POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Coordinated trajectory planning of a multi limb free-flying space manipulator for the capture of collaborative satellites using RRT*

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Andrea Zosi**

Student ID: 226258
Advisor: Prof. Mauro Massari
Co-advisors: Niccolò Faraco
Academic Year: 2023-24

Abstract

The use of satellites equipped with fully autonomous multi limb space manipulator is a topic towards which space companies are nowadays increasingly focusing their attention: indeed, the new paradigm of on-orbit servicing missions is arising, comprising a variety of operations which spans from refueling, inspecting and repairing of active satellites to the active removal of debris and defunct satellites.
In this context, this thesis addresses the investigation and development of numerical solutions to simulate the task of capturing cooperative satellites by a spacecraft equipped with a dual limb space manipulator.
Specifically, the work focuses on the modeling of the multibody dynamics of the system and on the coordinated trajectory planning for the end-effectors of the manipulator's limbs. The final goal is therefore to drive them toward predefined grasping points on the target satellite, ensuring great accuracy both in terms of position and orientation; to do that, their trajectories are generated using the near-optimal Rapidly-exploring Random Tree Star (RRT*) algorithm, taking into account problems like obstacle modeling, collision avoidance and satellite's base stabilization.

Keywords: On-orbit servicing; Multi limb space manipulator; Trajectory planning; RRT* algorithm

Abstract in lingua italiana

L'utilizzo di satelliti equipaggiati con manipolatori spaziali multi-arto è un argomento verso il quale le aziende operanti in ambito spaziale stanno sempre più concentrando le loro attenzioni: infatti, sta emergendo un nuovo paradigma rappresentato dalle missioni di manutenzione in orbita, comprendenti operazioni che vanno dal rifornimento, l'ispezione e la riparazione di satelliti attivi fino alla rimozione attiva di detriti e satelliti non più operativi.
In questo contesto, questa tesi affronta lo studio e lo sviluppo di soluzioni numeriche al fine di simulare la cattura di satelliti cooperativi ad opera di uno veicolo spaziale equipaggiato con un manipolatore spaziale a doppio arto.
Nello specifico, questo lavoro si concentra sulla modellazione della dinamica multicorpo del sistema e sulla generazione coordinata della traiettoria degli end-effectors dei due arti del manipolatore. Lo scopo finale è quindi quello di guidarli verso dei punti di affranco predefiniti, garantendo grande precisione sia in termini di posizione che di orientamento; per fare ciò, le loro traiettorie sono generate mediante l'utilizzo dell'algoritmo quasi-ottimale RRT*, tendendo in considerazione problemi come la modellazione degli ostacoli, l'evitamento delle collisioni e la stabilizzazione della base del satellite.

Parole chiave: Manutenzione in orbita; Manipolatore spaziale multi-arto; Pianificazione della traiettoria; Algoritmo RRT*

Contents

1 | Introduction

This section firstly provides a brief overview of the most important past and planned space missions that employ space manipulators, in order to contextualize this thesis in the current state of the art of research on Space Robotics. Then, a review of the multibody dynamics models used to simulate the coupled behavior of the satellite and the multi limb manipulator is provided. Eventually, a comparison of the algorithms developed to efficiently guide the manipulators toward the targets is presented, highlighting their pros and cons with respect to the objectives of this thesis.

1.1. Space Robotics for On-Orbit Servicing

On-Orbit Servicing refers to all the activities performed directly in space to maintain, inspect, repair, upgrade, refuel, or decommission satellites and other spacecrafts. Nowadays, in a world increasingly focused on sustainability and reuse, also the most important space organizations are focusing their efforts to develop technologies and operations to make space more sustainable; indeed, by 2030, the European on-orbit servicing market alone is estimated to reach more than €5 billion , with a growth rate of 11.5% per year [1].
In this context, the choice of employing space manipulators is gaining more and more popularity: first of all, it avoids exposing astronauts to the countless hazards that characterize the space environment; moreover, it allows to create very versatile servicing satellites that can be used for a variety of different tasks, from de-orbiting defunct satellites and large debris, to refueling and inspecting active satellites to extend their operational life.

Even though servicing of satellites has been theoretically considered since the early days of spaceflight, only in the last 30-40 years there have been actual missions and demonstrations proving that not only the use of robotic manipulators was possible, but also very helpful.
The most famous example is the key role that the *Shuttle Remote Manipulator System (SRMS)*, commonly known as *Canadarm1*, played in the five servicing missions of the Hubble Space Telescope between 1993 and 2009.

Another key mission, the *ETS-VII* (Engineering Test Satellite VII), launched by Japan in 1997, demonstrated the capability of autonomous rendezvous and docking using one tele-operated robotic arm with 6 degrees of freedom.
In 2007, instead, the *Orbital Express Mission* by the U.S. Defense Advanced Research Projects Agency (DARPA) and NASA, demonstrated satellite on-orbit servicing by autonomously refueling a satellite and performing other maintenance operations, marking a significant milestone in proving that those kind of tasks could be performed by space manipulator systems.

Currently, the majority of the actively used space manipulators for on-orbit operations are installed on the ISS. Here, four main arms can be identified: the *Canadarm2*, that is the principal and most used manipulator, has 7 degrees of freedom and is used for a variety of tasks, such as capturing cargo spacecraft, moving large payloads, assembling parts of the space station and assisting with astronauts' spacewalks; the *Dextre*, that is a smaller, two-limb robotic manipulator designed for finer manipulation task; the *European Robotic Arm (ERA)*, developed by ESA and designed for moving equipment, supporting spacewalks, and performing maintenance tasks; the *Japanese Experiment Module Remote Manipulator System (JEM-RMS)*, installed on the Kibo module and designed to perform with similar task as the *European Robotic Arm*.

For what concerns instead planned missions, there are two of them in particular that are the most relevant with the topics treated in this thesis:

- The Mission Robotic Vehicle (MRV), developed by Northrop Grumman Corporation: expected to be launched in 2026, the MRV satellite is equipped with two manipulators and is designed to extend geostationary satellite lifespans through satellite repair (including detailed inspections), repositioning, and debris removal [2].
- The ClearSpace-1 mission, developed by ESA: its launch is planned to occurr in 2028 and its final goal is to rendezvous with, capture and safely remove from orbit the ESA's 95 kg PROBA-1 satellite, launched in 2001. To accomplish this task, the spacecraft will be equipped with four manipulators, resulting in the very first-ever mission to remove an unprepared and uncooperative object from orbit [3].

1.2. Multibody Dynamics Models

A first distinction about the way the satellite-manipulator system operates can be made following the classification presented in [4], where five main maneuvering modes are iden-

tified:

1. **Floating**: a spacecraft-manipulator system is defined to be *floating* when maneuvering in a 6-dofs under-actuated mode, in which only the manipulator joints are actively controlled; therefore, the 3-dofs of orientation of the base-spacecraft and the 3-dofs of translation of the system's center-of-mass are not actively controlled.

2. **Rotation-Floating**: a spacecraft-manipulator system is defined to be *rotation-floating* when maneuvering in a 3-dofs under-actuated mode, in which the N-dofs associated to the manipulator's joints and the 3-dofs of orientation of the base-spacecraft are actively controlled by internal torques only.

3. **Rotation-Flying**: a spacecraft-manipulator system is defined to be *rotation-flying* when the N-dofs of the manipulator joints are actively controlled by joint motor torques and the 3-dofs of orientation of the base-spacecraft are actively controlled by external torques only (ex: using thrusters). Therefore, the system's total linear momentum is in this case conserved, while the angular momentum is time-varying.

4. **Translation-Flying**: a spacecraft-manipulator system is defined to be *translation-flying* when both the N-dofs of the manipulator joints and the 3-dofs of translation of the base-spacecraft are actively controlled. Therefore, the system's total angular momentum is in this case constant, while the linear momentum is time-varying.

5. **Flying**: a spacecraft-manipulator system is defined to be *flying* when maneuvering in a mode in which all of the N-dofs of the manipulator joints are actively controlled by joint motor torques and the 6-dofs of motion of the base-spacecraft are actively controlled by external forces. In this case, both the system's total angular momentum and the linear momentum are time-varying.

The choice of the most suited maneuvering mode can vary depending on the control strategy adopted during the servicing mission as well as on the assumptions introduced in the dynamic model; these same assumptions, which can be, for example, to consider only rigid bodies and to neglect the long-term effects of orbital mechanics, also drastically influence the way the multibody system is mathematically modeled. According to the literature, there are two main methods used for modeling the dynamics of such systems and eventually deriving the equations of motion [4]. The first one is the well known *Lagrangian approach*, which allows to retrieve in a systematic way the coupled equations of motion of the system starting with the definition of kinetic and potential energies and of a set of generalized coordinates; the other one is the *Newton-Euler method*, a recursive algorithm used to describe the motion of each body focusing only on their

physical interconnections and on the exchange of forces and moments between each of them.
The main advantage of the *Newton-Euler method* is its computational efficiency, since it allows to analyze separately each body before linking each of them via constraints. On the other hand, the *Lagrangian approach* wins the comparison regarding versatility and simplicity of derivation of the equations, thanks to its systematic procedure.

Some considerations must then be made on the possible ways to account for the dynamic coupling when controlling the position and orientation of both the end-effectors of the manipulators and of the satellite's base; according to [5], three main methods can be identified: the *Virtual Manipulator approach* (VM), the *Dynamically Equivalent Manipulator method* (DEM) and the *Generalized Jacobian Matrix approach* (GJM).
The VM approach works by replacing the physical satellite-manipulators system with a purely massless kinematic chain, formulated based on the geometrical and mass properties of the whole system. The base of this virtual manipulator is represented by a spherical joint located at the center of mass of the physical system and its orientation equals the orientation of the satellite's base with respect to an inertial fixed frame [6]. This method is widely used when dealing with manipulator's workspace analysis, but its main drawback is that it can not be represented by a physical manipulator. This last problem can be solved by using, instead, the DEM method: here, all the links of the DEM have now masses and moments of inertia that are calculated according to the real mass and inertia distribution of the satellite-manipulators system [7].
The GJM approach, on the other hand, allows to simplify the formulation of the relationship between joint velocities and end-effectors velocities with the motion of the base-satellite, even with systems where traditional kinematic models do not directly apply, like in redundant manipulators [4].

In general, whichever of the methods just illustrated is chosen to model multibody dynamics, a solid and accurate kinematic model of the system must be previously developed. In this context, two approaches can be identified in the literature, the *Direct Path method* and the *Barycentric Vector approach* [8]. The main difference between the two is the point of the system that is chosen as a reference for writing the kinematic relations: in the first one, this point is chosen to be the center of mass of the satellite's base, while, according to the other one, the reference is the center of mass of the whole system (composed by satellite's base and the robotic manipulators). The first method is usually preferred with respect to the latter, since it allows a easier implementation of the attitude control laws of the satellite-manipulator system.

1.3. Trajectory Planning Algorithms

In recent years many families of algorithms have been developed for planning a feasible trajectory to be followed by space robots, addressing problems like collision avoidance and dealing with all the other kind of constraints that can be typical of on-orbit servicing missions [9]. Hereafter, some of the key techniques used for achieving such collision-free trajectories are briefly introduced.

- **Grid-Based Methods**
  The working principle of this family of methods is to subdivide the workspace into a grid, where each cell/node represents a discretized region of solution search space. The goal is to find a feasible path from a *start node* to a *goal node* through these interconnected cells, while avoiding obstacles (which can be represented, for example, by high-cost nodes). The path's cost can be measured by the number of steps, distance, or other metrics like time or energy.
  In [10] it is shown how the A* algorithm, which belongs to this family, can be used efficiently to determine a minimum distance collision-free path from the starting stowed pose of a 3-dofs space manipulator to the designed grasping pose. In [11], instead, the authors propose a solution to obtain an optimal collision-free path of a 7-dofs space manipulator using the A* algorithm. Though, the main drawback of this family of methods is that they are generally not suited if the space in which they are searching a path is high-dimensional, as the number of nodes in the grid to be checked grows exponentially with an increase in space dimensions.

- **Artificial Potential Field**
  The Artificial Potential Field (APF) method is an approach to path planning where the manipulator is seen as a point that moves in the solution search space under the influence of an artificial force field: in particular, attractive forces are generated towards the goal, driving the manipulator's end-effector towards it, while repulsive forces are exerted by obstacles, pushing the end-effector away in order to avoid collisions. Authors in [12] demonstrate the effectiveness of the use of APF methods to address the issue of trajectory planning for a free-floating dual-arm space robot, considering also the constraints of collision avoidance while simultaneously minimizing the base position and attitude disturbance during on-orbit tasks. In [13], the APF method, usually used in 2D space, is firstly extended to 3D space and then integrated with the GJM approach to generate a collision-free trajectory of a 6-dofs space manipulator.
  One of the main drawbacks of APF methods is that they can get 'stuck' in local

minima or in a condition where the net force acting on the point-like manipulator is zero, leading in both cases to failure in reaching the target position.

- **Sampling-Based Methods**
  The idea behind sampling-based methods involves randomly or strategically sampling the configuration space of the system (i.e. the manipulator) to find a feasible path from a starting configuration to a desired one, while also avoiding all the obstacles present in the workspace. This family of methods do not require a complete exploration of the whole configuration space of the manipulator, making them particularly useful for high-dimensional problems where traditional grid-based methods would be impractical.
  One of the algorithms based on this working principle that is widely used in literature is the Rapidly-exploring Random Tree (RRT): in the standard formulation, the algorithm works by generating at each iteration random nodes in the joint space (each corresponding to a precise configuration of the manipulator) that are then linked together to create a tree-like structure that explores the space. The path that is able to connect the initial configuration and the desired one will therefore be nothing but one of the 'branches' of this tree.
  In [14] the authors prove that RRT algorithm is well suited for trajectory planning of a satellite-manipulator system in floating mode, obtaining satisfactory results even when dealing with a 14-dimensional configuration space and other constraints like obstacle avoidance and compliance with manipulator's kinematic joints limits.
  Many variants of RRT algorithms exist: for example, an improved version called RRT* also considers path optimization, aiming thus for the shortest path by 'rewiring' the tree at each iteration.
  In general, sampling-based methods are probabilistically complete, which means that, given a large enough number of nodes, they are guaranteed to find a solution if one exists. This probabilistic nature is what makes this family of algorithms effective for exploring high-dimensional spaces where, instead, other deterministic methods might struggle; but this also means that, for any given run of the algorithm, there's still a chance that it might not find a feasible path even when one exists and a relative high number of samples is considered.

1.4. Proposed Approach and Assumptions

The aim of this thesis is to investigate and develop numerical solutions to simulate the task of capturing cooperative satellites by a servicing-spacecraft equipped with a space

manipulator constituted by two limbs with 6-dofs each.
In this work, the servicing-spacecraft is considered to be in a *Rotation-Flying* mode, since the only degrees of freedom that are actively controlled are those associated to its rotation and those associated to the manipulator's joints. The behavior of this multibody system was modeled starting from an accurate kinematic description, using the *Direct Path method* and with the introduction of the *Denavit-Hartenberg* convention; this was followed by the use of the *Lagrangian approach* to retrieve the system's equations of motion.
For what concerns instead the trajectory planning for the two limbs of the space manipulator, a timing law which respects velocity and accelerations constraints of the joints was created to follow the near optimal collision-less path generated by the RRT* algorithm.

All the algorithms developed, which were created to be flexible and easily adaptable to different capture scenarios and satellites geometries, are based on the definition of some general assumptions, which are briefly described hereafter:

1. The satellite to be captured (*Target*) is assumed to be collaborative and initially in proximity of the servicing spacecraft (*Chaser*).
2. Disturbances like gravity gradient, solar radiation pressure and atmospheric drag are neglected given the limited duration of the capture maneuver.
3. Satellites and manipulator are assumed to be constituted by rigid bodies only.
4. State and geometrical properties are assumed to be known/retrievable for both the *Chaser* and the *Target*.

1.5. Thesis Organization

This thesis is organized as follows. In Chapter 2, the physical models and mathematical approaches used to describe the capture framework are formalized from a kinematic and dynamic point of view; the final goal of this chapter is therefore the derivation of the coupled equations of motion of the multibody system.
Chapter 3 illustrates the RRT* algorithm, used to generate the trajectories for the limbs of the space manipulator, with emphasis given to the fundamental steps involved; the final goal of this chapter is therefore the generation of the two collision-less trajectories that will be given as reference to be followed by the dynamic model.
In Chapter 4 all the parameters needed for the numerical simulation are introduced, and then the final results are presented and discussed. Lastly, in Chapter 5, the thesis work is summarized, with a discussion of possible further improvements of the developed model.

2 | System Modeling

The aim of this chapter is to describe in detail all the physical and mathematical formulations introduced in this thesis to model the behavior of the satellite-manipulator system. Initially, a general reference scenario of the capture environment is presented, with particular attention given to the description of the different reference frames involved in the analysis and the transformations used to move from one to another; then, a detailed kinematic model used to characterize the multi limb manipulator is introduced; afterwards, the approach followed to derive the coupled equations of motion of the multibody system is exhaustively analyzed; finally, an overview of the joints and base attitude controllers is presented.

The final goal of this chapter is therefore to be able to reproduce numerically what is shown in the block diagram of Figure 2.1, with the exception of the RRT* block.

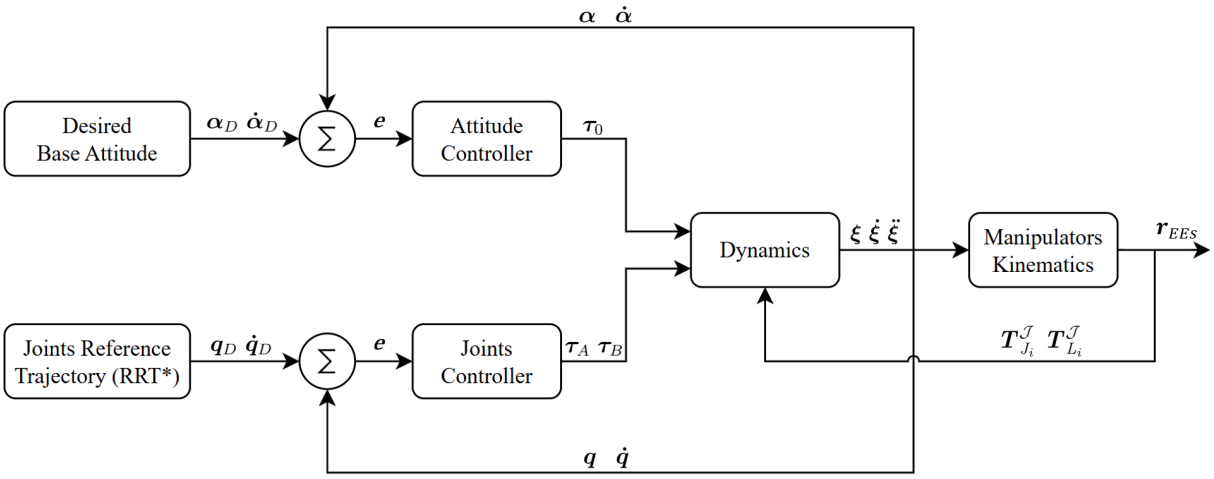

Figure 2.1: Numerical simulation block diagram

All the quantities highlighted in the block scheme will be extensively discussed in this chapter, while the generation of the reference trajectory for the joints of the multi limb manipulator will be analyzed in Chapter 3.

2.1. Capture Scenario and Reference Frames

A visual representation of the capture scenario, involving the servicing-spacecraft (equipped with the multi limb space manipulator) and the target satellite to be captured, is shown in Figure 2.2.

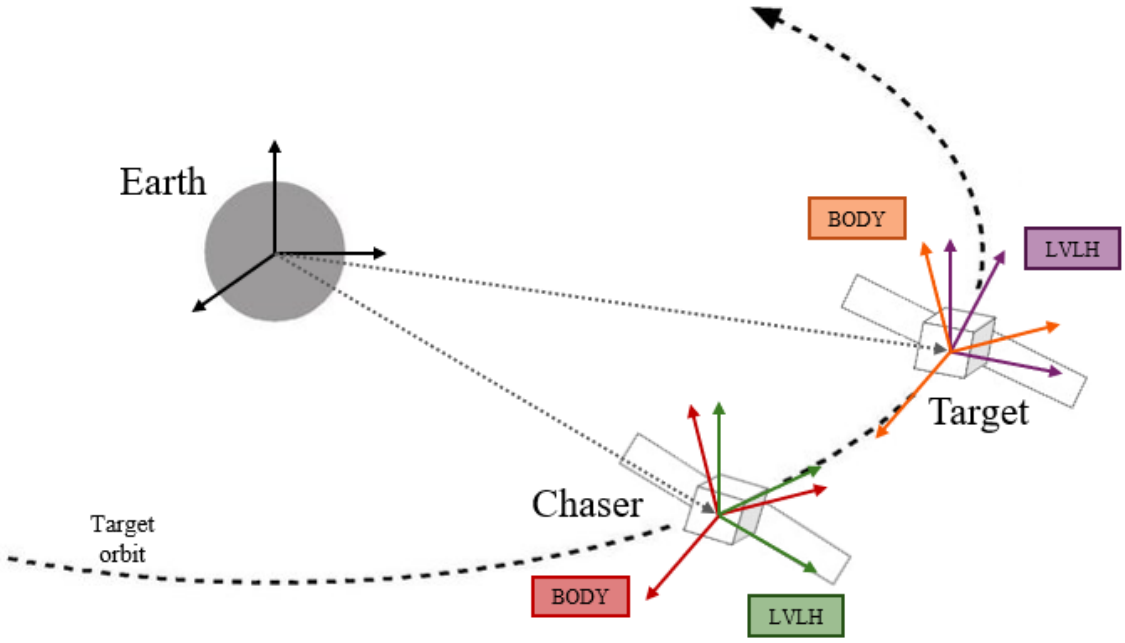

Figure 2.2: Typical capture scenario, highlighting the reference frames involved

In the context of this thesis, the *Chaser* is assumed to have just performed a rendez-vous maneuver, with the effect of bringing it at rather close distances with respect to the *Target*. In particular, this maneuver is thought to be done using a *v-bar* approach, where the *Chaser* progressively adjusts its trajectory to stay aligned with the *Target*'s orbital velocity vector. This kind of approach is well suited for this type of servicing missions because it is relatively fuel efficient and it also allows to be safely interrupted in the case of some problem occurs, thus avoiding potential catastrophic collisions between the two satellites [15].
Once the rendez-vous is completed, the *Chaser* performs a slew maneuver to be finally aligned with the *Target*, in order to ease the capture process.

To express quantitatively the relative position and orientation between the two satellites, three different types of reference frames are defined:

- ***Earth Centered Inertial Frame*** *($\mathcal{I}_{ECI}$)*
  This inertial reference frame is used to describe the position of the satellite relative to the Earth. Its origin is located in the center of the planet, while its axes are

defined as follows: $\hat{\mathcal{X}}_{ECI}$ points toward the vernal equinox, $\hat{\mathcal{Z}}_{ECI}$ is aligned with Earth's rotational axis and $\hat{\mathcal{Y}}_{ECI}$ completes the right-handed system.

- ***Satellite's Body Fixed Frame*** *($\mathcal{B}$)*
  The origins of the two body fixed frames $\mathcal{B}_T$ and $\mathcal{B}_C$ are located at the center of mass of the *Chaser* and *Target* bodies, respectively. Their axes $\{\hat{\mathcal{X}}_C, \hat{\mathcal{Y}}_C, \hat{\mathcal{Z}}_C\}$ and $\{\hat{\mathcal{X}}_T, \hat{\mathcal{Y}}_T, \hat{\mathcal{Z}}_T\}$ are aligned with the principal axes of inertia of the corresponding satellite.

- ***Local Vertical Local Horizontal Frame*** *($\mathcal{L}$)*
  These two frames ($\mathcal{L}_C$ and $\mathcal{L}_T$) have their origin located at the center of mass of the two satellites, as well; they are rotating frames, as their axes change direction while the satellites travel their orbit. In particular, $\hat{\mathcal{X}}_{LVLH}$ is directed as the satellite's position vector (expressed in the $\mathcal{I}_{ECI}$ frame), $\hat{\mathcal{Z}}_{LVLH}$ is perpendicular with respect to the orbital plane and its pointing in the direction of the angular momentum vector, whereas $\hat{\mathcal{Y}}_{LVLH}$ completes the right-handed frame while also pointing in the along-track direction. [16].

Being the two $\mathcal{L}_C$ and $\mathcal{L}_T$ reference frames rotating as the satellites travel along their orbit, and given that they are centered in the respective center of mass, at every time instant they will be differently oriented relative to the $\mathcal{I}_{ECI}$ frame, even if the two satellites are moving along the same orbit at rather close distances.
Nevertheless, taking into account the preliminary assumptions made in Section 1.4, a simplification can be made: indeed, for short-duration capture maneuvers, it is reasonable to assume that orbital mechanics effects and perturbations do not have any significant impact on the motion of these two satellites and, thus, their relative position does not change too much during the capture maneuver (especially if they are placed, as in the treated scenario, on the same orbit at close distances). These aspects allow to consider $\mathcal{L}_C$ and $\mathcal{L}_T$ equal to each other and, in addition, with their orientation fixed in time. Therefore, it is possible to introduce a new inertial frame "$\mathcal{J}$", centered in the *Target* satellite and oriented in space as the old $\mathcal{L}_T$ was. This inertial frame will be used, from now on, as the reference to express the relative state of the *Chaser* satellite and of the two limbs of the space manipulator.

2.2. Chaser Attitude Kinematics

Once all the different reference frames involved in the analysis are defined, it is possible to introduce the problem of how to switch from one frame to another, and thus of how to express the relative position and orientation of all the bodies composing the multibody

system.
To define the relative orientation of one frame with respect to another, a minimum number of three parameters are required: for what concerns the *Chaser*' base, it was chosen to use Euler angles sequence 321 to describe the rotation between its body-fixed reference frame ($\mathcal{B}_C$) and the newly introduced *Target*-centered inertial frame ($\mathcal{J}$).
This specific rotation sequence can be mathematically formulated with the introduction of the attitude matrix $\boldsymbol{A}_{\mathcal{B}/\mathcal{J}}$, that can be computed, as in Equation 2.1, as a sequence of three elementary rotations, each one specified by their respective Euler angle (φ, θ, ψ).

$$A_{\mathcal{B}/\mathcal{J}}(\varphi, \theta, \psi) = R_x(\psi)R_y(\theta)R_z(\varphi) \tag{2.1}$$

Where the elementary rotations[1] are defined as follows:

$$R_z(\varphi) = \begin{bmatrix} cos(\varphi) & sin(\varphi) & 0 \\ -sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2.3}$$

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & sin(\psi) \\ 0 & -sin(\psi) & cos(\psi) \end{bmatrix} \tag{2.4}$$

When dealing with Euler angles it is important to recall that angular rates $\dot{\boldsymbol{\alpha}} = [\dot{\varphi}, \dot{\theta}, \dot{\psi}]^T$ do not coincide with the angular velocity of satellite expressed in the body-fixed frame ${}^{\mathcal{B}}\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$; indeed, to link together these two quantities, it is necessary to introduce the relationship reported in Equation 2.5 [17].

$$\begin{cases} \dot{\varphi} = \omega_z \dfrac{cos(\psi)}{cos(\theta)} + \omega_y \dfrac{sin(\psi)}{cos(\theta)} \\ \dot{\theta} = \omega_y cos(\psi) - \omega_z sin(\psi) \\ \dot{\psi} = \omega_x + \omega_z \dfrac{cos(\psi)sin(\theta)}{cos(\theta)} + \omega_y \dfrac{sin(\psi)sin(\theta)}{cos(\theta)} \end{cases} \tag{2.5}$$

[1]Note that the labels x,y,z of the rotation matrices do not indicate the axis around which the rotation is performed, but rather the rotation type.

By rewriting Equation 2.5 in matricial form, it is possible to introduce the auxiliary matrix $\boldsymbol{B}_{\mathcal{B}}^{-1}$ to obtain the final kinematic relationship between the angle rates and the angular velocity expressed in the body-fixed frame, as shown in Equation 2.6:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \frac{sin(\psi)}{cos(\theta)} & \frac{cos(\psi)}{cos(\theta)} \\ 0 & cos(\psi) & -sin(\psi) \\ 1 & sin(\psi)tan(\theta) & cos(\psi)tan(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \iff \dot{\boldsymbol{\alpha}} = \mathbf{B}_{\mathcal{B}}^{-1} \, {}^{\mathcal{B}}\boldsymbol{\omega} \tag{2.6}$$

It is now convenient to express the angular velocity of the *Chaser* also in the frame $\mathcal{J}$, due to the fact that the derivation of the equations of motion requires that all quantities involved are expressed with respect to the same inertial coordinate system. Therefore, recalling that the matrix $\boldsymbol{A}_{\mathcal{B}/\mathcal{J}}$ allows to express the relative orientation between the two frames, it is possible to obtain the relationship[2] reported in Equation 2.7:

$${}^{\mathcal{J}}\boldsymbol{\omega} = \boldsymbol{A}_{\mathcal{J}/\mathcal{B}} \, {}^{\mathcal{B}}\boldsymbol{\omega} = \boldsymbol{A}_{\mathcal{J}/\mathcal{B}} \, \boldsymbol{B}_{\mathcal{B}} \, \dot{\boldsymbol{\alpha}} \quad \text{where: } \boldsymbol{B}_{\mathcal{B}} = \begin{bmatrix} -sin(\theta) & 0 & 1 \\ cos(\theta)sin(\psi) & cos(\psi) & 0 \\ cos(\theta)cos(\psi) & -sin(\psi) & 0 \end{bmatrix} \tag{2.7}$$

Up to now, only the relative orientation between the two frames has been treated. Therefore, it is necessary to introduce a coordinate transformation that could account also for the relative distance between two different reference frames. To do that, the *homogeneous transformation matrix* reported in Equation 2.8 was defined:

$$\boldsymbol{T}_{\mathcal{B}}^{\mathcal{J}}(\boldsymbol{\alpha}(t)) = \begin{bmatrix} \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}\left(\boldsymbol{\alpha}(t)\right) & {}^{\mathcal{J}}\mathbf{r}_{\mathcal{B}}(t) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^4 \tag{2.8}$$

In particular, $\boldsymbol{A}_{\mathcal{J}/\mathcal{B}}$ represents how the *Chaser*'s body-fixed frame $\mathcal{B}_C$ is oriented with respect to inertial frame $\mathcal{J}$, while ${}^{\mathcal{J}}\mathbf{r}_{\mathcal{B}}$ expresses, in inertial frame coordinates, the location of the origin of $\mathcal{B}_C$, that is nothing but the *Chaser*'s center of mass. Note that both these quantities are implicitly time-dependent, one being related to the time-varying Euler angles and the other to the overall evolution of the dynamics of the system.

