Università degli Studi di Padova

Dipartimento di Ingegneria Industriale

Corso di Laurea Magistrale in
Ingegneria Aerospaziale

Analysis and modeling of satellite flexible bodies in Simscape Multibody

Relatore:

Prof. Francesco Branz

Autore:

Gianmarco Grandis
n.2027435

Anno Accademico 2022/2023

Abstract

This Master Thesis provides an analysis to find a numerically efficient method to model a satellite flexible bodies in the MATLAB - Simulink environment. The numerical analysis is conducted by comparing two mathematical representations of flexible bodies: the *Lumped-parameter method* and the *Flexible-beam method*, which are implemented by the Simulation Tool to model and simulate the dynamics of a solar panel and a 3-DOF robotic arm.
This study is on purpose to contribute to a project conducted at the University of Padua in collaboration with the European Space Agency (ESA): it focuses on the development of the Guidance Navigation Control technologies (GNC) suitable to be applied to both In-Orbit Servicing (IOS) and Active Debris Removal (ADR) missions conducted by a chaser spacecraft equipped with a robotic arm that can grab space debris and lock onto other satellites; in particular, the research activity under contract is a simulator of the dynamics of Close Proximity Operations (CPOs) scenarios, called Functional Engineering Simulator (FES).
The solar panel and robotic arm mounted on the spacecraft behave as flexible bodies and need to be studied because their deformations may occasionally become severe enough to affect the performance of their respective systems. If these effects occur, vibrations are significantly amplified, accelerating the rate of mechanical wear, increasing power consumption, and interfering with high-precision tasks. The goals of this Master Thesis are: (1) to create a reliable simulation tool to study the dynamic response of the solar panel subject to external perturbations and the effects of the flexible elements of the robotic arm during its motion, (2) to compare the two flexible body methods in terms of accuracy of results and execution time, (3) to find the mathematical model that can adequately represent the flexible body theory so that it can be used to model a real-time simulator.
The solar panel is modeled according to two different scenarios, which represent two examples of space mission architectures requiring robotic operations in the near future. Its behavior is studied by analyzing the response to impulses caused by external perturbations acting on the solar panel surface. On the other hand, the robotic arm is modeled to follow a rectilinear path from a starting point to an end point and the behavior of its flexible elements is studied at each time-step of this motion.
The *flexible-beam method* consists of using an existing Simscape Multibody block to model the spacecraft element. Hence, it has been considered as a benchmark in order to verify the reliability of the *lumped-parameter method*, for which the results demonstrated that it is a valid tool for describing the behavior of flexible bodies of a spacecraft, mainly due to its fast execution times, which make it suitable for modeling in a real-time simulator.

Sommario

Questa Tesi Magistrale fornisce un'analisi per trovare un metodo numericamente efficiente per modellare i corpi flessibili di un satellite. L'analisi numerica è condotta su MATLAB - Simulink confrontando due rappresentazioni matematiche dei corpi flessibili: il metodo a "parametri forfettari" e il metodo della "trave flessibile", utilizzati per modellare un pannello solare e un braccio robotico a 3 GdL.
Questo studio ha lo scopo di contribuire ad un progetto condotto presso l'Università di Padova in collaborazione con l'Agenzia Spaziale Europea (ESA): esso si concentra sullo sviluppo di tecnologie di Guidance Navigation Control (GNC) per missioni di In-Orbit Servicing (IOS) e Active Debris Removal (ADR), condotte da un veicolo spaziale, dotato di un braccio robotico in grado di afferrare detriti spaziali e agganciare altri satelliti; in particolare, l'attività di ricerca oggetto del contratto è un simulatore della dinamica di scenari per Close Proximity Operations (CPOs), denominato Functional Engineering Simulator (FES).
Il pannello solare e il braccio robotico, montati sul satellite, devono essere studiati perché le loro deformazioni possono occasionalmente diventare abbastanza severe da influenzare sia le proprie prestazioni che quelle degli altri sistemi. Se ciò dovesse accadere, le vibrazioni verrebbero notevolmente amplificate, accelerando il tasso di usura meccanica, aumentando il consumo di energia e interferendo con i compiti che necessitano un'alta precisione.
Gli obiettivi di questa tesi Magistrale sono: (1) creare un Simulatore adatto per studiare la risposta dinamica del pannello solare soggetto a perturbazioni esterne e gli effetti degli elementi flessibili del braccio robotico durante il suo movimento, (2) confrontare i due metodi dei corpi flessibili in termini di accuratezza dei risultati e tempi di simulazione, (3) trovare il modello matematico che possa rappresentare adeguatamente la teoria dei corpi flessibili, in modo da poter essere utilizzato per modellare un vero simulatore.
Il pannello solare è stato modellato secondo due diversi scenari, i quali rappresentano due esempi di architetture di missioni spaziali che prevederanno operazioni con bracci robotici nel prossimo futuro. Il suo comportamento è stato studiato analizzando la risposta degli impulsi causati dalle perturbazioni esterne che agiscono sulla superficie del pannello solare. Invece, il braccio robotico è stato modellato per seguire un percorso rettilineo da un punto iniziale a un punto finale e il comportamento dei suoi elementi flessibili è stato studiato ad ogni istante del suo movimento.
Per modellare un elemento del veicolo spaziale, il metodo della *trave flessibile* utilizza un blocco già esistente di Simscape Multibody. Per questo motivo, esso è stato considerato come metodo di riferimento per verificare l'affidabilità di quello a *parametri forfettari*, per il quale i risultati hanno dimostrato essere uno metodo valido per descrivere il comportamento dei corpi flessibili di un veicolo spaziale, soprattutto grazie ai suoi rapidi tempi di simulazione, i quali lo rendono adatto per modellare in un vero simulatore.

Contents

List of Tables

List of Figures

1. Introduction

The following chapter introduces briefly the context in which the research work takes place. It is explained what In-Orbit Servicing missions are and how they are a solution to extend the operational life of existing spacecraft and to the space debris problem. Secondly, space manipulators and their applications to In-Orbit Servicing missions are presented. In conclusion, the motivation and objectives of the thesis are reported, and a brief summary of the performed work is given.

1.1 In-orbit servicing

In-orbit servicing (IOS) refers to in-orbit operations conducted by a space vehicle like refueling and commodities replenishment, orbit modification (relocation) and maintenance, upgrade, repair, assembly. Every mission requires a target which can be a non-cooperative or cooperative space object [1]. Non-cooperative targets are space objects like pieces of orbital debris, derelict rocket bodies, and non-operational satellites that offer no support for relative navigation [2]. On the contrary, cooperative targets offer features designed to help the servicer navigation system with active information [3].

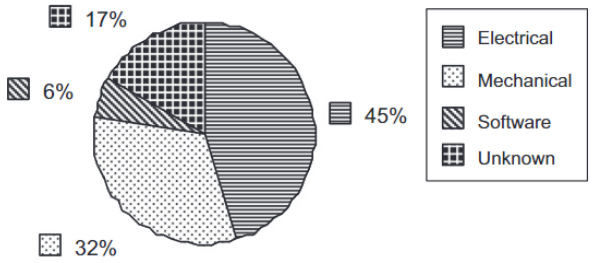
The interest in space missions involving close-proximity operations is growing worldwide [4] and in the past decades many missions emphasized the growing importance of IOS. For instance, the assembly of the International Space Station (ISS) would not be

possible without satellite servicing. This need to rely on in-orbit servicing missions is due to the presence of space mission-related difficulties that have arisen in past years. To begin with, the risk of failure due to the inaccessibility of spacecraft in orbit, which can lead to permanent or temporary mission degradation [5]. To avoid this problem, there are certain design philosophies that allow a satellite to be operational for a long time without ever being modified as unreachable. However, these solutions can lead to some side effects:

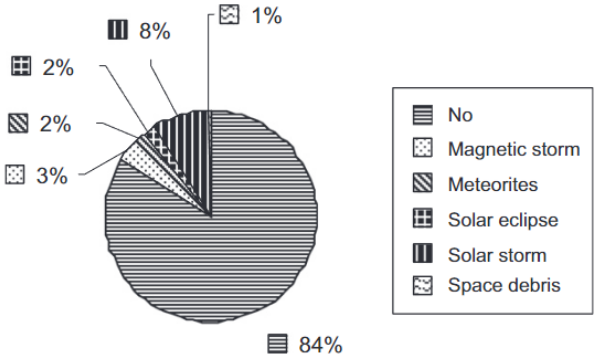
- increased production costs due to the increased mass of satellites;
- the system requirements may need to be modified during the spacecraft operational lifetime;
- satellites are more likely to become technically and commercially obsolete before the end of their missions [6].

Talking about spacecraft failures, Fig. 1a shows different types of failure that can involve the spacecraft. However, it is always difficult to attribute a cause to them. In fact, it has been verified that a few failures can occur due to environmental conditions (Fig. 1b), but a large number of spacecraft failures occur for unknown reasons.

Moreover, 41% of all failures happen within 1 year of in-orbit activities, which would suggest insufficient testing and inadequate modeling of the spacecraft and its environment, while 37% of all failures happen within 1-5 years [7]. One recent example is Intelsat-29e which was a geostationary communications satellite that suffered a fuel leak followed by a communication system failure, resulting in the loss of a satellite 3 years into its 15-year design life [1]. This event could have been avoided if an on-orbit servicing infrastructure in orbit could have helped save the spacecraft. The repair could have solved these problems by restoring some of the satellite's capabilities. In addition, an orbit modification could have moved Intelsat-29e out of the geostationary orbit (GEO) belt, removing a large piece of debris from a high-value orbit.



(a) Spacecraft failure type



(b) Proportion of failure due to space environment.

Figure 1: *Overall failure rate during an IOS mission [7]*

On the other hand, astronauts visited the Hubble Space Telescope five times between 1993 and 2009 to replace limited-life items such as batteries, gyroscopes and electronic boxes, and to install state-of-the-art science instruments [8]. Every time astronauts have visited Hubble, they have left it as a more capable and more productive observatory. In summation, the development of IOS solutions can impact satellite and launch service orders and, in general, the structure of the space industrial value chain [9]. Hence, if satellites are designed to be serviced, their nominal mission life-time could be extended to maximize the return on investment [10].

