is the largest value in the range [0,1] for which all the constraints are satisfied. Due to the fact that the constraints $i \in \mathcal{W}_k$ will be met regardless of the choice of α_k, this information leads to the derivation of an explicit definition of α_k. If $a_i^T p_k \geq 0$ for some $i \notin \mathcal{W}_k$, then $a_i^T(x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$ exists for all $\alpha_k \geq 0$. Therefore, for any nonnegative selections for the step-length parameter, the ith requirement will be satisfied. However, whenever $a_i^T p_k < 0$ for some $i \notin \mathcal{W}_k$, it's possible to state that $a_i^T(x_k + \alpha_k p_k) \geq b_i$ if

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k} \tag{2.60}$$

with α_k that has a value in the range [0, 1] while yet maintaining feasibility, to maximize the drop in q:

$$\min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \quad (1, \tfrac{b_i - a_i^T x_k}{a_i^T p_k}) \tag{2.61}$$

The constraints i for which the minimum in (2.62) is attained are referred to as the *blocking restrictions.* There are no blocking restrictions on this iteration if $\alpha_k = 1$ and no additional constraints are active at $x_k + \alpha_k + p_k$. A new working set $\mathcal{W}_{k+1}$ is generated by adding one of the blocking constraints to $\mathcal{W}_k$ if $\alpha_k < 1$, meaning that the step along p_k was blocked by a constraint not in $\mathcal{W}_k$.
Until a solution $\hat{x}$ that minimizes the quadratic objective function over its current working set is found, the iterations continue, adding constraints to the working set. Such a point can be easily identified because the subproblem (2.61) presents a solution in $p = 0$. If the multipliers corresponding to the inequality constraints that are not in the working set are zero, it follows that $\hat{x}$ and $\hat{\lambda}$ meet the first KKT condition in (2.58). The second and third KKT criteria are now also met, due to the restriction put on the step length.
At this point, the ASM checks the multiplier signs that correspond to the working set's inequality constraints, the indices $i \in \hat{\mathcal{W}} \cap \mathcal{I}$. $\hat{x}$ is a solution for the problem (2.49) if all of the multipliers are nonnegative, as this requirement also satisfies the fourth KKT condition. Having assumed that G is positive semidefinite, therefore it follows that x is a global solution for (2.49). The objective function $q()$ may be reduced by removing one of these limitations if, on the other hand, one or more of the multipliers $\hat{\lambda}_j, j \in \hat{\mathcal{W}} \cup \mathcal{I}$ are negative. In order to solve a new subproblem (2.62) for the current step, the index j is removed, corresponding to one of the negative multipliers from the working set.

Theorem 8 *Suppose that the point $\hat{x}$ satisfies first-order conditions for the equality-constrained subproblem with working set $\hat{\mathcal{W}}$ and assume that the constraint gradients $a_i, i \in \hat{\mathcal{W}}$ are linearly independent, and that there is an index $j \in \hat{\mathcal{W}}$ such that $\hat{\lambda}_j < 0$. Let p be the solution obtained by dropping the constraint j and solving the following subproblem:*

$$\begin{aligned} \min_p \quad & \tfrac{1}{2}p^T Gp + (G\hat{x} + c)^T p \\ s.t. \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k, \quad i \neq j \end{aligned} \tag{2.62}$$

Then p is a feasible direction for constraint j, that is, $a_j^T p \geq 0$. Moreover, if p satisfies second order sufficient conditions for (2.62), then we have that $a_j^T p > 0$, and that p is a descent direction for $q()$.

Generally, the index corresponding to the constraint j of the greatest negative multiplier is the one removed from the working set; this choice is justified by the fact that the amount of decrease of the objective function is proportional, in case of constraint removal, to the magnitude of the correspondent Langrangian multiplier.
The last result presented in this chapter shows that, whenerver p_k from (2.62) is nonzero and verifies the second-order sufficient optimality conditions, than it is a direction of strict descent for $q()$.

Theorem 9 *Supposed that the solution p_k of (2.62) is nonzero and satisfies the second-order sufficient conditions for optimality, then the function $q()$ is strictly decreasing along the direction p_k.*

The formal specification for the ASM implemented in the development of this work follows the one presented by *J. Nocedal, S. J. Wright, Numerical Optimization (Springer, 2009)*: The strength of the ASM resides in the possibility to exploit a warm start that improves

Algorithm 1 Active-set solver implementation, taken from [28]

```
1: procedure ACTIVE SET METHOD($inputs$)
2:     Compute a feasible starting point $x_0$;
3:     Set $\mathcal{W}$, to be a subset of the active constraints at $x_0$;
4:     for $k = 0, ..., 2$ do
5:         Solve (2.62) to find $p_k$;
6:         if $p_k = 0$ then
7:             if $\hat{\lambda}_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$ then
8:                 Stop with solution $x^* = x_k$
9:                 $xk + 1 \leftarrow x_k$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \backslash \{j\}$
10:            else
11:                $j \leftarrow$ arg $min_{i \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_i$
12:        else
13:            Compute $\alpha_k$ from (2.64)
14:            $xk + 1 \leftarrow x_k + \alpha_k + p_k$;
15:        if there are blocking constraints then
16:            Obtain $\mathcal{W}_k + 1$ by adding one of the blocking constraints to $\mathcal{W}_k$ ;
17:        else
18:            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
19:    return $outputs$
```

significantly the convergence speed, thus the choice of an initial feasible point is crucial for the efficiency of the algorithm.

Interior-point methods

Convex quadratic programs can be solved using the IPM by simply extending linear programming algorithms. In this section, the focus is on convex quadratic programs with inequality constraints, in the form:

$$\begin{aligned} \min_x \quad & q(x) = \tfrac{1}{2}x^T G x + x^T c \\ \text{s.t.} \quad & Ax \leq b \\ & 0 \leq \eta \end{aligned} \tag{2.63}$$

with G symmetric and positive semidefinite and A and b defined as:

$$\begin{aligned} A = [a_i]_{i \in \mathcal{I}} \\ b = [b_i]_{i \in \mathcal{I}} \end{aligned} \tag{2.64}$$

with $\mathcal{I} = \{1, ..., m\}$.
The approach described covers equality constraints too, simply extending the considerations that will be presented here.
Rewriting the KKT conditions, it's possible to obtain:

$$\begin{gathered} Gx - A^T\lambda + c = 0, \\ Ax - b \geq 0, \\ (Ax - b)_i \lambda_i = 0 \\ \lambda \geq 0 \end{gathered} \tag{2.65}$$

with $i = 1, ..., m$. The above conditions can be rewritten adopting the slack variable $y \geq 0$:

$$\begin{gathered} Gx - A^T\lambda + c = 0, \\ Ax - y - b = 0, \\ y_i \lambda_i = 0 \\ (y, \lambda) \geq 0 \end{gathered} \tag{2.66}$$

with $i = 1, ..., m$. The KKT conditions result as both sufficient and necessary due to the fact that G has been assumed positive semidefinite, so to find the solution of the system (2.67) it's possible to solve system (2.70).
For each iteration with (x, y, λ) that satisfies $(y, \lambda) > 0$, a complementary measure μ is defined by:

$$\mu = \frac{y^T\lambda}{m} \tag{2.67}$$

Primal-dual methods leads to the solutions (x, y) of the system above by applying different Newton's method iterations to the equalities in (2.70) and changing the search directions and step lengths so that the inequalities $(y, \lambda) \geq 0$ are satisfied at each iteration. The nonnegativity criterion causes all the complexities in the design and analysis of interior-point approaches,due to the fact that it could be hardly nonlinear, so to derive the primal-dual IPM the optimality conditions (2.70) are revised in a slightly different manner:

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ \mathcal{Y}\Lambda e - \sigma\mu e \end{bmatrix} = 0 \tag{2.68}$$

where

$$\begin{gathered} X = diag(x_1, ..., x_n) \\ Y = diag(y_1, ..., y_n) \\ e = (1, 1, ..., 1)^T \end{gathered} \tag{2.69}$$

with $\lambda \in R^m$ and $y \in R^n$.
Primal-dual techniques produce iterates (x^k, λ^k, y^k) that verify the constraints the non-negativity condition, i.e., $x_k > 0$ and $y_k > 0$. The word *interior-point* derives from this characteristic. Primal-dual IPM approaches share two fundamental features with the majority of iterative optimization algorithms: a method for figuring out the step and a way to gauge how desirable each point in the search space is. The average value of the pairwise products $x_i y_i$, $i = 1, ..., n$, which are all positive when $x > 0$ and $s > 0$, is called *duality measure* and is defined:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i y_i = \frac{x^T s}{n} \tag{2.70}$$

In order to solve the following system of linear equations, Newton's method models linearly F around the current solution point for each iteration and determines the search direction $(\Delta x, \Delta\lambda, \Delta y)$:

$$J(x,\lambda,y)\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta y\end{bmatrix} = -F(x,\lambda,y) \tag{2.71}$$

with J as the Jacoubian of F.
Newton equations can be rewritten at this point as:

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ Y & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta y\end{bmatrix} = \begin{bmatrix}-r_c\\ -r_b\\ -XYe\end{bmatrix} \tag{2.72}$$

with $r_b = Ax - b$ and $r_c = A^T\lambda + y - c$.
Each iterate is defined as $(x,\lambda,y)+\alpha(\Delta x,\Delta\lambda,\Delta y)$, with α that is generally chosen $\alpha \ll 1$ in order to not violate the nonnegative condition. Often the pure Newton direction (2.76) doesn't allow to progress much in this sense, so less-aggressive Newton directions are implemented, reducing the product x_iy_i to a lower value, not all the way to zero, i.e., $x_iy_i = \sigma\mu$, with μ the duality measure and $\sigma \in [0,1]$ the desired reduction to be achieved at each step, also called the *centering parameter*.

However, the majority of the implementations of interior-point solvers have to manage an unfeasible starting point, so algorithm enhancements are needed to find a feasible solution. A crucial feature is the use of corrector steps, adopted to compensate the effects of the linearization from Newton steps. If the affine scaling Newton step direction is considered:

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ Y & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x^{aff}\\ \Delta\lambda^{aff}\\ \Delta y^{aff}\end{bmatrix} = \begin{bmatrix}-r_c\\ -r_b\\ -XYe\end{bmatrix} \tag{2.73}$$

and a full step is taken in the direction defined as the solution of the system above, it's possible to obtain:

$$(x_i+\Delta x_i^{aff})(y_i+\Delta y_i^{aff}) = x_iy_i + x_i\Delta y_i^{aff} + y_i\Delta x_i^{aff} + \Delta x_i^{aff}\Delta y_i^{aff} = \Delta x_i^{aff}\Delta y_i^{aff} \tag{2.74}$$

The value of the product x_iy_i, $i = 1,...,n$ is $\Delta x_i^{aff}\Delta y_i^{aff}$ instead of ideal zero. To attempt to correct the deviation from the ideal result, a corrector step is introduced:

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ Y & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x^{cor}\\ \Delta\lambda^{cor}\\ \Delta y^{cor}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -\Delta X^{aff}\Delta Y^{aff}e\end{bmatrix} \tag{2.75}$$

The IPM for QP result to be more efficient if the step lengths $alpha^{pri}$ and $alpha^{dual}$ for the primal and dual variables are equal, due to the fact that they lead to the same rate of reduction in the residuals values r_b and r_c.

$$\alpha_\tau^{pri} = max\{\alpha \in (0,1] : y + \alpha\Delta y \geq (1-\tau)y\}; \tag{2.76}$$

$$\alpha_\tau^{dual} = max\{\alpha \in (0,1] : \lambda + \alpha\Delta y \geq (1-\tau)\lambda\}; \tag{2.77}$$

The *Mehrotra predictor-corrector* is the most well-known interior-point approach for convex QP. By setting $\sigma = 0$ in (2.82), first an affine scaling step $(\Delta x^{aff}, \Delta y^{aff}, \Delta\lambda^{aff})$ is computed, thus, by computing a corrector step, the iteration solution is improved and refined. The centering parameter is then calculated and the system below is solved in order to determine the total step:

$$\begin{bmatrix}G & 0 & -A^T\\ A & -I & 0\\ 0 & \Lambda & \mathcal{Y}\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta\lambda\end{bmatrix} = \begin{bmatrix}-r_c\\ -r_b\\ -\Lambda\mathcal{Y}e - \Delta\Lambda_{aff}\Delta\mathcal{Y}_{aff}e + \sigma\mu e\end{bmatrix} \tag{2.78}$$

Algorithm 2 Mehrotra Predictor-corrector Algorithm for QP, taken from [28]

```
1:  procedure INTERIOR POINT METHOD($inputs$)
2:      Compute $(x_0, y_0, \lambda_0)$ with $(y_0, \lambda_0) > 0$;
3:      for $k = 0, ..., 2$ do
4:          Set $(x, y, \lambda) = (x_k, y_k, \lambda_k)$ and solve $(...)$ with $\sigma = 0$ for $(\Delta x_{aff}, \Delta y_{aff}, \Delta\lambda_{aff})$
    ;
5:          Calculate $\mu = \frac{y^T \lambda}{m}$;
6:          Calculate $\hat{\alpha}_{aff} = \max\ \{\alpha \in (0, 1] | (y, \lambda) + \alpha(\Delta y_{aff}, \Delta\lambda_{aff}) \geq 0\}$;
7:          Calculate $\mu_{aff} = \frac{(y+\hat{\alpha}_{aff}\Delta y_{aff})^T(\lambda+\hat{\alpha}_{aff}\Delta\lambda_{aff})}{m}$;
8:          Set centering parameter to $\sigma = (\frac{\mu_{aff}}{\mu})^3$;
9:          Solve $(...)$ for $(\Delta x, \Delta y, \Delta\lambda)$
10:         Choose $\tau_k \in (0, 1)$ and set $\hat{\alpha}$=min $(\alpha^{pri}_{\tau_k}, \alpha^{dual}_{\tau_k})$
11:         Set $(x_{k+1}, y_{k+1}, \lambda_{k+1}) = (x_k, y_k, \lambda_k) + \hat{\alpha}(\Delta x, \Delta y, \Delta\lambda)$
12:     end
```

Chapter 3

GNC design of the relative translational state

In the following chapter, the focus will be put on the formulation of the optimization problem for the trajectory followed by the chaser spacecraft during the rendezvous manoeuvre, in order to control position and velocity of the vehicle for each step. The attitude is assumed to be fully under the control of a PD controller, fixed during the whole duration of the manoeuvre.

3.1 Model Predictive Control

In order to optimize the trajectory for the rendezvous manoeuvre, guidance and control of the translational state have been handled with MPC, formulating a fuel-optimal consumption and path-tracking problem under constraints that account for both control inputs and state. MPC results a suitable choice in the context of embedded implementations, due to the basic idea behind this advanced control scheme: each time that the MPC is called, an optimization problem is solved and the change in time of the optimization variable is forecasted. In this way, it's possible to handle efficiently the physical constraints of the systems and building a control scheme with a certain robustness implicit in, ensuring onboard path tracking of the optimized trajectory also in real-time applications. This means that, in the context of GNC systems, MPC implementations reduce to the minimum the need for ground support during the rendezvous operation, providing a further degree of automatization in the whole guidance and control loop.

MPC control schemes can use a model of the dynamics in the form:

$$x_{i+1} = f_d(x_i, u_i) \tag{3.1}$$

with x_i that represents the system state at time step i, u_i as the current control input and f_d for the discretized model of the dynamics.
For each time step, the MPC selects a sequence of control actions $\{u_k\}_{k=0}^{N-1}$ that optimizes a sequence of states $\{x_k\}_{k=0}^{N}$; in doing so, a forecast of the state over a defined number of time steps belonging to the so called **prediction horizon** N is realized. A discrete-time optimal control problem arise, with the prediction of the state constrained to the dynamics of the system:

$$\begin{aligned} \min_{U(i|i)} \quad & \textstyle\sum_{k=0}^{H_p-1} x^T(i+k|i)Qx(i+k|i) + u^T(i+k|i)Ru(i+k|i)+ \\ & +x^T(i+H_p|i)Sx(i+H_p|i) \end{aligned} \tag{3.2}$$

$$s.t. \quad U(i|i) = [u(i|i), u(i+1|i), ..., u(i+H_p-1|i)]^T$$

$$
\begin{gathered}
x(i+1) = Ax(i) + Bu(i) \\
u(i+k|i) \in U, \quad k = 0,..,H_p - 1 \\
x(i+k|i) \in X, \quad k = 0,..,H_p - 1 \\
x(i+H_p|i) \in X_f \\
u_{min} \leq u(i+k|i) \leq u_{max}, \quad k = 0,...,H_p - 1
\end{gathered}
$$

with matrices Q, R, S that are positive semidefinite. In this formulation, U represents the optimal input sequence, H_p stands for the prediction horizon and $U(i|i)$ represents the current control input. The first two terms of the objective function accounts for the cost of each i stage, while the latter corresponds to the cost of the terminal stage.
Once solved problem (3.2) and obtained the optimal control sequence, only the first control action u_0^* is applied to the system, discarding u_i^* with $i = 1, ..., N-1$, and the optimal control problem (3.2) is solved after the execution of u_0. Recomputing at every iteration the control action robustifies the control problem against deviations of the predicted trajectory from the actual trajectory/modelling errors/external disturbances, and indirectly takes into account what has happened beyond the modeled dynamics.