2.3. Manipulators Kinematic Model

A robotic manipulator can be thought as a series of rigid bodies (*links*) interconnected with each other through the use of kinematic pairs (*joints*) [18]. These joints introduce,

[2]Note that, being $\boldsymbol{A}_{\mathcal{B}/\mathcal{J}}$ an orthonormal matrix, $\boldsymbol{A}_{\mathcal{J}/\mathcal{B}}$ will be nothing but its transpose.

in general, 1 degree of freedom each and they can be, in most cases, of two different kinds: *revolute*, when they allow only a rotational motion around a certain axis, or *prismatic*, when they allow only linear motion along a particular axis.
In this thesis, a space manipulator composed by two limbs with 6 revolute joints each was considered, and since the kinematic characterization of each limb is identical, only the general case of a servicing-spacecraft equipped with one single manipulator with N-dofs will be discussed.

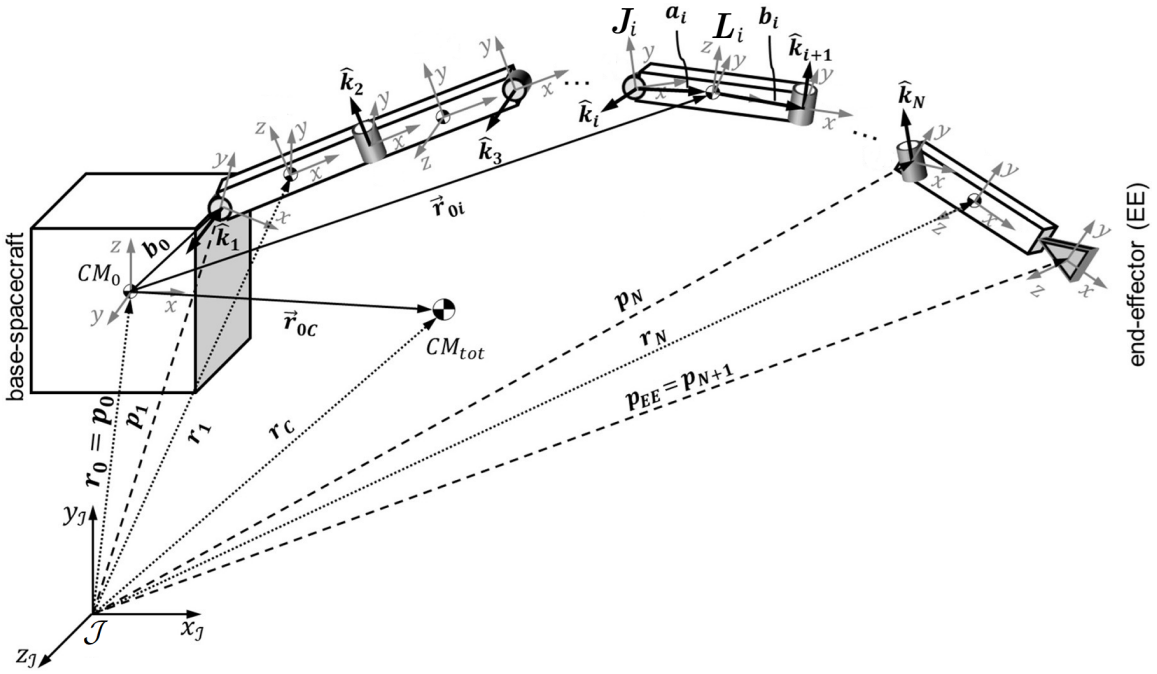

Figure 2.3: Spacecraft-Manipulator system [4]

2.3.1. Direct Kinematics

The direct kinematics of a robotic manipulator involves the calculation of the position and orientation of its end-effector based on a given set of joint parameters. To do that, the *Denavit-Hartenberg* convention, which is a standardized method used to represent the relative transformations between successive links in a manipulator, is presented, following the approach illustrated in [4].

The geometry of a generic servicing-satellite equipped with a single manipulator with N-dofs is reported in Figure 2.3. As it can be seen, three different types of reference frames are introduced in the analysis:

- An inertial frame ($\mathcal{J}$) with respect to which all quantities are eventually referred;
- A set of $N+2$ joint-fixed reference frames J_i, with $0 < i < N + 1$;

- A set of *N+1* link-fixed reference frames L_i, with $0 < i < N$.

The base of the spacecraft is fictitiously defined as link 0: thus, the two coordinates systems J_0 and L_0, that will have their origin at the center of mass of the base, will be arbitrarily oriented. For convenience, they will be chosen to coincide with the already defined body-fixed frame $\mathcal{B}_\mathcal{C}$. The end-effector, instead, is located at the end of the *N-th* link and therefore identified by the fictitious frame J_{N+1}.

Each joint-fixed coordinate system is build, according to the *Denavit-Hartenberg* convention, by following the procedure hereafter reported [18] [4]:

1. The axis $\hat{\boldsymbol{z}}_{J_i}$ is set to be parallel to the *i-th* joint rotation axis $\hat{\boldsymbol{k}}_i$;
2. The origin ϱ_i is located at the intersection of $\hat{\boldsymbol{z}}_{J_i}$ and the common normal between $\hat{\boldsymbol{z}}_{J_{i-1}}$ and $\hat{\boldsymbol{z}}_{J_i}$;
3. The axis $\hat{\boldsymbol{x}}_{J_i}$ is set to be directed as the aforementioned common normal, starting from ϱ_i;
4. The axis $\hat{\boldsymbol{y}}_{J_i}$ simply completes the right-handed frame.

The only exception to this procedure is represented by the frame J_1 (located in correspondence of the first joint of the manipulator): this is because its origin and orientation are usually arbitrarily defined in relation to the specific interface geometry between the base-spacecraft and the manipulator.

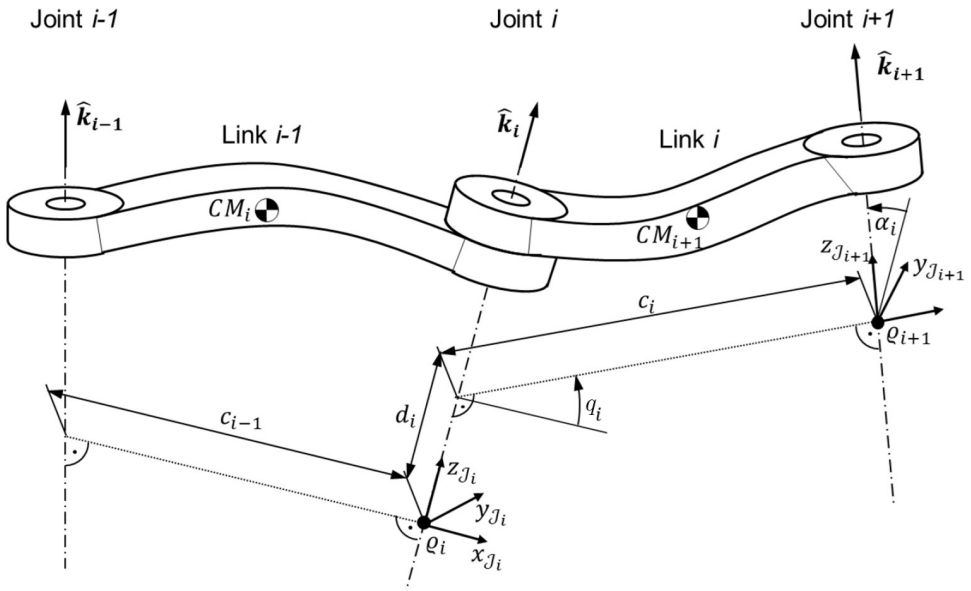

Figure 2.4: Denavit-Hartenberg kinematic parameters [4]

For what concerns instead the definition of the link-fixed frames, it is conventionally

assumed that each L_i is located at the *i-th* link's center of mass and that their orientation matches the one of the joint-fixed frame J_{i+1}.

By analyzing in detail Figure 2.4, it can be seen that the transformation between successive joint-fixed coordinate systems can be described with the introduction of only four geometric variables, called *DH parameters*; these are reported in Table 2.1. Since both manipulators are exclusively constituted by revolute joints, the only variable that will be free to change in time is the *joint angle* q_i, whereas all the others will be fixed geometric parameters arbitrarily defined.

Parameter	Geometrical Interpretation
q_i	Angle of rotation from $\hat{\boldsymbol{x}}_{J_i}$ to $\hat{\boldsymbol{x}}_{J_{i+1}}$ about axis $\hat{\boldsymbol{z}}_{J_i}$
d_i	Distance between ϱ_i and ϱ_{i+1} along $\hat{\boldsymbol{z}}_{J_i}$
α_i	Angle of rotation from $\hat{\boldsymbol{z}}_{J_i}$ to $\hat{\boldsymbol{z}}_{J_{i+1}}$ about axis $\hat{\boldsymbol{x}}_{J_{i+1}}$
c_i	Distance between ϱ_i and ϱ_{i+1} along $\hat{\boldsymbol{x}}_{J_{i+1}}$

Table 2.1: *DH parameters* and their geometrical interpretation [18]

Once these geometric parameters are defined, it is possible to derive the expression of the transformation matrix $\boldsymbol{T}_{J_{i+1}}^{J_i}$ that allows to express the position and orientation of a generic joint's frame J_{i+1} with respect to the previous J_i; this matrix is shown in Equation 2.9 and, as anticipated, it will be a function of the four *DH parameters* only.

$$\begin{aligned}\boldsymbol{T}_{J_{i+1}}^{J_i} &= \boldsymbol{A}(q_i,\, d_i,\, \alpha_i,\, c_i) \\ &= \begin{bmatrix} cos(q_i) & -sin(q_i)cos(\alpha_i) & sin(q_i)sin(\alpha_i) & c_i\, cos(q_i) \\ sin(q_i) & cos(q_i)cos(\alpha_i) & -cos(q_i)sin(\alpha_i) & c_i\, sin(q_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{2.9}$$

It is therefore evident that both the position and orientation, expressed with respect to the inertial frame, of each joint-fixed reference frame can be obtained by recursively multiplying those transformation matrices, as done in Equation 2.10:

$$\boldsymbol{T}_{J_i}^{\mathcal{J}} = \boldsymbol{T}_{J_{i-1}}^{\mathcal{J}}\, \boldsymbol{T}_{J_i}^{J_{i-1}} = \boldsymbol{T}_{J_{i-1}}^{\mathcal{J}}\, \boldsymbol{A}(q_{i-1},\, d_{i-1},\, \alpha_{i-1},\, c_{i-1}) \quad \forall i \in [2, N] \tag{2.10}$$

Hence, the resulting $\boldsymbol{T}_{J_i}^{\mathcal{J}}$ will contain nothing but the rotation matrix $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ and the position

vector ${}^{\mathcal{J}}\boldsymbol{p}_i$ that characterize each joint-fixed coordinate system, as follows:

$$\boldsymbol{T}_{J_i}^{\mathcal{J}} = \begin{bmatrix} \boldsymbol{R}_{J_i}^{\mathcal{J}} & {}^{\mathcal{J}}\boldsymbol{p}_i \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.11}$$

It is now necessary to define the transformation matrix $\boldsymbol{T}_{J_1}^{\mathcal{J}}$ that allows to express the position and orientation of the first joint-fixed frame (J_1) with respect to the inertial coordinate system. As it can be seen in Equation 2.12, this can be obtained using again a recursive multiplication of transformation matrices:

$$\boldsymbol{T}_{J_1}^{\mathcal{J}} = \boldsymbol{T}_{J_0}^{\mathcal{J}}\ \boldsymbol{T}_{J_1}^{J_0} = \boldsymbol{T}_{J_0}^{\mathcal{J}} \begin{bmatrix} \boldsymbol{R}_{J_1}^{J_0}(\eta,\zeta,\epsilon) & {}^{J_0}\boldsymbol{b}_{J_1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.12}$$

where $\boldsymbol{T}_{J_0}^{\mathcal{J}}$ is nothing but the matrix previously defined in Equation 2.8, whereas $\boldsymbol{T}_{J_1}^{J_0}$ defines the fixed-in-time position (${}^{J_0}\boldsymbol{b}_{J_1}$) and orientation ($\boldsymbol{R}_{J_1}^{J_0}$) of J_1 with respect to J_0 [3]. Euler angles in 321 sequence (η, ζ, ϵ) have been chosen to express $\boldsymbol{R}_{J_1}^{J_0}$.

Eventually, the complete direct kinematic chain of the manipulator, from the end-effector to the inertial frame, can be represented as in Equation 2.13:

$$\boldsymbol{T}_{EE}^{\mathcal{J}} = \boldsymbol{T}_{J_0}^{\mathcal{J}}\ \boldsymbol{T}_{J_1}^{J_0} \left(\prod_{i=2}^{N} \boldsymbol{T}_{J_i}^{J_{i-1}} \right) \boldsymbol{T}_{EE}^{J_N} \tag{2.13}$$

2.3.2. Differential Kinematics and Jacobians

The overall geometric configuration of the system can be unequivocally defined with respect to the inertial frame by knowing the position vectors of the joints (${}^{\mathcal{J}}\boldsymbol{p}_i$) and the ones of the links' center of mass (${}^{\mathcal{J}}\boldsymbol{r}_i$).
All these quantities, highlighted in Figure 2.3, can be computed once the homogeneous transformation matrices in Section 2.3.1 are defined. Indeed, the relative position between the base-spacecraft's center of mass and the *i-th* link's center of mass (${}^{\mathcal{J}}\boldsymbol{r}_{0i}$), as well as the position of the whole system's center of mass (${}^{\mathcal{J}}\boldsymbol{r}_C$), can be respectively expressed as in Equations 2.14 and 2.15 [4]. Note that it is now necessary to consider also the presence

[3] Recall that the base-spacecraft was fictitiously considered as the link 0, hence: $J_0 = L_0 = \mathcal{B}_{\mathcal{C}}$.

[4] The notation '*k,i*' is to be interpreted as follows: considering, for example, the quantity ${}^{\mathcal{J}}\boldsymbol{r}_{0i}$ in Figure 2.3, the new variable ${}^{\mathcal{J}}\boldsymbol{r}_{0k,i}$ stands for the distance between the base-spacecraft and the *i-th* link-fixed frame belonging to the *k-th* limb. Analogously, $m_{k,i}$ is the mass of the *i-th* link of the *k-th* limb.

of the second manipulator, as the concept of 'center of mass of the whole system' is introduced.

$$^{\mathcal{J}}\boldsymbol{r}_{0i} = {}^{\mathcal{J}}\boldsymbol{r}_i - {}^{\mathcal{J}}\boldsymbol{r}_0 \tag{2.14}$$

$$\begin{aligned} {}^{\mathcal{J}}\boldsymbol{r}_C &= \frac{1}{m_{tot}} \left(m_0 \, {}^{\mathcal{J}}\boldsymbol{r}_0 + \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \, {}^{\mathcal{J}}\boldsymbol{r}_{k,i} \right) \right) \\ &= \frac{1}{m_{tot}} \left(m_0 \, {}^{\mathcal{J}}\boldsymbol{r}_0 + \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \left({}^{\mathcal{J}}\boldsymbol{r}_0 + {}^{\mathcal{J}}\boldsymbol{r}_{0k,i} \right) \right) \right) \\ &= {}^{\mathcal{J}}\boldsymbol{r}_0 + \frac{1}{m_{tot}} \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \, {}^{\mathcal{J}}\boldsymbol{r}_{0k,i} \right) \end{aligned} \tag{2.15}$$

In Equation 2.15 the value $m_{tot} = m_0 + \sum_{k=1}^{2} \sum_{i=1}^{N} (m_{k,i})$ is the overall mass of the system, which accounts for the spacecraft-base itself and all the links composing the two limbs of the space manipulator.

Furthermore, the position of the whole system center of mass can be expressed, with respect to the base-spacecraft coordinate system, as in Equation 2.16:

$$^{\mathcal{J}}\boldsymbol{r}_{0C} = {}^{\mathcal{J}}\boldsymbol{r}_C - {}^{\mathcal{J}}\boldsymbol{r}_0 \tag{2.16}$$

The latter can be eventually used to compute a useful intermediate relation that will be exploited later on in Section 2.4, whose expression is shown in Equation 2.17:

$$m_{tot} \, {}^{\mathcal{J}}\boldsymbol{r}_{0C} = \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \, {}^{\mathcal{J}}\boldsymbol{r}_{0k,i} \right) \tag{2.17}$$

Starting from the definition of all these values, it was possible to compute the linear velocity of any generic point x_i located on the *i-th* link of the manipulator ($^{\mathcal{J}}\boldsymbol{v}_{x_i}$), as well as the angular velocities ($^{\mathcal{J}}\boldsymbol{\omega}_{L_i}$) of the links themselves. The expressions of these quantities are reported in Equations 2.18 and 2.19: as it can be seen, both of them are composed by a first part which takes into account the motion (linear and/or angular velocity) of the base-spacecraft, while the remaining terms consider the combined contribution of the rotation of all the links up to the generic point x_i.

$$^{\mathcal{J}}\boldsymbol{v}_{x_i} = {}^{\mathcal{J}}\boldsymbol{v}_0 + \left[{}^{\mathcal{J}}\boldsymbol{\omega}_0 \right]^{\times} \left({}^{\mathcal{J}}\boldsymbol{x}_i - {}^{\mathcal{J}}\boldsymbol{r}_0 \right) + \sum_{j=1}^{i} \left\{ \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_j \right]^{\times} \left({}^{\mathcal{J}}\boldsymbol{x}_i - {}^{\mathcal{J}}\boldsymbol{p}_j \right) \dot{q}_j \right\} \tag{2.18}$$

$$ {}^{\mathcal{J}}\boldsymbol{\omega}_{L_i} = {}^{\mathcal{J}}\boldsymbol{\omega}_0 + \sum_{j=1}^{i} \left({}^{\mathcal{J}}\hat{\boldsymbol{k}}_j \, \dot{q}_j \right) \tag{2.19} $$

Note that, in these two equations:

- ${}^{\mathcal{J}}\boldsymbol{v}_0$ and ${}^{\mathcal{J}}\boldsymbol{\omega}_0$ are, respectively, the linear and angular velocity of the base-spacecraft expressed in the inertial frame;
- ${}^{\mathcal{J}}\hat{\boldsymbol{k}}_j = \boldsymbol{R}^{\mathcal{J}}_{J_j} \, [\, 0 \, , \, 0 \, , \, 1 \,]^T$ is the rotation axis of the *j-th* joint with respect to inertial frame;
- the notation $[\boldsymbol{a}]^{\times}$ is used to indicate the skew-symmetric matrix such that the generic cross product $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{c} \;\Leftrightarrow\; [\boldsymbol{a}]^{\times}\boldsymbol{b} = \boldsymbol{c}$, with:

$$ [\boldsymbol{a}]^{\times} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{2.20} $$

- $\dot{q}_j$ is time derivative of the *j-th* joint angle.

By exploiting the definition of the *Translational* and *Rotational Manipulator's Jacobians* ($\boldsymbol{J}_{T_{x_i}}$ and $\boldsymbol{J}_{R_{x_i}}$) and of the *Base Jacobian* ($\boldsymbol{J}_{0_{x_i}}$), the two Equations 2.18 and 2.19 can be condensed in a single relationship, shown in Equation 2.21 [18] [4].
The introduction of those three new variables allows to emphasizes the dependence of the velocity of a generic point x_i of the manipulator (which could be, for example, the end-effector) with respect to the joint rates vector ($\dot{\boldsymbol{q}}$) and the linear and angular velocities of the base-spacecraft.

$$ \begin{bmatrix} {}^{\mathcal{J}}\boldsymbol{v}_{x_i} \\ {}^{\mathcal{J}}\boldsymbol{\omega}_{L_i} \end{bmatrix} = \boldsymbol{J}_{0_{x_i}} \begin{bmatrix} {}^{\mathcal{J}}\boldsymbol{v}_0 \\ {}^{\mathcal{J}}\boldsymbol{\omega}_0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_{T_{x_i}} \\ \boldsymbol{J}_{R_{x_i}} \end{bmatrix} \dot{\boldsymbol{q}} \tag{2.21} $$

Where:

$$ \boldsymbol{J}_{0_{x_i}} = \begin{bmatrix} \mathbb{I}_{3\times3} & -\left[\left({}^{\mathcal{J}}\boldsymbol{x}_i - {}^{\mathcal{J}}\boldsymbol{r}_0 \right) \right]^{\times} \\ \boldsymbol{0}_{3\times3} & \mathbb{I}_{3\times3} \end{bmatrix}_{6\times6} \tag{2.22} $$

$$\boldsymbol{J}_{T_{x_i}} = \left[\left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_1 \right]^{\times} \left({}^{\mathcal{J}}\boldsymbol{x}_i - {}^{\mathcal{J}}\boldsymbol{p}_1 \right) \ , \ldots , \ \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_i \right]^{\times} \left({}^{\mathcal{J}}\boldsymbol{x}_i - {}^{\mathcal{J}}\boldsymbol{p}_i \right) \ , \ \mathbf{0}_{3\times(N-i)} \right]_{3\times N} \quad \forall i \in [1, N] \tag{2.23}$$

$$\boldsymbol{J}_{R_{x_i}} = \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_1 \ , \ldots , {}^{\mathcal{J}}\hat{\boldsymbol{k}}_i \ , \ \mathbf{0}_{3\times(N-i)} \right]_{3\times N} \quad \forall i \in [1, N] \tag{2.24}$$

The presence of the null terms $\mathbf{0}_{3\times(N-i)}$ in Equations 2.23 and 2.24 represents the fact that the linear and angular velocities of the generic *i-th* link are not influenced by the motion of the successive links.

2.3.3. Inverse Kinematics

The inverse kinematics of a robotic manipulator refers to the process of determining the joint angles needed to achieve a specific position and orientation (*pose*) of the end-effector in space. In general, in the case of a 6-dofs manipulator, the calculation of the exact parameters needed to achieve a given configuration in space can involve solving multiple non-linear equations that may not have a unique closed-form analytical solution.
Therefore, a numerical approach was implemented to iteratively minimize the error between the calculated and the desired end-effector pose. This approach uses *DH parameters* to describe the manipulator's kinematics and employs Matlab's `fmincon` optimization algorithm to solve the problem.

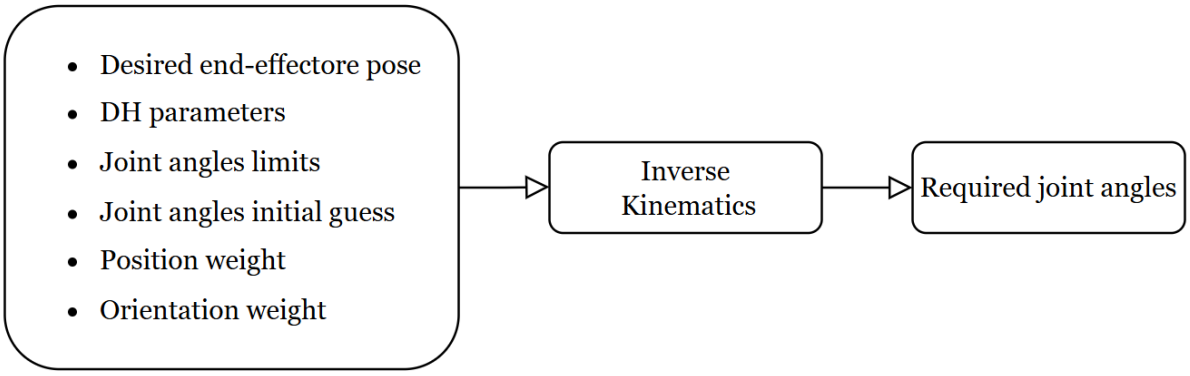

Figure 2.5: Inverse Kinematics

The first operation that the algorithm does is to use the *DH parameters* and the initial joint angles guess, both given as inputs, to compute the current end-effector's pose ($\boldsymbol{T}^{\mathcal{J}}_{EE,cur}$): this is done by simply applying the multiplication chain of transformation matrices shown in Equation 2.13.
Then, from the knowledge of the desired $\boldsymbol{T}^{\mathcal{J}}_{EE,des}$, an objective function that quantifies both the position and orientation errors with respect to $\boldsymbol{T}^{\mathcal{J}}_{EE,cur}$ is defined: its expression

is reported in Equation 2.25:

$$f(\boldsymbol{q}) = w_p \cdot \|\boldsymbol{p}_{cur} - \boldsymbol{p}_{des}\|_2 + w_o \cdot \|\boldsymbol{R}_{des} - \boldsymbol{R}_{cur}\|_F \tag{2.25}$$

where:

- $\boldsymbol{q}$ is the vector containing the joint angles;
- $\boldsymbol{p}_{cur}$ and $\boldsymbol{p}_{des}$ are the current and desired end-effector's position vectors, extracted respectively from $\boldsymbol{T}^{\mathcal{J}}_{EE,cur}$ and $\boldsymbol{T}^{\mathcal{J}}_{EE,des}$;
- $\boldsymbol{R}_{cur}$ and $\boldsymbol{R}_{des}$ are the current and desired end-effector's rotation matrices, again extracted respectively from $\boldsymbol{T}^{\mathcal{J}}_{EE,cur}$ and $\boldsymbol{T}^{\mathcal{J}}_{EE,des}$;
- $\|\cdot\|_2$ respresents the *euclidean norm*;
- $\|\cdot\|_F$ represents the *Frobenius norm*;
- w_p and w_o are two arbitrary scalar weights.

Eventually, the minimization problem has been formulated as below:

$$\begin{aligned} \min_{\boldsymbol{q}} f(\boldsymbol{q}) = \quad & \|\text{position error}\| \cdot w_p + \|\text{orientation error}\| \cdot w_o \\ \text{subject to:} \quad & \mathbf{LB} < \boldsymbol{q} < \mathbf{UB} \\ \text{where:} \quad & \boldsymbol{q} \in \mathbb{R}^6 \end{aligned} \tag{2.26}$$

The Matlab's optimization function `fmincon`, configured with the Sequential Quadratic Programming (SQP) algorithm, was selected to minimize this error while also ensuring that the resulting joint angles vector was compliant with the upper and lower joint's physical boundaries (**UB** and **LB**, respectively).

2.4. Multibody Dynamics Model

2.4.1. Equations of Motion

The *Lagrangian approach* was eventually chosen as the method to retrieve the fully coupled equations of motion of the multybody system under analysis, which is composed by the base-spacecraft and the multi limb space manipulator.
By recalling the simplifying assumptions made in Section 1.4, the gravitational potential energy of the system can be neglected: indeed, for a free-flying satellite, the kinetic energy (due to translational and rotational motion of both base and manipulator) dominates the

dynamics, especially in microgravity or when gravitational gradients are weak. Moreover, due to the rigid-bodies assumption, the additional contribution represented by the elastic potential energy is not accounted. Therefore, the Lagrangian simplifies to Equation 2.27:

$$\begin{aligned} L = T = \frac{1}{2}\left({}^{\mathcal{J}}\boldsymbol{\omega}_0{}^T \; {}^{\mathcal{J}}\boldsymbol{I}_0 \; {}^{\mathcal{J}}\boldsymbol{\omega}_0 + m_0 \; {}^{\mathcal{J}}\boldsymbol{v}_0{}^T \; {}^{\mathcal{J}}\boldsymbol{v}_0\right) \; + \\ + \frac{1}{2}\sum_{k=1}^{2}\sum_{i=1}^{N}\left({}^{\mathcal{J}}\boldsymbol{\omega}_{k,i}{}^T \; {}^{\mathcal{J}}\boldsymbol{I}_{k,i} \; {}^{\mathcal{J}}\boldsymbol{\omega}_{k,i} + m_{k,i} \; {}^{\mathcal{J}}\dot{\boldsymbol{r}}_{k,i}{}^T \; {}^{\mathcal{J}}\dot{\boldsymbol{r}}_{k,i}\right) \end{aligned} \tag{2.27}$$

Note that the quantity ${}^{\mathcal{J}}\boldsymbol{I}_{k,i}$ is the matrix containing the moments of inertia, expressed relative to the inertial frame, of the *i-th* link of the *k-th* limb of the manipulator; this can be calculated, for each link, starting from the known moment of inertia matrix expressed with respect to their link-fixed coordinate system, as done in Equation 2.28:

$${}^{\mathcal{J}}\boldsymbol{I}_i = \boldsymbol{R}_{L_i}^{\mathcal{J}} \; {}^{L_i}\boldsymbol{I}_i \; \boldsymbol{R}_{L_i}^{\mathcal{J}^T} \tag{2.28}$$

where the rotation matrix $\boldsymbol{R}_{L_i}^{\mathcal{J}}$ is extracted from:

$$\boldsymbol{T}_{L_i}^{\mathcal{J}} = \boldsymbol{T}_{J_i}^{\mathcal{J}} \boldsymbol{T}_{L_i}^{J_i} = \boldsymbol{T}_{J_i}^{\mathcal{J}} \boldsymbol{A}(q_i,\, d_i,\, \alpha_i,\, c_i/2) \quad \forall i \in [1, N] \tag{2.29}$$

By choosing as generalized coordinates of the system the variables grouped in the vector $\boldsymbol{\xi}$ (shown in Equation 2.30), that are the three linear position components of the base-spacecraft, the attitude parameters of the base-spacecraft and the joint angles of the two limbs (denoted with subscripts A and B, respectively):

$$\begin{aligned} \boldsymbol{\xi} &= [\,{}^{\mathcal{J}}\boldsymbol{r}_0{}^T \,,\, \varphi \,,\, \theta \,,\, \psi \,,\, \boldsymbol{q}_A \,,\, \boldsymbol{q}_B \,] \;\in\; \mathbb{R}^{(6+2N)\times 1} \\ \dot{\boldsymbol{\xi}} &= [\,{}^{\mathcal{J}}\dot{\boldsymbol{r}}_0{}^T \,,\, \dot{\varphi} \,,\, \dot{\theta} \,,\, \dot{\psi} \,,\, \dot{\boldsymbol{q}}_A \,,\, \dot{\boldsymbol{q}}_B \,] \;\in\; \mathbb{R}^{(6+2N)\times 1} \end{aligned} \tag{2.30}$$

the kinetic energy can be written, in matricial form, as in Equation 2.31.

$$T = \frac{1}{2}\dot{\boldsymbol{\xi}}^T \boldsymbol{H}(\boldsymbol{\xi})\, \dot{\boldsymbol{\xi}} \tag{2.31}$$

Note that $\boldsymbol{H}(\boldsymbol{\xi})$ is the system's overall inertia matrix, whose expression is reported in Equation 2.32:

$$\boldsymbol{H}(\boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{H}_0 & \boldsymbol{H}_{0m}^A & \boldsymbol{H}_{0m}^B \\ \boldsymbol{H}_{0m}^{A^T} & \boldsymbol{H}_m^A & \boldsymbol{0}_{N\times N} \\ \boldsymbol{H}_{0m}^{B^T} & \boldsymbol{0}_{N\times N} & \boldsymbol{H}_m^B \end{bmatrix} \tag{2.32}$$

The sub-matrices $\boldsymbol{H}_0$, $\boldsymbol{H}_{0m}^k$ and $\boldsymbol{H}_m^k$ represent, respectively, the $[6 \times 6]$ base-spacecraft inertia matrix, the $[6 \times N]$ dynamic coupling inertia matrix between the base and the *k-th* limb and the $[N \times N]$ inertia matrix of the *k-th* limb. All these quantities are computed as described in Section 2.4.2.

