POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering

Master Thesis

Control System Design with Reinforcement Learning Algorithm for a Space Manipulator

Advisor
Prof.ssa Elisa Capello

Candidate
Luca Di Ianni, s258082

April 2021

Dedicated to my grandmother Angela and my uncle Donato.

Abstract

On-Orbit Servicing (OOS) represents the advent of a new approach to space access and promises to be a key element in developing the future space infrastructures. Upcoming robotic spacecraft, mounting a robotic arm, may be able to perform a wider range of operations on a larger number of client spacecraft such as docking, berthing, refueling, repairing, upgrading, transporting, rescuing and orbital debris removal. Space manipulator systems, however, introduce relevant challenges due to the dynamic coupling between the manipulator and the spacecraft, that represents its base, and due to the growing need for autonomy and flexibility to perform new tasks and adapt to environment changes and disturbances. This implies the need for control system design that can reduce the reaction forces exchanged at the mounting point and that is robust to uncertainties on initial conditions.

The present Master thesis focuses on the modeling and control of a space manipulator system operating in a typical OOS mission environment. In order to obtain end-effector pose expressed as a function of the joint variables of the mechanical structure, the kinematic model of the robotic arm is first derived with a systematic and general approach using Denavit-Hartenberg (DH) convention. In addition, dynamic equations of the manipulator are obtained using Lagrange formulation to simulate the motion, compute the forces exchanged with the base and torques required for the execution of the task and thus design control algorithms.

Thus, a Reinforcement Learning (RL) controller for the manipulator is implemented exploiting the ability to learn how to complete a task within an unknown environment through repeated trial-and-error interactions with the environment without human involvement. Soft-Actor-Critic (SAC) algorithm, based on the maximum entropy framework, is selected to train the agent. The performance is then compared with a classical Proportional-Integral-Derivative (PID) controller.

Then a control system design for the spacecraft is proposed. Linear-Quadratic-Regulator (LQR) design, based on a fully controllable quaternion spacecraft model, is implemented for attitude purpose. LQR approach is also adopted for position control, showing pose-keeping ability and robustness of LQR design.

Last, a typical mission scenario of an OOS mission is presented, describing several phases that characterize this mission and showing the performance of the designed model.

The simulation scenario, the space manipulator system plant and the proposed controllers are developed using MATLAB R2020b and Simulink exploiting the numerous available toolboxes including Symbolic Toolbox, Deep Learning Toolbox, Reinforcement Learning Toolbox, Robotic System Toolbox and Simulink 3D Animation.

Acknowledgement

I would like to thank Prof.ssa Elisa Capello. Despite the difficulties and the need to work remotely, given the exceptional period, she was always kind and helpful in supporting me during the thesis work.

Thanks to Prof. Enrico Cestino and the Team S55 for making me grow and allowing me to be part of a fantastic working group that I will continue to follow.

Thanks to my father Pasqualino, my mother Patrizia, my sister Roberta, my brother Paolo, my grandfather Roberto and all my relatives who supported my choices every day and shared my dreams.

Thanks to Chiara who accompanied and supported me every day, during this beautiful journey and which I hope to hug again soon.

Thanks to all my colleagues and friends, in particular Pietro and Elena for the beautiful shared experiences, to Michele who has been several times an inspiration to me and to Davide, companion of a thousand adventures.

Finally, thanks to all the people I met along my way, even for just a minute, because they helped make me who I am.

Contents

List of Figures

List of Tables

Abbreviations

ADCS	Attitude Determination and Control System
AI	Artificial Intelligence
ARE	Algebraic Riccati Equation
CIMON	Crew Interactive MObile companioN
CoM	Center of Mass
ConOps	Concept of Operations
DARPA	Defense Advanced Research Project Agency
DoF	Degree of Freedom
EoM	Equation of Motion
ERA	European Robotic Arm
EVA	Extra-Vehicular Activity
GNC	Guidance and Navigation Control
HST	Hubble Space Telescope
IOC	Initial Operation Capabilities
ISS	International Space Station
JEMRMS	Japanese Experiment Module Remote Manipulator System
ML	Machine Learning
LEO	Low Earth Orbit
LQR	Linear Quadratic Regulator
LVLH	Local Vertical/Local Horizontal
OE	Orbital Express
OEDMS	Orbital Express Demonstration Manipulator System
OOS	On-Orbit Servicing
ORU	On-Orbit Replaceable Unit
PID	Proportional-Integral-Derivative
PWPF	Pulse-Width/Pulse-Frequency
RL	Reinforcement Learning
RPO	Rendezvous and Proximity Operations
RSO	Resident Space Object
RW	Reaction Wheels
SAC	Soft-Actor-Critic
SRMS	Shuttle Remote Manipulator System
SSRMS	Space Station Remote Manipulator System

Introduction

The goal of the present work is to analyze the dynamical behavior of a free-flying system operating in a typical OOS mission environment and exploit modern machine learning control method for control system design. In particular, more attention is paid to the possibility of developing a manipulator controller that can reduce the reaction forces and moments acting on the floating base, during the motion of the robotic arm. It also deals with the problem of maintaining attitude and position of the servicer during OOS operations.

On Orbit Servicing

On-Orbit-Servicing (OOS) is part of a future major disruption in the space landscape that will soon revolutionize space transportation and the way we use space with more focus on advanced robotics, vision-based navigation and rendezvous and proximity techniques.

Nowadays, with some exceptions, satellites are launched with everything they need for their entire mission, from IOC to end of life. OOS lets satellites to evolve over their lifetime and provides some benefits ([1]):

- risk mitigation of mission failure;
- mission cost reduction;
- mission performance increase in terms of mission life and payload utility;
- mission flexibility improvement offering option to modify space system requirements;
- new missions options.

The two main characters of the OOS are a servicer and a client. The former is a space vehicle designed to execute the task, while the latter is a Resident Space Object (RSO) receiving the OOS that can be cooperative or non-cooperative. A cooperative client is designed to provide important information, such as position, velocity, status, etc., to aid in acquisition, tracking, rendezvous or servicing operations. A non-cooperative client is not capable of communicate aiding information to the servicer making OOS operations more difficult to execute.

In the near future, servicer may be able to perform autonomously a wider range of operations ([2]):

- **Non-contact support** consists in observation, inspection for characterization of the spacecraft payload and assessment of the client for damage or remote enhancing the client with new capabilities (wireless connection).

Figure 1: Astronauts servicing HST during Service Mission 3A (STS-103)

- **Orbit modification and maintenance** occurs when a servicer performs propulsion and attitude control functions for the client to support, for example, constellation reconfiguration, rescue of satellites.
- **Refueling and commodities replenishment** is a service that supplies commodities essential for the client's mission. Commodities include fluids such as propellants, pressurants or coolants but also needed hardware.
- **Upgrade** is the replacement or addition of components, such as processors or payload, to a client to improves spacecraft performance.
- **Repair** is the replacement of components or correction of mechanical failures on a client to restore capability.
- **Assembly** is an activity in which two or more objects intentionally combine to create space system that could not be transported by existing launch vehicles or enhance existing space vehicles.
- **Debris mitigation** refers to a set of activities that allow controlled reentry of low Earth orbit (LEO) spacecraft, vehicles incapable of moving themselves, rocket bodies or orbital debris.

However, limited OOS activities [3] have been performed since Gemini and Apollo missions demonstrating Rendezvous and Proximity Operations (RPO). The activities on the first space stations (Skylab, Salyut and Mir) demonstrated the capability of installing and deploying solar arrays and sunshades, repairing critical components (science instruments, propellant system leak), removing unnecessary components (part of antenna) and maintaining science experiments.

The International Space Station (ISS) was assembled on orbit and designed to be resupplied with propellant and supplies by various vehicles (Space Shuttle, Soyuz, Cygnus, Dragon) and upgraded with new modules. During the Solar Maximum Repair Mission the astronauts, operating from the Space Shuttle, repaired the attitude control system of the Solar Maximum spacecraft, then it reached back the operative orbit. Hubble Space Telescope (HST), designed and built for servicing, is an eminent example of OOS. In fact, it has a modular design that allows astronauts to replace units, upgrade them with new technology or install additional hardware. HST was serviced five times over its 30 years of operation.

These activities were all performed by humans or with significant human-in-the loop presence. The first successful end-to-end robotic satellite servicing was performed by Defense Advanced Research Project Agency (DARPA). The Orbital Express (OE) project successfully demonstrated autonomous docking, refueling and unit replacing removing humans from the equation. The Orbital Express Demonstration Manipulator System (OEDMS), mounted on the servicer (ASTRO), was used to service the client satellite (NextSat) and in [4] it is shown how OEDMS played a critical part in achieving two key goals of the flight test: autonomous capture of the free-flying client and autonomous On-Orbit Replaceable Unit (ORU) transfer.

Space Manipulator Systems

A robotic manipulator consists of a sequence of rigid bodies interconnected by means of links that ensure its motion; it is characterized by an arm that ensures mobility, a wrist that confers dexterity and an end-effector that performs the task required of the robot. In the most general case of a task consisting of arbitrarily positioning and orienting an object in three-dimensional space, six DoFs are required, three for positioning a point on the object and three for orienting the object with respect to a reference coordinate frame. If more DoFs than task variables are available, the manipulator is said to be redundant from a kinematic viewpoint. Robotic systems have been used in many space missions, since the first deployment of the Shuttle Remote Manipulator System (SRMS). Over the life of the Space Shuttle program the SRMS evolved into an indispensable tool ([5]) not only for satellite deployment and retrieval, but was used extensively for satellite rescue and repair, as an EVA platform, International Space Station (ISS) construction, shuttle and payload bay surveys, as well as a host of contingency operations. Its success was a major driver in developing the robotic systems (SSRMS, ERA, JEMRMS) on board the ISS today ([6]) and it will serve as the foundation for all future space robotic systems. Actually, robotic manipulators are well suited to execute highly repetitive tasks that would be too time consuming, risky and expensive if performed by astronauts, but they represent a relevant challenge because of the dynamic coupling between the manipulator and the floating base. In fact, the motion of the manipulator generates reaction forces and moments on the spacecraft (and vice-versa) at the mounting point and this affects position and attitude of the spacecraft itself; this is true above all for small satellites.

This leads to two types of dynamic situation ([7]):

- free-floating systems, in which the spacecraft is not controlled during the motion of the manipulator; during OE demonstration for example, while the robotic arm was in motion, the servicer operated in free drift with a free-floating base than it corrected its attitude once arm motion was complete.

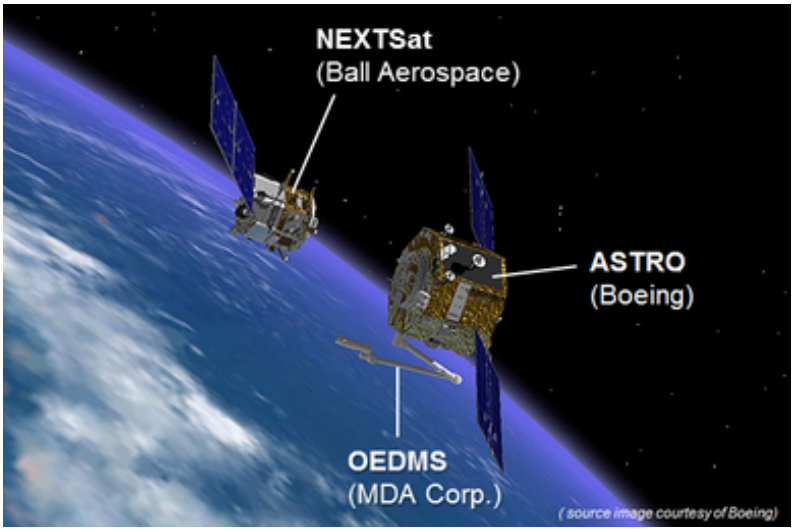


Figure 2: Orbital Express major subsystem contractors

- free-flying systems in which spacecraft pose is controlled during the arm motion; in this way the end-effector is able to reach the desired position and execute the desired task in a safe mode.

Design of a controller for a manipulator is a complex task and the user's requirements represent the starting point of the conceptual design ([14]). It is followed by the robot modeling which consists in the derivation of kinematics that can be formulated exploiting Denavit-Hartenberg (DH) parameters and dynamics that can be derived using the Lagrange formulation ([13]). Classic Proportional-Integral-Derivative controllers at each joint of manipulator are widely employed in industrial robots ([16]). Others control method, such as Computed Torque methods, Transpose Jacobian control are implemented and the experimental results are compared with the PID performance. In [8], the dynamic coupling between a robotic manipulator and the floating base platform is analyzed and a control strategy based on LQR and PID control laws is developed in the effort of keeping the platform attitude stable while the robotic arm is in service. [9] proposed a quaternion model that uses only the vector component of the quaternion. The author proved that the linearised reduced quaternion model using the vector component is fully controllable. [10] showed that the LQR controller design is a robust pole assignment design and demonstrated robustness and global stability of the design.

Modern robots should autonomously and flexibly adapt to new tasks, environment changes and disturbances. These requirements generated novel challenges in the area of robot control because it is no longer sufficient to implement control algorithms that are robust to noise. They should adapt to different working conditions overcoming the need of complex parameters identification and/or system re-modeling.

Artificial Intelligence for Space Application

Artificial Intelligence (AI) could drastically improve the future of space missions from greater planetary navigation and optimised mission operations, to analysing astronaut biometric data

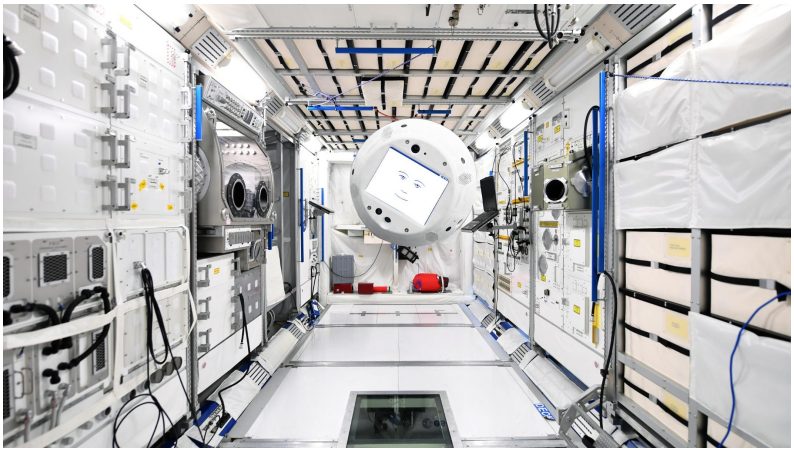


Figure 3: Robot assistant CIMON on ISS

and enhancing knowledge discovery. Intelligence can be thought of as the ability of machines (agents) to perceive and recognise their environment, to be able to learn from past experiences, and logically make decisions based on new scenarios. Greater autonomy is a new enabler for increased mission complexity and chance of success when human intervention is difficult or limited given the lack of visibility, the required amount of information and communication time lag.

The history of AI has far later beginnings dating back to 1998 with the use of an AI algorithm called Remote Agent, used onboard Deep Space 1–a comet probe. Remote Agent, whilst elementary in comparison to the AI capabilities of today, proved its worth with capabilities including the planning and scheduling of activities and diagnosing onboard failures. Since then there have been many other applications of AI for space exploration missions ranging from algorithms that give greater autonomy to planetary surface rovers, AI for discovering new Exoplanets, and AI powered assistants aboard the International Space Station. CIMON (Crew Interactive MObile companioN) is an example how RoboAssistants can help astronauts during on board activities. It is an IBM technology which aims to augment the information and learning available to astronauts and to reduce their exposure to stress. Other applications include mobile photography and videography and the ability to document experiments, search for objects and maintain an inventory. CIMON can also see, hear and understand what it observes and is equipped with an autonomous navigation system, allowing astronauts to issue voice commands.

The fastest growing branch of AI is Machine Learning (ML) whereby AI models learn by themselves, in essence by training a relatively simple algorithm to become increasingly complex. ML models process information in a similar way to humans by developing artificial neural networks. The system progressively improves its performance on a specific task by "learning" from its environment, without being explicitly programmed.

RL is prominently used as a control approach in robotics; both RL and optimal control address the problem of finding an optimal policy that optimizes an objective function and both rely on the notion of a system being described by an underlying set of states, controls and a plant or model that describes transitions between states ([23]). Deep learning-based approach can be promising in path planning for single arm manipulator ([20]) and also for multi-arm manipulators ([21]). [22] developed and implemented two RL-based compensation schemes to improve the suboptimal tracking performance of a feedback controller in a multi DoFs robot arm.

Overview of the thesis

The thesis contents are organized in 5 parts, 12 chapters and 3 appendices.

In Part I, mathematical model of the manipulator is presented. In Chap. 1, the kinematic model is derived with a systematic and general approach using Denavit-Hartenberg convention, direct and inverse kinematic are formulated considering open chain structures and two examples are presented. In Chap. 2, the dynamic equations are obtained considering the systematic approach based on Lagrange formulation and two example are presented.

In Part II, the mathematical model of the multy-body system is described. In Chap. 3, features of a typical servicer are listed, the reaction forces and moments acting on the floating base and the variations of the inertia matrix and the position of CoM of the whole system are computed. Then, reference frames are defined and the orientation of a rigid body in space, using Euler angles and quaternions representations, is presented. Thus, the general equations of the attitude dynamics and the equations of relative motion in the target reference frame are addressed.

In Part III, various control methods adopted for control system design are illustrated. Firstly Proportional Integral Derivative control is described, then Linear Quadratic Regulator control is treated and linearised attitude dynamics and relative position dynamics in state form are derived. Lastly, an introduction to Reinforcement Learning, its main features and its application are addressed.

In Part IV, simulation results are discussed. Chap. 5 is devoted to the presentation of the Matlab/Simulink environment and to the definition of the URDF file, necessary for the robotic model to be imported in the environment. In Chap. 6, design of the manipulator controllers are illustrated and their performance compared. Chap. 7 deals with the design and the performance of the position and attitude controllers, exploiting the LQR method, of the servicer.

In Part V, a typical mission scenario of an On Orbit Servicing mission is proposed. Chap. 8, starting from the Concept of Operations, describes the several phases that characterize this kind of missions and presents the performance of the designed model.

Part I

Manipulator Mathematical Model

Chapter 1

Kinematic Model

A manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies (links) connected by means of revolute or prismatic joints. The whole structure forms a kinematic chain: one end of the chain is constrained to a base, while an end-effector, useful to accomplish the mission objective, is mounted to the other end. Thus it is necessary to describe the end-effector position and orientation. This chapter deals with the derivation of the kinematic model, exploiting Denavit-Hartenberg convention; this allows to obtain the end-effector pose expressed as a function of the joint variables of the mechanical structure with respect to a reference frame. The chapter than introduces the inverse kinematics problem, which consists of the determination of the joint variables corresponding to a given end-effector pose and the differential kinematics is presented which gives the relationship between the joint velocities and the corresponding end-effector linear and angular velocity. At the end of the chapter two examples are presented exploiting MATLAB Symbolic Toolbox.