For each call of the MPC scheme, SOTB optimizes a QP of the form (3.2) adopting an IPM solver. The QP formulation is constrained taking into account the dynamic model of the system through the Clohessy-Wiltshire equations and the objective function is minimized over a *Receding Horizon* (a moving prediction horizon).

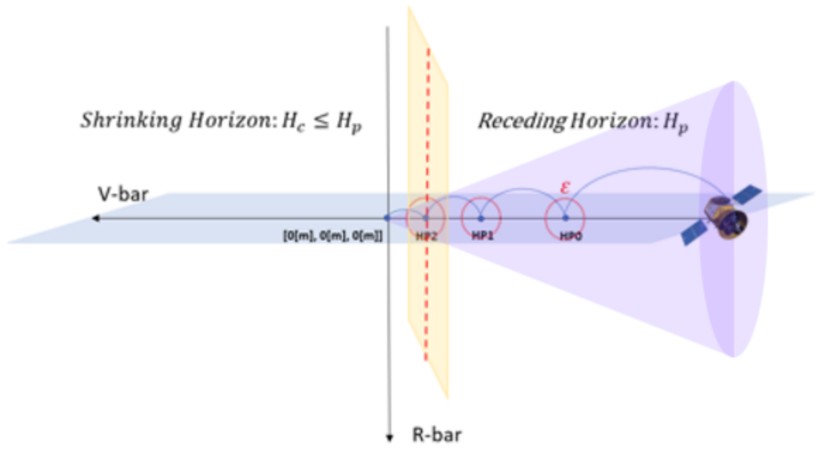


Figure 3.1: Differentiation between the phases of application of Receding Horizon scheme and Shrinking Horizon scheme

Generally, rendezvous manoeuvres cover two different phases:

- When the chaser moves from an holding point to the next one along the V-bar axes of the LVLH reference frame, stabilizing around each holding point before moving to the next one
- When the chaser gets sufficiently close to the target and, exploiting orbital dynamics drift, reaches the target

In the first case, a receding horizon scheme is the most indicated control scheme, to optimize the trajectory from transitioning between holding points, considering only the next fixed prediction horizon. In order to save propellant consume, if the spacecraft remain within a sphere of tolerance around the holding point just exploiting orbital mechanics, control is turned off, while in case of exit from the sphere thrusters are turned on again. When the chaser gets closer to the target, instead, **Shrinking Horizon** schemes allow to improve the prediction capability: the length of the prediction horizon is set as sufficiently large to get to the target and then, for each iteration of the MPC scheme, the horizon length is progressively reduced to the difference between the time step in which the controller is turned on with respect to the time step in which the final state is reached. In this way, it's possible to cut the control horizon as much as the chaser get closer to the target, forecasting the time step in which docking will happen:

$$U(i|i) = [u(i|i), u(i+1|i), ..., u(i+H_c-1|i)]^T, H_c \leq H_p$$

where H_c represents the **control horizon**, during which only the correspondent H_c control actions are optimized to make predictions over H_p.

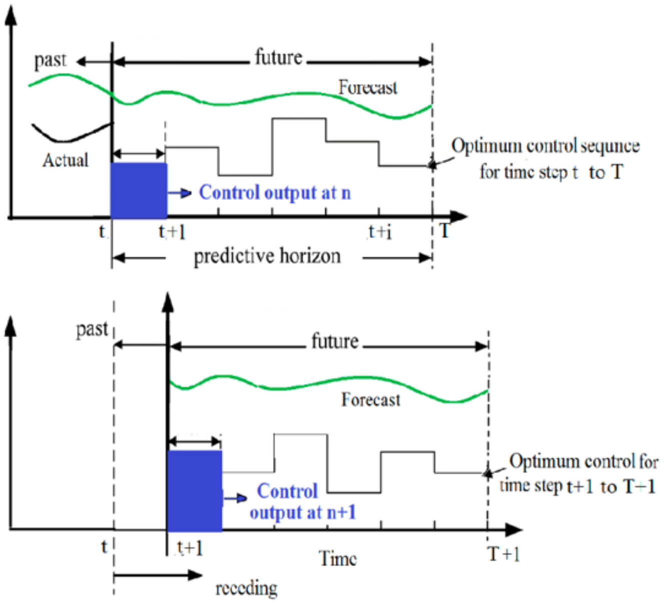


Figure 3.2: Graphic illustration of MPC control scheme, picture taken from [1]

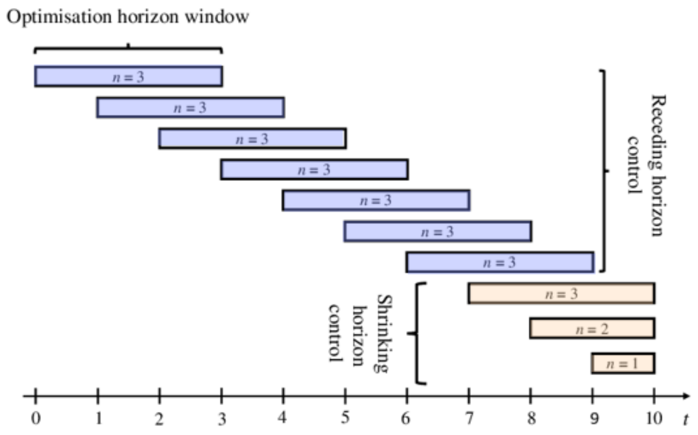


Figure 3.3: Receding Horizon and Shrinking Horizon schemes, picture modified from [29]

In this work, only the docking phase of the manoeuvre has been investigated, regarding a Shrinking Horizon scheme implementation; in order to deal with the high computational burden introduced by the MPC, due to the length of the prediction horizon, the application of a **Move-Blocking** (MB) strategy to the control loop has been investigated as an option, to reduce the number of control variables forcing a control input for multiple discretization steps (the same logic behind a Zero Order Hold filter), but maintaining a first step equal to the inverse of the MPC frequency, so that the complete u_0 computed is applied before executing again the MPC.

$$\Delta u = T\Delta\hat{u} = (T_b \otimes I_{n_u \times n_u})\Delta\hat{u} = \begin{bmatrix} E_0 & & & \\ & E_1 & & \\ & & \ddots & \\ & & & E_{M-1} \end{bmatrix} \begin{bmatrix} \Delta\hat{u}_0 \\ \Delta\hat{u}_1 \\ \vdots \\ \Delta\hat{u}_{M-1} \end{bmatrix}$$

with E_j that are identity matrices of size equal to the number of control inputs for each iteration n_u, and $\Delta\hat{u} = [\Delta\hat{u}_0^T, \Delta\hat{u}_1^T, ..., \Delta\hat{u}_{M-1}^T]$ with $M < N$ that stands for the sequence of the new control input applied for N_j intervals. $I = [I_0, ..., I_M]$ is the vector containing the starting index of each input block, that is computed:

$$I_0 = 0,$$

$$I_j = \sum_{i=0}^{j-1} N_i, \quad j = 1, ..., M-1$$

$$I_j = \sum_{i=0}^{M-1} N_i = N$$

However, during the implementation of the Shrinking Horizon scheme in the control loop in Simulink®, given the maximum time length of the capture manoeuvre, a MB scheme has not been foreseen required, keeping a higher prediction accuracy.

3.2 Translational state problem

The scenario modelled in this work regards the last 2 [m] of the rendezvous manoeuvre, along the V-bar axis of LVLH reference frame that mantains a ciruclar orbit with a suitable attitude; thus the formulation of the problem can be simplified by describing the motion of the spacecraft through the linearized HCW dynamics equations, assuming that the chaser is modeled as a 6-DOF rigid body with constant mass and that is affected only by the non-inertial forces of the relative motion dynamics in the LVLH frame and by the forces generated by the RCS thrusters.
The trajectory spans between two *holding points* laying on the V-bar axes of the LVLH reference frame, respectively located at -2[m] and in the origin. In order to simulate in a more efficient way the algorithms with the robotic test bed, it has been chosen to implement a linear trajectory, on the same orbital plane of the target spacecraft. This choice allows the user to exploit a default function of the Universal Robot model UR3e adopted for the chaser spacecraft: the Tool Central Point (TCP) (Tool Central Point) maintains the default starting orientation during the execution of the trajectory commanded, simulating the fixed attitude of the chaser.

In order to optimize the trajectory, **Linear Model Predictive Control** (MPC) has been implemented, due to its capability to optimize in the current prediction horizon while taking into account the future timeslots, making this advanced control scheme indicated when it comes to automatized on-board guidance. In order solve the problem, a tool developed in-house in SENER has been adopted, the Matlab® SOTB toolbox,

whose strength relies in the possibility for the user to solve large optimization problems just filling a user-friendly interface in Matlab®; once the interface is filled up, the problem is automatically handled by an IPM, with high computational speed already verified on representative hardware, that makes SOTB indicated for embedded on-board applications.
The optimization problem under study has been formulated as follows:

$$\min_{x_i,u_i,\gamma_i} \quad \begin{aligned} &\tfrac{1}{2}(x_N - x_{ref_N})^T P (x_N - x_{ref_N}) + \tfrac{1}{2}\gamma_i^T C \gamma_i + c^T \gamma_i + \\ &+ \textstyle\sum_{i=o}^{N-1} \tfrac{1}{2}(x_i - x_{ref_i})^T L_i (x_i - x_{ref_i}) + \tfrac{1}{2} u_i^T R_i u_i \end{aligned} \tag{3.3}$$

$$s.t. \quad x_{i+1} = f_i(x_i, u_i); \qquad \forall i = 0, ..., N-1 \tag{3.4}$$

$$A_i^x x_i \leq b_i^x + J_i^{affx} \gamma; \qquad \forall i = 0, ..., N \tag{3.5}$$

$$A_i^u u_i \leq b_i^u \gamma; \qquad \forall i = 0, ..., N-1 \tag{3.6}$$

$$\gamma \geq 0 \tag{3.7}$$

$$x_0 = x_{init} \tag{3.8}$$

where N represents the number of time steps needed by the chaser to cover the whole trajectory to the target; x_i stands for the discretized system state for each ith step (according to the fourth order Runge-Kutta method), u_i represents the discrete control input and γ_i are slack variables adopted to relax the constraints.

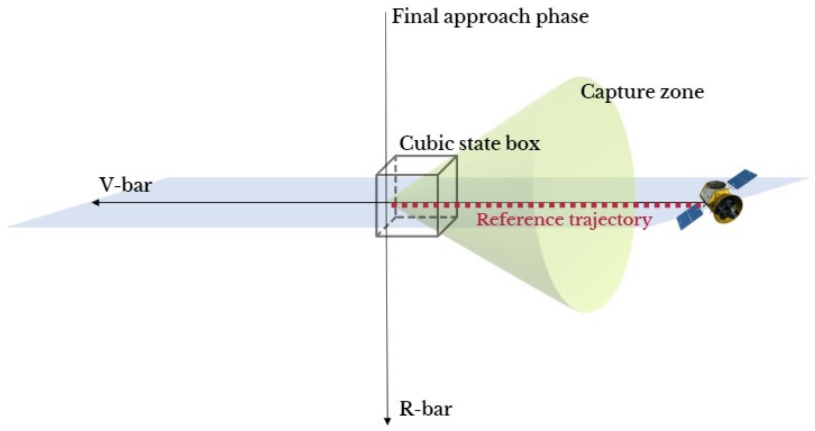


Figure 3.4: Final approach along V-bar axis

The aim of the minimization problem above is to minimize as much as possible the error between the position and the velocity of the chaser in the last step x_N, when reaching the target (that is assumed to be fixed in the origin of the LVLH reference frame), with an associated matrix cost function P. Additionally, the objective includes the minimization the error between the reference linear trajectory and the trajectory followed by the chaser, reaching as fast as possible the V-bar axes, weigthed by matrix L_i, and an homogeneous distribution of the force and torque that the spacecraft has to generate at each time step between the actuators, regulated by matrix cost R_i.
In order to robustify the formulation of the problem, slack variables γ_i have been included to soften the constraints (from Eq. (3.3) to Eq. (3.6)), a common practice in embedded applications; matrix J_i^{affx} selects the slack variable for each time step (in the case of

this work, it deals with just one slack). The slack variable prioritizes the re-entering in the allowed zone in case of violation of the state constraint, when the spacecraft with its maximum capabilities it is not able to avoid a future/current violation of the state constraint, and is associated with a cost matrix C_i, to increase the weight of the cost term of the slacks in case of violation of the constraint, and so the weight of the whole cost function.

$$C = 10^5;$$

$$c = 10^5;$$

In this study, the target's COM is located in a nearly circular orbit and that the distance between the chaser and the target is assumed to be small. Environmental disturbances including solar radiation pressure and air drag are disregarded in orbital motions, thus the Clohessy-Wiltshire (HCW) equation have been exploited to characterize the relative translational dynamics between the two vehicles and propagate the state (Eq. (3.2)).
Recalling the formulation of the relative motion between chaser and target:

$$\begin{cases} \delta\ddot{x} - 3n_2\delta x - 2n\delta\dot{y} = 0 \\ \delta\ddot{y} + 2n\delta\dot{x} = 0 \\ \delta\ddot{z} + n^2\delta z = 0 \end{cases} \tag{3.9}$$

the state dynamics has been translated in terms of state transition matrices, in order to simplify the implementation in the solver:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{3.10}$$

where the state matrix A can be rewritten as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}$$

and matrix B:

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For what concerns the constraints of the problem, a conic approach corridor with the vertex on the V-bar axes has been introduced as a state constraint to maintain the implemented trajectory in a conic envelope, and avoid dangerous deviations from the linear reference trajectory when the chaser gets closer to the target. However, such a constraint would have resulted in an heavy computational burden for the solver, due to the fact that a *Second-Order Cone constraint* would have been introduced; an approximation of the constraint has been adopted, replacing the cone with four tangent planes, each one defined by the point of intersection plane-cone and the normal to the tangent plane in the point of intersection.

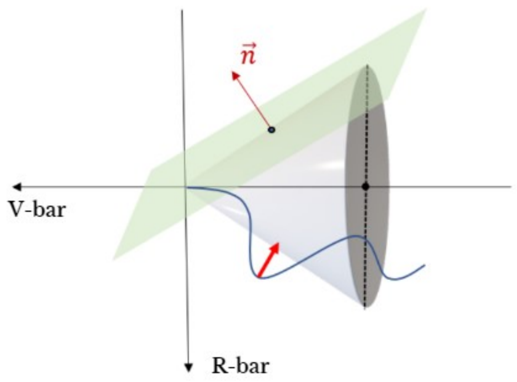


Figure 3.5: Violation of the conic approach corridor

In correspondence of the last step of the rendezvous manoeuvre, an additional state constraint has been introduced as a box located in correspondence of the target: in the volume of this box, thrusters are turned off to save propellant, exploiting only orbital drift to dock the chaser with the target on a flat surface of the target vehicle (the docking manoeuvre require the chaser to approach the target with almost zero velocity). The box has dimensions $30[cm] \times 30[cm] \times 30[cm]$, to safely absorb error in final position due to perturbations/noise; the constraint has been relaxed with slack variables, $\gamma_i \geq 0$, to ensure the feasibility of the problem formulation and the prioritizing of the reentering in the allowed region in case the chaser stops outside the box delimitation. This is a robustness measure in terms of convergence, but that the optimal solution will tend to be inside the box.

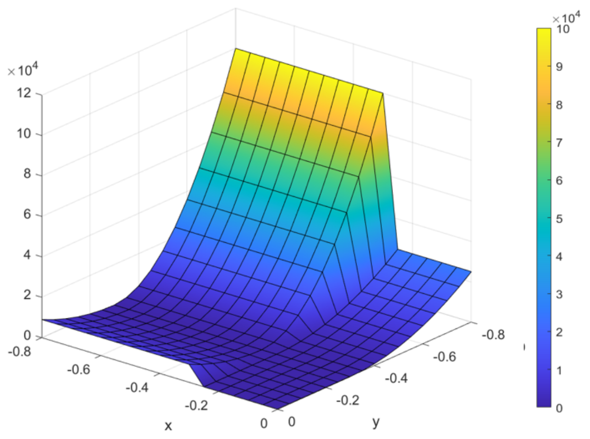


Figure 3.6: Illustration of slack variables weight change in case of violation of the constraints L

When dealing with rendezvous problem, F requested to the actuators to move in a certain time step can't be obtained without taking into account T and viceversa (they are linked by the attitude of the spacecraft). In order to take into account this pairing and not break problem (3.1) formulation when it's not possible to provide a certain value of F given a value of T for the thrusters, a thrust envelope has been built and inserted as constraint (3.4) in the MPC scheme. In the scenario under study, a value of $\|T\| = 0$ has been imposed, since the attitude is modelled as fixed and regulated by a PD controller. The thrust envelope in figure (3.4) has been plotted to represent the feasible magnitude and direction of F to be commanded to the actuators: each point within the surface of the envelope represents the F value that the thrusters have to produce given $\|T\| = 0$, while each point on the surface represents the value of F that is requested to the actuators if isn't possible to provide at the same time $\|T\| = 0$ and a value of F within the surface. In this case, the thrusters must provide the biggest value of F that it's possible to produce according to design specifications, so they will be all turned on at maximum, but some of the actuation capacity will be sacrified in order to produce $\|T\| = 0$.

