POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

Design of Guidance Algorithm for a Space Manipulator

Supervisors

Prof. Elisa CAPELLO

Ing. Massimiliano SAPONARA

Candidate

Alessandro MANGANIELLO

April 2022

Summary

In the past decades, due to the growing complexity of space missions, the need for operations such as maintenance, refueling, de-orbiting and re-orbiting has increased. Missions with such goals, the so called In Orbit Servicing (IOS) missions, can be accomplished with the help of space robotics systems (or space manipulators). The use of such systems has been proven to be a promising approach to IOS in several occasions. For instance space manipulators have been used during the Space Shuttle missions, on the ISS, and played a key role in the maintenance of the Hubble Space Telescope.
In addition, IOS missions could be a good solution removing the growing amount of space debris and inactive satellites orbiting around the Earth, which represent a hazard due to the risk of collision with new constellation of satellites.
Motivated by the previous considerations, the work of this thesis is to design an optimal guidance algorithm for a space manipulator. The trajectory planning was treated as an optimization problem, which gives the advantage of handling constraints on joint angles and angular velocity. A heuristic optimization algorithm based on searching methods, the so called *Particle Swarm Optimization* (PSO) algorithm, was used to solve such problem. The reference space manipulator is the one designed for the active debris removal ESA mission e.Deorbit. A simulator in MATLAB/SIMULINK environment was developed, modeling the dynamics and kinematics of the robot manipulator and of the base satellite. The designed guidance algorithm was tested together with two different type of controllers (a sliding mode controller and a Linear Time-Varying MPC). A Linear MPC was also developed to control the spacecraft attitude during the movement of the manipulator. A simulation was performed to analyse the effect of the manipulator's torque disturbances on the base satellite to ensure that the attitude was kept constant during the whole maneuver.

Acknowledgements

I would like to express my sincere gratitude to Prof. Capello who was always ready to help me at any time with her kindness, experience and guidance but, at the same time, allowing me to work autonomously on such an exciting project. I would like to thank Ing. Saponara for his precious advice and for the wonderful opportunity to work on this thesis at Thales Alenia Space, a truly inspiring place where I could fully express my deep passion about space and GNC. I would like to thank also Ing. Cometto and all the other wonderful people who welcomed me at Thales Alenia Space, for helping me in such a new environment.

My greatest gratitude goes to *Ma'*, who was always by my side ready to help me in every possible way, despite the distance, with the love that only a mother can give. I want to thank her for the patience and the immeasurable strength she had to believe in me and allow me to follow my dreams, even if this meant to let his son fly away from home. I want to thank *Pa'* for his constant support and his invaluable advice, that helped me during the the hardest moments of this journey. His experience and his approach to life are of great inspiration for me and will surely help me in the future. I would like to thank my whole family, for being so close, despite the many kilometers that separates each other, making me always fell loved.

I want to thank *Clh* for the huge patience and the help he constantly gives me everyday (expecially in washing the dishes), and I want to apologize for how much annoying I sometimes I can be. I want to thank all the friends who were by my side during this journey, both the near and the distant one, and everyone that supported me during my academic career

Finally, I would like to thank Giulia for her choice to stay by my side everyday during this wonderful journey, both in the joyous and in the hard days. I am immensely grateful for the love and support she gave me, always believing in me

and pushing me to give the best. Thank you for always being there for me.

Table of Contents

List of Figures

Acronyms

IOS
In Orbit Service

DOF
Degree Of Freedom

ISS
International Space Station

GNC
Guidance Navigation and Control

AOCS
Attitude and Orbit Control System

PSO
Particle Swarm Optimization

DE
Differential Evolution

MPC
Model Predictive Control

SMC
Sliding Mode Control

Chapter 1

Introduction

Nowadays, the current practice for satellites that are inoperable due to malfunctioning, a too-low amount of fuel or just aging, is to replace them with new spacecrafts. Although this may guarantee continuity of satellite services, it worsens the growing issue of space debris, since the inactive spacecrafts are often left orbiting around the Earth. In this scenario, IOS missions could play a key role in the sustainable management of the space environment around our planet, extending the operative life of the already orbiting satellites, and allowing easier de-orbiting operations for the disposal.
In the following chapter a brief overview on IOS and the role of space robotics during this missions will be given. Particular attention will be spent on ESA's e.Deorbit mission, whose space manipulator design was taken as a reference for the development of this thesis. Finally, a brief overview on the work will be presented.

1.1 Overview on IOS

The term *In Orbit Servicing* refers to a particular kind of space missions regarding the in-orbit maintenance of a space system. The most common goals of such missions include the repair, assembly, refuel and upgrade of a space system.[1] The concept of IOS was first introduced in the 1960's, and demonstrated by several space missions during the last century.[2]
For instance, it is worth to mention the five IOS missions on the Hubble Space telescope (Figure 1.1), which were executed with the aid of the SRMS (Space Remote Manipulator System), the robot manipulator installed on the Space Shuttle. Similarly, many more IOS missions where executed with the aid of the so-called space robotic systems (or space manipulators). A space manipulator is a system usually composed by two major elements[1]: a base satellite and a n-DOF robot manipulator (or robotic arm).

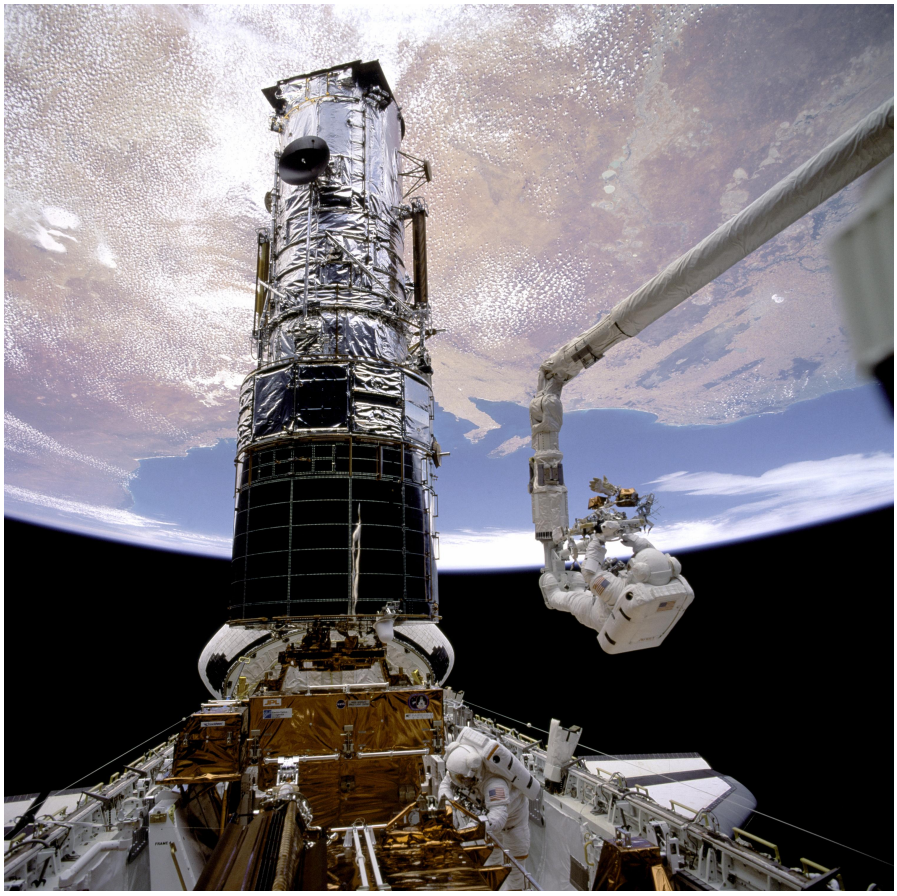


Figure 1.1: SRMS servicing the Hubble Space Telescope during STS-61[3]

The most relevant examples of such systems are the aforementioned SRMS installed on the Space Shuttle fleet and the MSS (Mobile Servicing System) mounted on the ISS.[4]
The SRMS, also known as Canadarm 1 is a 15.3 meters long robotic arm with 6 DOF, a mass of 408 kg, and a payload capability of 29,500 kg[5]. It was used during the shuttle program as a main tool for loading and unloading cargo from Shuttle payload bay on Earth orbit. In addition to the Hubble servicing missions, the space shuttle robotic arm played also a crucial role in the assembling of the ISS, removing the new elements of the station from the shuttle's cargo bay and transferring them to the SSRMS[4]. A schematic of the shuttle robotic manipulator is illustrated in figure 1.2.
The MSS (Mobile Servicing System) was developed by the Canadian Space Agency and consists of three main robotic elements[7]:

- **The Space Station Remote Manipulator System (SSRMS):** a 7-meter,

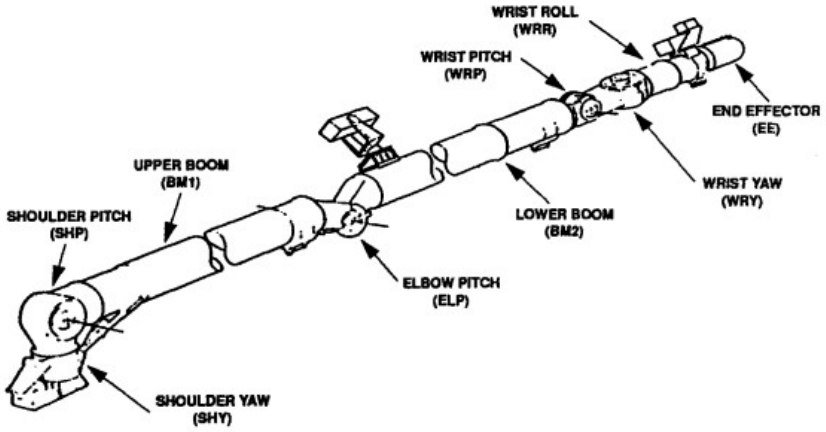


Figure 1.2: Schematic of SRMS[6]

7-DOF, robotic manipulator. It played a key role in the construction of the ISS. Its main tasks include helping with the IOS of the space station, and providing a platform for astronauts performing assembly and maintenance operations during EVA. It is also used to move supplies, equipment and the robotic arm Dextre from one point to another of the ISS. In addition, SSRMS is used to grasp and berth the vehicles visiting the ISS, such as SpaceX Dragon and Orbital Cygnus. The SSRMS can be either operated by the astronauts on the ISS or by the ground station operators[8].

- **The Special Purpose Dextrous Manipulator (SPDM) or Dextre:** a 15-DOF two arm robot, with a length of 2.5 m, a mass of 1,662 kg and payload capacity of 600 kg[5]. It can be attached either to the tip of SSRMS, onto the ISS Mobile Base System, or to a number of fixed locations around the ISS[3]. After its launch in March 2008, many delicate tasks previously performed by astronauts during EVAs, such as assembly and maintenance, were then carried out by the SPDM.

- **The Mobile Base System (MBS):** a support platform that provides power and data links for the SSRMS and the SPDM

A picture of the SSRMS and Dextre is shown in Figure 1.3.
In addition, other memorable examples of space robotics systems are the European systems ERA (European Robotic Arm) and ROTEX (Robot Technology Experiment), and the Japanese systems JEMRMS (Japanese Experiment Module Remote Manipulator System) and ETS-VII (Engineering Test Satellite VII).

Figure 1.3: The International Space Station's Canadarm2 and Dextre [9]

- **ERA** is a 7-DOF manipulator, with a length of 11.3 meters. It has a shoulder with 3 DOF, an elbow with 1 DOF and a wrist with 3 DOF[10]. Furthermore, the end-effector and the shoulder are operationally interchangeable. It was launched to the ISS in 2021 and installed on the Russian segment. Its main tasks include the installation, deployment and replacement of solar arrays, the inspection of the ISS, handling of external payloads and acting as a supporting platform for astronauts during spacewalks [2].

- **ROTEX** was a space robot technology experiment, developed by the German space agency DLR and successfully tested in space during the Spacelab Mission D2 in 1993. The ROTEX was the first remotely controlled robot ever launched in orbit [11] and was developed to study and experimentally demonstrate robotics technologies on the Space Shuttle. The main components of the systems included a small, six-axis robot, its gripper with sensors, an array of 9 laser range finders and a pair of stereo cameras [2]. It performed operations such as tests of assembling a truss structure and catching a free-floating object.

- **JEMRMS** is a robotic space manipulator developed by the Japan Aerospace Exploration Agency (JAXA), which included a main and a small fine arm. The main arm has 6 DOF and is 10 meters long and was launched in 2008 in a pressurized module of the JEM (Japanese Experiment Module) during STS-124. The small fine arm has 6 DOF and is 2 m long. It was launched on board of HTV-1 and mounted on the end effector of the main arm[2]. The main task of JEMRMS is to support the experiments conducted on the Exposed Facility (EF) of the JAM[1].

- **ETS-VII** was the first satellite in the world to be equipped with a robotic manipulator. It was developed by the National Space Development Agency of Japan (NASDA) and launched in November 1997 to verify on-orbit servicing technologies. It consisted of two satellites, Hikoboshi and Orihime, and had a total mass of 2860 kg. Hikoboshi, the chaser satellite, was equipped with a 6-DOF, 2 meters long robotic arm, with a mass of 45 kg. The tested servicing operations included the inspection of a predetermined position of the target, refueling, the assembly of the truss structure and the installation of the test antenna[2].

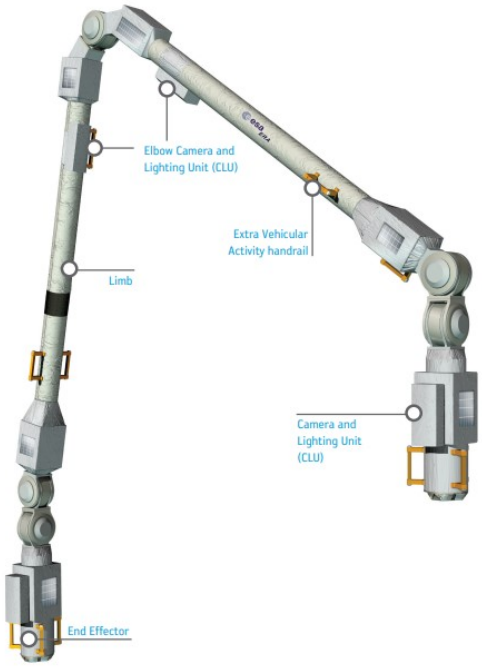


Figure 1.4: Schematics of ERA [12]

1.2 e.Deorbit mission

ESA's e.Deorbit mission concept was born in 2013 as an active debris removal (ADR) mission in the context of ESA Clean Space, an initiative with the goal of shifting towards a more sustainable space industry. The initial idea was to capture ESA's former Earth remote sensing satellite Envisat and burning it during a safely controlled atmospheric re-entry. The choice of Envisat as target of the mission was motivated by its large mass of about 8000 kg and its high probability of collision

with other satellites in sun-synchronous orbit, which may lead eventually to the so-called "Kessler Syndrome" [13]. The mission was scheduled for launch in 2024 on board of a Vega vector, and the satellite had to be delivered to a sun-synchronous orbit between 800 and 1000 km near the polar region[14]. Although the baseline option for the capture mechanism was a robotic arm, further strategies to capture the target were proposed and investigated *e.g.*robotic arm grippers, clamping mechanisms, nets, harpoons and tethers[15]. Some of these concepts are illustrated in Figure 1.5. In 2018, the fundings for e.Deorbit were stopped, and ESA Clean Space

Figure 1.5: Three different e.Deorbit capture concepts [15]

initiative started to work on the ClearSpace-1 mission, as follow-up of e.Deorbit.

For the purpose of this thesis the design concept of e.Deorbit chaser was taken as a reference. The chaser system can be divided into two subsystems: the base satellite and the robot manipulator.

- **The base satellite** (Figure 1.6) is a 1500-1600 kg spacecraft. Particular attention for the purpose of this work is payed to the GNC attitude control actuator configuration, which will be necessary for the implementation of the spacecraft attitude dynamics and kinematics model described in chapter 2. The satellite is provided with a cluster of 24 cold gas thrusters[16], as shown in figure 1.7. Further details on the GNC actuator configuration and modeling will be provided in the next chapter.

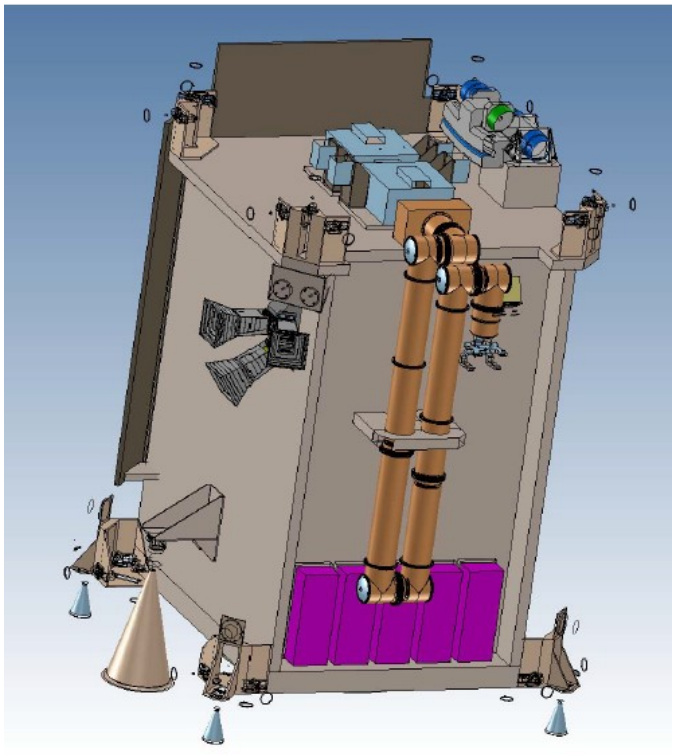


Figure 1.6: e.Deorbit base satellite configuration[17]

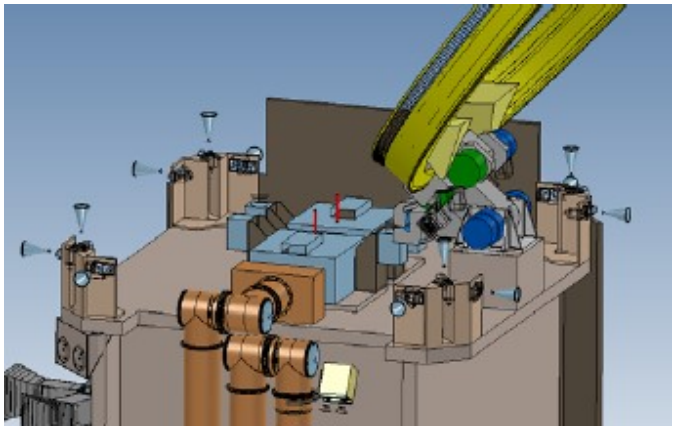


Figure 1.7: Detail on e.Deorbit ACS thruster configuration[17]

- **The manipulator** consists in a 7-DOF robotic arm with 7 identical revolute joints. The DOF are distributed as follows: 3-DOF (Roll-Pitch-Roll) in the shoulder, 1-DOF (Pitch) in the elbow, 3-DOF(Roll-Pitch-Roll) in the wrist. Such kinematic redundancy has several advantages, allowing for a greater

dexterity and robustness with respect to kinematics singularities[18]. For kinematically redundant manipulators the inverse kinematics problem has, indeed, infinite solutions, allowing the robotic arm to fulfill additional tasks, such as minimization of the base disturbances and avoidance of singular configurations. A schematics of the manipulator arm is reported in figure 1.8 Further details on the manipulator subsystem geometry and physical model will be provided in the following chapter.

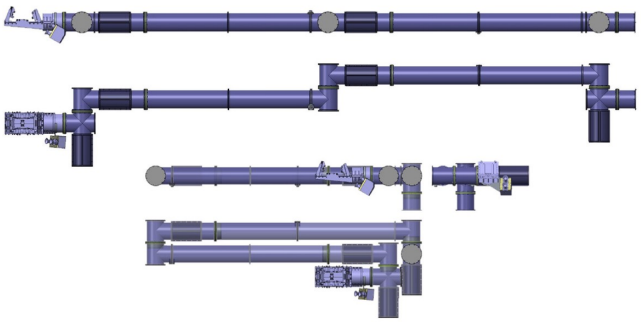


Figure 1.8: e.Deorbit robotic arm in stretched and stowed configurations[18]

All the data concerning the base satellite and the manipulator were provided as a gentle concession of Thales Alenia Space.

1.3 Work Overview

The scope of this thesis is to develop an optimal guidance algorithm for a space manipulator. This goal is achieved through innovative heuristic optimization algorithms based on searching methods. In particular the *Particle Swarm Optimization* (PSO) algorithm was used to solve the trajectory planning treated as an optimization problem. For the purpose of the thesis, a GNC model comprehensive of both the base satellite and the robot manipulator was developed. The navigation part was not studied, assuming, for simplicity, that all state variables are measurable, thus no observer is required. Therefore, both the satellite and the manipulator models consists of the following three main parts:

- **Guidance:** provides the desired state of the system. For the base satellite attitude, it is a constant output, since the ACS aims to stabilize the attitude during the maneuver. In the manipulator case, it consists of the optimal trajectory planning computed off-line by the PSO-based algorithm.
- **Control:** consists in an algorithm which provides the required control to achieve a certain state of the system.

- **Plant:** contains the kinematics and dynamics equation of the system, and provides the state variables.