1.1 Reference Frames

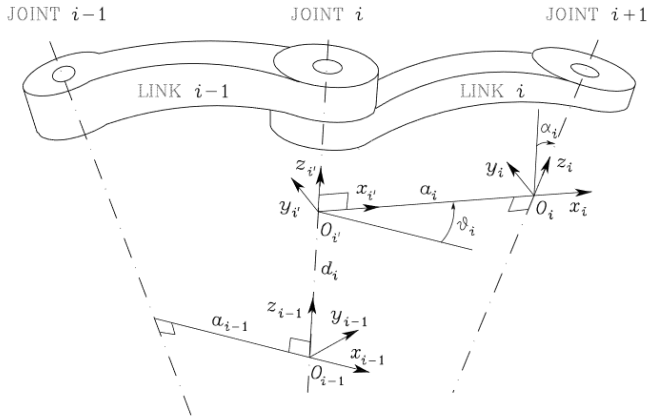


Figure 1.1: Denavit-Hartenberg kinematic parameters - Image from *Robotics. Modelling, Planning and Control*. Springer, 2010.

Consider an open-chain manipulator constituted by n+ 1 links connected by n joints, where Link 0 is conventionally fixed to the ground. It is assumed that each joint provides the mechanical structure with a single DoF, corresponding to the joint variable. In order to derive the direct kinematics equation for an open-chain manipulator, it is necessary to determine two frames attached to the two links and compute the coordinate transformations between them. It is possible to define the relative position and orientation of two consecutive links of the manipulator using an arbitrary frame attached to each link, but it's helpful to use a convention for selecting frame in robot applications. The Denavit-Hartenberg (DH) convention ([13]) let cut down the needed parameters to characterize a transformation matrix from six to four.

With reference to Fig. 1.1 let's define the Frame i:

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

Now it's possible to specify position and orientation of Frame i with respect to the Frame $i-1$ by the use of the four quantities a_i, d_i, α_i, θ_i. The four parameters are generally given the name link length, link offset, link twist and joint angle and defined as follow:

- a_i distance between O_i and $O_{i'}$;
- d_i coordinate of $O_{i'}$ along z_{i-1};
- α_i angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise;
- θ_i angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

Three of the above four parameters are constant for a given link, while the fourth parameter, θ_i for a revolute joint and d_i for a prismatic joint, is the joint variable.

At this point is possible to express the coordinate transformation between the Frame i and the Frame $i-1$ as

$$
\begin{aligned}
A_i^{i-1} &= R_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} R_{x,\alpha_i} \\
&= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1.1}
$$

1.2 Forward Kinematics

The mechanical structure of a manipulator is characterized by a number of DoFs, typically associated with joint articulations, which determine its posture, that denotes the pose of all rigid bodies composing the chain. The aim of forward kinematics is to compute the pose of the end-effector as a function of the joint variables. For simplicity, in this work the end-effector is considered as a part of the last link of the manipulator and it coincides with the origin of the last joint-fixed reference frame.

Since each joint connects two consecutive links, it reasonable to obtain the overall description of manipulator kinematics in a recursive way. Considering $\mathcal{F}_0$ the base frame and $\mathcal{F}_n$ the last joint-fixed frame, the coordinate transformation describing the position and orientation of $\mathcal{F}_n$ with respect to $\mathcal{F}_0$ is given by

$$\mathbf{T}_n^0 = \mathbf{A}_1^0(q_1)\mathbf{A}_2^1(q_2)\dots\mathbf{A}_n^{n-1}(q_n) \tag{1.2}$$

where $\mathbf{A}_i^{i-1}(q_i)$ is the homogeneous transformation matrix from $\mathcal{F}_i$ to $\mathcal{F}_{i-1}$, as derived in (1.1), function of the joint variable q_i.

1.3 Inverse Kinematics

The inverse kinematics problem consists of the determination of the manipulator posture corresponding to a given end-effector position and orientation. It is more complex than the forward kinematics for the following reasons:

- The equation to solve are, in general, non linear.
- Multiple solutions may exist.
- Infinite solutions may exist (redundant manipulator ([13])).
- There might be no admissible solutions, due to the manipulator structure.

Computation of closed-form solution requires algebraic or geometric intuition to find out significant equation and points respectively. On the other hand it might be useful to exploit numerical solution techniques that can be applicable to any kinematic structure, even if in general they do not allow the computation of all admissible solutions.

1.4 Differential Kinematics

The goal of the differential kinematics is to find the relationship between the joint velocities and the end-effector linear and angular velocities. In other words, it is desired to express the end-effector linear and angular velocity as a function of the joint velocities. The relations are both linear in the joint velocities

$$\dot{\mathbf{p}}_e = \mathbf{J}_v(\mathbf{q})\dot{\mathbf{q}} \tag{1.3}$$

$$\omega_e = \mathbf{J}_\omega(\mathbf{q})\dot{\mathbf{q}} \tag{1.4}$$

where $\mathbf{J}_v$ is the 3×n matrix relating the contribution of the joint velocities $\dot{\mathbf{q}}$ to the end-effector linear velocity $\dot{\mathbf{p}}_e$, while $\mathbf{J}_\omega$ is the 3×n matrix relating the contribution of the joint velocities $\dot{\mathbf{q}}$ to the end-effector angular velocity ω_e. In compact form, (1.3), (1.4) can be written as

$$\mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \omega_e \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{1.5}$$

which represents the manipulator differential kinematics equation. The 6×n matrix $\mathbf{J}$ is the manipulator geometric Jacobian

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_\omega \end{bmatrix} \tag{1.6}$$

which in general is a function of the joint variables. In order to compute the Jacobian, it is convenient to proceed separately for the linear velocity and the angular velocity.

For the contribution to the linear velocity, the time derivative of $\mathbf{p}_e(\mathbf{q})$ can be written as

$$\dot{\mathbf{p}}_e = \sum_{i=1}^{n} \frac{\partial \mathbf{p}_e}{\partial q_i}\dot{q}_i = \sum_{i=1}^{n} \mathbf{J}_{vi}\dot{q}_i \tag{1.7}$$

By distinguishing the case of a prismatic joint ($q_i = d_i$)from the case of a revolute joint ($q_i = \theta_i$), it is:

- for a prismatic joint
$$\mathbf{J}_{vi} = \mathbf{z}_{i-1}$$
- for a revolute joint
$$\mathbf{J}_{vi} = \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1})$$

For the contribution to the angular velocity, it is

$$\omega_e = \sum_{i=1}^{n} \omega_{i-1,i} = \sum_{i=1}^{n} \mathbf{J}_{\omega i}\dot{q}_i \tag{1.8}$$

in detail:

- for a prismatic joint
$$\mathbf{J}_{\omega i} = 0$$
- for a revolute joint
$$\mathbf{J}_{\omega i} = z_{i-1}$$

In summary, the Jacobian can be partitioned as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{v1} & & \mathbf{J}_{vn} \\ & \cdots & \\ \mathbf{J}_{\omega 1} & & \mathbf{J}_{\omega n} \end{bmatrix} \tag{1.9}$$

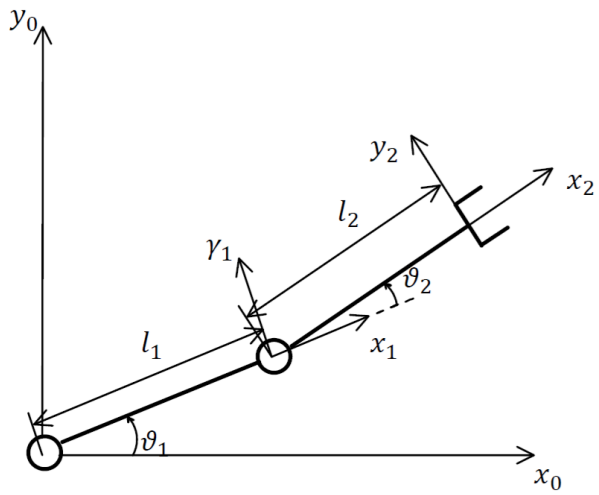


Figure 1.2: Two-link planar manipulator reference frames

1.5 Case Study 1 - Two-link planar arm

A two-link planar arm (Fig.1.2) is selected. Let l_1,l_2 be the lengths of the links and θ_1 and θ_2 be the joint variables. The joint axes z_0 e z_1 are normal to the page. The origin is chosen at the point of intersection of the z_0 axis with the page and the direction of the x_0 axis is completely arbitrary. Once the base frame is established, the frame $O_1x_1y_1z_1$ is fixed as shown by the DH convention, where the origin O_1 has been located at the intersection of z1 and the page. Final frame $O_2x_2y_2z_2$ is fixed by choosing the origin O_2 at the end of link 2 as shown.

Link parameters are shown in Table 1.1.

Link	a_i	α_i	d_i	θ_i
1	l_1	0	0	θ_1^*
2	l_2	0	0	θ_2^*

Table 1.1: Denavit-Hartenberg parameters for two-link planar arm

The **A**-matrices are derived from (1.1), with $c_1 = \cos\theta_1$, $c_2 = \cos\theta_2$, $s_1 = \sin\theta_1$, $s_2 = \sin\theta_2$, $c_{12} = \cos(\theta_1 + \theta_2)$ and $s_{12} = \sin(\theta_1 + \theta_2)$, as

$$\mathbf{A}_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & l_1c_1 \\ s_1 & c_1 & 0 & l_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

$$\mathbf{A}_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & l_2c_2 \\ s_2 & c_2 & 0 & l_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.11}$$

and the transformation matrix as

$$\mathbf{T}_2^0 = \mathbf{A}_1^0\mathbf{A}_2^1 = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1c_1 + l_2c_{12} \\ s_{12} & c_{12} & 0 & l_1s_1 + l_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.12}$$

The first two elements of the last column of $\mathbf{T}_2^0$ are the x and y components of the origin O_2 in the base frame. In fact,

$$\begin{aligned} x_2^0 &= l_1c_1 + l_2c_{12} \\ y_2^0 &= l_1s_1 + l_2s_{12} \end{aligned} \tag{1.13}$$

are the coordinates of the end-effector in the base frame. Rotational part of $\mathbf{T}_2^0$ gives the orientation ($\mathbf{R}_2^0$) of the frame $O_2x_2y_2z_2$ relative to the base frame.

With the chosen coordinate frames, computation of the Jacobians (as previously described), referred to CoMs of each link (c_1, c_2) and to the end-effector (ee), yields

$$\mathbf{J}_{c_1} = \begin{bmatrix} -\frac{1}{2}l_1s_1 & 0 \\ \frac{1}{2}l_1c_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{1.14}$$

$$\mathbf{J}_{c_2} = \begin{bmatrix} -(l_1s_1 + \frac{1}{2}l_2s_{12}) & -\frac{1}{2}l_2s_{12} \\ l_1c_1 + \frac{1}{2}l_2c_{12} & \frac{1}{2}l_2c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{1.15}$$

$$\mathbf{J}_{ee} = \begin{bmatrix} -(l_1s_1 + l_2s_{12}) & -l_2s_{12} \\ l_1c_1 + l_2c_{12} & l_2c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{1.16}$$

1.6 Case Study 2 - Three-link planar arm

Consider the three-link planar arm in Fig. 1.3 where the reference frames are illustrated. Let l_1,l_2, l_3 be the lengths of the links. Since the revolute axes are all parallel it is convenient to choice the x_i directions along the direction of relative link; in this way all the parameters d_i are null and the angles between the axes x_i directly provide the joint variables. Link parameters are shown in Table 1.2.

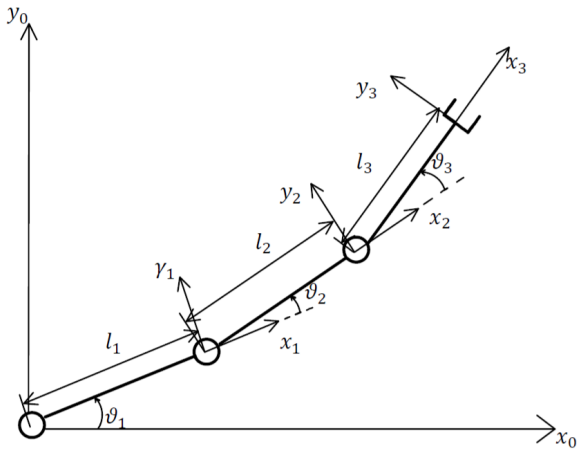


Figure 1.3: Three-link planar arm reference frames

Link	a_i	α_i	d_i	θ_i
1	l_1	0	0	θ_1^*
2	l_2	0	0	θ_2^*
2	l_3	0	0	θ_3^*

Table 1.2: Denavit-Hartenberg parameters for three-link planar arm

The coordinate transformations $\mathbf{A}_2^1$ and $\mathbf{A}_1^0$ are the same of the first case study (1.10), (1.11). The remaing coordinate transformation is derived from (1.1) and Tab. 1.2, with $c_3 = \cos\theta_3$ and $s_3 = \sin\theta_3$, as

$$\mathbf{A}_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & l_3c_3 \\ s_3 & c_3 & 0 & l_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the transformation matrix, $c_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$ and $s_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$, as

$$\mathbf{T}_3^0 = \mathbf{A}_1^0\mathbf{A}_2^1\mathbf{A}_3^2 = \begin{bmatrix} c_{123} & -s_{123} & 0 & l_1c_1 + l_2c_{12} + l_3c_{123} \\ s_{123} & c_{123} & 0 & l_1s_1 + l_2s_{12} + l_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

The first two elements of the last column of $\mathbf{T}_3^0$ are the x and y components of the origin O_3 in the base frame. In fact,

$$\begin{aligned} x_3^0 &= l_1c_1 + l_2c_{12} + l_3c_{123} \\ y_3^0 &= l_1s_1 + l_2s_{12} + l_3s_{123} \end{aligned} \tag{1.18}$$

are the coordinates of the end-effector in the base frame. Rotational part of $\mathbf{T}_3^0$ gives the orientation ($\mathbf{R}_3^0$) of the frame $O_3x_3y_3z_3$ relative to the base frame.

With the chosen coordinate frames, computation of the Jacobians, referred to CoMs of each link (c_1, c_2, c_3) and to the end-effector (ee), yields

$$\mathbf{J}_{c_1} = \begin{bmatrix} -\frac{1}{2}l_1 s_1 & 0 & 0 \\ \frac{1}{2}l_1 c_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{1.19}$$

$$\mathbf{J}_{c_2} = \begin{bmatrix} -(l_1 s_1 + \frac{1}{2}l_2 s_{12}) & -\frac{1}{2}l_2 s_{12} & 0 \\ l_1 c_1 + \frac{1}{2}l_2 c_{12} & \frac{1}{2}l_2 c_{12} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{1.20}$$

$$\mathbf{J}_{c_3} = \begin{bmatrix} -(l_1 s_1 + l_2 s_{12} + \frac{1}{2}l_3 s_{123}) & -(l_2 s_{12} + \frac{1}{2}l_3 s_{123}) & \frac{1}{2}l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + \frac{1}{2}l_3 c_{123} & l_2 c_{12} + \frac{1}{2}l_3 c_{123} & \frac{1}{2}l_3 c_{123} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.21}$$

$$\mathbf{J}_{ee} = \begin{bmatrix} -(l_1 s_1 + l_2 s_{12} + l_3 s_{123}) & -(l_2 s_{12} + l_3 s_{123}) & l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_2 c_{12} + l_3 c_{123} & l_3 c_{123} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.22}$$

Chapter 2

Dynamic Equations

In order to simulate the motion, compute the forces and torques required for the execution and thus design joints, transmissions, actuators and control algorithms, it is necessary to derive the dynamic model of the manipulator. This chapter explains the method for derivation of the EoMs of a manipulator based on *Lagrange formulation*. Then resulting EoMs, obtained with *Matlab Symbolic Toolbox*, for the two manipulators already introduced are presented.

2.1 Lagrange Formulation

The dynamic model of a manipulator provides a description of the relationship between the joint actuator torques and the motion of the structure.

Once the generalized coordinates ($q_i, i = 1 \dots n$), that describe the link positions of an n-DOF manipulator, are chosen, the Lagrangian of the mechanical system can be defined as a function of the generalized coordinates:

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{2.1}$$

where $\mathcal{T}$ and $\mathcal{U}$ are the kinetic energy and the potential energy of the system.

Lagrange equations are expressed by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \varepsilon_i \qquad i = 1, \dots, n \tag{2.2}$$

where ϵ_i is the generalized force associated with the generalized coordinate q_i.

Equations (2.2) can be written in compact form as

$$\frac{d}{dt}\Big(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\Big)^T - \Big(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}\Big)^T = \varepsilon \tag{2.3}$$

where, for a manipulator with an open kinematic chain, the generalized coordinates are gathered in the vector of joint variables q. The contributions to the generalized forces are given by the non conservative forces, i.e., the joint actuator torques, the joint friction torques, as well as the joint torques induced by end-effector forces at the contact with the environment. For our purpose only the joint actuator torques (τ_i) are considered, thus

$$\varepsilon = \sum_{i=1}^{n} \tau_i \tag{2.4}$$

The kinetic energy of the system is given by

$$\mathcal{T} = \sum_{i=1}^{n} \Big(\frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i + \frac{1}{2} \omega_i^T \mathbf{I}_i \omega_i \Big) \tag{2.5}$$

where linear velocity $\mathbf{v}_i$ and angular velocity ω_i can be expressed as a function of the generalized coordinates, as described in (1.5). Notice that the position of $Link_i$ depends on the manipulator configuration and thus the inertia tensor, when expressed in the base frame ($\mathbf{I}_i$), is configuration-dependent. It is easy to verify the following relation:

$$\mathbf{I}_i = \mathbf{R}_i^0 \mathbf{I}_{i,l} (\mathbf{R}_i^0)^T \tag{2.6}$$

where $\mathbf{R}_i^0$ is the rotation matrix from $Link_i$ frame to the base frame. Taking into account last considerations, the 2.5 becomes

$$\mathcal{T} = \sum_{i=1}^{n} \Big(\frac{1}{2} m_i \dot{\mathbf{q}}^T \mathbf{J}_v^T \mathbf{J}_v \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{J}_\omega^T \mathbf{R}_i^0 \mathbf{I}_{i,l} (\mathbf{R}_i^0)^T \mathbf{J}_\omega \dot{\mathbf{q}} \Big) \tag{2.7}$$

As done for kinetic energy, the potential energy stored in the manipulator is given by the sum of the contributions relative to each link. However, since the manipulator is designed to operate in an environment characterized by microgravity, it is worth to assume

$$\mathcal{U} = 0 \tag{2.8}$$

Having computed the total kinetic and assumed null the potential energy of the system as in (2.7), (2.8) and taking the derivatives required by Lagrange equations in (2.3), the EoMs for the manipulator can be written as

$$\mathbf{M}(\mathbf{q}, \dot{\mathbf{q}}) \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \varepsilon \tag{2.9}$$

2.2 Case Study 1 - Two-link planar arm

Consider the two-link planar arm in Fig. 2.1, for which the vector of generalized coordinates is $q = [\theta_1, \theta_2]^T$. Let l_1, l_2 be the lengths of the two links. Also let m_1, m_2 be the masses of the two links and I_1, I_2 the diagonal inertia matrices in the CoM-centered frame of the two links, respectively:

$$I_i = \begin{bmatrix} I_{ixx} & 0 & 0 \\ 0 & I_{iyy} & 0 \\ 0 & 0 & I_{izz} \end{bmatrix} \tag{2.10}$$

EoMs are obtained in the form

$$EoM_1 = M_{11}(q)\ddot{q}_1 + M_{12}(q)\ddot{q}_2 + C_1(q, \dot{q}) - \tau_1 \tag{2.11}$$

$$EoM_2 = M_{21}(q)\ddot{q}_1 + M_{22}(q)\ddot{q}_2 + C_2(q, \dot{q}) - \tau_2 \tag{2.12}$$