The thrust envelope derives from a further optimization:

$$\begin{aligned} \min_{f_i} \quad & c^T f_i + (F - A_{eq} f_i)^T Q (F - A_{eq} f_i) \\ s.t. \quad & A_{eq} f_i = T; \\ & A_{\Phi} f_i = F_{\infty}; \\ & MIB \le f_i \le F_{max} \end{aligned} \tag{3.11}$$

with variable f_i that stands for the distribution of the actuation capacity between the thrusters, that is constrained between a minimum value corresponding to the Minimum Impulse Bit that a Pulse Width Modulator is able to provide (more in Chapter 5) and a maximum value $F_{max} = 22[N]$. The objective is to minimize the error between the force generated and the one requested to the thrusters, softening the constraints on F while maintaining the requirement $T = 0$. Matrix Q homogenise the priority given to the softening of the equality constraint $A_{eq} f_i = F_{desired}$ between all the thrusters, with matrix A_{eq} that distributes the force value between the actuators according to their directional cosines $\alpha_i = \frac{F_{xi}}{\|F_{xi}\|}$, $\beta_i = \frac{F_{yi}}{\|F_{xi}\|}$ and $\gamma_i = \frac{F_{zi}}{\|F_{xi}\|}$.
The equality constraint $A_{\Phi} f_i = F_{\infty}$ is expressed as a trigonometric function of the actuation provided by each thruster: if it isn't possible to provide F compatible with $T = 0$, an very big value of F is requested to the actuators (ten times F_{max}). Due to the fact that they aren't able to reach this value, they will turn on at the maximum of their capability. The objective function of problem (3.9) contains also a unitary linear term c that minimizes the thrust value provided by each actuator, in order to turn off the largest number of thrusters:

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

The desired value of F requested to the thrusters corresponds to the first control action given as input to the MPC in the control loop of the robotic arm. However, it has been chosen to approximate the thrust envelope with a cube that lies within the bigger envelope, providing a further degree of conservatism in the F value requested to the actuators, not exploiting their maximum actuation capabilities.

Coming back to problem (3.1) formulation, the values of the entries of the matrix involved in the problem objective function are reported below. Matrix L_i maximizes the number of time steps in which the chaser moves linearly along V-bar axis (q_{x_i} term), the speed of approaching the V-bar axis (q_{y_i}) and the speed in reaching the condition $z = 0$ (q_{z_i}). The last three diagonal entries play a role in the determination of the shape of the manoeuvre, making it smoother ($q_{v_{x_i}}$, $q_{v_{y_i}}$, $q_{v_{z_i}}$). It is defined as stage-dependant,

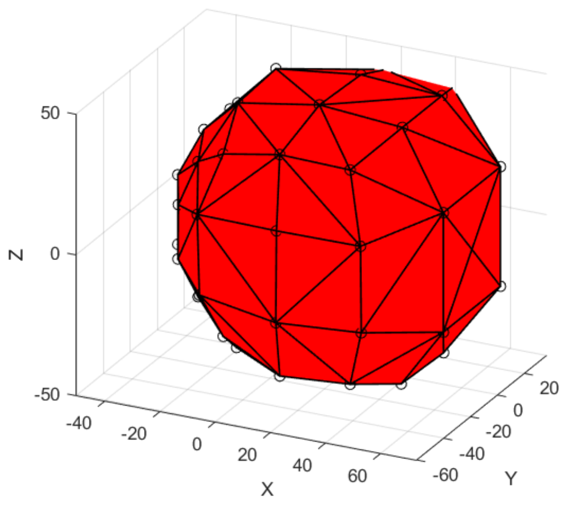


Figure 3.7: Thrust envelope

and in order to reach as fast as possible the reference trajectory (and so the V-bar axis), it defines a sigmoidal shape for trajectory in the first steps and a linear shape in the remaining steps.
In the last stage, in theory $L_N \equiv P$; in reality, matrix P has a higher cost value, to prioritize the minimization of the error in position in the last step of the prediction horizon with respect to the error in velocity.
Finally, control path cost matrix R_i presents a small weight if compared to the other matrices, to prioritize the minimization of path following associated error.

$$P = \begin{bmatrix} 10^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^7 \end{bmatrix}$$

$$L_i = \begin{bmatrix} q_{x_i} & 0 & 0 & 0 & 0 & 0 \\ 0 & q_{y_i} & 0 & 0 & 0 & 0 \\ 0 & 0 & q_{z_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{v_{x_i}} & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{v_{y_i}} & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{v_{z_i}} \end{bmatrix}$$

$$R = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

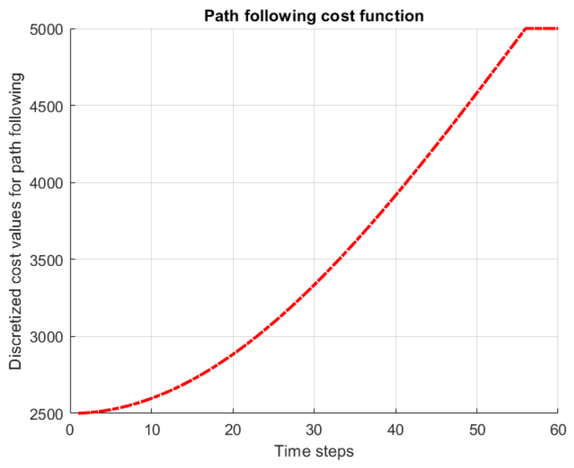


Figure 3.8: Path-following cost function L

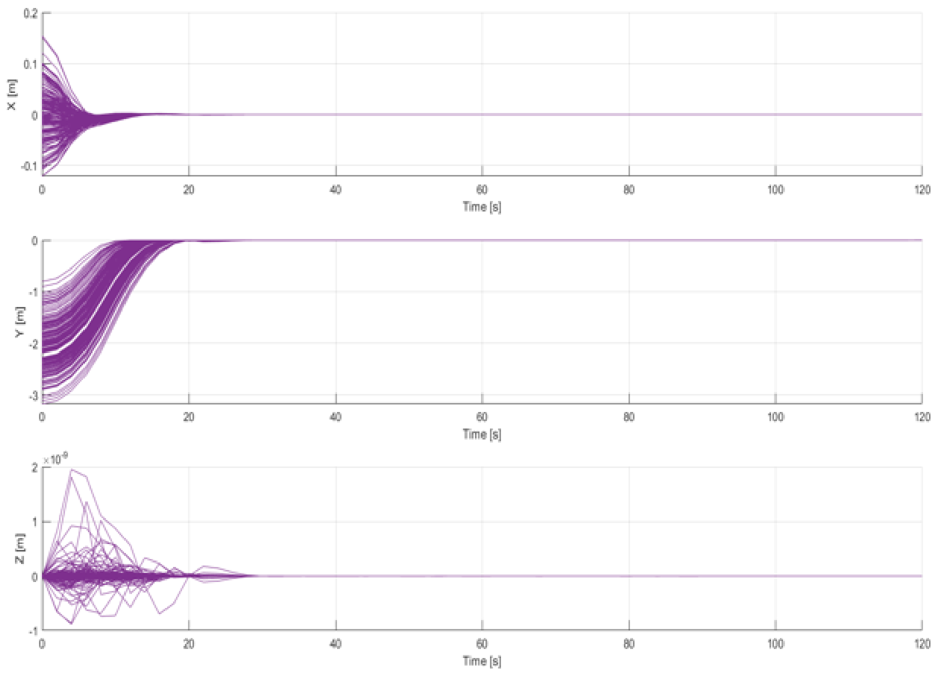


Figure 3.9: Monte Carlo simulation of the offline docking manoeuvre, to assess the prediction capabilities of the Shrinking Horizon scheme implemented when dealing with unstable initial conditions (the initial position deviates from the V-bar axis). The initial position in V-bar has been perturbed according to a normal distribution with $\mu = 2$ and $\sigma = 0.5$, while the initial position in R-bar varies according to a normal distribution with $\mu = 0$ and $\sigma = 0.05$.

Chapter 4

Control Allocation for flight control

Managing control allocation in a spacecraft involves the strategic distribution of control efforts among the actuators, generally redundant with respect to the number of control variables to ensure a further degree of robustness of the control scheme, in case of malfunctioning or failure of one of the actuators. The distributed control actions between the independently controlled actuators have to replicate the whole *virtual* control action that the feedback of the control system delivers.

The main features of the design of a control allocation strategy include:

- **System Modeling**: modeling in a detailed way the spacecraft, including its physical properties, dynamics and environmental factors
- **Objective Definition**: orbit insertion, attitude control, payload deployment or rendezvous with another spacecraft present specific performance requirements that influence the propellant usage
- **Control Actuators**: such as reaction wheels, thrusters, gyroscope that generates the forces and torques needed to control the spacecraft
- **Control Allocation Algorithm**: these tools determine how much control authority should be assigned to each actuator to achieve the desired objectives, taking into account spacecraft's dynamics, constraints and the defined objectives.
- **Optimization and error minimization**: crucial to determine the most efficient and effective distribution of control efforts, includes minimizing fuel consumption, maximizing stability or ensuring redundancy for fault tolerance, taking into account the case in which commanded torque/force is not feasible
- **Autonomous Management of Safety and Redundancy**: redundancy is a main feature of spacecraft control, ensuring that if one control fails, the others can compensate, as in the case of fault detection (FDI) and isolation systems.
- **Testing and Validation**: including testing on ground and during mission simulations to ensure it performs as expected under various scenarios.

This work presents a thrust dispatching strategy for control allocation applied to the capturing phase of a rendezvous manoeuvre, subdividing the workflow in two parts: first the determination of the desired force and torque to be generated to reach the target, then an active set solver has been developed and tested on the allocation module.

4.1 The actuators

The choice of the actuators depends on specific mission requirements, spacecraft size and tasks; many vehicles are equipped with combinations of actuators to provide redundancy and ensure that the spacecraft can work effectively in the harsh space environment.
The most common categories of actuators include *thrusters*, used to generate thrust for orbit changes, trajectory adjustments and attitude control. Based on expelling propellant to produce an action in the opposite direction, thrusters can be chemical, electric or cold-gas type. They include the possibility to design electric propulsion systems, that generate thrust by ionizing propellant and accelerating the ions using electric fields, guaranteeing a high efficiency especially during deep space missions. *Reaction wheels*, instead, are spinning flywheels mounted inside the body of the spacecraft that, by changing their speed, alter angular momentum enabling control of spacecraft's orientation.
On the other hand, *magnetic torquers* interact with Earth's magnetic field, adjusting the orientation of the spacecraft controlling the intensity and the direction of the magnetic field generated; *Control Moment Gyroscopes* are high-speed spinning flywheels that provide both angular momentum and torque for attitude control, allowing rapid changes in spacecraft's orientation.

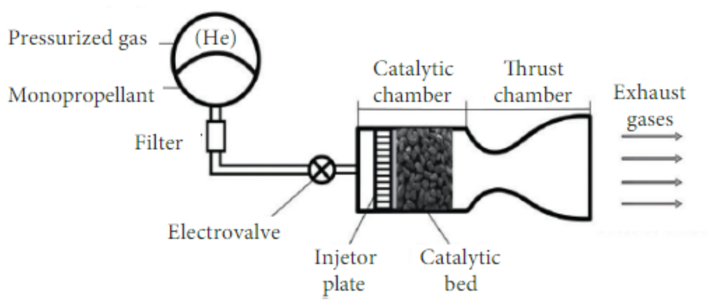


Figure 4.1: Monopropellant thruster, picture taken from [3]

Figure 4.2: Bipropellant thruster, picture taken from [5]

For what concerns the scenario under analysis in this work, it has been assumed that the chaser spacecraft is controlled by cold-gas thrusters, being highly indicated to execute small adjustments requiring control precision. The control allocation proposed in this work focuses on the thrust modulation for the cold-gas thrusters via *Pulse-Width Modulation*, with the objective of minimizing the propellant consumption and minimizing the error between the thrust requested to the thrusters and the thrust actually generated.
Cold gas thrusters (also known as *cold gas propulsion systems*) use the expansion of

an inert pressurized gas as propellant to generate thrust (typically nitrogen or helium), avoiding combustions as conventional monopropellant and bipropellant rocket engines; this leads to less efficiency that the second ones in terms of specific impulse, but allows them to operate at a relative low temperature and to be chemically stable. The gas is stored in high pressure tanks on the spacecraft and, when fired, a valve or regulator releases the compressed gas into a nozzle. As the gas escape the thruster chamber, it expands rapidly generating a high-speed jet of gas. The thrust is generated in the opposite direction with respect to the desired motion, propelling the spacecraft forward.
In order to control thrust in output, Pulse-Width Modulation allows to regulate thrust

Figure 4.3: Cold-gas thruster, picture taken from [10]

level controlling the ratio of time that the thruster is turned on with respect to the time it is off:

$$f_{ij}\Delta t_{ctrl} = f_{nom}\Delta t_{ij}$$

$$\Delta t_{ij} = \frac{f_{ij}}{f_{nom}}\Delta t_{ctrl}$$

where Δt_{ij} indicates how much time the ith thruster keeps turned on at time step j, f_{ij} is value of thrust requested at the ith thruster at time step j, f_{nom} stands for the maximum actuation requested to thruster ith and Δt_{ctrl} is the duration of a guidance time step.
To control thrust, the spacecraft onboard computer adjusts the duty cycle of the PWM signal: if the duty cycle is increased, the thruster is on for a larger portion of each cycle, resulting in a higher thrust. PWM leads to high precision in control due to the small increments that can be obtained by altering the duty cycle and allowing the spacecraft to control the temperature (and so the thermal load) on the thrusters, making PWM well-suited for automatized operation and dynamics adjustments during missions, regulating it digitally by the spacecraft's onboard computer.

4.2 Control allocation by thrust dispatching

In this section it will be analyzed how to optimize the distribution of the available thrust among multiple thrusters to achieve precise control of the chaser spacecraft's position. Once determined the required thrust value and direction, control module dispatches commands to the individual thrusters (the control actions to be provided by the thrusters during each pulse of the optimized trajectory are obtained as output from the translational state problem, imposing the force envelope as constraint).
Thrust dispatching optimization techniques not only achieve fine adjustments in the chaser's position and velocity, but also handle efficiently redundancies. In spacecraft's design, the vehicle is often provided with redundant thrusters to ensure continued operation in the case of failures or faults; thrust dispatching is well-suited to handle and compensate the asymmetrical forces or disturbances that could generate in this case,

maintaining stability of the whole spacecraft and mission flexibility. Furthermore, it is particularly indicated as optimization strategy for the mission under study due to fact that it avoids the overusage of a single thruster, preventing overheating and reducing workload on individual components.

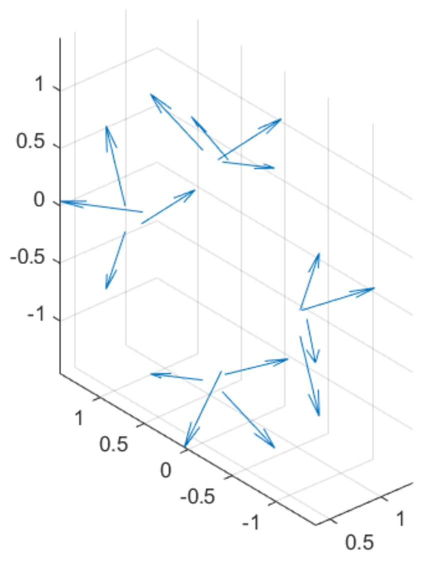


Figure 4.4: Thrusters configuration for the chaser satellite for the study case under analysis

The control allocation problem has been formulated as a QP, assuming a fixed configuration of the thrusters:

$$\min_{f_i} \quad \tfrac{1}{2} f^T R f + \tfrac{1}{2}(F - A_{eq} f_i)^T Q (F - A_{eq} f_i) + \tfrac{1}{2}(T - A_{eq} f_i)^T G (T - A_{eq} f_i) \tag{4.1}$$

$$s.t. \quad A_{\phi} f_i = 0;$$

$$A_{eq} f_i = T$$

$$MIB \leq f_i \leq F_{max}$$

with matrix Q that homogenise the relative weight of the error produced by each thruster in providing the requested force value, matrix G that gives the same priority to the minimization of the error in the norm of the T provided by the thrusters and matrix R that regulates the distribution of controls f_i between the thrusters. The thrust value generated by each actuator lies between a *Minimum Impulse Bit* (MIB) and a maximum value f_{max}; the lower bound is referred to the minimum time interval during which the pulse provided by the thruster recalls the ideal value, as represented in figure (4.5) by the blue stair: The thrust level for each thruster is bounded between a MIB and a maximum value: that means that all the thrusters work in an homogeneous way and none of them will be deactivated. This will for sure worsen the propellant consume, so an $f_{min} = 0$ constraint has to be added to the problem formulation. However, requiring that the thrust generated by each thruster lies between a MIB minimum value and a maximum and requiring at the same time a value of $f_{min} = 0$ constitutes a non-convex constraint for the problem formulation, so it has been relaxed assuming $0 \leq f_{min} \leq f_{max}$. The error

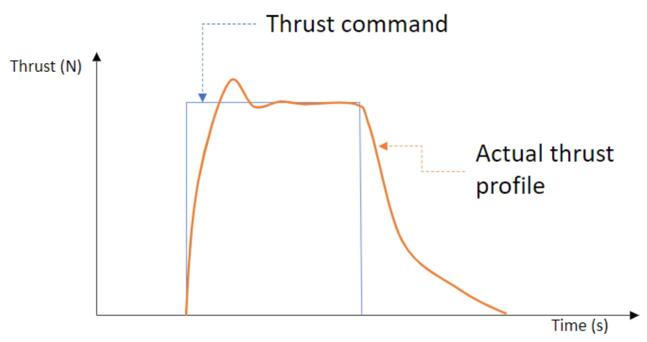


Figure 4.5: PWM of the control input: ideal and actual action, picture taken by [8]

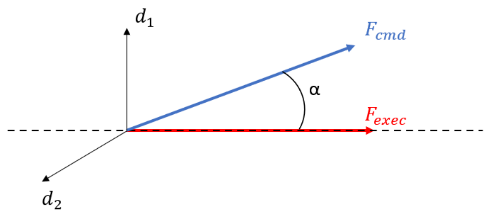


Figure 4.6: Graphic representation of the softened equality constraint for the force zonotope: F_{cmd} stands for the control input commanded to the thrusters, while F_{exec} is the actual thrust level produced by the actuators

generated by this softened constraint is assumed to be compensated by the PWM action and by the feedback in the control loop.

$$0 \leq f_i \leq f_{nom} \frac{\Delta t_{PWM}}{\Delta t_{ctrl}}$$

In order to make the problem formulation more robust, the force allocation constraint $A_{eq} f_i = F_{cmd}$ has been softened, hard-constraining only the direction of the force produced by the thrusters through the equality constraint $A_\Phi f_i = 0$: in this way, the only error that must be compensated and minimized is the one in the magnitude of the F provided by the thrusters, while the direction is always the one requested, as shown in figure (4.6):

$$A_\Phi f_i = d_j \cdot F_{exec} = d_j \cdot A_{eq} f_i = 0$$

with direction d_j defined as: $d_j := \{d_j | F_{exec} \cdot d_j = 0 \cup \|d_j\|_2 = 1\}$.