Recalling now the generalized coordinates chosen to describe the dynamics of the multi-body system (Equation 2.30) and the definition of the system's overall kinetic energy (Equation 2.31), the equations of motion can be formulated by firstly calculating the derivatives[5] shown in Equations 2.33 to 2.35:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{x}}_0}\right) - \frac{\partial T}{\partial \boldsymbol{x}_0} = \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ \boldsymbol{\tau}_0 \end{bmatrix} \quad \text{where:} \quad \boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{\alpha} \end{bmatrix} \tag{2.33}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}_A}\right) - \frac{\partial T}{\partial \boldsymbol{q}_A} = \boldsymbol{\tau}_A \tag{2.34}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}_B}\right) - \frac{\partial T}{\partial \boldsymbol{q}_B} = \boldsymbol{\tau}_B \tag{2.35}$$

Eventually, after properly rearranging the terms, the $6+N+N$ fully coupled equations of motion of the system can be obtained as below:

$$\begin{bmatrix} \boldsymbol{H}_0 & \boldsymbol{H}_{0m}^A & \boldsymbol{H}_{0m}^B \\ \boldsymbol{H}_{0m}^{A^T} & \boldsymbol{H}_m^A & \boldsymbol{0}_{N\times N} \\ \boldsymbol{H}_{0m}^{B^T} & \boldsymbol{0}_{N\times N} & \boldsymbol{H}_m^B \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}}_0 \\ \ddot{\boldsymbol{q}}_A \\ \ddot{\boldsymbol{q}}_B \end{bmatrix} + \begin{bmatrix} \dot{\boldsymbol{H}}_0 & \dot{\boldsymbol{H}}_{0m}^A & \dot{\boldsymbol{H}}_{0m}^B \\ \dot{\boldsymbol{H}}_{0m}^{A^T} & \dot{\boldsymbol{H}}_m^A & \boldsymbol{0}_{N\times N} \\ \dot{\boldsymbol{H}}_{0m}^{B^T} & \boldsymbol{0}_{N\times N} & \dot{\boldsymbol{H}}_m^B \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}_0 \\ \dot{\boldsymbol{q}}_A \\ \dot{\boldsymbol{q}}_B \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_A \\ \boldsymbol{c}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ \boldsymbol{\tau}_0 \\ \boldsymbol{\tau}_A \\ \boldsymbol{\tau}_B \end{bmatrix} \tag{2.36}$$

where:

$$\boldsymbol{c}_0 = -\frac{\partial T}{\partial \boldsymbol{x}_0} \qquad \boldsymbol{c}_A = -\frac{\partial T}{\partial \boldsymbol{q}_A} \qquad \boldsymbol{c}_B = -\frac{\partial T}{\partial \boldsymbol{q}_B} \tag{2.37}$$

The right hand side of Equation 2.36 is the vector containing the actuation torques: $\boldsymbol{\tau}_0$ is referred to the attitude control torques of the base-spacecraft, while $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_B$ are the $[N \times 1]$ vectors collecting the actuation torques of each joint of the two limbs.
The computation of the all the partial derivatives of the different inertia sub-matrices introduced in previous equations is analyzed in detail in the following sections.

[5]Note that, since all the generalized coordinates are expressed with respect to the inertial frame, the superscript $\mathcal{J}$ will be dropped for the sake of notation simplicity.

2.4.2. System Inertia Matrix

The $[6 \times 6]$ base-spacecraft inertia matrix is calculated as in Equation 2.38:

$$\boldsymbol{H}_0 = \begin{bmatrix} m_{tot}\, \mathbb{I}_{3\times 3} & -m_{tot} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0C}\right]^{\times} \boldsymbol{B}_{\mathcal{N}} \\ m_{tot}\, \boldsymbol{B}_{\mathcal{N}}^{T} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0C}\right]^{\times} & \boldsymbol{B}_{\mathcal{N}}^{T}\, \boldsymbol{H}_s\, \boldsymbol{B}_{\mathcal{N}} \end{bmatrix} \tag{2.38}$$

where:

- $\boldsymbol{B}_{\mathcal{N}} = \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}\, \boldsymbol{B}_{\mathcal{B}}$, recalling Equation 2.7;
- The $[3 \times 3]$ sub-matrix $\boldsymbol{H}_s$ collects the contributions of the base-spacecraft's and manipulator's moments of inertia about the center of mass of the base. Its expression is shown in Equation 2.39:

$$\boldsymbol{H}_s = {}^{\mathcal{J}}\boldsymbol{I}_0 + \sum_{k=1}^{2} \sum_{i=1}^{N} \left({}^{\mathcal{J}}\boldsymbol{I}_{k,i} - m_{k,i} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times} \right) \tag{2.39}$$

The $[6 \times N]$ dynamic coupling inertia matrix between the *k-th* limb and the base-spacecraft is calculated as in Equation 2.40:

$$\boldsymbol{H}_{0m} = \begin{bmatrix} \boldsymbol{J}_{TS} \\ \boldsymbol{B}_{\mathcal{N}}^{T}\, \boldsymbol{H}_{sq} \end{bmatrix} \quad \text{where:} \quad \begin{cases} \boldsymbol{J}_{TS} = \sum_{i=1}^{N} \left(m_i\, \boldsymbol{J}_{T_i} \right) \\ \boldsymbol{H}_{sq} = \sum_{i=1}^{N} \left({}^{\mathcal{J}}\boldsymbol{I}_i \boldsymbol{J}_{R_i} + m_i \left[{}^{\mathcal{J}}\boldsymbol{r}_{0i}\right]^{\times} \boldsymbol{J}_{T_i} \right) \end{cases} \tag{2.40}$$

In particular, $\boldsymbol{J}_{TS}$ and $\boldsymbol{H}_{sq}$ are two $[3 \times N]$ matrices that collect the contribution, to the system's kinetic energy, of the manipulator's joint rates and the base-spacecraft linear velocity and of the manipulator's joint rates and the base-spacecraft angular velocity, respectively. Note that, in their expressions, the values $\boldsymbol{J}_{T_i}$ and $\boldsymbol{J}_{R_i}$ are nothing but the *Jacobians* already described in Equations 2.23 and 2.24, now referred to the center of mass of the *i-th* link instead of to a generic point x_i of the manipulator.

Finally, the $[N \times N]$ inertia matrix of the *k-th* limb is expressed as in Equation 2.41:

$$\boldsymbol{H}_m = \sum_{i=1}^{N} \left(\boldsymbol{J}_{R_i}^{T}\, {}^{\mathcal{J}}\boldsymbol{I}_i\, \boldsymbol{J}_{R_i} + m_i \boldsymbol{J}_{T_i}^{T} \boldsymbol{J}_{T_i} \right) \tag{2.41}$$

2.4.3. Time Derivative of the System Inertia Matrix

First of all, the value of the matrix $\dot{\boldsymbol{H}}_m$, associated to the *k-th* limb, is reported in Equation 2.42:

$$\dot{\boldsymbol{H}}_m = \sum_{i=1}^{N} \left(\boldsymbol{J}_{R_i}^T \, {}^{\mathcal{J}}\boldsymbol{I}_i \, \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T \, {}^{\mathcal{J}}\dot{\boldsymbol{I}}_i \, \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T \, {}^{\mathcal{J}}\boldsymbol{I}_i \, \dot{\boldsymbol{J}}_{R_i} + m_i \left(\dot{\boldsymbol{J}}_{T_i}^T \boldsymbol{J}_{T_i} + \boldsymbol{J}_{T_i}^T \dot{\boldsymbol{J}}_{T_i} \right) \right) \tag{2.42}$$

For the computation of $\dot{\boldsymbol{J}}_{T_i}$, $\dot{\boldsymbol{J}}_{R_i}$ and ${}^{\mathcal{J}}\dot{\boldsymbol{I}}_i$, the time derivative of the rotation matrices $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{R}_{L_i}^{\mathcal{J}}$, associated respectively to the *i-th* joint-fixed and link-fixed frames, is required. By recalling that all values are referred with respect to the inertial frame, the Darboux equation can be used to compute the time derivative of the generic $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ [4]:

$$\dot{\boldsymbol{R}}_{J_i}^{\mathcal{J}} = \left[{}^{\mathcal{J}}\boldsymbol{\omega}_{J_i} \right]^{\times} \boldsymbol{R}_{J_i}^{\mathcal{J}} \tag{2.43}$$

where ${}^{\mathcal{J}}\boldsymbol{\omega}_{J_i}$ is the angular velocity of the *i-th* joint with respect to inertial frame. This value can be computed recursively $\forall i \in [1, N+1]$ as done in Equation 2.44:

$${}^{\mathcal{J}}\boldsymbol{\omega}_{J_i} = {}^{\mathcal{J}}\boldsymbol{\omega}_{J_{i-1}} + {}^{J_{i-1}}\boldsymbol{\omega}_{J_i} = {}^{\mathcal{J}}\boldsymbol{\omega}_{J_{i-1}} + {}^{\mathcal{J}}\hat{\boldsymbol{k}}_{J_{i-1}} \, \dot{q}_{i-1} \tag{2.44}$$

Then, recalling the definition of $\boldsymbol{R}_{L_i}^{\mathcal{J}}$ in Equation 2.29, its time derivative will be nothing but:

$$\dot{\boldsymbol{R}}_{L_i}^{\mathcal{J}} = \dot{\boldsymbol{R}}_{J_{i+1}}^{\mathcal{J}} \quad \forall i \in [1, N] \tag{2.45}$$

Therefore, by exploiting Equation 2.45, the value of ${}^{\mathcal{J}}\dot{\boldsymbol{I}}_i$ can be computed as:

$${}^{\mathcal{J}}\dot{\boldsymbol{I}}_i = \dot{\boldsymbol{R}}_{L_i}^{\mathcal{J}} \, {}^{L_i}\boldsymbol{I}_i \, \boldsymbol{R}_{L_i}^{\mathcal{J}^T} + \boldsymbol{R}_{L_i}^{\mathcal{J}} \, {}^{L_i}\boldsymbol{I}_i \, \dot{\boldsymbol{R}}_{L_i}^{\mathcal{J}^T} \quad \forall i \in [1, N] \tag{2.46}$$

For what concerns instead the time derivatives of the two *Jacobians*, the value of $\dot{\boldsymbol{J}}_{R_i}$ can be simply calculated as shown in Equation 2.47:

$$\dot{\boldsymbol{J}}_{R_i} = \begin{bmatrix} \dot{\boldsymbol{R}}_{J_1}^{\mathcal{J}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & , \dots , & \dot{\boldsymbol{R}}_{J_i}^{\mathcal{J}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & , \; \boldsymbol{0}_{3\times(N-i)} \end{bmatrix} \quad \forall i \in [1, N] \tag{2.47}$$

while the value of $\dot{\boldsymbol{J}}_{T_i}$ can be calculated as:

$$\dot{\boldsymbol{J}}_{T_i} = \left[\dot{\boldsymbol{j}}_{T_{i_1}} \ , \ \dots \ , \ \dot{\boldsymbol{j}}_{T_{i_m}} \ , \ \boldsymbol{0}_{3\times(N-i)}\right] \quad \forall i \in [1, N] \tag{2.48}$$

where $\dot{\boldsymbol{j}}_{T_{i_m}}$ is the time derivative of the *m-th* column of the *i-th* $\boldsymbol{J}_{T_i}$. Therefore, recalling the definition of the latter in Equation 2.23, the quantity $\dot{\boldsymbol{j}}_{T_{i_m}}$ can be finally computed as below:

$$\dot{\boldsymbol{j}}_{T_{i_m}} = \left[\dot{\boldsymbol{R}}^{\mathcal{J}}_{J_m} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right]^{\times} (\boldsymbol{r}_i - \boldsymbol{p}_m) + \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_m\right]^{\times} (\dot{\boldsymbol{r}}_i - \dot{\boldsymbol{p}}_m) \quad \forall m \leq i \tag{2.49}$$

Then, the value of the matrix $\dot{\boldsymbol{H}}_0$ is reported in Equation 2.50:

$$\dot{\boldsymbol{H}}_0 = \begin{bmatrix} \boldsymbol{0}_{3\times3} & -m_{tot}\left(\left[{}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0C}\right]^{\times}\boldsymbol{B}_{\mathcal{N}} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0C}\right]^{\times}\dot{\boldsymbol{B}}_{\mathcal{N}}\right) \\ m_{tot}\left(\dot{\boldsymbol{B}}^T_{\mathcal{N}}\left[{}^{\mathcal{J}}\boldsymbol{r}_{0C}\right]^{\times} + \boldsymbol{B}^T_{\mathcal{N}}\left[{}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0C}\right]^{\times}\right) & \dot{\boldsymbol{B}}^T_{\mathcal{N}}\boldsymbol{H}_s\boldsymbol{B}_{\mathcal{N}} + \boldsymbol{B}^T_{\mathcal{N}}\dot{\boldsymbol{H}}_s\boldsymbol{B}_{\mathcal{N}} + \boldsymbol{B}^T_{\mathcal{N}}\boldsymbol{H}_s\dot{\boldsymbol{B}}_{\mathcal{N}} \end{bmatrix} \tag{2.50}$$

where the values of $\dot{\boldsymbol{H}}_s$ and ${}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0C}$ are respectively computed as follows:

$$\dot{\boldsymbol{H}}_s = {}^{\mathcal{J}}\dot{\boldsymbol{I}}_0 + \sum_{k=1}^{2}\sum_{i=1}^{N}\left({}^{\mathcal{J}}\dot{\boldsymbol{I}}_{k,i} - m_{k,i}\left(\left[{}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0k,i}\right]^{\times}\left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times}\left[{}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0k,i}\right]^{\times}\right)\right) \tag{2.51}$$

$${}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0C} = \sum_{k=1}^{2}\sum_{i=1}^{N}\left(m_{k,i}\left(\left[\boldsymbol{\omega}_0\right]^{\times}\left({}^{\mathcal{J}}\boldsymbol{r}_{k,i} - {}^{\mathcal{J}}\boldsymbol{r}_0\right) + \sum_{j=1}^{i}\left(\left[\left[\boldsymbol{R}^{\mathcal{J}}_{J_{k,j}}\begin{bmatrix}0\\0\\1\end{bmatrix}\right]^{\times}(\boldsymbol{r}_{k,i} - \boldsymbol{p}_{k,j})\right]\dot{q}_{k,j}\right)\right)\right) \tag{2.52}$$

For what concerns instead the computation of $\dot{\boldsymbol{B}}_{\mathcal{N}}$, it yields:

$$\dot{\boldsymbol{B}}_{\mathcal{N}} = \dot{\boldsymbol{A}}_{\mathcal{J}/\mathcal{B}}\,\boldsymbol{B}_{\mathcal{B}} + \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}\,\dot{\boldsymbol{B}}_{\mathcal{B}} \quad \text{where:} \begin{cases} \dot{\boldsymbol{A}}_{\mathcal{J}/\mathcal{B}} = \left[{}^{\mathcal{J}}\boldsymbol{\omega}_0\right]^{\times}\boldsymbol{A}_{\mathcal{J}/\mathcal{B}} \\ \\ \dot{\boldsymbol{B}}_{\mathcal{B}} = \dfrac{\partial \boldsymbol{B}_{\mathcal{B}}}{\partial\varphi}\dot{\varphi} + \dfrac{\partial \boldsymbol{B}_{\mathcal{B}}}{\partial\theta}\dot{\theta} + \dfrac{\partial \boldsymbol{B}_{\mathcal{B}}}{\partial\psi}\dot{\psi} \end{cases} \tag{2.53}$$

Lastly, the value of $\dot{\boldsymbol{H}}_{0m}$, associated to the *k-th* limb, is reported in Equation 2.54:

$$\dot{\boldsymbol{H}}_{0m} = \begin{bmatrix} \dot{\boldsymbol{J}}_{TS} \\ \dot{\boldsymbol{B}}_{\mathcal{N}}^{T}\dot{\boldsymbol{H}}_{sq} + \boldsymbol{B}_{\mathcal{N}}^{T}\dot{\boldsymbol{H}}_{sq} \end{bmatrix} \tag{2.54}$$

where the values of $\dot{\boldsymbol{J}}_{TS}$ and $\dot{\boldsymbol{H}}_{sq}$ are respectively calculated as follows:

$$\dot{\boldsymbol{J}}_{TS} = \sum_{i=1}^{N} \left(m_i \, \dot{\boldsymbol{J}}_{T_i} \right) \tag{2.55}$$

$$\dot{\boldsymbol{H}}_{sq} = \sum_{i=1}^{N} \left({}^{\mathcal{J}}\dot{\boldsymbol{I}}_i \boldsymbol{J}_{R_i} + {}^{\mathcal{J}}\boldsymbol{I}_i \dot{\boldsymbol{J}}_{R_i} + m_i \left(\left[{}^{\mathcal{J}}\dot{\boldsymbol{r}}_{0i} \right]^{\times} \boldsymbol{J}_{T_i} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0i} \right]^{\times} \dot{\boldsymbol{J}}_{T_i} \right) \right) \tag{2.56}$$

2.4.4. Euler Angles Derivative of the System Inertia Matrix

In principle, to compute the parameter $\boldsymbol{c}_0$ defined in Equation 2.37, it would be necessary to calculate the derivative of the system inertia matrix with respect to both the base-spacecraft's position and the Euler angles chosen to parametrize its attitude.
Nevertheless, the system inertia matrix does not depend on the base-spacecraft's position: indeed, since this matrix is determined by the mass and inertia distribution relative to the satellite's center of mass, a translation of the satellite in the inertial frame would not alter its value. Therefore, only the Euler angles derivative of the system inertia matrix will be analyzed in this section.

The first step requires the calculation of the Euler angles derivatives of the homogeneous transformation matrices $\boldsymbol{T}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{T}_{L_i}^{\mathcal{J}}$ associated to the generic *k-th* limb of the manipulator. Starting from these values, it would then be possible to simply extract the derivatives of the rotation matrices $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{R}_{L_i}^{\mathcal{J}}$, as well as the ones of the position vectors ${}^{\mathcal{J}}\boldsymbol{p}_i$ and ${}^{\mathcal{J}}\boldsymbol{r}_i$, all needed to eventually calculate the derivatives of the two *Jacobians* and of the matrices containing the moments of inertia of each link.
Recalling the direct kinematic chain in Equation 2.13, the derivatives of $\boldsymbol{T}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{T}_{L_i}^{\mathcal{J}}$ with respect to the generic *n-th* Euler angle will be:

$$\frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial \alpha_n} = \frac{\partial \boldsymbol{T}_{J_0}^{\mathcal{J}}}{\partial \alpha_n} \boldsymbol{T}_{J_i}^{J_0} \quad \Longrightarrow \quad \frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial \alpha_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{R}_{J_i}^{\mathcal{J}}}{\partial \alpha_n} & \dfrac{\partial \, {}^{\mathcal{J}}\boldsymbol{p}_i}{\partial \alpha_n} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.57}$$

$$\frac{\partial \boldsymbol{T}_{L_i}^{\mathcal{J}}}{\partial \alpha_n} = \frac{\partial \boldsymbol{T}_{J_0}^{\mathcal{J}}}{\partial \alpha_n} \boldsymbol{T}_{L_i}^{J_0} \quad \Longrightarrow \quad \frac{\partial \boldsymbol{T}_{L_i}^{\mathcal{J}}}{\partial \alpha_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{R}_{L_i}^{\mathcal{J}}}{\partial \alpha_n} & \dfrac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_i}{\partial \alpha_n} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.58}$$

From Equations 2.57 and 2.58 it can be noted that only $\boldsymbol{T}_{J_0}^{\mathcal{J}}$ will explicitly depend on the three Euler angles, since the transformation matrices relating the relative position and orientation of each joint/link-fixed frame to each other will be only function of the *DH parameters*. In particular, its derivative with respect to the generic *n-th* Euler angle can be computed as:

$$\frac{\partial \boldsymbol{T}_{J_0}^{\mathcal{J}}}{\partial \alpha_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}}{\partial \alpha_n} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.59}$$

where, recalling Equation 2.1:

$$\begin{cases} \dfrac{\partial \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}}{\partial \varphi} = \dfrac{\partial \boldsymbol{R}_z(\varphi)}{\partial \varphi}\, \boldsymbol{R}_y(\theta)\, \boldsymbol{R}_x(\psi) \\ \dfrac{\partial \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}}{\partial \theta} = \boldsymbol{R}_z(\varphi)\, \dfrac{\partial \boldsymbol{R}_y(\theta)}{\partial \theta}\, \boldsymbol{R}_x(\psi) \\ \dfrac{\partial \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}}{\partial \psi} = \boldsymbol{R}_z(\varphi)\, \boldsymbol{R}_y(\theta)\, \dfrac{\partial \boldsymbol{R}_x(\psi)}{\partial \psi} \end{cases} \tag{2.60}$$

As anticipated before, the Euler angles derivatives of the two *Jacobians* can be finally calculated as in Equations 2.61 to 2.63:

$$\frac{\partial \boldsymbol{J}_{R_i}}{\partial \alpha_n} = \left[\frac{\partial \boldsymbol{R}_{J_1}^{\mathcal{J}}}{\partial \alpha_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \dots , \frac{\partial \boldsymbol{R}_{J_i}^{\mathcal{J}}}{\partial \alpha_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \mathbf{0}_{3\times(N-i)} \right] \tag{2.61}$$

$$\frac{\partial \boldsymbol{J}_{T_i}}{\partial \alpha_n} = \left[\frac{\partial \boldsymbol{\jmath}_{T_{i_1}}}{\partial \alpha_n} , \dots , \frac{\partial \boldsymbol{\jmath}_{T_{i_m}}}{\partial \alpha_n} , \mathbf{0}_{3\times(N-i)} \right] \tag{2.62}$$

with:

$$\frac{\partial \boldsymbol{\jmath}_{T_{im}}}{\partial \alpha_n} = \left[\frac{\partial \boldsymbol{R}^{\mathcal{J}}_{J_m}}{\partial \alpha_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]^{\times} (\boldsymbol{r}_i - \boldsymbol{p}_m) + \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_m \right]^{\times} \left(\frac{\partial \boldsymbol{r}_i}{\partial \alpha_n} - \frac{\partial \boldsymbol{p}_m}{\partial \alpha_n} \right) \quad \forall m \leq i \tag{2.63}$$

Then, similarly to what was done in Equation 2.46, the derivative of each link's moment of inertia matrix with respect to the *n-th* Euler angle can be computed as:

$$\frac{\partial {}^{\mathcal{J}}\boldsymbol{I}_i}{\partial \alpha_n} = \frac{\partial \boldsymbol{R}^{\mathcal{J}}_{L_i}}{\partial \alpha_n} {}^{L_i}\boldsymbol{I}_i \, \boldsymbol{R}^{\mathcal{J}^T}_{L_i} + \boldsymbol{R}^{\mathcal{J}}_{L_i} {}^{L_i}\boldsymbol{I}_i \frac{\partial \boldsymbol{R}^{\mathcal{J}^T}_{L_i}}{\partial \alpha_n} \tag{2.64}$$

Lastly, also the derivative of the matrix $\boldsymbol{B}_{\mathcal{N}}$ is required to be able to construct the final Euler angles derivative of whole system inertia matrix; this is simply calculated as shown in Equation 2.65 by recalling the two Equations 2.7 and 2.60.

$$\frac{\partial \boldsymbol{B}_{\mathcal{N}}}{\partial \alpha_n} = \frac{\partial \boldsymbol{A}_{\mathcal{J}/\mathcal{B}}}{\partial \alpha_n} \boldsymbol{B}_{\mathcal{B}} + \boldsymbol{A}_{\mathcal{J}/\mathcal{B}} \frac{\partial \boldsymbol{B}_{\mathcal{B}}}{\partial \alpha_n} \tag{2.65}$$

It is now possible to assemble the derivatives of all the different inertia sub-matrices; starting from $\boldsymbol{H}_0$, it yields:

$$\frac{\partial \boldsymbol{H}_0}{\partial \alpha_n} = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & -m_{tot} \left(\left[\frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial \alpha_n} \right]^{\times} \boldsymbol{B}_{\mathcal{N}} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0C} \right]^{\times} \frac{\partial \boldsymbol{B}_{\mathcal{N}}}{\partial \alpha_n} \right) \\ m_{tot} \left(\frac{\partial \boldsymbol{B}^T_{\mathcal{N}}}{\partial \alpha_n} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0C} \right]^{\times} + \boldsymbol{B}^T_{\mathcal{N}} \left[\frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial \alpha_n} \right]^{\times} \right) & \frac{\partial \boldsymbol{B}^T_{\mathcal{N}}}{\partial \alpha_n} \boldsymbol{H}_s \boldsymbol{B}_{\mathcal{N}} + \boldsymbol{B}^T_{\mathcal{N}} \frac{\partial \boldsymbol{H}_s}{\partial \alpha_n} \boldsymbol{B}_{\mathcal{N}} + \boldsymbol{B}^T_{\mathcal{N}} \boldsymbol{H}_s \frac{\partial \boldsymbol{B}_{\mathcal{N}}}{\partial \alpha_n} \end{bmatrix} \tag{2.66}$$

where:

$$\frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial \alpha_n} = \frac{1}{m_{tot}} \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial \alpha_n} \right) \tag{2.67}$$

$$\frac{\partial \boldsymbol{H}_s}{\partial \alpha_n} = \frac{\partial {}^{\mathcal{J}}\boldsymbol{I}_0}{\partial \alpha_n} + \sum_{k=1}^{2} \sum_{i=1}^{N} \left(\frac{\partial {}^{\mathcal{J}}\boldsymbol{I}_{k,i}}{\partial \alpha_n} - m_{k,i} \left(\left[\frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial \alpha_n} \right]^{\times} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i} \right]^{\times} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i} \right]^{\times} \left[\frac{\partial {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial \alpha_n} \right]^{\times} \right) \right) \tag{2.68}$$

Note that in Equations 2.67 and 2.68 it has been exploited the fact that the position vector

of the base-spacecraft ($^{\mathcal{J}}\boldsymbol{r}_0$) is not function of any Euler angle, leading to the following simplification:

$$\frac{\partial\, ^{\mathcal{J}}\boldsymbol{r}_{0i}}{\partial \alpha_n} = \frac{\partial\, ^{\mathcal{J}}\boldsymbol{r}_i}{\partial \alpha_n} - \cancel{\frac{\partial\, ^{\mathcal{J}}\boldsymbol{r}_0}{\partial \alpha_n}} \tag{2.69}$$

Then, the Euler angles derivative of the matrix $\boldsymbol{H}_{0m}$, associated to the *k-th* limb, can be expressed as:

$$\frac{\partial \boldsymbol{H}_{0m}}{\partial \alpha_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{J}_{TS}}{\partial \alpha_n} \\ \dfrac{\partial \boldsymbol{B}_{\mathcal{N}}^T}{\partial \alpha_n}\boldsymbol{H}_{sq} + \boldsymbol{B}_{\mathcal{N}}^T \dfrac{\partial \boldsymbol{H}_{sq}}{\partial \alpha_n} \end{bmatrix} \tag{2.70}$$

where:

$$\frac{\partial \boldsymbol{J}_{TS}}{\partial \alpha_n} = \sum_{i=1}^{N} \left(m_i \frac{\partial \boldsymbol{J}_{T_i}}{\partial \alpha_n} \right) \tag{2.71}$$

$$\frac{\partial \boldsymbol{H}_{sq}}{\partial \alpha_n} = \sum_{i=1}^{N} \left(\frac{\partial\, ^{\mathcal{J}}\boldsymbol{I}_i}{\partial \alpha_n}\boldsymbol{J}_{R_i} + {}^{\mathcal{J}}\boldsymbol{I}_i \frac{\partial \boldsymbol{J}_{R_i}}{\partial \alpha_n} + m_i \left(\left[\frac{\partial\, ^{\mathcal{J}}\boldsymbol{r}_i}{\partial \alpha_n} \right]^{\times} \boldsymbol{J}_{T_i} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0i} \right]^{\times} \frac{\partial \boldsymbol{J}_{T_i}}{\partial \alpha_n} \right) \right) \tag{2.72}$$

Lastly, the Euler angles derivative of the matrix $\boldsymbol{H}_m$, associated again to the *k-th* limb, can be expressed as shown in Equation 2.73:

$$\frac{\partial \boldsymbol{H}_m}{\partial \alpha_n} = \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{J}_{R_i}^T}{\partial \alpha_n} {}^{\mathcal{J}}\boldsymbol{I}_i \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T \frac{\partial\, ^{\mathcal{J}}\boldsymbol{I}_i}{\partial \alpha_n} \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T {}^{\mathcal{J}}\boldsymbol{I}_i \frac{\partial \boldsymbol{J}_{R_i}}{\partial \alpha_n} + m_i \left(\frac{\partial \boldsymbol{J}_{T_i}^T}{\partial \alpha_n}\boldsymbol{J}_{T_i} + \boldsymbol{J}_{T_i}^T \frac{\partial \boldsymbol{J}_{T_i}}{\partial \alpha_n} \right) \right) \tag{2.73}$$

2.4.5. Joint Angles Derivative of the System Inertia Matrix

The last two parameters needed to fully characterize the system's equations of motion are $\boldsymbol{c}_A$ and $\boldsymbol{c}_B$, defined in Equation 2.37. Their calculation requires the derivative of the kinetic energy (and so, of the system inertia matrix) with respect to the joint angles $\boldsymbol{q}_A$ and $\boldsymbol{q}_B$, associated respectively to the two different limbs of the manipulators: it is evident that the mathematical formulation will be exactly the same, regardless of the choice of which limb is analyzed.