The two models were chosen to be kept separated for simplicity, and influence one another through the share of information on the state of the models. In particular, the satellite angular velocity and acceleration influences the manipulator, and the base reaction torque of the manipulator is introduced as an external disturbance affecting the spacecraft attitude.
The overall work can be summarized as follows. Initially a simulator of the dynamics and kinematics of the robot manipulator was developed in a MATLAB/SIMULINK environment. A simple polynomial guidance and a sliding mode controller were developed, and the model was tested simulating some simple maneuvers with both a joint-space and a cartesian control. After this initial phase, an off-line optimal guidance algorithm was developed. In order to test the trajectory planned by the guidance algorithm, the model was completed with a Linear Time-Varying MPC, and a simple maneuver was simulated. Finally, to analyse the disturbance torque exerted by the manipulator on the base satellite, the spacecraft attitude model was introduced. A simple Linear MPC was designed to control the spacecraft attitude, stabilizing it during the whole maneuver. The rest of this thesis is organized as follows:

- **Chapter 2** contains a mathematical description of the systems modeled in this work. It is divided into three main sections: Satellite, where the attitude kinematics and dynamics is discussed together with the actuator model, Robot Manipulator, where the kinematics and dynamics of the robotic arm is introduced, and Disturbance torque, where the complete space manipulator model will be briefly introduced in order to compute the reaction torque of the robotic arm.

- **Chapter 3** introduces the optimal guidance algorithm and is subdivided into three subsections: Trajectory Planning, where the optimization problem is described, PSO , where the Particle Swarm Optimization is introduced, and Results.

- **Chapter 4** reports the different controllers implemented and their theoretical base.

- **Chapter 5** introduces the MATLAB/SIMULINK model and simulations and discusses the results obtained.

- **Chapter 6** discusses the conclusions of the work and the future works.

Chapter 2

System Mathematical Model

In the following chapter an accurate mathematical description of the space manipulator system will be given. The overall system is composed of two main components: the base satellite and the manipulator. For the purpose of this work only the attitude of the base satellite is considered, whereas the position dynamics were not studied. The first section of this chapter presents the mathematical model regarding the base satellite attitude. In addition, in the second section, the mathematical model of the robotic manipulator is introduced. Finally a brief overview of the complete space manipulator is given, which is necessary to calculate the base torque of the robotic arm.

2.1 Base Satellite

As anticipated in the introduction, the reference base satellite considered for this work is the one designed for the mission e.Deorbit. The spacecraft attitude kinematics and dynamics are modeled in order to analyze the effect of the external disturbances introduced by the movement of the manipulator. A quaternion-based attitude kinematics is studied since it can avoid gimbal lock singularities which may occur with the use of other attitude representations, such as Euler angles or axis-angle. Furthermore, the attitude dynamics is described by the Euler equations. Finally, in the last subsection, the mathematical model of the ACS actuators (Reaction Wheels and Thrusters) is described.

2.1.1 Attitude Kinematics

When it comes to attitude representation, several valid possibilities exist. For instance a very common method of expressing an orientation is the so called rotation matrix. However, this solution gives a redundant description of a frame

orientation[19]. It is, indeed, characterized by nine elements that are not linearly independent from one another. Since 6 equations for the orthonormality condition hold, it follows that a minimal attitude representation can be given by a vector of three linearly independent elements only. This is the case of the Euler angles. Such parameters are given by a set of three angles, expressing three consecutive rotations around the frame axes (it must be guaranteed that two successive rotations are not made about parallel axes). Although this option allows the use of three parameters only, the angles are not uniquely defined in some cases: the so-called gimbal lock. This problematic can be overcome with the use of a four-parameters representation called unit quaternion (also known as Euler parameters). In general, a quaternion is a collection of four real parameters, of which the first is a scalar and the other three constitutes a vector[20]:

$$\mathbf{q} = \begin{bmatrix} q_0 \\ \mathbf{q_v} \end{bmatrix} \tag{2.1}$$

where $\mathbf{q_v} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T$.
In particular, the unit quaternion is a special type of quaternion that holds the following property:

$$q_0^2 + \mathbf{q_v^T q_v} = 1 \tag{2.2}$$

To understand how the unit quaternion can be used to express attitude orientation, the *Euler's eigenaxis rotation theorem* must be introduced first. The theorem states that it is possible to express the rotation from an initial to a final reference frame with a rotation by an angle θ about an axis $\hat{\mathbf{a}}$ (Euler's axis) that is fixed in both frames[21]. The unit quaternion representing the rotation is then defined as follows:

$$\mathbf{q} = \begin{bmatrix} cos(\frac{\theta}{2}) \\ \hat{\mathbf{a}} sin(\frac{\theta}{2}) \end{bmatrix} \tag{2.3}$$

The unit quaternion is always defined no matter the attitude configuration of the spacecraft. For this reason this option was chosen to represent the orientation of the base satellite. Before introducing the spacecraft attitude kinematics equation, the quaternion product must be defined as:

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{Q(q)p} = \mathbf{P(p)q} \tag{2.4}$$

Where:

- $\mathbf{Q(q)} = \begin{bmatrix} q_0 & -\mathbf{q_v^T} \\ \mathbf{q_v} & q_0\mathbf{I_3} + \mathbf{q_v^x} \end{bmatrix}$
- $\mathbf{P(p)} = \begin{bmatrix} p_0 & -\mathbf{p_v^T} \\ \mathbf{p_v} & p_0\mathbf{I_3} - \mathbf{p_v^x} \end{bmatrix}$

$\mathbf{I_3}$ is the 3x3 identity matrix and $\mathbf{q_v^x} = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}$ is the skew symmetric matrix expressing the cross product operator $\mathbf{q_v} \times (\cdot)$. The spacecraft attitude kinematics equation can be now expressed as the quaternion evolution over time and involves the quaternion and the angular velocity of the spacecraft.

$$\dot{\mathbf{q}} = \frac{1}{2}\omega \otimes \mathbf{q} = \frac{1}{2}\mathbf{\Omega}(\omega)\mathbf{q} \tag{2.5}$$

Where $\omega = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ is the quaternion extension of the angular velocity. From Eq. 2.5 another advantage in using the unit quaternion formulation follows: this linear kinematics equation is less computationally expensive than the one derived with the Euler's angle[21]. However, in this case, since the 4 components of the unit quaternion are not linearly independent, it must hold the unit norm constraint. In order to avoid numerical errors during the integration of the kinematics equation, an additional term is added and the kinematics equation is modified as follows:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}(\omega)\mathbf{q} + k(1 - \mathbf{q^T q})\mathbf{q} \tag{2.6}$$

Where k is a positive real constant.

2.1.2 Attitude Dynamics

The equation of the attitude dynamics is the so called Euler's equation. It is possible to derive such equation starting from the law of conservation of angular momentum of the spacecraft[21].

$$\frac{d\mathbf{L}}{dt} = \mathbf{M} \tag{2.7}$$

Where $\mathbf{L}$ is the total angular momentum and $\mathbf{M}$ is the total torque acting on the system. Expressing the equation in a reference frame fixed with the base satellite (body frame) and denoting all the quantities expressed in such frame with the notation $(\cdot)_b$ it becomes:

$$\frac{d\mathbf{L_b}}{dt} + \omega_b \times \mathbf{L_b} = \mathbf{M_b} \tag{2.8}$$

The total torque acting on the base satellite can be written as the sum of the control torque $\mathbf{M_c}$ and the external disturbances $\mathbf{M_{ext}}$, that in this work will be given by the reaction torque of the manipulator. All the environmental torques (e.g. gravity gradient) are neglected for simplicity. In addition, assuming a constant inertia matrix $\mathbf{I_b}$ and rewriting in function of $\dot{\omega}$, the equation becomes:

$$\dot{\omega} = \mathbf{I_b}^{-1}(\mathbf{M_b}) - \mathbf{I_b}^{-1}(\omega_b \times \mathbf{I_b}\omega_b) \tag{2.9}$$

For the purpose of the control algorithm design it can be useful to write complete equation of the dynamical system combining both the attitude kinematics and dynamic equation and obtaining the following non-linear system of differential equations describing the evolution of the spacecraft attitude:

$$\begin{cases} \dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}(\omega_\mathbf{b})\mathbf{q} \\ \dot{\omega} = -\mathbf{I_b}^{-1}\left(\omega_b \times \mathbf{I_b}\omega_b\right) + \mathbf{I_b}^{-1}\left(\mathbf{M_b}\right) \end{cases} \tag{2.10}$$

2.1.3 Actuators

In the following section, the mathematical model of the attitude control actuators is described. Two different actuation solutions were chosen. In addition to the cluster of 24 cold gas thruster designed during e.Deorbit B1 phase, a pyramidal cluster of 4 Reaction Wheel was added as main ACS actuator, in order to allow a fine attitude stabilization. A comparison of the results of the stabilization with the Reaction Wheels and with the Thrusters will be given in chapter 5.

Reaction Wheel Modeling

A reaction wheel is a devices that allows the exchange of angular momentum with the spacecraft. In this way, a continuous actuation of the control torque calculated by the ACS is provided, modifying the attitude orientation of the overall system. On the other hand, a reaction wheel is subject to saturation, hence the imposition of constraints on the maximum angular momentum and the maximum feasible torque. To take into account these limitations, a model of the reaction wheel system was implemented in MATLAB/SIMULINK. The reaction wheel pyramidal configuration chosen is reported in fig.2.1.

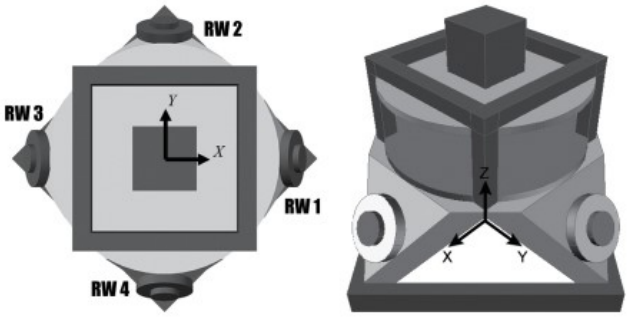


Figure 2.1: Reaction Wheel pyramidal configuration [22]

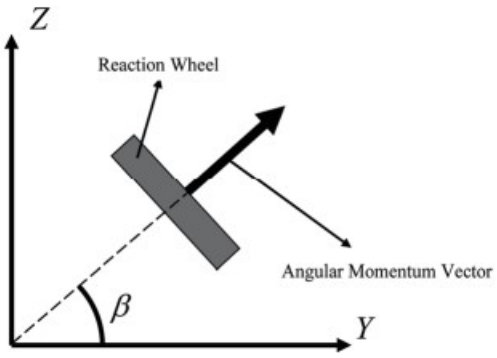


Figure 2.2: Detail on reaction wheel tilt angle [22]

The relation between the desired control torque computed by the controller and the torque corresponding to each reaction wheel is the following:

$$\mathbf{M_c} = -\mathbf{Z}\mathbf{M_{RW}} \tag{2.11}$$

where

- $\mathbf{M_{RW}} \in \mathbb{R}^{4,1}$ is the column vector containing the torque of the four reaction wheel about their rotation axis.
- $\mathbf{Z} = \begin{bmatrix} cos(\beta) & 0 & -cos(\beta) & 0 \\ 0 & cos(\beta) & 0 & -cos(\beta) \\ sin(\beta) & sin(\beta) & sin(\beta) & sin(\beta) \end{bmatrix}$, with β representing the reaction wheel tilt angle as shown in fig 2.2.

Since the reaction wheels introduce an internal torque in the system, the attitude Euler equation becomes the following:

$$\dot{\omega} = \mathbf{I_b}^{-1}(\mathbf{M_c}) - \mathbf{I_b}^{-1}(\omega_b \times (\mathbf{I_b}\omega_b + h_{RW})) \tag{2.12}$$

Where h_{RW} is the reaction wheel angular momentum. The MATLAB/SIMULINK model of a single reaction wheel is the one reported in fig.2.3. The control torque in input is multiplied by -1 to express it with respect to the reaction wheel. The it is received as input by a low-pass filter and finally a saturator is added to take into account the actuator limitations on the maximum torque. The resulting torque is then integrated to compute the reaction wheel angular momentum needed as input for the Euler equation. The model is defined by the filter parameter τ_{RW} which has been carefully tuned by trial and error.

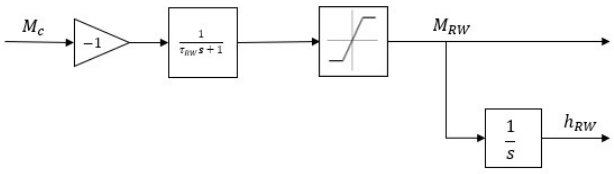


Figure 2.3: Scheme of the reaction wheel model

Thrusters Modeling - PWPF Modulator

To stay coherent with the original e.Deorbit mission design, a cluster of 24 attitude control thruster was implemented. The configuration, shown in fig1.7 consists of 4 thrusters on the corners of each face of the spacecraft, pointing with an outgoing direction, perpendicular to the face. The relation between thruster forces and resulting torque is the following:

$$\mathbf{M_{th}} = \mathbf{T}\mathbf{F_{th}} \tag{2.13}$$

where

- $\mathbf{M_{th}} \in \mathbb{R}^{3,1}$ is the column vector of the torque acting on the spacecraft along the x y and z direction in a body frame.
- $\mathbf{F_{th}} \in \mathbb{R}_{+}^{24,1}$ is the column vector containing the force of each one of the 24 thrusters.
- $\mathbf{T} \in \mathbb{R}^{3,24}$ is called *thruster configuration matrix* and allows the conversion of the thruster force into the resulting torque.

Even though the control output coming from the controller is a continuous signal, the thrusters have an on-off non-linear behaviour by nature[23]. In addition, for certain types of controllers the control torque does not take into account the saturation level of the thrusters that imposes a constraint on the maximum feasible torque. Therefore, it is necessary to translate the desired torque computed by the controller into an on-off signal that can be actually actuated by the thrusters. The Pulse-Width/Pulse-Frequency (PWPF) modulator can be used to give such on-off quasi-linear steady state behaviour to the control signal. The system is depicted in fig 2.4 and consists of a feedback loop with of two main parts: a first order filter and a schmitt trigger.

- **First order filter**: it is used to give a quasi-linear steady state behaviour to the output. It receives as input the error between the output of the schmitt trigger and the input command signal.

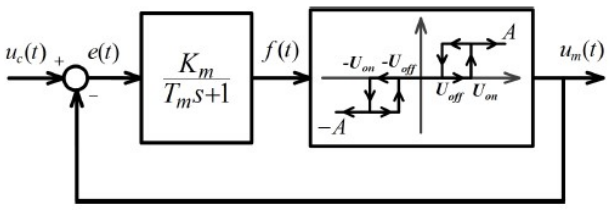


Figure 2.4: Schematics of a PWPF modulator [24]

- **Schmitt Trigger**: it is an hysteresis/dead-zone system. The hysteresis gives both on and off values to the trigger. When the input is greater then U_{on} the trigger is activated until the value drops below U_{off}. In addition, to avoid excessive thruster chattering, a dead-zone is introduced. When the input is within this interval, the filter gives a null output. This implies a reduced thruster activity, thus resulting in a reduction in fuel consumption[25].

In general, the modulator is defined by 4 parameters:

- Two filter parameters: K_f and τ_f, provided the following transfer function for the filter: $\frac{K_f}{\tau_f+1}$
- Two trigger parameters: U_{on} and U_{off}

The aforementioned parameters have been carefully tuned with a trial-and-error-based method.

The complete thruster model is reported in fig 2.5. The input received is the

Figure 2.5: Schematics of the thruster model implemented on MATLAB/SIMULINK

desired control torque computed by the controller. This torque is multiplied by the

Moore-Penrose pseudo-inverse of the thruster configuration matrix $\mathbf{T}^{\dagger}$, resulting in the desired force for each one of the 24 thrusters. Then each force is modulated with a PWPF filter and, finally the resulting vector is multiplied by $\mathbf{T}$, resulting in the modulated feasible torque.

2.2 Robot Manipulator

In the following section the mathematical model of the robot manipulator is introduced. At first, a brief introduction on the main geometric and physical parameters is given. Then, the manipulator direct and differential kinematics is described. Finally the manipulator dynamics is derived, following two different approaches: the Lagrangian approach based on the Lagrangian mechanics and the Newton-Euler approach based on the equilibrium of the forces and momenta acting on the manipulator.

2.2.1 Geometric Characterization

The robot manipulator is modeled as a kinematic chain composed of seven rigid bodies called links connected by seven rotational joints. In addition, seven reference frames are defined, each one fixed with a link and with the origin lying in the respective i-th joint. The robot manipulator scheme is reported in fig.2.6. The

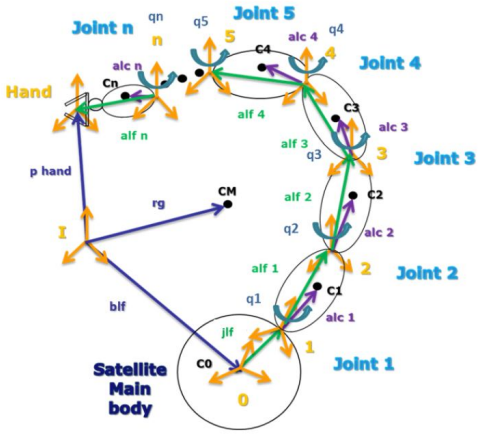


Figure 2.6: Schematics of the space manipulator[26]

robot arm is defined form a geometric and physical standpoint by the following parameters:

- *an* is the number of joints (or Degree of Freedom). In this case the manipulator has 7-DoF.

- *alf*[m] is the distance between the i-th joint frame and the (i+1)-th expressed in the i-th joint frame.
- *alc*[m] is the distance between the i-th joint and the i-th link CoM expressed in the i-th joint frame.
- *arot* is the [3x3] rotation matrix transforming points expressed in the i-th joint frame into the (i+1)-th joint frame.
- *am*[kg] is the mass of the i-th link
- *ai* is the [3x3] inertia matrix of the i-th link expressed in a frame with origin in the i-th CoM oriented as the (i+1)-th frame.

These parameters will be useful to define the direct and differential kinematics of the manipulator in the following subsection.

2.2.2 Manipulator Kinematics

When it comes to manipulator kinematics, it is useful to distinguish between *direct kinematics* and *differential kinematics*. The direct kinematics of the manipulator has the aim to compute the position of the end-effector with respect to the base frame, as a function of the manipulator configuration, uniquely defined by the joint angles or joint variables. The differential kinematics is the relationship between the end-effector linear and angular velocities and the joints' angular velocities.

Direct Kinematics

To express the direct kinematics, an homogeneous transformation matrix must be defined, that links the expression of a vector in the base frame to the expression in the end-effector frame with the following direct kinematics equation:

$$\mathbf{v^0} = \mathbf{T^0_n v^n} \tag{2.14}$$

Where $\mathbf{T^0_n} \in \mathbb{R}^{4,4}$ is the homogeneous transformation matrix, transforming vectors from the end effector frame to the base frame. The notation $(\cdot)^{\mathbf{i}}$ denotes a vector expressed in the frame i. In addition, $\mathbf{v^0} \in \mathbb{R}^{4,1}$ and $\mathbf{v^n} \in \mathbb{R}^{4,1}$ are expressed in the base frame denoted with 0 and the end-effector frame denoted with n, and they are a four dimensional extension of a three elements vector, as shown in the following equation:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix} \tag{2.15}$$

The homogeneous transformation matrix $\mathbf{T_n^0}$ can be computed as a combination of the n homogeneous transformation matrices $\mathbf{T_i^{i-1}}$, with $i = 1, \ldots, n$ transforming a vector form the i-th joint frame to the (i-1)-th joint frame. The general expression for these matrices is:

$$\mathbf{T_i^{i-1}} = \begin{bmatrix} \mathbf{R_z}(\theta_\mathbf{i})\mathbf{arot_i} & \mathbf{R_z}(\theta_\mathbf{i})\mathbf{alf_i} \\ \\ \mathbf{0_{1,3}} & 1 \end{bmatrix} \tag{2.16}$$

where $\mathbf{R_z}(\theta_\mathbf{i}) = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the elementary rotation matrix expressing a rotation by the i-th joint angle θ_i about the z axis of the i-th joint frame. It follows that:

$$\mathbf{T_n^0} = \mathbf{T_1^0}\ \mathbf{T_2^1}\ \ldots\ \mathbf{T_i^{i-1}}\ \ldots\ \mathbf{T_n^{n-1}} \tag{2.17}$$

Differential Kinematics

The goal of the differential kinematics is to find a relationship to express the end-effector linear and angular velocity as a function of the joint angles and angular velocities. Such relation is given by a the linear equation:

$$\mathbf{v} = \begin{bmatrix} \dot{\mathbf{p}} \\ \omega \end{bmatrix} = \mathbf{J}(\theta)\dot{\theta} \tag{2.18}$$

The matrix $\mathbf{J}$ is the *manipulator geometric Jacobian* matrix. It is a function of the manipulator configuration and can be derived through kinematics considerations on the contributes of each joint angular velocity to the end-effector linear and angular velocity[19]. The general expression for the jacobian matrix is the following:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 & \ldots & \mathbf{J}_i & \ldots & \mathbf{J}_n \end{bmatrix} \in \mathbb{R}^{6,n} \tag{2.19}$$

$\mathbf{J}_i$ is the i-th joint jacobian matrix and is related to the i-th joint contribute to the end-effector linear and angular velocity. It can be computed as follows:

$$\mathbf{J}_i = \begin{bmatrix} \mathbf{z}_{i-1}^0 \times (\mathbf{p_n^0} - \mathbf{p_i^0}) \\ \mathbf{z}_{i-1}^0 \end{bmatrix} \in \mathbb{R}^{6,1} \tag{2.20}$$

where $\mathbf{z}_{i-1}^0$ is the z axis of the (i-1)th joint frame expressed in the base frame, and $\mathbf{p}_n^0$ and $\mathbf{p}_i^0$ are respectively the position of the origin of the end-effector and the i-th joint frames with respect to the base frame. The differential kinematics has several applications, and plays a key role in solving the so-called inverse kinematics problem[19]. For the purpose of this thesis the differential kinematics will be useful for the derivation of the manipulator dynamics, in particular with the Lagrangian formulation, as described in the following section.