4.3 Active-set method application to control allocation

An ASM has been developed to optimize the Control Allocation problem and its performance, compared with the one of SOTB. The solver has been structured as presented in Chapter 2, following the reference of *Nocedal, 2006* with the option for the user to provide to the algorithm a warm starting of the active set by exploiting a pre-processing function; the initial guess of the active set will be coincident with the equality constraints, always active.

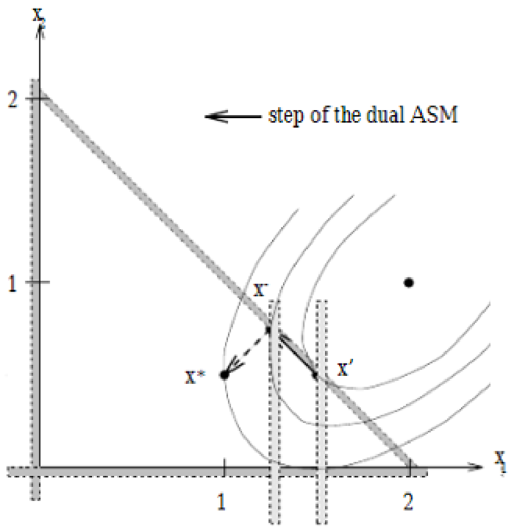


Figure 4.7: Illustration of the ASM algorithm implementation, picture taken from [23]

The control allocation problem has been reformulated in five variants and solved adopting the ASM (results are shown in Chapter 6). In order to avoid ill-conditioning of the Hessian matrix of the QPs and consequently problems in the inversion of the KKT matrix (the computational complexity of ASM is mostly dominated by the computation of the Hessian of the Lagrangian associated), a regularized inverse of the Hessian matrix has been implemented in the ASM, resorting to SVD: in particular, the least significant singular values have been removed and an approximate of the inverse has been built.
SVD allows to express the Hessian matrix H as:

$$H = USV^T$$

with U and V orthogonal matrices (in detail, square matrix whose columns form a orthonormal basis) ans S is a diagonal matrix, whose diagonal entries are called *singular values* s_i and are $s_i \leq 0$. The SVD makes it easy to compute the inverse of the Hessian matrix H:

$$H^{-1} = VS^{-1}U^T$$

If any of the singular values s_i is close or equal to zero, matrix S^{-1} doesn't exist, due to the fact that the diagonal presents terms like $\frac{1}{s_i} = \frac{1}{0}$. However, it is possible to recover some information from the Hessian matrix by selecting only the p higher singular values and the first p left and right singular singular vector to represent the original data (the process is known as *low-rank approximation*):

$$H = \sum_{i=1}^{p} \sigma_i u_i v_i^T = U_p S_p V_p^T$$

with matrices U_D and V_D that contains only the first p columns of U and V, and S_D is the diagonal matrix with only the p highest singular values.

Chapter 5

Model-based driven design

This chapter contains an overview of GNC systems validation and verification (V&V) processes, focusing on **Model-In-The-Loop** (MIL) and **Hardware-In-The-Loop** (HIL). Then a presentation of the simulator developed in Simulink®, Matlab® and Python® to test the algorithms is presented, together with a description of the robotic testbed in the GNC Laboratory of the Universidad Rey Juan Carlos (URJC) of Madrid.

Building and developing systems for applications in space environment is indeed a complicated engineering task, especially when it comes to recreate all the circumstances faced in space prior to the launch of the spacecrafts in-orbit. Due to the unfeasiblity of correcting faults on-orbit, software testing relies on **hardware-in-the-loop** (HIL) as verification method: in the case of this work, the hardware of the satellites has been replicated exploiting robotics facility, reproducing real sensors signals with models and feeding them back to a control loop that simulates the action of actuation.

According to the guidelines of the *European Cooperation for Space Standardization* (ECSS):

- "(A) verification process [.] demonstrates through the provision of objective evidence that the product is designed and produced according to its specifications and the agreed deviations and waivers, and is free of defects. A waiver can arise as an output of the verification process. Verification can be accomplished by one or more of the following methods: analysis (including similarity), test, inspection, review of design."
- "(A) validation process [.] demonstrates that the product is able to accomplish its intended use in the intended operational environment. Verification is a prerequisite for validation."

In the case of GNC systems, V&V is applied by steps, progressively moving from the modeling environment to real-world implementation:

- MIL (Model-in-the-loop) simulation
- SIL (Software-in-the-loop) simulation
- PIL (Processor-in-the-loop) simulation
- HIL (Hardware-in-the-loop) simulation
- In-orbit test (IOT)

The baseline to prototype space GNC software algorithms is to build a Functional Engineering Simulator (FES) in Matlab®/Simulink® via a model-based approach; the models are validated in a **MIL** environment, where the GNC algorithms are tested in closed loop

with modellizations of the mission environment and of the spacecraft subsystem simulation.
In industry, to validate a model simulator means mostly employing Monte Carlo simulations, in which a probability distribution function is linked with significant variables and a fixed degree of errors, actuator/sensor noises, and perturbation is defined.
After the MIL validation phase, the algorithms are translated into embedded-like code, in most cases in C/C++ (**SIL** validation). At this point, code translation is an automated function (autocoding): by following basic guidelines when developing the simulation model, the user can easily retrieve the corresponding embedded-like code.

The **PIL** test is the next step in the V&V process, during which the controller code is incorporated in a dedicated processor, with performance comparable with the one of the final spaceship on-board computer. This test verifies that the synthesized controller can be run on a CPU, albeit with restricted performance in comparison to an on-ground computer.
The **HIL** test is the last on-ground validation phase, in which the real plant and/or sensors/actuators are interfaced with the control loop running on the embedded processor; HIL testings occur in real time, thus processes are executed or simulated in situations as close to reality as possible. As a result, the correct timing execution of all the equipment associated with its necessary software is verified.
For spacecraft applications, it is often difficult to recreate on-ground the identical setting as in space; while some environmental characteristics, such as Sun radiation, can be replicated, others, such as the microgravity condition, cannot be handled in the testings. Thus, sensors and actuators like reaction wheels are tested in an HIL testbed, while spacecraft dynamics are simulated on a real-time platform. The real-time software simulates both the plant and the environment in a functional model of the *On-Board Computer* (OBC) and sends analog commands to the actuators, which are subsequently computed by the control unit processor.

Software simulation is made up of *Dynamics and Kinematics Environment* (DKE) model as well as sensor and actuator models, that are not interfaced with the GNC software at the hardware level. The dynamics and kinematics models represent the spacecraft's behavior in terms of attitude motion and linear motion, while the environment models represent the Earth gravity model, Sun and Moon position propagation, and the action of disturbances such as atmospheric drag, solar pressure, magnetic interference. Because a real-time conversion of the models in software simulation is required, the hardware interface part manages the signal frequency. The software signals of sensor and actuator models are subsequently transformed into physical signals and supplied to the appropriate electrical connector to facilitate a flight-like connection between the OBC and the equipment. In the case of sensors, signals will be delivered from the hardware interface to the OBC, whereas in the case of actuators, command signals from the OBC will be received and used to influence plant dynamics via the hardware interface, allowing the user to take into account aspects not covered till later mission phases.

5.1 Hardware and software set-up

In order to simulate the on-board guidance algorithm with representative sensors in the loop, a robotic test bench has been exploitedthat recreates the space scenario for what concerns illumination conditions, with the support of the GNC Laboratory of the URJC.
In order to simulate the on-board guidance algorithm with representative sensors in the loop, a robotic test bench that recreates the space scenario has been exploited, with the support of the GNC Laboratory of the URJC.
The first step of this process has been to build a control loop of the robotics facility exploiting Simulink®, Matlab® and Python®, to command the optimized guidance to

the robotic arm that represents the chaser dynamics and to receive/handle the information coming from Navigation sensors.

The control loop has been divided into functional blocks:

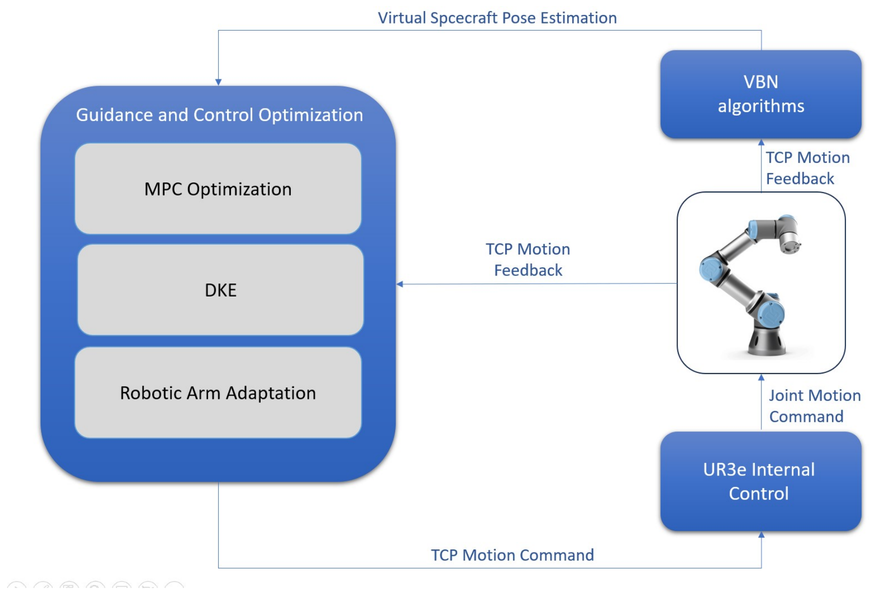


Figure 5.1: Robotic facility control architecture

- **MPC Optimization**: in order to optimize the rendezvous manoeuvre, SOTB has been implemented in the control loop to obtain optimized control inputs exploiting an MPC advanced control strategy. Every time that the solver in SOTB receives a call, the translational state problem presented in Chapter 3 is solved, providing as output to the user the optimized trajectory in terms of position and velocity of the chaser spacecraft $x = [x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3$ and the pulses requested to the thrusters to obtain such trajectory, for the whole prediction horizon. However, the position and the velocity values come from a linearization and discretization of the dynamics, so it has been chosen to propagate in the following blocks the optimized control inputs for each step, in order to maintain an acceptable level of accuracy in the iterations.
  In order to provide an output, SOTB needs an initial condition in terms of optimization variables: in the case of this work, it coincides with the data about the position of the Tool Central Point of the chaser-robot provided by Navigation Visual Processing algorithms.

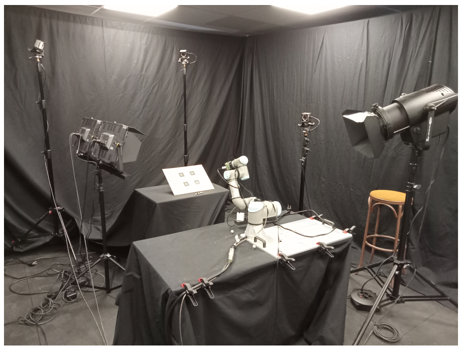
- **Differential Kinematics Equations (DKE)**: The optimal control inputs are used to propagate the DKE presented in Chapter 2 and obtain an optimized trajectory to be commanded to the robotic arm. The first step of the optimized trajectory is taken from the the internal sensors of the robotic arm, in order to replicate the measurement errors coming from the real world.

- **Robotic Arm Adaptation**: In order to send commands to the robotic arms, the optimized position and velocities from SOTB must be scaled and rotated in terms of reference frame. The chaser dynamics is represented by a 6 degree-of-freedom UR3e robot, whose work envelope is a sphere of 50[cm] of radius; a sequential scaling has been applied to replicate a docking manoeuvre of 2[m] in 40[cm], such that the last 0.3[m] of the manoeuvre are in scale 1:1.
  The commanded trajectories are executed by the robotic arm with respect to its

basis reference frame, along positive x-axis and approaching the middle point between four ArUco markers (detected by the visual sensors of Navigation) positioned on a table orthogonal with respect to robot basis reference frame.

- **Communication protocol**: To send the optimized trajectories step by step to the robotic arm, a socket has been opened in Matlab® (version R2020b) to handle the communication between the PC on which the control loop is implemented and the robotic arm. The commands are sent exploiting *URX*, a library developed in Python® (version 3.8.0, to be compatible with Matlab®) by Universal Robots, that allows the user to send commands in terms of linear displacements of the TCP (*movel* function) or to control the angles of the single joints of the robotic arm (*movej* function). Every time that the robot receives a command in terms of linear displacement by Python®, the internal controllers of each joint compute the joint angles that are most suitable to reach such position; however, this way of proceeding don't provide any control on robotic singularities, blocking in a sudden way the whole simulation if reached.
  The socket not only sends commands to the robotic arm but also receives the information coming from the TCP in terms of position and velocity, and throught a broadcaster developed with *URX* (created by the URJC work team), the data are send back to the control loop, as information coming from the internal sensors of the robotic arm.

- **Navigation (VBN algorithms)**: this block includes the Visual Processing algorithms developed by the Navigation area of the work team and integrated in the control loop. The idea is to identify the middle point between four ArUco markers of lateral length 6[mm] taken as target of the manoeuvre, virtually set in the origin of the LVLH reference frame. The markers are positioned at the corners of a flat table, orthogonal with respect to the xy plane of the basis reference frame of the robotic arm; the Navigation algorithms detect the relative distance between the markers and the camera placed in the TCP of the robotic arm, connected to the RaspberryPi board. With this information, the algorithm gets the position of the middle point between the markers and feeds back the information to Guidance algorithms.