The approach followed to retrieve the expression of the derivative of the system inertia matrix is basically the same as the one explained in Section 2.4.4: the calculation of the joint angles derivatives of $\boldsymbol{T}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{T}_{L_i}^{\mathcal{J}}$ (associated to the generic *k-th* limb of the manipulator) will be used to eventually extract the derivatives of the rotation matrices $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{R}_{L_i}^{\mathcal{J}}$, as well as the ones of the position vectors $^{\mathcal{J}}\boldsymbol{p}_i$ and $^{\mathcal{J}}\boldsymbol{r}_i$.

In general, by rewriting the definition of the homogeneous transformation matrix associated to the *i-th* joint-fixed frame as in Equation 2.74, it can be noted that both $\boldsymbol{T}_{J_n}^{\mathcal{J}}$ and the matrices $\boldsymbol{A}(q_l, d_l, \alpha_l, c_l)$, with $n+1 \leq l \leq i-1$, are independent of the *n-th* joint angle q_n.

$$\boldsymbol{T}_{J_i}^{\mathcal{J}} = \boldsymbol{T}_{J_n}^{\mathcal{J}}\, \boldsymbol{A}(q_n,\, d_n,\, \alpha_n,\, c_n) \prod_{l=n+1}^{i-1} \boldsymbol{A}(q_l,\, d_l,\, \alpha_l,\, c_l) \tag{2.74}$$

Therefore, in the complete direct kinematics chain, only one of these matrices $\boldsymbol{A}$ will depend on q_n, accordingly to the derivative shown in Equation 2.75:

$$\frac{\partial \boldsymbol{A}(q_n,\, d_n,\, \alpha_n,\, c_n)}{\partial q_n} = \begin{bmatrix} -sin(q_n) & -cos(q_n)cos(\alpha_n) & cos(q_n)sin(\alpha_n) & -c_n sin(q_n) \\ cos(q_n) & -sin(q_n)cos(\alpha_n) & sin(q_n)sin(\alpha_n) & c_n\, cos(q_n) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.75}$$

Now, starting with the joint angles derivative of $\boldsymbol{T}_{J_i}^{\mathcal{J}}$, three different cases must be analyzed, depending onto which particular joint rotation is being described by q_n.

1. If $n > i-1$, the transformation matrix is independent of q_n, resulting in:

$$\frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial q_n} = \boldsymbol{0}_{4\times 4} \tag{2.76}$$

2. If $n = i-1$, only the last matrix $\boldsymbol{A}$ in the product depends on q_n, leading to:

$$\frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial q_n} = \boldsymbol{T}_{J_n}^{\mathcal{J}}\, \frac{\partial \boldsymbol{A}(q_n,\, d_n,\, \alpha_n,\, c_n)}{\partial q_n} \tag{2.77}$$

3. If $n < i-1$, one of the matrices $\boldsymbol{A}$ in the product depends on q_n, resulting in:

$$\frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial q_n} = \boldsymbol{T}_{J_n}^{\mathcal{J}}\, \frac{\partial \boldsymbol{A}(q_n,\, d_n,\, \alpha_n,\, c_n)}{\partial q_n} \prod_{l=n+1}^{i-1} \boldsymbol{A}(q_l,\, d_l,\, \alpha_l,\, c_l) \tag{2.78}$$

Similar considerations can be made also for the joint angles derivative of $\boldsymbol{T}_{L_i}^{\mathcal{J}}$.

1. If $n > i$, the transformation matrix is independent of q_n, resulting in:

$$\frac{\partial \boldsymbol{T}_{L_i}^{\mathcal{J}}}{\partial q_n} = \boldsymbol{0}_{4\times 4} \tag{2.79}$$

2. If $n = i$, only the last matrix $\boldsymbol{A}$ in the product depends on q_n, leading to:

$$\frac{\partial \boldsymbol{T}_{L_i}^{\mathcal{J}}}{\partial q_n} = \boldsymbol{T}_{J_n}^{\mathcal{J}} \frac{\partial \boldsymbol{A}(q_n,\, d_n,\, \alpha_n,\, c_n/2)}{\partial q_n} \tag{2.80}$$

3. If $n < i$, one of the matrices $\boldsymbol{A}$ in the product depends on q_n and, by exploiting the definition of the link-fixed frame in Equation 2.29, it can be simply obtained that:

$$\frac{\partial \boldsymbol{T}_{L_i}^{\mathcal{J}}}{\partial q_n} = \frac{\partial \boldsymbol{T}_{J_i}^{\mathcal{J}}}{\partial q_n} \boldsymbol{A}(q_i,\, d_i,\, \alpha_i,\, c_i/2) \tag{2.81}$$

At this point, for each of the cases discussed above, the joint angles derivatives of $\boldsymbol{R}_{J_i}^{\mathcal{J}}$ and $\boldsymbol{R}_{L_i}^{\mathcal{J}}$, as well as the ones of the position vectors ${}^{\mathcal{J}}\boldsymbol{p}_i$ and ${}^{\mathcal{J}}\boldsymbol{r}_i$, can be simply extracted following the same procedure shown in Equations 2.57 and 2.58.
Afterwards, the derivatives of the two *Jacobians* can be obtained as follows:

$$\frac{\partial \boldsymbol{J}_{R_i}}{\partial q_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{R}_{J_1}^{\mathcal{J}}}{\partial q_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & , \dots , & \dfrac{\partial \boldsymbol{R}_{J_i}^{\mathcal{J}}}{\partial q_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & , \; \boldsymbol{0}_{3\times(N-i)} \end{bmatrix} \tag{2.82}$$

$$\frac{\partial \boldsymbol{J}_{T_i}}{\partial q_n} = \begin{bmatrix} \dfrac{\partial \boldsymbol{J}_{T_{i_1}}}{\partial q_n} & , \dots , & \dfrac{\partial \boldsymbol{J}_{T_{i_m}}}{\partial q_n} & , \; \boldsymbol{0}_{3\times(N-i)} \end{bmatrix} \tag{2.83}$$

with:

$$\frac{\partial \boldsymbol{J}_{T_{i_m}}}{\partial q_n} = \left[\frac{\partial \boldsymbol{R}_{J_m}^{\mathcal{J}}}{\partial q_n} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]^{\times} (\boldsymbol{r}_i - \boldsymbol{p}_m) + \left[{}^{\mathcal{J}}\hat{\boldsymbol{k}}_m \right]^{\times} \left(\frac{\partial \boldsymbol{r}_i}{\partial q_n} - \frac{\partial \boldsymbol{p}_m}{\partial q_n} \right) \quad \forall m \leq i \tag{2.84}$$

Finally, the derivative of the matrix containing the moments of inertia of the *i-th* link can be computed as:

$$\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{I}_i}{\partial q_n} = \frac{\partial \boldsymbol{R}_{L_i}^{\mathcal{J}}}{\partial q_n}\, {}^{L_i}\boldsymbol{I}_i\, \boldsymbol{R}_{L_i}^{\mathcal{J}^T} + \boldsymbol{R}_{L_i}^{\mathcal{J}}\, {}^{L_i}\boldsymbol{I}_i \frac{\partial \boldsymbol{R}_{L_i}^{\mathcal{J}^T}}{\partial q_n} \tag{2.85}$$

It is now possible to assemble the joint angle derivatives of all the various inertia sub-

matrices. Starting from $\boldsymbol{H}_0$, it yields:

$$\frac{\partial \boldsymbol{H}_0}{\partial q_n} = \begin{bmatrix} \mathbf{0}_{3\times 3} & -m_{tot}\left[\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial q_n}\right]^{\times} \boldsymbol{B}_{\mathcal{N}} \\ m_{tot}\, \boldsymbol{B}_{\mathcal{N}}^T \left[\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial q_n}\right]^{\times} & \boldsymbol{B}_{\mathcal{N}}^T \frac{\partial \boldsymbol{H}_s}{\partial q_n} \boldsymbol{B}_{\mathcal{N}} \end{bmatrix} \tag{2.86}$$

where, recalling the same considerations done for Equation 2.69 but now for the joint angles:

$$\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{0C}}{\partial q_n} = \frac{1}{m_{tot}} \sum_{k=1}^{2} \sum_{i=1}^{N} \left(m_{k,i} \frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial q_n} \right) \tag{2.87}$$

$$\frac{\partial \boldsymbol{H}_s}{\partial q_n} = \sum_{k=1}^{2} \sum_{i=1}^{N} \left(\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{I}_{k,i}}{\partial q_n} - m_{k,i} \left(\left[\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial q_n}\right]^{\times} \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0k,i}\right]^{\times} \left[\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_{k,i}}{\partial q_n}\right]^{\times} \right) \right) \tag{2.88}$$

Then, the joint angles derivative of the matrix $\boldsymbol{H}_{0m}$, associated to the *k-th* limb, can be expressed as:

$$\frac{\partial \boldsymbol{H}_{0m}}{\partial q_n} = \begin{bmatrix} \frac{\partial \boldsymbol{J}_{TS}}{\partial q_n} \\ \boldsymbol{B}_{\mathcal{N}}^T \frac{\partial \boldsymbol{H}_{sq}}{\partial q_n} \end{bmatrix} \tag{2.89}$$

where:

$$\frac{\partial \boldsymbol{J}_{TS}}{\partial q_n} = \sum_{i=1}^{N} \left(m_i \frac{\partial \boldsymbol{J}_{T_i}}{\partial q_n} \right) \tag{2.90}$$

$$\frac{\partial \boldsymbol{H}_{sq}}{\partial q_n} = \sum_{i=1}^{N} \left(\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{I}_i}{\partial q_n} \boldsymbol{J}_{R_i} + {}^{\mathcal{J}}\boldsymbol{I}_i \frac{\partial \boldsymbol{J}_{R_i}}{\partial q_n} + m_i \left(\left[\frac{\partial\, {}^{\mathcal{J}}\boldsymbol{r}_i}{\partial q_n}\right]^{\times} \boldsymbol{J}_{T_i} + \left[{}^{\mathcal{J}}\boldsymbol{r}_{0i}\right]^{\times} \frac{\partial \boldsymbol{J}_{T_i}}{\partial q_n} \right) \right) \tag{2.91}$$

Lastly, the joint angles derivative of the matrix $\boldsymbol{H}_m$, associated again to the *k-th* limb, can be expressed as shown in Equation 2.92:

$$\frac{\partial \boldsymbol{H}_m}{\partial q_n} = \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{J}_{R_i}^T}{\partial q_n} {}^{\mathcal{J}}\boldsymbol{I}_i \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T \frac{\partial\, {}^{\mathcal{J}}\boldsymbol{I}_i}{\partial q_n} \boldsymbol{J}_{R_i} + \boldsymbol{J}_{R_i}^T {}^{\mathcal{J}}\boldsymbol{I}_i \frac{\partial \boldsymbol{J}_{R_i}}{\partial q_n} + m_i \left(\frac{\partial \boldsymbol{J}_{T_i}^T}{\partial q_n} \boldsymbol{J}_{T_i} + \boldsymbol{J}_{T_i}^T \frac{\partial \boldsymbol{J}_{T_i}}{\partial q_n} \right) \right) \tag{2.92}$$

2.5. Controllers

The servicing-spacecraft is assumed to operate in a *Rotation-Flying* mode: this means that only the manipulator's joints and the orientation of the base are actively controlled, thus leaving the satellite free to translate in space accordingly to the motion of the limbs. Hence, by also recalling the block diagram presented in Figure 2.1, it is evident that two different controllers will be needed to ensure a successful capture of the *Target* satellite. Their tasks are summarized in Algorithm 2.1 and an overview of these two controllers is presented in the following two sections.

Algorithm 2.1 Numerical Model Outer Loop

Given an initial configuration for *Chaser*, *Target* and the two manipulator's limbs
while *EEs have not reached GPs* **do**
  if *Chaser's base not aligned with frame* $\mathcal{J}$ **then**
    Guide limbs from 'stowed' to 'home' configuration
    Rotate the *Chaser* to match $\mathcal{J}$ within predefined tolerance
  else
    Limbs to follow the reference joints trajectories provided by *RRT**
    Keep the *Chaser*'s base still
  end if
end while
Provide the generated trajectories of the joints as input to the dynamic model

2.5.1. Joints Controller

The first one is the *Joints Controller*, essential for the multibody dynamics model to accurately track the N+N different joints reference trajectories generated by the RRT* algorithm. The proposed solution is the implementation of a Proportional-Derivative (PD) feedback control technique: even though this type of linear controller is among the simplest that can be used for this kind of applications, it showed to be very effective in controlling the trajectory of the joints, providing satisfactory results.
Its working principle is based on the minimization of the error between the actual and the desired joint angles references, as well as on the minimization of the error's time derivative.

$$\boldsymbol{e} = \boldsymbol{q}_D - \boldsymbol{q} \qquad \dot{\boldsymbol{e}} = \dot{\boldsymbol{q}}_D - \dot{\boldsymbol{q}} \tag{2.93}$$

The main advantage of this control system is that it is very easy to implement, being the control law simply defined as in Equation 2.94:

$$\boldsymbol{\tau} = \boldsymbol{K}_P\, \boldsymbol{e} + \boldsymbol{K}_D\, \dot{\boldsymbol{e}} \quad \Longrightarrow \quad \boldsymbol{\tau} = \boldsymbol{K}_P\, (\boldsymbol{q}_D - \boldsymbol{q}) + \boldsymbol{K}_D\, (\dot{\boldsymbol{q}}_D - \dot{\boldsymbol{q}}) \tag{2.94}$$

where $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ are , respectively, the proportional and derivative gain matrices, both of dimension $[N \times N]$. Those two matrices were set to be diagonal, so that each joint is controlled independently of the others and thus there are no cross-coupling terms between the input torques. In principle, there are a variety of techniques that can be used to tune the values of the coefficients of these two matrices, like the classical *Ziegler-Nicholson* method or the *poles-placement* and *Linear Quadratic Regulator* (LQR) techniques [19]. Nevertheless, since this topic is not the main scope of the thesis, the gains were set using a trial and error approach.

2.5.2. Chaser's Attitude Controller

The other one is the *Chaser's Attitude Controller*, which has a twofold purpose: first of all, it enables the servicing-spacecraft to perform the slew maneuver to acquire a certain initial attitude that is favorable for the capture operations; secondly, it is used to stabilize the *Chaser*'s base orientation during the actual motion of the manipulator's limbs.
To define this controller it is useful to recall that the equations of motion of the system are directly expressed, among the other variables, with respect to the Euler angles, which characterize the attitude of the *Chaser* relative to the *Target*, and not with respect to the *Chaser*'s angular velocity. This particular choice represent an advantage that can be exploited in the design of the controller. Indeed, the expression of the relative attitude and angular velocity between the two satellites can be simplified by directly directly using the Euler angles (and their derivatives) integrated by the *Dynamics* block shown in Figure 2.1. Therefore, using a similar approach to what was done for the *Joints Controller*, a simple PD controller can be defined, where, in this case, the attitude error and its derivative, as well as the final control law, are obtained as follows:

$$\boldsymbol{e} = \boldsymbol{\alpha}_D - \boldsymbol{\alpha} \qquad \dot{\boldsymbol{e}} = \dot{\boldsymbol{\alpha}}_D - \dot{\boldsymbol{\alpha}} \tag{2.95}$$

$$\boldsymbol{\tau} = \boldsymbol{K}_P\, \boldsymbol{e} + \boldsymbol{K}_D\, \dot{\boldsymbol{e}} \quad \Longrightarrow \quad \boldsymbol{\tau} = \boldsymbol{K}_P\, (\boldsymbol{\alpha}_D - \boldsymbol{\alpha}) + \boldsymbol{K}_D\, (\dot{\boldsymbol{\alpha}}_D - \dot{\boldsymbol{\alpha}}) \tag{2.96}$$

Thus, there is no need to take into account in the control law the non-trivial conversion between Euler angles and angular velocity, since this process in already incorporated in derivation of the equations of motion. Moreover, this formulation avoids to deal with

the singularities characterizing the particular sequence of Euler angles that was chosen in Section 2.2.
In this case, the two gain matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ of dimension $[3 \times 3]$ have different coefficients in relation to which one of the two operating modes of the controller is being used. Again, these gains were retrieved using a trial and error approach.
Of course, it can be noticed that the value $\dot{\boldsymbol{\alpha}}_D$ must be set to be zero in both the two operation modes of this controller, due to the fact that the presence of a desired non-null angular velocity is physically not compatible with trying to stabilize the Euler angles at a certain fixed set $\boldsymbol{\alpha}_D$.

The formulation of the *Chaser's Attitude Controller* made with this approach allows also to model a different capture scenario with respect to the one analyzed in this thesis, which however have not been further discussed but that can be still of relevance and interest with respect to real world operations: here, the *Target* may be a defunct satellite spinning around its major axis of inertia and the *Chaser*, after performing a rendez-vous, needs to synchronize its rotation to the one of the *Target* before starting the capture procedure. This can be achieved by controlling $\dot{\boldsymbol{e}}$ only.

3 | Trajectory Planning

In principle, the trajectory to be followed by the space manipulator could be defined according to two approaches, which are one the dual of the other:

- **Joint Space Trajectory**: The motion of the manipulator is defined in terms of the characteristic angle of each of its joints. Direct Kinematics is then used to retrieve the actual position of the end-effector in the 3D Cartesian space.
- **Task Space Trajectory**: The motion of the manipulator is defined in terms of its end-effector's position and orientation in the real 3D world (Cartesian space). Inverse Kinematics is therefore used at each time step to retrieve the particular set of joint angles that ensures a certain desired pose of the end-effector.

Since in real applications it is ultimately the joints that are actively controlled by the on-board computer, a trajectory defined in joint space should be preferred to avoid sudden jumps, jerks or discontinuities in the motion of the manipulator. Direct Kinematics is then used to just map the joint angles into the end-effector's position, but the smoothness of the movement is therefore embedded at the joints level. What just explained gains even more importance when dealing with collision avoidance: indeed, it is preferable to not have sudden movements of the manipulator when the latter is in proximity of any kind of obstacle along its path. Moreover, this approach is also computationally more efficient than a trajectory planning in the task space, since the optimization problem of the Inverse Kinematics algorithm is solved only at specific waypoints (that are then interpolated in the joint space) and not at every time step of the simulation.

In this chapter, the process of generating the trajectory for the multi limb manipulator using the Rapidly-exploring Random Tree Star (RRT*) algorithm is explained. The developed algorithm is firstly introduced with a focus on its key features, followed by a detailed description of the collision avoidance procedure. The final goal of RRT* will therefore be to obtain a feasible trajectory for the N joints of each limb of the space manipulator, thus ensuring that the two end-effectors will eventually reach a predefined grasping point on the *Target* satellite with great accuracy, both in terms of position and orientation.

3.1. RRT* Algorithm

First introduced just over a decade ago by Karaman and Frazzoli [20], the Rapidly-exploring Random Tree Star (RRT*) algorithm is an improved version of the classical RRT formulated in [21]; as anticipated in Section 1.3, it belongs to the family of *Sampling-Based Methods*, a class of algorithms commonly used in robotics for trajectory planning. Some definitions shall be introduced before outlining the algorithm's working principle:

- **Configuration Space**
  The configuration space $\mathcal{C} \subseteq \mathbb{R}^N$ is the N-dimensional manifold representing all possible configurations $\boldsymbol{q} = (q_1, q_2, ..., q_N)$ of the manipulator, and is delimited by the joints' kinematic constraints.

- **Node**
  A node $\boldsymbol{q} = (q_1, q_2, ..., q_N) \in \mathcal{C}$ is defined as a specific set of N joint angles. This corresponds to a unique point in the N-dimensional configuration space.

- **Edge**
  An edge is a directed segment in $\mathcal{C}$ connecting two different nodes $\boldsymbol{q}_a, \boldsymbol{q}_b \in \mathcal{C}$, such that the path between $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ is collision-free and satisfies the joints' motion constraints.

- **Tree**
  The tree $T = (V, E)$ is a directed graph where V is a finite set of nodes and $E \subset V \times V$ is the corresponding set of edges. Its origin $\boldsymbol{q}_{start}$ is the manipulator's starting configuration.

The main goal of RRT* is therefore to find a feasible path, in the configuration space of the manipulator, from an initial starting point to a predefined goal point. This is done by randomly exploring this space through the use of a fast growing tree of feasible states, while also ensuring that the solution path becomes increasingly optimal as more nodes are added to the tree. Many kind of constraints can be integrated in this algorithm, the most important of which is, undoubtedly, collision avoidance: this is particularly true when dealing with robotic solutions in the context of on-orbit servicing missions, where possible collisions involving the manipulator could result in catastrophic consequences that could lead not only to the failure of the mission, but also to the possible irreversible damage of critical components of the satellite being serviced.
Therefore, a good RRT* algorithm shall produce not only a near-optimal path that minimizes, for example, the total distance traveled by each limb of the manipulator, but also a path that is robust against possible collisions.

The general structure of the proposed RRT* algorithm is presented in Algorithm 3.1

Algorithm 3.1 RRT*

Given: GP coordinates and orientation in *Chaser*'s body-fixed frame ($\mathcal{B}_C$)
$\boldsymbol{q}_{goal} \leftarrow$ `INVERSE_KIN`(GP pose) : Inverse Kinematics to compute $\boldsymbol{q}_{goal}$
$\boldsymbol{q}_{start} \leftarrow \mathbf{0}_{6\times1}$: Initialize `Tree` with the first node
while `iter` $<$ $\texttt{iter}_{\texttt{max}}$ **do**
  $\boldsymbol{q}_{rand} \leftarrow$ `RANDOM( )` : Generate random node
  $\boldsymbol{q}_{nearest} \leftarrow$ `NEAREST`(`Tree`,$\boldsymbol{q}_{rand}$) : Find nearest node in `Tree`
  $\boldsymbol{q}_{new} \leftarrow$ `STEER`($\boldsymbol{q}_{nearest}$,$\boldsymbol{q}_{rand}$) : Generate new possible node
  if `COLLISION_FREE`($\boldsymbol{q}_{new}$) **then**
    `BEST_PARENT`($\boldsymbol{q}_{new}$) : Search $\boldsymbol{q}_{new}$'s best parent among $\boldsymbol{q}_{neighbours}$ within R_{search}
    if `BEST_PARENT`($\boldsymbol{q}_{new}$) == `TRUE` **then**
      if `COLLISION_FREE`($\boldsymbol{q}_{new} \rightarrow$ `BEST_PARENT`($\boldsymbol{q}_{new}$)) **then**
        $\boldsymbol{q}_{new.parent} \leftarrow$ `BEST_PARENT`($\boldsymbol{q}_{new}$)
        for all $\boldsymbol{q}_{neighbours}$ **do**
          if $(\boldsymbol{q}_{start} \rightarrow \boldsymbol{q}_{new} \rightarrow \boldsymbol{q}_{neighbour}) < (\boldsymbol{q}_{start} \rightarrow \boldsymbol{q}_{neighbour})_{current}$ **then**
            `REWIRE`($\boldsymbol{q}_{new}$, $\boldsymbol{q}_{neighbour}$)
          end if
        end for
        $\boldsymbol{q}_{new} \leftarrow$ `ADD_NODE`(`Tree`)
      else
        $\boldsymbol{q}_{new} \leftarrow$ `NaN` : The node $\boldsymbol{q}_{new}$ is discarded
      end if
    else if `BEST_PARENT`($\boldsymbol{q}_{new}$) == `FALSE` **then**
      if `COLLISION_FREE`($\boldsymbol{q}_{new} \rightarrow \boldsymbol{q}_{nearest}$) **then**
        $\boldsymbol{q}_{new.parent} \leftarrow \boldsymbol{q}_{nearest}$
        $\boldsymbol{q}_{new} \leftarrow$ `ADD_NODE`(`Tree`)
      else
        $\boldsymbol{q}_{new} \leftarrow$ `NaN` : The node $\boldsymbol{q}_{new}$ is discarded
      end if
    end if
    `iter` = `iter` + 1
  end if
end while
`Path` $\leftarrow$ `BEST_PATH`(`Tree`) : Retrieve the shortest possible path by scanning `Tree`
`Trajectory` $\leftarrow$ `TRAJECTORY`(`Path`) : Generate the final trajectory

The algorithm is designed to handle one limb at a time, so that the trajectory information from the first limb can be used as input for the second, thus ensuring the synchronization of the two trajectories and preventing potential collisions between them.
Moreover, as it can be seen in the *while* loop in Algorithm 3.1, RRT* continues to generate random nodes until a predefined maximum number of iterations is reached: this way of operating ensures that the more samples are added to the tree, the more optimal the final path will be, thanks to the rewiring process. However, since the configuration space of the manipulator may be very large, even a considerably high number of iterations may be not sufficient to eventually find a valid path from start to goal nodes. A trade-off shall then be considered, that is whether to increase a priori the number of maximum iterations, with the effect of raising the odds of finding a valid path which is also near optimal, or to save computational time end effort and stopping the algorithm as soon as a valid path is found.
The main operations performed by the proposed RRT* algorithm, from the very beginning to the final generation of the joints' trajectories, are described hereafter:

Pre-processing

Recalling the hypothesis made in Section 1.4, the state and all the geometrical properties are assumed to be known/retrievable for both *Chaser* and *Target*. This means that, at every time instant, the relative position and orientation of each point of one of the two satellites can be expressed in the body-fixed frame of the other, and vice-versa. Therefore, if two grasping points (GPs), to be reached by the end-effectors of the two manipulator's limbs, are defined on the external surface of the *Target*'s body, their pose can always be expressed with respect to the *Chaser*'s body-fixed frame. Based on these considerations, the transformation matrix containing the information about the desired end-effector's pose can be retrieved by Equation 3.1.

$$\boldsymbol{T}_{GP}^{J_1} = \left(\boldsymbol{T}_{J_0}^{\mathcal{J}}\ \boldsymbol{T}_{J_1}^{J_0}\right)^{-1} \boldsymbol{T}_{GP}^{\mathcal{J}} = \begin{bmatrix} \boldsymbol{R}_{GP}^{J_1} & {}^{J_1}\boldsymbol{r}_{GP} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.1}$$

In this way, the matrix just computed will be provided as input to the Inverse Kinematics algorithm, explained in Section 2.3.3, in order to retrieve the set $\boldsymbol{q}_{goal}$ of joint angles required for the end-effector to reach the desired pose.

On the other hand, the starting configuration of the manipulator was chosen with all joint angles, and thus $\boldsymbol{q}_{start}$, set to zero. This was merely a design choice and did not affect the operation of the RRT* algorithm.

Random Sampling

The function `RANDOM( )` simply generates, through an uniform distribution, one random state $\boldsymbol{q}_{rand}$ belonging to the manipulator configuration space. Being the latter delimited by the maximum and minimum allowed values for each of the N joint angles, there is the certainty that every randomly sampled configuration can be potentially achieved by the manipulator, unless, of course, any collision occurs.