A NASA Report on IOS [11] concludes that in-orbit repair and refurbishment will directly improve overall mission reliability and will help to ensure mission success. Furthermore, these benefits will become more and more important as costs and missions challenges will increase [3]. Servicing missions can replace failed components to keep spacecraft operational, and can upgrade onboard components to improve spacecraft performance. In fact, the possibility to recycle existing spacecraft by extending their operational life instead of

replacing them, may be one of the possible solutions to make the space sector long-term sustainable [4]. A measure of system performance used to prioritize this decision is the operational availability (Eq. 1.1), which is a measurement of how long a system has been available to use when compared with how long it should have been available to be used.

$$\text{Availability}, A = MTBF/(MTBF + MTTR + MTFS) \tag{1.1}$$

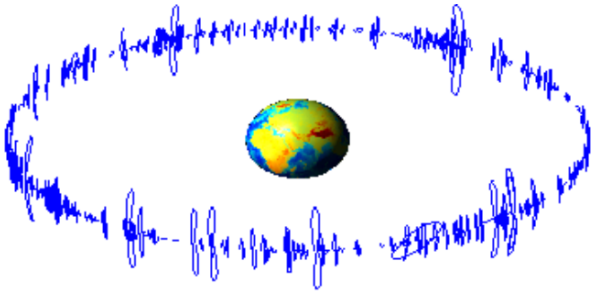
where MTBF is the mean time between failures and reflects reliability, MTTR is the mean time to repair and reflects maintainability and MTFS is the mean time for supply and reflects logistic capability [5].

Many IOS demonstrators have been launched over the last years and one of the most prominent projects in this area was the Defense Advanced Research Projects Agency's (DARPA) Orbital Express program, which was launched in 2007 and provided resounding evidence of the technical usefulness of in-orbit servicing [12]. The Orbital Express program demonstrated autonomous in-orbit refueling and reconfiguration of satellites [13] as well as autonomous delivery of a small payload representing an avionics upgrade package [14]. Another example is the Mission Extension Vehicle 1 (MEV-1), which raised its orbit to rendezvous with its client satellite, Intelsat 901 (IS-901), in geosynchronous equatorial orbit (GEO). On April 2, 2020, after three and a half months from launch, MEV-1 repositioned the spacecraft so that it could come back on line in its designated geosynchronous spot where it will remain for five years [15]. At the end of the mission it will repeat the process once again on new client spacecraft and provide new mission extension services. More recently, on 12 April 2021, the Mission Extension Vehicle 2 (MEV-2) successfully attached to the target satellite Intelsat 10-02 which had been active in geosynchronous orbit since 2004 carring communication traffic. However, MEV-2 did not move its satellite to a different orbit, as MEV-1 did; instead, it serves as a new engine and fuel tank to extend the life of the spacecraft for another 5 years [16].

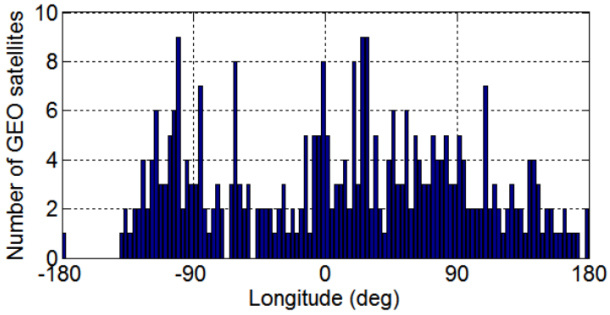
In conclusion, in-orbit servicing offers the capability to physically have access to satellites once in orbit. By doing so, space systems will be able to be repaired thanks to maintenance and upgrade operations. This is going to be a way to allow space operators to mitigate risks and these technologies will allow a greater return on investment by extending the life of existing satellites [17].

Another global interest involving close-proximity operations is the need to mitigate the spread of space debris, which seriously threatens the sustainability of space utilization. In 1978, Kessler and Cour-Palais [18] theorized that as debris generated from launch vehicles and damaged satellites accumulate in orbit, a tipping point is reached when they will continually collide with themselves, increasing the population of orbital debris and leading to an escalation of spacecraft failures. This outcome is called "Kessler Syndrome" [19]. Currently, many events occur where two satellites approach each other several kilometers apart. In 2009, the derelict Russian military satellite Kosmos-225 collided with the active commercial communications satellite Iridium 33 due to a not executed avoidance maneuver at an altitude of 789 km. The U.S. Space Surveillance Network had catalogued over 2000 large debris fragments from the collision [20]. Ram S. Jakhu et al. [11] show that more than 21,000 human-made objects larger than 10 cm in diameter are currently in orbit and over 95% of them are non-functional objects. In addition, there are about 600,000 objects measuring between 1 and 10 cm in diameter, and many hundreds of millions between 1 mm and 1 cm.

The space debris problem is particularly acute in some common orbits such as sun-synchronous orbits, between 600 to 800 km, or geo-synchronous (GEO) orbits, at an altitude of 36,000 km above the Earth equator. These are highly valuable orbital region with high traffic due to their advantageous characteristics for observation and communication missions (Fig. 2). Due to the non-spherical nature of the Earth, there are gravitational forces that attract objects toward two ”geopotential wells” [21]. The centers of the two wells are at 75°E and 105°W and GEO satellites must combat these gravitational forces when executing east-west station keeping.



(a) Orbit tracks of GEO satellites in the ECEF coordinate system



(b) Distribution of longitude of the satellites in GEO region

Figure 2: *Satellites orbits which contains most of the object passing through the GEO region and satellites longitude distribution [21]*

Therefore, if objects are left uncontrolled in GEO, they will naturally be captured in the closest geopotential well and will start oscillating around the center of the well [22]. The reason is that there is no atmospheric drag that removes abandoned objects over time [23].

The probability of collision (PC) can be calculated with the following equation [24]:

$$PC = 1 - exp(-VR \cdot AC \cdot SPD \cdot T)$$

where VR is the relative collision velocity [Km/s], AC is the collision cross-section [Km^2], SPD is the spatial density [objects/Km^3] and T is the time at risk [s].

There are ways to limit any collision damage and to control the spread of potentially harmful objects: new satellites can be either placed at orbital altitudes where the danger of collision is low, space launches can be planned to minimize the release of nonfunctional

objects and space junk can be directed to special orbits (graveyard orbits). Low Earth Orbit (LEO) Satellites at specific altitudes can be also retrieved and returned to Earth when their useful life is ended. Another strategy to minimize the effects of damage to functioning spacecraft can be reached by implementing redundant load path and fewer but stiffer and stronger structural members. In addition, shielding can be added to protect against dust-sized debris [25].

In-orbit servicing and Active Debris Removal (ADR) can be part of the solution. If satellites can be inspected and repaired in orbit, malfunctions can be identified and mitigated without the satellite becoming part of the debris population [26]. Furthermore, the active removal of large space debris by deorbiting, especially in critical sun-synchronous orbits, may prevent them from collide with other satellites [27]. There are many solutions to safely capture orbital objects, the majority relying on robotic systems. The European Space Agency (ESA) has been developing technologies in this field for decades. An encouraging option is the employment of an autonomous spacecraft (chaser) equipped with a highly dexterous robotic arm able to capture an uncontrolled space object [28].

1.1.1 Manoeuvering a robotic arm in space

The mechanical system of a robot manipulator consists of a sequence of links interconnected by means of articulations (joints). A manipulation apparatus is characterized by an arm with joints that ensures mobility, a wrist that confers dexterity, and an end-effector at the end of the manipulator that performs the task required of the robot [29].

Fig. 3 shows the control loop of a system. The capability to execute a task is given by the actuators block, which provides motion to the manipulation and movement apparatus. The connection with the outside world is made possible by the presence of sensors, which enable data acquisition. Finally, the control block permits to make the whole system an smooth working machine, reading data from the sensors and commanding the actuators with well-tuned control laws.

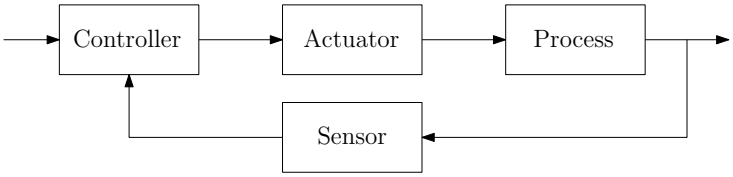


Figure 3: *Typical components of a feedback control loop [30]*

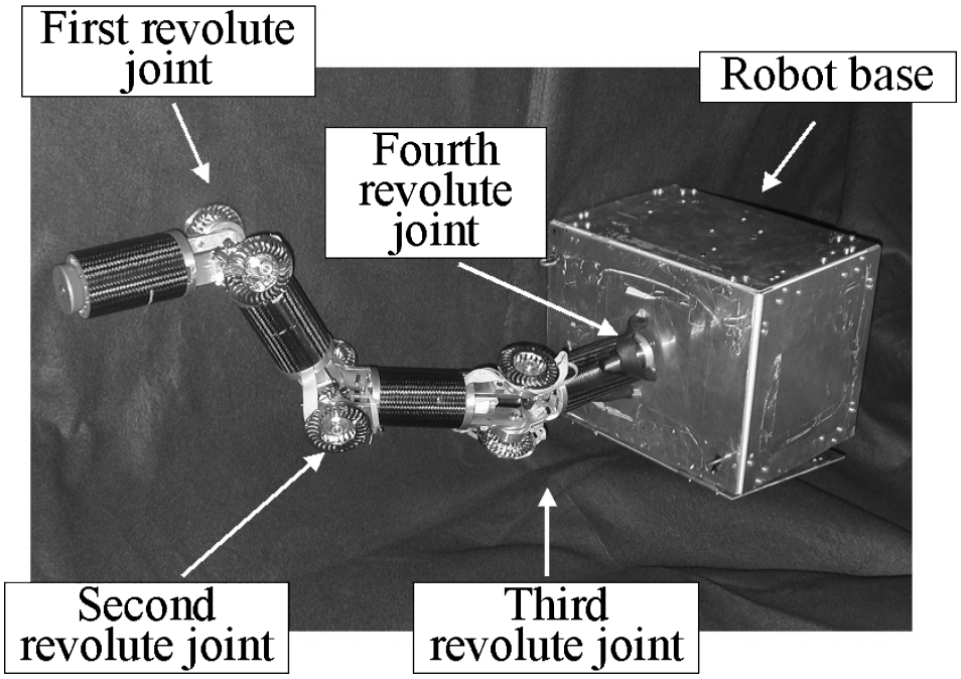


Figure 4: *Free-flying robot prototype with 4-DOF arm [32].*

The degrees of freedom (DOF) are appropriately distributed along the mechanical system: if the manipulator moves in Cartesian space, it must have at least 6 DOF, while if it moves in the plane it must have at least 3 DOF. Fig. 4 shows an example of free-flying robot prototype with 4-DOF arm, therefore, each joint adds a degree of freedom to the mechanism. [31]. The articulation between two consecutive links can be realized with a prismatic joint, which creates a relative translational motion between the two links, or a revolute joint, which creates a relative rotational motion between the two links. [29].