(a) Illumination set-up

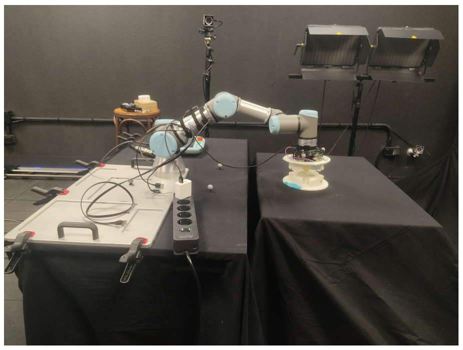

(b) UR3e cobot with the installation of SIROM, SENER coupling interface

Figure 5.2: GNC Laboratory set-up

5.1.1 Kinematic analysis of the UR3e robotic arm

During the execution of the optimized trajectories with the robotic arm, some of the commanded steps of the trajectory led to singular configurations for the joint angles,

due to the fact that the robotic arm is equipped with internal controllers, one for each joint. In order to prevent a sudden stop of the implementation of the trajectories, the control loop has been provided with an additional warning in correspondence of the robot adaptation block: every time that a singularity is detected, the robotic arm comes back to the starting point and executes the remaining part of the optimized trajectory from this point. In this way, the robotic arm avoids damages due to the sudden stops.
To implement the singularity detecting strategy, the kinematics of the UR3e robotic arm has been analyzed: the UR3e is constituted by six rotational joints, each one representing a single degree-of-freedom of motion (the angle of rotation θ_i). Any manipulator presents in general n joints and $n+1$ links, since each joint creates a connection between two links; joints are numbered from 1 to n while links from 0 to n, regarding link 0 as the basis of the robotic arm, always fixed and not moving when the joints are actuated.
[ht] In order to analyze the kinematics of the robotic arm, a reference frame has been

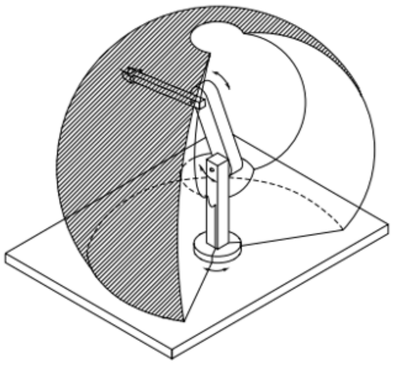


Figure 5.3: Spherical work envelope, picture taken from [39]

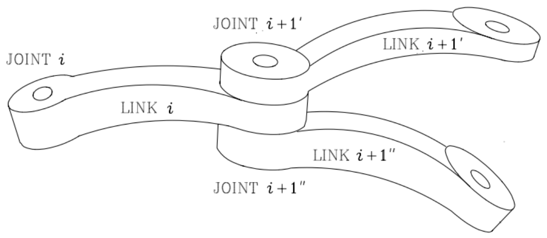


Figure 5.4: Links-joints identification, picture taken from [39]

associated to each link of the robotic arm link; in the case of UR3e cooperative robot, the kinematic chain (the whole structure of the robot comprehensive of joints and links) is **open**, as there is only a sequence of links connecting the ends of the chain. The position of a generic point P can be expressed with respect to the basis reference frame $O_0 - x_0 y_0 z_0$ as:

$$p^0 = o_i^0 + R_i^0 p^i \tag{5.1}$$

taking into account the rotation matrix R_i^0 of the ith frame with respect to the basis frame and vector o_1^0, that describes the origin of frame i with respect to frame 0. The **homogeneous transformation matrix** A_i provides a compact notation of this coordinates transformation:

$$A_i^0 = \begin{bmatrix} R_i^0 & o_1^0 \\ 0^T & 1 \end{bmatrix} \tag{5.2}$$

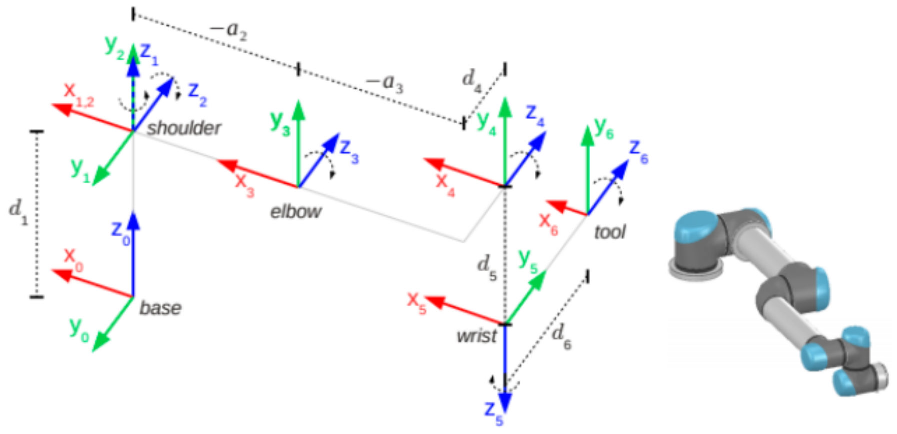


Figure 5.5: Reference frames associated to the joints of the UR3e, picture taken from [9]

A sequence of transformations can be represented as:

$$\tilde{p}^0 = A_1^0 A_2^1 \cdots A_n^{n-1} \tilde{p}^n \tag{5.3}$$

Relating the end-effector position with respect to the joint angles of the robot constitutes the **direct kynematics** analysis of the robotic arm, expressed in a compact notation by the homogeneous transformation matrix:

$$T_e^b(q) = \begin{bmatrix} n_e^b(q) & s_e^b(q) & a_e^b(q) & p_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

with $\mathbf{q}$ the vector of joint variables, the unit vectors labeled $\mathbf{s}$ (the sliding direction of fingers of the gripper), a (approach direction of the gripper with respect to an object) and $\mathbf{n}$ (normal direction with respect to s and $\mathbf{a}$) attached to the end-effector; vector $\mathbf{p}_e$ describes the origin of the end-effector with respect to the basis.

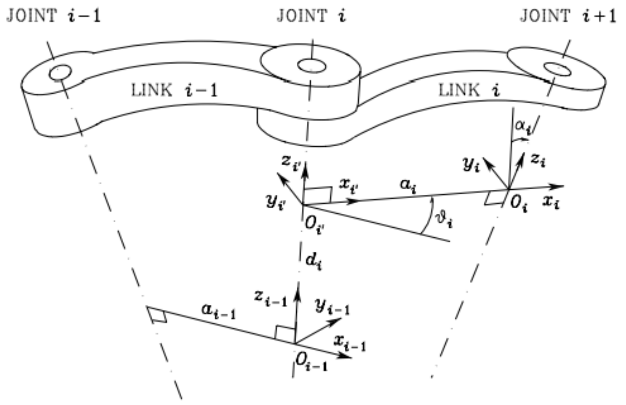


Figure 5.6: Description of the end-effector reference frame, picture taken from [39]

However, matrix A_i can assume a complicated formulation when it comes to express the position and orientation of the reference frame of the end-effector of the robotic arm with respect to the basis; to simplify further computations, it is possible to rewrite the same matrix adopting the **Denavit-Hartenberg** convention.

5.1.2 Denavit-Hartenberg representation

The **Denavit-Hartenberg** convention reformulate the homogeneous transformation matrix A_i as the product of basic transformations. Denoting with i the axis connecting link i-1 to link i, frame i is defined (according to *B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics, Modelling, Planning and Control (Springer, 2000))*:

- Choose axis z_i along the axis of joint $i+1$
- Locate the origin O_i at the intersection of axis z_i with the common normal to axes z_{i1} and z_i. Also, locate O_i at the intersection of the common normal with axis z_{i1}.
- Choose axis x_i along the common normal to axes z_{i1} and z_i with direction from Joint i to Joint $i+1$.
- Choose axis y_i so as to complete a right-handed frame.

In order to describe position and orientation of frame i with respect to frame $i-1$, the convention involves four parameters:

- θ_i is the angle between x_{i-1} and x_i around z_{i-1}.
- α_i is the angle from z_{i-1} to z_i around x_i.
- a_i is the distance between the origin of the *i-1* frame and the origin of the i frame along the x_i direction.
- d_i is the distance from x_{i-1} to x_i along the z_{i-1} direction.

For each link of the robotic arm, three of the four parameters result to be constant, while the fourth one constitutes the joint variable (θ_i for a revolute joint and d_i for a prismatic joint).
In the case of the UR3e cobot, Denavit-Hartenberg parameters are summed up in table (5.1).

The coordinate transformation from basis frame to end-effector frame can be obtained

UR3e				
Kinematics	θ_i[rad]	a[m]	d[m]	α_i[rad]
Joint 1	θ_1	0	0.15185	$\pi/2$
Joint 2	θ_2	-0.24355	0	0
Joint 3	θ_3	-0.2132	0	0
Joint 4	θ_4	0	0.13105	$-\pi/2$
Joint 5	θ_5	0	0.08535	$\pi/2$
Joint 6	θ_6	0	0.0921	0

Table 5.1: Denavit-Hartenberg parameters for the UR3e robotic arm

by post-multiplication of the corresponding homogeneous transformation matrices (for the UR3e robot, all the joints are rotational):

$$A_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -sin(\alpha_i)cos(\theta_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^0(q) = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 \tag{5.5}$$

5.1.3 Singular configurations for the robotic arm

During the implementation of the trajectories with the robotic arm, singularities arise when a position is commanded to the TCP in Cartesian space but the internal controllers of the manipulator encounter problem in the inverse mapping from Cartesian space to joint space. The consequence is that in correspondence of such configurations the manipulator loses a degree-of-freedom, reducing its mobility.
Singular configurations are distinguished between **boundary singularities** and **internal singularities**: in the first case, one of the joints of the robotic arm reaches its full extension, with the robot trying to reach the limit of its work envelope, while in case of internal singularities two axes of the robot get aligned in space. This means that infinite inverse kinematic solutions (determining joint configuration for which the TCP reaches the target) may exist, and that small Cartesian displacement may require infinite joint velocities.
In order to understand how to handle singular configurations, the mapping between the Cartesian space and the joint angles space is analyzed, which is represented in the Jacoubian matrix J. The Jacoubian matrix relates changes in joint parameter velocities to Cartesian velocities, with a number of columns equal to the number of degrees of freedom in joint space, and a number of rows equal to the number of degrees of freedom in Cartesian space.

$$\dot{p}_e = J_P(q)\dot{(q)} \tag{5.6}$$

$$\omega_e = J_O(q)\dot{(q)} \tag{5.7}$$

that are the manipulator **differential equations**.
For the general case, the Jacoubian matrix can be obtained:

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}$$

As a consequence, if Cartesian velocities are specified, the inverse of the Jacoubian leads to joint velocities. When a singularity occurs, however, the determinant of the Jacobian gets close to zero, and its not possible to invert the matrix. In the case of the six revolute

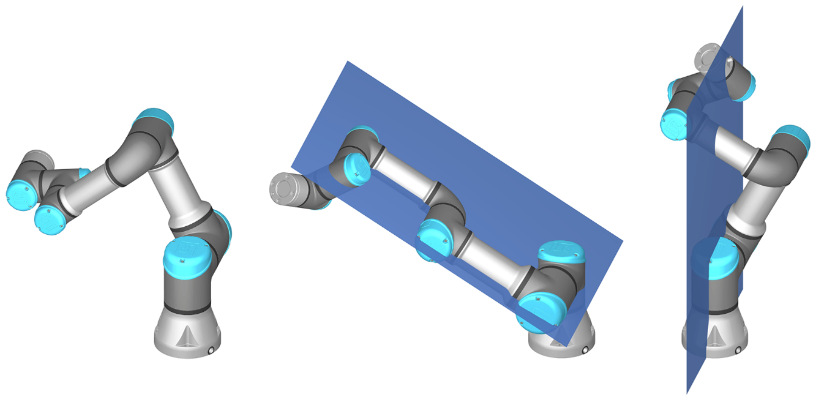


Figure 5.7: Singular configurations of UR3e, picture taken from [36]

joints cobots, the most frequent singularities occur when:

- **Wrist singularities**: it happens when the axes of joints 4 and 6 become parallel; this condition corresponds to $\theta_5 = \deg 0$, $\theta_5 = \pm \deg 180$ or $\theta_5 = \pm \deg 360$.
- **Elbow singularities**: in this case, the singular configuration is reached when the axes of joints 2 and 3 lie on the plane passing through the axes of joints 4, leading to the full extension of the arm; this situation correspond to $\theta_3 = \deg 0$.

- **Shoulder singularities**: it occurs when the intersection point of the axes of joints 5 and 6 lies in the plane passing through the axes of joints 1 and 2.

In order to detect when these configurations occur while commanding the optimized position and velocities to the robotic arm, the control loop has been integrated with the calculation of the Jacoubian corresponding to each time step of the optimized trajectory, to prevent the simulation of the rendezvous scenario by stopping.
In order to compute the contribution of each revolute joint to the linear velocity and to the angular velocity of the end-effector frame, the Jacoubian for the UR3e robot can be obtained:

$$\begin{bmatrix} J_O \end{bmatrix} = \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix}$$

If the Jacoubin degenerates in correspondence of a singularity, the dimension of the space of the end-effector velocities that can be generated by the joints decreases, increasing the dimension of the null space (joint velocities that don't produce movements in the end-effector), according to:

$$dim(\mathcal{R}(J)) + dim(\mathcal{N}(J)) = n$$

In correspondence of a singular configuration, the Jacoubian is no more full rank, meaning that the determinant becomes close to zero.
In the Simulink® control loop for the chaser robot, every time that a optimal trajectory is calculated, the determinant of the Jacoubian associated to the each position commanded to the robotic arm is computed, checking if the corresponding determinant is close to zero. In that case, the robotic arm implements the commanded trajectory starting from the feasible point $x_0 = [0, -2, 0, 0, 0, 0]^T$ in robotic basis coordinates, in order to continue to implement the simulation.

Chapter 6

Model&Results

The following chapter has the double goal of showing the prediction capabilities of the MPC implemented in the control loop of the robotic arm to optimize the rendezvous manoeuvre, in-line with offline computations of the trajectory, and to demonstrate the accuracy and the fast convergence of the ASM solver, applied to CA problem presented in Chapter 4.
The first section of the chapter is dedicated to the comparison between the optimized rendezvous manoeuvre computed offline with SOTB, based on an MPC control scheme powered by a SH strategy, with respect to the results of the MIL testings, that show the capability of the control scheme of prediction of the docking instant when feeded with Visual Based Navigation estimation of the pose. Once assessed the feasiblity of the optimized manoeuvre, the optimization strategy has been applied online to the robotic arm in the GNC Laboratory of the URJC, focusing this time on demonstrating the high level of accuracy that is possible to reach when simulating an orbital manoeuvre through a robotic facility, and the robustness of the optimization scheme in presence of noise and disturbances coming from real sensors.

The second part of the chapter, instead, aims at comparing the optimal solution provided by the ASM designed during the development of this work with the computational capabilities of a professional solver (SOTB), when applied to the optimization of the Control Allocation problem of the chaser spacecraft involved in the rendezvous manoeuvre under analysis; for each one of the test cases, the execution time of the solver has been computed, to evaluate a future implementation of the same solver in the Simulink structure that controls the simulations executed by the robotic facility.

6.1 Robotic arm testings

In order to simulate the optimization of the rendezvous manoeuvre under analysis, the translational state problem presented in Chapter 3 has been first solved offline adopting SOTB. The manoeuvre that has been plant is linear, assuming the attitude of the chaser spacecraft under the full control of a PD, with the chaser that approached the target along the V-bar axis of the LVLH reference frame within the last 2 [m] of the trajectory.

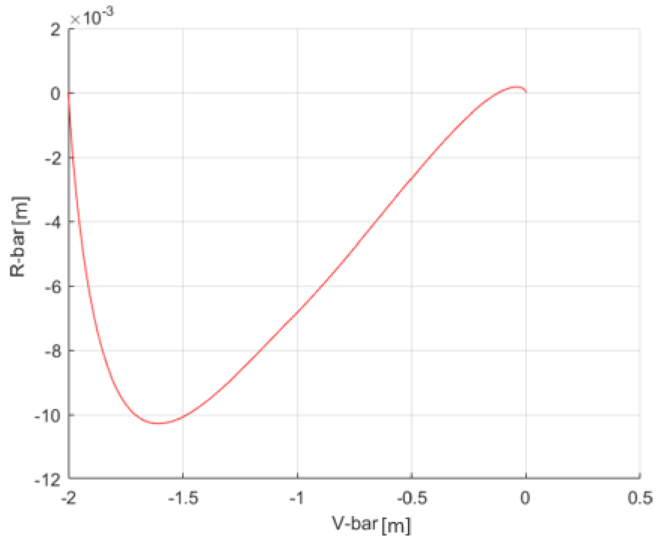


Figure 6.1: SOTB offline trajectory

Figure 6.1 presents an overview of the optimized linear trajectory in the LVLH reference frame, while in figure 6.2, where it's possible to appreciate the speed of the convergence of the formulated optimization problem and the compliance of the trajectory with the requirements expressed in problem (3.3): the manoeuvre results to be linear along the V-bar axis (X and Z optimization variables result to be zero, correspondent respectively to the linear displacement along the R-bar axis and to the linear displacement orthogonal with respect to the orbit plane), approaching the target in only 16 [s] (as it's possible to notice from the Y optimization variable, correspondent to the linear displacement along the V-bar axis). Such speed of convergence is emphasized by the implementation of the SH scheme, that lets the user shrinks the prediction horizon until the docking instant is reached, based on iterative predictions of the docking state.

In order to simulate correctly the optimized rendezvous manoeuvre on hardware, the optimization scheme has been first tested in a MIL framework, developed as a Simulink structure, that recalls the MPC control scheme, computes an optimized control input sequence and propagates the translational state through the HCW dynamic model based only on the first control input. Once the state has been propagated, a Matlab function block simulates the change of reference frame and the sequential scaling strategy, essential to replicate the trajectory of a spacecraft with a robotic arm, whose maximum extension is limited by its work envelope. The Simulink control loop has also been integrated with a Navigation block, to estimate the position of the virtual robotic arm through Visual Based Navigation algorithms; such information is then fed back to the optimization block,

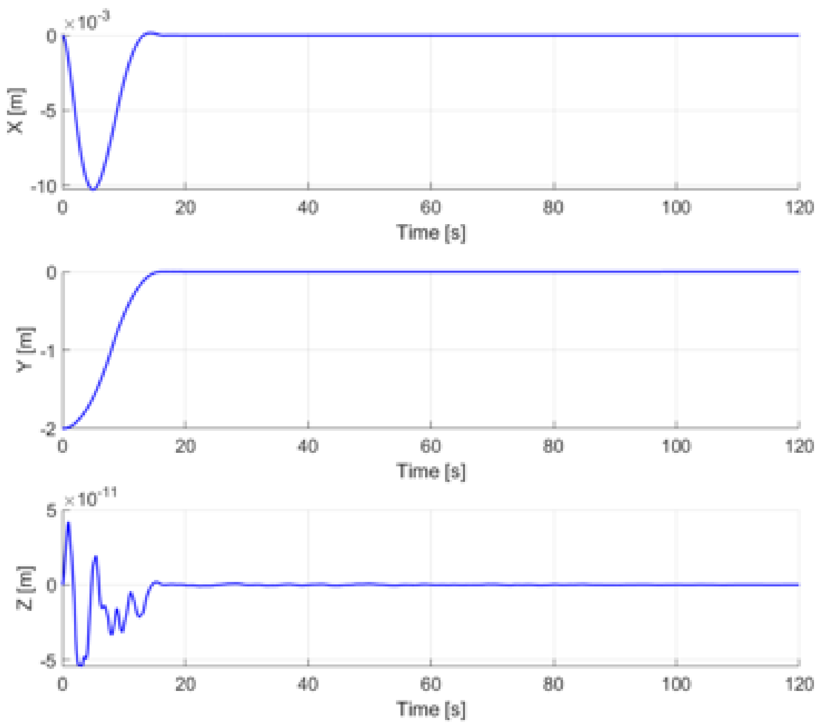


Figure 6.2: Time evolution of the optimization variables for problem (3.1), solved offline by SOTB

to compute a new prediction of the docking state and a new optimized trajectory based on Navigation data. Figure 6.3 presents a comparison between the recomputed optimized trajectory by the MPC after implementing only the first control input of the optimized sequence (blue lines) and the HCW propagation of the translational state (red line), based on the first control input; at the same time, it's possible to appreciate how the integration of the Visul Based Navigation algorithms in the control loop leads to the exact tracking of the trajectory computed by the MPC, that shows robustness against the noise level introduced by the Navigation algorithms during the simulations, modeled as a normal distribution whose parameters are reported in table 6.2.