Nearest Node Identification

The function `NEAREST(Tree` , $\boldsymbol{q}_{rand}$) simply takes as input the newly generate random node $\boldsymbol{q}_{rand}$ as well as the already existing tree and returns the node $\boldsymbol{q}_{nearest}$ which is the closest to $\boldsymbol{q}_{rand}$, accordingly to a certain metric. In this case, the distance between two nodes is evaluated in terms of the N-dimensional Euclidean distance and, therefore, $\boldsymbol{q}_{nearest}$ can be found as:

$$\boldsymbol{q}_{nearest} = \arg \min_{\boldsymbol{q} \in \texttt{Tree}} \|\boldsymbol{q} - \boldsymbol{q}_{rand}\| \tag{3.2}$$

Steering Process

The aim of the function `STEER`($\boldsymbol{q}_{nearest}$, $\boldsymbol{q}_{rand}$) is to return a third node $\boldsymbol{q}_{new}$ located at an arbitrary distance δ from $\boldsymbol{q}_{nearest}$ along the N-dimensional line connecting $\boldsymbol{q}_{nearest}$ and $\boldsymbol{q}_{rand}$. This variable is defined prior to the start of the RRT* algorithm, and thus will be the same for each new iteration. The procedure is schematically reported in Figure 3.1:

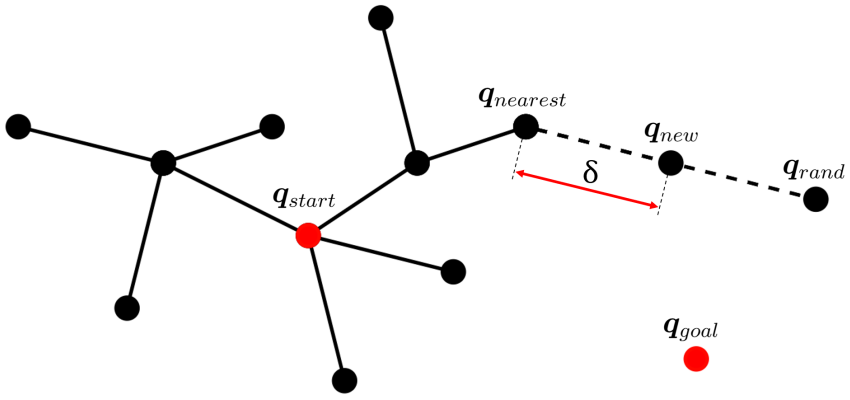

Figure 3.1: Steering Process

Best Parent Research

This step is, among with the rewiring one, what distinguishes RRT* from a classical RRT. In particular, the function `BEST_PARENT`($\boldsymbol{q}_{new}$) firstly identifies, among all the existing nodes in the tree, the subset $\boldsymbol{q}_{neighbours}$, which consists of the nodes whose distance relative

from $\boldsymbol{q}_{new}$ is less than a certain arbitrary threshold (R_{search}). At this point, $\boldsymbol{q}_{new}$ is no longer connected to the closest node, but rather to the one, chosen among $\boldsymbol{q}_{neighbours}$, such that the total distance from $\boldsymbol{q}_{start}$ to $\boldsymbol{q}_{new}$, along the edge of the tree passing through $\boldsymbol{q}_{neighbour}$, is minimized. The procedure is schematically reported in Figure 3.2:

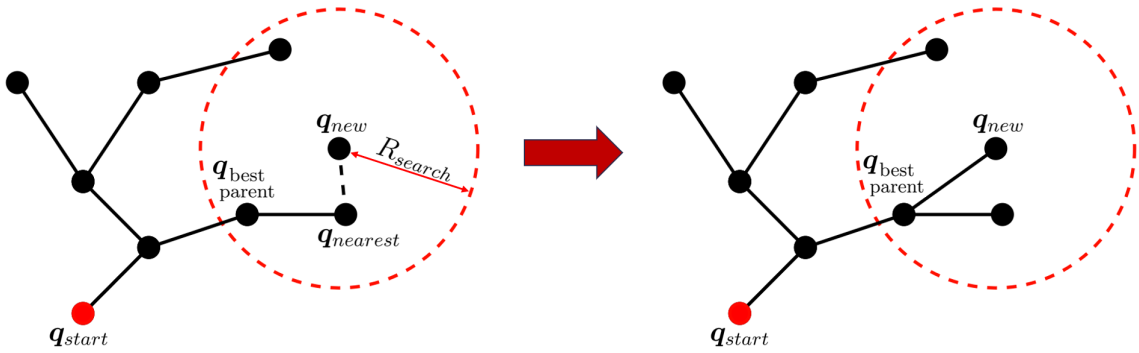

Figure 3.2: Best parent research

Rewiring

The concept of tree rewiring is the key factor that contributes to the near-optimal performance of RRT*. The working principle of the function `REWIRE`($\boldsymbol{q}_{new}$, $\boldsymbol{q}_{neighbour}$) in Algorithm 3.1 is the following: for each of the $\boldsymbol{q}_{neighbours}$ nodes (the same ones identified in the *Best Parent Research* step), the function checks iteratively whether connecting $\boldsymbol{q}_{new}$ to that neighbor results in a lower-cost path than the current path. If a shorter path is found, the parent of the neighbor node is updated to be the $\boldsymbol{q}_{new}$ and the edge connecting $\boldsymbol{q}_{neighbour}$ to its previous parent is deleted, effectively rewiring the tree. Eventually, the cost of $\boldsymbol{q}_{neighbour}$ and of each of its 'sons' is updated as a consequence of the rewire.
This procedure is visually explained in Figure 3.3.

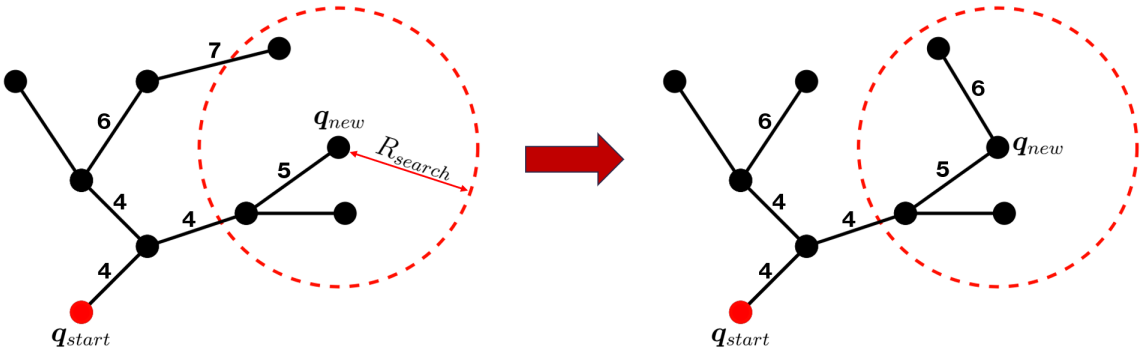

Figure 3.3: Rewiring process, with fictitious lines' cost highlighted

It can be noticed how the specific choice of values of two free parameters (δ and R_{search}) introduced up to now in the algorithm has a great influence on its behavior, especially regarding the *Best Parent Research* and the *Rewiring* process.
Indeed, by choosing $R_{search} > \delta$, it is possible to have a double favorable effect: $\boldsymbol{q}_{new}$ has higher chances of finding a parent-node which could guarantee a lower-cost path and, at the same time, there will be an increase in the number of neighbor nodes that could be rewired, resulting thus in an higher probability of finding a near-optimal final path. Of course, a trade-off shall be considered, since a value of R_{search} much higher than the one of δ could result in the creation of connections between nodes that are much distant from each other, leading to a general decrease of performances of the algorithm.
An example of this behavior is shown, in a simplified two dimensional configuration space, in Figures 3.4 and 3.5:

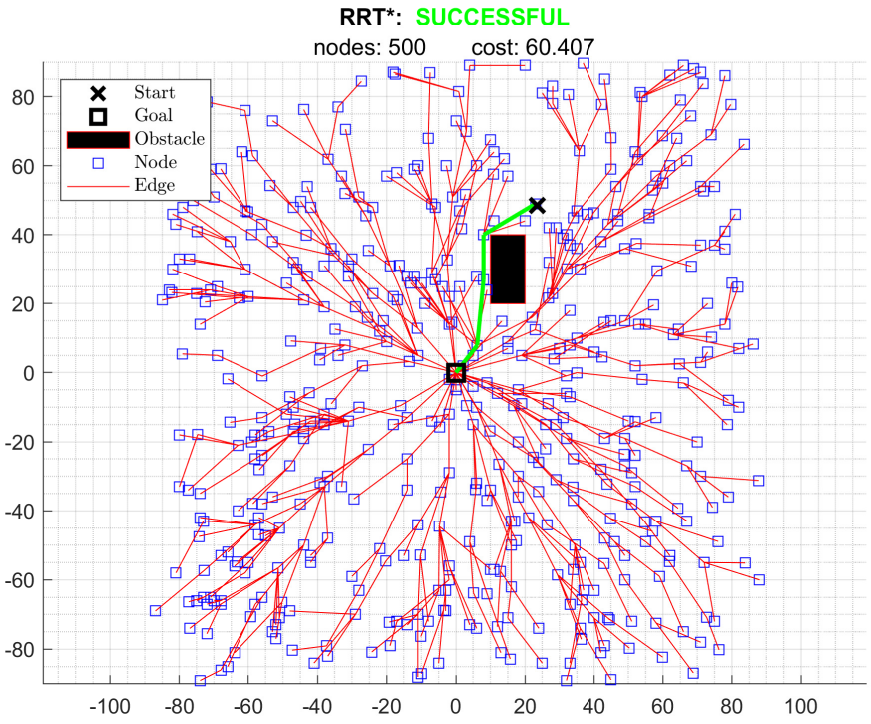

Figure 3.4: RRT* with $\delta = 10$, $R_{search} = 20$

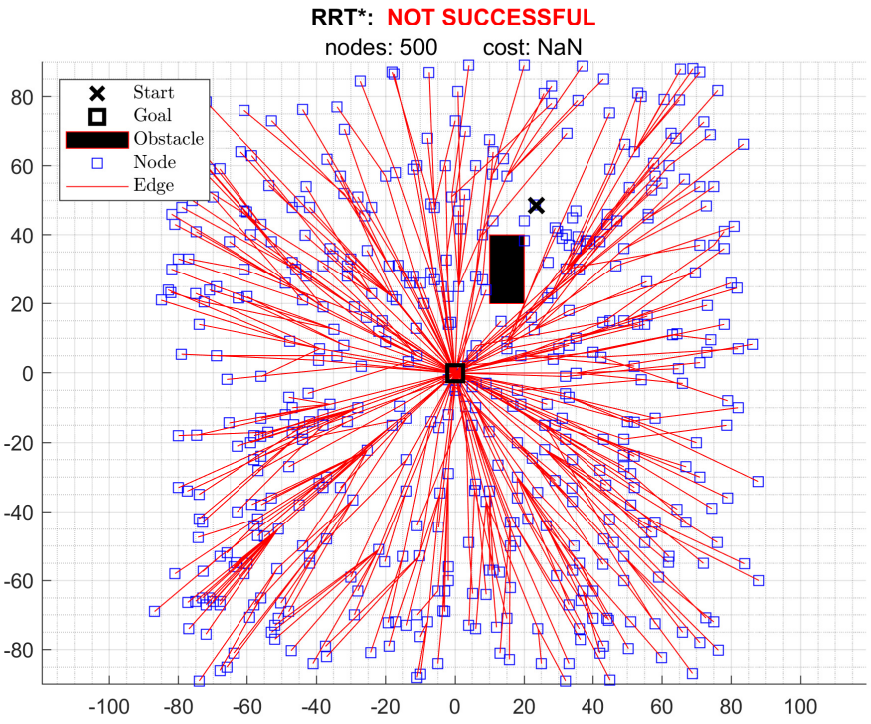

Figure 3.5: RRT* with $\delta = 10$, $R_{search} = 40$

Collision Avoidance

Up to now, only the process of generating new nodes and edges was considered. What is missing is the most important feature of the RRT* algorithm, that is the capability to generate a path that avoids all the obstacles in the workspace of the manipulator. In particular, the function `COLLISION_FREE`() checks for the three different kinds of possible collisions that have been identified in the satellite's capture scenario under analysis:

1. *External Collision*: Collision of any part of the manipulator with the *Chaser's* base or with the *Target*;
2. *Self Collision*: Collision of any part of the manipulator's limb with the manipulator's limb itself;
3. *Mutual Collision*: Collision between the two limbs of the manipulator.

Since each type of collision has its own distinct characteristics, they were not treated in the same way. As a result, three different approaches were developed in order to grasp, for each one, the most relevant aspects.

Starting with the *External Collision*, the first problem to be addressed is how to model the different kind of obstacles that could have, in principle, any kind of shape in the 3D

Cartesian space. One way of operating could be, as done in [12], to map the geometric shape of the known obstacles directly in the configuration space of the manipulator: this would result in forbidden regions of the configuration space and, therefore, the validity of new nodes/edges could be easily checked by looking if those items fall into those regions. However, this approach is effective only when dealing with simplified contexts, where the number of degrees of freedom of the system is very low, such as planar capture problems. Indeed, when more complex scenarios are analyzed, not only the mapping itself becomes computationally challenging (since the Inverse Kinematics algorithm shall be used for each point of the obstacles), but also the managing of these N-dimensional forbidden regions would become practically unfeasible due to the high number of degrees of freedom considered.
Therefore, another approach shall be followed: one possible solution is to apply the methodology presented in [22] and [23], where each obstacle is modeled using a certain *super quadratic function* whose parameters are adjusted to describe the shape and dimensions characteristic of that obstacle. This approach is advantageous especially for space applications, where satellites and their appendages usually have, as a first approximation, simple shapes which can in turn be obtained as a superposition of geometric primitives, like cubes, spheres and cylinders.
The general expression of a *super quadratic function*, used to approximate the external surface of a geometric primitive (whose volume is identified by the three parameters h_1, h_2, h_3), is presented in Equation 3.3.

$$^{obs}\mathcal{S}(x,y,z) = \left(\frac{x}{h_1}\right)^{2m} + \left(\frac{y}{h_2}\right)^{2n} + \left(\frac{z}{h_3}\right)^{2p} - 1 \tag{3.3}$$

The superscript '*obs*' means that this function is centered with respect to the obstacle's body-fixed frame, while the parameters 'm','n' and 'p' are the powers that determine the actual shape of the object. In this thesis, for simplicity, only cuboidal obstacles has been considered: the corresponding *super quadratic function* can be therefore expressed by substituting, in Equation 3.3, the values $m = 4$, $n = 4$, $p = 4$ and $h_i = l_i$, where l_i represent half of the *i-th* dimension of the cuboid [1].

The peculiarity of this kind of functions is that they can be very useful not only to approximate the envelope of a generic obstacle, but also to determine whether an arbitrary point $\boldsymbol{P}$, whose coordinates relative to the obstacle's body-fixed frame are $^{obs}\boldsymbol{P} = (x_p, y_p, z_p)$,

[1]Since this way of approximating the shape of an obstacle tends to underestimate its actual dimensions, especially in proximity of the corners, a safety factor shall be introduced in the h_i values before substituting them in Equation 3.3.

lies inside, on the surface or outside the obstacle. Indeed, the following relations hold:

$$\begin{cases} {}^{obs}\mathcal{S}(x_p, y_p, z_p) < 0 & \text{if } {}^{obs}\boldsymbol{P} \text{ is inside the obstacle} \\ {}^{obs}\mathcal{S}(x_p, y_p, z_p) = 0 & \text{if } {}^{obs}\boldsymbol{P} \text{ is on the surface of the obstacle} \\ {}^{obs}\mathcal{S}(x_p, y_p, z_p) > 0 & \text{if } {}^{obs}\boldsymbol{P} \text{ is outside the obstacle} \end{cases} \tag{3.4}$$

Therefore, it is evident that Equation 3.4 can be used to assess whether a collision between the manipulator and an external obstacle has occurred or not. For each newly generated node $\boldsymbol{q}_{new}$, Direct Kinematics is used to retrieve the specific configuration of the manipulator and, consequently, the position of each of its points. Then, a finite set of these points, biased toward the end-effector but still representative of the overall configuration of the manipulator, is selected and rotated into the obstacle's body-fixed frame using:

$${}^{obs}\boldsymbol{P}_i = \boldsymbol{R}_{\mathcal{J}}^{obs}\ {}^{\mathcal{J}}\boldsymbol{P}_i \quad \forall i \in [1, N_{points}] \tag{3.5}$$

Eventually, the coordinates of each of these points ${}^{obs}\boldsymbol{P}_i$ are iteratively substituted in the *super quadratic function* associated to the different obstacles: whenever the condition ${}^{obs}\mathcal{S}(x_{p_i}, y_{p_i}, z_{p_i}) \leq 0$ is verified, the process is terminated and the corresponding manipulator's configuration (i.e the node $\boldsymbol{q}_{new}$) is discarded.
The same reasoning applies to the generation of edges connecting two nodes in the tree, as well as to those created during the *Rewiring* step: these edges, which represent nothing but sequences of manipulator's configurations, are densely sampled to form a finite set of configurations that will be eventually checked for collisions using the same approach described earlier.

For what concerns instead the problem of *Self Collision*, a slightly different approach was implemented. Firstly, in the same way as done for the *External Collision*, from each newly generated node (or edge) the corresponding manipulator's pose(s) in Cartesian space is retrieved by the mean of the Direct Kinematics. Then, this configuration(s) is discretized into an arbitrary number of equally-spaced points, whose coordinates are therefore known. Finally, for each of these points, it is checked whether their mutual distance is less than a certain predefined safety threshold. If this condition is met, a collision is considered to have occurred, and the corresponding node (or edge) is discarded. Naturally, the value of the safety threshold must be selected such that it is smaller than the distance between two consecutive discrete points of the manipulator; otherwise, a collision may be erroneously detected.

Finally, regarding the *Mutual Collision*, it is important to recall that this version of the

RRT* algorithm is designed to first operate for one limb and then for the other. By doing so, the first limb is 'free' to move is space, subject only to the constraints imposed by the other two types of collision already discussed. Consequently, the trajectory generated by RRT* for the first limb will be treated as an 'obstacle' for the second limb, which must then maintain a safety distance from the points of the first limb. Therefore, every time a node or edge is created, it is checked using Direct Kinematics that the corresponding limb's configuration respect this minimum distance; if not, the node or edge is discarded.

Best Path Research

Once the algorithm has completed the predefined number of iterations, the function `BEST_PATH(Tree)` starts to scan the tree to determine whether a feasible path exists. If successful, it eventually determines which of the retrieved paths ensures the lowest overall cost. However, if no path is found, RRT* restarts from the beginning.
To find the lower cost path an approach similar to the *Best Parent Research* is used: firstly, the subset of nodes $\boldsymbol{q}_{neighbours}$, whose distance to $\boldsymbol{q}_{goal}$ is less than a predefined threshold (R_{goal}), is identified; then, $\boldsymbol{q}_{goal}$ is connected to the the node among $\boldsymbol{q}_{neighbours}$ which guarantees the lowest cost path back to $\boldsymbol{q}_{start}$.
Even in this step a trade-off shall be considered: indeed, if the value of R_{goal} is large, there will be a better chance of finding a feasible path, since the probability of finding neighbor nodes is increased. However, this could generate a connection between two nodes (i.e manipulator's configurations) that could be very distant from each other, resulting in undesired sudden jumps of the manipulator's state in the most delicate part of the its trajectory, when the end-effectors will likely be very close to the *Target*. Therefore, the value of R_{goal} shall be chosen such that it is not too restrictive, but, at the same time, also high enough to avoid sudden jumps between consecutive manipulator's configurations.

Trajectory Generation

Once a feasible path is eventually identified, the following step is to generate a timing law that enables the dynamic model to follow the aforementioned path by exploiting the *Joints Controller*. There are various methods that can be used to create a timing law, but all of these must consider the time and kinematic constraints involved in the analysis. Specifically, the total maneuver duration should be not too long, as this would compromise the assumption of neglecting orbital mechanics and environmental disturbances. However, it should also not be too short, as this would cause the manipulator to move too quickly, thus increasing the risk of collisions and potentially violating the joints' constraints on maximum angular rates and accelerations. Moreover, the end effector's velocity shall be

null both at the beginning and at the end of the trajectory: the first condition is because the capture will begin with the manipulator deployed in a certain starting configuration, while the other is needed in order to be able to securely grasp the *Target*.
Hereafter are briefly described the steps performed by the function `TRAJECTORY(Path)` under the constraints just mentioned, in order to eventually generate the final trajectory for the manipulator:

1. The first step is to smooth the discrete path generated by RRT*, which will therefore be constituted by a certain number N of waypoints (i.e nodes). This can be achieved by interpolating the segment between each pair of consecutive nodes with a 7-*th* degree polynomial, in order to ensure continuity in position, velocity, acceleration, and jerk.

2. Then, the function computes the times at which the trajectory should pass through each waypoint. For simplicity, these waypoints will be evenly spaced in time according to a predefined maneuver duration provided as an input to the function.

3. Afterwards, the function computes the position, velocity, acceleration and jerk at the specified sample times, using the piecewise polynomial assembled in the first step. It also ensures that both initial and final values of these variables are all set to zero.

4. The velocities and accelerations just calculated are then checked to ensure that they comply with the joints' constraints on maximum angular rates and accelerations. If they do not meet the requirements, the *Trajectory Generation* procedure is repeated by increasing, at each iteration, the total maneuver duration until the constraints are eventually satisfied.

5. Lastly, the newly interpolated path is densely sampled to check for potential collisions. If any collision is detected, the RRT* algorithm restarts from the beginning and the whole process is repeated until a feasible collision-less trajectory is obtained.

As already anticipated, the trajectory is initially computed only for the first limb, and only after a feasible collision-less trajectory is obtained, the total maneuver duration is given as a constraint for the generation of the trajectory for the other limb, thus to guarantee the synchronization between the two. At this point, the same checks as before, regarding potential collisions and compliance with kinematic constraints, are performed. If any violation of the constraints occur, the process is iteratively restarted by modifying the maneuver duration or, if not sufficient, by restarting the RRT* algorithm.

Trajectory Correction

In principle, by exploiting the process described up to now, a final collision-less trajectory could be eventually generated for each one of the two manipulator's limbs.

What it can be noticed, however, is that if these trajectories are given as input to the dynamic model, in such a way to be the references to be followed by the *Joint Controllers*, the simulated final position and orientation of the end-effectors will not coincide with their desired pose with a sufficiently satisfactory accuracy. This is caused by the fact that only the attitude of the *Chaser*'s base is actively controlled: therefore, the satellite will respond to the motion of the manipulators with a, albeit slight, translation in space. The latter will eventually cause a small discrepancy between the desired and actual pose of the end-effectors. Naturally, this condition must be avoided, since in the context under analysis even a slight deviation from the desired values can lead to the failure of the servicing mission.

A possible solution to reduce this bias and to eventually increase the performances of the numerical simulation is to slightly modify the very last part of the trajectories of the two manipulator's limbs. Specifically, a first numerical simulation is performed to evaluate how much the relative position between *Chaser* and *Target* has changed, at the end of the nominal capture maneuver, due to the *Chaser*'s movements caused by the motion of the manipulator. This information, contained in the transformation matrix $\boldsymbol{T}_{J_0}^{\mathcal{J}}\big|_{t=t_{final}}$, is then used together with the Inverse Kinematics algorithm (recall Section 2.3.3 as well as Equation 2.13) to compute what should be the new desired final configuration of the manipulator (which was previously designated as the node $\boldsymbol{q}_{goal}$). Essentially, the position attained by the *Chaser* at the end of the maneuver is used as a pseudo-initial configuration in order to compute the new, corrected, $\boldsymbol{q}_{goal}$.

This new node is eventually added at the end of the known path (the same that was previously obtained by RRT*) and a new trajectory is generated.

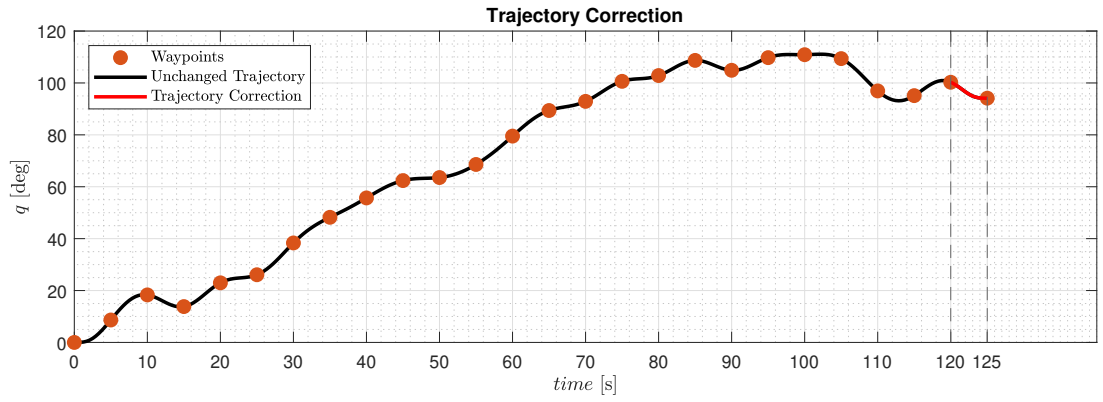

Figure 3.6: Example of Trajectory Correction

The final effect is to obtain, as shown in Figure 3.6, a final trajectory constituted by the the older one, which is kept unchanged, appended with the terminal correction part. In this way, given the very short duration of this small correction, the position of the base will be practically unchanged compared to one resulting from the trajectory computed prior to correction. Consequently, the end-effectors can finally reach the grasping points with great accuracy.

After repeating this *Trajectory Correction* procedure for both the limbs of the manipulator and performing a final check for potential collisions, it is possible to finally obtain the two actual trajectories than can be used as input for the multibody dynamic model.

4 | Simulation Setup and Results

In this chapter, the model proposed in Figure 2.1 and analyzed in detail in the previous sections is validated in a *Matlab-Simulink*® environment.
Firstly, the physical parameters of the two satellites and of the manipulator are defined. Then, all the other parameters needed for the simulation, like the initial conditions and the variables for the RRT* algorithm, are introduced. Finally, the results of a complete reference simulation are presented and discussed.

4.1. Simulation Setup

The simulation was performed in a realistic yet simplified 3D scenario, thus taking into account the hypothesis introduced in Section 1.4.
The rendez-vous maneuver is assumed to be completed before the beginning of the capture operations, with the effect of aligning the velocity vectors of the *Chaser* and the *Target* as they are placed in the same orbit at very close distances. So, at the end of this maneuver, the relative position between the two satellites is assumed to be small enough to allow the manipulator to operate: on the other hand, the initial relative attitude may not be the optimal to perform the capture. Therefore, as anticipated in Section 2.1, an initial slew maneuver is performed to reorient the *Chaser* such that the *LVLH* frames ($\mathcal{L}_C$ and $\mathcal{L}_T$) of the two satellites are aligned. Only at this point the actual capture can be performed by the manipulator.

4.1.1. Satellites and Manipulator Physical Parameters

Chaser Satellite and Manipulator

The system composed by the *Chaser* and the manipulator is shown in Figure 4.1, represented relative to the *Chaser*'s body-fixed coordinate system and with the manipulator's limbs in their *home configuration*. To distinguish between the two limbs, the one located in the positive region of axis $\mathcal{X}_C$ is denoted as *Limb A*, while the other as *Limb B*.

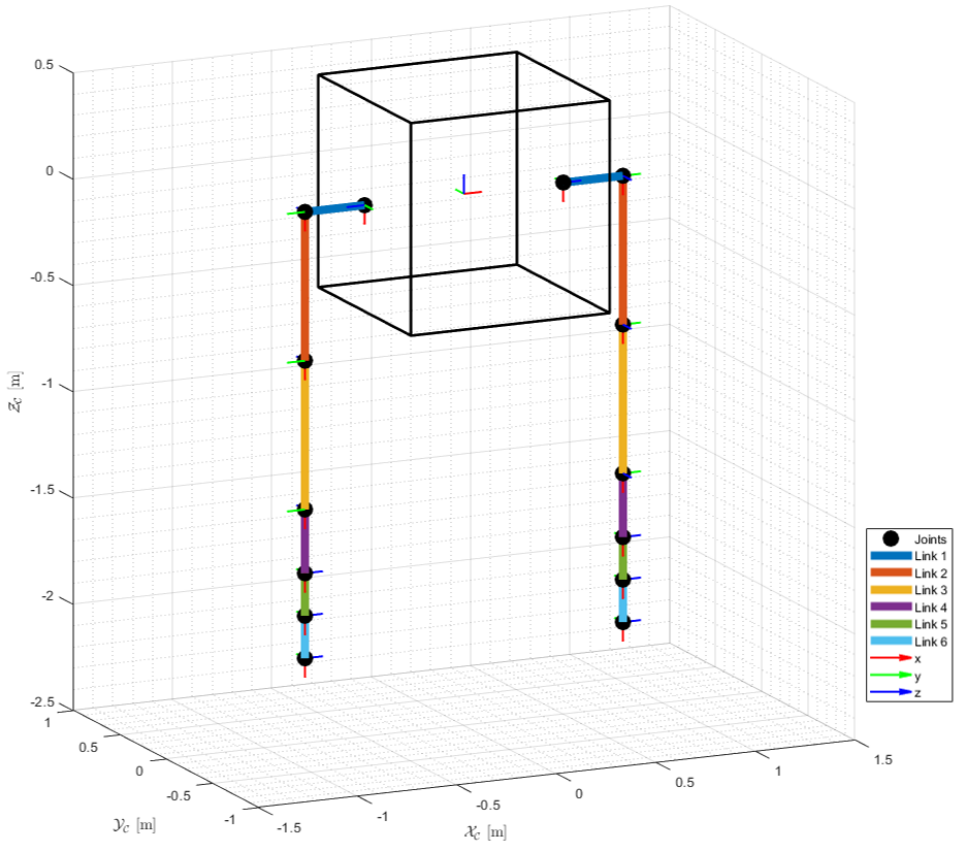

Figure 4.1: 3D model of Chaser and Manipulator (with limbs in *home configuration*)

All their physical parameters, such as mass, inertia and dimensions are reported in Tables 4.1 and 4.2. In particular, mass and geometry were assumed for the base, while all the links were considered as aluminum cylinders with uniform density ($\rho_{Al} = 2700\, kg/m^3$).