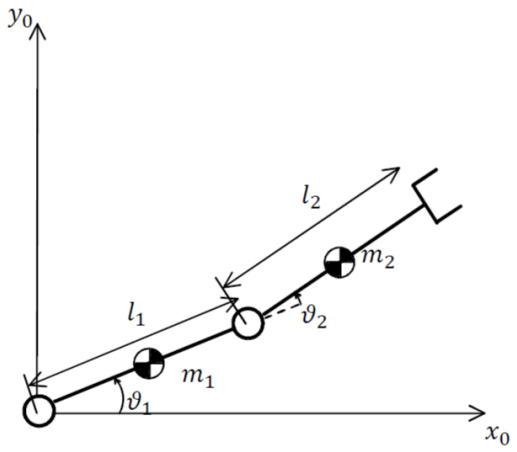


Figure 2.1: Two-link planar arm

where

$$
\begin{aligned}
M_{11} =& I_{1zz} + I_{2zz} + (l_1^2 m_1)/4 + l_1^2 m_2 + (l_2^2 m_2)/4 + l_1 l_2 m_2 \cos(q_2) \\
M_{12} =& I_{2zz} + (l_2^2 m_2)/2 + l_1 l_2 m_2 \cos(q_2) \\
M_{21} =& I_{2zz} + (l_2^2 m_2)/2 + l_1 l_2 m_2 \cos(q_2) \\
M_{22} =& I_{2zz} + l_2^2 m_2 \\
C_1 =& - l_1 l_2 m_2 \sin(q_2) \dot{q}_2 (\dot{q}_1 + \dot{q}_2) \\
C_2 =& (-l_1 l_2 m_2 \sin(q_2) \dot{q}_1^2)/2
\end{aligned}
$$

2.3 Case Study 2 - Three-link planar arm

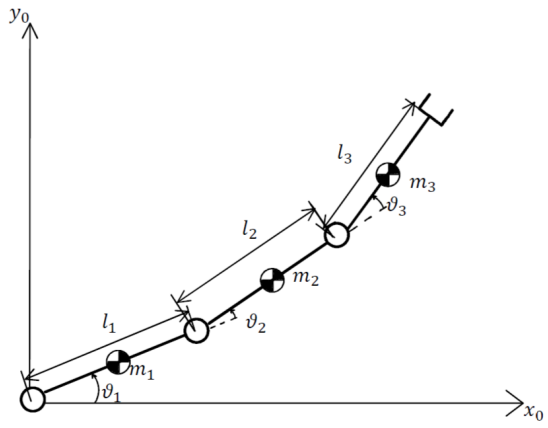


Figure 2.2: Three-link planar arm

Consider the three-link planar arm in Fig. 2.2, for which the vector of generalized coordinates is $q = [\theta_1, \theta_2, \theta_3]^T$. As for the Case Study 1, let l_1, l_2, l_3 be the lengths of the three links. Also let

m_1, m_2, m_3 be the masses of the three links and I_1, I_2, I_3 the diagonal inertia matrices relative to the CoMs of the three links, respectively (as defined in 2.10), EoMs are obtained in the form

$$EOM_1 = M_{11}(q)\ddot{q}_1 + M_{12}(q)\ddot{q}_2 + M_{13}(q)\ddot{q}_3 + C_1(q,\dot{q}) - \tau_1 \tag{2.13}$$
$$EOM_2 = M_{21}(q)\ddot{q}_1 + M_{22}(q)\ddot{q}_2 + M_{23}(q)\ddot{q}_3 + C_2(q,\dot{q}) - \tau_2 \tag{2.14}$$
$$EOM_3 = M_{31}(q)\ddot{q}_1 + M_{32}(q)\ddot{q}_2 + M_{33}(q)\ddot{q}_3 + C_3(q,\dot{q}) - \tau_3 \tag{2.15}$$

where

$$\begin{aligned}
M_{11} =& I_{1zz} + I_{2zz} + I_{3zz} + (l_1^2 m_1)/4 + l_1^2 m_2 + l_1^2 m_3 + (l_2^2 m_2)/4 + l_2^2 m_3 + (l_3^2 m_3)/4 \\
&+ l_1 l_3 m_3 \cos(q_2+q_3) + l_1 l_2 m_2 \cos(q_2) + 2 l_1 l_2 m_3 \cos(q_2) + l_2 l_3 m_3 \cos(q_3) \\
M_{12} =& I_{2zz} + I_{3zz} + (l_2^2 m_2)/4 + l_2^2 m_3 + (l_3^2 m_3)/4 + (l_1 l_3 m_3 \cos(q_2+q_3))/2 \\
&+ (l_1 l_2 m_2 \cos(q_2))/2 + l_1 l_2 m_3 \cos(q_2) + l_2 l_3 m_3 \cos(q_3) \\
M_{13} =& I_{3zz} + (l_3^2 m_3)/4 + (l_1 l_3 m_3 \cos(q_2+q_3))/2 + (l_2 l_3 m_3 \cos(q_3))/2 \\
M_{21} =& I_{2zz} + I_{3zz} + (l_2^2 m_2)/4 + l_2^2 m_3 + (l_3^2 m_3)/4 + (l_1 l_3 m_3 \cos(q_2+q_3))/2 \\
&+ (l_1 l_2 m_2 \cos(q_2))/2 + l_1 l_2 m_3 \cos(q_2) + l_2 l_3 m_3 \cos(q_3) \\
M_{22} =& I_{2zz} + I_{3zz} + (l_2^2 m_2)/4 + l_2^2 m_3 + (l_3^2 m_3)/4 + l_2 l_3 m_3 \cos(q_3) \\
M_{23} =& I_{3zz} + (m_3 l_3^2)/4 + (l_2 m_3 \cos(q_3) l_3)/2 \\
M_{31} =& I_{3zz} + (l_3^2 m_3)/4 + (l_1 l_3 m_3 \cos(q_2+q_3))/2 + (l_2 l_3 m_3 \cos(q_3))/2 \\
M_{32} =& I_{3zz} + (m_3 l_3^2)/4 + (l_2 m_3 \cos(q_3) l_3)/2 \\
M_{33} =& I_{3zz} + (m_3 l_3^2)/4 \\
C_1 =& -(l_1 l_3 m_3 \dot{q}_2^2 \sin(q_2+q_3))/2 - (l_1 l_3 m_3 \dot{q}_3^2 \sin(q_2+q_3))/2 - (l_1 l_2 m_2 \dot{q}_2^2 \sin(q_2))/2 \\
&- l_1 l_2 m_3 \dot{q}_2^2 \sin(q_2) - (l_2 l_3 m_3 \dot{q}_3^2 \sin(q_3))/2 - l_1 l_3 m_3 \dot{q}_1 \dot{q}_2 \sin(q_2+q_3) \\
&- l_1 l_3 m_3 \dot{q}_1 \dot{q}_3 \sin(q_2+q_3) - l_1 l_3 m_3 \dot{q}_2 \dot{q}_3 \sin(q_2+q_3) - l_1 l_2 m_2 \dot{q}_1 \dot{q}_2 \sin(q_2) \\
&- 2 l_1 l_2 m_3 \dot{q}_1 \dot{q}_2 \sin(q_2) - l_2 l_3 m_3 \dot{q}_1 \dot{q}_3 \sin(q_3) - l_2 l_3 m_3 \dot{q}_2 \dot{q}_3 \sin(q_3) \\
C_2 =& (l_1 l_3 m_3 \dot{q}_1^2 \sin(q_2+q_3))/2 + (l_1 l_2 m_2 \dot{q}_1^2 \sin(q_2))/2 + l_1 l_2 m_3 * \dot{q}_1^2 \sin(q_2) \\
&- (l_2 l_3 m_3 \dot{q}_3^2 \sin(q_3))/2 - l_2 l_3 m_3 \dot{q}_1 \dot{q}_3 \sin(q_3) - l_2 l_3 m_3 \dot{q}_2 \dot{q}_3 \sin(q_3) \\
C_3 =& (l_3 m_3 (l_1 \dot{q}_1^2 \sin(q_2+q_3) + l_2 \dot{q}_1^2 \sin(q_3) + l_2 \dot{q}_2^2 \sin(q_3) + 2 l_2 \dot{q}_1 \dot{q}_2 \sin(q_3)))/2
\end{aligned}$$

Part II

Multi-Body Mathematical Model

Chapter 3

Multi-Body Model Overview

The robotic arm, modeled in previous chapter, is mounted on the satellite, constituting the servicer. As already introduced, the motion of the manipulator generates reaction forces and moments on the spacecraft (and vice-versa) at the mounting point and this affects position and attitude of the spacecraft itself. It is worth to assume that the base-satellite is controlled (as it will be shown in Part IV) and thus the reaction forces and moments affecting the manipulator dynamics, due the motion of the satellite, are neglected. In this chapter geometrical features of the servicer are described, reaction forces and moments acting on the floating base and variations of the inertia matrix of the whole system are computed. It also deals with the definition of the reference frames adopted and the ways to represent the orientation of a rigid body in space using Euler angles and quaternions representations. Lastly, some details required for the derivation of the general equations of the relative motion for circular orbit and of the attitude dynamics are provided.

3.1 Geometrical and inertial features

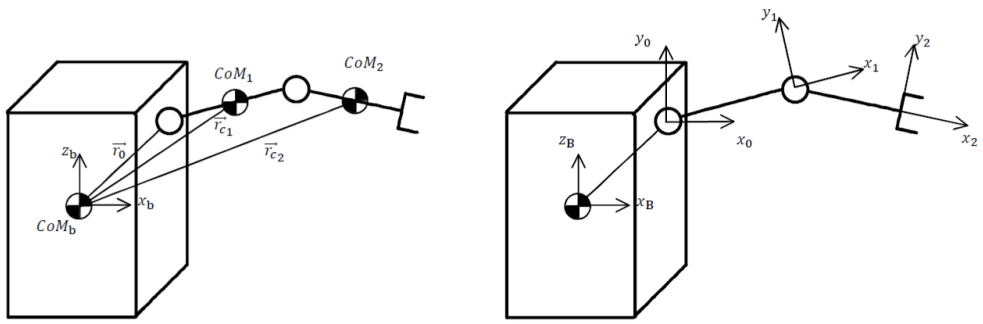


Figure 3.1: Multi-body reference frames

The configuration of the servicer consists of a base and a robotic arm characterized by the number of links and the type of joints. For the sake of simplicity the base is represented by a parallelogram with the CoM_B fixed in the geometrical center of the prismatic shape where it is

located the body reference frame($\mathcal{F}_B$) and the joints of the manipulator are all revolute ones. The first step is to define the reference frames to obtain the effects of the motion of the robotic arm with respect to the CoM_B. The joint-fixed reference frames related to the manipulator have been already introduced in Chap. 1, using the DH convention. The frame $\mathcal{F}_0$, called base frame, is located at the mounting point where the forces exchange between the two bodies take place and its position in $\mathcal{F}_B$ can be expressed as

$$\mathbf{r}_0 = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \tag{3.1}$$

The links are considered as homogeneous ones, thus the CoM_i of each link is located in the middle of it and the link-fixed reference frames centered in the CoM_i are oriented the same way the previous joint-fixed is. Thus, it is possible to define the inertia matrix of the whole system, in $\mathcal{F}_B$ as

$$\mathbf{I} = \mathbf{I}_0 + \sum_{i=1}^{n} \mathbf{I}_i \tag{3.2}$$

where $\mathbf{I}_0$ and $\mathbf{I}_i$ are the inertia matrices of the base satellite and of the i-th link in $\mathcal{F}_B$, respectively. The latter can be derived as

$$\mathbf{I}_i = (\mathbf{R}_i^B)^T \mathbf{J}_i \mathbf{R}_i^B - m_i \mathbf{r}_{ci}^{\times} \mathbf{r}_{ci}^{\times} \tag{3.3}$$

where m_i is the mass of the i-th link, $\mathbf{R}_i^B$ is the rotation matrix from the i-th joint-fixed frame to $\mathcal{F}_B$, $\mathbf{J}_i$ is the inertia matrix of i-th link in the local link-fixed frame and $\mathbf{r}_{ci}$ is the position of the CoM_i in $\mathcal{F}_B$.

3.2 Reaction forces and moments computation

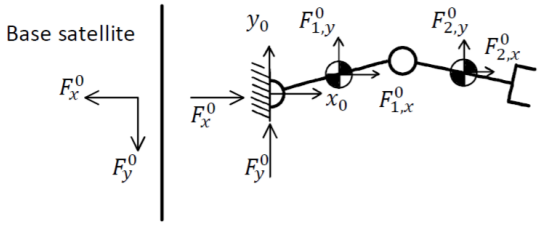


Figure 3.2: Reaction forces scheme

Actually, the mounting point behaves as an hinge support that allows rotation about any axis (z as suggested by DH convention) but prevents movement in the horizontal and vertical directions. Considering a planar arm, as seen in the previous chapters, mounted along the vertical symmetry axis of the base, this yields to the generation of reaction forces $\mathbf{F}^0 = [F_x^0, F_y^0, 0]^T$ in base frame $\mathcal{F}_0$, as shown in Fig. 3.2:

$$F_x^0 = \sum_{i=1}^{n} F_{i,x}^0 \qquad F_z^0 = \sum_{i=1}^{n} F_{i,z}^0 \tag{3.4}$$

where $F^0_{i,x}$ and $F^0_{i,z}$ are the components of the forces in $\mathcal{F}_0$ each CoM is subjected to. Since $\mathbf{F} = m\mathbf{a}$, it is necessary to multiply the mass of each joint by its acceleration, which can be obtained by deriving the speed; the latter can be obtained as already shown in (1.5). Then, reaction forces with respect to $\mathcal{F}_0$ are obtained by adding the contributions of each link.

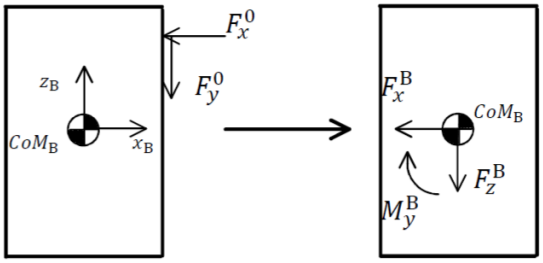


Figure 3.3: Reaction forces translation

These forces correspond to the disturbances affecting the base satellite. Since the mounting point is distant $\mathbf{r}_0$ from the CoM_B, it is necessary to translate and rotate the reaction forces with reference to $\mathcal{F}_B$:

$$\mathbf{F}_B = \mathbf{R}_0^B \mathbf{F}_0 \tag{3.5}$$

where $\mathbf{R}_0^B$ is the rotation matrix from $\mathcal{F}_0$ to $\mathcal{F}_B$. Thus, a reaction moment is expected as depicted in Fig. 3.3

$$\mathbf{M}_B = \mathbf{r}_0 \times \mathbf{F}_B \tag{3.6}$$

3.3 Reference Frames

The purpose of this section is to describe the coordinate frame used in this thesis for the description of the attitude and the relative motion. Each frame $\mathcal{F}_i$ is defined by its origin O_i and a set of three orthogonal vectors a_1, a_2, a_3.

3.3.1 LVLH Frame $\mathcal{F}_{LVLH}$

Local Vertical Local Horizontal coordinate frame is used to describe motions w.r.t. the moving position and direction towards the center of the Earth of an orbiting body.

Referring to the Fig. 3.4, we have:

- the origin coincides with the CoM of the spacecraft;
- axis a_1 is in the direction of the orbital velocity vector and it is called V_{bar};
- axis a_2 is in the opposite direction of the angular momentum vector of the orbit and it is called H_{bar};
- axis a_3 is radial from the spacecraft CoM to the center of the Earth and it is called R_{bar}

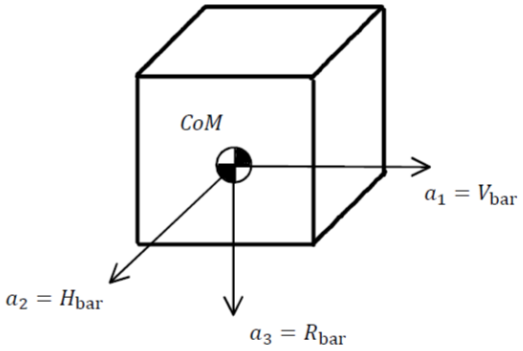


Figure 3.4: Spacecraft LVLH frame representation

3.3.2 Spacecraft body Frame $\mathcal{F}_B$

The spacecraft attitude frame is used to describe all rotations of a body in space and it is fixed to the nominal CoM of the satellite. The axis a_1 points in the direction of the orbital velocity vector, while axis a_2 is often aligned with the positive or negative direction of the angular momentum vector of the orbit. Referring to the Fig. 3.5, the features of the body frame are:

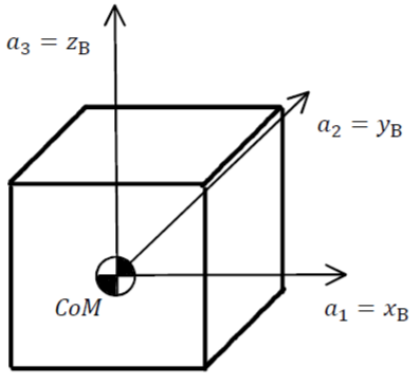


Figure 3.5: Spacecraft body frame representation

- O_i coincides with the CoM of the spacecraft;
- a_1, a_2, a_3 directions depend on the mission purposes and $a_3 = a_1 \times a_2$ forming a right handed system.

The coordinate transformation $\mathbf{L}^B_{LVLH}$ from the LVLH frame ($\mathcal{F}_{LVLH}$) to the spacecraft body frame ($\mathcal{F}_B$) is obtained by a rotation of the frame by the attitude angles α_z (azimuth), α_y

(elevation) and α_x (roll):

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha_x & s\alpha_x \\ 0 & -s\alpha_x & c\alpha_x \end{bmatrix} \begin{bmatrix} -s\alpha_y & 0 & c\alpha_y \\ 0 & 1 & 0 \\ c\alpha_y & 0 & s\alpha_y \end{bmatrix} \begin{bmatrix} c\alpha_z & s\alpha_z & 0 \\ -s\alpha_z & c\alpha_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{LVLH} \\ y_{LVLH} \\ z_{LVLH} \end{bmatrix} \tag{3.7}$$

3.4 Euler's Angles

It is possible to use rotation matrix to describe the attitude of the spacecraft through the unit vectors $\hat{e}_i$ of the body frame attached to it. This approach gives a redundant description of frame orientation; in fact, it is characterized by nine elements which are not independent but related by six constraints due to the orthogonality conditions. This implies that three parameters are sufficient to describe orientation of a rigid body in space.

One of the set of three parameters most widely used to describe the attitude of a rigid body w.r.t. a fixed frame are the Euler's angles, a sequence of three rotation (ϕ, θ, ψ). The original sequence of rotations proposed by Euler is the sequences 3-1-3:

- Rotate the reference frame by the angle ψ (precession angle) about axis z;
- Rotate the current frame by the angle θ (nutation angle) about axis x';
- Rotate the current frame by the angle ϕ (spin angle) about axis z".