Furthermore, if a robotic has more than 6 DOF, the manipulator is redundant, and the same position can be reached through different configurations. However, there are some limitations on the EE orientation if a robotic arm has less than 6 DOF.

Nowadays, autonomous space manipulator systems are considered as a valid solution to perform various in-orbit space missions. A space manipulator system consists of a spacecraft with a robotic manipulator arm mounted on it [2]. The difference between a space robot and a fixed-base robot is the free-flying base. When a robotic arm is mounted on a satellite, its motions result in a dynamic coupling effect: the reaction force or torque induced by the arm motions will produce translation or rotation disturbances at the base [33]. This effect between the robotic arm and the base can severely affect the motion accuracy of space robots in some operational tasks [34], such as in-orbit assembly and in-orbit refueling [35]. In order to stabilize the satellite behavior during these tasks, attitude control of the base acts against the reaction of the robotic arm and control of the satellite-mounted robotic arm in order not to generate excess reaction against the base, must be achieved through the coordination between these two systems [36].

Researchers have proposed several motion planning and control methods to study active compensation control strategies for free-flying space robots. Zong et al. [37] proposed a concurrent learning algorithm that simultaneously uses past motion data points and instantaneous motion data of the system to minimize base disturbances problems after the space manipulator has captured an unknown target. Richard W. Longman et al. [38] studied the case of the Shuttle's Remote Manipulator System (RMS) (Fig. 5) by compensating the coupling force and torque with the Newton–Euler recursive equations. On the other hand, Papadopoulos and Dubowsky [39] proposed a method to control the position and attitude of a system spacecraft by using a Jacobian transpose controller to control the manipulator and the satellite base in a coordinated manner. This control scheme has the double advantage of allowing system motion planning to avoid impacts with the environment and maintaining a favorable manipulator configuration during the end-effector motion. However, this method is not suitable for continuous trajectory tracking in operational space.

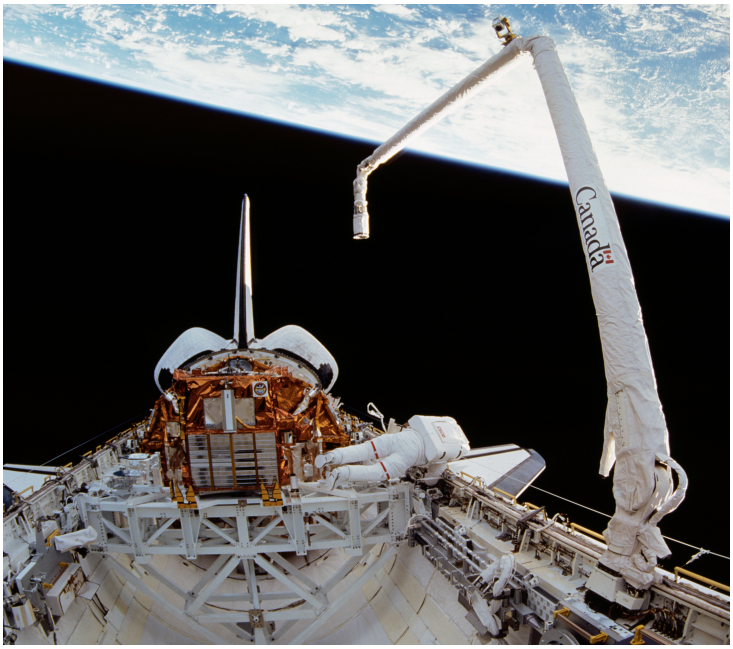


Figure 5: *Canadarm (right) during Space Shuttle mission STS-72 [40]*

In past decades, several mission proven the feasibility of space manipulators. The remote manipulator system (RMS), Canadarm, was successfully demonstrated in orbit on November 1981 [41]. It was given by Canada to NASA for installation on the shuttle orbiter and was used for deployment, retrieval, and handling of payloads up to 65000 lb mass, 15 ft in diameter and 60 ft in length (Fig. 5). In addition, the arm became essential in the assembly of the International Space Station (ISS) and provided a mobile and stable work platform for Extra Vehicular Activities (EVA), enabling astronauts on spacewalks to perform a greater variety of tasks in a shorter period time [42]. A more recent example is the e.Deorbit mission [43], which began to be developed in 2013 by ESA Clean Space initiative but was never launched. It was dedicated around the task of capturing and safely deorbiting the Envisat Earth-observing satellite from its orbit by robotic capture means: the robotic arm had to place the end-effector at the grasping point, a well-defined position at the target launch adapter ring, while compensating the station keeping errors of the chaser platform.

Within a space mission involving the use of a space manipulator, the robotic arm stays retracted until after the satellite is placed in orbit and it is deployed in later phases.

The most critical mission phase is the capture phase, in which the chaser platform and the robotic arm are operated simultaneously during the rendezvous manoeuvres: the chaser has to perform station keeping close to the target, while the robotic arm has to compensate the chaser station keeping errors. In particular, the main technical challenges of these missions are related to the design of the Guidance, Navigation and Control (GNC) system. If a space manipulator is employed during the capture phase, the GNC aspects become even more crucial, given the physical interaction between the robotic arm and the satellite itself [44].

In the previous paragraphs it was explained why the interest in In-Orbit Servicing and Active Debris Removal missions is growing worldwide. In addition to the benefits previously presented, IOS can be of great value for scientific purposes, like supporting large observation constellations, for orbital assembly of large structures and to support human spaceflight. Several companies and space agencies are pushing the research in this field and a few IOS and ADR demonstration missions have flown over the years: the "Restore-L", which is a NASA mission originally planned for 2020 that aimed to demonstrate the ability to service and refuel a satellite in orbit [45], the "RemoveDEBRIS", which is a European Space Agency (ESA) mission launched in 2018 that consisted of a deploying satellite releasing simulated space debris, which were then captured by a net, harpoon or drag sail [46], and the "ELSA-d", which is a mission developed by Astroscale, a Japanese startup, and launched in 2021, consisting of a mother spacecraft and a target satellite, which was supposed to be captured and removed from orbit using a magnetic docking system [47]. These missions demonstrate the potential of IOS and ADR technologies and highlight the growing interest in developing sustainable space environments.

1.2 Thesis motivation and objectives

The University of Padua, in collaboration with the University of Naples and the Polytechnic of Milan, is involved in a research activity under contract with ESA to study new GNC

algorithms for proximity operations for missions conducted by an autonomous spacecraft equipped with a robotic arm. This Master Thesis aims to contribute to the efforts of the University of Padua by analyzing the modeling of a spacecraft flexible bodies and finding a numerically efficient method to accomplish this task. Specifically, the goal of this thesis is to simulate in a MATLAB - Simulink environment the dynamics of a spacecraft solar panel and robotic arm considered as flexible bodies. This numerical analysis is conducted by comparing two mathematical representation of the flexible bodies: the *Lumped-parameter method* and the *Flexible-beam method*. In particular, the objectives of this Master Thesis can be summarized as follows:

1. Analysis of these two methods for modeling flexible satellite bodies
2. Creation of a MATLAB - Simulink tool which can simulate the dynamic response of spacecraft flexible bodies
3. Methods comparison to find a mathematical model that can adequately simulate the flexibility of bodies, so that it can be used to model a real-time simulator

1.3 Thesis Outline

The thesis work is divided into five Chapters, excluding the Introduction:

- **Chapter 2** provides a study of the Guidance, Navigation and Control design, introducing two different simulation scenarios and a description of the Functional Engineering Simulator. The physical model of the robotic arm and its control strategy are then presented.
- **Chapter 3** introduces the numerical analysis description. After a general approach to the flexible bodies theory, the description of the *Lumped-parameter*, the *Flexible-beam* methods and Simulink solvers is provided.
- **Chapter 4** describes in detail the Simulation Tool implementation.

- **Chapter 5** shows the results obtained from the numerical analysis. First, the results of the solar panel for both scenarios are presented; second, the results of the robotic arm motion are given. For both spacecraft elements, flexible body methods are compared. Finally, tests and results to find the best solver for all simulations are provided.

- **Chapter 6** sums up the thesis conclusions and explains which flexible body method is best suited to be implemented to model a real time simulator.

2. GNC and robotic arm combined control

Chapter 1.1 described why in-space robotic operations are of great interest for In-Orbit Servicing (IOS) and Active Debris Removal (ADR) mission applications. The potential of In-Orbit Servicing to extend the operational life of satellites and the need to implement Active Debris Removal to effectively address the space debris problem are well known to the space community. Furthermore, research into technical solutions to enable this class of mission is flourishing, driven in part by the development of next-generation sensors and control systems [44].

The Guidance, Navigation & Control (GNC) subsystem includes components used for position determination and the components used by the Attitude Determination and Control System (ADCS). Its design and performances strongly depend on the cooperativeness and collaborativeness of the target to be captured during an In-Orbit space robotic mission. Close proximity operations are divided into six phases [2]: (1) close-range rendezvous and short range closing in maneuver, (2) target identification, (3) attitude synchronization with the target, (4) manipulator deployment, (5) capture and (6) post-capture maneuvers. The GNC system guides the chaser through the rendezvous maneuver to a very short distance from the target; then, phases (2) and (3) are carried out. After the manipulator is released, the robotic arm reaches and grasps the target at the selected capture

point [44]. The University of Padua has created a complete simulation environment to test the developed GNC solutions, called the Functional Engineering Simulator (FES).