Parameter	**Symbol**	**Value**
Mass [kg]	m_0	500
Dimensions [m]	$[l_x, l_y, l_z]$	[1, 1, 1]
Inertia matrix [kg·m^2]	$\boldsymbol{I}_0$	$\begin{bmatrix} 83.3 & 0 & 0 \\ 0 & 83.3 & 0 \\ 0 & 0 & 83.3 \end{bmatrix}$

Table 4.1: Chaser physical parameters

Link ID	Parameter	Symbol	Value
Link #1	**Mass** [kg]	m_1	6.36
	Radius [m]	r_1	0.05
	Length [m]	l_1	0.30
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_1$	diag(0.052, 0.052, 0.008)
Link #2	**Mass** [kg]	m_2	14.84
	Radius [m]	r_2	0.05
	Length [m]	l_2	0.70
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_2$	diag(0.615, 0.615, 0.019)
Link #3	**Mass** [kg]	m_3	14.84
	Radius [m]	r_3	0.05
	Length [m]	l_3	0.70
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_3$	diag(0.615, 0.615, 0.019)
Link #4	**Mass** [kg]	m_4	6.36
	Radius [m]	r_4	0.05
	Length [m]	l_4	0.30
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_4$	diag(0.052, 0.052, 0.008)
Link #5	**Mass** [kg]	m_5	4.24
	Radius [m]	r_5	0.05
	Length [m]	l_5	0.20
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_5$	diag(0.017, 0.017, 0.005)
Link #6	**Mass** [kg]	m_6	4.24
	Radius [m]	r_6	0.05
	Length [m]	l_6	0.20
	Inertia matrix [kg·m^2]	$\boldsymbol{I}_6$	diag(0.017, 0.017, 0.005)

Table 4.2: Manipulator physical parameters: both limbs are identical

The limbs of the manipulator are identical in terms of physical parameters, thus the values in Table 4.2 are intended for both of them. A distinction, instead, must be made for what concerns their mounting on the *Chaser*'s base: this results in a different set of *DH parameters*, as shown in Table 4.3 and 4.4, and in two different $\boldsymbol{T}_{J_1}^{J_0}$ matrices, reported in Equation 4.1.

Joint ID	c [m]	α [deg]	d [m]
Joint #2	0	90	0.3
Joint #3	0.7	0	0
Joint #4	0.7	0	0
Joint #5	0.3	-90	0
Joint #6	0.2	0	0
Joint EE	0.2	0	0

Table 4.3: DH parameters for the *limb A*

Joint ID	c [m]	α [deg]	d [m]
Joint #2	0	90	0.3
Joint #3	0.7	0	0
Joint #4	0.7	0	0
Joint #5	0.3	90	0
Joint #6	0.2	0	0
Joint EE	0.2	0	0

Table 4.4: DH parameters for the *limb B*

$$
{}^{A}\boldsymbol{T}_{J_1}^{J_0} = \begin{bmatrix} 0 & 0 & 1 & 0.5 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{B}\boldsymbol{T}_{J_1}^{J_0} = \begin{bmatrix} 0 & 0 & -1 & -0.5 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}
$$

Each joint has an upper and lower boundary for its rotation, reported in Table 4.5. These values are identical for the two limbs and, thanks to the particular choice of *DH parameters* and $\boldsymbol{T}_{J_1}^{J_0}$ matrices, the same set of six joint angles allows to have the two limbs in a specular configuration relative to the *Chaser*.

Joint ID	Lower Limit [deg]	Upper Limit [deg]
Joint #1	-170	170
Joint #2	-170	170
Joint #3	-170	170
Joint #4	-170	170
Joint #5	-170	170
Joint #6	-170	170

Table 4.5: Upper and lower limits of joints' angles

The maximum and minimum allowed joint velocities and accelerations (which were considered identical for each joint) are instead reported in Table 4.6.

Parameter	Lower Limit	Upper Limit
Joint rate [deg/s]	-20	20
Joint acceleration [deg/s^2]	-6	6

Table 4.6: Upper and lower limits of joints' angular rates and accelerations

Target Satellite and Grasping Points

In Figure 4.2 is shown the 3D model of the *Target* in its body-fixed coordinate system.

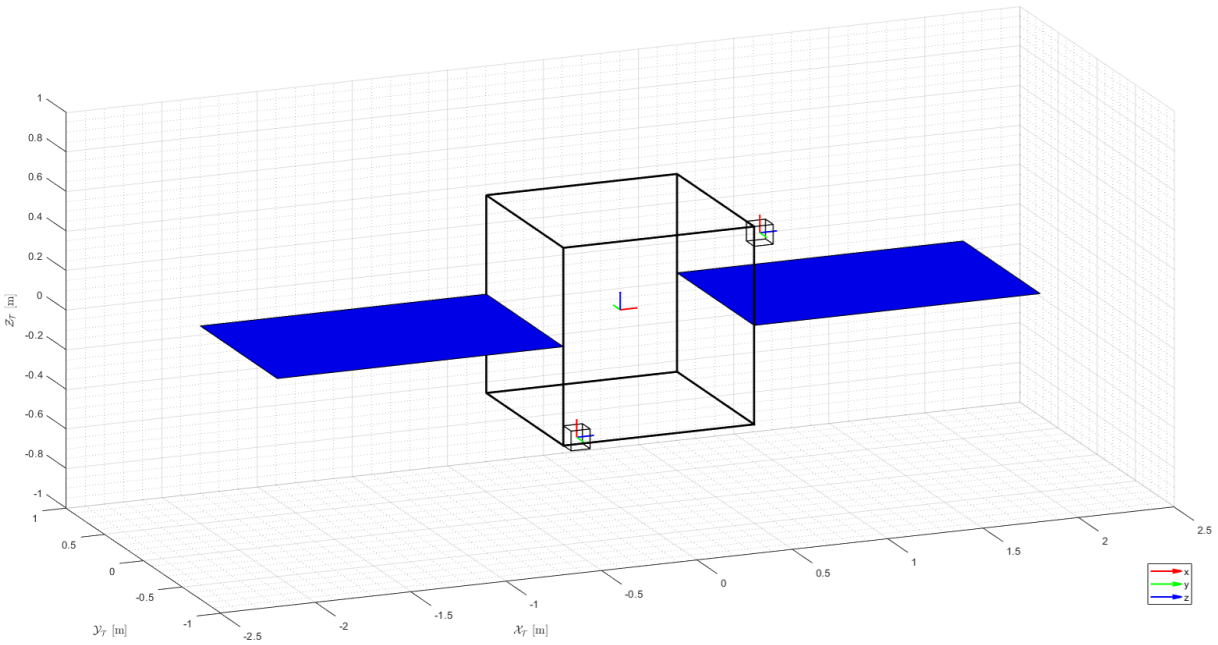

Figure 4.2: 3D model of Target satellite

Since it was assumed that the *Target* was in a certain control regime (for example, nadir-pointing) that caused its attitude to be fixed relative to the *LVLH*, there was no necessity to define neither its mass nor the inertia, but only its dimensions to finally be able to model the obstacles (in the way explained in Chapter 3) and set the position of the two grasping points.
As it can be seen in Figure 4.2, the *Target* is assumed to have two solar panels, which are treated as obstacles in the simulation.

Table 4.7 summarizes the geometries of the bodies constituting the *Target* together with a safety factor, in order to be conservative in the process of obstacles modeling.

Component	**Dimensions** $[l_x, l_y, l_z]$ [m]	**Safety Factor**
Base	[1, 1, 1]	[5%, 5%, 5%]
Solar Panels	[1.5, 1, 0.075]	[5%, 10%, 100%]

Table 4.7: Target physical parameters

For what concerns instead the two grasping points (highlighted by the two small cubes in Figure 4.2), their position and orientation with respect to the *Chaser*'s body-fixed frame are defined by the two transformation matrices reported in Equation 4.2.

$$\boldsymbol{T}_{GP_1}^{\mathcal{B}_C} = \begin{bmatrix} 0 & 0 & 1 & 0.55 \\ 0 & -1 & 0 & -0.45 \\ 1 & 0 & 0 & 0.45 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{GP_2}^{\mathcal{B}_C} = \begin{bmatrix} 0 & 0 & 1 & -0.45 \\ 0 & -1 & 0 & -0.55 \\ 1 & 0 & 0 & -0.45 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

4.1.2. Simulation Parameters

First of all, the position and orientation weights defined in Section 2.3.3 were set to be $w_p = 100$ and $w_o = 30$, respectively; this choice showed to be a good compromise to finally retrieve a solution having satisfactory accuracy both in position and orientation.

Then, the chosen values of the free parameters which characterize the RRT* algorithm, introduced in Chapter 3, are reported in Table 4.8.

Parameter	Symbol	Value
Maximum number of iterations	$\texttt{iter}_{\texttt{max}}$	25000
Steering distance [deg]	δ	18
Neighbors radius [deg]	R_{search}	20
Goal radius [deg]	R_{goal}	25

Table 4.8: RRT* algorithm parameters

For what concerns instead the values of the controllers' gain matrices defined in Section 2.5, a distinction is made regarding the different phases of the capture process, as shown in Table 4.9:

Phase	Controlled Variable	$\boldsymbol{K}_p$	$\boldsymbol{K}_d$
Slew Maneuver	**Joints**	$\begin{bmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 80 \end{bmatrix}$	$\begin{bmatrix} 107.3 & 0 & 0 \\ 0 & 107.3 & 0 \\ 0 & 0 & 107.3 \end{bmatrix}$
	Chaser Attitude	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix}$
Target Capture	**Joints**	$\begin{bmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 80 \end{bmatrix}$	$\begin{bmatrix} 107.3 & 0 & 0 \\ 0 & 107.3 & 0 \\ 0 & 0 & 107.3 \end{bmatrix}$
	Chaser Attitude	$\begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$	$\begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$

Table 4.9: Gain matrices for the controllers

Lastly, the initial conditions are defined: it must be recalled that the *Chaser* is initially along the same orbit of the *Target*, but its initial attitude may not optimal to perform the capture.
Both the satellites and the manipulator's limbs are considered to be initially still; in particular, for the *Target* this condition persists along the whole simulation, as it was assumed to be collaborative. The initial conditions for all the variables concerning the *Chaser* and the *Manipulator* are instead reported in Table 4.10, where they are all expressed relative to the inertial coordinate system. Note that the initial values of all the parameters related

to the two limbs are the same and therefore they are reported only once in the table.

Parameter	Symbol	Value
Initial base position [m]	$\boldsymbol{r}_0\big\vert_{t=0}$	$[0, -2, 0]^T$
Initial base Euler angles [deg]	$(\varphi, \theta, \psi)\big\vert_{t=0}$	$[29, 21, -34]^T$
Initial base linear velocity [m/s]	$\dot{\boldsymbol{r}}_0\big\vert_{t=0}$	$[0, 0, 0]^T$
Initial base angular velocity [deg/s]	$\boldsymbol{\omega}_0\big\vert_{t=0}$	$[0, 0, 0]^T$
Initial joints angles [deg]	$\boldsymbol{q}\big\vert_{t=0}$	$[0, 0, 150, 90, 0, 90]^T$
Initial joints rates [deg/s]	$\dot{\boldsymbol{q}}\big\vert_{t=0}$	$[0, 0, 0, 0, 0, 0]^T$

Table 4.10: Initial conditions for the numerical simulation

Finally, the considered initial capture scenario is depicted in Figure 4.3 with respect to the inertial frame, whose origin coincides with the *Target*'s center of mass:

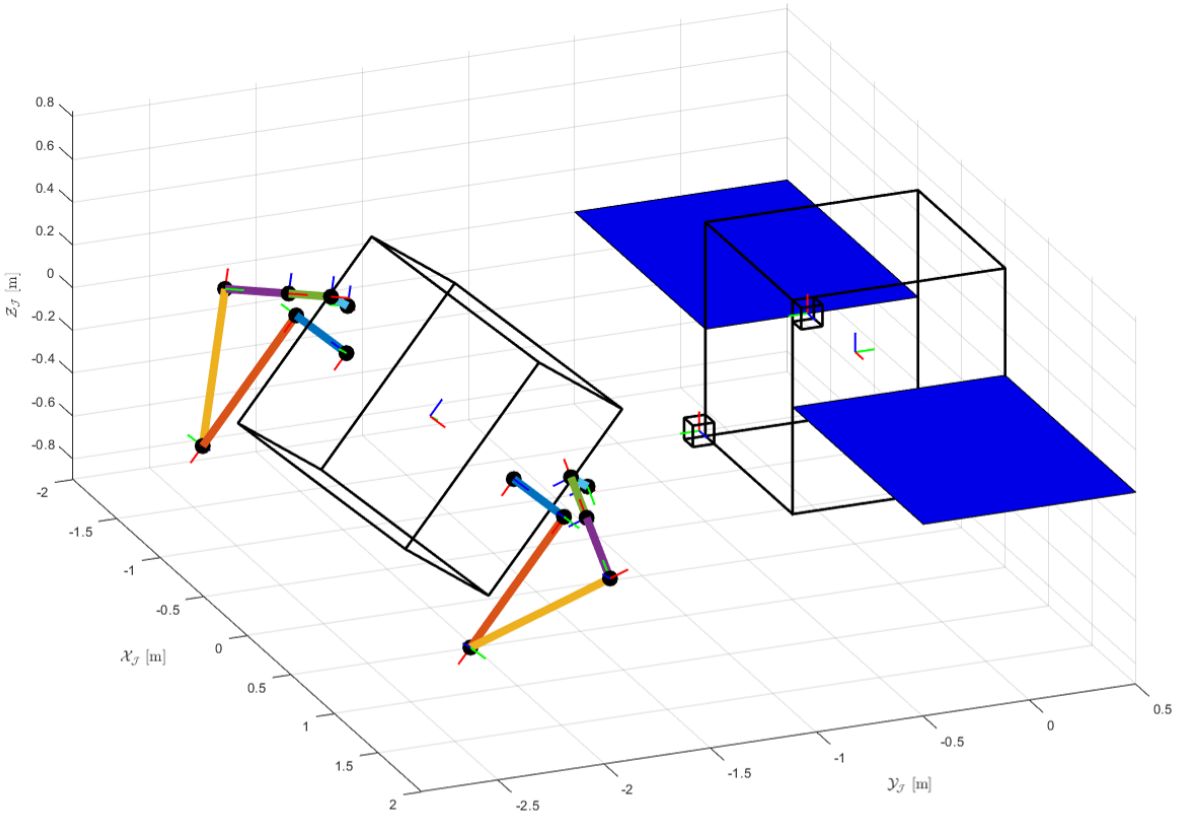

Figure 4.3: Initial condition (limbs in their stowed configuration)

4.2. Results

In this section the results of a reference numerical simulation, performed in a *Matlab-Simulink*® environment, are presented and discussed. Firstly, the initial slew maneuver is analyzed: this has the main objective of reorienting the *Chaser* toward a favorable attitude for the capture process, as well as to bring the two limbs from the initial stowed pose to their *home configuration*. Afterwards, the *Target*'s capture process is examined. Lastly, some considerations on the RRT* algorithm are discussed.

4.2.1. Slew Maneuver

The slew maneuver is considered to be concluded when both the following conditions are verified:

1. The mean values of the Euler angles, used to parametrize the *Chaser*'s attitude, are below a certain threshold (assumed to be [-1°,1°] for each angle) and they do not show any significative variation in time.
2. Both the limbs of the space manipulator have reached their *home configuration*, represented by the set of joint angles $\boldsymbol{q}_{home} = [0, 0, 0, 0, 0, 0]^T$

In this phase, for what concerns the motion of the two limbs, a simple trapezoidal trajectory was chosen to guide them from the initial stowed pose to the final *home configuration*. The joint angles variation in time, as well as the angular rates, are identical for the two limbs (as they perform the same exact trajectory) and are shown in Figures 4.4 and 4.5. Note that, in both figures, the curves corresponding to the variable associated to the 4^{th} and 6^{th} joint are overlapped, since they undergo the same trajectory.

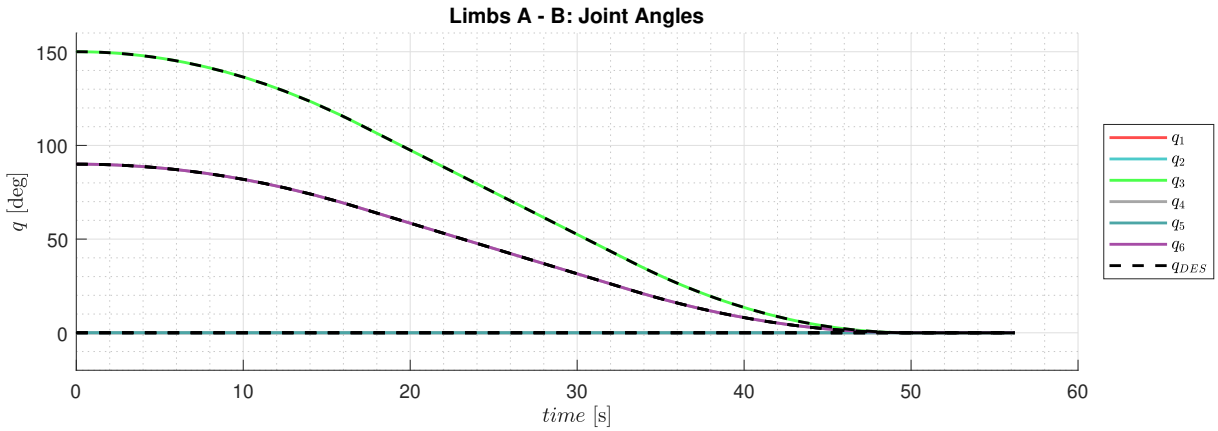

Figure 4.4: Computed vs reference joint angles for both limbs in the slew maneuver

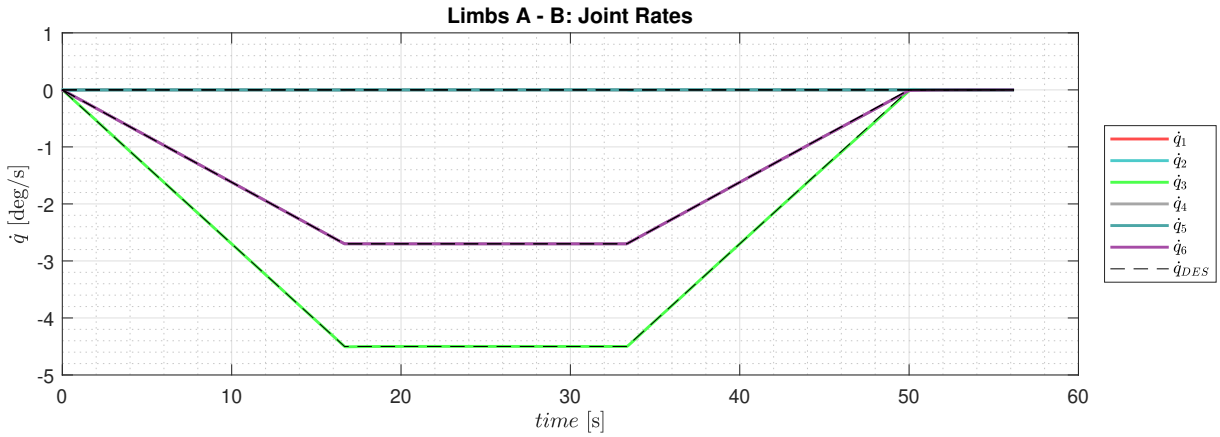

Figure 4.5: Computed vs reference joint rates for both limbs in the slew maneuver

For what concerns instead the *Chaser*'s base, in Figure 4.6 can be observed the time evolution of the three Euler angles, representing its attitude relative to the inertial frame, during the slew maneuver. The rotation is stopped as soon as all three angles fall in the desired range of [-1°,1°]: this occur just after 56 s.

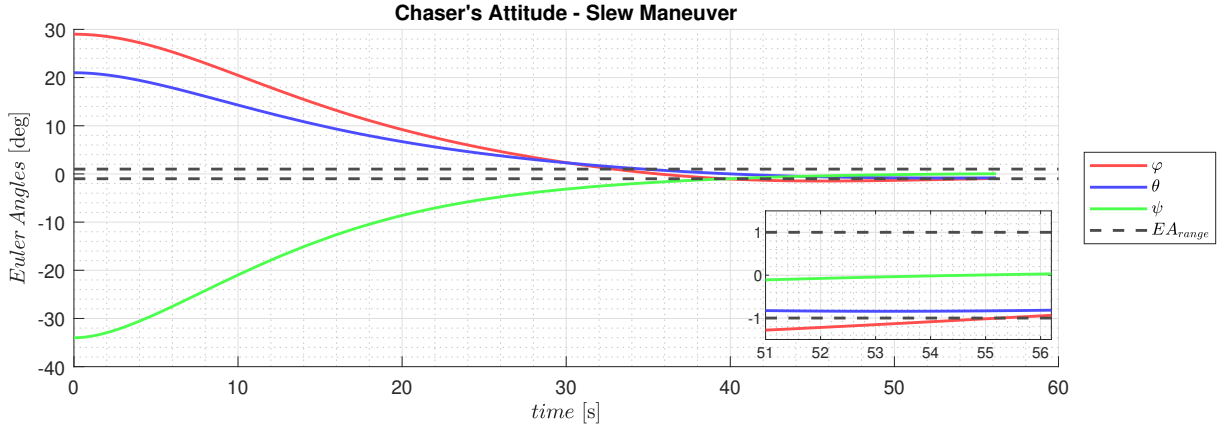

Figure 4.6: Variation of Chaser's attitude during the slew maneuver

Then, the required control torques, needed to be applied to the base perform the slew maneuver, were computed: the results are shown in Figure 4.7. Their value is coherent to the initial choice of Euler angles: two negative torques (meaning that a rotation opposite to the right hand rule is required around that specific axis) are required to bring φ and θ into the admissible range and, as expected, the value of $|\tau_x|$ is slightly greater than $|\tau_y|$. In addition, a positive τ_z is obtained for the same reasons and as expected, it has the highest initial absolute value.

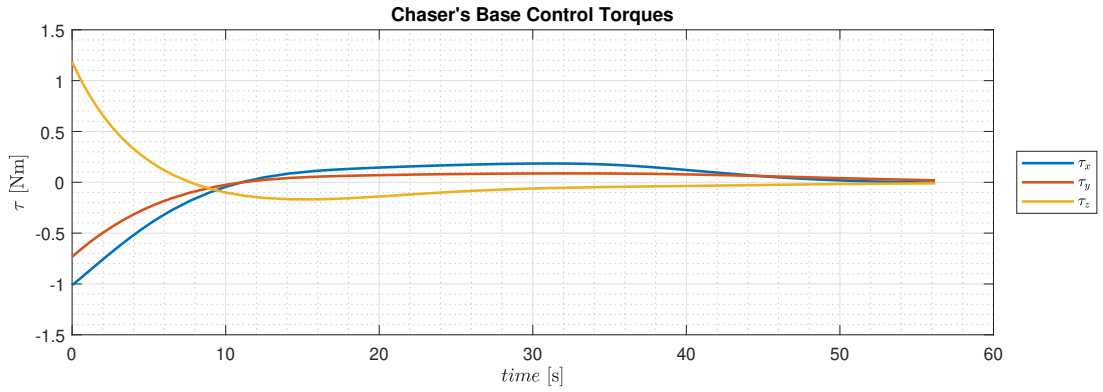

Figure 4.7: Control torques for the Chaser's base in the slew maneuver

The overall slew maneuver is depicted in Figure 4.8: here, it is highlighted the trajectory in space performed by the two end-effectors, as well as the initial (shaded) and final overall configurations.

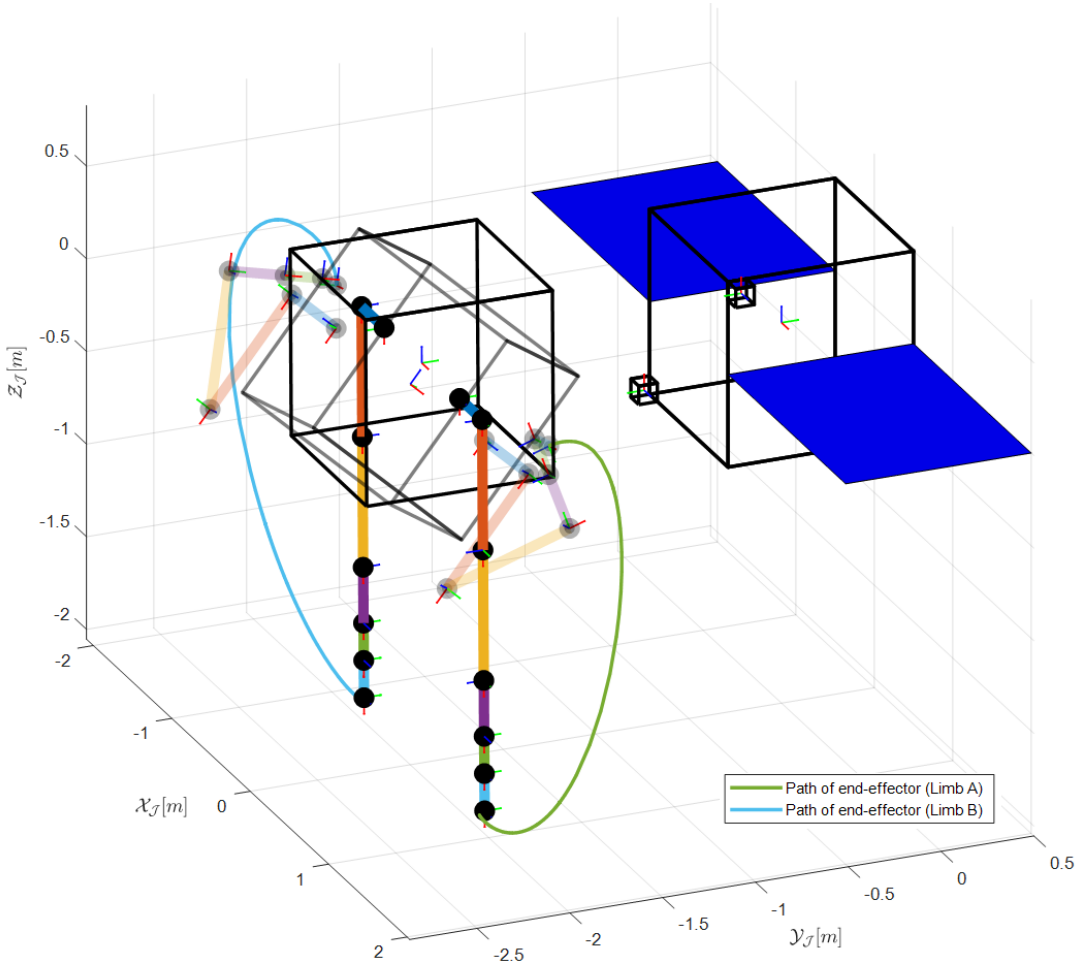

Figure 4.8: Satellites and Manipulator after the slew maneuver

At the end of the slew maneuver, the following transformation matrix, describing the

position and orientation of the *Chaser* relative to the *Target*, is obtained.

$$\boldsymbol{T}_{J_0}^{\mathcal{J}} = \begin{bmatrix} 0.99 & 0.01 & -0.01 & 0.09 \\ -0.01 & 0.99 & 0 & -1.97 \\ 0.01 & 0 & 0.99 & 0.14 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Regarding instead the final values reached by the joint angles and joints rates, it was observed that, for both the limbs, $\boldsymbol{q} \in [10^{-7}, 10^{-5}]$ deg and $\dot{\boldsymbol{q}} \in [10^{-8}, 10^{-6}]$ deg/s. Thus, for what concerns the initial conditions of the *Target*'s capture phase, all these variables were assumed to be zero.

4.2.2. Satellite Capture

The final phase analyzed is the actual capture of the *Target* satellite, where the limbs are guided toward the two predefined grasping points by the trajectories generated using the RRT* algorithm. The results obtained for each joint of the limb A are shown in Figures from 4.9 to 4.14, while the limb B counterpart is shown in Figures from 4.15 to 4.20.