2.2.3 Manipulator Dynamics

The manipulator dynamics aims to describe the motion of the manipulator through a dynamics equation as a function of the forces and momenta acting on it[19]. There are typically two different approaches[19]:

- **Lagrangian approach**: is based on Lagrangian mechanics and and has the advantage to give a systematic derivation of the equation of motion which is independent from the chosen reference frame.
- **Newtonian approach**: is based on Newtonian mechanics and derives the dynamics equation starting from the study of the equilibrium of the forces and momenta acting on the manipulator. This approach allows to derive the system model through a recursive algorithm, thus it is computationally more efficient[19].

These two methods for the derivation of the manipulator dynamics will be now described in detail.

Lagrangian Approach

The Lagrangian approach is based on the Lagrangian mechanics, whose fundamental equation, the so called Euler-Lagrange equation, is the following:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} - \frac{\partial \mathcal{L}}{\partial \theta_i} = \tau_i \qquad i = 1, ..., n \tag{2.21}$$

Here the Lagrangian of the system is defined as the difference between the system total kinetic energy and the system total potential energy

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{2.22}$$

The therm θ is a proper generalized coordinate, which in the case of the manipulator is the joint variable, and the therm τ is the generalized force associated to θ. The contribution to the generalized force is in general given by all the non conservative forces acting on the system, *i.e.,* joint torques, friction acting on the joints, contact forces acting on the end-effector, etc. For the purpose of the thesis a simplified model is considered, where all the friction forces are neglected and the joint torques are considered only as contribution to the generalized force. In addition, since the space manipulator is supposed to operate in a micro/zero-gravity environment, another simplification hypothesis is made, neglecting all the gravity dependent therms. Thus, the total potential energy is negligible and the only contribution to the Lagrangian is the total kinetic energy. The total kinetic energy is given by the sum of the total kinetic energies of the n links:

$$\mathcal{T} = \sum_{i=1}^{n} \mathcal{T}_i \tag{2.23}$$

The kinetic energy of the i-th link can be derived as follows:

$$\mathcal{T}_i = \frac{1}{2} \int_{V_i} \rho \ \dot{\mathbf{p}}_\mathbf{i}^{*T} \dot{p}_i^* \ dV \tag{2.24}$$

Where $\mathbf{p_i^*}$ is the position of an infinitesimal particle in the i-th link, expressed in base frame, as shown in fig.2.7, and ρ is the infinitesimal particle density.

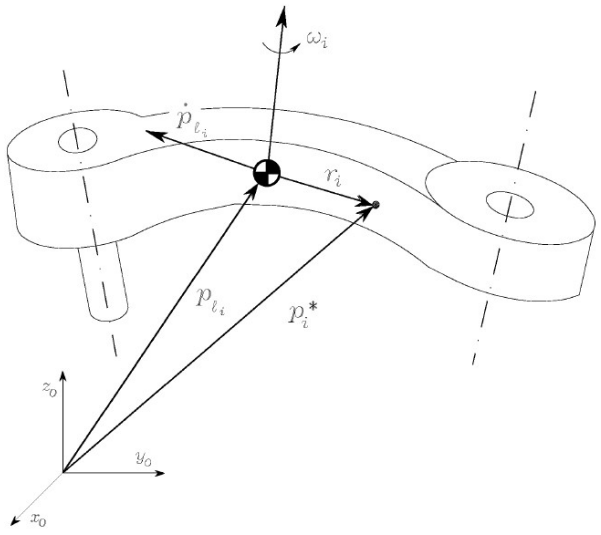


Figure 2.7: Kinematic description of the i-th link for Lagrangian formulation[19]

The i-th link center of mass is defined as follows:

$$\mathbf{p_{li}} = \frac{1}{m_i} \int_{V_i} \rho \ \mathbf{p_i^*} \ dV \tag{2.25}$$

Differentiating $\mathbf{p_i^*}$ it follows that:

$$\dot{\mathbf{p}}_\mathbf{i}^* = \dot{\mathbf{p}}_\mathbf{li} + \omega_i^x \mathbf{r_i} \tag{2.26}$$

Where $\mathbf{r_i} = \mathbf{p_i^*} - \mathbf{p_{li}}$. Substituting in Eq. 2.24, three contributes to the i-th link kinetic energy can be highlighted:

1. **Translational contribute**:

$$\frac{1}{2} \int_{V_i} \rho dV \ \dot{\mathbf{p}}_\mathbf{li}^\mathbf{T} \dot{\mathbf{p}}_\mathbf{li} = \frac{1}{2} m_i \dot{\mathbf{p}}_\mathbf{li}^\mathbf{T} \dot{\mathbf{p}}_\mathbf{li} \tag{2.27}$$

2. **Rotational contribute**:

$$\frac{1}{2} \int_{V_i} \rho \ \omega_i^x \mathbf{r_i}^T \omega_i^x \mathbf{r_i} dV = \frac{1}{2} \omega_i^T \int_{V_i} \rho \ \mathbf{r_i}^{xT} \mathbf{r_i}^x dV \omega_i = \frac{1}{2} \omega_i^T \mathbf{I_i} \omega_i \tag{2.28}$$

where $\mathbf{I_i} = \mathbf{R_i a i_i R_i^T}$ is the i-th link inertia matrix in the base frame

3. **Mutual contribution**:

$$\dot{\mathbf{p}}_{\mathbf{li}}^{\mathbf{T}} \omega_i^x \int_{V_i} \rho \ \mathbf{r_i} dV = 0 \tag{2.29}$$

for the Eq. 2.25

By summing the three contributions and rewriting the expression in matricial form, it results:

$$\mathcal{T}_i = \frac{1}{2} \begin{bmatrix} \dot{\mathbf{p}}_{\mathbf{li}} \\ \omega_i \end{bmatrix}^T \begin{bmatrix} m_i \mathbb{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & I_i \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}}_{\mathbf{li}} \\ \omega_i \end{bmatrix} \tag{2.30}$$

Defining:

$$\mathbf{M_i} = \begin{bmatrix} m_i \mathbb{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{I_i} \end{bmatrix} \tag{2.31}$$

and remembering that:

$$\begin{bmatrix} \dot{\mathbf{p}}_{\mathbf{li}} \\ \omega_i \end{bmatrix} = \mathbf{J_{li}} \dot{\theta} \tag{2.32}$$

where $\mathbf{J_{li}}$ is the geometric Jacobian matrix referred to the i-th link center of mass. It is possible to rewrite the total kinetic energy of the system as follows:

$$\mathcal{T} = \frac{1}{2} \dot{\theta}^T \left(\sum_{i=1}^{n} \mathbf{J_{li}}^T \mathbf{M_i} \mathbf{J_{li}} \right) \dot{\theta} = \frac{1}{2} \dot{\theta}^T \mathbf{B}(\theta) \dot{\theta} \tag{2.33}$$

Substituting the total kinetic energy in the Euler-Lagrange equation it is possible to obtain the dynamics equation of the robot manipulator:

$$\mathbf{B}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} = \tau \tag{2.34}$$

where:

- $\mathbf{B}(\theta) = \sum_{i=1}^{n} \mathbf{J_{li}}^T \mathbf{M_i} \mathbf{J_{li}} \in \mathbb{R}^{n,n}$ is called the *inertia* matrix, and is computed in the MATLAB/SIMULINK model using the equation derived with the Lagrange formulation

- $\mathbf{C}(\theta, \dot{\theta}) \in \mathbb{R}^{n,n}$ is called the *Coriolis/centripetal* matrix and contains the therms related to Coriolis and centripetal acceleration. This matrix will be computed using the recursive algorithm based on the Newton-Euler formulation of the dynamics.

Newton-Euler Approach

The Newton-Euler approach to the manipulator dynamics consists in a recursive algorithm that takes the joint angles θ, angular velocities $\dot{\theta}$ and angular accelerations $\ddot{\theta}$ the angular velocity ω_0, angular acceleration $\dot{\omega}_0$ and linear acceleration $\ddot{p}_0$ acting on the base of the manipulator, and the force f_{n+1} and torque τ_{n+1} acting on the end-effector. The algorithm computes the force f, and moment μ acting on each link and gives as output the torque τ acting on each joint. The use of a recursive algorithm allows an higher computational efficiency[19]. The N-E algorithm is based on the Newtonian mechanics, and is formulated through the study of the equilibrium of the forces and momenta acting on each one of the n links. Starting from the kinematics relations between the links, it is possible to write the angular velocity, angular acceleration and linear acceleration on the i-th link as a function of the angular velocity, angular acceleration and linear acceleration on the (i-1)th link[19].

$$\begin{cases} \omega_i = f_\omega(\omega_{i-1}, \dot{\theta}_i) \\ \dot{\omega}_i = f_{\dot{\omega}}(\dot{\omega}_{i-1}, \omega_{i-1}, \dot{\theta}_i, \theta_i) \\ \ddot{p}_i = f_{\ddot{p}}(\ddot{p}_{i-1}, \dot{\omega}_i, \omega_i) \end{cases} \tag{2.35}$$

Starting from the forces and moments equilibrium, it is possible to write the force and moment acting on the i-th link as a function of the force and moment acting on the (i+1)th link.

$$\begin{cases} f_i = f_f(f_{i+1}, \ddot{p}_i) \\ \mu_i = f_\mu(\mu_{i+1}, f_{i+1}, f_i, \dot{\omega}_i, \omega_i) \end{cases} \tag{2.36}$$

Finally the joint torque can be computed as a function of the moment acting on the i-th link:

$$\tau_i = f_\tau(\mu_i) \tag{2.37}$$

Starting from the manipulator's base boundary conditions ω_0, $\dot{\omega}_0$, $\ddot{p}_0$ a forward recursion is performed to calculate ω, $\dot{\omega}$ and $\ddot{p}$ for each link using Eq. 2.35. Then, a backward recursion is performed, starting from the boundary conditions at the end effector f_{n+1} and τ_{n+1} to calculate f and τ acting on each link through Eq. 2.36 and subsequently computing the joint torques τ through Eq. 2.37. A schematic representation of the algorithm is illustrated in fig.2.8.
The Newton-Euler algorithm can be used to solve the direct dynamics problem. It consists of determining the evolution of the joint angles, angular velocities and angular accelerations, starting from an initial known condition and knowing the joint torques. To do so it is sufficient to rewrite the dynamics equation explicitating $\ddot{\theta}$.

$$\ddot{\theta} = \mathbf{B}(\theta)^{-1}\left(\tau - \mathbf{C}(\theta, \dot{\theta})\dot{\theta}\right) \tag{2.38}$$

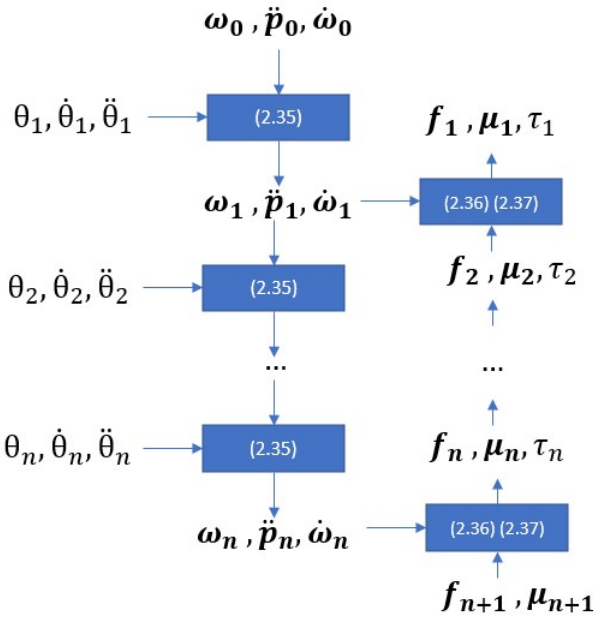


Figure 2.8: Schematics of the Newton-Euler recursive algorithm[19]

as previously explained, the inertia matrix $\mathbf{B}(\theta)$ can be computed with the Lagrange formulation, whereas it is possible to compute the product $\mathbf{C}(\theta, \dot{\theta})\dot{\theta}$ as the torque output of the Newton-Euler algorithm, when $\ddot{\theta} = \mathbf{0}_{n,1}$. Then substituting into Eq. 2.38 $\ddot{\theta}$ is obtained. Finally $\dot{\theta}$ and θ can be computed as follows:

$$\dot{\theta} = \dot{\theta}_0 + \int_{t_0}^{t} \ddot{\theta}\ dt \tag{2.39}$$

$$\theta = \theta_0 + \int_{t_0}^{t} \dot{\theta}\ dt \tag{2.40}$$

In the MATLAB/SIMULINK simulator Eq. 2.39 and 2.40 are, of course, performed through numerical integration.

2.3 Disturbance Torque

In order to define the disturbance torque acting on the base spacecraft a complete dynamic and kinematic model must be defined, describing the whole space manipulator comprehensive of both the base spacecraft and the space manipulator. During the last decades of the past century, several attempts to define a model describing the complicated coupled kinematics and dynamics of a space manipulator have been

made[1]. For instance the so-called *Virtual Manipulator* introduced by Vafa and Dubowksy in 1987[1][27], the *Dynamically Equivalent Manipulator* introduced by Liang et. al[1][28], and the *Generalized Jacobian* approach introduced by Umetani and Yoshida in 1989 [1][29]. The guidelines introduced by Umetani and Yoshida in their publication were taken as a reference to compute the base reaction torque of the manipulator. In particular, the total angular momentum acting on the space manipulator system is given by the sum of two contributes: the one linked to the base satellite angular rate and the one given by joint angular velocities of the manipulator:

$$L = H_b\omega_b + H_{bm}\dot{\theta} \tag{2.41}$$

Where:

- ω_b is the base satellite angular velocity
- $H_b \in \mathbb{R}^{3,3}$ is a function of the geometric and physical characteristics of the base satellite.
- $H_{0b} \in \mathbb{R}^{3,3}$ is called the *dynamic coupling matrix*[30] and can be calculated as follows[30],[29]:

$$H_{0b} = \sum_{1}^{n} \begin{bmatrix} m_i \mathbf{r_{0i}}^x & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{I_i} \end{bmatrix} \mathbf{J_{li}} \tag{2.42}$$

In conclusion the base disturbance torque given by the manipulator motion can be computed as the derivative over time of the contribute of the manipulator on the total angular momentum. Therefore, the equation of the base disturbance is the following:

$$\mathbf{M_{ext}} = \frac{d(\mathbf{H_{0b}}\dot{\theta})}{dt} \tag{2.43}$$

The MATLAB/SIMULINK simulator performs the computation through numerical differentiation.

Chapter 3

Guidance Algorithm

In this chapter the problem of guidance of the manipulator is discussed. At first the trajectory planning is introduced, with a brief overview on the state of the art, then the optimization problem formulation is given. Finally, the Particle Swarm Optimization algorithm is described.

3.1 Trajectory Planning

The goal of the trajectory planning is to generate the reference input for the control system. Subsequently, the controller ensures the tracking of the desired trajectory computed by the guidance. For the purpose of this thesis a point-to-point trajectory is generated, which means that only the initial and final positions and orientations of the end-effector are assigned. The trajectory can be planned in the joint space, specifying the time sequence of the desired joint angles, or in the operational space, where the reference end-effector position over time is generated. The joint space trajectory has the advantage of generating the motion of each joint independently, but it does not take into account the actual position of the end-effector between the initial and final pose. With the operational (or Cartesian) space trajectory, on the other hand, the joint motion must be computed through the solution of the inverse kinematics problem, which can be generally done using algorithms based on the pseudo-inverse of the Jacobian [19].
Usually, the easiest solution for guidance, both in the joint or in the cartesian space, is to choose a polynomial trajectory interpolating the initial and final desired point, in a given execution time[31]. Even though this solution may be simple, it does not take into account the several constraints that the manipulator is usually subject to during its motion. For instance, constraints on the joints' limits or on the joint velocities must be handled. For this reason, a good approach may be to treat the trajectory planning as an optimization problem. The optimal

trajectory planning has been of great research interests during the years, and several approaches have been employed. For instance, an analytical approach to the optimization issue is given by Seweryn and Banaszkiewicz [32] and by Rybus et al. [33], using Hamiltonian mechanics in combination with the Lagrangian multipliers to handle the optimization constraints. Another innovative and rather successful approach is given by solving the optimization problem through searching algorithms, such as variational approaches [34], genetic algorithms [35], differential evolution [36][37], and Particle Swarm Optimization[38][39][40][41]. In addition, treating the trajectory planning as an optimization issue gives the advantage of the fulfillment of additional tasks, such as minimizing the base disturbances or collision avoidance capability. In particular, the minimization of the base reaction torque may be of great interest when it comes to space manipulators, since the spacecraft attitude needs to be stable in order to satisfy pointing requirements for communication or payload purposes. Several methods to minimize the base disturbances have been investigated in literature. For example the *Reaction Null Space* method, introduced by Yoshida et al. [42], and widely used in several works [43][44][37][40] consists in using the null space of the dynamic coupling matrix to formulate a velocity profile that gives zero angular momentum to the base spacecraft throughout the whole maneuver. However, for low redundant manipulators, the reaction null space has a limited volume[38], and other approaches may be eventually employed to obtain satisfying results. For example, the base disturbance may be added to the objective function to be minimized as done by Wang et al. [38], leaving to the optimization process the task of finding the best solution in terms of lowest reaction torque. In the work in this thesis, an original approach is studied, where the reaction torque is subject to a dynamic constraint, given, eventually, by the ACS actuators limitations (e.g. maximum control torque exerted by thrusters/ reaction wheels). As explained in the following sections, the use of a parametric curve to describe the joint trajectory is crucial for the handling of the aforementioned constraints.

3.1.1 Optimization Problem Description

The optimization problem aims to find an appropriate joint angle profile $\theta(t)$ such that an objective function (or fitting function) $\Gamma(\theta)$ gives the minimum possible output. Usually the problem is subject to a certain number of kinematic or dynamic equality and inequality constraints.

$$\begin{cases} \textbf{min } \Gamma(\theta) \\ g_i(\theta) = 0 \ i = 1, ..., n_g \\ f_i(\theta) < 0, \ i = 1, ..., n_f \end{cases} \tag{3.1}$$

In the case treated in this work, the objective function is given by the error of the final pose of the end-effector with respect to the desired final pose. Such quantity

can be computed as a function of the joint angles, as follows. The end-effector position is calculated through the direct kinematics relation derived in chapter 2.

$$\begin{bmatrix} \mathbf{r_e}(\theta) \\ 1 \end{bmatrix} = \mathbf{T}_n^0(\theta) \begin{bmatrix} \mathbf{0}_{3,1} \\ 1 \end{bmatrix} \tag{3.2}$$

The end-effector orientation is formulated with the unit quaternion, and it is computed starting from the elements of the rotation matrix $\mathbf{R}_n^0(\theta)$ as follows[19]:

$$\mathbf{q_e}(\theta) = \frac{1}{2} \begin{bmatrix} \sqrt{r_{11} + r_{22} + r_{33} + 1} \\ sgn(r_{32} - r_{23})(r_{11} - r_{22} - r_{33} + 1) \\ sgn(r_{13} - r_{31})(r_{22} - r_{33} - r_{11} + 1) \\ sgn(r_{21} - r_{12})(r_{33} - r_{11} - r_{22} + 1) \end{bmatrix} \tag{3.3}$$

where r_{ij} is the element of the i-th row and j-th column of $\mathbf{R}_n^0$, and $sgn(x)$ is the sign function defined as:

$$sgn(x) = \begin{cases} 1 \; if \; x \geq 0 \\ -1 \; if \; x < 0 \end{cases} \tag{3.4}$$

The vector expressing the end-effector pose is thus defined as:

$$\mathbf{x_e}(\theta) = \begin{bmatrix} \mathbf{r_e} \\ \mathbf{q_e} \end{bmatrix} \tag{3.5}$$

The error vector with respect to the desired position is defined as follows:

$$\delta \mathbf{x_e} = \begin{bmatrix} \mathbf{r}_d - \mathbf{r}_e \\ \delta \mathbf{q} \end{bmatrix} \tag{3.6}$$

where $\delta \mathbf{q} \in \mathbb{R}^{3,1}$ is the quaternion error, defined as the vector part of $\mathbf{q}_d \otimes \mathbf{q}_e^{-1}$ and $\mathbf{q}^{-1} = [q_0, -\mathbf{q}_v]^T$. The quaternion error is an intuitive way to express the error between orientations, since, when two frames are aligned all the components of $\delta \mathbf{q}$ become zero. The fitting function can now be defined as the norm of the error vector multiplied by an appropriate weighting matrix.

$$\Gamma(\theta) = \|\mathbf{Q} \delta \mathbf{x_e}\| \tag{3.7}$$

The weighting matrix $\mathbf{Q} \in \mathbb{R}^{6,6}$ can be chosen starting from the tolerance on the position and orientation error.

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{toll_{position}} \mathbb{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \frac{1}{toll_{orientation}} \mathbb{I}_3 \end{bmatrix} \tag{3.8}$$

Finally, the complete optimization problem can be formulated by adding the equality and inequality constraints on the joint angles, velocities and accelerations and on the reaction torque.

$$\begin{cases} \mathbf{min} \ \|\mathbf{Q} \ \delta\mathbf{x_e}\| \\ \dot{\theta}(t_i) = \dot{\theta}(t_f) = \ddot{\theta}(t_i) = \ddot{\theta}(t_f) = \mathbf{0}_{n,1} \\ \theta_{min} < \theta < \theta_{max} \\ \left\|\dot{\theta}\right\| \leq \dot{\theta}_{max} \\ \|\mathbf{M}_{ext}\| \leq \mathbf{M}_{max} \end{cases} \tag{3.9}$$

where t_i and t_f are the initial and final time of the maneuver respectively. In order to ensure the satisfaction of all the constraints, a parametrization, together with a constraint handling strategy in the PSO algorithm, is introduced in the following sections.