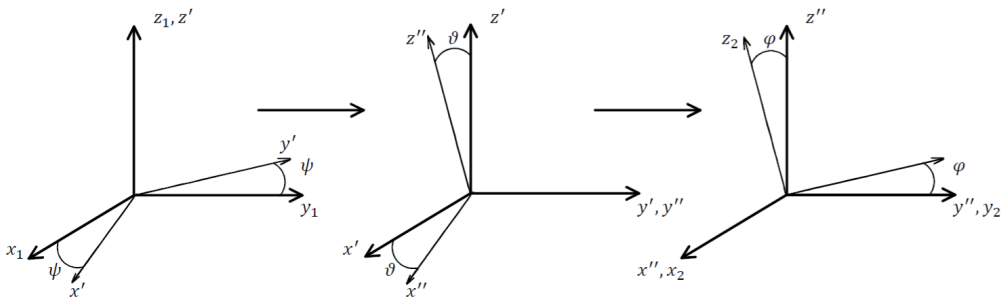


Figure 3.6: Bryant's rotation sequence

Many other sequences are available and equally useful. In atmospheric flight mechanics the most widely used sequence of rotations is the3–2–1, also known as the Bryant's angles:

- Rotate the reference frame by the angle ψ (yaw angle) about axis z;
- Rotate the current frame by the angle θ (pitch angle) about axis y';
- Rotate the current frame by the angle ϕ (roll angle) about axis x".

This set of angles is used also in space flight dynamics, to describe the attitude of a spacecraft with respect to the LVLH Frame.

There is a problem with the representation of rotations in a three dimensional space, that is the singularity of all the descriptions in terms of three parameters. This means that there will always be positions of the two frames that can be described in different ways, once a particular sequence of rotations is chosen.

3.5 Quaternions

Unit quaternions (Euler parameters) are mostly used as attitude parameterizations to overcome the singularity problems. Euler's theorem states that the most general motion of a rigid body with one point fixed is a rotation about an axis through that point ([15]). The elements of unit quaternions can be expressed in terms of the principal eigenvector $\hat{e} = \{e_1, e_2, e_3\}^T$ and the rotation angle α as follows:

$$\begin{aligned} q_0 &= \cos(\alpha/2) \\ q_1 &= e_1 \sin(\alpha/2) \\ q_2 &= e_2 \sin(\alpha/2) \\ q_3 &= e_3 \sin(\alpha/2) \end{aligned} \tag{3.8}$$

where q_0 is called the scalar part of the quaternion while $\mathbf{q} = \{q_1, q_2, q_3\}^T$ is called the vector part of the quaternion. They are constrained by the condition

$$q_0^2 + q_1^2 + q_2^2 + q_3^3 = 1 \tag{3.9}$$

hence, the name unit quaternion. The advantage of using quaternions is that the expression for the rotation matrix from a fixed frame $\mathcal{F}_I$ onto any arbitrary frame $\mathcal{F}_B$ is purely algebraic (it contains no trigonometric functions):

$$\mathbf{L}_{BI} = (q_0^2 - \mathbf{q} \cdot \mathbf{q})I + 2\mathbf{q}\mathbf{q}^T - 2q_0\mathbf{q}^\times \tag{3.10}$$

where the $^\times$ indicates the cross product matrix equivalent

$$\mathbf{q}^\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{3.11}$$

The quaternion error (q_{err}), that is the magnitude of the angular displacement between the current attitude (q_{true}) and the desired one (q_{des}), must be computed by using the quaternion product:

$$q_{err} = q_{des}^{-1} \otimes q_{true} \tag{3.12}$$

where the quaternion product can be expressed as

$$p = q \otimes r = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \cdot \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} \tag{3.13}$$

The evolution of the quaternions is described by the set of linear differential equations, represented in matrix form as

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \tag{3.14}$$

and the equivalent form is given by

$$\begin{aligned} \dot{q}_0 &= \frac{1}{2}\omega \cdot \mathbf{q} \\ \dot{\mathbf{q}} &= \frac{1}{2}(q_0\omega - \omega \times \mathbf{q}) \end{aligned} \tag{3.15}$$

3.6 Relative Motion Dynamics

The objective is to represent the chaser motion with respect to the target local orbital frame ($\mathcal{F}_{LVLH}$), which has its origin in the CoM of the target spacecraft.

The general assumption for the derivation of the equations of motion is that the motion of a body subject to the effects of a central spherical gravity field and to forces from thruster actuation or disturbances ([15]). The relative position between chaser and target is denoted by

$$\mathbf{s} = \mathbf{r}_c - \mathbf{r}_t \tag{3.16}$$

where $\mathbf{r}_c$ and $\mathbf{r}_t$ are position vectors in inertial space of chaser and target, considered as point masses, respectively.

The general equation for motion under the influence of a central force is Newton's law of gravitation:

$$F_g(r) = -G\frac{Mm}{r^2}\frac{r}{r} = -\mu\frac{m}{r^3}r \tag{3.17}$$

where F_g is the gravitational force, G is the universal gravitational constant, M is the mass of the central body (Earth), m is the second mass (spacecraft), r is the radius vector and μ is the product GM. From (3.17) it's possible to define the target motion

$$\begin{aligned} F_g(r_t) = m_t\ddot{r}_t &= -\mu\frac{m_t}{r_t^3}r_t \\ f_g(r_t) = \ddot{r}_t &= -\mu\frac{r_t}{r_t^3} \end{aligned} \tag{3.18}$$

and the chaser motion

$$\begin{aligned} m_c\ddot{r}_c = F_g(r_c) + F &= -\mu\frac{m_c}{r_c^3}r_c + F \\ \ddot{r}_t = f_g(r_c) + \frac{F}{m_c} &= -\mu\frac{r_c}{r_c^3} + \frac{F}{m_c} \end{aligned} \tag{3.19}$$

The relative motion is defined from (3.16) as

$$\ddot{s} = \ddot{r_c} - \ddot{r}_t \tag{3.20}$$

thus, by substituting (3.18) and (3.19) in (3.20), one obtains

$$\ddot{s} = f_g(r_c) - f_g(r_t) + \frac{F}{m_c} \tag{3.21}$$

After the linearization of $f_g(r_c)$ around the vector r_t by means of a Taylor expansion to first order

$$f_g(r_c) = f_g(r_t) + \left.\frac{df_g(r)}{dr}\right|_{r=r_t}(r_c - r_t) \tag{3.22}$$

the (3.21) becomes

$$\ddot{s} = -\frac{\mu}{r_t^3} M s + \frac{F}{m_c} \tag{3.23}$$

where

$$M = \begin{bmatrix} 1 - 3\frac{r_x^2}{r_t^2} & 3\frac{r_x r_y}{r_t^2} & 3\frac{r_x r_z}{r_t^2} \\ 3\frac{r_y r_z}{r_t^2} & 1 - 3\frac{r_y^2}{r_t^2} & 3\frac{r_y r_z}{r_t^2} \\ 3\frac{r_z r_x}{r_t^2} & 3\frac{r_z r_y}{r_t^2} & 1 - 3\frac{r_z^2}{r_t^2} \end{bmatrix} \tag{3.24}$$

Now, starting from a general kinematic equation for translation and rotating systems and taking into account (3.23), it is possible to obtain the chaser acceleration in the rotating frame as

$$\frac{d^{*2}\mathbf{s}^*}{dt^2} + \omega \times (\omega \times \mathbf{s}^*) + 2\omega \times \frac{d^*\mathbf{s}^*}{dt} + \frac{d\omega}{dt} \times \mathbf{s}^* + \frac{\mu}{r_t^3}\mathbf{M}\mathbf{s}^* = \frac{\mathbf{F}}{m_c} \tag{3.25}$$

where the starred frame (*) is rotating with the orbital rate ω. By expressing $\mathbf{r}_t$ and ω in the target frame

$$\mathbf{r}_t = \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix} \quad \text{and} \quad \omega = \begin{bmatrix} 0 \\ -\omega \\ 0 \end{bmatrix}$$

the terms of (3.25) become

$$\omega \times \mathbf{s}^* = \begin{bmatrix} -\omega z \\ 0 \\ \omega x \end{bmatrix}$$

$$\omega \times (\omega \times \mathbf{s}^*) = \begin{bmatrix} -\omega^2 x \\ 0 \\ -\omega^2 z \end{bmatrix}$$

$$\omega \times \frac{d^*\mathbf{s}^*}{dt} = \begin{bmatrix} -\omega \dot{z} \\ 0 \\ \omega \dot{x} \end{bmatrix}$$

$$\frac{d\omega}{dt} \times \mathbf{s}^* = \begin{bmatrix} -\dot{\omega} z \\ 0 \\ \dot{\omega} x \end{bmatrix}$$

$$\mathbf{M}\mathbf{s}^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \quad \mathbf{s}^* = \begin{bmatrix} x \\ y \\ -2z \end{bmatrix}$$

For the special case of circular orbits the angular rate ω is constant and it can be expressed as

$$\omega^2 = \frac{\mu}{r_t^3} \tag{3.26}$$

By substituting (3.26) and the terms just obtained in (3.25), the general linear equations for the relative motion, known as Hill's equations, are derived as

$$\begin{aligned} \ddot{x} - 2\omega\dot{z} &= \frac{1}{m_c}F_x \\ \ddot{y} + \omega^2 y &= \frac{1}{m_c}F_y \\ \ddot{z} + 2\omega\dot{x} - 3\omega^2 z &= \frac{1}{m_c}F_z \end{aligned} \tag{3.27}$$

It is worth to resume the assumptions that bring to the validity of (3.27):

- Only circular orbits are analyzed.
- The distance between the chaser and the target is lower than the orbital radius.
- The position of the chaser is expressed in the local orbital frame ($\mathcal{F}_{LVLH}$) centered in the CoM of the target.
- The target moves under the influence of a central gravity field only.

3.7 Attitude Dynamics

The total angular momentum of a spacecraft with Reaction Wheels (RW) is:

$$\mathbf{H}_B = \mathbf{I}\omega_B + \mathbf{h}_{RW} \tag{3.28}$$

where $\mathbf{I}$ is the inertia matrix of the satellite, ω_B is the angular velocity of satellite in body frame and $\mathbf{h}_{RW}$ is the angular momentum of the RW, which can be expressed in body frame as

$$\mathbf{h}_{RW} = \mathbf{L}\mathbf{I}_{RW}\omega_{RW} \tag{3.29}$$

where $\mathbf{I}_{RW}$ is the $n \times n$ RW diagonal inertia matrix, $\mathbf{L}$ is the $3 \times n$ RW distribution matrix and ω_{RW} is the angular speed of the wheels with n the total number of RW.

The second fundamental law of rigid body dynamics states that the time derivative of the angular momentum is equal to the total external torque applied to the body:

$$\frac{d\mathbf{h}}{dt} = \mathbf{M}_B \tag{3.30}$$

where the total moment acting on the spacecraft includes the external disturbances ($\mathbf{T}_d$) and the moment due to thrusters ($\mathbf{M}_{thr}$):

$$\mathbf{M}_B = \mathbf{M}_{thr} + \mathbf{T}_d \tag{3.31}$$

Expressing the vector quantities in body axis component one gets

$$\dot{\mathbf{h}}_B + \omega_B \times \mathbf{h}_B = \mathbf{M}_B \tag{3.32}$$

Assuming that moments due to thrusters are negligible and the general case where the inertia matrix $\mathbf{I}$ is not constant over time, it is possible to derive the Euler equations from (3.32) as

$$\dot{\omega}_B = \mathbf{I}^{-1}(-\omega_B \times (\mathbf{I}\omega_B + \mathbf{L}\mathbf{I}_{RW}\omega_{RW}) - \omega_B\dot{\mathbf{I}} - \dot{\mathbf{h}}_{RW} + \mathbf{T_d}) \tag{3.33}$$

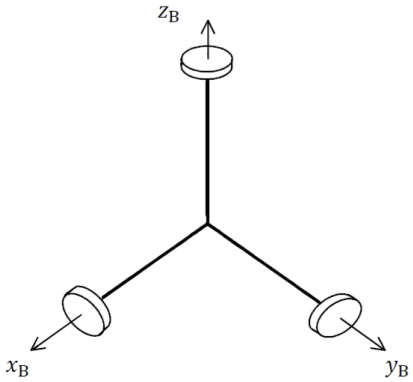


Figure 3.7: Standard orthogonal 3-wheel configuration

Since the RW work on the principle of momentum exchange, the angular momentum produced by RW is transferred to the satellite with opposite sign

$$\dot{\mathbf{h}}_{RW} = -\mathbf{T}_c \tag{3.34}$$

where $\mathbf{T}_c$ is the command torque, which is determined by the controllers. Moreover, $\mathbf{L}$ uniquely depends on RW configuration and consists of n columns corresponding to n RW. Each column vector represents the distribution of the RW torques on to the axes of rotation of the satellite. For the purposes of this work the standard orthogonal 3-wheel configuration (Fig. 3.7) is adopted, thus $\mathbf{L}$ is a 3×3 identity matrix:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

By substituting (3.34) and (3.35) into (3.33), the following attitude dynamics equation can be found

$$\dot{\omega}_B = \mathbf{I}^{-1}(-\omega_B \times (\mathbf{I}\omega_B + \mathbf{I}_{RW}\omega_{RW}) - \omega_B\dot{\mathbf{I}} + \mathbf{T}_c + \mathbf{T_d}) \tag{3.36}$$

Part III

Control System Design

Chapter 4

Introduction to implemented control methods

This chapter presents various control methods adpoted for control system design. Classical Proportional Integral Derivative control is described, then mathematical properties of Linear Quadratic Regulator control are proposed. Lastly, a brief overview of Reinforcement Learning theory and its implementation are addressed.

4.1 PID Control

The Proportional Integrative Derivative (PID) controller is a control loop mechanism based on a feedback architecture that continuously calculate the signal error $e(t)$ as the difference between the desired (s_d) and the current (s) state and applies a correction based on appropriate gains. This correction modifies the control action $u(t)$ that attempts to minimize $e(t)$ in order to restore the desired state.

Starting from a proportional controller, in the form

$$u(t) = K_P e(t) \tag{4.1}$$

it is useful to decrease the steady state error of the system but can never manage to eliminate it. In fact, applying a higher K_P decreases the rise time and after a certain value of reduction on the steady state error, this only leads to overshoot of the system response and causes oscillations.

To overcome partially the problems derived from the proportional controller, the integrative gain is added:

$$u(t) = K_I \int_0^t e(\tau) d\tau \tag{4.2}$$

It is mainly used to eliminate the steady state error resulting from the proportional controller, but it has a negative impact on the speed of response and overall stability of the system.

Finally, a derivative gain can be added:

$$u(t) = K_D \frac{de(t)}{dt} \tag{4.3}$$

Its contribution is to increase the stability of the system since it has the ability to predict the future error of the system response preventing sudden changes in the error signal.

	Rise Time	Overshoot	Settling Time	SS Error	Stability
$K_P \uparrow$	decrease	increase	small increase	decrease	degrade
$K_I \uparrow$	small decrease	increase	increase	large decrease	degrade
$K_D \uparrow$	small decrease	decrease	decrease	minor change	improve

Table 4.1: Effects of K_P, K_I and K_D on a closed-loop system

The sum of the three terms is a linear combination of the signal error, its integral and its derivative over time, leading to the PID controller formulation in the time domain:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{4.4}$$

where K_P, K_I and K_D, all non-negative, denote the coefficients for proportional, integrative and derivative terms respectively.

In order to obtain the desired control response it is necessary to tune the parameters to their optimum values. Manual tuning is achieved by arranging the parameters according to the system response taking into account the effects of each controller on a closed-loop system (Table 4.1). Although this method seems simple it requires a lot of time and experience.

Ziegler–Nichols method is one of the most effective methods that increase the usage of PID controllers. Firstly it is checked if desired proportional control gain is positive or negative. Then, K_I and K_D are set to zero and only K_P value is increased until it crates a periodic oscillation at the output response. This critical K_P value is attained to be the "ultimate gain" K_C and the period where the oscillation occurs is named "ultimate period" P_C. As a result, the whole process depends on two variables and the other control parameters are computed according to the Table 4.2.

Control type	K_P	K_I	K_D
P	$0.50K_C$	0	0
PI	$0.45K_C$	$1.2K_P\frac{dT}{P_C}$	0
PID	$0.60K_C$	$2K_P\frac{dT}{P_C}$	$K_P\frac{P_C}{8dT}$

Table 4.2: Ziegler–Nichols PID Controller tuning method

4.2 LQR Control

To apply the LQR technique the system equations are linearized about the equilibrium configuration. To implement this technique the linear equations are written in the state form as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.5}$$

For a continuous time system, the state-feedback law $\mathbf{u} = -\mathbf{K}\mathbf{x}$ minimizes the quadratic cost function

$$J(u) = \int_0^\infty \mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)d\tau \tag{4.6}$$

It can be shown that the optimal feedback law is given by

$$\mathbf{K} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{4.7}$$

where $\mathbf{P}$, positive definite matrix, is the solution of the Algebraic Riccati Equation (ARE)

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = -\mathbf{Q} \tag{4.8}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are symmetric, positive definite and semi-positive matrices respectively defined as state and control weighting matrices. A typical choice for $\mathbf{Q}$ and $\mathbf{R}$, introduced in [17], leads to

$$\mathbf{Q} = \begin{bmatrix} q_1 & & & \\ & q_2 & & \\ & & \ddots & \\ & & & q_n \end{bmatrix} \qquad \mathbf{R} = \rho \begin{bmatrix} r_1 & & & \\ & r_2 & & \\ & & \ddots & \\ & & & r_n \end{bmatrix} \tag{4.9}$$

with

$$q_i = \frac{1}{t_{si}(x_{imax})^2}, \qquad q_i = \frac{1}{(u_{imax})^2} \qquad \text{and} \qquad \rho > 0$$

where t_{si} is the desired settling time of x_i, x_{imax} is a constraint on $|x_i|$, u_{imax} is a constraint on actuation magnitude $|u_i|$ and ρ is chosen to trade off regulation versus control effort.

4.2.1 Linearized attitude dynamics

Nonlinear spacecraft kinematics equations of motion (3.36) can be replaced by a set of independent nonlinear spacecraft kinematics equations of motion that leads to a controllable linearized quaternion model.

Lemma 1 [9]: For any given $\mathbf{q}$, if $\alpha \neq \pi$, there exists a one-to-one mapping between ω and $\dot{\mathbf{q}}$. Moreover, let

$$f(q) = \sqrt{1 - q_1^2 - q_2^2 - q_3^2} \tag{4.10}$$

and

$$\mathbf{\Omega} = \begin{bmatrix} f(q) & -q_3 & q_2 \\ q_3 & f(q) & -q_1 \\ -q_2 & q_1 & f(q) \end{bmatrix} \tag{4.11}$$

then, the one-to-one mapping is given by

$$\omega = 2\mathbf{\Omega}^{-1}\dot{\mathbf{q}} \tag{4.12}$$

Therefore, (l'equazione della cinematica) can be replaced by

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} f(q) & -q_3 & q_2 \\ q_3 & f(q) & -q_1 \\ -q_2 & q_1 & f(q) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.13}$$

Therefore

$$\begin{bmatrix} \dot{\omega} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ \frac{1}{2}\mathbf{1}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \omega \\ \mathbf{q} \end{bmatrix} + \begin{bmatrix} \mathbf{J}^{-1} \\ \mathbf{0}_3 \end{bmatrix} \mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.14}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ \frac{1}{2}\mathbf{1}_3 & \mathbf{0}_3 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \omega \\ \mathbf{q} \end{bmatrix} \qquad \text{and} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{I}^{-1} \\ \mathbf{0}_3 \end{bmatrix} \tag{4.15}$$

where $\mathbf{1}_3$ is the 3×3 identity matrix and $\mathbf{I}^{-1}$ the 3×3 inverse inertia matrix. By considering the non constant inertia matrix and external disturbances affecting the spacecraft attitude, (4.14) becomes

$$\mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{B}_d\mathbf{T}_d \tag{4.16}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}^{-1}\dot{\mathbf{I}} & \mathbf{0}_3 \\ \frac{1}{2}\mathbf{1}_3 & \mathbf{0}_3 \end{bmatrix}, \qquad \mathbf{B}_d = \begin{bmatrix} \mathbf{I}^{-1} \\ \mathbf{0}_3 \end{bmatrix} \qquad \textit{and} \qquad \mathbf{T}_d = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.17}$$

where T_d is the vector of external disturbances in $\mathcal{F}_B$.