2.1 Functional Engineering Simulator (FES)

The FES is developed in the MATLAB - Simulink environment: it includes realistic models of actuators (thrusters) and sensors such as Star Tracker, Inertial Measurement Unit (IMU), and cameras. It also simulates environmental disturbances that affect system dynamics. To comply with the multibody approach, external perturbations are applied to each component of the system as external forces/torques; therefore, if a satellite with deployed solar panels is considered, the perturbations act not only on the center of mass of the satellite, but also on the center of mass of the appendages. In addition, the FES can simulate the capture phase of different targets in different scenarios and the stack dynamics after capture. In both phases, to assist the development of the GNC architecture and to validate its performance, the FES includes two validation tools: (1) the contact detection tool, which must ensure that no collisions or compenetrations occur between two distinct bodies; (2) the energy conservation tool, which verifies that the total energy and the linear and angular momenta of the system are conserved in the presence of external disturbances, non-linearities (flexure and sloshing), and deactivated actuators.

The principles on which the FES is designed are:

- Reliability and simulation accuracy;
- modular structure in order to maximize its operational flexibility, reusability and re-configurability;
- user-friendliness, with a complete documentation, a straightforward setup interface and detailed outputs in different forms.

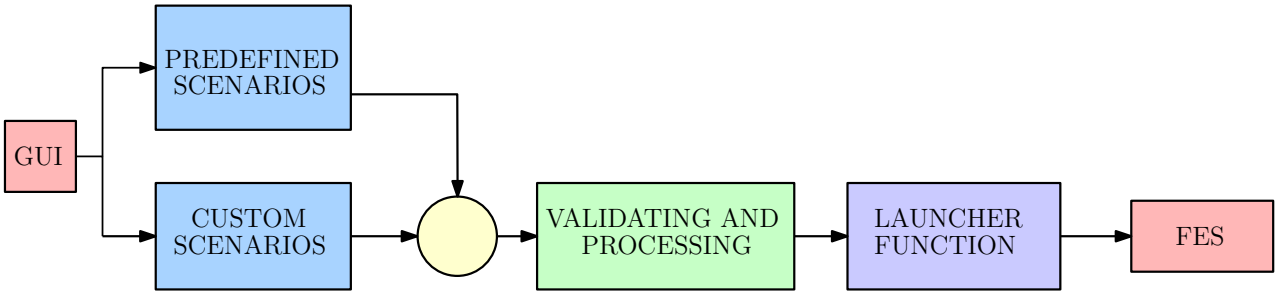


Figure 6: *FES high level data flow*

Futhermore, because the navigation sensors and algorithms rely heavily on synthetic images to operate, the PANGU (Planet and Asteroid Natural Scene Generation Utility) software plugin is fully integrated into the FES: PANGU is a third-party software that allows users to generate synthetic images of spatial environments and objects. It is also used to simulate visible/infrared cameras or electro-optical (EO) sensors-and active ranging devices.

Once launched, the FES evaluates all the parameters required for simulation of the chosen scenario and initializes a Simulink model. Fig.6 shows the data flow diagram of the simulator. Upon completion of the simulation, the data is saved in the workspace and in a user-designated working directory, a PDF report is generated, and additional custom post-processing tasks are performed to enable streamlined and fully automated execution. Finally, a realistic 3D representation of the simulation is provided using 3D Animation Toolbox.

2.2 Mission scenarios

This chapter defines two different scenarios, which represents two examples of mission architectures requiring robotic operations in the near future.

2.2.1 Servicing a large GEO platform

The first chosen scenario represents an IOS mission in a "geosynchronous equatorial orbit" (GEO), which is a circular geosynchronous orbit 35 786 km (22.236 mi) in altitude above Earth equator and following the direction of Earth rotation. This IOS mission architecture is designed for mission extension and servicing of large geostationary communication platforms. In addition, the chaser and the target are considered to have the similar geometry (Fig. 7).

The target is a communication satellite, which is expected to be operative and controllable and will be refueled, updated or repaired by the GEO platform, which is equipped with fiducial markers for the navigation aid task. In terms of capture interface, the target is assumed to be not prepared for capture and to be semi-cooperative: a semi-cooperative target is equipped with either active or passive artificial markers mounted according to a specific configuration which provide indirect information about the relative state. The capture point is considered the launcher adapter ring (LAR), a rigid interface common to many GEO satellites that fall under this scenario.

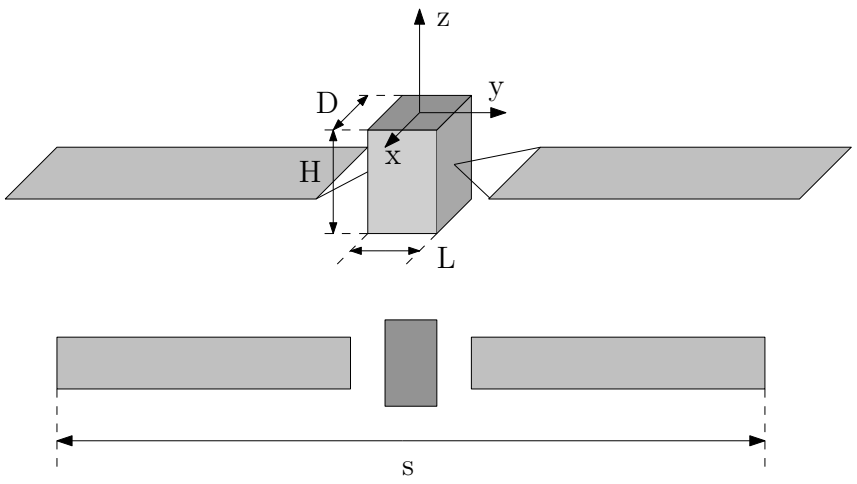


Figure 7: *Simplified chaser and target model for the Scenario 1.*

Table 1: *Scenario 1 chaser properties*

	Value	Unit
Dimensions (D,L,H)	$2100 \times 2300 \times 3100$	mm
Length (s)	25	m
Mass	1900	kg
Principal moments of inertia	[13500, 2000, 14000]	$\mathrm{kg\,m^2}$
Solar panels	$2.1 \times 10.85 \times 0.05$	m

Table 2: *Scenario 1 target properties and characteristics*

	Value	Unit
Dimensions (D,L,H)	$2500 \times 2800 \times 3500$	mm
Length (s)	31	m
Mass	2000	kg
Principal moments of inertia	[17000, 3000, 18000]	$\mathrm{kg\,m^2}$
Solar panels	$2.5 \times 13.6 \times 0.03$	m

Target	**Collaboration**	**Cooperation**
Geo communication satellite	Semi-collaborative	Semi-cooperative

2.2.2 Servicing a small platform of a large constellation in LEO

The second scenario represents a servicing mission in a "low Earth orbit" (LEO), particularly tailored to a small platform of a large constellation. The interest is focused on the capability of having a servicer platform used to maintain the constellation operative, providing services such as repair, refuelling, and deorbiting in case of failures or malfunction of some satellite subsystems.

Table 3: *Scenario 2 chaser properties*

	Value	Unit
Dimensions (D,L,H)	$1300 \times 1300 \times 1750$	mm
Mass	372	kg
Principal moments of inertia	[200, 200, 140]	$\mathrm{kg\,m^2}$

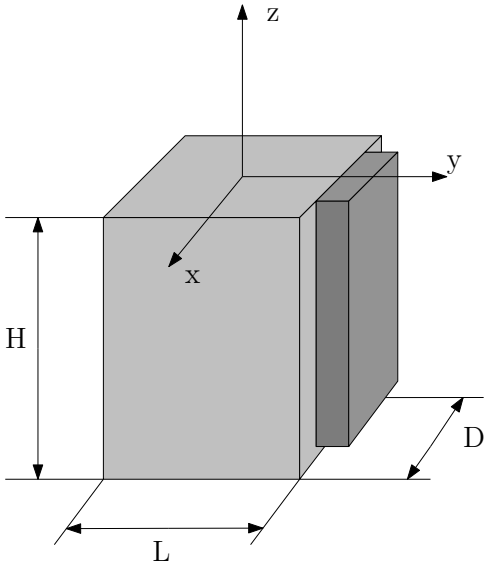


Figure 8: *Chaser spacecraft for SC2 operations. On the right is shown the preliminary allocation of the stowed robotic arm subsystem.*

In this scenario the chaser is a target pointing LEO platform, without solar panels (Fig. 8). To avoid occlusion of relative navigation sensors during capture operations, the robotic arm subsystem is placed in the ±X or ±Y faces of the spacecraft.

The target is spinning around the axis along solar panels (Y) (Fig. 9) and is considered semi-cooperative and prepared for servicing, featuring a grapple fixture as grapple interface and fiducial markers across the spacecraft body as navigation aid. However, it is also considered non-collaborative during the mission phase; this assumption allows including both IOS and ADR mission architectures to a non-controllable and controllable satellite.

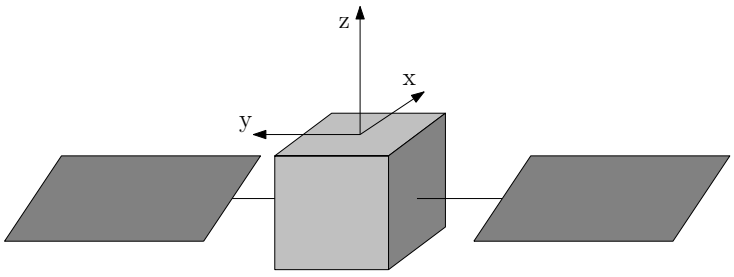


Figure 9: *Scenario 2 target representation and geometry.*

Table 4: *Scenario 2 target properties and characteristics*

	Value	Unit
Dimensions (D,L,H)	$1 \times 1 \times 1$	m
Mass	150	kg
Principal moments of inertia	$[45,\ 20,\ 50]$	$\mathrm{kg\,m^2}$
Solar panels	$1 \times 1 \times 0.03$	m

Target	**Collaboration**	**Cooperation**
Leo large constellation satellite	Non collaborative	Semi-cooperative

As explained before, a non-cooperative target is characterized by a known geometry but there are no easily recognizable artificial markers on its surface.

2.3 Robotic Arm physical model

For both scenarios, the chaser robotic arm architecture features a 7 DOF manipulator realized using 7 joints and a gripper, which connect four links and an EE (End-Effector). Hence, the arm is kinematically redundant to increase its dexterity. Fig. 10 shows the robotic arm configuration. Links are made by hollow cylinders with external radius equal to 0.05 m and thickness equal to 0.002 m while the length is not the same for all links (Table 5). The deployed robotic arm total length of *Scenario* 1 is preliminary defined as 3 m from Table 5, whereas that of *Scenario* 2 is defined as 2 m. The cylinders are made of 7075 aluminium alloy (AA7075) which is widely used for aerospace application: its physical and mechanical properties are shown in Table 6.