3.1.2 Parametrization

The angular velocity profile of the manipulator's joints is parametrized with a Bézier curve, a particular kind of curve widely used in computer graphics for its smoothness[38]. The general expression for the i-th joint velocity written as a Bézier curve is the following:

$$\dot{\theta}_i(t) = \frac{1}{T}\dot{\bar{\theta}}_i(\tau) = \frac{1}{T}\sum_{j=0}^{m}\binom{m}{j}P_{ij}(1-\tau)^{m-j}\tau^j \tag{3.10}$$

where:

- $T = t_f - t_i$ is the execution time of the maneuver
- $\tau = \frac{t}{T}$ is the normalized time
- $\dot{\bar{\theta}}$ is the parametrized joint velocity
- P_{ij} is the j-th coefficient of the Bézier curve
- m is the order of the Bézier curve, which in this case was chosen as $m = 4$.

To ensure that all the equality constraints are satisfied, the initial and final conditions on the joint velocities and accelerations are imposed:

$$\begin{cases} \dot{\bar{\theta}}_i(0) = 0 \Rightarrow P_{i0} = 0 \\ \dot{\bar{\theta}}_i(1) = 0 \Rightarrow P_{i4} = 0 \\ \ddot{\bar{\theta}}_i(0) = 0 \Rightarrow P_{i1} = 0 \\ \ddot{\bar{\theta}}_i(1) = 0 \Rightarrow P_{i3} = 0 \end{cases} \tag{3.11}$$

Hence, substituting into Eq.3.10, the joint angular velocity profiles becomes:

$$\dot{\theta}_i(t) = \frac{1}{T}\dot{\bar{\theta}}_i(\tau) = \frac{6}{T}P_{i2}(1-\tau)^2\tau^2 \tag{3.12}$$

Subsequently, the joint angles can be computed integrating the velocity profile:

$$\theta = \theta_0 + \int_{t_i}^{t_f} \dot{\theta}(t)\ dt = \theta_0 + \int_0^1 \dot{\bar{\theta}}(\tau)\ d\tau \tag{3.13}$$

The joint velocity profile and, therefore, the joint angles profile are completely defined if the n Bézier coefficients $[P_{12}, P_{12}, ..., P_{n2}]^T$ are defined. Thus, the problem can be reformulated writing the fitting function as a function of the Bézier coefficients. Hence, these coefficients become the design variables to be determined by the PSO algorithm. Furthermore, it is possible to choose an appropriate execution time of the maneuver T, in order to satisfy the inequality constraints on the joint velocity and on the reaction torque. Once the all the Bézier coefficient are determined, the joint velocity profile can be written as:

$$\dot{\theta} = \frac{1}{T}\dot{\bar{\theta}} \tag{3.14}$$

and the corresponding reaction torque is given by:

$$\mathbf{M_{ext}} = \frac{d(H_{0n}\dot{\theta})}{dt} = \frac{1}{T^2}\frac{d(H_{0n}\dot{\bar{\theta}})}{d\tau} = \frac{1}{T^2}\bar{\mathbf{M}}_{\mathbf{ext}} \tag{3.15}$$

The execution time can then be chosen accordingly:

$$T \geq \mathbf{max}\left(\frac{\dot{\bar{\theta}}_{max}}{\dot{\theta}_{max}}, \sqrt{\frac{\bar{\mathbf{M}}_{\mathbf{max}}}{\mathbf{M_{max}}}}\right) \tag{3.16}$$

where $\dot{\bar{\theta}}_{max}$ and $\bar{\mathbf{M}}_{max}$ are the maximum joint velocity and the maximum reaction torque reached during the maneuver, respectively. Concerning the inequality constraints on the joint angles, a constraint handling strategy is introduced in the PSO algorithm in order to guarantee that these conditions are not violated. Even though all the constraints are satisfied and the final end-effector pose error is minimized, this optimization strategy does not take into account the magnitude of the execution time, eventually leading to excessively slow maneuvers. To handle this problem, the execution time T may be introduced as a secondary objective function to be minimized. In this regard, the PSO algorithm is a really convenient choice since it has been proven to be able to successfully solve multi-objective optimization problems[45][46].

3.2 PSO

The **Particle Swarm Optimization** is a heuristic stochastic searching algorithm used to solve optimization problems. Like many similar algorithms, such as genetic algorithms, it mimics a behaviour found in nature[47]. The PSO was introduced by J. Kennedy and R. Eberhart in 1995 [48], inspired by the movements of swarms of birds and schools of fishes. These animals cooperates in a group to find food, changing their position according to the shared information on the experience of their own and of other members of the swarm. Similarly, it is possible to define a swarm of candidate solutions (particles) that will change their position at each iteration of the algorithm according to their personal best position and the global best position of the swarm, in order to find the optimum of a certain function by searching in a large space of solutions. Like many meta-heuristic optimization algorithms, it is not gradient-based and therefore can be employed in the solution of complex non-differentiable optimization problems. In addition, a great advantage of the PSO is its rather simple mathematical formulation together with a low computational effort. However, the convergence of the algorithm to the global optimum is not guaranteed, and it may suffer from problems of stagnation at local optimum points.
The algorithm's principle can be illustrated with the schematics in Fig.3.1. At first,

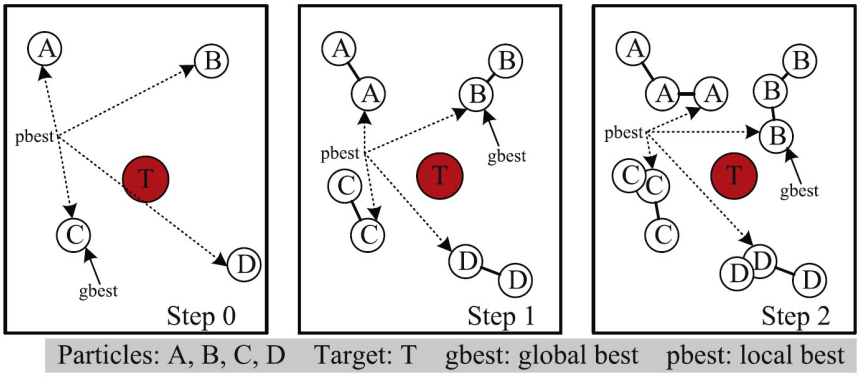


Figure 3.1: Schematics of PSO[38]

the algorithm initializes a swarm of random possible solutions as particles. In the example in Fig.3.1 the swarm is composed by 4 particles with a dimension of 2, but in general the dimension of the particle is represented by the number of design variables which, for the algorithm developed in this thesis, are given by the vector of Bézier coefficients $\mathbf{p} = [P_{12}, P_{22}, P_{32}, P_{42}, P_{52}, P_{62}, P_{72}]^T$ with dimension n=7, the number of DoF of the manipulator. The algorithm calculates the personal best position of each particle and the global best position of the entire swarm, based

on the objective function. Subsequently, the position of each particle is updated according to the following formula:

$$p_i^{t+1} = p_i^t + v_i^{t+1} \tag{3.17}$$

where v_i is the particle velocity and is calculated at each iteration with the following equation:

$$v_i^{t+1} = wv_i^t + c_1 r_1 (p_{pbest}^t - p_i^t) + c_2 r_2 (p_{gbest}^t - p_i^t) \tag{3.18}$$

According to Eq. 3.18, the velocity of a particle consists of three major contributes illustrated in Fig.3.2:

1. **Previous velocity contribute:** wv_i^t, where w is called inertia weighting factor and balances between global and local search in the solution space.

2. **Cognitive learning contribute:** $c_1 r_1 (p_{pbest}^t - p_i^t)$, proportional to the distance between the current position of the particle and the personal best position ever reached by the particle in the past. The coefficient c_1 is the cognitive coefficient, and is a parameter of the algorithm giving the acceleration constant of the cognitive learning contribute to the velocity. The term r_1 is a random normal distributed real number between 0 and 1.

3. **Social learning contribute:** $c_2 r_2 (p_{gbest}^t - p_i^t)$, proportional to the distance between the current position of the particle and the global best position ever reached by the swarm in the past. The coefficient c_2 is the social coefficient, and gives the acceleration constant of the social learning contribute to the velocity. The term r_2 is a random normal distributed real number between 0 and 1.

A slightly modified formula for the velocity was adopted for this thesis, taking as a reference the work of Clerk and Kennedy[50], who analyzed the convergence behavior of PSO and introduced a variant of the algorithm with a constriction factor χ to improve the velocity of convergence[51].

$$v_i^{t+1} = \chi (v_i^t + c_1 r_1 (p_{pbest}^t - p_i^t) + c_2 r_2 (p_{gbest}^t - p_i^t)) \tag{3.19}$$

The constriction factor can be calculated as follows:

$$\chi = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|}, \ \phi = c1 + c2 \tag{3.20}$$

The algorithm will perform the iterations, calculating the personal best and global best and then updating the particles' position, until the cost function $\Gamma(\mathbf{p}_{gbest})$ drops below a pre-defined tolerance ϵ. In addition, a maximum number of iterations

Algorithm 1 PSO pseudo-code

▷ Random initialization of $\mathbf{p}_i^0$ and $\mathbf{v}_i^0$
▷ Set $\mathbf{p}_{pbest,i} = \mathbf{p}_i^0$ and $\mathbf{p}_{gbest}$ as the $\mathbf{p}_{pbest,i}$ with the lowest $\Gamma(\mathbf{p}_{pbest,i})$
▷ Set iter=1
while $\Gamma(\mathbf{p}_{gbest}) > \epsilon$ *and* $iter < iter_{max}$ **do**
    ▷ Evaluate $\mathbf{p}_{pbest}$ and $\mathbf{p}_{gbest}$
    ▷ Update $\mathbf{p}_i$ and $\mathbf{v}_i$ according to Eq. 3.17 and 3.19
    ▷ iter=iter+1
end while
▷ return $\mathbf{p}_{gbest}$ as the optimal solution

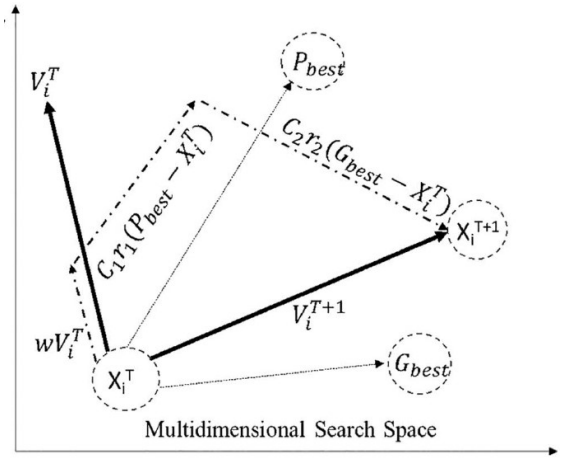


Figure 3.2: Contributes to the velocity of a particle in the PSO[49]

is defined to stop the algorithm if the objective function does not reach below the threshold. The pseudo-code for the PSO is given in Algorithm 1:
In order to evaluate the personal best and the global best particles, the objective function must be involved. In general, when the constraints are not taken into account, for an optimization problem with one objective function only, the personal best position of the i-th particle is initially set equal to the first random generated position, and subsequently updated with $\mathbf{p}_i$ if:

$$\Gamma(\mathbf{p}_i) < \Gamma(\mathbf{p}_{pbest,i}),\ i = 1, ..., n_p \tag{3.21}$$

where $n_p$is the number of particles in the swarm. Similarly, the global best position of the swarm is selected among all the personal best positions as the personal best solution with the minimum objective function.

$$\Gamma(\mathbf{p}_{gbest}) = \min_{\forall i \in [1, n_p]} \Gamma(\mathbf{p}_{pbest,i}) \tag{3.22}$$

3.2.1 Constraint Handling

An important issue, introduced in the previous section, is the fact that the optimization problem defined in Eq. 3.9 is subject to a constraint on the joint angles. Therefore, a strategy for constraint handling is introduced in the algorithm, based on the concept of *constraint domination* defined in the work presented by L.D. Li et al. [52]. The degree of constraint violation can be measured with a function $\Phi(\mathbf{p}_i)$:

$$\Phi(\mathbf{p}_i) = \sum_{k=1}^{n} [\mathbf{max}(0, \theta(\mathbf{p}_i)_{k,max} - \theta_{max}) + \mathbf{max}(0, \theta_{min} - \theta(\mathbf{p}_i)_{k,min})] \tag{3.23}$$

When $\Phi(\mathbf{p}_i) = 0$ the solution is said to be *feasible*, otherwise the higher is $\Phi(\mathbf{p}_i)$ the more the solution is *infeasible*. A solution $\mathbf{p}_A$ is *constraint dominant* with respect a solution $\mathbf{p}_B$ in two cases:

1. The solution $\mathbf{p}_A$ has less constraint violations than $\mathbf{p}_B$
2. Both the solutions have the same degree of constraint violations but $\mathbf{p}_A$ has a better fitting value than $\mathbf{p}_B$

That is, transposed into a logical expression:

$$\Phi(\mathbf{p}_A) < \Phi(\mathbf{p}_B) \ \vee \ [\Phi(\mathbf{p}_A) = \Phi(\mathbf{p}_B) \ \wedge \ \Gamma(\mathbf{p}_A) < \Gamma(\mathbf{p}_B)] \tag{3.24}$$

Therefore, the constraint it taken into account during the selection of the personal best position of a particle, substituting the condition in Eq. 3.21 with the following condition:

$$\Phi(\mathbf{p}_i) < \Phi(\mathbf{p}_{pbest,i}) \ \vee \ [\Phi(\mathbf{p}_i) = \Phi(\mathbf{p}_{pbest,i}) \ \wedge \ \Gamma(\mathbf{p}_i) < \Gamma(\mathbf{p}_{pbest,i})] \tag{3.25}$$

3.2.2 Multi-objective Optimization

As anticipated in the previous section, the strategy adopted to handle the inequality constraints on the joint velocity and the reaction torque by adjusting the execution time of the maneuver accordingly, may eventually lead to excessively slow maneuvers. A solution can be obtained introducing the execution time T as a secondary objective function to be minimized. However, since the primary objective of the optimization is the minimization of the final pose error of the end-effector, a strategy to prioritize the different objectives of the optimization must be introduced, to ensure that $\|\mathbf{Q}\ \delta\mathbf{x}\|$ drops below the threshold ϵ. The Lexicographic method[53] is used to ensure the priority of the the objective to optimize. With this method, the functions are ordered according to their importance and the optimization problem is reformulated as follows:

$$\begin{cases} \min T(\mathbf{p}) \\ \textit{subject to } \Gamma(\mathbf{p}) \leq \epsilon \end{cases} \tag{3.26}$$

The minimization of the first objective function can be treated as a constraint. The constraint domination concept can be then applied and, in this case, the constraint violation function is $\Gamma(\mathbf{p})$ itself. The condition for the selection of the personal best is now modified as follows. A particle's best position $\mathbf{p}_{best,i}$ is updated with its current position $\mathbf{p}_i$ in either of these three cases:

1. $\mathbf{p}_i$ has less joint constraint violations than $\mathbf{p}_{best,i}$

2. $\mathbf{p}_i$ has the same joint constraint violations as $\mathbf{p}_{best,i}$ and $\Gamma(\mathbf{p}_i)$ is lower than $\Gamma(\mathbf{p}_{pbest,i})$

3. $\mathbf{p}_i$ has the same joint constraint violations and objective function as $\mathbf{p}_{pbest,i}$ but $T(\mathbf{p}_i)$ is lower than $T(\mathbf{p}_{pbest,i})$

That is, transposed into a logical expression:

$$\begin{aligned} &\Phi(\mathbf{p}_i) < \Phi(\mathbf{p}_{pbest,i}) \vee \\ &[\Phi(\mathbf{p}_i) = \Phi(\mathbf{p}_{pbest,i}) \ \wedge \ \Gamma(\mathbf{p}_i) < \Gamma(\mathbf{p}_{pbest,i})] \vee \\ &[\Phi(\mathbf{p}_i) = \Phi(\mathbf{p}_{pbest,i}) \ \wedge \ \Gamma(\mathbf{p}_i) = \Gamma(\mathbf{p}_{pbest,i}) \ \wedge \ T(\mathbf{p}_i) < T(\mathbf{p}_{pbest,i})] \end{aligned} \tag{3.27}$$

Similarly, the global best position is updated with a particle among the n personal best position. In this way the prioritized optimization of $\Gamma(\mathbf{p})$ is guaranteed.

A schematics of the complete PSO algorithm designed for the guidance of the manipulator is illustrated in Fig.3.3. The algorithm starts by randomly initializing n_p

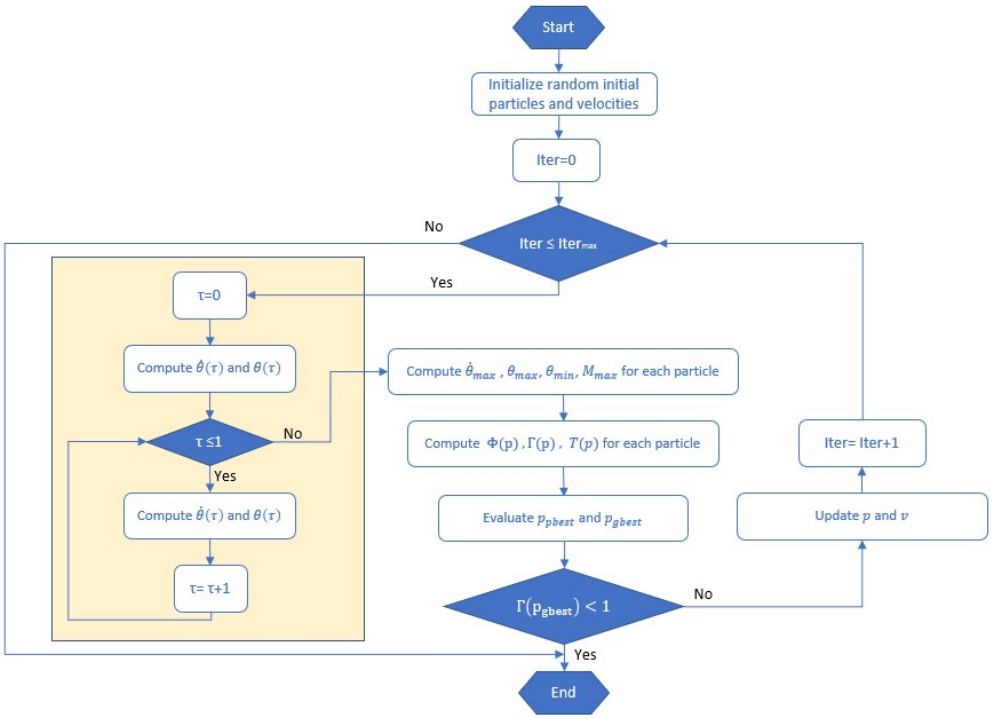


Figure 3.3: Flowchart of the designed PSO algorithm

particles $\mathbf{p}^0 = [P_{12}, P_{22}, P_{32}, P_{42}, P_{52}, P_{62}, P_{72}]^T$ and their velocities. Subsequently, the iterations starts and for each particle the relative joint velocities (Eq.3.12) and

joint angles (Eq.3.13) are computed, parametrized over τ. Then, the maximum velocity, base torque and maximum and minimum joint angles are computed. Finally, for each particle, Φ (Eq.3.23), Γ (Eq.3.7) and T (Eq.3.16) are computed. The algorithm then proceeds to select for each iteration the n_p personal best positions through Eq.3.27 and , with the same criteria, the global best position. Finally, the threshold for $\Gamma(\mathbf{p}_{gbest})$ is checked. If the objective function is below the tolerance, the algorithm stops, otherwise, each particle and its velocity are updated following Eq.3.17 and Eq.3.19. The iteration number is increased and the algorithm starts again until the threshold is reached or the iteration surpass the limit.

Chapter 4

Controllers

In this chapter the different control strategies adopted for the space manipulator are described and discussed. The Control system is composed of two main parts:

- Attitude control, used to stabilize the spacecraft at a constant attitude, expressed with the unit quaternion $\mathbf{q} = [1, 0, 0, 0]^T$.
- Manipulator Control, used to track the optimal trajectory generated by the guidance algorithm.

For the attitude control a linear MPC was designed, whereas, for the manipulator two different approaches were chosen. The first one is a Linear Time-Varying MPC and the second one is a sliding mode control. These two control strategies are then confronted and discussed in chapter 5.

4.1 Attitude Control

The attitude control of a spacecraft has the goal to maintain a stable, constant orientation of the satellite. This may be necessary during a mission for the pointing requirements of the payload or the communication system. In addition, during the maneuver with the robotic arm, because of the complex dynamics of the space manipulator, the base disturbances perturb the satellite attitude, which acquires angular acceleration and velocity that in turn influences the manipulator. If not kept stabilized, the attitude variations may cause the manipulator to not follow properly the optimal trajectory and therefore not reach the desired pose of the end-effector. In this regard, a linear MPC may be an effective approach, since the attitude should be kept constant and the system can be easily linearized around the equilibrium point.

4.1.1 Linear Model Predictive Attitude Control

The model predictive control is a rather famous modern control strategy that combines the theory behind feedback control and optimization. Because of its computational cost, which is higher than that of other simpler control strategies (PID, Sliding Mode, etc.) it was not immediately applied in the control of spacecraft attitude[54]. However, thanks to the increasingly rapid advancement in terms of computational capabilities of the spacecraft On Board Computers, the application of MPC for attitude control started to be investigated more and more in literature during the last decades. The MPC is mainly composed of two main parts:

- Predictive model, which in this case is derived by linearizing the attitude state-space equation around the equilibrium point $\mathbf{q} = [1, 0, 0, 0]^T$ and $\omega = [0, 0, 0]^T$
- Optimizer, which solves an On-line optimization problem at each time step of the simulation.