4.2.2 Relative position dynamics in state space form

For control purpose it is convenient to represent Hill's equations, introduced in (3.27), in state space form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_c} & 0 & 0 \\ 0 & \frac{1}{m_c} & 0 \\ 0 & 0 & \frac{1}{m_c} \end{bmatrix} \mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4.18}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_c} & 0 & 0 \\ 0 & \frac{1}{m_c} & 0 \\ 0 & 0 & \frac{1}{m_c} \end{bmatrix} \tag{4.19}$$

where ω is the angular rate for the case of circular orbits and m_c is the mass of the chaser vehicle (servicer). By considering external disturbances affecting the spacecraft position, (4.18) becomes

$$\mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{B}_d\mathbf{F}_d \tag{4.20}$$

with

$$\mathbf{B}_d = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_c} & 0 & 0 \\ 0 & \frac{1}{m_c} & 0 \\ 0 & 0 & \frac{1}{m_c} \end{bmatrix} \qquad \mathbf{F}_d = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{4.21}$$

where F_d is the vector of external disturbances in $\mathcal{F}_{LVLH}$.

4.3 RL Control

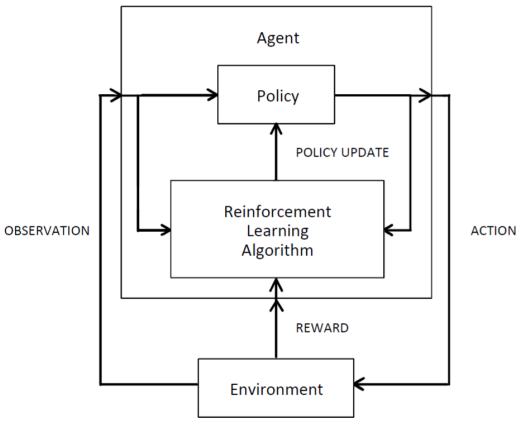


Figure 4.1: RL scheme

The goal of reinforcement learning is to train an agent to complete a task within an unknown environment. The agent receives observations and rewards from the environment and sends actions to the environment. The reward is a measure of how successful an action is with respect to completing the task goal. In other words, reinforcement learning involves an agent learning the optimal behavior through repeated trial-and-error interactions with the environment without human involvement.

The behavior of a reinforcement learning policy, that is how the policy observes the environment and generates actions to complete a task in an optimal manner, is similar to the operation of a controller in a control system. RL operates directly on measured data and rewards from interaction with the environment. It has placed great focus on addressing cases which are analytically intractable using approximations and data-driven techniques.

4.3.1 Artificial Neural Networks and agent definition

Neural networks, on which RL is based, consist of many simple processing nodes that are interconnected and based on how a human brain works. Though not as efficient, they perform in roughly similar ways. The brain learns from what it experiences, and so do these systems. An artificial neural network uses a collection of connected nodes called artificial neurons. A

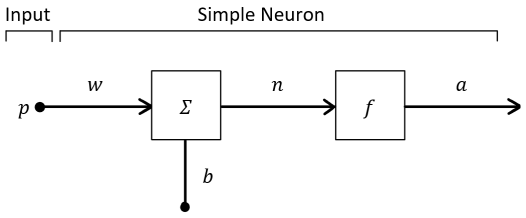


Figure 4.2: Single-input neuron scheme

single-input neuron is the fundamental building block for neural networks. There are three processes that take place in a neuron as in Fig. 4.2.

- Weight function, the scalar input p is multiplied by the scalar weight w to form the product wp, again a scalar.
- Net input function, the weighted input wp is added to the scalar bias b to form the net input n.
- Transfer function, which produces scalar output a from the net input.

The terms w and b are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior.

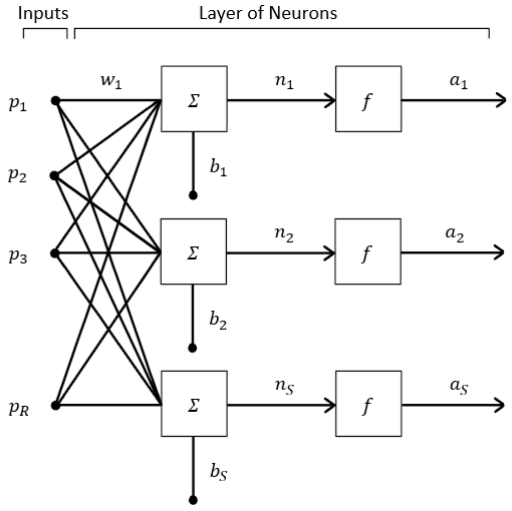


Figure 4.3: Single layer Neural Network scheme

Two or more neurons can be combined in a layer, and a particular network could contain one or more such layers. By considering one-layer network with R input elements and S neurons as in Fig. (4.2), each element of the input vector p is connected to each neuron input through the weight matrix W:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & \vdots & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix} \tag{4.22}$$

where the row indices indicate the destination neuron of the weight and the column indices indicate which source is the input for that weight. The i-th neuron has a summer that gathers its weighted inputs and bias to form its own scalar output $n(i)$. The various $n(i)$ taken together form an S-element net input vector n. Finally, the neuron layer outputs form a column vector a. It is common for the number of inputs to a layer to be different from the number of neurons: a layer is not constrained to have the number of its inputs equal to the number of its neurons.

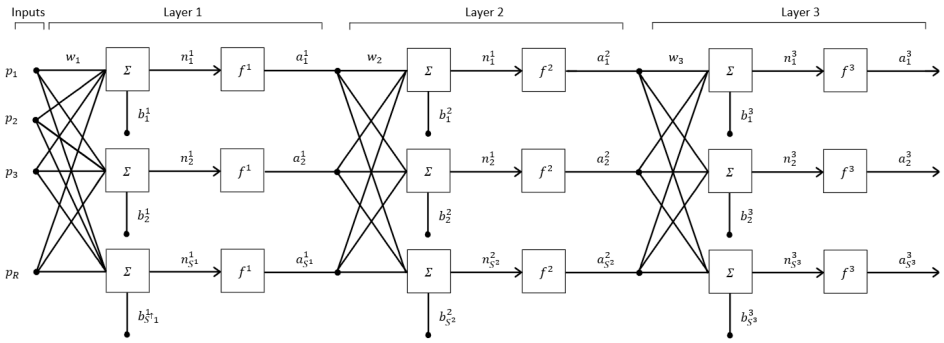


Figure 4.4: Neural Network scheme

As already said, a network can have several layers. Each layer has a weight matrix W, a bias vector b, and an output vector a. The network shown above has R_1 inputs, S_1 neurons in the first layer, S_2 neurons in the second layer, and so on and it is common for different layers to have different numbers of neurons. Layers of a multi-layer network play different roles:

- The output layer is a layer that produces the network output.
- The hidden layer is a layer between the input and the output layer.

An artificial neural network processes information in two ways; when it is being trained it is in "learning mode" and when it puts what it has learned into practice it is in "operating mode". For neural networks to learn, they must be told when they do something right or wrong. This feedback process is often called backpropagation and allows the network to modify its behavior so that the output is exactly as intended. In other words, it is trained with many learning examples and eventually learns how to reach the correct output every time, even when it is presented with a new range or set of inputs.

An agent, key element for RL, contains two components:

- a policy that selects actions based on the observations from the environment. Typically, the policy is a function approximator with tunable parameters, such as a deep neural network.
- a learning algorithm that continuously updates the policy parameters based on the actions, observations, and reward. The goal of the learning algorithm is to find an optimal policy that maximizes the cumulative reward received during the task.

Depending on the learning algorithm, an agent maintains one or more parameterized function approximators for training the policy. Approximators can be used in two ways:

- Critic, for a given observation and action, returns as output the expected value of the cumulative long-term reward for the task.
- Actor, for a given observation, returns as output the action that maximizes the expected cumulative long-term reward.

Agents that use both an actor and a critic are referred to as actor-critic agents. In these agents, during training, the actor learns the best action to take using feedback from the critic (instead

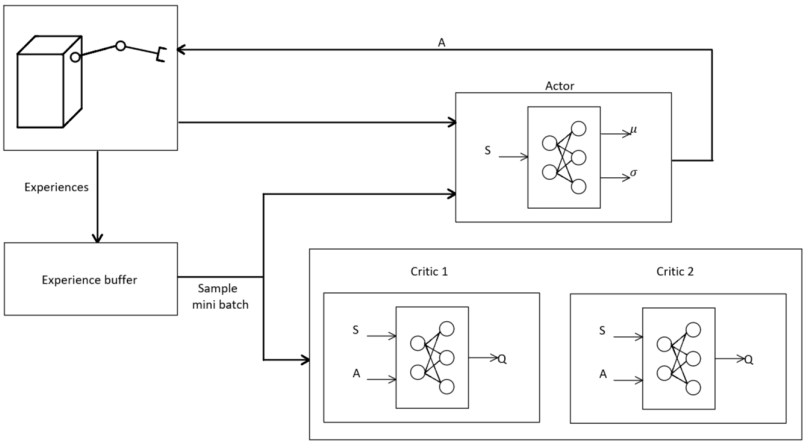


Figure 4.5: SAC agent scheme

of using the reward directly). At the same time, the critic learns the value function from the rewards so that it can properly criticize the actor. In general, these agents can handle both discrete and continuous action spaces.

For this reason Soft Actor Critic (SAC) agent is selected for the training. It is an actor-critic RL method with the following key features:

- model free, because the agent uses experience to learn the policy or value function directly without using a model of the environment.
- online, because the agent has the ability to interact with the environment and collect additional transitions using the behavior policy.
- off-policy, because the updated policy is different from the behavior policy and the algorithm can reuse data collected for another task (good at predicting movement in robotics).

The SAC algorithm computes an optimal policy that maximizes both the long-term expected reward and the entropy of the policy, that is a measure of policy uncertainty given the state. A higher entropy value promotes more exploration. A high level scheme of SAC algorithm is presented in Fig. 4.5. The experience buffer is used to store experiences from the agent environment interactions and the agent learns by using sample data from this buffer. As already said, the learning algorithm for this agent follows the actor-critic approach with one actor and two critics, each of which are represented by neural networks. The actor, starting from the observations deriving from the environment, selects the actions based on a mean and variance outputs form the network, that define the stochastic policy of the actor used to compute the agent's action. During training the weights of the actor network are updated by a trade-off between the expected reward and the entropy of the policy. The algorithm uses multiple critics to reduce overestimation of the Q-function ([11]). Critics criticize the actions whether they're good or bad and their weights are also updated periodically.

Training an agent using reinforcement learning is an iterative process and the general workflow for the training includes the following steps:

1. Formulate the problem, defining the tasks for the agent to learn, including how the agent interacts with the environment and any primary and secondary goals the agent must achieve.
2. Create the environment within which the agent operates, including the interface between agent and environment and the environment dynamic model.
3. Define reward signal that the agent uses to measure its performance against the task goals and how this signal is calculated from the environment.
4. Create the agent, which includes defining a policy representation and configuring the agent learning algorithm.
5. Train the agent policy representation using the defined environment, reward and agent learning algorithm.
6. Evaluate the performance of the trained agent by simulating the agent and environment together.
7. Deploy the trained policy representation.

Part IV

Simulation Results

Chapter 5

Simulation Environment

In order to validate the behavior of the controllers designed for the manipulator only and the whole system, a MATLAB/Simulink model is implemented. It considers two different dynamic systems, referred to the manipulator and the base satellite, which interact by the forces exchange between the two systems. In this chapter geometrical and physical features of the space manipulator system are presented, its MATLAB/Simulink model is described and the URDF file, necessary for the robotic model to be imported and animated in the simulation environment, is illustrated.

5.1 Space manipulator system configuration

Space manipulator system configuration, considered for the simulations, consists of a base satellite with a two-link planar arm mounted on the lateral face in the direction of the satellite motion, as shown in Fig. 5.1.

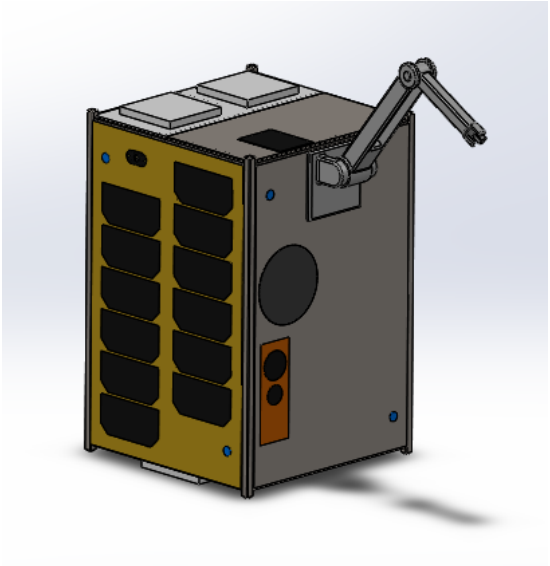


Figure 5.1: CAD of the space manipulator system

Here, geometrical and physical features of the system are listed:

- The base satellite is a 12U Cubesat (0.2×0.2×0.3m).
- The mass of the base satellite is 10Kg.
- The base satellite is assumed to be homogeneous, thus the inertia matrix is diagonal. Non-zero components of the inertia matrix are: I_{0xx} = 1.083e-1Kgm^2, I_{0yy} = 1.083e-1Kgm^2, I_{0zz} = 6.67e-2Kgm^2.
- The mounting point is located along the vertical symmetry axis of the base satellite ($\mathbf{r}_0 = [0.1, 0, 0.15]^T m$) and the manipulator motion is confined in the x-z plane.
- The links are assumed to be cylindrical and the dimensions are: 1st link length, radius and mass are 15cm, 2.5cm and 1.5Kg respectively; 2nd link length, radius and mass are 10cm, 1.5cm and 1.0Kg respectively.
- The links are assumed to be homogeneous, thus inertia matrices are diagonal. Non-zero components of inertia matrices are: I_{1xx} = 9e-4Kgm^2, I_{1yy} = 3.66e-2Kgm^2, I_{1zz} = 3.66e-2Kgm^2, I_{2xx} = 2e-4Kgm^2, I_{2yy} = 1.07e-2Kgm^2, I_{2zz} = 1.07e-2Kgm^2.

Starting from the derivation of the mathematical model (kinematics and dynamics) of the manipulator (Part. I), MATLAB/Simulink model is obtained and its general structure is shown in Fig. 5.2. Control strategies to allow the end-effector to reach the desired position and perform

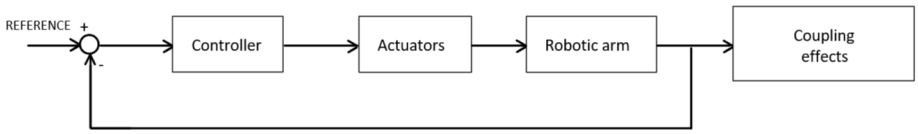


Figure 5.2: Robotic arm general scheme

the delicate task for which it was designed, are implemented in the "Controller" block. Different strategies adopted and the effects they have on the general scheme will be described in Chap. 10. In the "Actuator" block, physical limitations of the actuators (max torques with RL method) are taken into account, while in the "Robotic arm" block the manipulator dynamics is implemented and in Fig. 5.3 it is explained in more detail.

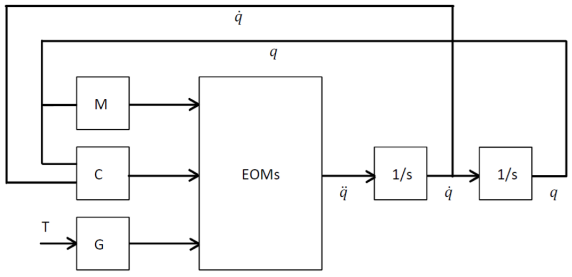


Figure 5.3: Robotic arm plant in Simulink

The "Coupling effects" block, instead, allows the computation of the reaction forces acting at the mounting point and the effects in terms of variation of the moment of inertia due to the motion of the robotic arm, as derived in Chap. 3.

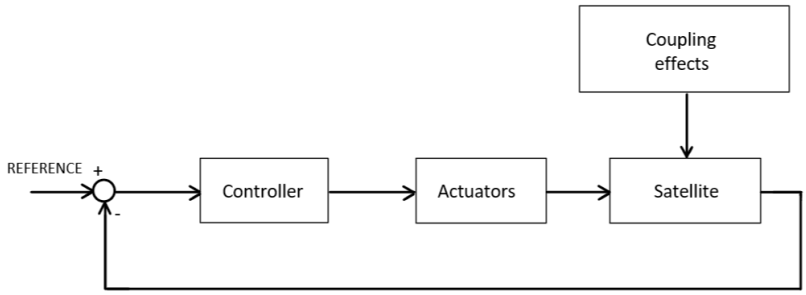


Figure 5.4: Satellite general scheme

The general scheme of the satellite model implemented in MATLAB/Simulink is quite similar to the one just described for the robotic arm with the significant difference, that the computed coupling effects represent an input to the "Satellite" block in which the attitude and position dynamics of the satellite are implemented (Fig. 5.5), referring to Part II. In the "Controller"

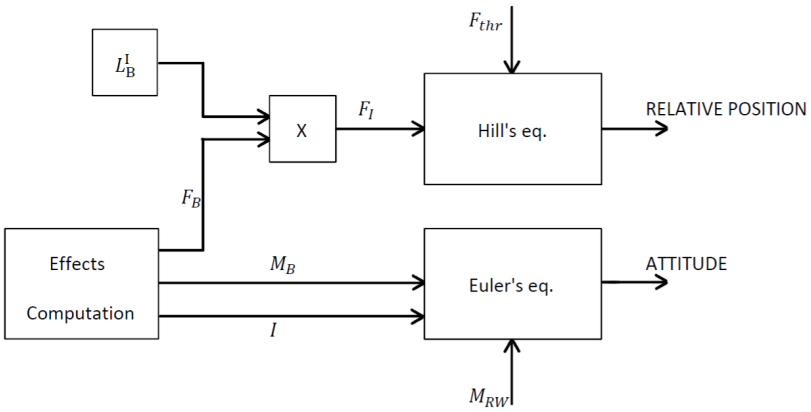


Figure 5.5: Satellite plant in Simulink

block, LQR controllers are designed in order to maintain desired attitude and position during manipulator motion. Last, in the "Actuator" block, dynamics of actuators (Reaction Wheels and thrusters) is introduced. In particular, a Pulse Width - Pulse Frequency (PWPF) modulator is designed (Fig. 5.6) for translating the continuous desired force signal to an on-off signal for the spacecraft impulse thrusters. PWPF modulators are widely used in ADCS systems for space use and their main elements are a Schmidt trigger, which consists of a double relay with hysteresis separated by a dead band and a first order filter, the output of which is the activation signal for the Schmidt trigger. Several parameters have to be defined for a correct implementation:

- K_m is the filter gain.
- T_m is the time constant of the filter.