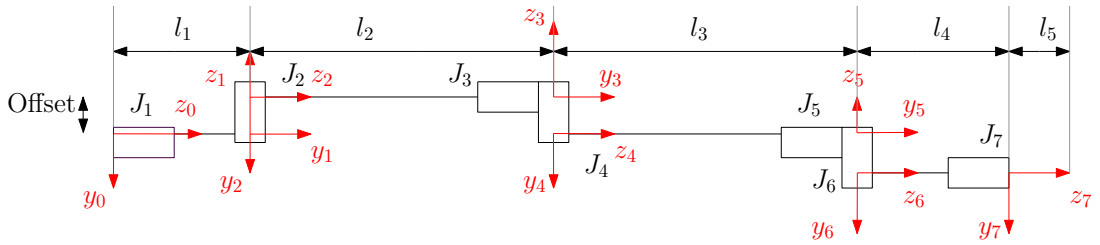


Figure 10: *7 DOF multi-offset arm configuration*

Table 5: *Kinematics parameters for the SC1 robotic arm.*

	Length	Radius	Thickness	Unit
Offset	160	/	/	mm
l_1	300	50	20	mm
l_2	1150	50	20	mm
l_3	1150	50	20	mm
l_4	400	50	20	mm
l_5	100	50	20	mm
Total length	3000	/	/	mm

Table 6: *7075 Aluminium alloy physical and mechanical properties*

	Value	Unit
Density	2810	$\mathrm{kg\,m^{-3}}$
Young's module of Elasticity	71.7	GPa
Tensile strength (σ_t)	572	MPa
Poisson's ratio (ν)	0.33	

The robotic manipulator that will be modeled and analyzed in this thesis is a simplified version of the one in Fig. 10: its model is described in Chapter 4.3.2 and involves a 3 DOF (three joint) robotic arm with two links considered as flexible bodies, while the third link is considered a rigid body.

2.4 Robotic Arm control

In this thesis a control strategy is implemented to control the dynamics of the robotic arm: the Proportional-Integrative-Derivative (PID) controller, which is a feedback control loop mechanism, widely used in industrial control systems and other applications requiring continuous modulation control. Since the PID controller can only be applied to *Single-Input Single-Output* (SISO) systems, it is necessary to use three controllers (one for each joint), each performing decentralized control, that is, considering only one joint at a time.

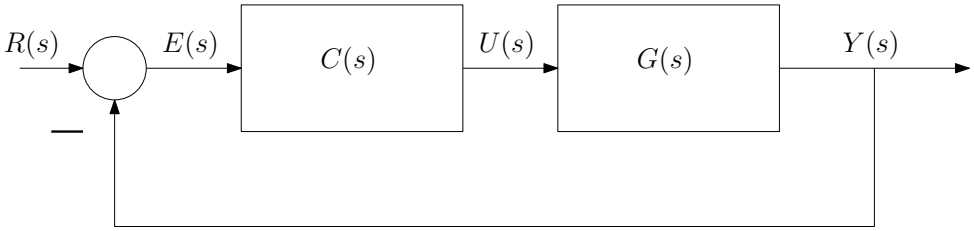


Figure 11: *Simple feedback control loop*

Fig. 11 shows a simple SISO (Single-input Single-output) control loop, where $R(s)$ is the reference signal, $E(s) = R(s) - Y(s)$ is the error between the reference and the output signal values and $C(s)$ is the controller. In addition, PID controllers can be described by their transfer functions by relating error $E(s)$ and controller output $U(s)$ [48]:

$$C_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.1}$$

However, an amplification of measurement noise in the manipulated variable is caused by the high-frequency gain of the pure derivative action. As a result, Eq. 2.1 cannot be implemented in practice. To solve this problem, the derivative action can be filtered with a first-order low-pass filter so that the real PID transfer function becomes [49]:

$$C_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N} s + 1}\right) \tag{2.2}$$

where N is a value between 8 and 16. Eq. 2.2 shows that the PID controller is composed of three actions:

- K_p, which is a proportional action. In a proportional controller, the steady-state error depends inversely on the proportional gain: an high value of K_p results in an increased response overshoot.

- K_i, which is an integral action that can automatically set the correct value of the output $u(t)$ so that the steady-state error is zero but, on the other hand, can worsen the transient response. It should be noted that, in the presence of an integral action,

the so-called integrator windup phenomenon may occur in the presence of saturation of the control variable.

- K_d, which is a derivative action that have the effect of increasing system stability, reducing overshoot and improving transient response.

Fig.12 shows the simplest block diagram of a PID controller. However, if the input signal $R(s)$ is a step function, the derived term will be impulsive. Thus, Fig. 13 shows a different PID block diagram, in which the low filter characteristic of the system is used to avoid impulsive responses of the derivative term.

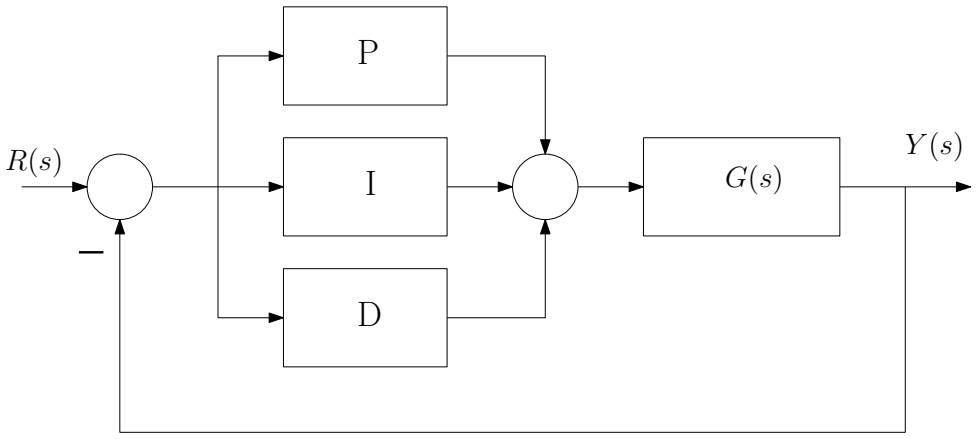


Figure 12: *PID block diagram*

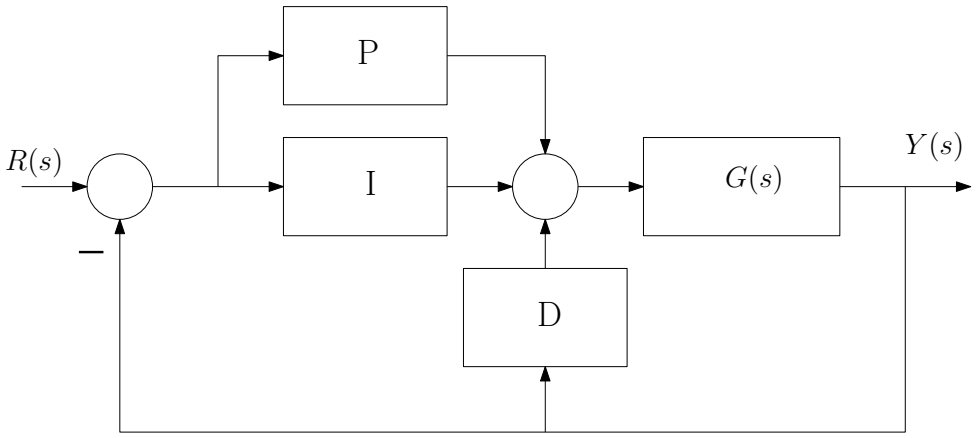


Figure 13: *Modified PID block diagram*

3. Numerical analysis description

The main objective of this Master thesis is to analyze the modeling of flexible bodies and find a numerically efficient method to study the behavior of a spacecraft solar panel and robotic arm, both considered as flexible bodies during IOS and ADR missions. To be used to model a real-time simulator, the method must be able to simulate the responses of flexible bodies with a execution time less than that imposed by the simulation: real time is the wall clock time, also called real-world time, which refers to the elapsed time determined by a chronometer, while the execution time is derived from the clock frequency and is the time required by the model to produce the response.

Numerous modeling techniques to describe a flexible body are available in the literature and to achieve this goal, two different flexible body methods have been compared: the *lumped-parameter method* and the *flexible-beam method*. The first section of this chapter introduces the general theory of the elastic problem, then the main differences between these two methods are provided, along with a general description of how the system responds to a given impulse and how discretization affects the body response. Secondly, a description of the lumped-parameter method in Simscape Multibody, shows how this method has been implemented to study the behaviour of a flexible body. Finally, the chapter describes in detail the features of Simulink solvers, which are used to simulate a dynamic system by computing its model states.

3.1 Flexible body theory

In constructing a multibody model, it is often assumed that bodies do not deform: each body is treated as a rigid unit, unable to undergo the mechanically induced distortions that real materials often experience. However, deformation can play an important role in some systems: the movements of neighboring bodies, the loads on their joints, the performance of control algorithms, are all vulnerable to the effects of deformation. For example, aerodynamic forces are known to induce visible flutter in aircraft wings, while impact forces often cause significant shaking in excavator arms. Wings and arms behave like flexible bodies, and their deformations occasionally become severe enough to affect the performance of their respective systems.

These effects tend to be particularly pronounced in resonant systems: resonance greatly amplifies vibration, accelerating the rate of mechanical wear, increasing energy consumption, and interfering with high-precision tasks. Chapter 1.1.1 describes that robotic manipulators employ active vibration control to mitigate the effect of vibration on the positioning accuracy of the end effector. To properly model such systems, it is necessary to capture the behavior of their flexible bodies [50].

3.1.1 General theory

One of the most important cases in which the elastic problem can be solved is that of the deformable beam, also known as the De Saint Venant problem (D.S.V.) [51]. The body analyzed within this problem is a constant section rectilinear beam, also called D.S.V solid (Fig. 14). The z-axis corresponds with the axis of the beam, while x and y axes are the barycentric and principal axes of inertia.