The presence of the optimization gives a notable advantage, since it makes it possible to eventually handle constraints on the command torque, which in a real spacecraft is usually limited by the nature of the actuators (maximum thrust for the thrusters and saturation for the reaction wheels).
The concept behind MPC is depicted in Fig 4.1. A prediction horizon n_p and a control horizon n_c are defined. The goal of the controller is to find the optimal control command sequence for the next n_c time steps that minimizes a certain cost function $\mathbf{J}$ over the n_p time steps of the prediction horizon. Once the optimal control sequence is computed, the first control command is taken as output of the MPC and the rest of the sequence is discarded. The process is then repeated for each time step, until the system reaches the desired steady state.
Starting from the spacecraft attitude dynamics and kinematics equation, the state-space equation can be written. Recalling the attitude dynamics and kinematics introduced in chapter 2:

$$\begin{cases} \dot{\mathbf{q}} = \frac{1}{2}\omega_{\mathbf{b}} \otimes \mathbf{q} \\ \dot{\omega} = -{\mathbf{I_b}}^{-1}\left(\omega_b \times \mathbf{I_b}\omega_b\right) + {\mathbf{I_b}}^{-1}\left(\mathbf{M_c}\right) \end{cases} \tag{4.1}$$

The kinematic equation can be further simplified, using the linear dependence of the scalar quaternion. Since $q_0 = \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)}$, the number of variables in the equation can be reduced and the quaternion kinematics equation becomes:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)} & -q_3 & q_2 \\ q_3 & \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)} & -q_1 \\ -q_2 & q_1 & \sqrt{1 - (q_1^2 + q_2^2 + q_3^2)} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.2}$$

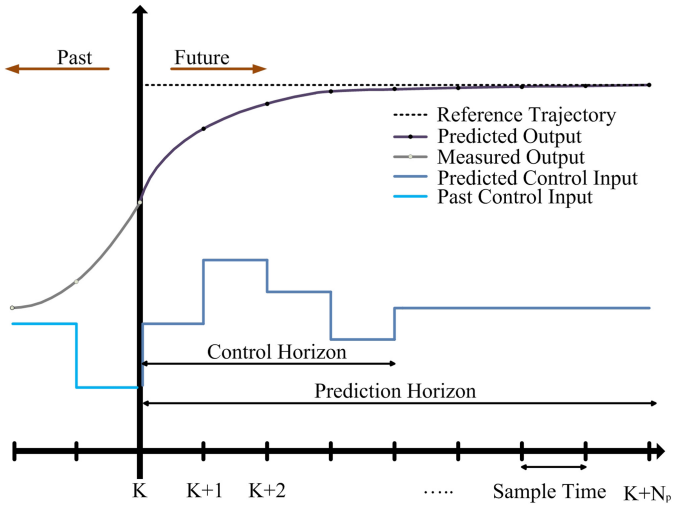


Figure 4.1: The principle of MPC[55]

The state-space equation can be rewritten in the form:

$$\dot{x} = f(x) + Bu \tag{4.3}$$

where:

- x is the state vector

$$x = \begin{bmatrix} \omega \\ \mathbf{q}_v \end{bmatrix} \tag{4.4}$$

- u is the control vector

$$u = \mathbf{M_c} \tag{4.5}$$

-

$$f(x) = \begin{bmatrix} -\mathbf{I_b}^{-1}(\omega_b \times \mathbf{I_b}\omega_b) \\ \frac{1}{2}\mathbf{Q}(\mathbf{q_v})\omega_b \end{bmatrix} \tag{4.6}$$

-

$$B = \begin{bmatrix} \mathbf{I_b}^{-1} \\ \mathbf{0}_{3,3} \end{bmatrix} \tag{4.7}$$

In order to implement the Linear MPC, the system must be linearized around the equilibrium point, and then discretized. Using the first order Taylor expansion around the point $x = [0,\ 0,\ 0,\ 0,\ 0,\ 0]^T$, the linearized system becomes[54]:

$$\begin{bmatrix} \dot{\omega}_b \\ \dot{\mathbf{q}}_v \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \frac{1}{2}\mathbb{I}_3 & \mathbf{0}_{3,3} \end{bmatrix} \begin{bmatrix} \omega_b \\ \mathbf{q}_v \end{bmatrix} + \begin{bmatrix} \mathbf{I_b}^{-1} \\ \mathbf{0}_{3,3} \end{bmatrix} u = \mathbf{A}x + \mathbf{B}u \tag{4.8}$$

The discretization of the system is performed through the embedded MATLAB command *"c2d"* and the resulting discrete time system becomes:

$$\begin{cases} x(k+1) = \mathbf{A_d}x(k) + \mathbf{B_d}u(k) \\ y(k) = \mathbf{C}x(k) \end{cases} \tag{4.9}$$

with

- $y = \mathbf{q}_v$
- $C = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbb{I}_3 \end{bmatrix}$

Once the system has been discretized it is possible to write the vector containing the predicted output of the system over the next n_p time-steps.

$$\begin{cases} Y_{pre} = \mathbf{S}x + \mathbf{W}U \\ U = U_{old} + \mathbf{L}\Delta U \end{cases} \tag{4.10}$$

where:

$$\mathbf{S} = \begin{bmatrix} \mathbf{CA_d} \\ \mathbf{CA_d^2} \\ \vdots \\ \mathbf{CA_d^{n_p}} \end{bmatrix} \in \mathbb{R}^{3n_p,6} \tag{4.11}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{CB_d} & \mathbf{0}_{3,3} & \dots & \mathbf{0}_{3,3} \\ \mathbf{CA_dB_d} & \mathbf{CB_d} & \dots & \mathbf{0}_{3,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA_d^{n_p-1}B_d} & \mathbf{CA_d^{n_p-2}B_d} & \dots & \mathbf{CB_d} \end{bmatrix} \in \mathbb{R}^{3n_p,3n_p} \tag{4.12}$$

$$\mathbf{L} = \begin{bmatrix} \mathbb{I}_3 & \dots & \mathbf{0}_{3,3} \\ \vdots & \ddots & \vdots \\ \mathbb{I}_3 & \dots & \mathbb{I}_3 \\ \vdots & \ddots & \vdots \\ \mathbb{I}_3 & \dots & \mathbb{I}_3 \end{bmatrix} \in \mathbb{R}^{3n_p,3n_c} \tag{4.13}$$

$$U_{old} = \begin{bmatrix} u_{old} \\ u_{old} \\ \vdots \\ u_{old} \end{bmatrix} \in \mathbb{R}^{3n_p,1} \tag{4.14}$$

$$\Delta U = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+n_c-1) \end{bmatrix} \in \mathbb{R}^{3n_c,1} \tag{4.15}$$

From equation 4.10 it is possible to calculate the prediction of the outputs over the next n_p time-steps taking as input the current state vector x and the previous command vector u_{old}. Once the discrete linearized predicting model is defined, the objective function to be minimized is written as:

$$\mathbf{J} = \|\mathbf{Q}(Y_{pre} - Y_{ref})\|^2 + \|\mathbf{R}\Delta U\|^2 \tag{4.16}$$

where:

- $\mathbf{Q} \in \mathbb{R}^{3n_p,3n_p}$ and $\mathbf{R} \in \mathbb{R}^{3n_c,3n_c}$ are two appropriately chosen weighting matrices
- $Y_{ref} = \begin{bmatrix} \mathbf{0}_{3,1} \\ \vdots \\ \mathbf{0}_{3,1} \end{bmatrix} \in \mathbb{R}^{3n_p,1}$ is the vector containing the desired $\mathbf{q}_v$ over the n_p time steps

The online optimization problem to be solved at each time step by the MPC can be defined, adding the constraints on the control and on the variation of the control command between time steps, to make it as smooth as possible:

$$\begin{cases} \min \mathbf{J} \\ u_{min} \leq u \leq u_{max} \\ \Delta u_{min} \leq \Delta u \leq \Delta u_{max} \end{cases} \tag{4.17}$$

In order to be solved, the optimization can be converted into a *quadratic programming* (QP) problem through a mathematical reformulation of the objective function and of the constraints. Defining:

$$E = Y_{ref} - \mathbf{S}x - \mathbf{W}U_{old} \tag{4.18}$$

The objective function becomes:

$$\mathbf{J} = \|\mathbf{Q}(\mathbf{WL}\Delta U - E)\|^2 + \|\mathbf{R}\Delta U\|^2 \tag{4.19}$$

Carrying out the calculations it is possible to come up with the following quadratic form:

$$\mathbf{J} = \Delta U^T \mathbf{H} \Delta U + f^T \Delta U \tag{4.20}$$

where:

- $\mathbf{H} = \mathbf{L^T W^T Q^T Q W L} + \mathbf{R^T R}$
- $f = -2\mathbf{L^T W^T Q^T Q E}$

In addition, the constraints can be rewritten as:

$$\begin{bmatrix} \mathbf{L_1} \\ -\mathbf{L_1} \\ \mathbf{L_2} \\ -\mathbf{L_2} \end{bmatrix} \Delta U \leq b_c \tag{4.21}$$

where

$$\mathbf{L_1} = \begin{bmatrix} \mathbb{I}_3 & \mathbf{0}_{3,3} & \dots & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbb{I}_3 & \dots & \mathbf{0}_{3,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \dots & \mathbb{I}_3 \end{bmatrix} \in \mathbb{R}^{3n_c, 3n_c} \tag{4.22}$$

$$\mathbf{L_2} = \begin{bmatrix} \mathbb{I}_3 & \mathbf{0}_{3,3} & \dots & \mathbf{0}_{3,3} \\ \mathbb{I}_3 & \mathbb{I}_3 & \dots & \mathbf{0}_{3,3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{I}_3 & \mathbb{I}_3 & \dots & \mathbb{I}_3 \end{bmatrix} \in \mathbb{R}^{3n_c, 3n_c} \tag{4.23}$$

$$b_c = \begin{bmatrix} \Delta u_{max}(k) \\ \vdots \\ \Delta u_{max}(k + n_c - 1) \\ -\Delta u_{min}(k) \\ \vdots \\ -\Delta u_{min}(k + n_c - 1) \\ u_{max}(k) - u_{old} \\ \vdots \\ u_{max}(k + n_c - 1) - u_{old} \\ u_{old} - u_{min}(k) \\ \vdots \\ u_{old} - u_{min}(k + n_c - 1) \end{bmatrix} \tag{4.24}$$

The quadratic programming problem is solved online at each time step, through the MATLAB embedded function *quadprog*.

4.2 Manipulator Control

In the following section the two different control strategies adopted for the manipulator's controller are discussed: a Linear Time-Varying MPC and a sliding mode control. A comparison between the two controller is then carried out in chapter 5.

4.2.1 Linear Time-Varying MPC

As previously discussed in the section on the attitude control, MPC is a complex modern control strategy that is quite computationally expensive. Up to now, several much simpler controllers such as PID are widely used for the control robot manipulators. However, these methods rarely take into account dynamic constraints[56]. For this reason, in this thesis, a linear time-varying MPC is proposed, taking as a reference the work of Quirong Tang et al. [56]. The dynamics equation of the manipulator, derived in chapter 2, is recalled below:

$$\mathbf{B}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} = \tau \tag{4.25}$$

Let $x = [x_1,\ x_2]^T = \left[\theta,\ \dot{\theta}\right]^T$ be the vector of the state variables. The state space representation of the dynamic model at time t is then:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = k(x_1)u(t) + g(x_1, x_2) \\ y(t) = x_1 \end{cases} \tag{4.26}$$

where $k = \mathbf{B}^{-1}$ and $g = -\mathbf{B}^{-1}(\mathbf{C}\dot{\theta})$. Following [56] the state variable is discretized through the first and second Taylor expansion as follows:

$$x(t + \Delta t) = \begin{bmatrix} x_1(t + \Delta t) \\ x_2(t + \Delta t) \end{bmatrix} = \begin{bmatrix} x_1(y) + \dot{x}_1(y)\Delta t + \ddot{x}_1(y)\ \frac{\Delta t^2}{2} \\ x_2(t) + \dot{x}_2(t)\Delta t \end{bmatrix} \tag{4.27}$$

substituting equation 4.26 into the expression above, a linear state space equation can be obtained:

$$\begin{cases} x(i+1) = \mathbf{A}x(i) + \mathbf{B}u(i) + \mathbf{D} \\ y(i) = \mathbf{C}x(i) \end{cases} \tag{4.28}$$

where

$$A = \begin{bmatrix} \mathbb{I}_n & \Delta t\mathbb{I}_n \\ \mathbf{0}_{n,n} & \mathbb{I}_n \end{bmatrix},\ B = \begin{bmatrix} \mathbf{k}\frac{\Delta t^2}{2} \\ \mathbf{k}\Delta t \end{bmatrix},\ C = \begin{bmatrix} \mathbb{I}_n & \mathbf{0}_{n,n} \end{bmatrix},\ D = \begin{bmatrix} \mathbf{g}\frac{\Delta t^2}{2} \\ \mathbf{g}\Delta t \end{bmatrix} \tag{4.29}$$

It is important to notice that, in order to linearize the state space equation, these matrices are considered constant throughout the whole prediction horizon. This is a strong assumption since, in general, $\mathbf{k}$ and $\mathbf{g}$ depends on the manipulator's configuration, which varies with time. However, the MPC is linear time-varying, which means that at each time step $\mathbf{k}$ and $\mathbf{g}$ and, consequently, $\mathbf{B}$ and $\mathbf{D}$ are updated on-line with the current manipulator's configuration. Once the state-space equation is defined, it is possible to define the vector containing the predicted

outputs at each time-step of the prediction horizon similarly to the case of the spacecraft's attitude.

$$\begin{cases} Y_{pre} = \mathbf{S}x + \mathbf{W}U + \mathbf{V} \\ U = U_{old} + \mathbf{L}\Delta U \end{cases} \tag{4.30}$$

The matrices and the vectors are similar to the one defined in the previous section, but with appropriate dimensions. In addition, the matrix $\mathbf{V}$ is defined as:

$$\mathbf{V} = \begin{bmatrix} \mathbf{CD} \\ \mathbf{C(AD + D)} \\ \vdots \\ \mathbf{C(A^{n_p-1}D + A^{n_p-2}D + ... + AD + D)} \end{bmatrix} \tag{4.31}$$

Then, the optimization problem is defined:

$$\begin{cases} \min \|\mathbf{Q}(Y_{pre} - Y_{ref})\|^2 + \|\mathbf{R}\Delta U\|^2 \\ u_{min} \leq u \leq u_{max} \\ \Delta u_{min} \leq \Delta u \leq \Delta u_{max} \end{cases} \tag{4.32}$$

Finally, the optimization is transposed to a quadratic programming problem. Let $E = Y_{ref} - \mathbf{S}x - \mathbf{W}U_{old} - \mathbf{V}$. By substituting into 4.30 and carrying out the calculations it follows:

$$\mathbf{J} = \Delta U^T \mathbf{H} \Delta U + f^T \Delta U \tag{4.33}$$

where:

- $\mathbf{H} = \mathbf{L^T W^T Q^T Q W L} + \mathbf{R^T R}$
- $f = -2\mathbf{L^T W^T Q^T Q E}$

Finally, the constraints are rewritten as:

$$\begin{bmatrix} \mathbf{L_1} \\ -\mathbf{L_1} \\ \mathbf{L_2} \\ -\mathbf{L_2} \end{bmatrix} \Delta U \leq b_c \tag{4.34}$$

where $\mathbf{L_1}$, $\mathbf{L_2}$ and b_c are defined similarly to the attitude case, but with appropriate dimensions.

4.2.2 Sliding Mode Control

The sliding mode control is a well established approach for the control of systems with non-linear dynamics. The SMC is characterized by a robust mathematical

base, and, in general, has proven a good rejection of disturbances over a limited knowledge of the system dynamics[57]. The principle behind the SMC is to drive the state of the system towards a surface σ defined in the state-space, and called *sliding surface*. Then, the controller keeps the system's state on the surface and guides it toward the desired steady-state equilibrium point. Therefore, the controller can be implemented in two steps:

1. Definition of the sliding surface σ
2. Design of the control law to make the surface attractive

The sliding surface is typically a function of the state error and its derivatives. A rather common choice is to define σ as a polynomial function as follows.

$$\sigma = \sum_{i=0}^{N} k_i e^{(i)} \tag{4.35}$$

where

- e is the state error
- $e^{(i)}$ denotes the i^{th} derivative of the state error
- the coefficients k_i are usually positive definite constants with appropriate dimension.

Let $P(x) = \sum_{i=0}^{N} k_i x^i$ be the polynomial associated to σ. The choice of the coefficients k_i is crucial for the stabilization of the system, since they have to lead to roots of $P(x)$ with a negative real part. In such a way, when the state will reach the sliding surface, the tracking error will exponentially drop to zero. Since the purpose of the SMC for this thesis is the tracking of the optimal trajectory, the state error considered is the tracking error $e = \theta_d - \theta$. The sliding surface is defined as:

$$\sigma = \ddot{e} + \mathbf{K}_d \dot{e} + \mathbf{K}_p e \tag{4.36}$$

where $\dot{e} = \dot{\theta}_d - \dot{\theta}$ and $\ddot{e} = \ddot{\theta}_d - \ddot{\theta}$. Once the sliding surface is designed, it is possible to define the control law to make the surface attractive, starting with the dynamics of the system.

$$\mathbf{B}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} = \tau \tag{4.37}$$

Imposing the sliding surface equal to zero and rewriting the expression to explicit $\ddot{\theta}$ it is possible to obtain:

$$\sigma = 0 \Rightarrow \ddot{\theta} = \ddot{\theta}_d + \mathbf{K_d}\dot{e} + \mathbf{K_p} e \tag{4.38}$$

Substituting into 4.37 it follows:

$$\tau = \mathbf{B}(\theta)(\ddot{\theta}_d + \mathbf{K_d}\dot{e} + \mathbf{K_p}e) + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} \tag{4.39}$$

To make the surface more attractive, usually a term proportional to the sign of the sliding surface is added to the command input. However, since the sign function is discontinuous, this may lead to the so-called chattering, a discontinuous high-frequency control. In the case of the manipulator this is particularly problematic, since the chattering may cause relevant disturbances to the base satellite, rendering useless the effort of the guidance to generate a low disturbance trajectory. To overcome this issue usually continuous functions (like sigmoids) are employed. In this work a term proportional to the hyperbolic tangent is added to the control input as follows:

$$\tau = \mathbf{B}(\theta)(\ddot{\theta}_d + \mathbf{K_d}\dot{e} + \mathbf{K_p}e) + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} + c\ tanh(\sigma) \tag{4.40}$$

where c is a positive coefficient of appropriate dimension.
Furthermore, additional attention must be paid to the choice of the coefficient matrices $\mathbf{K}_d$ and $\mathbf{K}_p$, which have to be carefully tuned in order to preferably obtain an over-dumped response the system. Even though the reaching time of the system may be larger, an over-dumped design will avoid excessive oscillations around the desired trajectory, further minimizing the base disturbances on the base satellite.

Chapter 5

Simulation and Results

In the following chapter, the simulation environment is described and the results of the simulations are discussed. The chapter is organized as follows: at first the MATLAB/SIMULINK simulator is briefly introduced. Then, the different simulations are described. The results of both the guidance algorithm and the simulation of the trajectory tracking are then reported and discussed. The different actuators modeled for the attitude control (thrusters and reaction wheels) are tested and compared in performance. Afterwards, the same maneuver is performed with both the SMC and the linear time varying MPC for the trajectory tracking of the manipulator. The results are shown and discussed. Finally, the results of the tracking error during a Montecarlo simulation are reported for both the SMC and the linear time-varying MPC. The different performance are discussed and conclusion are drawn on which may be the best solution and on how to eventually improve the controllers.

5.1 Simulator Overview

The simulator is implemented in a MATLAB/SIMULINK environment and it is composed of two main subsystems:

- 3 DoF base satellite attitude
- 7 DoF robotic arm

The schematics of the simulator are depicted in fig. 5.1. It is possible to notice that the dynamics of the two subsystems influence one another. The attitude model receives the base torque disturbance as input form the robotic arm, and the manipulator model receives the the satellite's angular velocity and acceleration. The two subsystems are now discussed in detail.

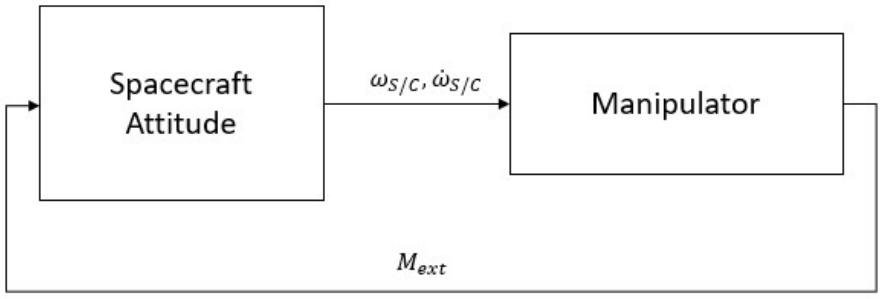


Figure 5.1: MATLAB/SIMULINK complete space manipulator simulator

5.1.1 Attitude Subsystem

The attitude simulator is reported in fig 5.2. The subsystem is composed by four

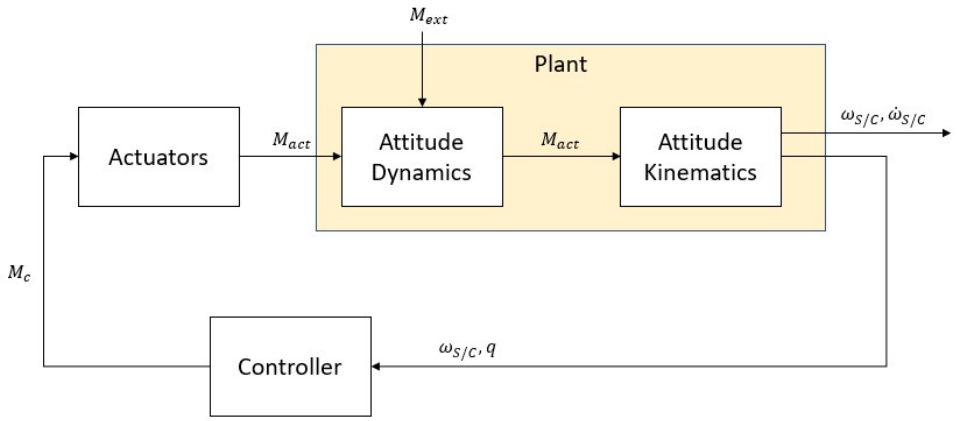


Figure 5.2: Attitude subsystem

main parts:

- **Attitude Dynamics**: It contains the Euler's equations derived in chapter 2. It receives the actuator's torque and angular momentum as input, together with the base disturbance generated by the robot manipulator. The block computes the angular acceleration and velocity of the spacecraft as output.