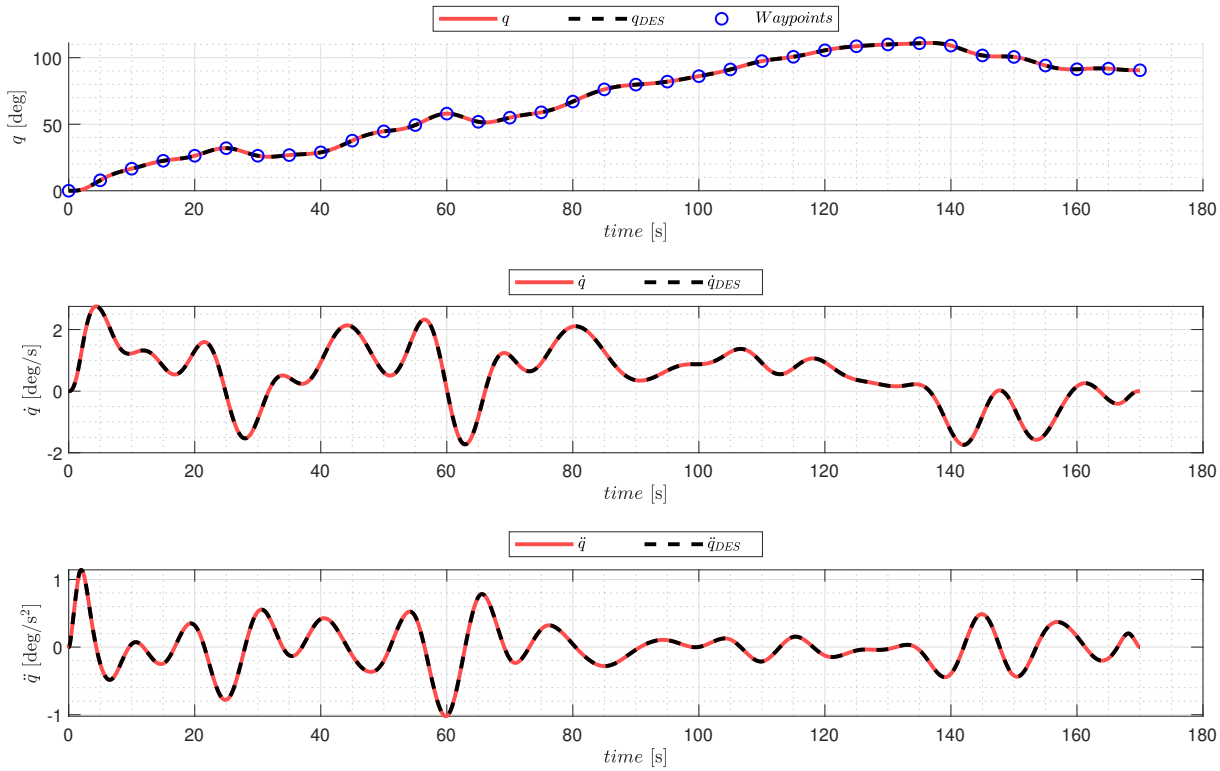

Figure 4.9: 1^{st} joint trajectory (Limb A)

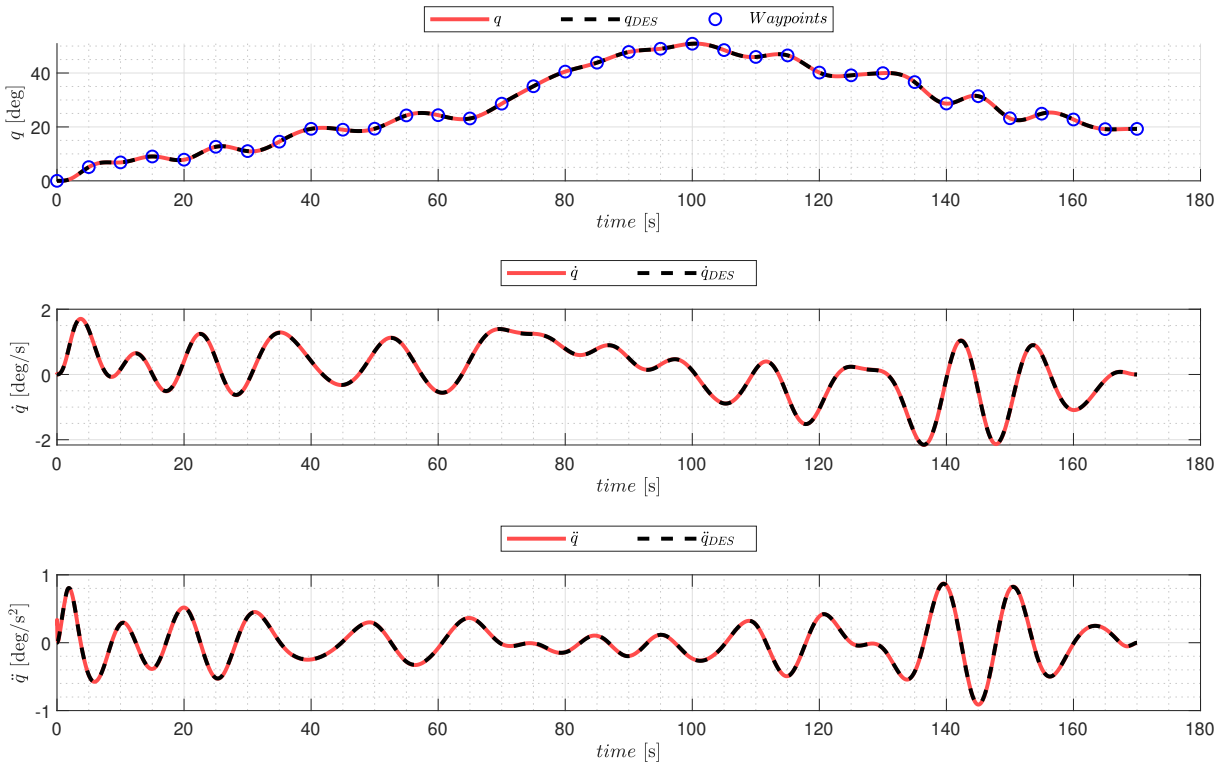

Figure 4.10: 2^{nd} joint trajectory (Limb A)

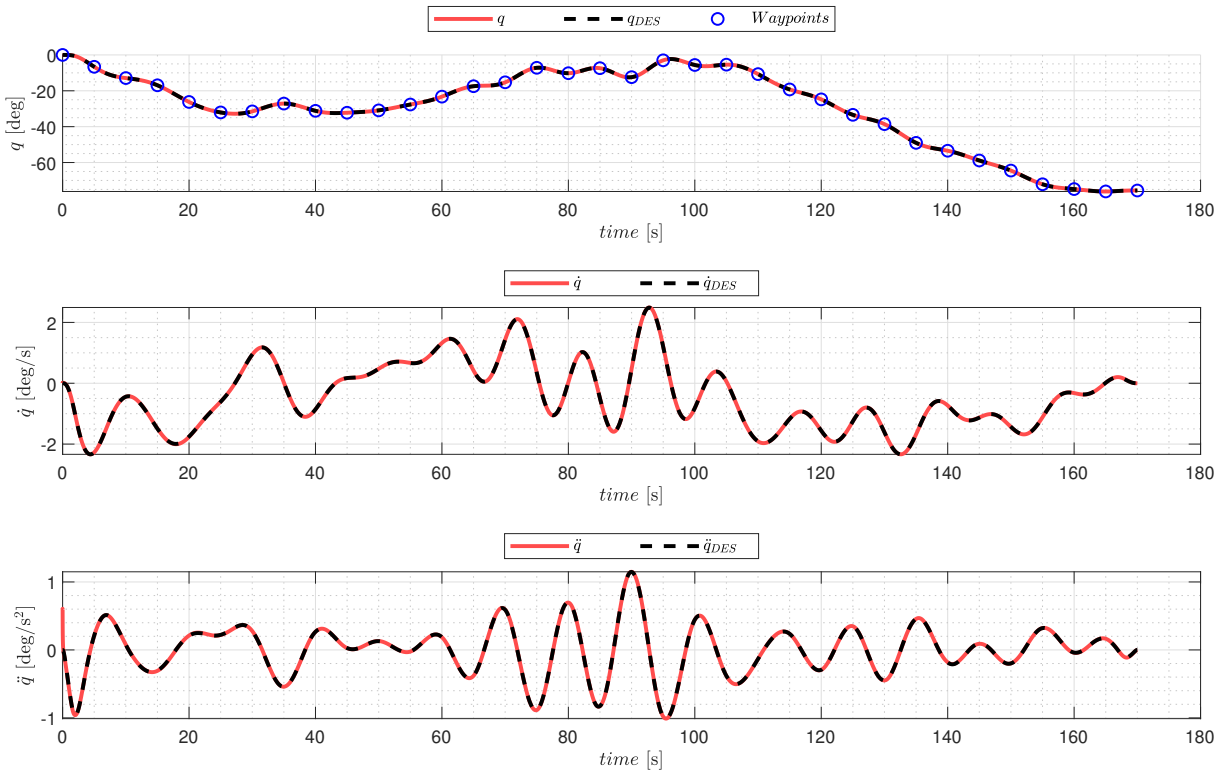

Figure 4.11: 3^{rd} joint trajectory (Limb A)

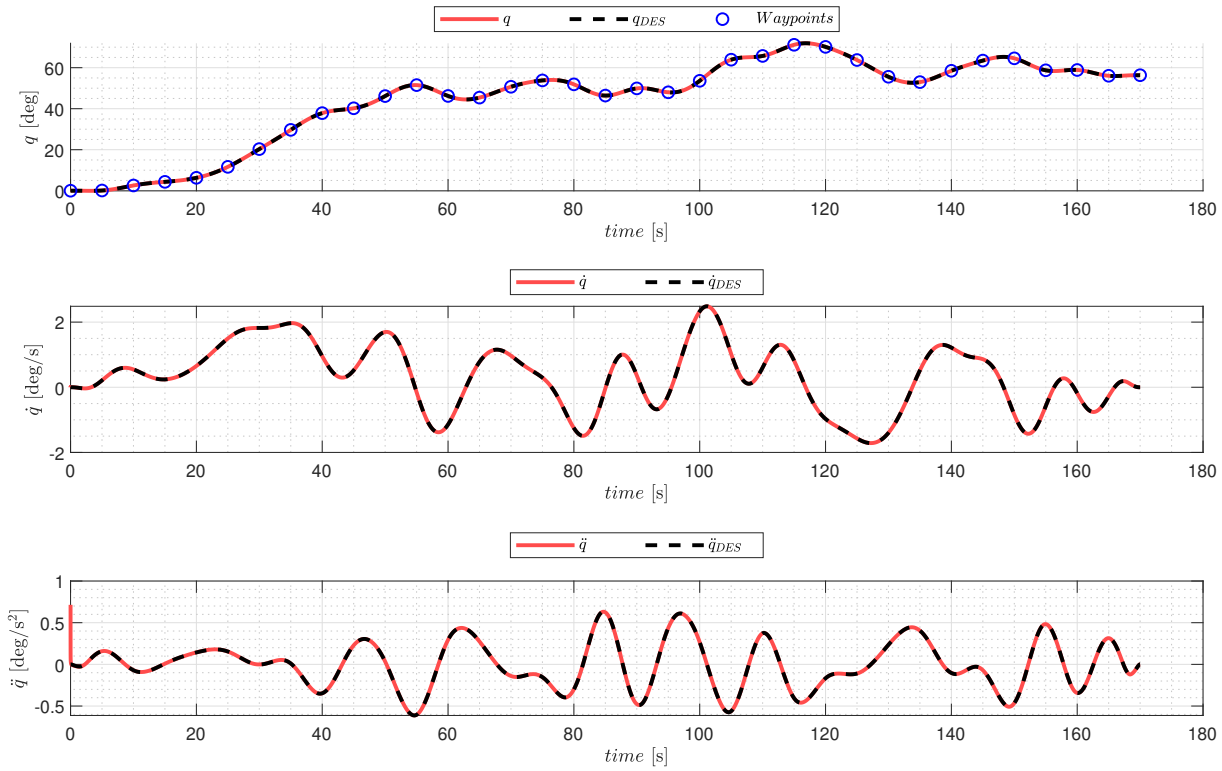

Figure 4.12: 4^{th} joint trajectory (Limb A)

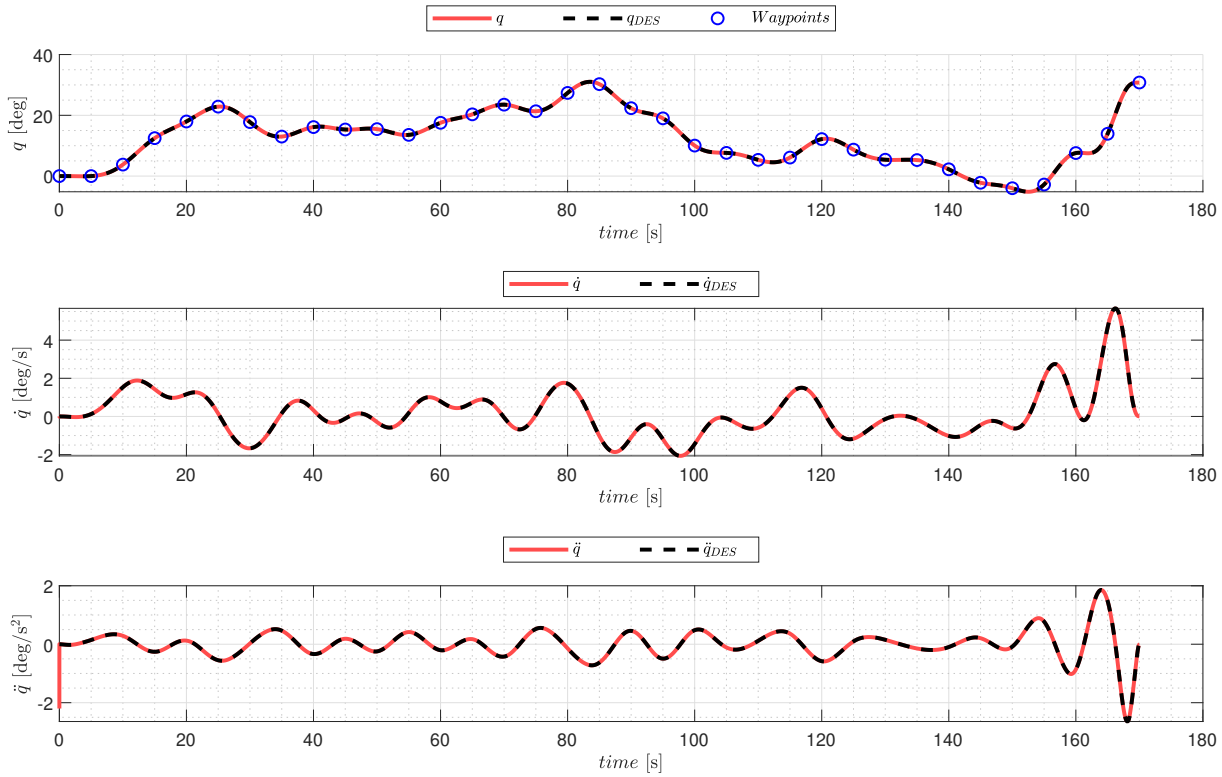

Figure 4.13: 5^{th} joint trajectory (Limb A)

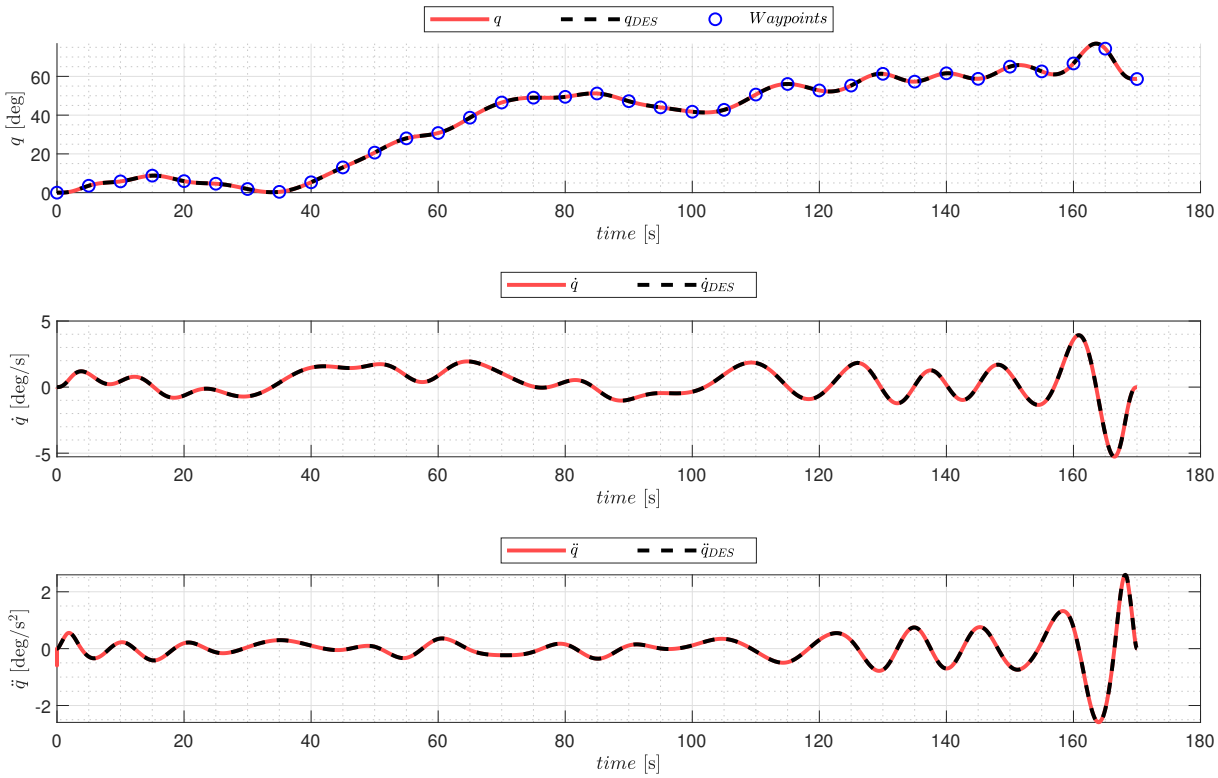

Figure 4.14: 6^{th} joint trajectory (Limb A)

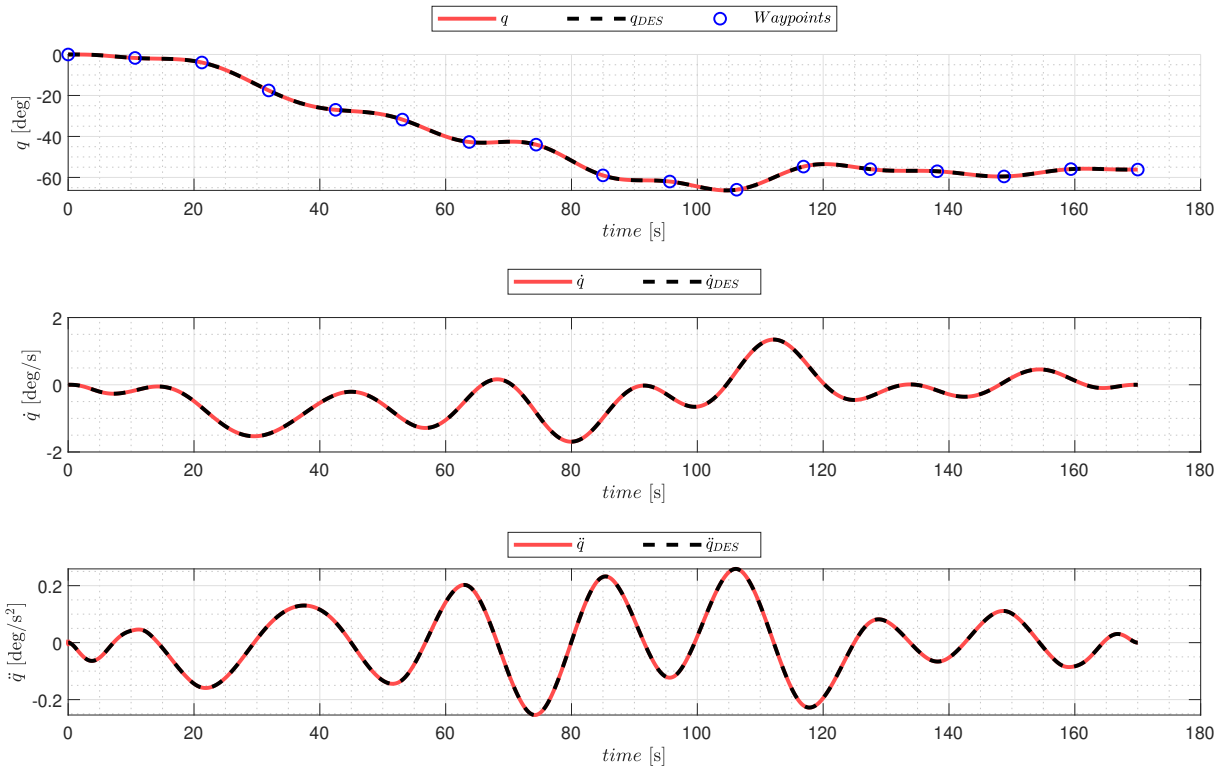

Figure 4.15: 1^{st} joint trajectory (Limb B)

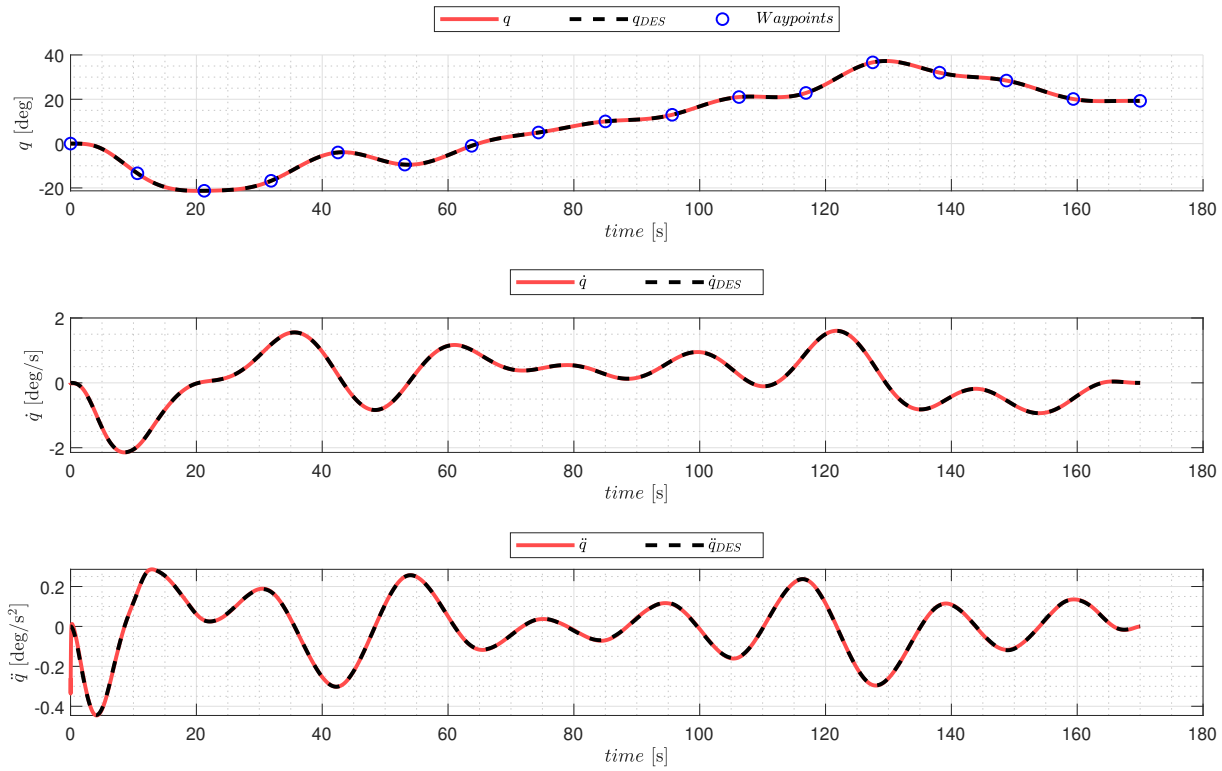

Figure 4.16: 2^{nd} joint trajectory (Limb B)

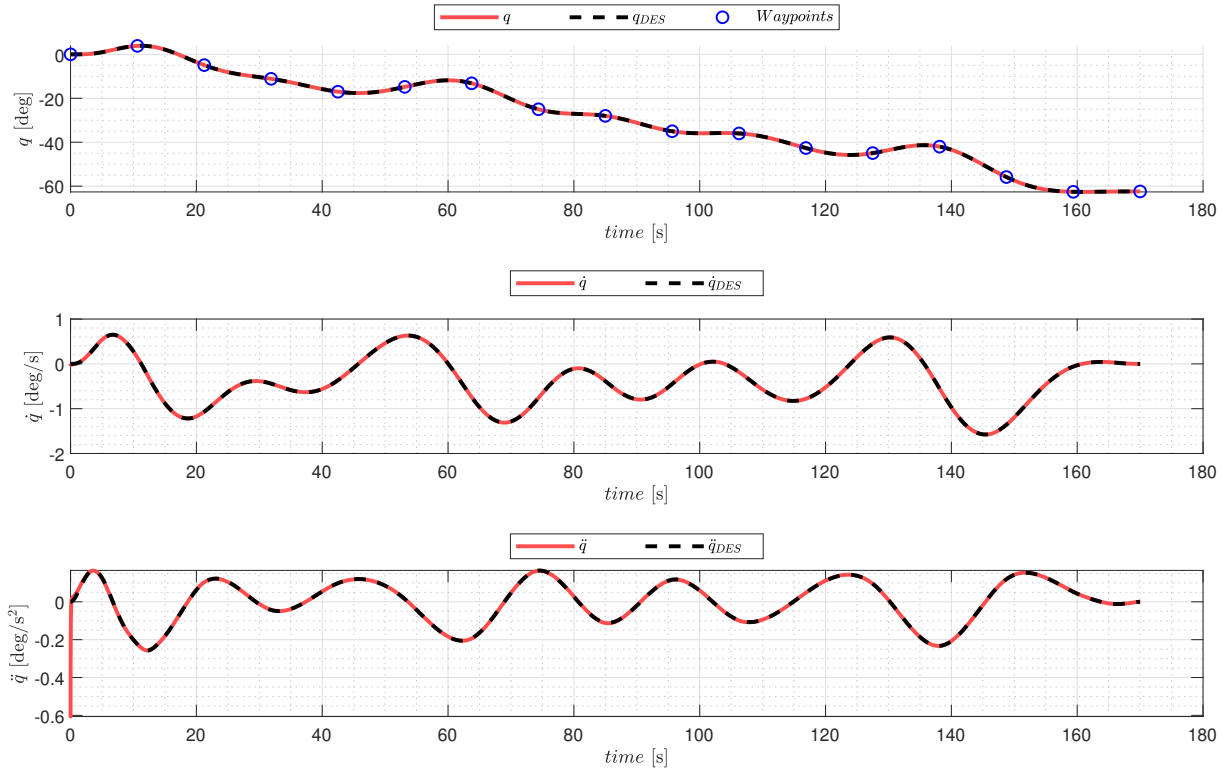

Figure 4.17: 3^{rd} joint trajectory (Limb B)

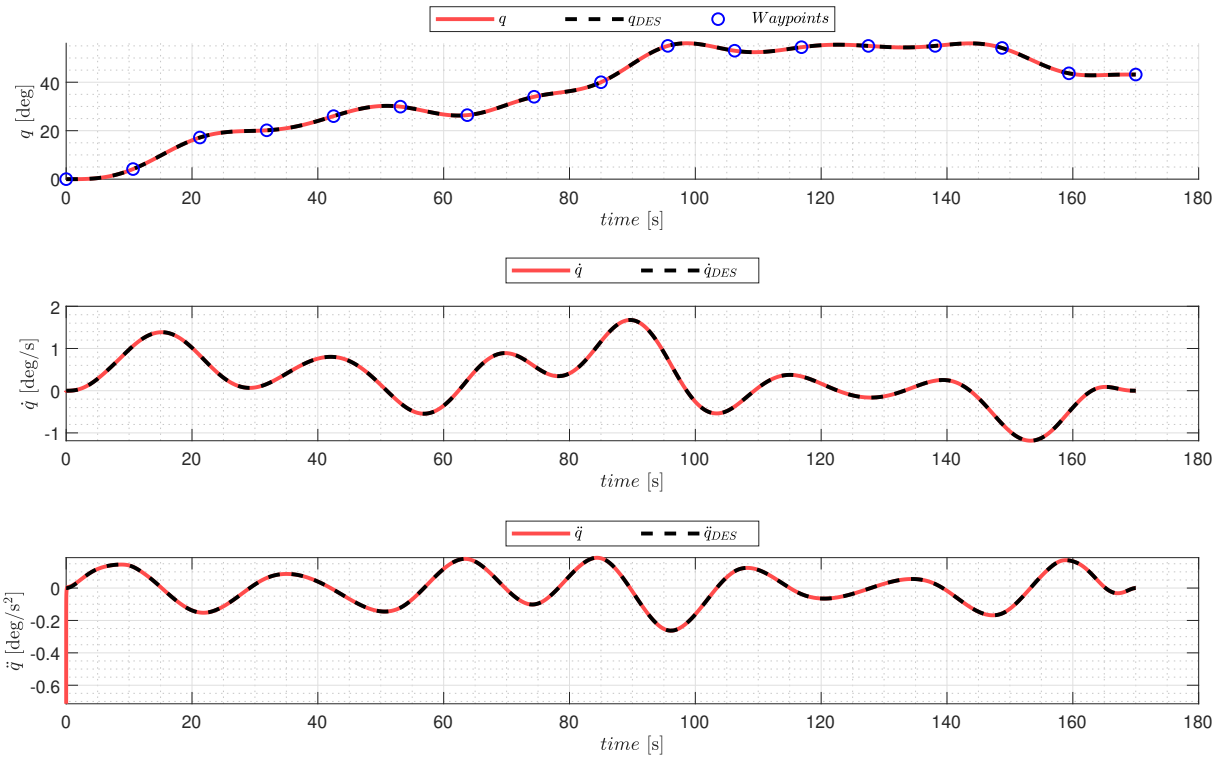

Figure 4.18: 4^{th} joint trajectory (Limb B)

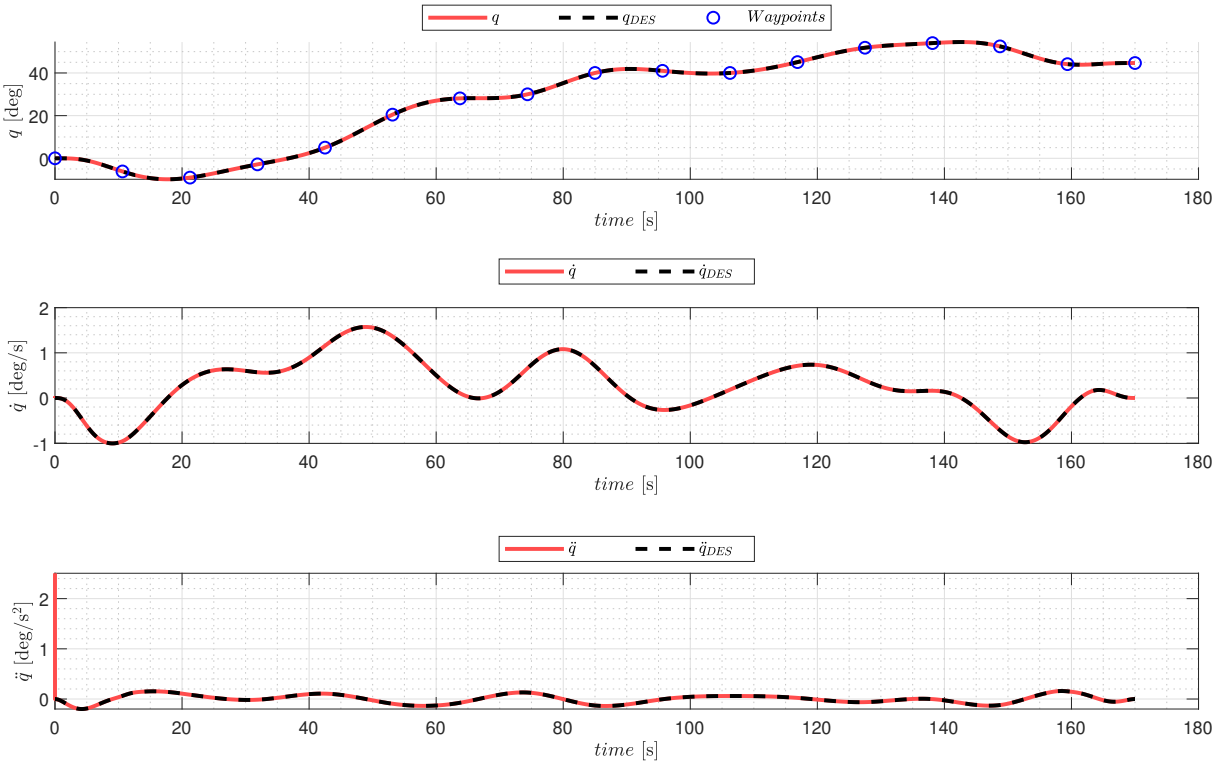

Figure 4.19: 5^{th} joint trajectory (Limb B)

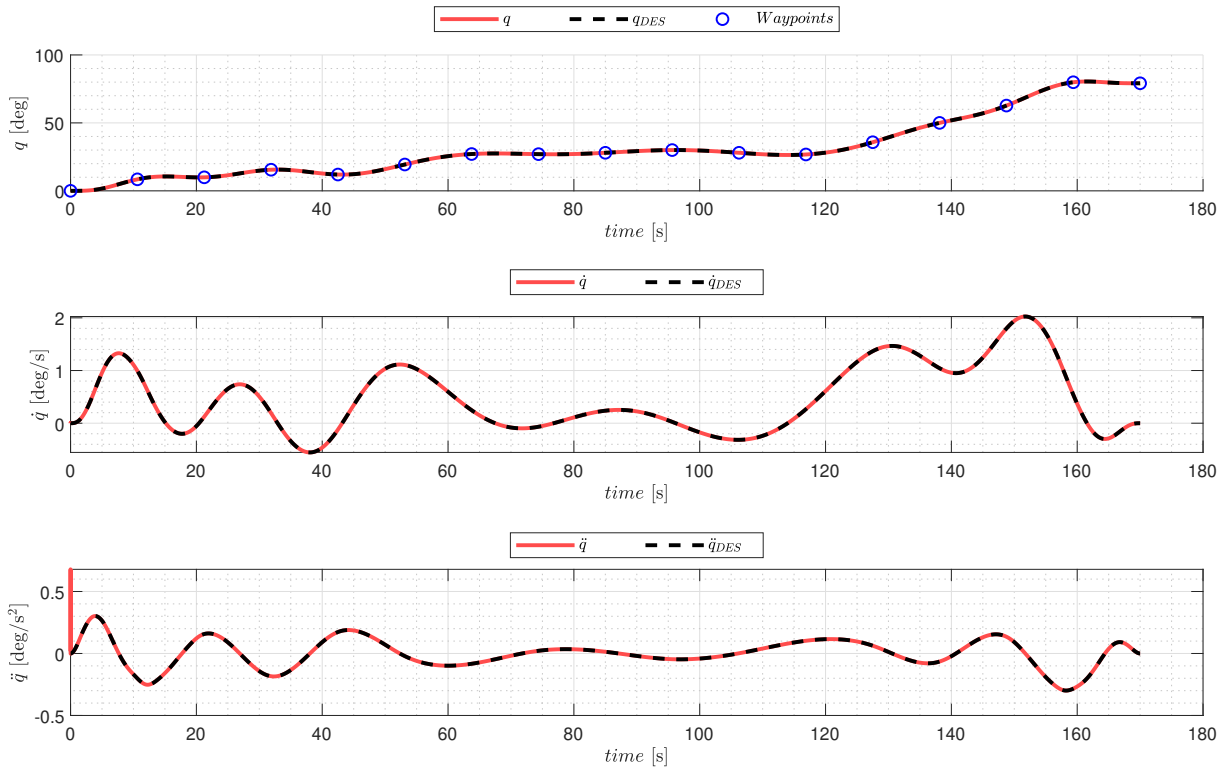

Figure 4.20: 6^{th} joint trajectory (Limb B)

As it can be seen, the resulting trajectories are relatively smooth in time and both the angular rates and accelerations are way below their maximum allowed value for every joint. This is because it was preferred to increase the maneuver time rather than exploiting the joints performances at their maximum, also because a more gentle movement of the manipulator induces relatively small variations in position and attitude of the *Chaser*, simplifying the task of the base's attitude controller.