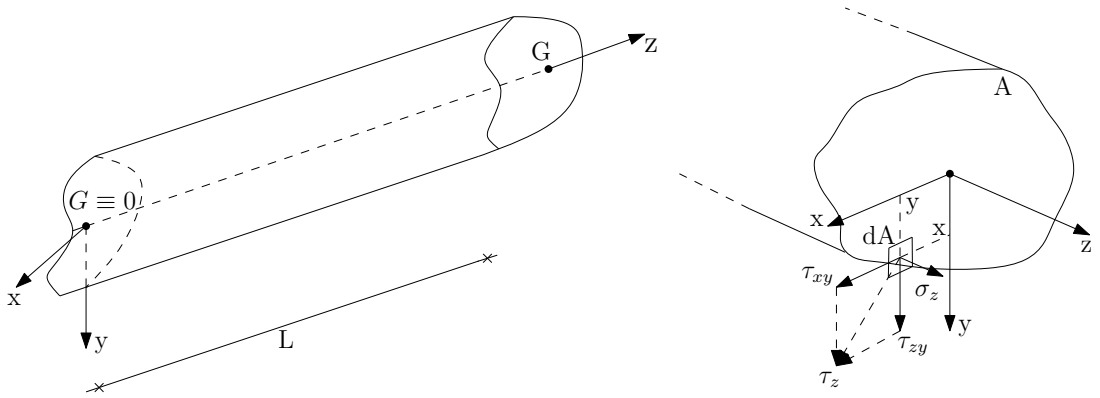


Figure 14: *Rectilinear beam with constant section A (left) and the stress vector* $t_z = \{\tau_{zx}, \tau_{zy}, \sigma_z\}^T$ *at a generic point of the beam section A (right) [52]*

The resultants, i.e., internal actions of the section dA are:

$$\begin{aligned} N &= \int_A \sigma_z dA \\ T_x &= \int_A \tau_{zx} dA \\ T_y &= \int_A \tau_{zy} dA \\ M_x &= \int_A y\sigma_z dA \\ M_y &= -\int_A x\sigma_z dA \\ M_t &= \int_A (x\tau_{zy} - y\tau_{zx}) dA \end{aligned} \tag{3.1}$$

where N is the normal force, T_x and T_y are the shear forces along x and y, M_x and M_y are the bending moments around x and y, and M_t is the torque moment. These resultants give the internal actions as a function of stresses τ_{zx}, τ_{zy} and σ_z; solving the D.S.V. problem allows this relationship to be reversed, returning stresses as a function of internal actions. However, this solution considers only homogeneous and isotropic beams, and to obtain the solution for inhomogeneous and anisotropic beams, it is necessary to proceed with the semi-inverse solution [53].

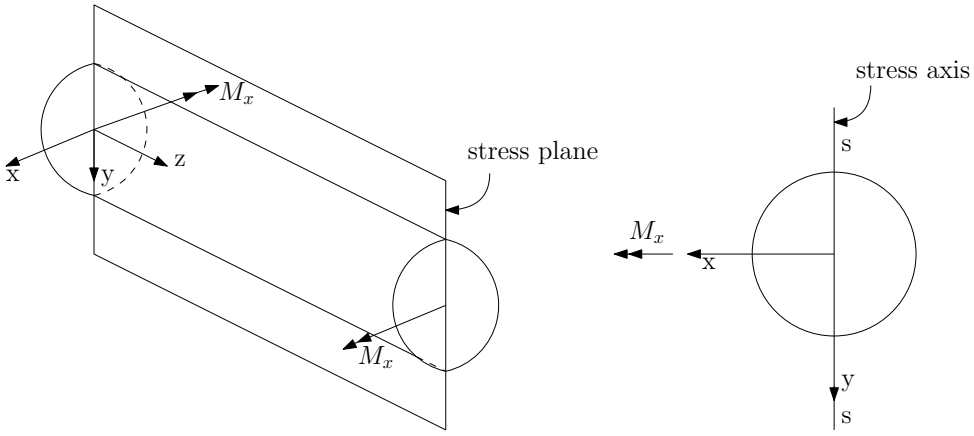


Figure 15: *Stress plane and axis of the beam [52]*

Thanks to this solution, it is possible to study the bending case of a flexible beam. The moment vector (M_x) is pointed as a main direction of inertia of the section and it is perpendicular to the stress plane (y,z); the intersection between the stress plane and the section of the beam (y) is called stress axis (s) (Fig.15).

In addition, if a beam is flexed with two equal and opposite moments at the end, the midsection translate and rotate without bending (Fig.16); in that case, a rigid motion is added in the origin (z = 0) to guarantee the framing condition in the origin. This property of the mid-section of the beam is related to the Navier name and is the basis of the Timoshenko beam model [54]. However, if it is assumed that the section remains also perpendicular to the deformed axis of the beam, then the Euler-Bernoulli theory [55] must be considered.

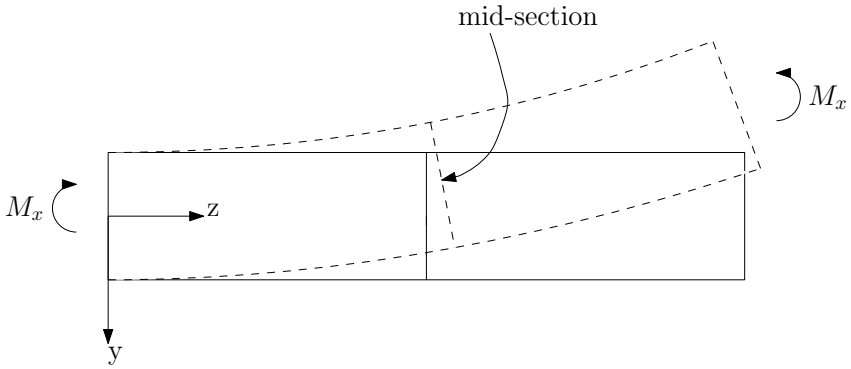


Figure 16: *Example of flexed beam [52]*

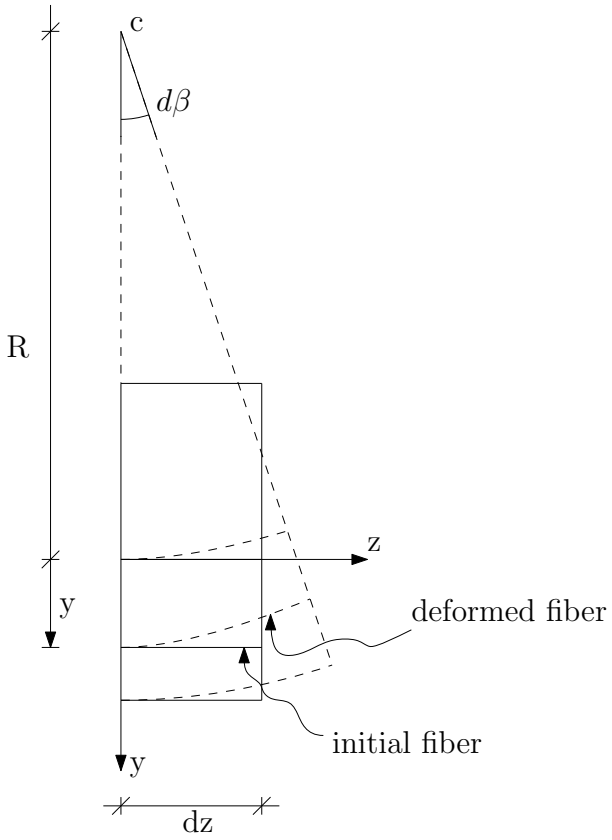


Figure 17: *Deformed beam block [52]*

Given a deformed beam block as in Fig.17, the initial length of the fiber y is $l_i = dz = Rd\beta$, while its length after the deformation becomes $l_f = (R + y)d\beta$. So the beam deformation is:

$$\varepsilon_z = \frac{l_f - l_i}{l_i} = \frac{(R + y)d\beta - Rd\beta}{Rd\beta} = \frac{y}{R} = \kappa_x y \tag{3.2}$$

where R is the bending radius of curvature and $\kappa_x = \frac{1}{R}$ is the curvature.

In addition, by combining the beam deformation of the semi-inverse solution ($\varepsilon_z = \frac{\sigma_z}{E}$) with Eq.3.2, it is possible to obtain the normal tension of the deformed beam:

$$\sigma_z = E\kappa_x y, \quad \text{while} \quad \tau_{xy} = \tau_{zx} = 0 \tag{3.3}$$

where E is Young's Modulus of Elasticity.

Hence, $T_x = T_y = M_t = 0$, so the internal actions (Eq.3.1) become:

$$
\begin{aligned}
N &= \int_A \sigma_z dA = E\kappa_x \int_A y dA = E\kappa_x S_x = 0 \\
M_x &= \int_A y\sigma_z dA = E\kappa_x \int_A y^2 dA = E\kappa_x I_x \\
M_y &= -\int_A x\sigma_z dA = E\kappa_x \int_A xy dA = E\kappa_x I_{xy} = 0
\end{aligned}
\tag{3.4}
$$

At this point, from Eq.3.4, it is possible to get the curvature as:

$$
\kappa_x = \frac{1}{R} = \frac{M_x}{EI_x} \tag{3.5}
$$

which, within Eq.3.3, returns the Navier formula:

$$
\sigma_z = \frac{M_x}{I_x} y \tag{3.6}
$$

where I_x is the second moment of area.

This relationship describes the tensional state as a function of internal action.

Furthermore, Eq.3.5 shows that if a beam is rigid (high EI_x), it is difficult to bring it to the curvature because a higher bending moment (M_x) would be required.