- **Quaternion Kinematics**: It receives the spacecraft's angular velocity as input and computes the unit quaternion describing the attitude state of the spacecraft. Together with the attitude dynamics block, they constitute the subsystem of the plant dynamics.
- **Controller**: It contains the model predictive control algorithm, comprehensive of both the predictive model and the optimizer. It receives the attitude state (angular velocity and quaternion) as input and computes the desired control command in terms of torque.
- **Actuator**: It contains the model of the attitude actuators which can be either reaction wheels or thrusters (PWPF filter). The schematics of these models have already been described in detail in chapter 2. The block receives the desired control torque generated by the MPC and gives the actuated control torque as output.

5.1.2 Manipulator Subsystem

The schematics of the robot manipulator subsystem are reported in fig. 5.3. From

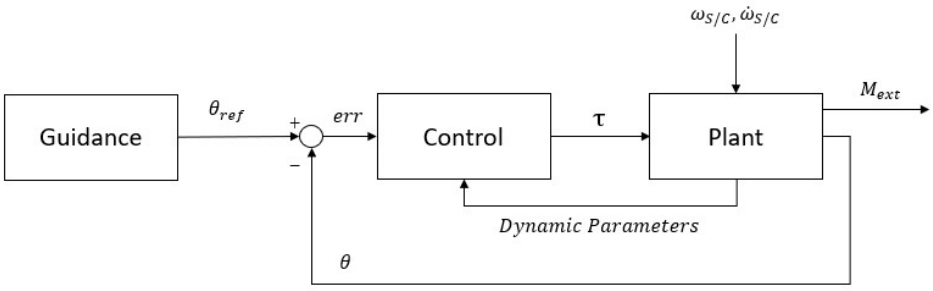


Figure 5.3: Manipulator subsystem (MPC version)

the figure, three main components can be identified:

- **Guidance**: It gives as output the optimal reference trajectory already computed off-line by the PSO algorithm.
- **Control**: Two different version of this block were implemented. It contains either the linear time-varying MPC or the sliding mode controller for the

trajectory tracking of the manipulator. The block takes the state of the system $x = [\theta, \dot{\theta}]^T$ as input, together with the dynamics parameters computed in the plant dynamics subsystem $\mathbf{B}(\theta)$ and $\mathbf{C}(\theta, \dot{\theta})\dot{\theta}$, and the reference trajectory generated by the guidance block. The algorithm then computes the desired command torque τ as output.

- **Plant Dynamics**: this block takes as input the desired command torque τ computed by the controller and the angular velocity and acceleration of the base satellite. It computes the state variables of the system $(\theta, \dot{\theta}, \ddot{\theta})$ using the equation of the dynamics of the manipulator. In addition, the dynamics matrices $\mathbf{B}(\theta)$ and $\mathbf{C}(\theta, \dot{\theta})\dot{\theta}$ are computed with the Lagrangian and Newton-Euler approach, respectively. The state variable and the dynamics parameters are given as output to the controller. Finally, the block computes and gives as output the base reaction torque of the manipulator, generated during the maneuver.

5.2 Results

During the simulated maneuver the end-effector of the robotics arm moves from an initial position and orientation $s_0 = \begin{bmatrix} x_0 & y_0 & z_0 & q_0 & q_{x0} & q_{y0} & q_{z0} \end{bmatrix}^T$ which corresponds to the deployed configuration of the arm, $\theta_0 = [90,\ 140,\ -90,\ 90,\ 0,\ 0,\ 0]^T$ in joint space, to a final position and orientation s_f. For the Monte Carlo simulations 100 random s_f are generated.

5.2.1 Guidance Results

A Monte Carlo simulation was performed to test the capability of convergence of the PSO algorithm. 100 random final poses were generated and the optimal trajectory to steer the end-effector from s_0 to s_f was computed. The results of the simulation is reported in fig. 5.4. From the figure it is possible to notice that the algorithm converged in most of the cases after 100/200 iterations. However, in about the 10% of the cases the algorithm did not converge to a satisfying solution, and the iteration stopped because the maximum number of iterations was reached. The reason behind this phenomenon has to be searched in the PSO behaviour. Indeed, as anticipated in chapter 4, the algorithm is defined in such a way that does not guarantee the convergence to the global minimum. When the function to minimize is rather complex, several local minima are present, and the swarm may eventually be trapped by one of them, leading to the stagnation of the PSO. During the stagnation, no improvements are made on the personal best position or global best position of the swarm, making it impossible to escape the local

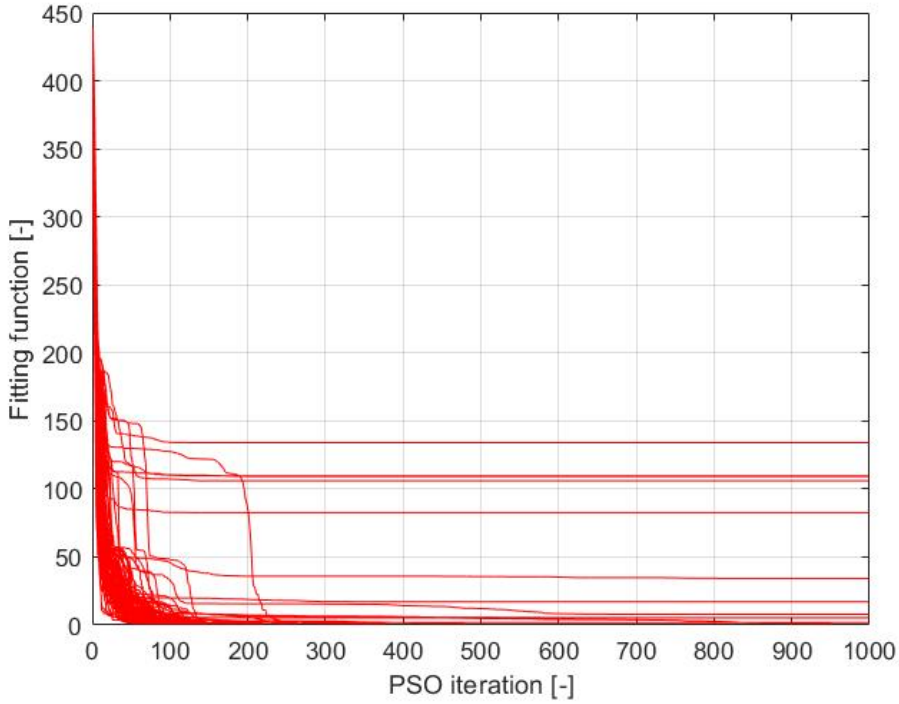


Figure 5.4: Fitting function Vs Iterations for the Monte Carlo simulation

minimum. Several strategies for stagnation avoidance already exists in literature, for example[58]:

- **Modified PSO**, where the algorithm is slightly improved by modifying the tuning parameters (e.g. adaptive inertia weights)
- **PSO with mutation**, where a random mutating agent is added to the formula used to update the population. This will help to increase the diversity and eventually escape the local minimum.
- **Hybrid PSO**, where the PSO is used in combination with other meta-heuristic algorithms such as genetic algorithm or differential evolution.
- **Multi Swarm PSO**, where more than one swarm is defined. In this way a larger search space can be employed and the stagnation should eventually be avoided.

In addition, other approaches for stagnation avoidance can be employed, for example by re-initializing the positions and velocities every time the stagnation occurs.

5.2.2 Trajectory Tracking Results

In this section, the results of the simulation of the maneuver are reported and discussed. The different actuators for attitude control and strategies for trajectory tracking are compared, and the solutions are discussed.

Thrusters Vs Reaction Wheels

To compare the efficiency of the different actuators, a maneuver that aims to move the end-effector from the initial position and orientation s_0 to a random chosen final position $s_f = [0.99, 1.38, 1.08, 0.27, 0.03, 0.27, 0.92]^T$ is performed . The results are reported in fig. 5.5 to fig. 5.11.

From the tracking error it is possible to immediately notice the difference between the continuous action of the reaction wheel and the on-off behaviour of the thrusters. This is true also for the joint torques and the base disturbances, that follows a similar trend in both cases, but presents a series of oscillations in the thrusters' case. However, the magnitude of the reaction torque is contained, therefore preserving the effort of the guidance in computing a trajectory with low disturbance on the base. The attitude is maintained constant throughout the maneuver in both cases, and the angular acceleration of the spacecraft is relatively low, around 10^{-4} rad/s in both cases. Although, a slightly more disturbed angular acceleration for the thrusters is observed , with peaks that keeps occurring even after the manoeuvre is concluded. By comparing the control torques computed by the controllers and the actuated torque, the reason behind the difference of the results between the two simulations is evident. In fact, the PWPF modulation makes it harder for the thrusters to follow the desired torque computed by the controller. This results in additional disturbances and in a much less fine attitude control than the one of the reaction wheels' case. Therefore, the trajectory tracking control of the manipulator must compensate for these additional disturbances, by generating a control that oscillates around the main trend. For the reasons above, the reaction wheels actuation resulted in a better, more efficient attitude control strategy. To improve the thruster actuation, several strategies may be employed. For example thrusters with lower maximum thrust level may be used. In addition, the parameter tuning for the PWPF could be performed following more rigorous methodologies. For instance, PSO based methods have been investigated in literature [24].

SMC Vs MPC

The same maneuver used for the previous simulation is taken as a reference for the comparison between the two controllers implemented for the trajectory tracking of the manipulator. The results of the simulation are reported in fig. 5.12 to fig. 5.18.

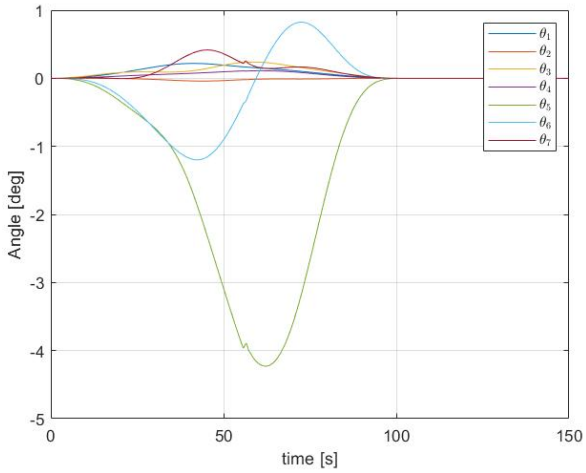
From the comparison of the tracking error it is possible to immediately notice a difference of more than 3 orders of magnitude. The tracking error of the MPC is relatively high, with peaks around 5 deg, whereas the SMC gives an extremely low error of 10^{-4} deg at most. The same phenomenon takes place with the joint torques, which are lower in the case of SMC, with a maximum value of $8 \cdot 10^{-2}$ Nm compared to the higher value of $2.6 \cdot 10^{-1}$ Nm of the MPC case. However, the reaction torque appears to be more or less the same in both cases. The attitude is maintained stable and constant during the whole maneuver. The spacecraft angular velocity is low in both cases, but in the SMC case is 2 orders of magnitude lower than the one in the MPC case. Finally, the control torque generated by the MPC appears to be much greater than the one computed by the MPC.

In addition, to further understand the differences in terms of performance of the two controllers, two Monte Carlo simulations were performed, with 100 maneuvers with a random desired final pose of the end effector each. The results of the Monte Carlo simulations are reported in fig. 5.19 to fig. 5.21.
During the Monte Carlo simulation the MPC was highly unstable and the error diverged in 40% of the simulations. Only the cases where the error did not diverge are reported. For the SMC case, instead, the error converged to zero in 100% of the cases. From the comparison of the tracking error it is possible to notice a very high tracking error generated by the MPC, which in some cases has peaks up to 20 deg. The tracking error of the SMC, on the other hand, is maintained low for all the simulations, at a magnitude of 10^{-4} deg. The control torque, however, appears to have the same average magnitude for both the MPC and the SMC case. Finally, the reaction torque appears to be lower on average in the SMC case. The huge difference between the MPC and the SMC in terms of tracking error can be motivated by the intrinsic characteristics of the linear time-varying MPC. The dynamics of the manipulator is highly non-linear, therefore, the sliding mode was expected to behave as a good controller for the trajectory tracking. On the other hand, the MPC linearizes the dynamics of the manipulator, and, moreover, makes assumptions on the dynamic parameters, which are considered constant at each time-step when the model computes the predicted state. In order to improve the controller, a non-linear MPC should be studied, to better handle the non linearities of the manipulator's dynamics. The selection of a SMC as a controller may result in a good compromise, however, some strategies have to be studied to take into account the constraints given by the limitations of the actuators.

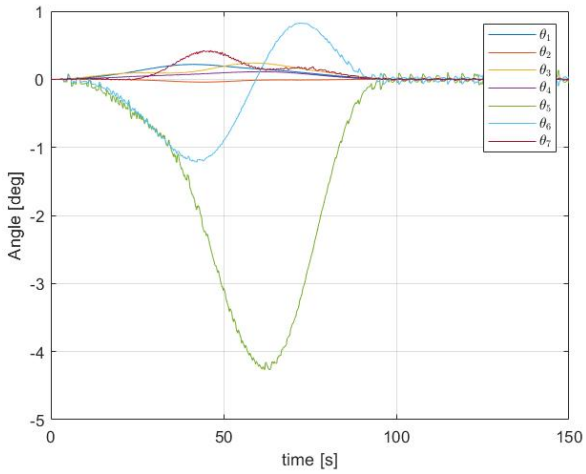
5.3 Discussion of Results

In this section the simulation results are discussed in detail. For what concerns the results of the guidance algorithm, the PSO has been proven to be a promising

approach to the generation of a low disturbance optimal trajectory. In fact, the algorithm plays a crucial role for the success of the maneuver, being able to generate a feasible trajectory while satisfying the constraints on the joint angles, angular velocity and on the reaction torque. However, a well known problem[58] of premature convergence to a sub-optimal solution has raised during the simulation. In fact, as shown in fig.5.4, in 10% of the maneuvers the final trajectory was not satisfying, since the final error of the end-effector's position and orientation was not lower than a tolerance threshold. The reason to this phenomena is given by the fact that, as the iteration number increases, the momenta of the particles reduces and the swarm tends to converge to a single point. The convergence of the algorithm may be a favorable property, however, for particularly complex problems, with many local minima, this may result in a solution which is not the global optimum. It is obvious that, in order to avoid such phenomenon, the whole structure of the algorithm must be modified. As already mentioned in the previous section, several strategies have already been investigated in literature [58] to handle the convergence to a sub-optimal solution.
In addition, the comparison between the linear time-varying MPC and the SMC as highlighted a huge difference in performance between the two control strategies. The sliding mode control is notoriously able to handle the non-linearities of a dynamic system, and, as expected, as given rather satisfying results in therms of tracking error. Furthermore, the low reaction torques of the optimal guidance have been preserved, making it easier for the attitude control to keep the attitude stable during the maneuvers. On the other hand, the linear time-varying MPC was highly unstable and resulted in a divergent tracking error in 40% of the cases. The remaining maneuvers presented, anyway, a very large tracking error, with peaks around 20 degrees. The inability to properly follow the optimal trajectory resulted in a high reaction torque transferred to the base satellite and , therefore, higher effort of the ACS to stabilize the attitude of the spacecraft. The radical performance difference between the two controllers lays on the approximation of the manipulator's dynamics performed in the linear time-varying MPC. At each time step the state-space equation are linearized around the desired point of the optimal trajectory, and the dynamic matrices are assumed constant for the whole duration of the prediction horizon. This presumably results in a poor ability of the predictive model to accurately forecast the future state of the system. Therefore, the control output computed by the MPC does not actually minimize the objective function of the on-line optimization leading to a tracking error higher than expected. In order to overcome this issue a different formulation for the controller must be employed, considering, for instance, a non-linear MPC, in order to better predict the state of the system over the prediction horizon.

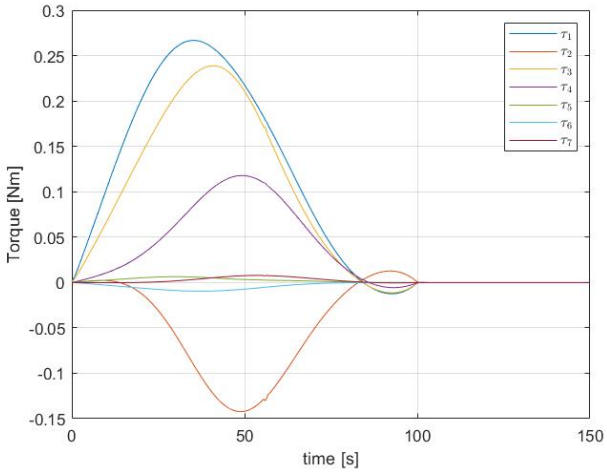


(a) Reaction Wheels

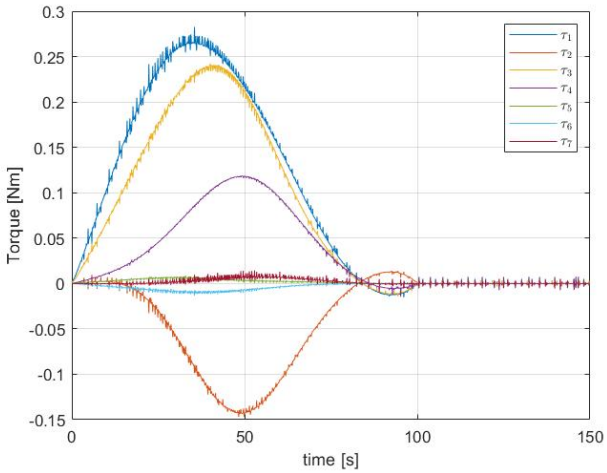


(b) Thrusters

Figure 5.5: Manipulator tracking error with reaction wheels and thrusters as actuators for the ACS

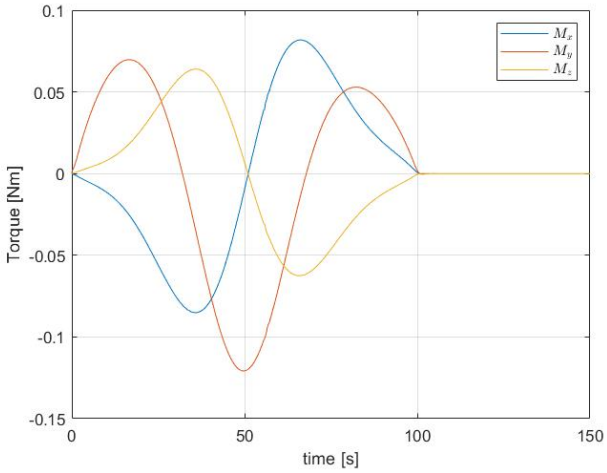


(a) Reaction Wheels

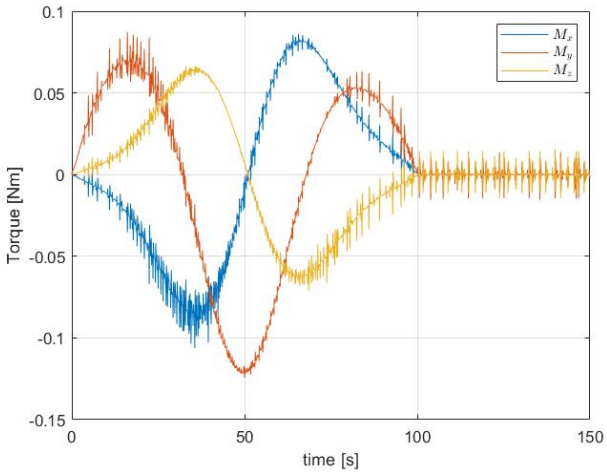


(b) Thrusters

Figure 5.6: Joint torques during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

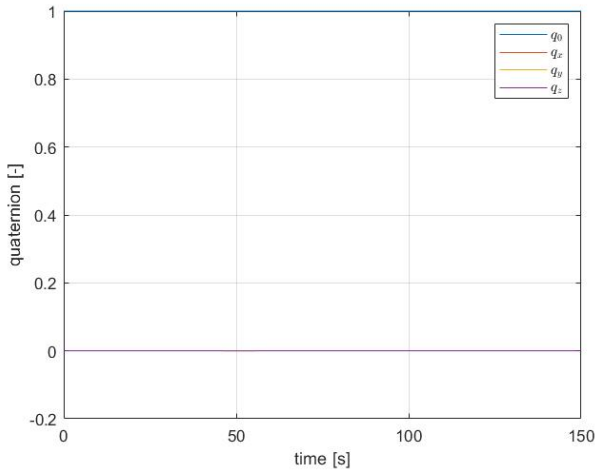


(a) Reaction Wheels

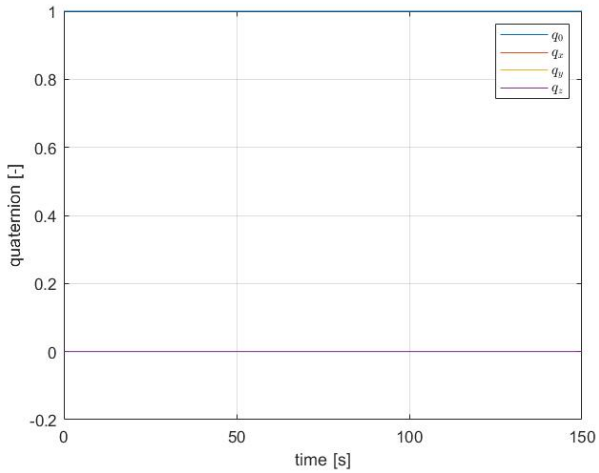


(b) Thrusters

Figure 5.7: Reaction torque during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