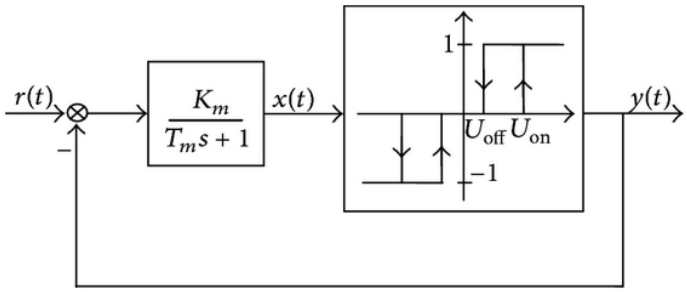


Figure 5.6: PWPF general scheme

- U_{on} is the activation threshold.
- U_{off} is the deactivation threshold.
- $h = U_{on} - U_{off}$ is the hysteresis width.
- Δt_{on} is the on-time of the thruster.

5.2 Unified Robot Description Format

The URDF is an XML file with dedicated tags for physical features of the robotic model. With

```
<!-- Link 1 -->
<link name="Link_1">
  <inertial>
    <origin rpy="0 0 0" xyz="0.075 0 0"/>
    <mass value="1.5"/>
    <inertia ixx="0.0009" ixy="0" ixz="0" iyy="0.0366" iyz="0" izz="0.0366"/>
  </inertial>
  <visual>
    <origin rpy="0 1.57 0" xyz="0.075 0 0"/>
    <geometry>
      <cylinder radius="0.025" length="0.15"/>
    </geometry>
    <material name="Blue"/>
  </visual>
</link>
```

Figure 5.7: URDF file: definition of i^{th} link

reference to Fig. 5.7, it is necessary to define the name of the link and, between $< inertial >< /inertial >$ tags, the origin (xyz) and orientation (rpy) of the local frame with respect to the previous joint. The origin and orientation of the "Base" frame are fixed (xyz="0 0 0", rpy="0 0 0"). Visual properties, within $< visual >< /visual >$ tags, are useful to observe the arm in a 3D space and study volumes and permeations during the motion. In fact URDFs support visual representation using primitive shapes or 3D meshes like COLLADA and STL files for a more accurate environment. It is also possible to add a collision model, similar to the visual one, useful during MATLAB simulations. For dynamical purposes in the simulation environment (with the Robotic System toolbox), it is necessary to define inertial properties, within $< inertial >< /inertial >$ tags:

- mass;

- components of the inertia matrix expressed in the local frame.

```
<joint
  name="joint_01" type="revolute">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="base_link" />
  <child  link="link_01" />
  <axis xyz="0 0 -1" />
  <limit lower="-3.14" upper="3.14" effort="1000" velocity="0.5" />
</joint>
```

Figure 5.8: URDF file: definition of i^{th} joint

The joints properties are defined, as shown in Fig. 5.8 within the $< joint >< /joint >$ tags. It is necessary to define the name and type of the joint:

- prismatic
- revolute
- fixed

As for the link, the origin and orientation are defined with respect of the previous joint. Other information required are about the axis of rotation (for a revolute joint), parent and child links and limits for positions, velocities, accelerations, and forces. They often have friction and damping coefficient values as well.

By inserting all the geometrical and physical features listed in Chap. 9.1, "Home configuration" of the imported model in the MATLAB/Simulink environment is obtained, with null joint variable values $q = [0, 0]^T$, as shown in Fig. 5.9. With reference to the same figure, the base satellite is gray, the first link is orange and the second link (with the end-effector at its end) is red.

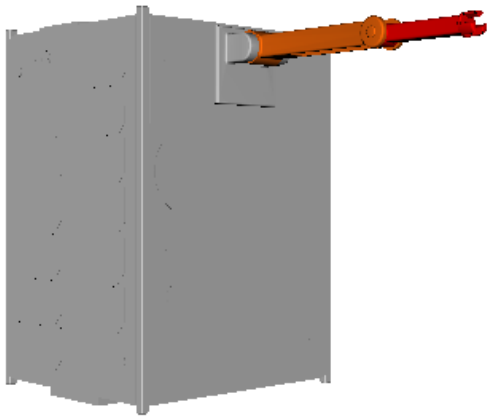
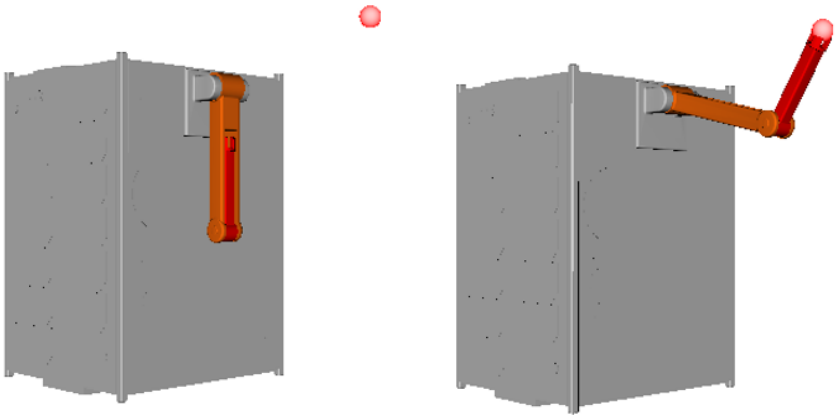


Figure 5.9: Space manipulator home configuration

Chapter 6

Manipulator Controller Design

The theoretical domain in which controllers are designed can be divided into joint space-based and task space-based controllers. With the former, tasks are first mapped into joint space through inverse kinematic techniques and then the controller is designed in torque space based on the information about the joint space. The latter is first designed in task space and then transformed into torque space, resulting in increased accuracy for high dimensional system. In this chapter both the strategies are addressed depending on the implemented control method and the results are presented and compared. For the simulations, the base satellite is supposed to be fixed in the desired position ($\mathbf{p} = [-0.3, 0, 0]^T$) and attitude (ideal quaternion), and the manipulator is moving from the initial configuration $\mathbf{q} = [-\frac{\pi}{2}, \pi]$. Both controllers are validated in reaching the desired end-effector position $\mathbf{p}_{ee} = [0.3, 0, 0.2]^T$ in $\mathcal{F}_B$.



(a) Start configuration (b) End configuration

Figure 6.1: Space manipulator configurations

6.1 PID Controller

PID controller design is based on the independent joint space control architecture, shown in Fig.6.2. The "Inverse Kinematic" block transform the desired end-effector pose into desired joint

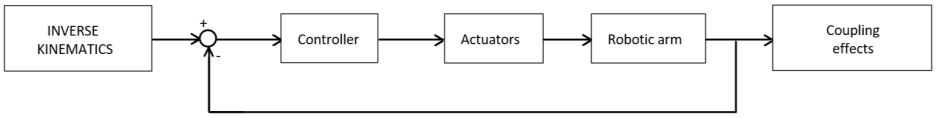


Figure 6.2: Joint space control architecture

values, using the "inverseKinematic()" function of the Robotic System Toolbox. The input of the controller are the joint position errors, while its output represents the required torques to drive the robotic arm in the desired posture.

The control gains of the PID controller are $k_P = 0.8$, $k_D = 0.5$ and $k_I = 0$. The choice of a null integral term is dictated by the absence of the steady state error during the simulations.

Next figures show the result of the simulation with PID controller. In particular, Fig. 6.3 shows the command torque provided to the two joints. The maximum values assumed by the two signals are just over $1Nm$ for the first joint and $1.5Nm$ for the second. Fig. 6.4 (a) shows

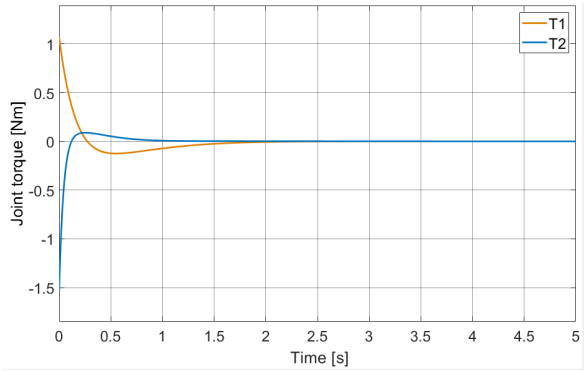
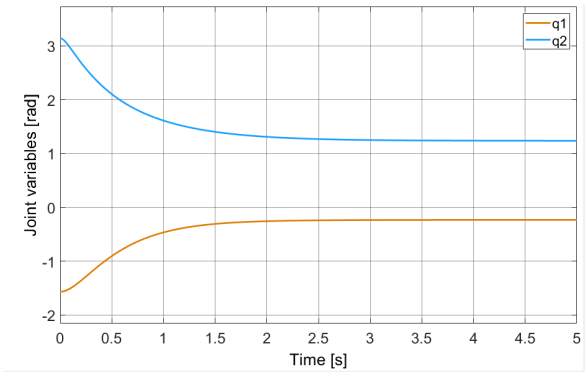


Figure 6.3: PID: joint command torques

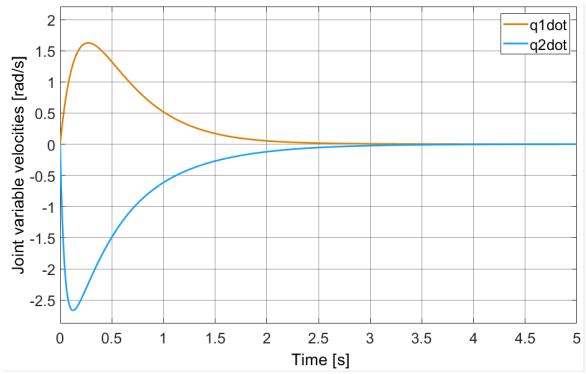
the evolution over time of the joint variables due to the input torques. The robotic arm reaches, successfully, the desired configuration corresponding to joint variable values $\mathbf{q} = [1.1, -0.2]^T$. Fig. 6.4 (b) highlights the joint variable velocities that it is convenient to keep low to reduce disturbances on the base satellite. Maximum values assumed by the joints are $1.6m/s$ for the first one and $-2.7m/s$ for the second one.

Position of the end-effector and its velocity in $\mathcal{F}_B$ are presented in Fig. 6.5 (a) and Fig. 6.5 (b) respectively. Notice that the end-effector reaches the desired position to perform the task and maximum values of velocities registered are about $0.08m/s$ in x and $0.05m/s$ in z.

Reaction forces and resulting moments, acting on the mounting point and translated to the CoM_B, are depicted in Fig. 6.6 (a) and (b) respectively. As introduced in Chap. 3, since the planar arm motion is constrained in $x - z$ plane, reaction forces are generated along x and z directions, while the resulting disturbing moment is about y direction. F_x oscillates reaching a peak value of $1N$, while F_z assumes the maximum value of $-0.3N$. These will affect the position of the servicer that needs to be fixed as much as possible for the assumptions previously done. Also the moments affecting the base oscillate before becoming null, once the final configuration is reached. In this case the maximum value registered is of $0.16Nm$.

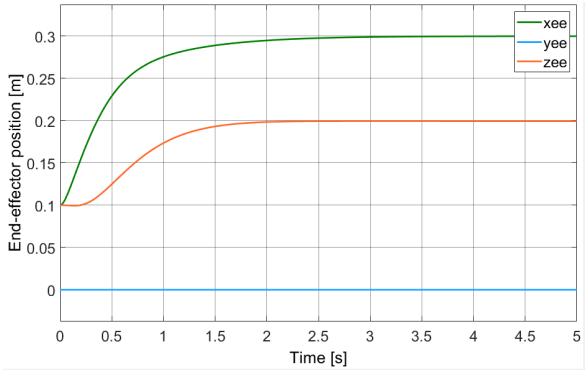


(a) Joint variable values

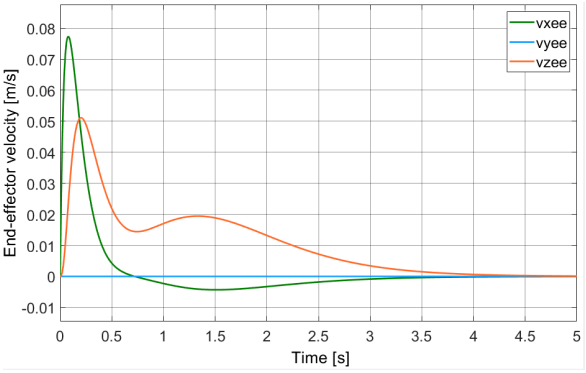


(b) Joint variable velocities

Figure 6.4: PID: joint variables evolution

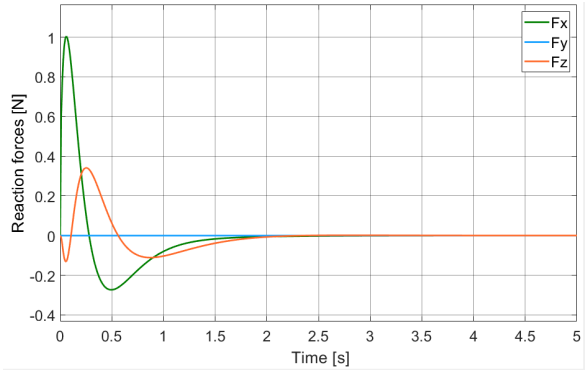


(a) End-effector position

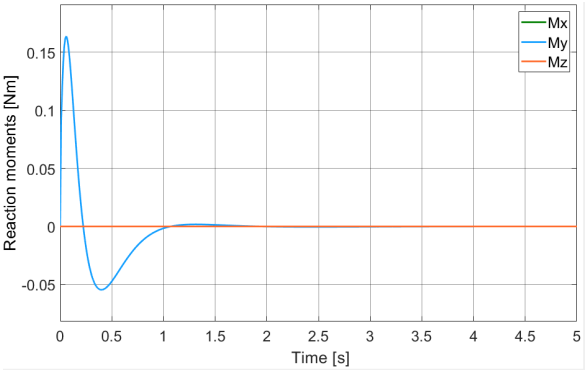


(b) End-effector velocity

Figure 6.5: PID: End-effector motion



(a) Reaction forces



(b) Reaction moments

Figure 6.6: PID: reaction forces and moments acting on the base satellite

6.2 RL Controller

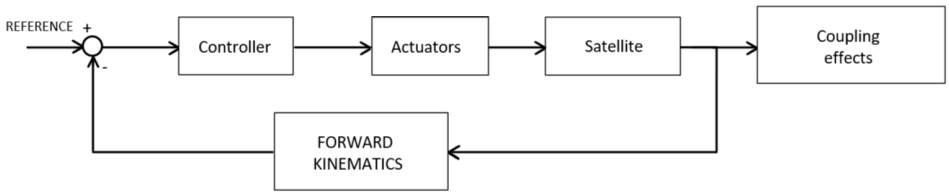


Figure 6.7: Task space control architecture

RL controller falls into the category of operational space control and its general architecture is depicted in Fig. 6.7. It is similar to the one already presented for PID controller, but in this case the "Inverse kinematic" block is not needed anymore since the logic of the controller is task space based. For the same reason the "Forward kinematic" block is added in order to obtain the end-effector position given a robotic arm posture.

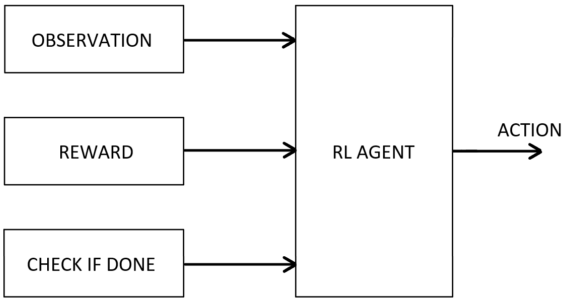


Figure 6.8: RL training scheme

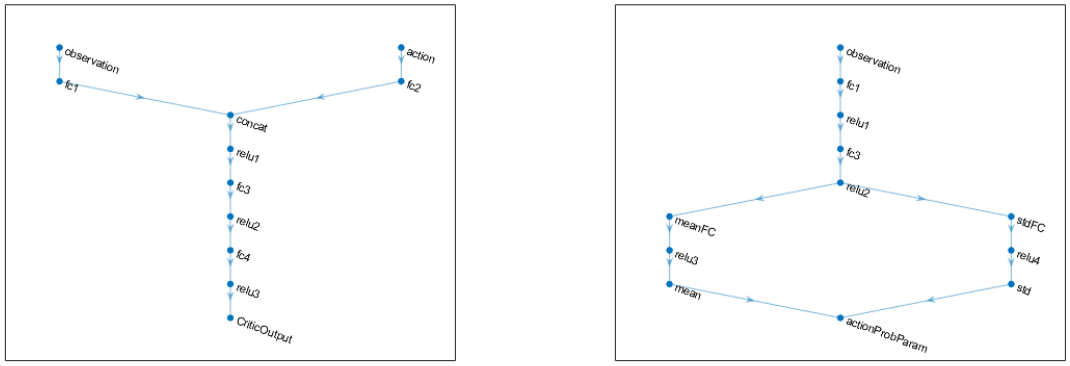
The scheme of the RL controller, based on the concepts introduced in Chap. 8, is presented in Fig. 6.8. Observations represent the state of the system at any given time t and are enclosed in a 14 element vector containing information about positions (sine and cosine of joint angles to avoid discontinuities) and velocities (joint angle derivatives) of the actuated joints, positions (x,y and z distances from $\mathcal{F}_B$ origin) and velocities (x,y and z derivatives) of the end-effector and joint torques from the last time step. Because these signals have different units, some of them are multiplied with scale factors to assure them to be in the same range as others. This is a key aspect because these signals will be fed into a neural network and high magnitude signals may cause one section of the network to dominate and undermine the effects of other signals.

Actions are normalized torque values $[-1, 1]$ for the actuated joints.

Rewards at each time step t are given by the relationships shown here

$$\begin{aligned} r_{ee} &= e^{-0.001(\Delta x^2+\Delta z^2)} \\ r_{angvel} &= -0.15(\dot{q}_1^2 + \dot{q}_2^2) \\ r_{action} &= -0.05(\tau_1 + \tau_2) \end{aligned} \tag{6.1}$$

where r_{ee} is a positive reward (Gaussian function) provided to the agent as the distances of the



(a) First critic (b) Actor

Figure 6.9: Plot of the first critic and the actor

end-effector from the desired position go to zero, r_{angvel} is a negative reward for high angular velocities and r_{action} is a penalty to the agent when it uses too much control effort. Note that each reward function is multiplied with a scaling factor to specify the relative importance on the overall reward. The latter is obtained by summing the positive and negative rewards as

$$r_t = r_{ee} + r_{angvel} + r_{action} \tag{6.2}$$

The final set of elements from the environment is a logical "is done" signal that determines when the simulation should be terminated because the system has reached a "bad" state and it does not make sense to simulate further. In this case the simulation is terminated when the end-effector intersects the base satellite.

In order to create an RL environment from the Simulink model, it is necessary to create specifications for the inputs and outputs and then an environment object, using the function "rlNumericSpec()" and "rlSimulinkEnv()" respectively, provided by the Reinforcement Learning Toolbox. One important aspect on training an RL agent is domain randomization. It is convenient to randomize the Simulink environment at the beginning of each training episode by providing a function handle to reset the environment. This helps the agent to learn a policy that is robust to the variations within the environment. In this case the reset function introduces randomness in the initial joint angles in a range of 5 degrees, keeping the other parameters constant.