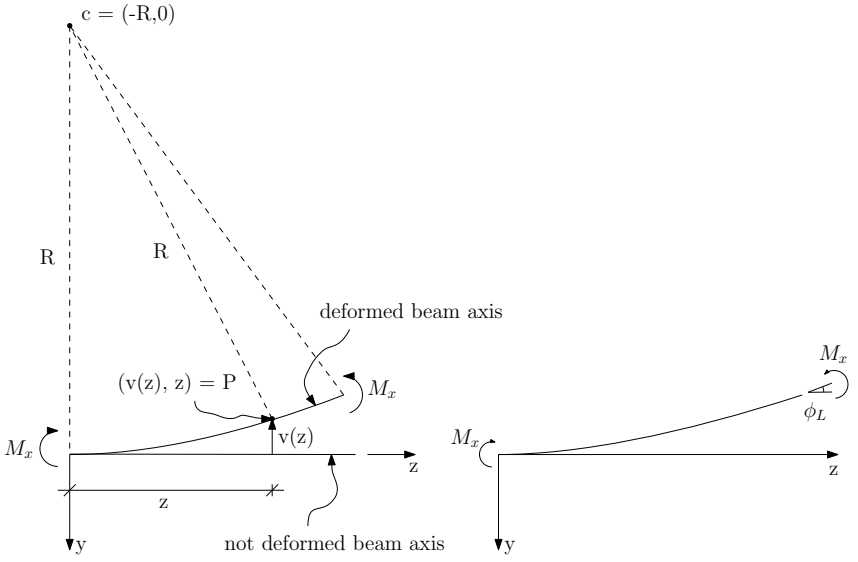


Figure 18: *Deformed beam axis [52]*

The final step is to analyze the deformation of the beam axis (Fig.18), which performs a transversal displacement $v(z)$ equal to:

$$v(z) = -\frac{z^2}{2R} = -\frac{1}{2}\kappa_x z^2 = -\frac{1}{2}\frac{M_x}{EI_x}z^2 \tag{3.7}$$

The inclination angle $\phi(z)$ is obtained by deriving Eq.3.7 and it is considered positive if counterclockwise:

$$\phi(z) = -v'(z) = \kappa_x z = \frac{M_x}{EI_x}z \tag{3.8}$$

In conclusion, the final angle of deformation is:

$$\phi_L = \frac{M_x L}{EI_x} \tag{3.9}$$

where $z = L$ is the length of the beam axis.

The number $\frac{EI_x}{L}$ is the bending stiffness of the beam and describes the moment required to have a rotation, while the number $\frac{L}{EI_x}$ is the bending deformability of the beam and describes the rotation corresponding to a uniform moment [52].

3.2 Lumped-parameter and Flexible-beam method description

According to the flexible-beam method, the bending and axial deformations of a flexible beam follow the classical Euler-Bernoulli beam theory: bending can occur about any axis in the plane of the cross-section (xy plane) of the beam. Cross-sections are assumed to be rigid in plane, to remain planar during deformation, and to always be perpendicular to the deformed neutral axis of the beam. The torsion of the beam is derived from De Saint Venant torsion theory, and the cross sections are rigid in plane but free to deform out of plane.

In Simscape Multibody, the implementation of the flexible beam method consists of using the *Flexible Rectangular Beam* block [56]. In this block, flexible beams are assumed to consist of a homogeneous, isotropic and linearly elastic material. Furthermore, they are assumed to be thin bodies whose length must far exceed the cross-sectional dimension, and all deformations are assumed to be linear and small. Material properties, such as density and Young's Modulus of Elasticity can be specified by the user, while the beam cross-sectional properties, such as axial, flexural, and torsional rigidities, are automatically calculated by the block based on the specified geometry and material properties. The "Number of Elements" (NOE) parameter within the beam block specifies the number of finite elements used to discretize the beam. Its value can be specified by the user to achieve a good compromise between simulation accuracy, which may require more elements, and simulation speed, which requires fewer elements. In addition, for bending deformations, the beam blocks use the cubic Hermite interpolation method [57] to compute the displacement distributions throughout each element, while the distributions of axial displacement and torsional rotation are obtained by linear interpolation method.

On the other hand, the lumped-parameter method [58] treat the flexible body as a collection of discrete flexible units fixed to one another. According to this method, each flexible unit consists of two rigid mass elements connected through joints with internal springs k and dampers b (Fig.19). The joints provide the necessary degrees of freedom for deformation: rotational degrees of freedom enable bending and torsion, while translational degrees of freedom enable axial deformation and shear. The mass, spring and damper elements provide the inertial, restoring and dissipating forces that, collectively, cause the deformation. The accuracy of the lumped-parameter method depends in part on the number of mass elements used: increasing this number tends to improve the accuracy obtained, but at a higher computational cost.

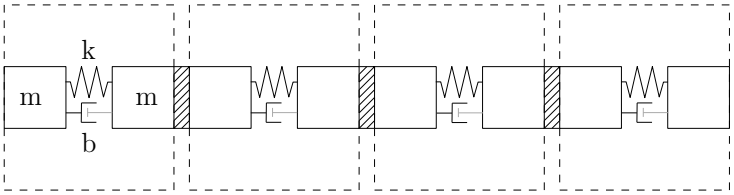


Figure 19: *Example of a flexible body with 4 flexible units, according to the lumped-parameter method [50]*

3.2.1 Response to an impulse

To properly study the behavior of flexible bodies, each method is applied in a Simulink model and their response to an external impulse is analyzed. The impulse response of a dynamic system is the reaction of a dynamic system in response to an external load and describes the reaction of the system as a function of time. To compare the frequencies of the lumped-parameter and flexible-beam methods, the response data and time are used to compute the "Fast Fourier Transform" (FFT) of each method.

In general, for continuous periodic functions of time $f(t)$ it is common to use the Fourier series representation of the function:

$$f(t) = \sum_{n=-\infty}^{\infty} F(n) e^{jn2\pi f_1 t} \tag{3.10}$$

The period of the function is $T_1 = 1/f_1$ and $F(n)$ is the complex Fourier coefficient, given by

$$F(n) = \frac{1}{T_1} \int_{-T_1/2}^{T_1/2} f(t) e^{-jn2\pi f_1 t}\, dt \tag{3.11}$$

Furthermore, another representation for continuous aperiodic functions of time $x(t)$ is given by the integral

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{j\omega t}\, d\omega \tag{3.12}$$

where $\omega = 2\pi f$. So the complex continuous frequency spectrum of the aperiodic function

$S(f)$ is expressed by:

$$S(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt \tag{3.13}$$

Eq.3.12 and its reciprocal Eq.3.13 are the Fourier transform pair and $S(f)$ is referred to as the Fourier transform of $x(t)$.

Since the Fourier transform is a powerful tool for analysis, research was undertaken to establish techniques for numerical calculation of the Fourier transform. The "Discrete Fourier Transform" (DFT) thus evolved but was considered impractical until the development of the Fast Fourier Transform (FFT) [59].

Hence, the FFT is an optimized algorithm for implementing the Discrete Fourier Transform that converts a signal into individual spectral components and thus provides information about the frequency of the signal. In the sample-data case, the Fourier transform pair given in Eq. 3.12 and Eq. 3.13 for N samples becomes

$$S(f_n) = \Delta t \sum_{k=0}^{N-1} x(t_k)e^{-j2\pi f_n t_k}, \quad n = 0, \pm 1, ..., \pm N/2 \tag{3.14}$$

$$x(t_k) = \Delta f \sum_{n=-N/2}^{N/2} S(f_n)e^{j2\pi f_n t_k}, \quad k = 0, 1, ..., N-1 \tag{3.15}$$

However, if $t_k = k\Delta t$, $f_n = n\Delta f$, where $\Delta t = T/N$ and $\Delta f = 1/T$, Eqs. 3.14 and 3.15 becomes [60]:

$$S(n) = \Delta t \sum_{k=0}^{N-1} x(k)e^{-2\pi j(nk)/N}, \quad n = 0, ..., N-1 \tag{3.16}$$

$$x(k) = \Delta f \sum_{n=0}^{N-1} S(n)e^{2\pi j(nk)/N}, \quad k = 0, ..., N-1 \tag{3.17}$$

where $S(n)$ is the nth coefficient of the Discrete Fourier transform, $x(k)$ denotes the kth sample of the time series consisting of N samples and $j = \sqrt{-1}$. The values of $x(k)$ can be complex numbers, while the numbers of $S(n)$ are always complex. For convenience of notation, Eq. 3.16 is often written as:

$$S(n) = \sum_{k=0}^{N-1} x(k) W^{nk}, \quad n = 0, ..., N-1 \tag{3.18}$$

where

$$W = e^{-2\pi j/N} \tag{3.19}$$

The index n is called the "frequency" of the DFT since the values of $x(k)$ are often of a function at discrete time points.

In conclusion, the FFT is an algorithm that allows the DFT of a time series to be calculated faster than other available algorithms: it takes advantage of the fact that the computation of the coefficients of the DFT can be done iteratively, which saves considerable computation time and also substantially reduces round-off errors associated with these computations.

3.2.2 Effects of the discretization on the beam response

In general, discretization is the process of transferring continuous functions, models, variables and equations into discrete counterparts. This process is usually performed as a first step to make them suitable for numerical evaluation and implementation.

The lumped-parameter method discretizes the body into flexible units, each comprising two mass elements coupled through a joint with an internal spring and internal damper. The accuracy of this method is largely dependent on the number of flexible units used. Indeed, accuracy improves as the number of flexible units increases. Similarly, the flexible-beam method improves in accuracy when the "Number of elements" (NOE) increases. This number is located in the Discretization section of the Flexible Rectangular Beam block and specifies the number of finite elements used to discretize the beam.

The discretization must be modified to achieve a good trade-off between simulation accuracy, which may require a larger number of elements, and simulation speed, which requires a smaller number of elements. For both methods, a larger number of elements

increases the computational cost and slows down the speed of the simulation. Also, at some point, the increase in accuracy is negligible when there are many elements.

Whenever a model is discretized, there is always a certain amount of *discretization error*. The goal is to reduce this amount to a level that is considered negligible for modeling purposes. In numerical analysis and simulation, discretization error is the error resulting from the fact that a function of a continuous variable is represented by a finite number of evaluations, such as on a grid [61]. Discretization error can usually be reduced by using a more finely spaced grid, that is, by reducing the simulation step size with an increase in computational cost.

Furthermore, in signal processing, the discretization error can be related to aliasing, which is the effect of the appearance of new frequencies in the sampled signal after reconstruction, which were not present in the original signal. In the analysis of the impulse response of a dynamical system, sampling is lossless if the conditions of the *Sampling Theorem* are met. This theorem specifies the minimum sampling frequency at which a continuous-time signal must be uniformly sampled so that the original signal can be completely recovered or reconstructed from these samples only. This condition is satisfied if the sampling frequency is at least twice as high as the highest frequency component of the analog signal to be sampled [62]. The condition is described as:

$$f_s \geq 2f_{max} \tag{3.20}$$

where f_s is the sampling rate (how many samples of a continuous signal is taken per second) and f_{max} is the maximum-frequency component of the analog signal to be sampled. The minimum sampling rate allowed by the sampling theorem is $f_s = 2f_{max}$ and it is called the *Nyquist rate* [63]. However, if the condition of the sampling theorem (Eq. 3.20) is not satisfied, the signal reconstructed from the samples is different from the original continuous signal and results in a permanent loss of signal information. This undesirable effect is "aliasing" and is caused by too low sampling rate for a particular signal or by

too high frequencies present in the signal for a particular sampling rate. An example of a signal sampled at different rates is shown below.