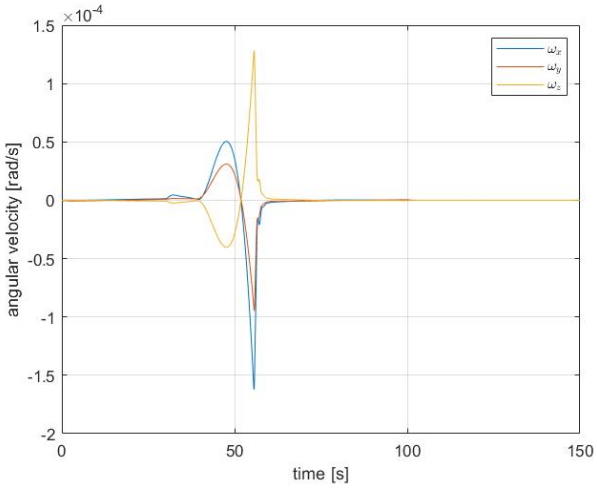


(a) Reaction Wheels

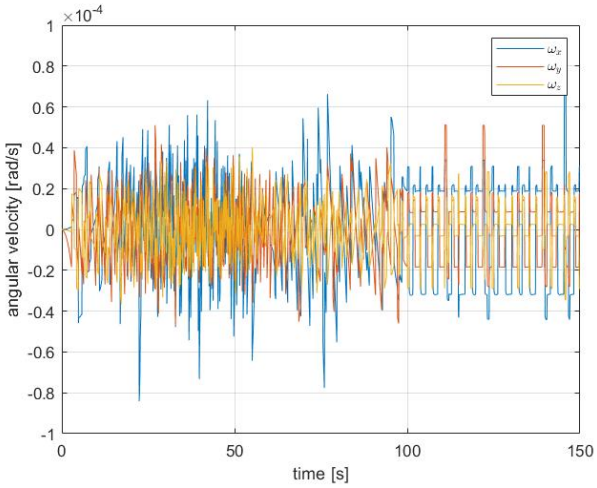


(b) Thrusters

Figure 5.8: S/C quaternions during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

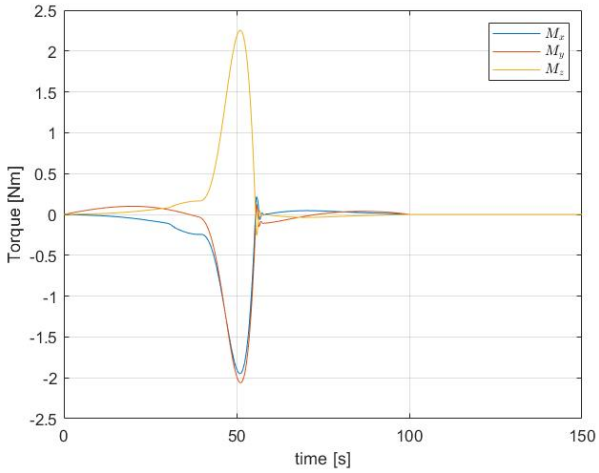


(a) Reaction Wheels

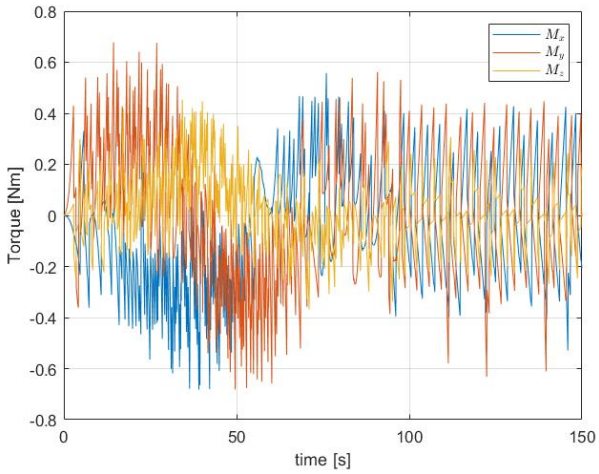


(b) Thrusters

Figure 5.9: S/C angular velocity during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

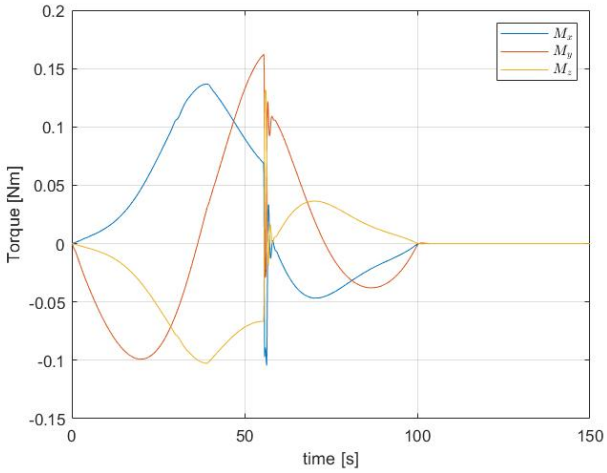


(a) Reaction Wheels

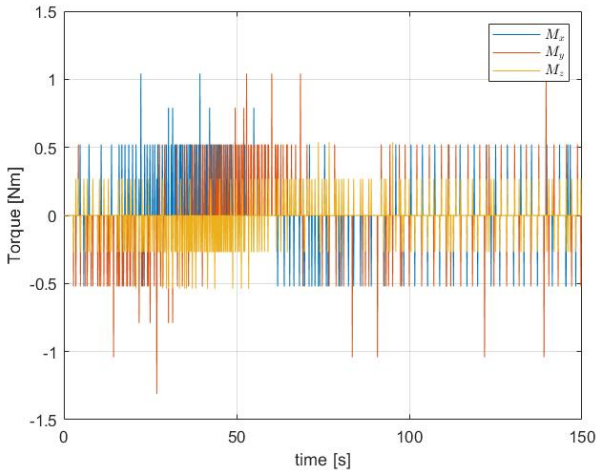


(b) Thrusters

Figure 5.10: Command torque generated by the controller during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

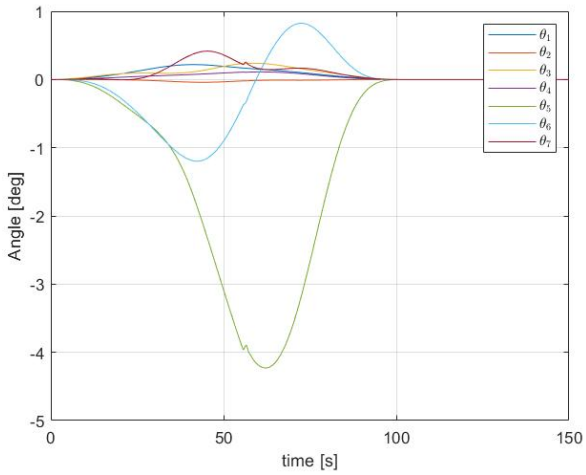


(a) Reaction Wheels

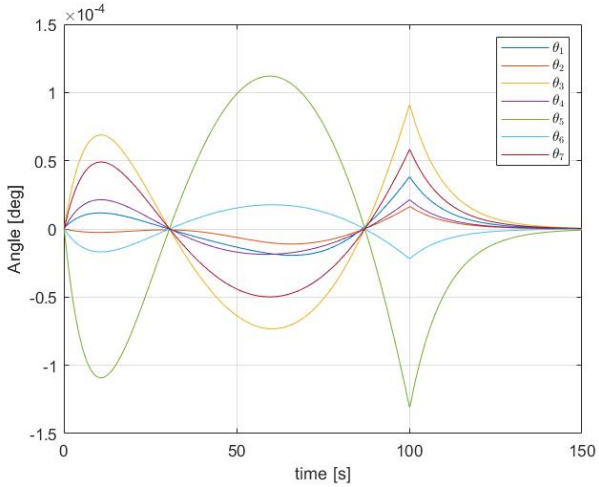


(b) Thrusters

Figure 5.11: Actuated control torque during trajectory tracking with reaction wheels and thrusters as actuators for the ACS

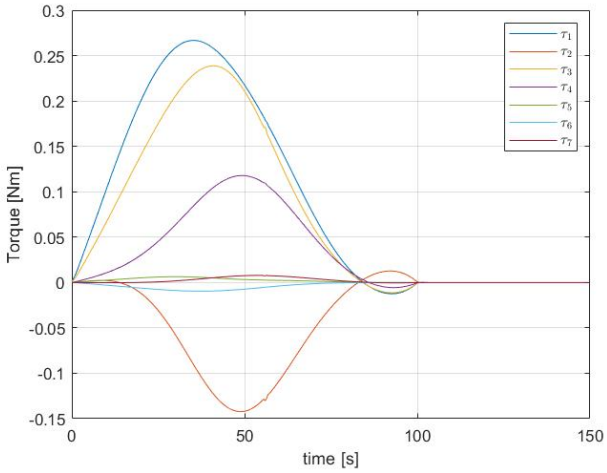


(a) MPC

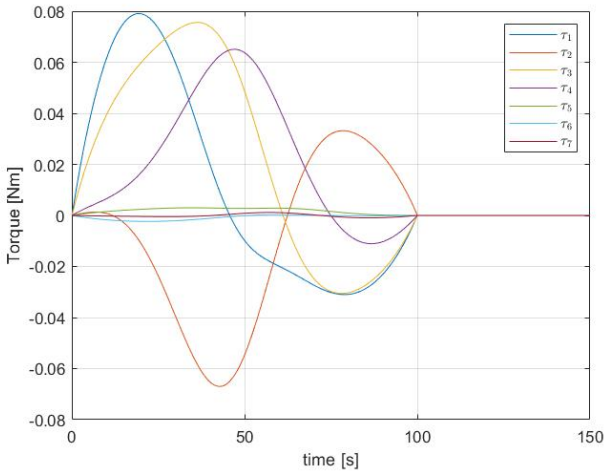


(b) SMC

Figure 5.12: Manipulator tracking error with MPC (a) and SMC (b) as controllers

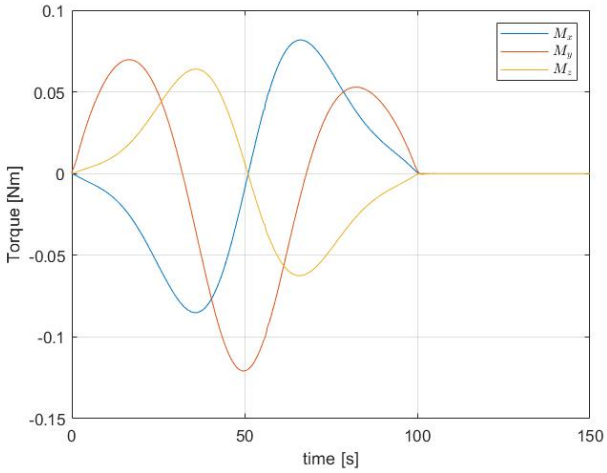


(a) MPC

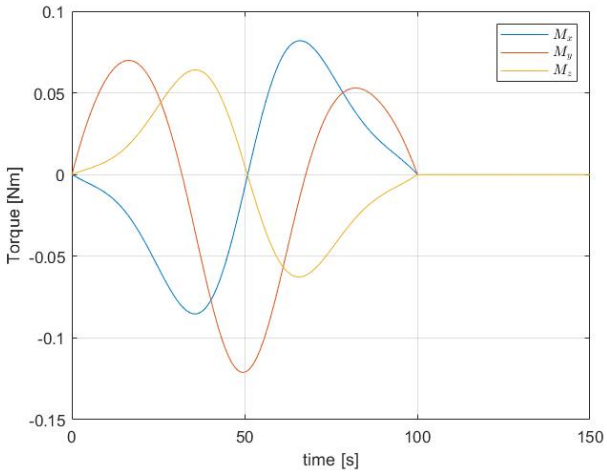


(b) SMC

Figure 5.13: Joint torques during trajectory tracking with MPC (a) and SMC (b) as controllers

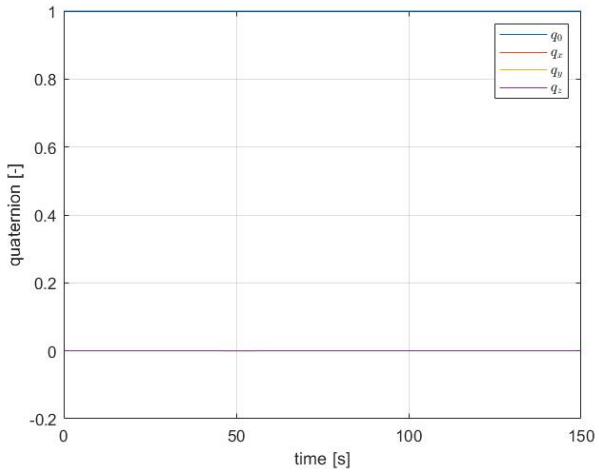


(a) MPC

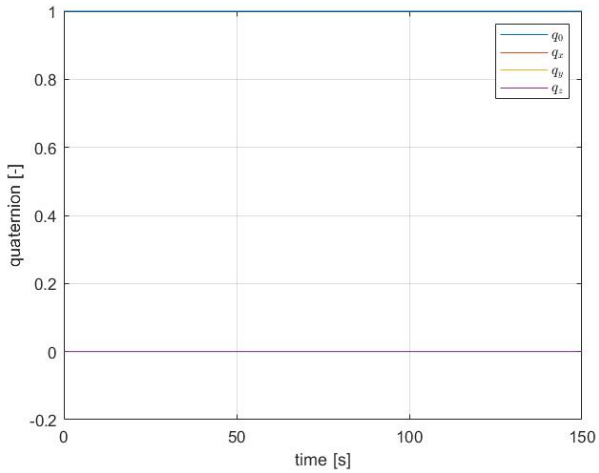


(b) SMC

Figure 5.14: Reaction torque during trajectory tracking with MPC (a) and SMC (b) as controllers

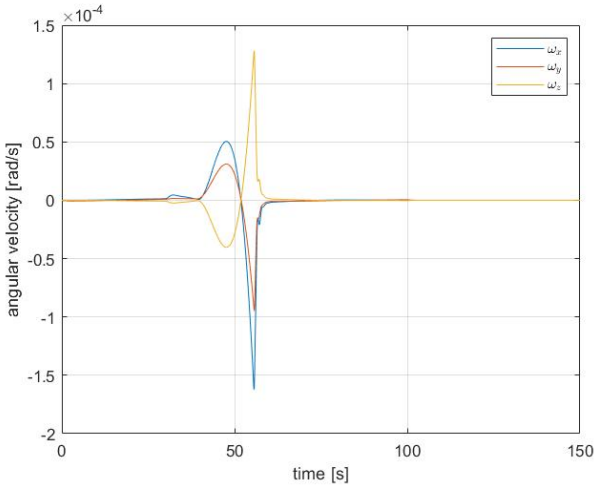


(a) MPC

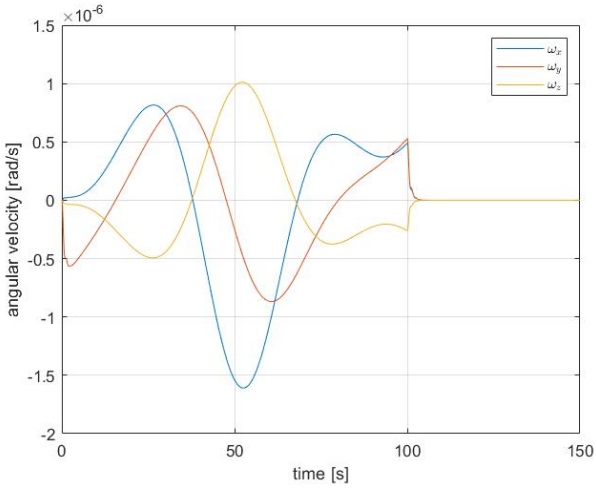


(b) SMC

Figure 5.15: S/C quaternions during trajectory tracking with MPC (a) and SMC (b) as controllers

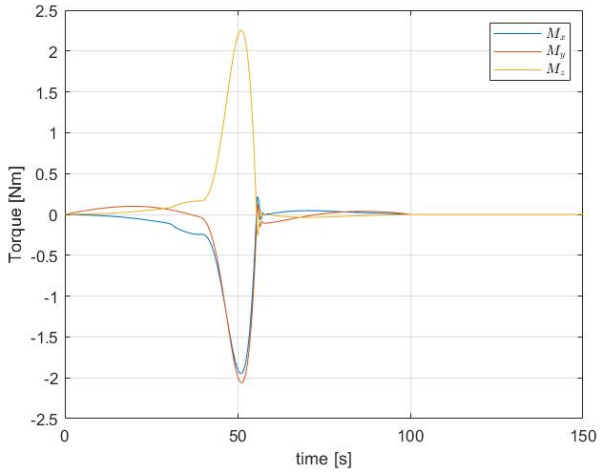


(a) MPC

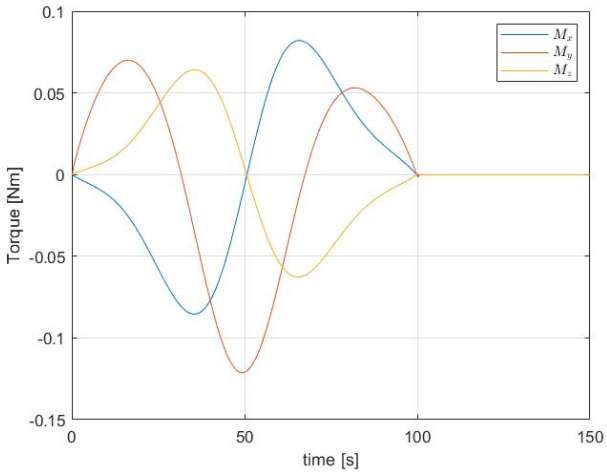


(b) SMC

Figure 5.16: S/C angular velocity during trajectory tracking with MPC (a) and SMC (b) as controllers

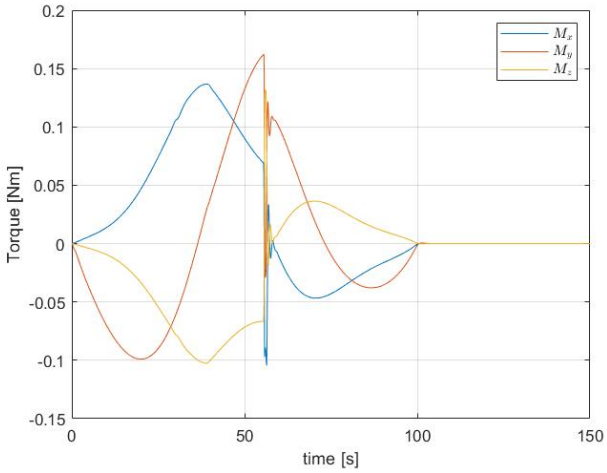


(a) MPC

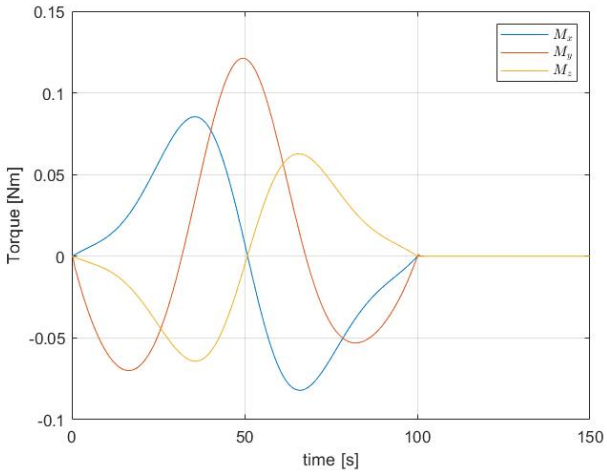


(b) SMC

Figure 5.17: Command torque generated by the controller during trajectory tracking with MPC (a) and SMC (b) as controllers

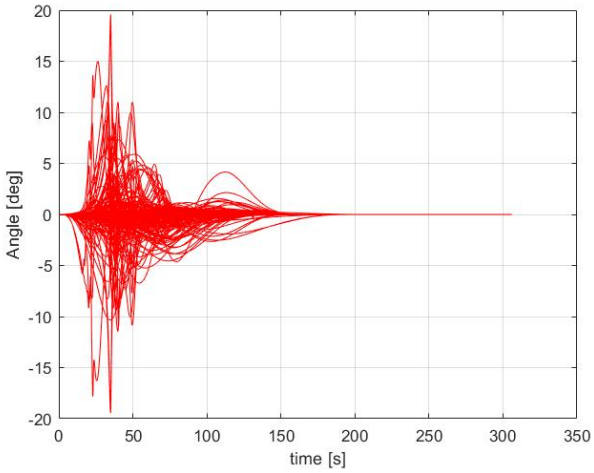


(a) MPC

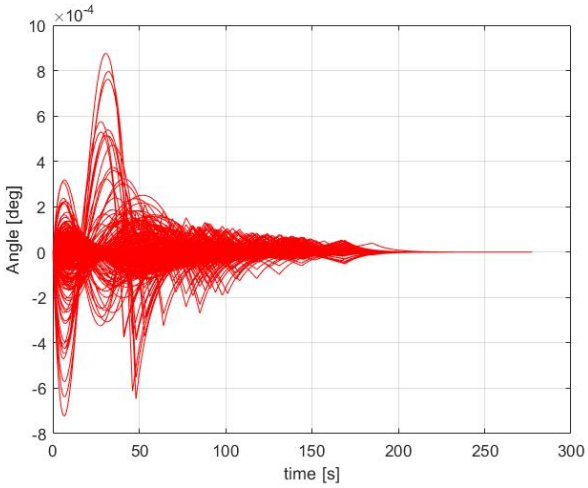


(b) SMC

Figure 5.18: Actuated control torque during trajectory tracking with MPC (a) and SMC (b) as controllers