As introduced in Chap. 8, the agent in this work is a SAC agent. In order to create the two critics, used to learn the optimal Q-value function ([11]), it is necessary to create a deep neural network with two inputs (observation and action) and one output (Q-value function). As shown in Fig. 6.9 (a), critic networks consist of a few fully connected layers with reLu layers in between ([12]). From this network it is possible to obtain the two critics with the "rlQvalueRepresentation()" function.

The actor is modeled in a similar way. It takes observations as inputs and outputs the mean and variance of the policy distribution (Fig. 6.9 (b)).

After creating the actor and critics, SAC agent is derived using the "rlSACAgent()" function and the hyper-parameters are kept to default values. More information on SAC agents are available on the MATLAB documentation([11]).

Training option	Value
MaxEpisodes	3000
MaxStepPerEpisode	100
ScoreAveragingWindowLength	100
StopTrainingValue	60

Table 6.1: Training options

Now the final step is to train the algorithm using the ”train()” command. It is possible to specify with the ”rlTrainingOptions()” function a few training options such as

- ”MaxEpisodes”, that is how long the agent is going to train.
- ”MaxStepPerEpisode”, that is the maximum number of steps to run in the episode if termination conditions are not met.
- ”ScoreAveragingWindowLength”, that is the window length for averaging the scores, rewards, and number of steps for each agent.
- ”StopTrainingCriteria”, that is the training termination condition, specified in this case as ”AverageReward”; the training stops when the running average reward equals or exceeds the critical value.
- ”StopTrainingValue”, that is the critical value of the training termination condition.

Some of the values, as the ”StopTrainingValue”, are obtained after analyzing the agent's performance for a few training iterations. Training options used for the simulation are listed in the Table 6.1.

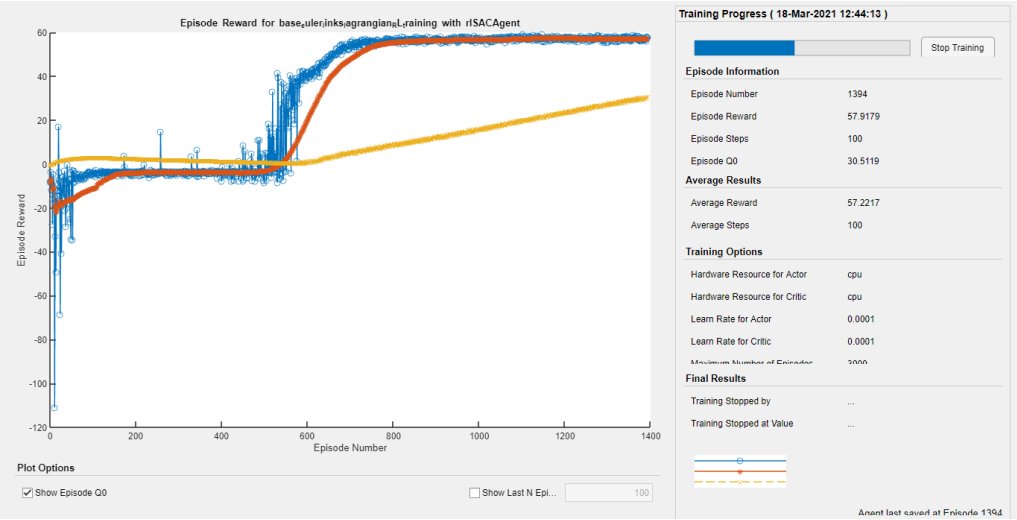


Figure 6.10: Episode manager

After the software setting up the training, it is possible to keep track of the training progress with the Episode Manager. It is evident that at the beginning of the training the agent fails to

perform the task for which it is trained, thus it does not receive too many reward per episode. In Fig. 6.10, blue line represents the episode reward and red line represents the average reward over a window of 100 episodes, while the yellow line is the expected long-term reward. Since the agent learns from experience, the episode reward increases over time and episodes, and converges to the maximum value.

Now it is possible to simulate the model with a pre-trained agent and verify that the robotic arm is able to reach the desired position successfully. Since there is randomness in the process, results may be different from one simulation to another.

Next figures show the performance of the RL controller. In particular, Fig. 6.11 shows the command torques provided to the two joints, by multiplying the actions with the maximum torque value ($0.1Nm$). This is a key aspect in the design of the actuators because it allows to take into account more stringent requirements about dimensions or weight of the same. The signal seems pretty noisy for practical purposes and it may be necessary to handle it, but this goes beyond the purpose of the thesis.

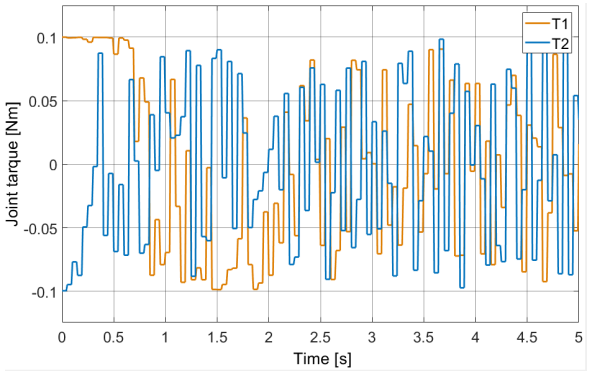
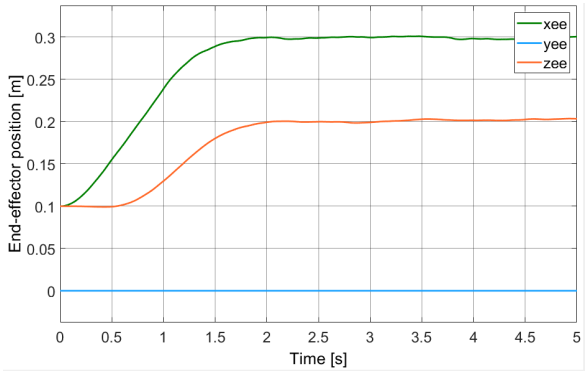


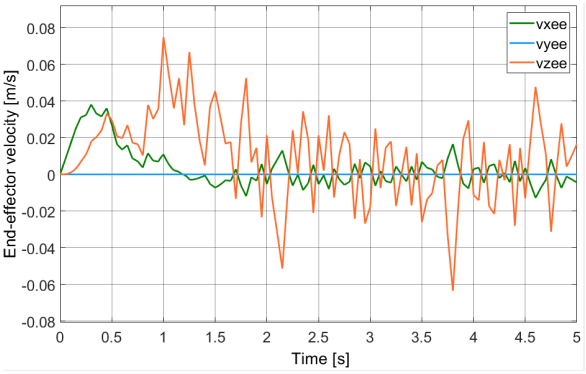
Figure 6.11: RL: joint command torques

As shown in Fig. 6.12 (a), the trained agent allows the robotic arm to reach the desired position and it is able to keep this position with an accuracy of the order of millimeters. Maximum velocities values of the end-effector are lower than ones obtained with the PID controller. In fact, in Fig. 6.12 (b) there are very short peaks with values equal to $0.04m/s$ and $0.06m/s$ along x and z directions, respectively. This is an advantage because it allows the end-effector to approach the mating point, positioned on the target, with a slower speed and therefore reducing the risks of a possible collision with the target itself.

Fig. 6.13 (a) shows the evolution over time of the joint variables, that is comparable with the evolution depicted in 6.4 (a). Another benefit in the use of RL method is showed in 6.13 (b): due to the torque limitation, resulting joint variable velocities are smaller than the ones detected with PID controller. This results in a reduction in the magnitude of the forces exchanged between the two bodies that make up the system and consequently in reduced disturbances affecting attitude and position of the base satellite. In fact, as shown in Fig. 6.14,maximum values of reaction forces and consequently of the reaction moment, even if prolonged in time to mantain the position of the end-effector, are lower than the ones obtained with the PID controller. In particular, F_x, F_z, M_y peaks are 60% lower, 5% lower and 65% lower respectively. Having lower disturbances acting on the satellite is very advantageous especially if dealing with small satellites, often equipped with limited resources intended for position and attitude control.

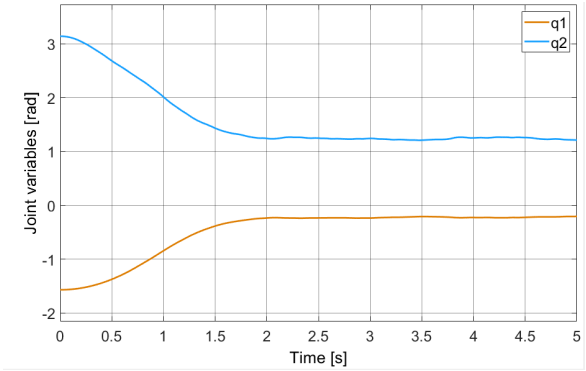


(a) End-effector position

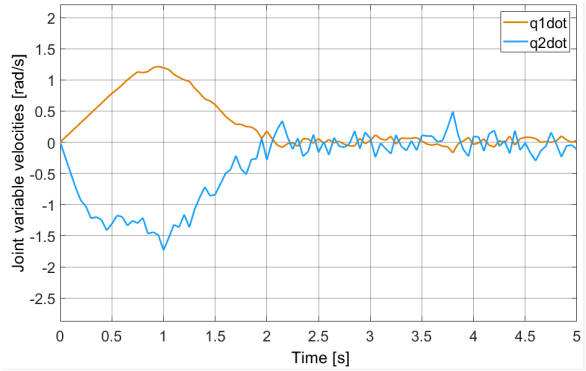


(b) End-effector velocity

Figure 6.12: RL: effects on the end-effector

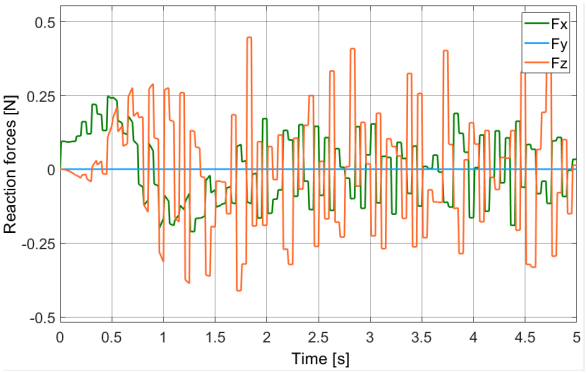


(a) Joint variable values

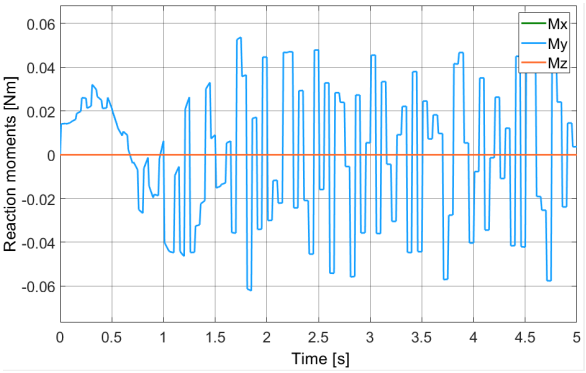


(b) Joint variable velocities

Figure 6.13: RL: effects on joint variables



(a) Reaction forces



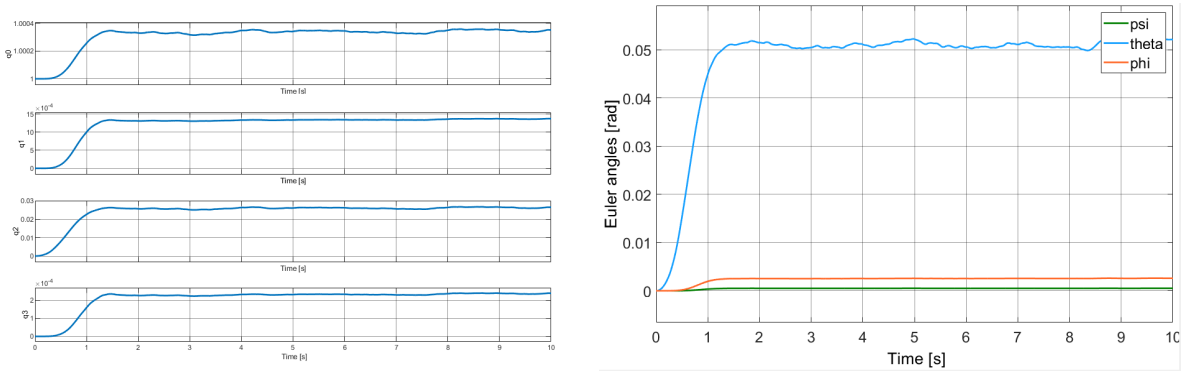
(b) Reaction moments

Figure 6.14: RL: reaction forces and moments acting on the base satellite

Chapter 7

Base Satellite Controller Design

This chapter deals with the design of a controller system that allows the spacecraft to keep desired position and attitude during the manipulator motion. In fact, reaction forces and moments derived by the actuation of the joints by means of RL manipulator controller (Chap. 6), affect the position and attitude of the non-controlled floating base, making them deviate from desired values, as shown in Fig. 7.1 and Fig. 7.2.



(a) Quaternions

(b) Euler angles

Figure 7.1: Attitude deviation

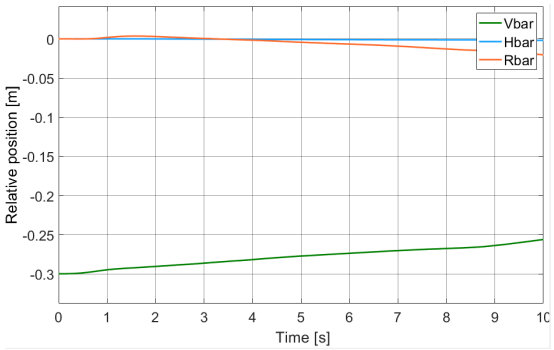


Figure 7.2: Position deviation

7.1 LQR Attitude Controller

For the implementation of the LQR attitude controller the following weighting matrices are used

$$Q = \begin{bmatrix} \frac{1}{0.05} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{0.05} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{0.05} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{0.01} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{0.01} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{0.01} \end{bmatrix} \quad \text{and} \quad R = 0.05 \begin{bmatrix} \frac{1}{0.2^2} & 0 & 0 \\ 0 & \frac{1}{0.2^2} & 0 \\ 0 & 0 & \frac{1}{0.2^2} \end{bmatrix}$$

Resulting command torque from the LQR controller is showed in Fig. 7.3. It allows the service to maintain the desired attitude with a maximum deviation of about 0.13 degree in the linear system (7.4 (a)), during the manipulator motion. The performance are quite similar for the more complex non-linear system (7.4 (b)) with a maximum deviation of 0.17 degree, but with the presence of a gyroscopic effect that, over time, generates a variation in attitude around the other axes, due to the variation in satellite inertia.

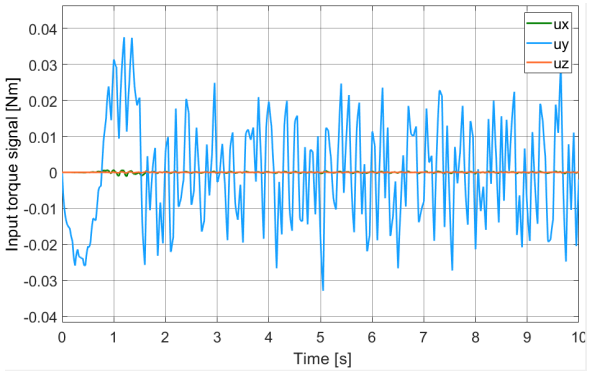
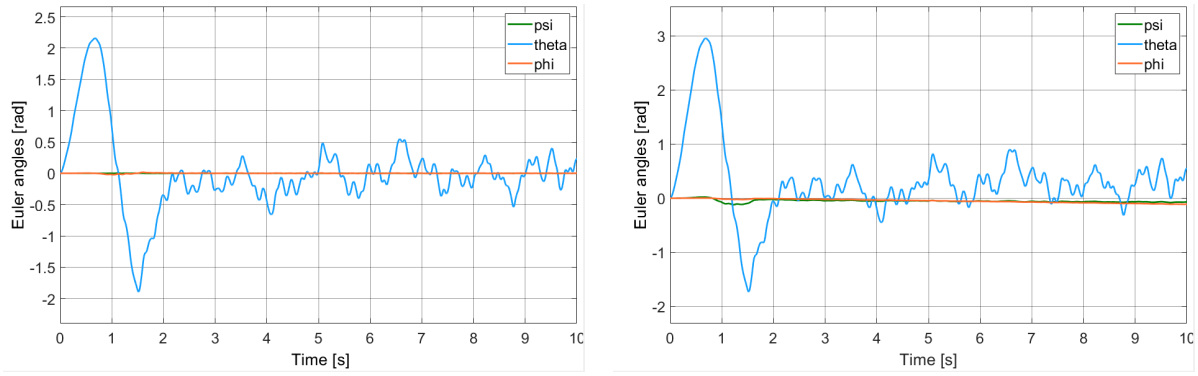


Figure 7.3: Input torque signal



(a) Linear system

(b) Non-linear system

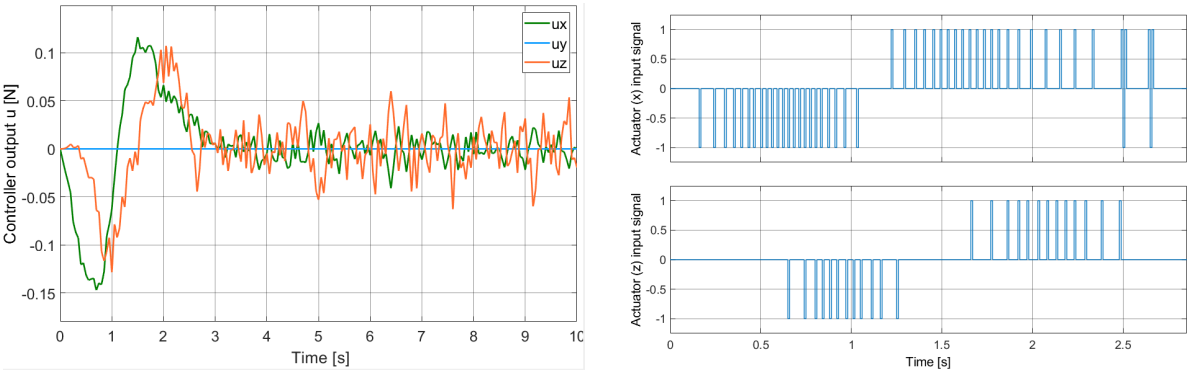
Figure 7.4: Servicer controlled attitude

7.2 LQR Position Controller

For the implementation of the LQR position controller the following weighting matrices are used

$$Q = \begin{bmatrix} \frac{1}{0.02} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{0.02} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{0.02} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{0.015} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{0.015} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{0.015} \end{bmatrix} \quad \text{and} \quad R = 0.01 \begin{bmatrix} \frac{1}{0.3^2} & 0 & 0 \\ 0 & \frac{1}{0.3^2} & 0 \\ 0 & 0 & \frac{1}{0.3^2} \end{bmatrix}$$

Control signal generated by the controller is depicted in Fig. 7.5 (a). As might be expected, the non-zero components are x and z to counteract the disturbances that affect the satellite base. Fig. 7.5 (b) shows the trend of the modulated signal, as introduced in the Chap. 5. Parameters chosen to configure the PWPF modulator are $K_m = 4.5$, $T_m = 0.12$, $U_{on} = 0.17$, $U_{off} = 0.05$. The controlled servicer is able to maintain almost constant its relative position: maximum deviation, both in x and z, is less than 3 millimeters during the motion of the manipulator.