After the MIL testings, the focus moved to simulations on hardware, replicating the manoeuvre in two different frameworks: first feeding the optimization block with the TCP pose estimation coming from the internal sensors of the robotic arm, assumed as the ideal case, while in a second moment the optimization block has been fed with the data coming from the Visual Navigation algorithms developed by the Navigation area of the work team, reproducing in the GNC Laboratory the same illumination conditions to which the spacecraft is subjected on-orbit and exploiting as sensors a microcamera connected to a RaspberryPi board installed on the TCP of the robotic arm. A socket has been opened in Matlab®/Pyhton to connect with the IP of the UR3e cobot adopted during the testings, and to the real-time port of the robotic arm, to acquire data in real-time concerning the position of the TCP (that simulates the chaser spacecraft); table 6.1 shows the frequency at which each step of the optimized trajectory is commanded to the robotic arm when the socket is opened (port 30003) and the frequency at which the data are collected (port 30004).

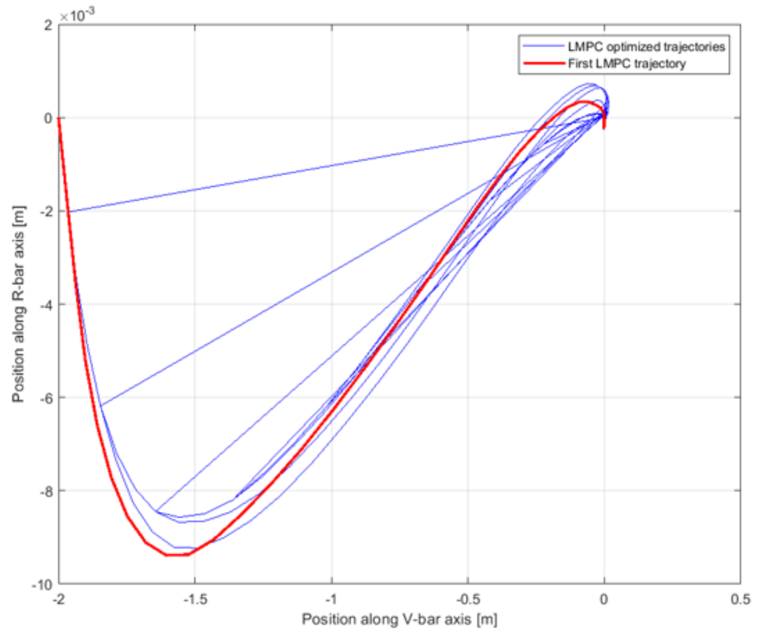


Figure 6.3: Trajectory obtained propagating the DKE equations according to the control input provided by SOTB, executed by the robotic arm and tracked by the Visual Navigation algorithms

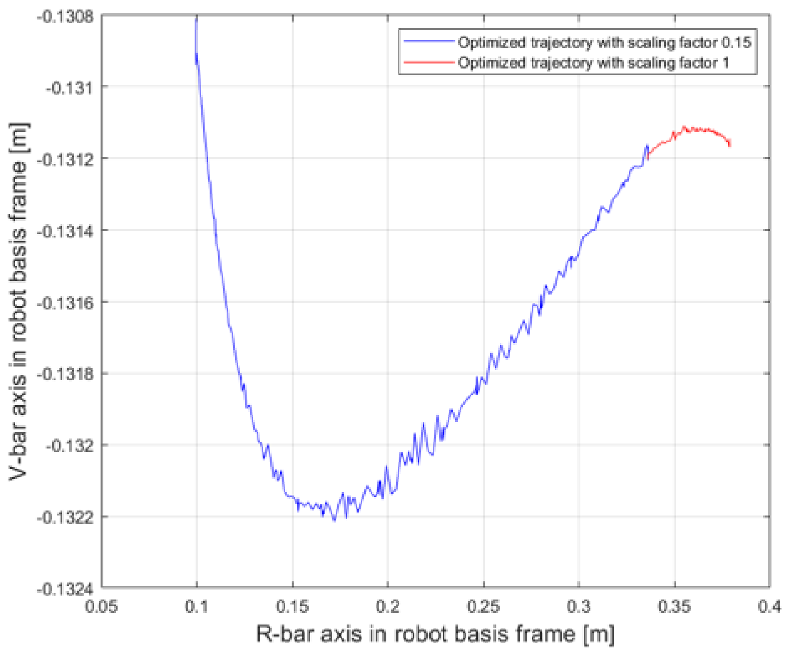


Figure 6.4: Real-time data from the TCP, acquired at a frequency of 500 [Hz].

	Primary		**Secondary**		**Real-time**		**RTDE**
Port no.	30001	30011	30002	30012	30003	30013	30004
Frequency [Hz]	10	10	10	10	500	500	500
Receive	URScript	-	URScript	-	URScript	-	Various data

Table 6.1: Remote control via TCP/IP overview

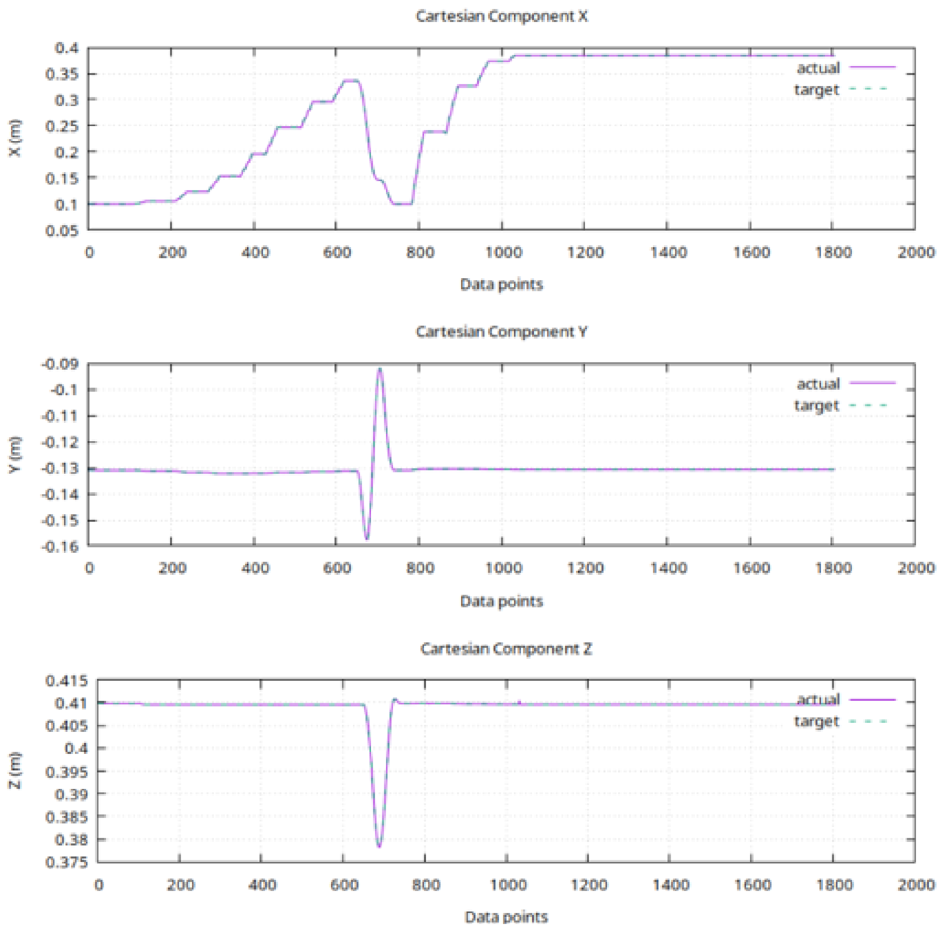


Figure 6.5: Evolution of the trajectory in real-time with respect to the basis reference frame of the robotic arm

Figure 6.4 presents the data acquired in real-time from the TCP of the robotic arm, assuming perfect Navigation, and showing how the MPC control scheme tracks the trajectory implemented offline; the whole 2-meter-long manoeuvre is replicated in 40[cm], compliant with the work envelope of the UR3e robotic arm (50[cm] of radius), applying a scaling factor of 1 (red line, that has been scaled to fit the first part of the executed trajectory) only to the last 30 [cm] of the manoeuvre, while the previous part presents a scaling factor of 0.175.
The optimized trajectory leads to no singular configurations for the joints of the robotic arm, as it's possible to notice by the absence of sudden deviations from the reference trajectory in figure 6.4. In correspondence of a possible violation of the work envelope

of the robot, the manipulator is programmed to reach a feasible point for the trajectory implemented (in the case under analysis, the starting point of the optimized trajectory) and, exploiting the sequential scaling strategy implemented in the control loop, to continue to implement the commanded steps. This recovery strategy effect is shown in figure 6.5, that reports the evolution in time of the optimization variables, according to the real-time data received by the internal sensors of the robotic arm; in correspondence of the change of the scaling factor, the robotic arm returns to its initial position (X equal to 0.1[cm] and Y equal to -0.13[cm], referred to the basis reference frame of the UR3e), then it continues to implement the optimized trajectory until the docking instant.

After the testing of a perfect Navigation scenario, the optimized guidance algorithms have been tested assuming as initial condition the pose estimation coming from the Visual Based Navigation algorithms, introducing the error range presented in figure 6.2. In this case, only the last 30[cm] of the 2-meter-long manoeuvre have been replicated, due to the fact that, for the time being, the VBN are not programmed to take into account a scaling of the manoeuvre during simulations.

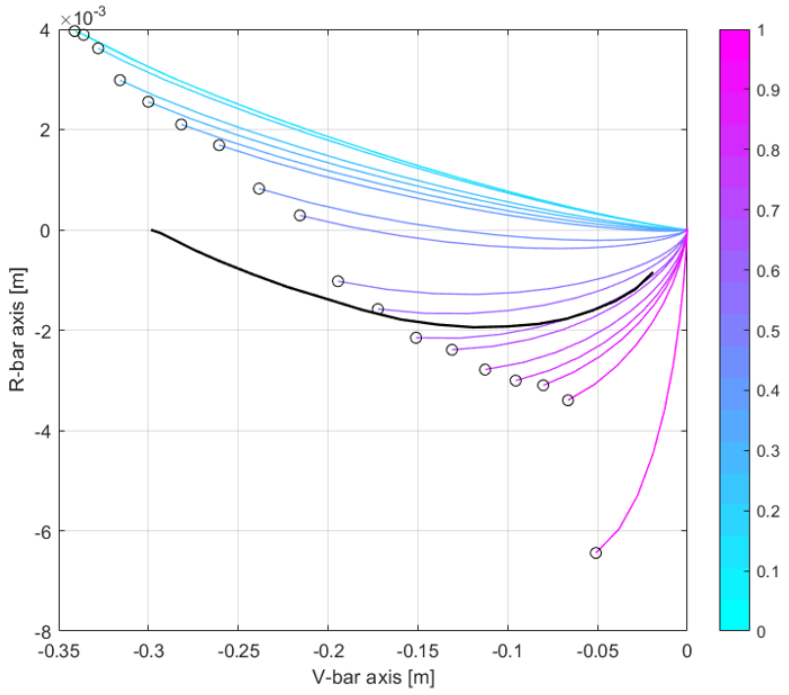


Figure 6.6: Comparison between navigation estimation of the position of the virtual spacecraft on-orbit (black points), DKE propagation of the trajectory (black line) and MPC predictions of the docking state. Only the last 30[cm] of the trajectory have been implemented adopting VBN algorithms integration, to work in scale 1:1.

The plot in figure 6.6 shows that the optimization strategy successfully handles Navigation error, reaching the target positioned at the origin of the virtual LVLH reference frame. The black line tracks the ideal manoeuvre, coming from the propagation of the state based on the HCW equations, assuming as initial condition the pose estimation of the internal sensors of the robotic arm. In contrast, Navigation algorithms introduce a source of error when estimating the correspondent position of the TCP, deviating from the black line, but the optimization block recovers the error and lets the virtual chaser

Estimation error	**1[m]**
Lateral μ	-0.2[mm]
Lateral 3σ	0.5[mm]
Range μ	0[cm]
Range 3σ	0[cm]
Yaw, pitch μ	0°
Yaw, pitch 3σ	4°
Roll μ	0°
Roll 3σ	0.2°

Table 6.2: Pose estimation error of ArUco markers in 3σ deviation; the values in the table have been estimated by the Navigation area of the team work

to dock at the origin of the virtual LVLH reference frame.
The main error source is introduced when measuring the distance between the middle point of the four ArUco markers regarded as the target and the basis of the robotic arm, to translate the optimized manoeuvre from the virtual LVLH frame to the reference frame of the basis of the cobot. In future, such error range will be decreased by adopting an OptiTrack PrimeX 13 capturing system, equipped with cameras for motion tracking of four spherical markers that will be distributed on the TCP of the robotic arm.

6.2 Active Set Method test campaign

In order to optimize fuel consumption on the chaser spacecraft and to reduce the error between the force and torque values commanded to the actuators and the ones actually obtained through thrust dispatching, the Control Allocation problem for the chaser has been optimized as presented in Chapter 4, in the form of a QP. Once formulated, the problem has been adopted to test the computation capabilities of the ASM developed during this thesis, with the aim of integrating a control allocation block in the Simulink structure that controls the robotic arm; however, due to time issues, the solver has been only tested offline.
The CA problem presented in Chapter 4 has been reformulated in five different versions, to compare the thrust dispatching performed by the ASM with the results of SOTB, a professional solver. For each test case, 100 shots of F and 100 shots of T from a random distribution have been fed to the ASM solver, to obtain the f_i control efforts for each one of the 16 cold-gas thrusters with which the chaser is equipped; the values of f_i have been then used to compute the actual F and T values that have been obtained by the actuators, and the relative error has been computed.

Case 1: hard constraints on force and torque values

The first test case is constituted by a reformulation of problem (4.1), with hard constraints on both the F and T values commanded to the actuators.

$$\begin{aligned} \min_{f_i} \quad & \tfrac{1}{2} f^T R f \\ s.t. \quad & A_{eq} f_i = F \\ & A_{eq} f_i = T \end{aligned} \tag{6.1}$$

$$MIB \le f_i \le F_{max}$$

The boxplot in figure 6.7 presents the comparison between the computational time of

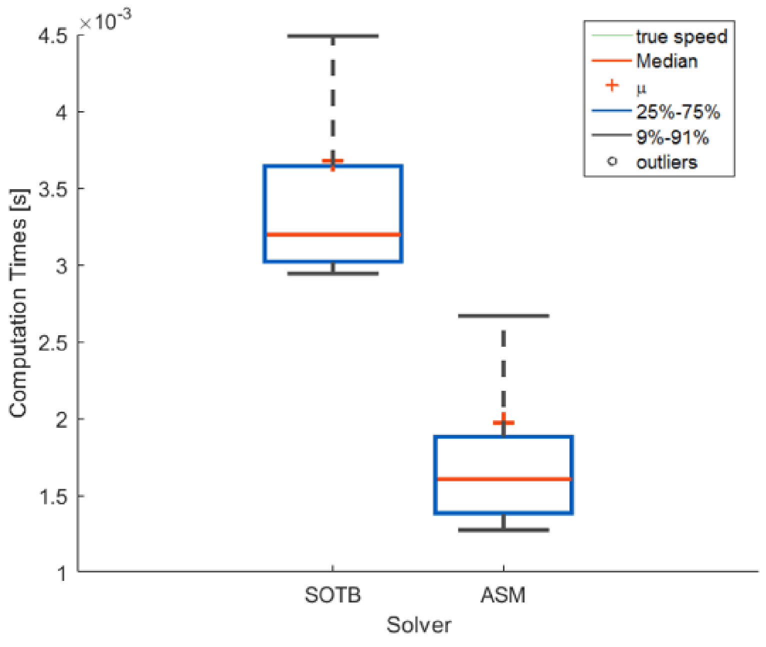


Figure 6.7: Statistics of computation times for the ASM compared with SOTB on an Intel® Core™ i5 processor

the ASM with respect to SOTB, in terms of mean value (red cross) and 3σ (black lines): it's possible to appreciate how the first solver presents a lower computational time with respect to SOTB according to the mean values of each solver (red cross), due to the fact that the computations required by the ASM are less expansive with respect to an IPM (the heavier operation executed by the ASM is the inversion of the KKT matrix associated to the QP problem). In both cases, the execution time is compliant with real-time implementations, making the developed solver suitable for future integrations in the Simulink control loop of the robotic test bed.
Figure 6.8 and 6.9 show that for both the solvers the error in reconstructing the F and T values is very low, due to the presence of the hard constraints on both force and torque. The IPM presents a higher lever of accuracy, as it's possible to notice from the mean value and from the 3σ distribution, due to the lower tolerance level set when defining the convergence criterion (10^{-15} for the IPM, 10^{-10} for the SOTB); this means that the ASM shows almost the same level of accuracy of the IPM but converges in fewer iterations.