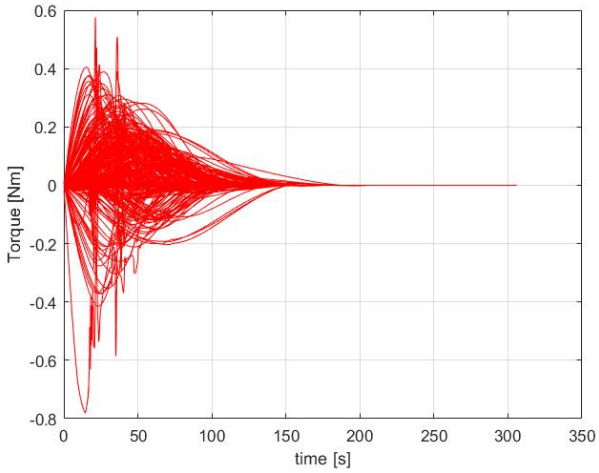


(a) MPC

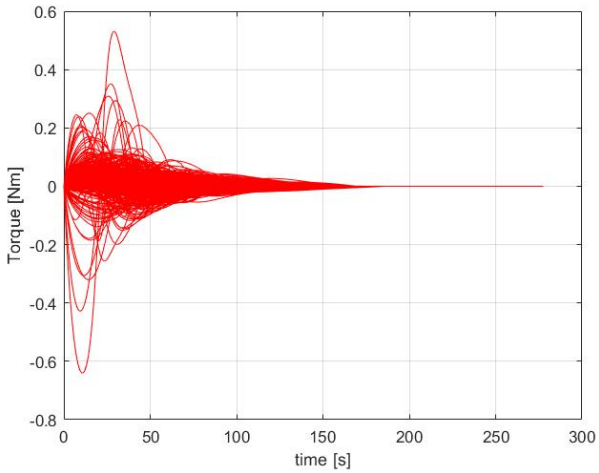


(b) SMC

Figure 5.19: Trajectory tracking error with MPC (a) and SMC (b) as controllers during a Monte Carlo simulation

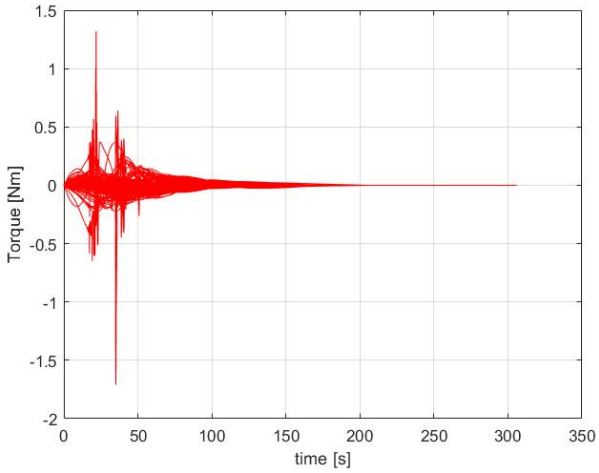


(a) MPC

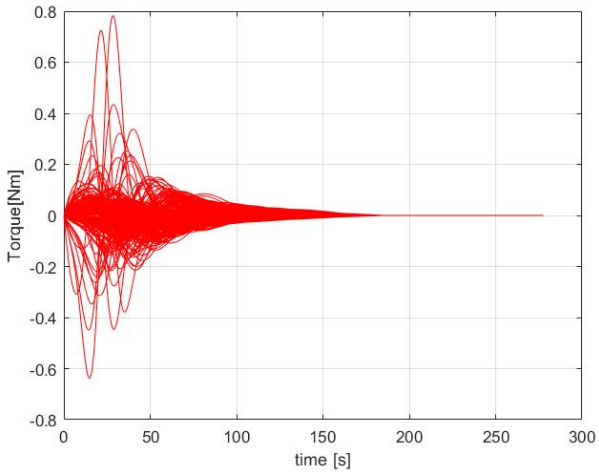


(b) SMC

Figure 5.20: Joint torque with MPC (a) and SMC (b) as controllers during a Monte Carlo simulation



(a) MPC



(b) SMC

Figure 5.21: Reaction torque with MPC (a) and SMC (b) as controllers during a Monte Carlo simulation

Chapter 6

Conclusions and Future Work

The aim of this thesis is to develop an innovative guidance algorithm for a space manipulator. In order to develop and analyze the performances of such algorithm a complete simulator on MATLAB/SIMULINK had to be developed, comprehensive of both the base satellite and the robotic arm. The model simulated the attitude dynamics of the base spacecraft, in order to analyze the effects of the reaction disturbances exerted by the manipulator's motion on the whole system. The trajectory planning was treated as an optimization issue and solved off-line with a meta-heuristic algorithm called *Particle Swarm Optimization*, a method inspired by to movement of the swarms of birds. Then, a Monte Carlo simulation was performed to test the convergence of the algorithm, solving 100 different trajectory planning problems. The simulation suggested that the PSO may be a promising method for the guidance of space manipulators, being able to converge in 90% of the cases. This approach gives several advantages, for instance the possibility to impose constraints on the join angles, velocities and on the reaction torque generated by the movement of the manipulator. In addition to the guidance algorithm, several controllers were implemented to track the optimal trajectory generated by the PSO. In particular, a linear time-varying MPC and a sliding mode controller. The two controllers were tested, together with an MPC for the attitude control of the spacecraft, with a Montecarlo simulation, performing 100 different maneuvers of the manipulator. The simulation suggested that the SMC would be a good solution for the tracking of the optimal trajectory, given its ability to deal with non linear dynamics such as the one of the manipulator. The linear time-varying MPC, on the other had did not give satisfying results, showing instability in 40% of the cases. This may be due to the linearization of the manipulator's dynamics which, as a consequence, makes the model unable to accurately predict the evolution of the

system over the prediction horizon. As a result of these simulations, this work has opened the scenario to several ideas for further improvements to be eventually studied in future works. For instance, the convergence rate of the algorithm can be further increased by adopting a strategy for stagnation handling. The PSO can be modified, used in combination with other meta-heuristic optimization methods (such as genetic algorithms or differential evolution), or a multi swam PSO may be eventually employed to ensure the convergence to a satisfying solution. Furthermore, several improvements may be applied to the controller for the trajectory tracking, adopting a non-linear MPC to better handle the non-linearities of the manipulator's dynamic. In addition, a more complete simulator may be studied, introducing the complete model of the space manipulator's dynamics using, for instance, the generalized jacobian method. Finally, in order to develop a more realistic model, the environment disturbances (such as the gravity gradient and the solar pressure) may be taken into account, together with a model of the joint motors, to actuate the desired joint torques generated by the controller.

Bibliography

[1] Angel Flores Abad, Ou Ma, Khanh Pham, and Steve Ulrich. «A review of space robotics technologies for on-orbit servicing». In: *Progress in Aerospace Sciences* 68 (July 2014). DOI: 10.1016/j.paerosci.2014.03.002 (cit. on pp. 1, 4, 25).

[2] Wei-Jie Li et al. «On-orbit service (OOS) of spacecraft: A review of engineering developments». In: *Progress in Aerospace Sciences* 108 (May 2019). DOI: 10.1016/j.paerosci.2019.01.004 (cit. on pp. 1, 4, 5).

[3] NASA. *STS061-98-50*. [Online; accessed February 12, 2022]. 1997. URL: https://eol.jsc.nasa.gov/SearchPhotos/photo.pl?mission=STS061&roll=98&frame=50 (cit. on pp. 2, 3).

[4] NASA. «On-orbit satellite servicing study project report». In: (Oct. 2010) (cit. on p. 2).

[5] J. Sasiadek. «Space Robotics and its Challenges». In: (Feb. 2019) (cit. on pp. 2, 3).

[6] Walter Peart. «Dynamic Influence of Flexible Payloads on Space Shuttle RMS». In: *Journal of Aerospace Engineering - J AEROSP ENG* 9 (Apr. 1996). DOI: 10.1061/(ASCE)0893-1321(1996)9:2(39) (cit. on p. 3).

[7] Christian Sallaberger. «Canadian space robotic activities». In: *Acta Astronautica* 41 (Aug. 1997). DOI: 10.1016/S0094-5765(98)00082-4 (cit. on p. 2).

[8] NASA. *Remote Manipulator System (Canadarm2)*. [Online; accessed February 14, 2022]. 2018. URL: https://www.nasa.gov/mission_pages/station/structure/elements/remote-manipulator-system-canadarm2/ (cit. on p. 3).

[9] NASA. [Online; accessed February 14, 2022]. URL: https://www.nasa.gov/sites/default/files/thumbnails/image/iss041e049099.jpg (cit. on p. 4).

[10] Patten Laryssa, Evans Lindsay, Oshinowo Layi, Ochisor Marius, Kazuharu Nara, Lodewijk Aris, and Tabarah Ed. «International Space Station Robotics: A Comparative Study of ERA, JEMRMS and MSS». In: (Nov. 2002) (cit. on p. 4).

[11] G. Hirzinger, B. Brunner, J. Dietrich, and Johann Heindl. «ROTEX-the first remotely controlled robot in space». In: (June 1994), 2604–2611 vol.3. DOI: 10.1109/ROBOT.1994.351121 (cit. on p. 4).

[12] «European Robotic Arm (ERA), Large relocatable symmetrical robotic arm with 7 degrees of freedom». In: ESA-HSO-COU-007 (2012) (cit. on p. 5).

[13] Juergen Telaar, stéphane Estable, Marco De Stefano, Wolfgang Rackl, Roberto Lampariello, Finn Ankersen, and Jesus Gil-Fernandez. «Coupled Control of Chaser Platform and Robot Arm for the e.Deorbit Mission». In: May 2017 (cit. on p. 6).

[14] Robin Biesbroek, Tiago Soares, Jakob Husing, and Luisa Innocenti. «The e.Deorbit CDF Study: A Design Study for the Safe Removal of a Large Space Debris». In: 2013 (cit. on p. 6).

[15] ESA. [Online; accessed February 14, 2022]. URL: https://blogs.esa.int/cleanspace/2018/12/18/from-active-debris-removal-to-in-orbit-servicing-the-new-trajectory-of-e-deorbit-part-two (cit. on p. 6).

[16] Juergen Telaar et al. «GNC Architecture for the e.Deorbit Mission». In: Jan. 2017 (cit. on p. 6).

[17] J. Telaar, I. Ahrns, S. Estable, R. Lampariello, W. Rackl, G. Panin, E. di Soto, N. Santos, and M. Canetri. «e.Deorbit Phase B1 GNC and Combined Control Simulation Results». In: May 2016 (cit. on p. 7).

[18] Steffen Jaekel et al. «Design and Operational Elements of the Robotic Subsystem for the e.deorbit Debris Removal Mission». In: *Frontiers in Robotics and AI* 5 (Aug. 2018), p. 100. DOI: 10.3389/frobt.2018.00100 (cit. on p. 8).

[19] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing. Springer London, 2001. ISBN: 9781852332211. URL: https://books.google.it/books?id=v9PLbcYd9aUC (cit. on pp. 11, 19–21, 23, 24, 26, 28).

[20] Arend L Schwab. «Quaternions, finite rotation and euler parameters». In: *Cornell University Notes, Ithaca NY* (2002), p. 28 (cit. on p. 11).

[21] Giulio Avanzini. «Spacecraft Attitude Dynamics and Control». In: *Politecnico di Torino, Tech. Rep* (2008) (cit. on pp. 11, 12).

[22] Abolfazl Shirazi and Mehran Mirshams. «Pyramidal reaction wheel arrangement optimization of satellite attitude control subsystem for minimizing power consumption». In: *International Journal of Aeronautical and Space Sciences* 15.2 (2014), pp. 190–198 (cit. on pp. 13, 14).

[23] Arantes Gilberto, Luiz Martins-Filho, and Adrielle Santana. «Optimal On-Off Attitude Control for the Brazilian Multimission Platform Satellite». In: *Mathematical Problems in Engineering* 2009 (Jan. 2009). DOI: 10.1155/2009/750945 (cit. on p. 15).

[24] Alireza Khosravi and Pouria Sarhadi. «Tuning of Pulse-Width Pulse-Frequency Modulator using PSO: An Engineering Approach to Spacecraft Attitude Controller Design». In: *Automatika* 57 (July 2016). DOI: 10.7305/automatika.2016.07.618 (cit. on pp. 16, 53).

[25] T Krovel. «Optimal tuning of PWPF modulator for attitude control». PhD thesis. Master Thesis, Norwegian University of Science and Technology, 2005 (cit. on p. 16).

[26] Elisa Capello, Andrea Bacchetta, and Pietro Chevallard. «Mission Design and Modeling of Space Manipulators for In-Orbit Servicing Missions.» In: (2019) (cit. on p. 17).

[27] Z. Vafa and S. Dubowsky. «On the dynamics of manipulators in space using the virtual manipulator approach». In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. 1987, pp. 579–585. DOI: 10.1109/ROBOT.1987.1088032 (cit. on p. 25).

[28] Bin Liang, Yangsheng Xu, and Marcel Bergerman. «Mapping a Space Manipulator to a Dynamically Equivalent Manipulator». In: *Journal of Dynamic Systems, Measurement, and Control* 120.1 (Mar. 1998), pp. 1–7. ISSN: 0022-0434. DOI: 10.1115/1.2801316. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/120/1/1/5779050/1_1.pdf. URL: https://doi.org/10.1115/1.2801316 (cit. on p. 25).

[29] Yoji UMETANI and Kazuya Yoshida. «Resolved Motion Rate Control of Space Manipulators with Generalized Jacobian Matrix». In: *Robotics and Automation, IEEE Transactions on* 5 (July 1989), pp. 303–314. DOI: 10.1109/70.34766 (cit. on p. 25).

[30] Markus Wilde, Stephen Kwok Choon, Alessio Grompone, and Marcello Romano. «Equations of Motion of Free-Floating Spacecraft-Manipulator Systems: An Engineer's Tutorial». In: *Frontiers in Robotics and AI* 5 (Apr. 2018). DOI: 10.3389/frobt.2018.00041 (cit. on p. 25).

[31] Atef ATA. «OPTIMAL TRAJECTORY PLANNING OF MANIPULATORS: A REVIEW». In: *Journal of Engineering Science and Technology* 2 (Apr. 2007) (cit. on p. 26).

[32] Karol Seweryn and Marek Banaszkiewicz. «Optimization of the Trajectory of a General Free-Flying Manipulator During the Rendezvous Maneuver». In: Aug. 2008. ISBN: 978-1-60086-999-0. DOI: 10.2514/6.2008-7273 (cit. on p. 27).

[33] Tomasz Rybus, Karol Seweryn, and Jerzy Z. Sasiadek. «Control System for Free-Floating Space Manipulator Based on Nonlinear Model Predictive Control (NMPC)». In: *Journal of Intelligent & Robotic Systems* 85 (2017), pp. 491–509 (cit. on p. 27).

[34] Om P Agrawal and Yangsheng Xu. «On the global optimum path planning for redundant space manipulators». In: *IEEE transactions on systems, man, and cybernetics* 24.9 (1994), pp. 1306–1316 (cit. on p. 27).

[35] Panfeng Huang, Yangsheng Xu, and Bin Liang. «Minimum-torque path planning of space robots using genetic algorithms». In: *International Journal of Robotics and Automation* 21.3 (2006), pp. 229–236 (cit. on p. 27).

[36] Serkan Aydin and Hakan Temeltas. «Fuzzy-differential evolution algorithm for planning time-optimal trajectories of a unicycle mobile robot on a predefined path». In: *Advanced Robotics* 18.7 (2004), pp. 725–748 (cit. on p. 27).

[37] Mingming Wang, Jianjun Luo, Jing Fang, and Jianping Yuan. «Optimal trajectory planning of free-floating space manipulator using differential evolution algorithm». In: *Advances in Space Research* 61.6 (2018), pp. 1525–1536 (cit. on p. 27).

[38] Mingming Wang, Jianjun Luo, and Ulrich Walter. «Trajectory planning of free-floating space robot using Particle Swarm Optimization (PSO)». In: *Acta Astronautica* 112 (2015), pp. 77–88 (cit. on pp. 27, 29, 31).

[39] Mingming Wang, Jianjun Luo, Jianping Yuan, and Ulrich Walter. «Coordinated trajectory planning of dual-arm space robot using constrained particle swarm optimization». In: *Acta Astronautica* 146 (2018), pp. 259–272 (cit. on p. 27).

[40] E Guiffo Kaigom, Thomas Josef Jung, and Jürgen Roßmann. «Optimal motion planning of a space robot with base disturbance minimization». In: *11th Symposium on Advanced Space Technologies in Robotics and Automation*. 2011, pp. 1–6 (cit. on p. 27).

[41] Tongtong Hu, Jianxia Zhang, and Qiang Zhang. «Trajectory planning to optimize base disturbance of 7-DOF free-floating space manipulator based on QPSO». In: *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*. Springer. 2015, pp. 281–293 (cit. on p. 27).

[42] Kazuya Yoshida, Kenichi Hashizume, and Satoko Abiko. «Zero reaction maneuver: flight validation with ETS-VII space robot and extension to kinematically redundant arm». In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 1. IEEE. 2001, pp. 441–446 (cit. on p. 27).

[43] Dimitar Dimitrov and Kazuya Yoshida. «Utilization of holonomic distribution control for reactionless path planning». In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3387–3392 (cit. on p. 27).

[44] Aditya Gattupalli, Suril V Shah, K Madhava Krishna, and AK Misra. «Control strategies for reactionless capture of an orbiting object using a satellite mounted robot». In: *Proceedings of conference on advances in robotics*. 2013, pp. 1–6 (cit. on p. 27).

[45] K. E. Parsopoulos and M. N. Vrahatis. «Particle Swarm Optimization Method in Multiobjective Problems». In: *Proceedings of the 2002 ACM Symposium on Applied Computing*. SAC '02. Madrid, Spain: Association for Computing Machinery, 2002, pp. 603–607. ISBN: 1581134452. DOI: 10.1145/508791.508907. URL: https://doi.org/10.1145/508791.508907 (cit. on p. 30).

[46] CA Coello Coello and Maximino Salazar Lechuga. «MOPSO: A proposal for multiple objective particle swarm optimization». In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 2. IEEE. 2002, pp. 1051–1056 (cit. on p. 30).

[47] Manfred Gilli and Peter Winker. «A review of heuristic optimization methods in econometrics». In: *Swiss Finance Institute Research Paper* 08-12 (2008) (cit. on p. 31).

[48] J. Kennedy and R. Eberhart. «Particle swarm optimization». In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968 (cit. on p. 31).

[49] Abhishek Sharma, Abhinav Sharma, Jitendra Kumar Pandey, and Mangey Ram. *Swarm Intelligence: Foundation, Principles, and Engineering Applications*. CRC Press, 2022 (cit. on p. 34).

[50] Maurice Clerc and James Kennedy. «The particle swarm-explosion, stability, and convergence in a multidimensional complex space». In: *IEEE transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73 (cit. on p. 32).

[51] Dongshu Wang, Dapei Tan, and Lei Liu. «Particle swarm optimization algorithm: an overview». In: *Soft Computing* 22.2 (2018), pp. 387–408 (cit. on p. 32).

[52] Lily D. Li, Xinghuo Yu, Xiaodong Li, and William Guo. «A Modified PSO Algorithm for Constrained Multi-objective Optimization». In: *2009 Third International Conference on Network and System Security*. 2009, pp. 462–467. DOI: 10.1109/NSS.2009.72 (cit. on p. 34).

[53] Jasbir Arora. *Introduction to optimum design*. Elsevier, 2004 (cit. on p. 35).

[54] Yaguang Yang. *Spacecraft Modeling, Attitude Determination, and Control Quaternion-based Approach: Quaternion-Based Approach*. CRC Press, 2019 (cit. on pp. 39, 40).

[55] Sadegh Jalili, Behrooz Rezaie, and Zahra Rahmani. «A novel hybrid model predictive control design with application to a quadrotor helicopter». In: *Optimal Control Applications and Methods* 39.4 (2018), pp. 1301–1322 (cit. on p. 40).

[56] Qirong Tang, Zhugang Chu, Yu Qiang, Shun Wu, and Zheng Zhou. «Trajectory Tracking of Robotic Manipulators with Constraints Based on Model Predictive Control». In: June 2020, pp. 23–28. DOI: 10.1109/UR49135.2020.9144943 (cit. on p. 44).

[57] Yuri Shtessel, Christopher Edwards, Leonid Fridman, Arie Levant, et al. *Sliding mode control and observation*. Vol. 10. Springer, 2014 (cit. on p. 46).

[58] Bahareh Nakisa, Mohd Zakree Ahmad Nazri, Mohammad Naim Rastgoo, and Salwani Abdullah. «A survey: Particle swarm optimization based algorithms to solve premature convergence problem». In: *Journal of Computer Science* 10.9 (2014), pp. 1758–1765 (cit. on pp. 52, 55).