(a) Controller output u

(b) Modulated thruster input signal, during manipulator motion

Figure 7.5: Thruster control signal

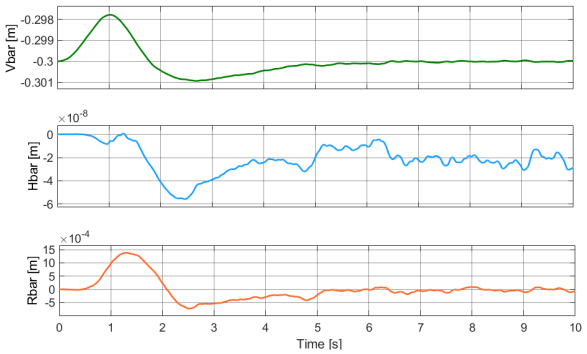


Figure 7.6: Servicer controlled relative position

Part V

Mission Scenario

Chapter 8

Rendezvous-Berthing Mission

Rendezvous and berthing is a key operational technology required for many missions involving more then one spacecraft. The process consists of a series of orbital manoeuvres (maneuvers) and controlled trajectories, which successfully bring the active vehicle (chaser) into the vicinity, and eventually in contact with, the passive vehicle (target).

A rendezvous mission ca be divided into a number of major phases ([18]):

- Launch, during which the launcher brings the chaser vehicle into a stable lower orbit in the orbital plane of the target.
- Phasing, whose objective is to reduce the phase angle between the chaser and the target spacecraft, by making use of the fact that a lower orbit has a shorter orbital period; this phase ends with the acquisition of an "initial aim point" from which the final part of the approach starts.
- Far range rendezvous, sometimes called "Homing", starts when the relative navigation between the two vehicles is available; it allows the chaser to reduce trajectory dispersion, acquire the target orbit, reduce the approach velocity and synchronize the mission timeline.
- Close range rendezvous is usually divided in two phases: closing and final approach. During the former the chaser reduces the range to the target and achieves conditions allowing the acquisition of the final approach corridor; the objective of the latter is to achieve berthing capture conditions in terms of positions and velocities and of relative attitude and angular rates.
- Mating consists of tasks that the robotic arm must perform to ensure a structural link between the two spacecrafts.

In this chapter a typical OOS mission scenario is proposed, the chaser configuration is described and the performance of designed model are shown.

8.1 Chaser configuration

In [24] a Navigation and Guidance system for OOS mission is designed, focusing on close range rendezvous phase. With reference to the Concept of Operations (ConOps) depicted in Fig. 8.1

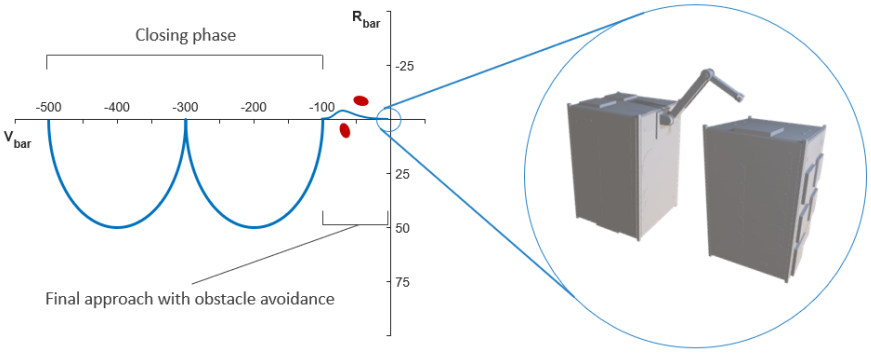


Figure 8.1: Concept of Operations scheme

the GNC system of the chaser delivers the vehicle, through a number of radial boost up to 100 meters from the target and then following a final linear approach, to a meeting point with zero relative velocities and angular rates. Here starts the mating phase in which the manipulator, a three-link planar arm mounted on the lateral face in the direction of the motion of the chaser, grapples the target to accomplish the tasks for which the mission is designed.

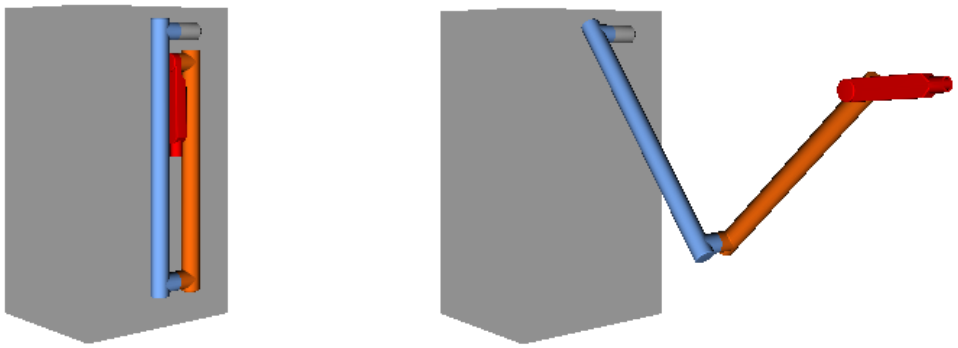
Here, geometrical and physical features of the spacecraft, useful for the simulation, are listed:

- The base satellite dimensions are 1.35×1.35×2.6m.
- The mass of the base satellite is 1500Kg.
- The mounting point is located along the vertical symmetry axis of the base satellite ($\mathbf{r}_0 = [10, 0, 15]^T cm$) and the manipulator motion is confined in the x-z plane.
- The links are assumed to be cylindrical and the dimensions are: 1st link length, radius and mass are 2m, 0.075m and 35Kg respectively; 2nd link length, radius and mass are 1.75m, 0.075m and 30Kg respectively; 3rd link length, radius and mass are 0.75m, 0.05m and 5Kg respectively.
- The links are assumed to be homogeneous, thus inertia matrices are diagonal. Non-zero components of inertia matrices are: $I_{1xx} = 1.97\text{e-}1 Kgm^2$, $I_{1yy} = 1.41\text{e}1 Kgm^2$, $I_{1zz} = 1.41\text{e}2 Kgm^2$, $I_{2xx} = 1.69\text{e-}1 Kgm^2$, $I_{2yy} = 9.24\text{e}1 Kgm^2$, $I_{2zz} = 9.24\text{e}1 Kgm^2$, $I_{3xx} = 1.25\text{e-}2 Kgm^2$, $I_{3yy} = 2.85 Kgm^2$, $I_{3zz} = 2.85 Kgm^2$.

More information about the satellite (physical features, GNC system, actuators, etc...) can be found on [24].

As done for the space manipulator system used for simulations in Chap. 9, by considering all the geometrical and physical features just listed, the URDF file of the chaser to be imported in MATLAB/Simulink environment is obtained. The "Home configuration" is represented by null joint variable values $q = [0, 0, 0]^T$.

RL controller is tested with the space manipulator system presented in the previous chapter. The same simulation environment and the same assumptions for RL algorithm are considered, even if some parameters are different due to the complexity of the system, such as the number of links and their larger dimensions. For this reason a little accuracy has been sacrificed in reaching



(a) Start configuration (b) End configuration

Figure 8.2: Space manipulator configurations

the desired end-effector position to obtain a valid solution in an acceptable time, considering the limited computational capacity of the available PC.

Observations are represented by 20 element vector containing the same kind of information already considered. More important in this case is the uniformity of values and the necessity of scaling factors to avoid undermining of the effects of some signals; in fact there is a great difference of orders of magnitude for example between the distances involved (meters) and the accuracy in reaching the desired position (millimeters first, then centimeters). Consequently, rewards at each time step t are changed and are given by the relationships shown here

$$\begin{aligned} r_{ee} &= 0.2e^{-0.01\Delta x^2} + 0.1e^{-0.01\Delta z^2} \\ r_{forrw} &= 0.04x \qquad (x <= 4) \\ r_{angvel} &= -0.5(\dot{q}_1^2 + \dot{q}_2^2 + \dot{q}_3^2) \\ r_{action} &= -0.05(\tau_1 + \tau_2 + \tau_3) \end{aligned} \tag{8.1}$$

where r_{forrw} reward is added to incentive the robotic arm to reach the position in x and prevent the simulation from converging to the solution that corresponds to reaching only the position in z (easier to reach). Notice that scaling factors are changed. The overall reward for each time step t is given by

$$r_t = r_{ee} + r_{forw} + r_{angvel} + r_{action} \tag{8.2}$$

For the simulation, the base satellite is supposed to be fixed in the desired relative position $p = [-4, 0, 0]^T$ and attitude (ideal quaternion) and the manipulator is moving from the initial configuration $q = [-\frac{\pi}{2}, \pi, \pi]^T$. The desired end-effector position to reach is $p_{ee} = [4, 0, 1]^T$ with respect to the body frame centered in the CoM of the satellite, which corresponds to its geometrical center. Simulation results, after the training of the agent, are presented in next figures.

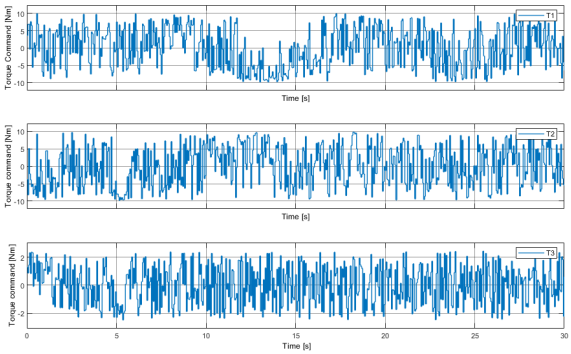
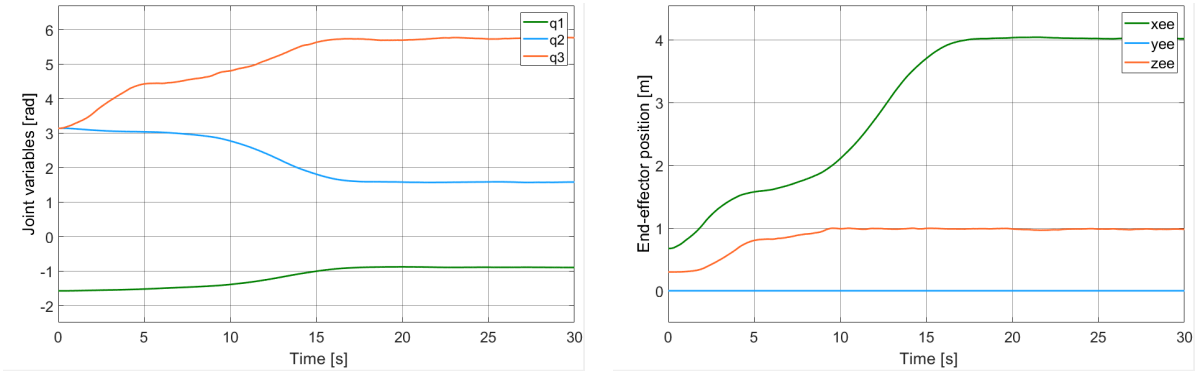


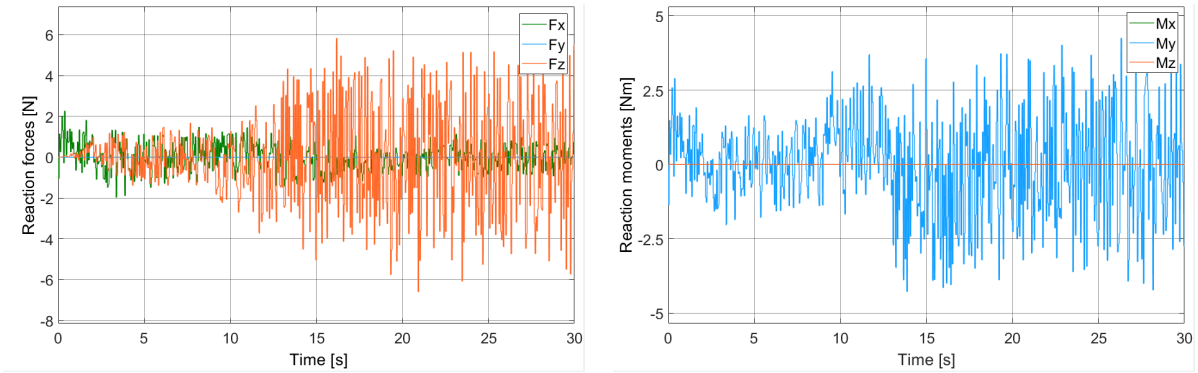
Figure 8.3: Joint command torques



(a) Joint variable values

(b) End-effector position

Figure 8.4: RL: effects on joint variables



(a) Reaction forces

(b) Reaction moments

Figure 8.5: Reaction forces and moments acting on the base satellite

Conclusions

The proposed work illustrates the control problem of a space manipulator, operating in a typical On Orbiting Servicing mission environment. Its kinematic model is first derived with a systematic and general approach using Denavit-Hartenberg convention, then dynamic equations are obtained using Lagrange formulation. Thus, a control system is designed with Reinforcement Learning algorithm, since machine learning could allow more complex missions design to be carried out with greater autonomy and chance of success when human intervention is difficult or limited given the lack of visibility, the required amount of information and communication time lag.

Simulations carried out are conditioned by the computational capacity of the available PC: obtained data are the result of a compromise between performance and computation time. Obtained results, compared with a classical PID controller, show anyway the ability of a trained agent to perform successfully a task, adapting to different working conditions. In this case, the agent allows the end-effector (considered as a point located at the end of the mechanical structure) to reach the desired position with different initial conditions, actuating the joints of the mechanical system. In addition, it is possible to train the agent by entering some design constraints such as the maximum torque available for joints actuation, the angular velocity of the links, or end-effector speed to safely perform the task that characterizes the mission. This indirectly improves the control of the base satellite as it is affected by minor intensity disturbance forces and moments, consequences of the manipulator motion. In fact, the proposed control system based on LQR method is able to maintain position and attitude in an acceptable range, showing pose-keeping ability and robustness of LQR design.

As shown in the last part of the thesis, designed model is valid also for more complex mechanical structure such as robotic arms with more links or more complex geometries. It is also possible to add an end-effector, which in turn consists of links and joints (revolute or prismatic ones), that can perform the task for which the mission was designed. In fact, parameters of the controller can be changed according to the number of variables (number of joints to be implemented and therefore number of observations, rewards and actions to be performed) and the constraints of the problem (limits on position, speed, etc.). Obviously a more complex model will have a higher computational cost and consequently longer simulation times, with the same machine.

Another future development of the proposed research could be the implementation of more complex movements consisting, for example, of a succession of maneuvers to avoid an obstacle, to reach a point that is not easy to access or even to transport an object from one point to another. To achieve these goals different ways are available: one is to train multiple agents, each one to reach an intermediate position (hold point), and switch the controller once the end-effector reaches the hold point; another could be to upgrade a pre-trained agent with new training making

the system able to complete the mission autonomously. All this could require, for example, the implementation of advanced guide and navigation system based on image recognition with machine learning and, in particular, transporting an object from one point to another introduces the problem of multi-body dynamics related to the acquisition and release of an object in space environment.

Artificial intelligence is a very promising field that breaks into our everyday lives in an increasingly obvious way. Once it reaches the necessary maturity, it can be a very important resource in the space environment that can relieve the workload of astronauts and allow them to stay in space safer. It could be, along with all the new technologies emerging in recent years, essential for future exploration, for the construction of new orbiting stations and, who knows, to establish a human outpost on another planet.

Bibliography

[1] Andrew M. Long, Matthew G. Richards and Daniel E. Hastings. *On-Orbit Servicing: A New Value Proposition for Satellite Design and Operation.* Journal of Spacecraft and Rockets, July–August 2007.

[2] Joshua P. Davis, John P. Mayberry and Jay P. Penn. *On-Orbit Servicing: inspection, repair, refuel, upgrade, and assembly of satellite in space.* Center for Space Policy and Strategy, April 2019.

[3] Stephen J. Leete. *Design for On Orbit spacecraft Servicing.* NASA Goddard Space Flight Center.

[4] Andrew Ogilvie, Justin Allport, Michael Hannah, John Lymer. *Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System.* DARPA.

[5] Glenn Jorgensen and Elizabeth Bains. *SRMS History, Evolution and Lessons Learned.* NASA.

[6] Nancy J. Currie, Brian Peacock. *International Space Station Robotic System Operations - A human factors perspective.* NASA – Johnson Space Center, National Space Biomedical Research Institute.

[7] Alex Ellery. *Tutorial Review on Space Manipulators for Space Debris Mitigation.* Robotics, 2019.

[8] Ijar M. Da Fonsecaa, Luiz C.S. Goesa, Narumi Seitoa, Mayara K. da Silva Duartea, Élcio Jeronimo de Oliveira. *Attitude dynamics and control of a spacecraft like a robotic manipulator when implementing on-orbit servicing.* Acta Astronautica 137, 2017.

[9] Yaguang Yang. *Quaternion based model for momentum biased nadir pointing spacecraft.* Aerospace Science and Technology, 2010.

[10] Yaguang Yang. *Quaternion-Based LQR Spacecraft Control Design is a robust pole assignment design.* Journal of Aerospace Engineering, January/February 2014.

[11] MATLAB(R2020b). *Reinforcement Learning Toolbox, User's Guide.* `http://www.mathworks.com/`.

[12] MATLAB(R2020b). *Deep Learning Toolbox, User's Guide.* `http://www.mathworks.com/`.

[13] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo. *Robotics. Modelling, Planning and Control*. Springer, 2010.

[14] Álvarez Chavarría J S, Jiménez Builes J A, Ramírez Patiño J F. *Design cycle of a robot for learning and the development of creativity in engineering*. DYNA, 2011.

[15] E. Capello. *Dynamics and control of aerospace vehicles lectures, "Dinamica di posizione e di assetto"*. Politecnico di Torino, 2020.

[16] S. Ali A. Moosavian and Evangelos Papadopoulos. *Free-flying robots in space: an overview of dynamics modeling, planning and control*. Robotica (2007).

[17] F. Dabbene, E. Capello. *Dynamics and control of aerospace vehicles lectures, "State-space based modern control"*. Politecnico di Torino, 2020.

[18] Wigbert Fehse. *Automated Rendezvous and Docking of Spacecraf*. Cambridge University Press, 2003.

[19] Thomas Duriez, Steven L. Brunton, Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer, 2017.

[20] Gu S., Holly E., Lillicrap T., Levine, S. *Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates*. IEEE International Conference on Robotics and Automation (ICRA), 2017.

[21] Liu C., Gao J., Bi Y., Shi X., Tian D. *A Multitasking-Oriented Robot Arm Motion Planning Scheme Based on Deep Reinforcement Learning and Twin Synchro-Control*. Sensors, 2020.

[22] Yudha P. Pane, Subramanya P. Nageshrao, Jens Kober, Robert Babuska. *Reinforcement Learning Based Compensation Methods for Robot Manipulators*. Engineering Applications of Artificial Intelligence, 2018.

[23] Kober, J., Bagnell, J.A., Peters, J. *Reinforcement learning in robotics: A survey*. International Journal of Robotics Research, 2013.

[24] D. Celestini. *Navigation and Guidance Algorithm for In-Orbit servicing Rendezvous Mission*. Master Thesis, April 2021