Case 2: absence of equality constraints

$$\min_{f_i} \quad \frac{1}{2}f^T R f + \frac{1}{2}(F - A_{eq}f_i)^T Q (F - A_{eq}f_i) + \frac{1}{2}(T - A_{eq}f_i)^T G (T - A_{eq}f_i) \tag{6.2}$$

$$MIB \le f_i \le F_{max}$$

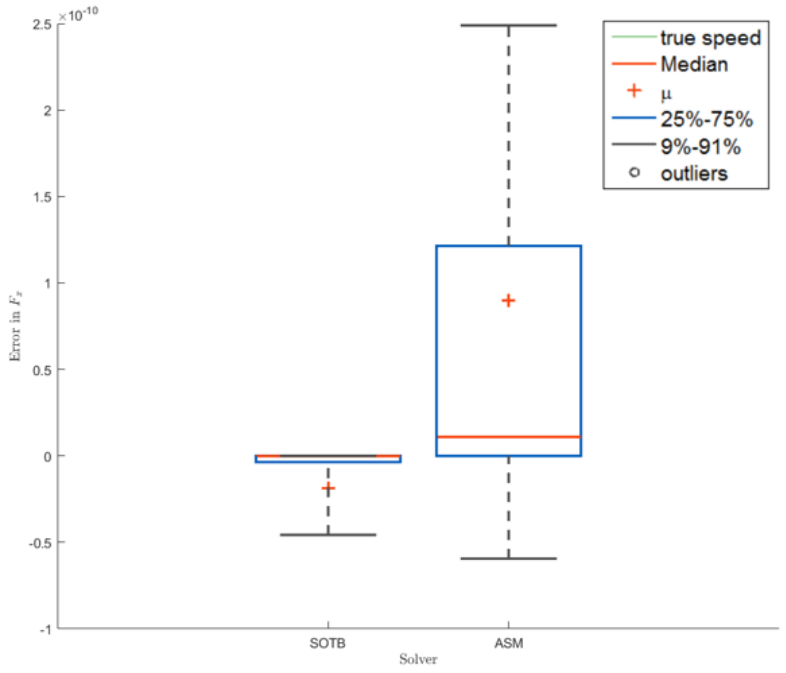


Figure 6.8: Statistics of the error distribution for the ASM compared with SOTB in F_x

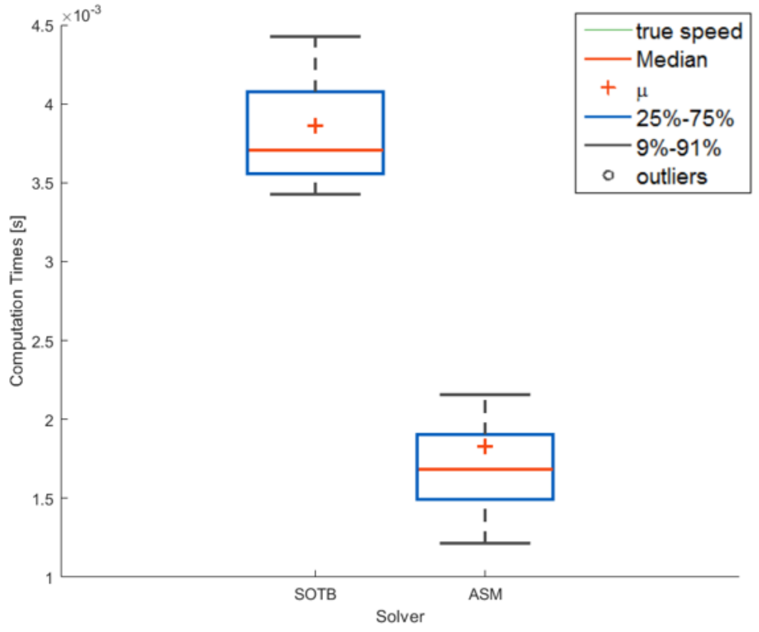


Figure 6.10: Statistics of computation times for the ASM compared with SOTB on an Intel® Core™ i5 processor

In this case, as it's possible to notice from boxplots in figure 6.10, the computational

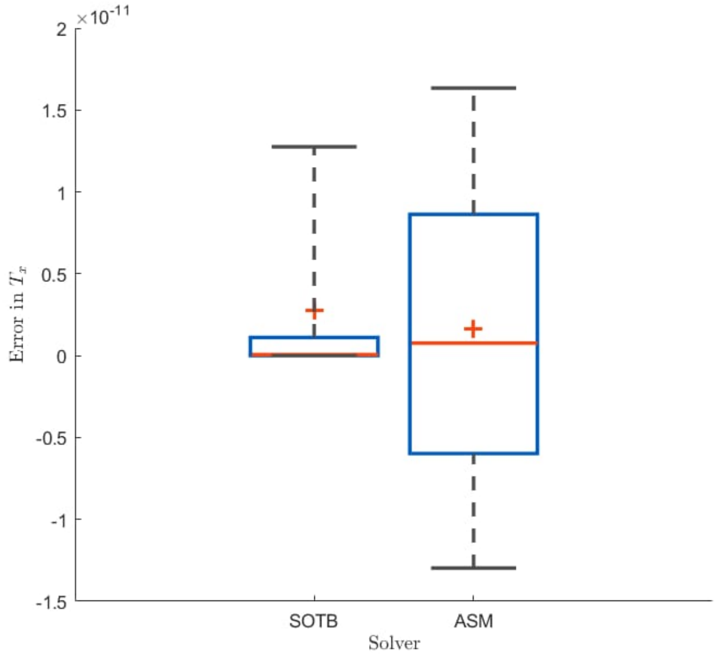


Figure 6.9: Statistics of the error distribution for the ASM compared with SOTB in T_x

times of the ASM is confirmed to be lower that the IPM, always compliant with real time applications. Figure 6.11 and 6.12 shows that the computed control effort for both the solvers present a higher error with respect to the previous case, due to the fact that hard constraints on magnitude of commanded F and T have been softened (has shown in the cost function).
It's also interesting to notice that in this case the two solvers present the same accuracy, even if the tolerance set for the ASM and IPM is different, with the ASM that converges in a fewer number of iterations with respect to the IPM but with the same level of accuracy.

Case 3, 4, 5: softened constraints

Test cases 3, 4 and 5 are constitued by variations of the QP presented below:

$$\min_{f_i} \quad \frac{1}{2}f^T Rf + \frac{1}{2}(F - A_{eq}f_i)^T Q(F - A_{eq}f_i) + \frac{1}{2}(T - A_{eq}f_i)^T G(T - A_{eq}f_i) \tag{6.3}$$

$$s.t. \quad A_\phi f_i = 0$$

$$A_\psi f_i = 0$$

$$MIB \leq f_i \leq F_{max}$$

Boxplots in figures 6.13, 6.14 and 6.15 confirm the considerations previously presented for the cases of hard constraints on F and T and absence of constraints. In this cases, the Hessian matrix associated to the QP resulted to be ill-conditioned most of the times, but the SVD implemented in the solver guarantees a further degree of numerical stability, leading in any case to a fast convergence in a finite number of iterations.

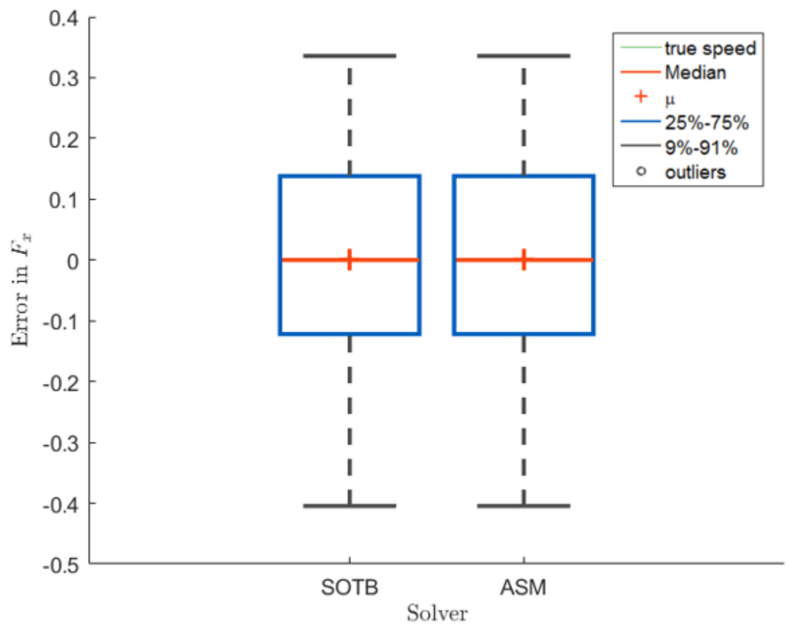


Figure 6.11: Statistics of the error distribution for the ASM compared with SOTB in F_x

Case 5: hard constraints on F, softened constraints on T

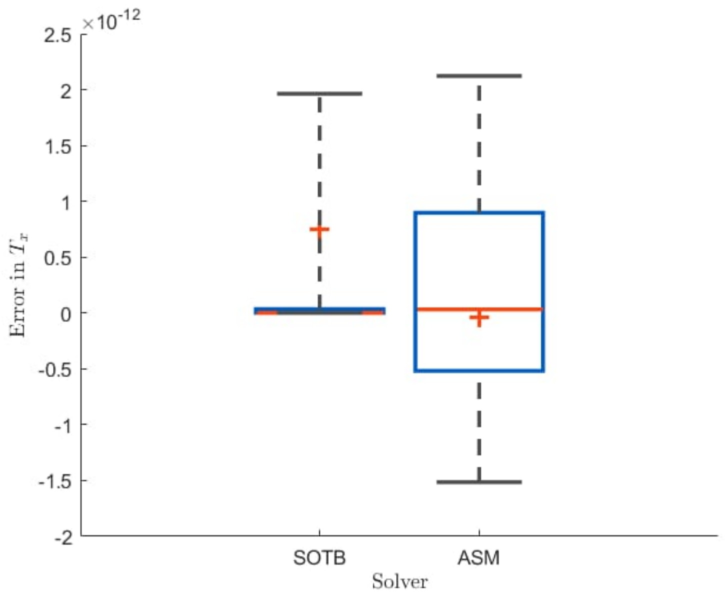


Figure 6.18: Statistics of the error distribution for the ASM compared with SOTB in T_X

Test case 5 combines both hard constraints on T and softened constraints on F to the formulation presented for test cases 3 and 4; from figure 6.19 it's possible to appreciate leads to small reconstruction error for both the ASM and the IPM solver for what concerns the T value. Figure 6.18 presents the reconstruction error on F, higher if compared

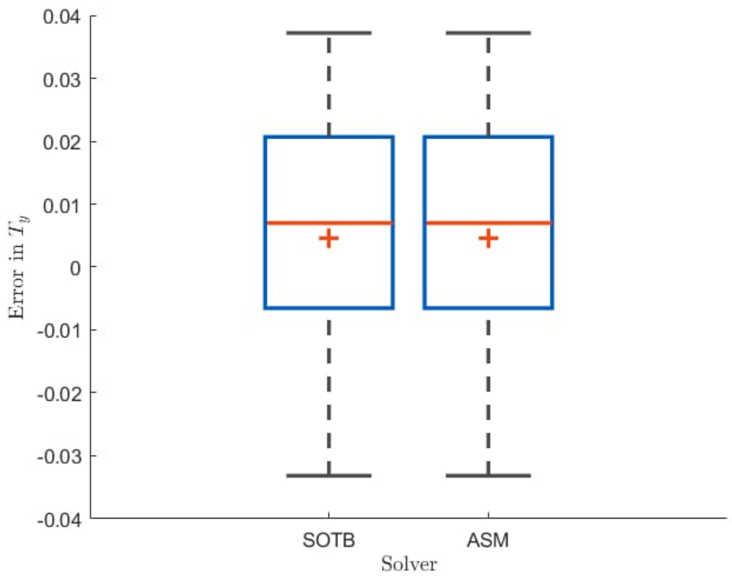


Figure 6.12: Statistics of the error distribution for the ASM compared with SOTB in T_y

with torque reconstruction error but justified by the softening of the hard constraints on force.

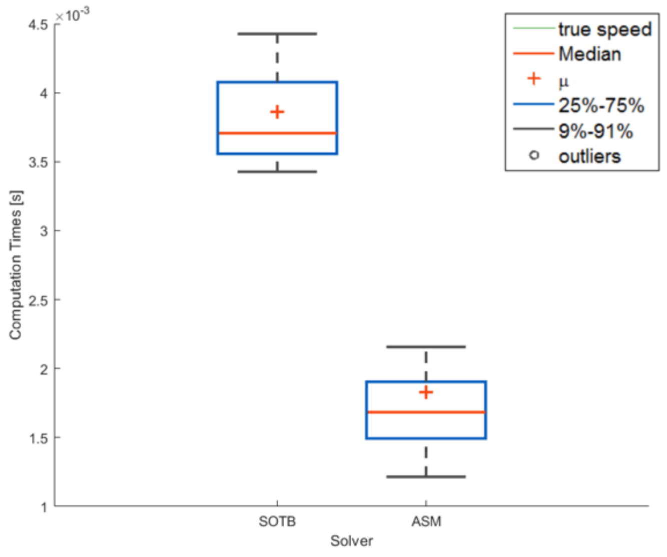


Figure 6.13: Statistics of computation times for the ASM compared with SOTB

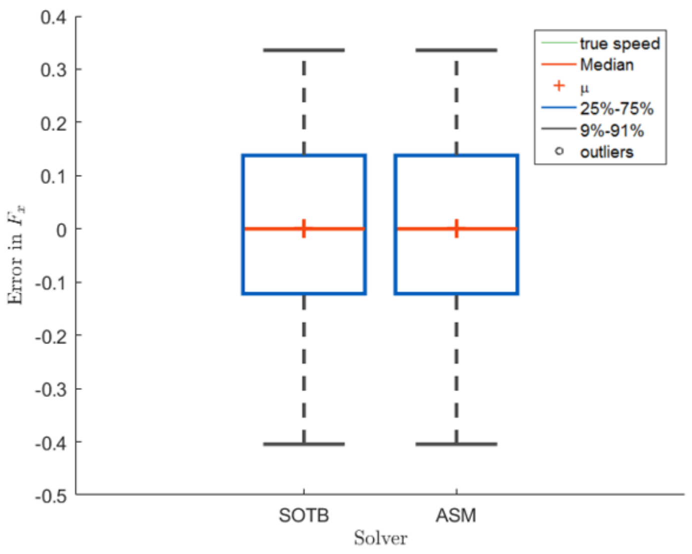


Figure 6.14: Statistics of the error distribution for the ASM compared with SOTB in F_x

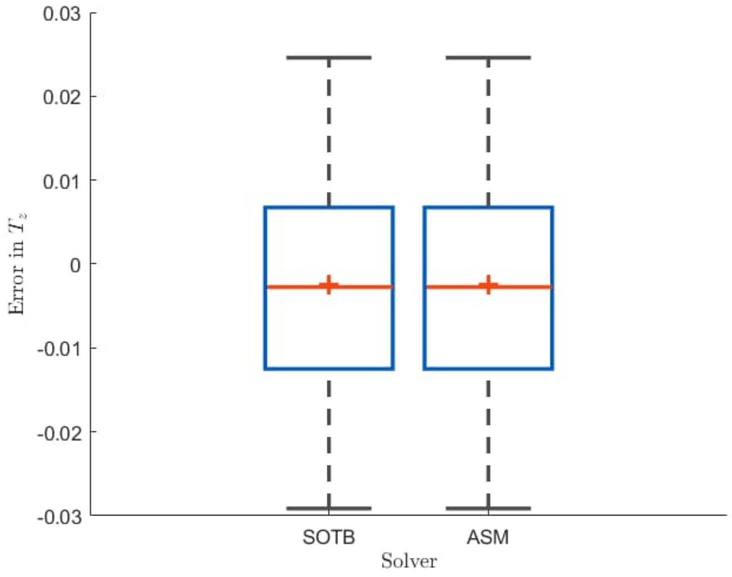


Figure 6.15: Statistics of the error distribution for the ASM compared with SOTB in T_z