The 'spikes' observed in some of the previous acceleration profiles are due to an overshoot of the controller; however, as shown, the reference acceleration is accurately followed right after. In general, though, the results provided by the RRT* algorithm and by the joints controllers are satisfactory: quite smooth trajectories are generated for all the joints, thus avoiding dangerous sudden changes in the manipulator configuration, and negligible tracking errors, in the order of 10^{-2} deg, are obtained for all the profiles.

The associated joints control torques that were generated by the controllers are shown in Figure 4.21 for the Limb A, while in Figure 4.22 for Limb B.

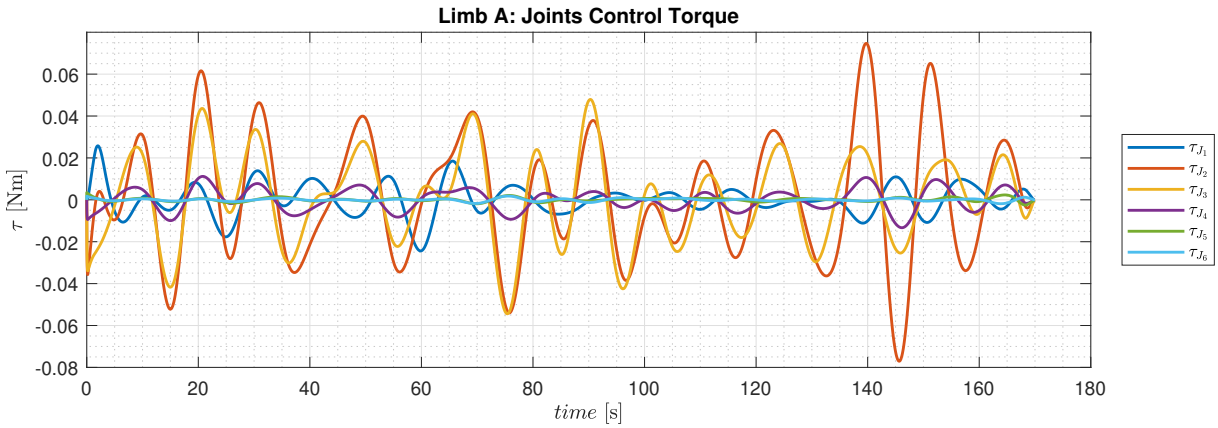

Figure 4.21: Control torques for the joints of Limb A

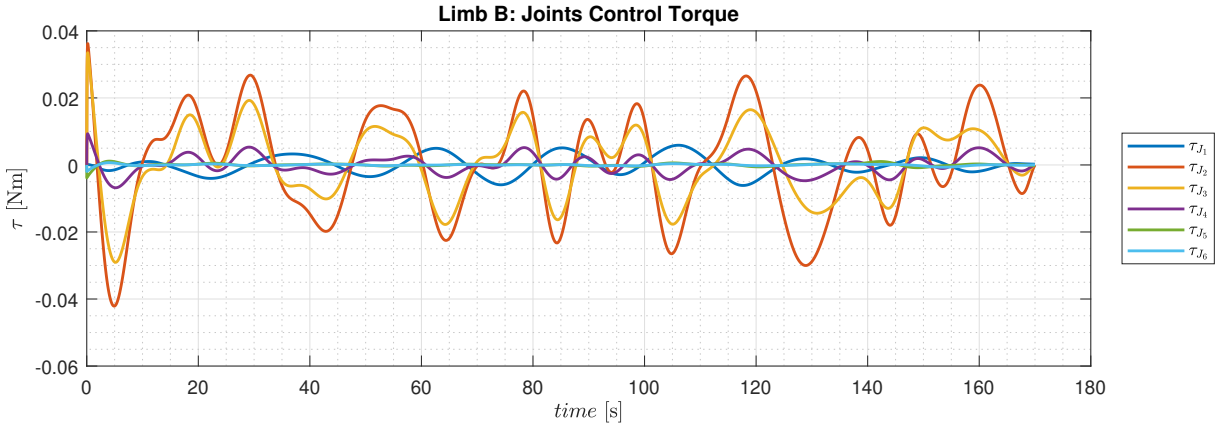

Figure 4.22: Control torques for the joints of Limb B

Then, as it can be observed in Figure 4.23, the attitude variation of the *Chaser* is kept, by the controller, well below the predefined admissible range of [-1°,1°] for each Euler angle. Note that, here, the initial condition corresponds to the final one shown in Figure 4.6.

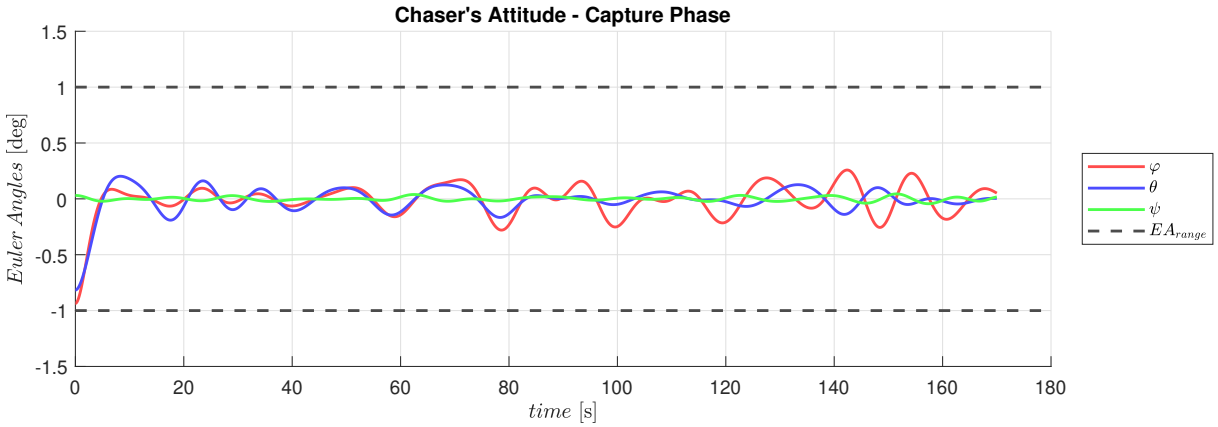

Figure 4.23: Variation of Chaser's attitude during the Target's Capture phase

The associated base control torques are shown in Figure 4.24: their behavior is again expected as it reflects the same considerations made for Figure 4.7 in the previous section.

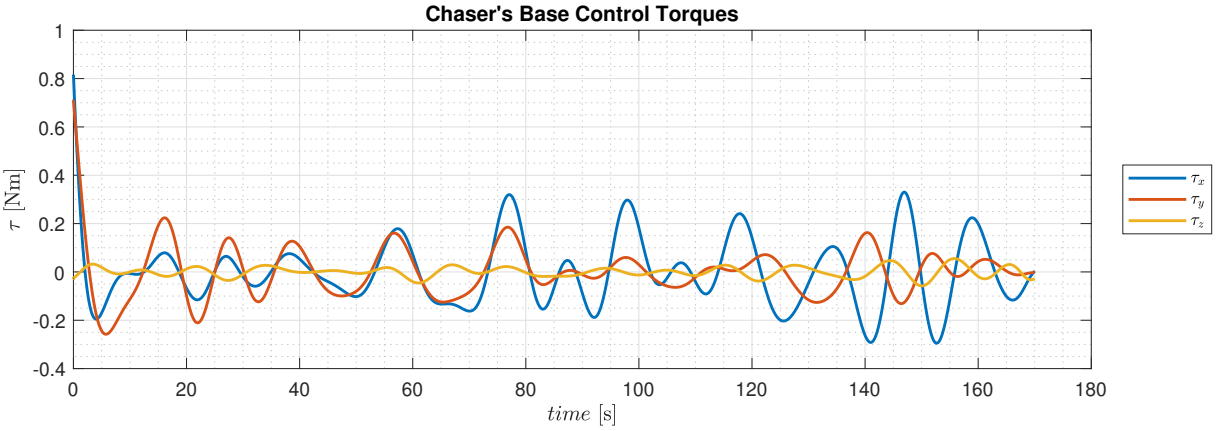

Figure 4.24: Control torques for the Chaser's base in the Target's Capture phase

Regarding then the overall performances of the model, in Table 4.11 are reported the position and orientation errors ($\Delta \boldsymbol{r}_{EE}$ and $\Delta \boldsymbol{\alpha}_{EE}$, respectively) of the end-effectors of the two limbs. In particular, it is highlighted the effect produced by the *Trajectory Correction* process explained at the end of Chapter 3.

End-Effector	Error Type	Pre-Correction	Post-Correction
A	$\Delta \boldsymbol{r}_{EE}$ [cm]	$[1.86, -2.03, 2.28]^T$	$[-0.71, 0.22, -0.05]^T$
	$\Delta \boldsymbol{\alpha}_{EE}$ [deg]	$[0.11, 1.29, 1.30]^T$	$[0.03, 0.35, 0.34]^T$
B	$\Delta \boldsymbol{r}_{EE}$ [cm]	$[3.20, 0.23, 0.60]^T$	$[-0.90, -0.35, 0.21]^T$
	$\Delta \boldsymbol{\alpha}_{EE}$ [deg]	$[0.88, 1.46, 1.70]^T$	$[0.22, 0.31, 0.39]^T$

Table 4.11: Position and orientation errors of the two end-effectors (Pre vs Post trajectory correction)

As it can be seen, the already good performances of the model are even increased after the correction, thus ensuring the accuracy required for this kind of on-orbit servicing missions.

Lastly, the time durations of the Slew Maneuver and of the actual *Target*'s Capture are reported in Table 4.12.

Phase	Duration [s]
Slew Maneuver	56
Target Capture	170
Total	232

Table 4.12: Duration of the whole capture process

As it can be seen, the time required for the whole maneuver is sufficiently low in order not to challenge the model's hypothesis of not considering major orbital mechanics effects.

The 3D representation of the *Target*'s capture maneuver is finally depicted in Figure 4.25, where the initial, intermediate and final system's configurations are highlighted.

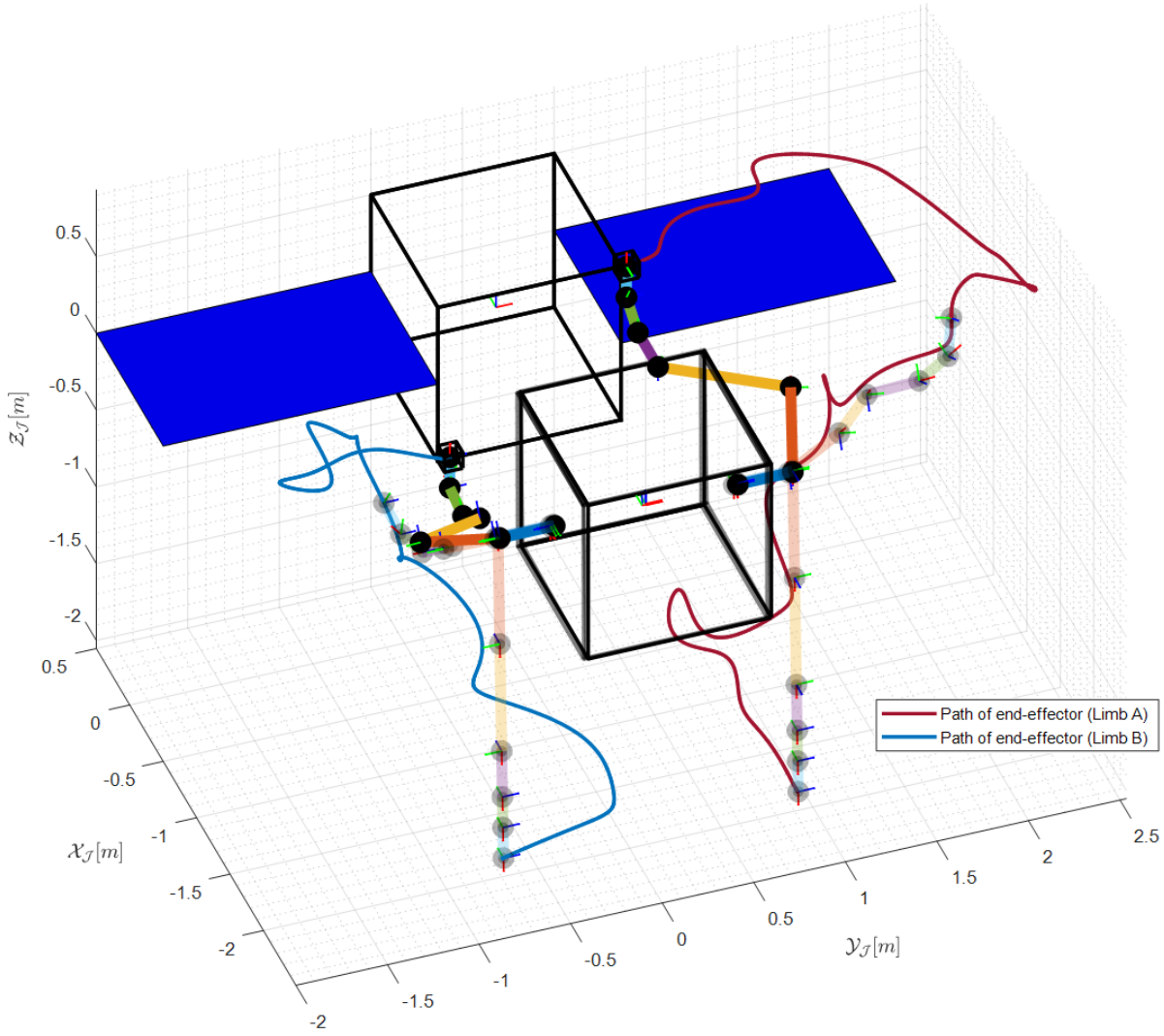

Figure 4.25: Target's capture maneuver, highlighting initial, intermediate and final configuration

Some final considerations can be then made regarding the RRT* algorithm itself. Due to the fact that the algorithm is based on a configuration space that is six dimensional, it was not physically possible to show, in a similar way to what was done for Figures 3.4 and 3.5, the resulting structure of the tree with the final path highlighted. Nevertheless, by only focusing on three degrees of freedom (that could be, for example, the first three joint angles of one manipulator's limb), it was possible to obtain useful plots that were used to validate the effectiveness of the proposed method of collision check, particularly in the case of *External Collisions*. Indeed, as it can be observed in Figures 4.26 and 4.27, it is evident the presence of some distinct empty regions of the configuration space, even with a tree constituted by a much smaller number of nodes (2500 in this example) than the one used for the complete trajectory planning (which is reported in Table 4.8). These regions are nothing but the forbidden zones of the configuration space, where no valid nodes can be created because they will result, in this particular example, in a certain collision with the *Chaser*'s base. Therefore, it can be seen that RRT* correctly discards all nodes that were initially generated in these regions.

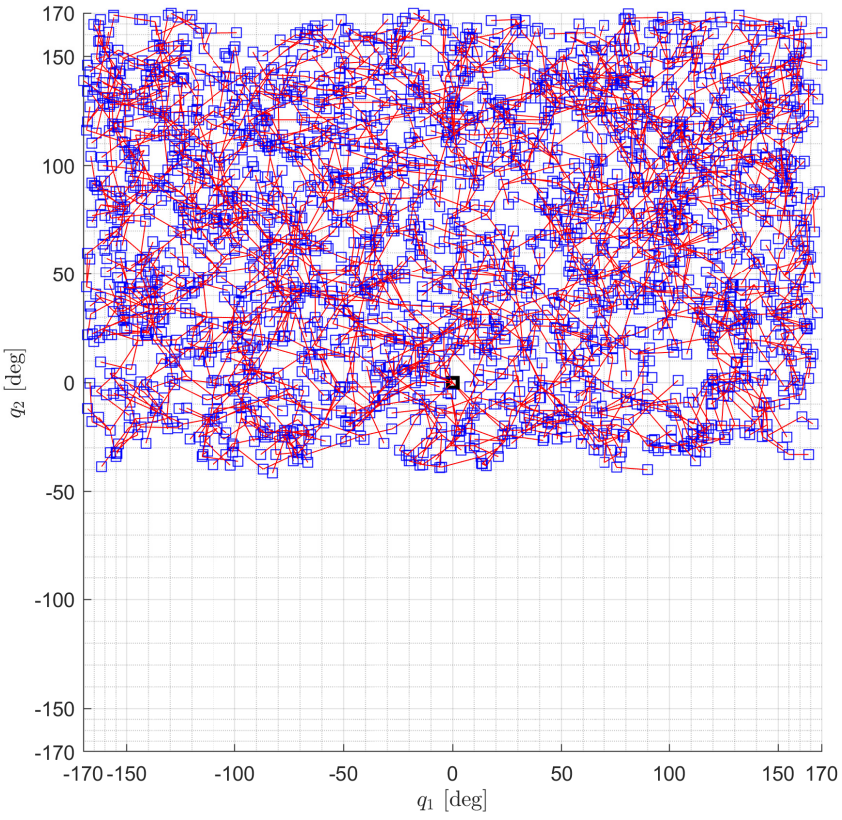

Figure 4.26: Example of Configuration Space with 2500 nodes (view of q_1-q_2 plane)

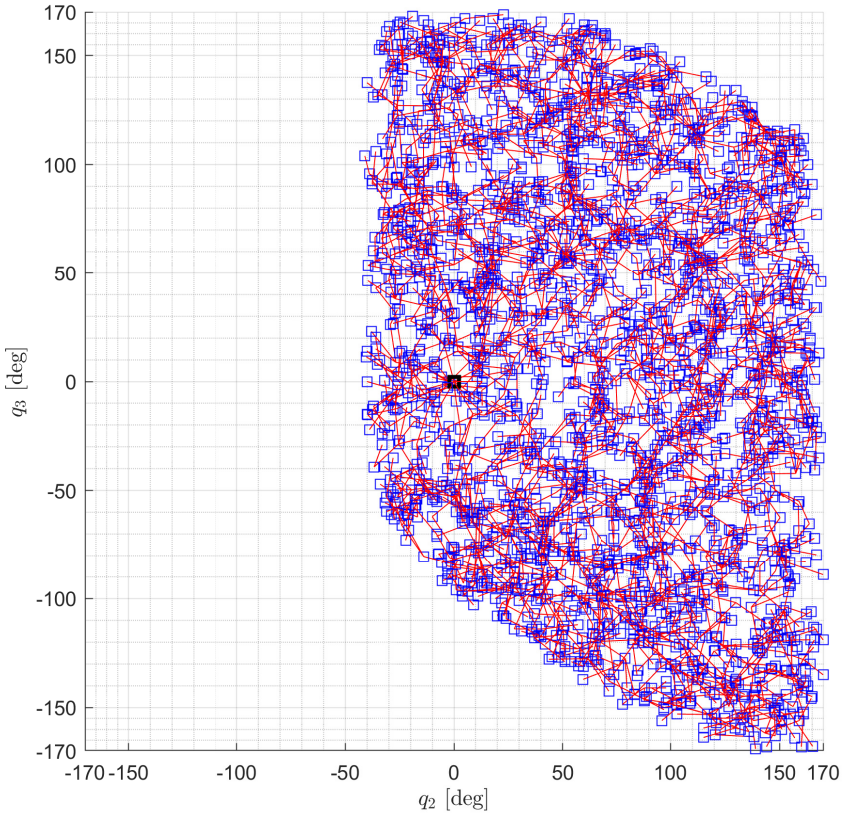

Figure 4.27: Example of Configuration Space with 2500 nodes (view of q_2-q_3 plane)

Concerning then the computational time of the proposed version of the RRT* algorithm, it was noticed that, on average, it was in the range of 4/5 minutes per limb, given the number of maximum iterations reported in Table 4.8. This is mainly due to the the effect that the high number of maximum iterations considered and the particular values chosen for the free parameters of the algorithm had on the *Rewiring* step: indeed, since the value of R_{search} was selected to be greater than δ (in order to increase the chances of generating a more optimal path), an increasingly higher number of tree's rewirings is expected to occur at each iteration of the algorithm, resulting thus in more and more time needed to scan the whole tree in order to find all the 'son' nodes of the rewired-node and to eventually update their cost. However, the high number of iterations considered allowed to obtain a feasible collision-free trajectory for both the limbs almost always on the first attempt, thus without the need of restarting the algorithm when a collision on the retrieved path was detected in the final checks performed after the creation of the timing law.
On the other hand, when a lower number of nodes was selected, the final trajectories, if found, resulted to be, as expected, less optimized with respect to the previous case and

thus some restarts of the algorithm were needed, after the the just mentioned final checks were performed, to finally produce a valid collision-less trajectory for both the limbs. It can be seen, therefore, that computational time that was saved by choosing less iterations to be performed by RRT* was eventually spent by restarting the algorithm multiple times, with the additional effect of generating trajectories that were less optimized with respect to the previous case.

5 | Conclusions

This thesis presents and validates a numerical model whose final aim is to simulate the process of capturing a cooperative satellite by a multi limb space manipulator installed on a servicing spacecraft. The proposed strategy for what concerns the coordinated trajectory planning of the space manipulator consists in a version of the RRT* algorithm specifically designed to operate within the constraints typically involved in this kind of on-orbit servicing missions, like collision avoidance and compliance with physical, kinematics and dynamics constraints of the satellites and the manipulator. Numerical simulations, based on an accurate kinematic and dynamic description of the multibody system, demonstrated the effectiveness of the proposed version of the RRT* algorithm. Indeed, as shown in Chapter 4, very satisfactory results, which ensured, at the same time, compliance with all the constraints previously mentioned, were obtained. Nevertheless, this works presents some limitations, first of all that of not considering orbital mechanics effects during the entire capture maneuver. Therefore, further investigations are needed to extend the validity of the proposed approach, reducing thus the number of hypotheses on which this works is based. Moreover, the RRT* algorithm could be further optimized, resulting in a potential reduction of computational time and effort, especially when dealing with complex problems like the one analyzed in this thesis.

Some of the upgrades that can be made to further improve the quality of the results and to be able to generalize the use of the proposed model to different capture scenarios are listed hereafter:

- Generalize the RRT* algorithm so that is is able to generate trajectories for the space manipulator to capture non-collaborative satellites, that could, for example, be tumbling in an uncontrolled manner.
- Describe in more detail the rendez-vous maneuver, thus considering in the model the change in relative position and attitude between the the two satellites.
- Develop a more robust controller for the attitude of the servicing-spacecraft, especially when taking into account the possible improvements of the model just described.

List of Figures

List of Tables

Bibliography

[1] Telespazio. In orbit servicing, 2024. Accessed: 05/01/2025.

[2] A. Dennison. Northrop grumman eyes 2026 launch of robot-armed satellite servicer, 2024. Accessed: 05/01/2025.

[3] ESA. Clearspace-1, 2024. Accessed: 05/01/2025.

[4] M. Wilde, S. Kwok Choon, A. Grompone, and M. Romano. Equations of motion of free-floating spacecraft-manipulator systems: An engineer's tutorial. *Frontiers in Robotics and AI*, 5, 04 2018.

[5] A. Flores-Abad, M. Ou, K. Pham, and S. Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26, 2014.

[6] S. Dubowsky and E. Papadopoulos. Kinematics, dynamics, and control of free-flying and free-floating space robotic systems. *Robotics and Automation, IEEE Transactions on*, 9, 1993.

[7] B. Liang, Y. Xu, and M. Bergerman. Dynamically equivalent manipulator for space manipulator system. 1. In *Proceedings of International Conference on Robotics and Automation*, 1997.

[8] S.A.A. Moosavian and E. Papadopoulos. On the kinematics of multiple manipulator space free-flyers and their computation. *Journal of Robotic Systems*, 15, 04 1998.

[9] T. Rybus. Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Progress in Aerospace Sciences*, 101:31–48, 2018.

[10] T. Venkata Bhargava and K. Kurien Issac. Minimum time collision-free trajectories for grabbing a non-tumbling satellite. *IFAC-PapersOnLine*, 49(1):142–147, 2016. 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016.

[11] X. Gao, Q. Jia, H. Sun, and G. Chen. Research on path planning for 7-dof space manipulator to avoid obstacle based on a* algorithm. *Sensor Letters*, 9, 2011.

[12] M. Hong, L. Wang, L. Liu, Q. Wang, and Y. Guo. Trajectory planning of a free-floating dual-arm space robot with minimal base disturbance in obstacle environments. *Advances in Space Research*, 2024.

[13] S. Liu, Q. Zhang, and D. Zhou. Obstacle avoidance path planning of space manipulator based on improved artificial potential field method. *Journal of The Institution of Engineers*, 2014.

[14] T. Rybus and K. Seweryn. Application of rapidly-exploring random trees (rrt) algorithm for trajectory planning of free-floating space manipulator. In *10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 91–96, 2015.

[15] D. Lee and H. Pernicka. Optimal control for proximity operations and docking. *International Journal of Aeronautical and Space Sciences*, 11, 2010.

[16] H.C. Lim, H. Bang, and S. Lee. Adaptive backstepping control for satellite formation flying with mass uncertainty. *Journal of Astronomy and Space Sciences*, 23, 2006.

[17] H. Schaub and J.L. Junkins. *Analytical Mechanics Of Space Systems*. American Institute of Aeronautics and Astronautics, 2003.

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics Modelling, Planning and Control*. Springer, 2010.

[19] R. Jaiswal and O. Prakash. Classical and modern gain estimation approach of pid controller for the pitch control of the rcta aircraft. *INCAS BULLETIN*, 14, 2022.

[20] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 05 2010.

[21] S.M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[22] Z. Mu, W. Xu, X. Gao, L. Xue, and C. Li. Obstacles modeling and collision detection of space robots for performing on-orbit services. In *2014 4th IEEE International Conference on Information Science and Technology*, 2014.

[23] Z. Mu, W. Xu, and B. Liang. Avoidance of multiple moving obstacles during active debris removal using a redundant space manipulator. *International Journal of Control, Automation and Systems*, 2017.