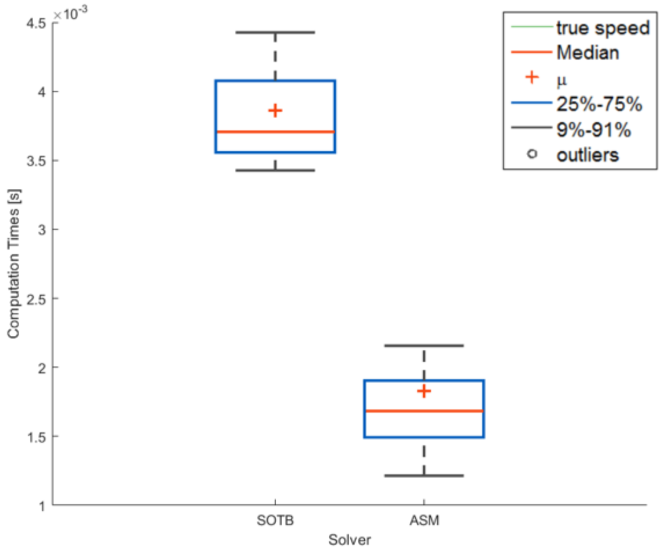


Figure 6.16: Statistics of computation times for the ASM compared with SOTB

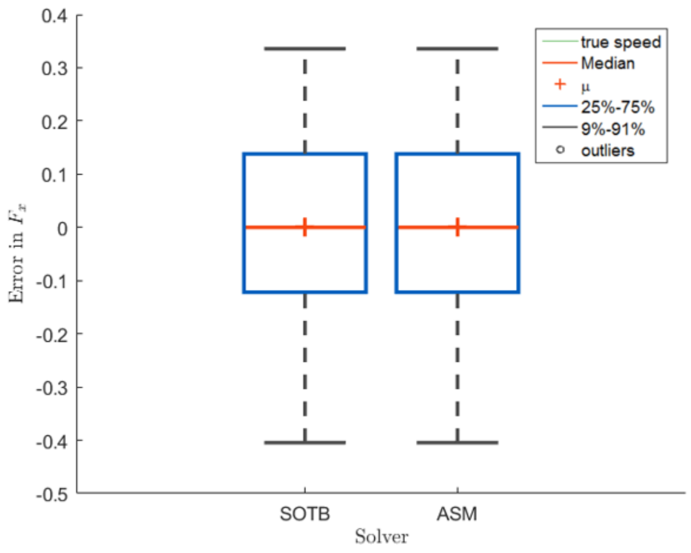


Figure 6.17: Statistics of the error distribution for the ASM compared with SOTB in F_x

Chapter 7

Conclusions and future work

7.1 Conclusions

This work aims at evaluating the online optimization strategy planned for a rendezvous manoeuvre through the usage of MPC, starting from the analysis of the scenario and the formulation of a convex optimization problem to the development of an HIL framework capable of reproduce the 6 DOF motion of a chaser spacecraft that executes the trajectory. The optimization of the Guidance of the vehicle has been handled only focusing in the translational state problem, adopting a SENER Aeroespacial in-house developed tool, SOTB, to handle the MPC control scheme powered by an IPM solver.
Once the Guidance problem has been formulated, the MPC logic has been translated into a control loop developed in Matlab®/Simulink®/Python®, to execute the optimization of the trajectory online with the usage of a robotic arm, regarding as target of the manoeuvre the middle point between four ArUco markers, placed on a flat surface orthogonal with respect to the basis reference frame of the robot. The control loop has been first provided only with Guidance algorithms, while in a second moment a Navigation block has been added, to simulate the visual sensors data exchange of the spacecraft in orbit, recreating in the GNC Laboratory the illumination conditions to which the vehicle is subject in space.
The HIL testings show the adaptability of the MPC to real-time implementations, optimizing the rendezvous manoeuvre online and preventing the robotic arm to fall into singular configurations; additionally, the integration of Guidance and Navigation proved to be successful, allowing an accurate tracking of the optimized trajectory in realistic illumination conditions and taking into account the dynamics of the system under analysis (chaser and target). The implementation of a SH strategy in the MPC shows a dynamic approach to forecasting that adapts to changing conditions by updating the prediction horizon, supported by an appropriate tuning of the optimization problem parameters and simulation of the system dynamics.
In the context of the development of the HIL control loop, the integration of a Control Allocation optimization strategy has been planted, starting from five different formulations of the thrust dispatching problem as a QP, powered by an ASM based solver that has been developed in Matlab® without recalling any pre-defined Matlab® function. The ASM algorithm solution has been compared to the one provided by SOTB, revealing a fast convergence to the global optimum when a warm starting of the active constraints is provided, characteristic that results particularly useful when it comes to real-time implementations, due to the fact that the computations required at each iteration of the ASM solver are cheaper with respect to the IPM (the most expansive one is the inversion of the KKT matrix associated to the QP). Also, the ASM presents an accuracy level in in the same range as SOTB when optimizing each one of the thrust dispatching test cases, avoiding saturation of the thrusters and minimizing the number of actuators turned on.

7.2 Future work

The implementation of the framework presented in this work and the ASM offer various starting points for further developments, some of them yet planted in correspondence of the end of this first investigation phase.
The HIL testing facility has as final objective the capability of execute simulations of in-orbit manoeuvres in an automatized way and in real-time; the robotic arm will be upgraded with a processor integrated with the autocoded control loop, getting rid of the communication protocol needed by the socket Matlab®/Python®/Simulink®, to evaluate the computational performance of the algorithms developed. In the specific case of the scenario under analysis, this objective is related to the formulation of a definite Guidance optimization problem; for the time being, attitude is assumed to be fully controlled by a PD, however a more realistic simulation of the GNC scenario will include the control and optimization of the rotational configuration of the spacecraft, handling the nonlinearity of the attitude problem in through Sequential Convex Programming (SCP) for example.
The scenario presented in this work regards a cooperative target, however further investigations will regard uncooperative scenarios, in which the target can't provide any information about its state, as it happens in the case of space debris. In this case, the target will be simulated with a UR10e robotic arm, yet available in the GNC Laboratory of the URJC. The simulation facility will be integrated with a 4-meter-long rail to overcome the limitation in the maximum extension of the robotic arm and progressively increasing the scaling factor, to simulate the online optimization with a higher accuracy.
For what concerns the ASM, the algorithm will be reviewed in order to make it suitable for the rapid accelerator mode in Simulink® and autocoding, without recalling pre-defined Matlab® toolboxes/functions, to integrate it in the framework of more complete GNC simulation environments and to plan real-time HIL simulations, investigating a wider range of control schemes based on QPs, such as MPC or SQP.

Bibliography

[1] Feras Alasali et al. “A Comparative Study of Optimal Energy Management Strategies for Energy Storage with Stochastic Loads”. In: (May 2020).

[2] Bassam Alrifaee. *Networked Model Predictive Control for Vehicle Collision Avoidance*. May 2017.

[3] Jenny C. R. Asencio, Roman Ivanovitch Savonov, and Rodrigo Intini Marques. “An Open-source Solver to Model the Catalytic Decomposition of Monopropellants for Space Thrusters”. In: *Journal of Aerospace Technology and Management* (Apr. 2020).

[4] *Base coordinate system documentation*. https://forum.universal-robots.com/t/base-coordinate-system-documentation/140.

[5] *Bipropellant Thrusters*. https://www.space-propulsion.com/spacecraft-propulsion/bipropellant-thrusters/index.html.

[6] M. Bodson. “Evaluation of Optimization Methods for Control Allocation”. In: *Journal of Guidance, Control and Dynamics, Volume 25* (July 2012).

[7] T. Boge et al. “EPOS-A Robotics-Based Hardware-in-the-Loop Simulator for Simulating Satellite RvD Operations”. In: (Aug. 2010).

[8] F. Cacciatore. “Attitude Dynamics and GNC, Sensors and Actuators”. In: (2022).

[9] Juan Pablo Romero Camacho, David Paez Ramirez, and José Guillermo Guarnizo Marín. *UR3 Modelo Cinemático Inverso*.

[10] *Cold-Gas Thruster*. https://www.satnow.com/products/thrusters/dawn-aerospace/36-1161-b20.

[11] A. Fejzic et al. “Results of SPHERES Microgravity Autonomous Docking Experiments in the Presence of Anomalies”. In: (Jan. 2008).

[12] Piotr Felisiak. *Control of Spacecraft for Rendezvous Maneuver in an Elliptical Orbit*. June 2015.

[13] *Geometric Interpretation of Convexity*. https://math24.net/convex-functions.html.

[14] Jesus Omar Ocegueda González. “Active Sets, Gradient Projection and Interior Point Methods for Quadratic Programming”. In: ().

[15] W. Haolong and S. Haoping. “Online Trajectory Planning for Docking to Tumbling Spacecraft with Active Maneuver”. In: *33rd Chinese Control and Decision Conference (CCDC)* (22-24 May 2021).

[16] O. Härkegård. “Efficient Active Set Algorithms for Solving Constrained Least Squares Problems in Aircraft Control Allocation”. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002* (Mar. 2003).

[17] Juan L. Jerez, Eric C. Kerrigan, and George A. Constantinides. “A Condensed and Sparse QP Formulation for Predictive Control”. In: *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)* (Jan. 2011).

[18] C. M. Jewison. “Guidance and Control for Multi-stage Rendezvous and Docking Operations in the Presence of Uncertainty”. In: (2017), pp. 251–267.

[19] S. Kannan et al. “Model Predictive Control for Spacecraft Rendezvous”. In: *Control Engineering Practice, Volume 20, Issue 7* (July 2012), pp. 695–713.

[20] Lau et al. “A comparison of interior point and active set methods for FPGA implementation of model predictive control”. In: *European Control Conference, EUCA 2009* (2009).

[21] Giovanni Lavezzi, Mariusz Eivind Grøtte, and Marco Ciarcià. “Attitude Control Strategies for an Imaging CubeSat”. In: *19th Annual IEEE INTERNATIONAL CONFERENCE ON ELECTRO INFORMATION TECHNOLOGY* (July 2019).

[22] D. Lee and H. Pernicka. “Optimal Control for Proximity Operations and Docking”. In: *International Journal of Aeronautical and Space Sciences* (Sept. 2011).

[23] Sven Leyffer. *Deterministic Methods for Mixed-Integer Nonlinear Programming.* Jan. 1993.

[24] Q. Li et al. “Model predictive control for autonomous rendezvous and docking with a tumbling target”. In: *Aerospace Science and Technology, Volume 69* (Oct. 2017), pp. 700–711.

[25] J. M. Longuski, J. J. Guzmán, and J. E. Prussing. *Optimal Control with Aerospace Applications.* Springer, Nov. 2013.

[26] D. Malyuta and Behçet Açıkmeşe. “Fast Homotopy for Spacecraft Rendezvous Trajectory Optimization with Discrete Logic”. In: *AIAA Journal of Guidance, Control, and Dynamics* (July 2021).

[27] D. Malyuta et al. “Convex Optimization for Trajectory Generation-A Tutorial On Generating Dynamically Feasible Trajectories Reliably And Efficiently”. In: *IEEE Control Systems Magazine, Volume 42* (Oct. 2022), pp. 40–113.

[28] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization-Second Edition.* Springer, Dec. 2006.

[29] Farzad Noorian and Philip Leong. “On Time Series Forecasting Error Measures for Finite Horizon Control”. In: *IEEE Transactions on Control Systems Technology* (June 2016).

[30] Miguel A. Nunes, Trevor C. Sorensen, and Eric J. Pilger. “On the development of a 6DoF GNC framework for docking multiple small satellites”. In: *AIAA Guidance, Navigation, and Control Conference, AIAA Science and Technology* (Jan. 2015).

[31] *ODYS QP Solver.* https://www.odys.it/qp-solver-for-embedded-optimization/.

[32] Carlo Alberto Pascucci, Michael Szmuk, and Behçet Açıkmeşe. “Optimal Real-Time Force Rendering for On-Orbit Structures Assembly”. In: *GNC 2017: 10th International ESA Conference on Guidance, Navigation & Control Systems* (May 2017).

[33] Camille Pirat et al. “Guidance, Navigation and Control for Autonomous Cooperative Docking of CubeSats”. In: *The 4S Symposium 2018* (28th May - 1st June 2018 2018).

[34] Rien Quirynen and Stefano Di Cairano. “Block-Structured Preconditioning of Iterative Solvers within a Primal Active-Set Method for fast MPC”. In: (Dec. 2019).

[35] J. Ramírez et al. “Embedded Optimization for Space Rider Re-entry Module Parafoil GNC”. In: *8th European Conference for Aeronautics and Space Sciences (EUCASS)* (2023).

[36] *Robot singularity.* https://www.mecademic.com/academic_articles/singularities-6-axis-robot-arm/.

[37] G. Rouleau et al. "Autonomous capture of a tumbling satellite". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006* (26 June 2006).

[38] J. C. Sanchez et al. *A flatness-based predictive controller for six-degrees of freedom spacecraft rendezvous.* Feb. 2020, pp. 391–403.

[39] Bruno Siciliano et al. *Robotics-Modelling, Planning and Control.* Springer, Aug. 2010.

[40] Paulina Swiatek et al. "Guidelines on Safe Close Proximity Operations". In: *First Int'l. Orbital Debris Conf.* (2019).

[41] *Universal Robots UR3e.* https://www.wiredworkers.io/cobot/brands/universal-robots/ur3e/.

[42] S. Silvestrini V. Pesce A. Colagrossi. *Modern SpacecraftGuidance, Navigation and Control-From System Modeling to AI and Innovative Applications.* Elsevier, Nov. 2022.

Appendix A

GNC Laboratory Equipment

A.1 UR3e robotic arm

The UR3e robotic arm is a collaborative robotic arm designed for tasks that require precision and flexibility. It is known for its ease of use, quick setup, and safety features, making it suitable for a wide range of applications, including assembly, packaging, machine tending and more.

A.1.1 Technical Specifications

- **Payload Capacity:** The UR3e has a maximum payload capacity of 3 [kg] (6.6 pounds).
- **DOF:** The UR3e presents 6 DOF, which allows it to move in six different directions and orientations, providing great flexibility in motion.
- **Weight:** The robot itself weights approximately 11 kilograms (24.3 pounds), making it relatively lightweight for a robot of its capabilities.
- **Speed:** The UR3e is designed for precision and accuracy rather than for high speed operations. It moves at a maximum speed of 1[m/s] (39.4 inches per second).
- **Precision:** It has a repeatability of $\pm$ 0.1[mm], which ensures precise and consistent movements.
- **Collaborative Features:** The UR3e is designed to work safely alongside humans without the need for safety cages. It is equipped with force/torque sensors in its joints, enabling it to detect and respond to collisions, making it a collaborative robot (*cobot*).
- **Control System:** The UR3e uses the Universal Robots'intuitive programming system, which allows for easy programming by teaching the robot tasks manually or using a graphical user interface. It also supports external programming interfaces.
- **Mounting Options:** The UR3e can be mounted in various orientations, including floor, ceiling or tabletop (the case of this thesis), providing flexibility for different workspaces setup.
- **Communication:** It offers various communication options, including Ethernet and USB interfaces, to connect and control the robot from external devices or systems.
- **Safety Features:** The robot has built-in safety features like collision detection, a TCP speed limit and adjustable force and torque limits to ensure safe operation around humans.

- **Power Supply:** The UR3e typically operates on a standard $110 - 230[VAC]$ power source.

The robotic arm is accompanied by a control box that houses the robot's control system and provides the necessary power and communication interfaces.

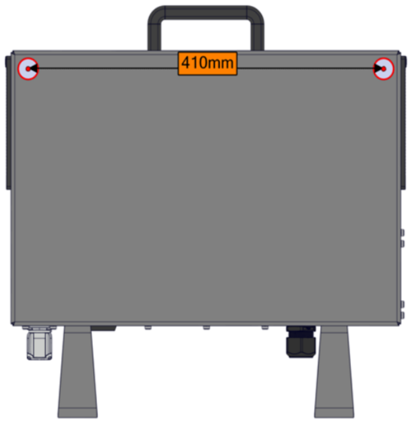


Figure A.1: UR3e control box, picture taken from [41]

A.1.2 Technical Specifications

- **Hardware:** The control box contains the hardware required for the robot's operation, including the CPU and memory.
- **Power Supply:** It includes a power supply unit, compatible with standard $110 - 230[VAC]$ power sources.
- **Communication Ports:** It provides various communication options, including Ethernet and USB ports, used to connect the robot to external devices, networks or control systems, enabling remote operations and data exchange.

The PolyScope teach pendant is an accessory that accompanies the UR3e robotic arm, providing a user-friendly interface for users to interact with the robot and execute various tasks. It can be conncected to the robot's control box via cable and includes safety features such as an emergency stop button and password protection to prevent unauthorized access to robot's functions.

Figure A.2: UR3e teach pendant, picture taken from [41]

Specifications	
Payload	3.3[kg]
Reach	500[mm]
DOF	6 rotating joints
Programming	Touchscreen with PolyScope graphical user interface
Footprint	Diameter: 128[mm]
Materials	Aluminium, Plastic, Steel
Weight included cable	11.2[kg]
Power Consumption	Maximum Average 300[W]
Force Sensing, Tool Flange (Force)	Range: 30.0[N]
	Precision: 2.0[N]
	Accuracy: 3.5[N]
Force Sensing, Tool Flange (Torque)	Range: 10.0[Nm]
	Precision: 0.1[Nm]
	Accuracy: 0.1[Nm]
Pose Repeatability per ISO 9283	± 0.03[mm]
Axis movement (Working range)	Base: ± 360°
	Shoulder:± 360°
	Elbow: ± 360°
	Wrist 1: ± 360°
	Wrist 2: ± 360°
	Wrist 3: Infinite
Axis movement (Maximum speed)	Base: ± 180[°/s]
	Shoulder:± 180[°/s]
	Elbow: ± 180[°/s]
	Wrist 1: ± 360[°/s]
	Wrist 2: ± 360[°/s]
	Wrist 3: ± 360[°/s]
Typical TCP speed	1[m/s]
Noise	Less than 60[dB]
I/O Ports	Digital in: 2
	Digital out: 2
	Analog in: 2

Table A.1: Technical details of the UR3e and of the correlated teach pendant