Fig. 21 shows the original cosine signal of 40 Hz (Fig. 20) sampled at three different rates:

- Fig. 21a, provides an accurately sampled signal, which preserves the shape of the original signal. In fact, the sampling frequency of 200 Hz is well above 80 Hz, which is twice the cosine wave frequency.

- Fig. 21b, uses a sampled rate equal to the *Nyquist rate*. This sampled signal is at the limit of reconstructability.

- In Fig. 21c, aliasing is expected to occur. In fact, the signal is sampled at too low a rate to be reconstructed accurately. Therefore, it is no longer possible to recover the original cosine wave from these sampled points.

Aliasing can be avoided by using a sufficiently large sampling rate or by filtering the signal with a low-pass filter before sampling to ensure the condition $f_{max} < f_s/2$.

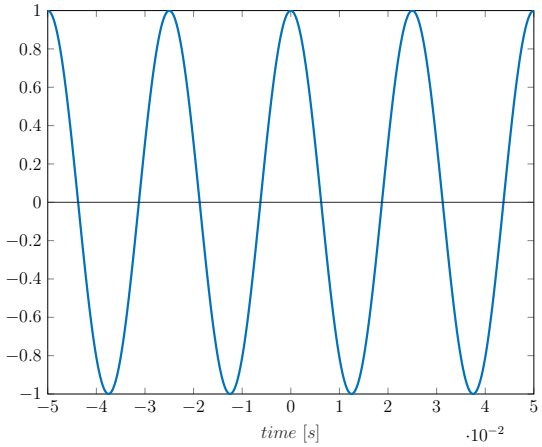
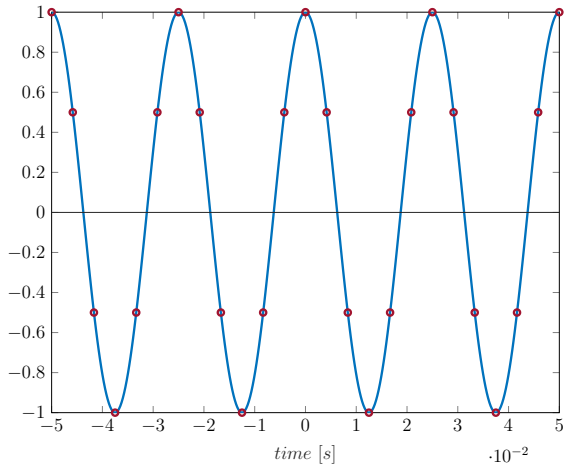
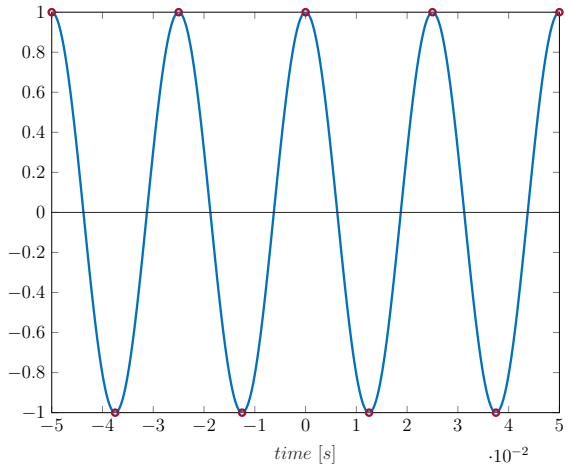


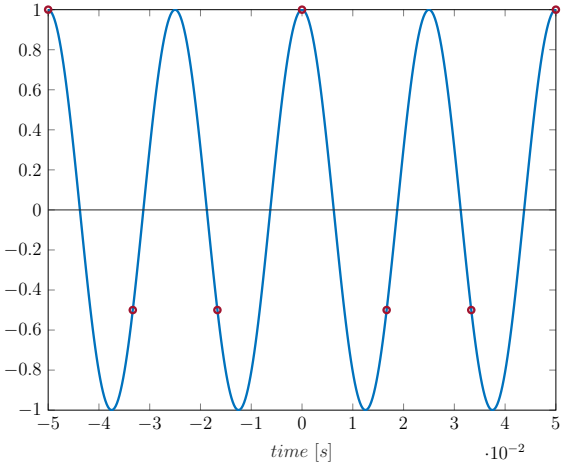
Figure 20: *Continuous-time cosine signal with a frequency of* 40 *Hz*



(a) $f_s = 5f_{max} = 200Hz$



(b) $f_s = 2f_{max} = 80Hz$



(c) $f_s = 1.5f_{max} = 60Hz$

Figure 21: *Signal in Fig. 20 sampled at three different rates*

3.3 Application of lumped theory on models

A complete lumped flexible body (Fig. 22) when subjected to purely cross-sectional forces behaves as a damped harmonic oscillator with a rotational degree of freedom provided for each joint. This behaviour is described by the equation of motion [50]:

$$I\ddot{\theta} + b_R\dot{\theta} + k_R\theta = \tau \tag{3.21}$$

where τ is the external torque acting on the rotating mass element and θ is the rotational displacement of that mass element from its equilibrium position.

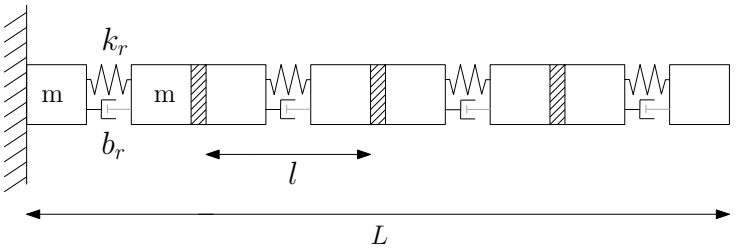


Figure 22: *A lumped flexible body with 4 rotational degrees of freedom [50]*

The parameters I, b_R and k_R are, respectively, the moment of inertia of a mass element about the axis of rotation, the rotational damping coefficient of the joint, and the rotational elastic coefficient of the joint.

In addition, since it is an harmonic system, it is possible to introduce two new parameters: the damping ratio (ζ) and the natural frequency (ω). The damping ratio is a system parameter that describes how quickly the oscillations decay from one bounce to the next: it can range from undamped ($\zeta = 0$), underdamped ($\zeta < 1$), critically damped ($\zeta = 1$) and overdamped ($\zeta > 1$). The natural frequency is the rate at which an undamped oscillator tends to vibrate in the absence of an applied load. Expressing the rotational equation (Eq. 3.21) in terms of these parameters gives:

$$I(\ddot{\theta} + 2\zeta\omega\dot{\theta} + \omega^2\theta) = \tau \tag{3.22}$$

where

$$\zeta = \frac{b_R}{2\sqrt{Ik_R}} \qquad \text{and} \qquad \omega = \sqrt{\frac{k_R}{I}}$$

$2\sqrt{Ik_R}$ is called critical damping coefficient, which is the value of damping at which the oscillator switches between underdamping and overdamping conditions.

According to the lumped-parameter method, the flexible body is divided into flexible body units, the length of which is equal to:

$$l = \frac{L}{N} \tag{3.23}$$

where L is the total length of the flexible body and N is the number of discretizations (flexible body units) used to describe the model.

The external load is chosen to achieve at least 1 cm deformation of the body end (Fig. 33), while the second moment of area I_A is calculated from the standard analytical expression for a rectangular cross section:

$$I_A = \frac{WT^3}{12} \tag{3.24}$$

where W and T are respectively the width and the thickness of the body.

Considering the case of a beam subjected to a deflection angle ϕ, Eq. 3.5 gives the bending moment on a continuous beam unit:

$$M = \frac{EI_A}{R} \tag{3.25}$$

where E is Young modulus of elasticity and R is the bending radius of curvature. However, if very small deflections are considered, ϕ reduces to $\frac{l}{R}$ and the spring coefficient k_R of the rotary joint can be calculated by the bending formula:

$$k_R = \frac{EI_A}{l} \tag{3.26}$$

The damping coefficient b_R of the revolute joint is taken as linear and proportional to the spring coefficient k_R and calculated as

$$b_R = \alpha k_R \tag{3.27}$$

where α is a proportional constant derived empirically and related to the discretizations level of the beam. The scaling of the damping coefficient is provided by the spring coefficient as calculated Eqs. 3.23 and 3.26: if the number of flexible beam units increase, the damping coefficient increases also.

Solar panel and robotic arm models implemented using the lumped-parameter method are realized within Simscape Multibody. Dimensions and material properties are summarized in Chapters 5.1 and 5.2, while the Simscape diagram of the flexible body unit used for both models is presented in Chapter 4.3.2.

3.3.1 Comparison against the benchmark

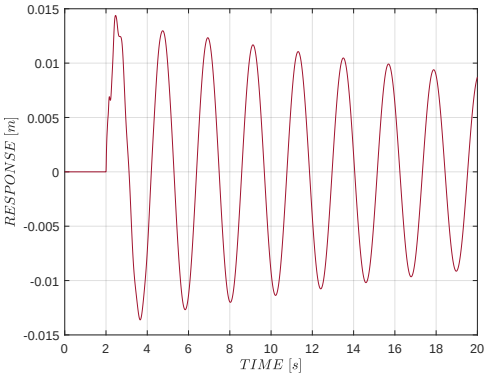
Since the flexible-beam method follows the classical theory of flexible beams (Chapter 3.1.1) and is modeled with a block, created by Simulink, that automatically computes properties related to beam deformation, it was considered as a "Benchmark" against the lumped-parameter method for their evaluation and comparison in all simulations.

For example, this property was also used to calibrate the proportional constant α for the lumped model: the damping coefficient b_R (Eq. 3.27) of the revolute joint depends on the constant α, which is related to the level of discretization of the beam. Since α is empirical, it was calibrated by the following method:

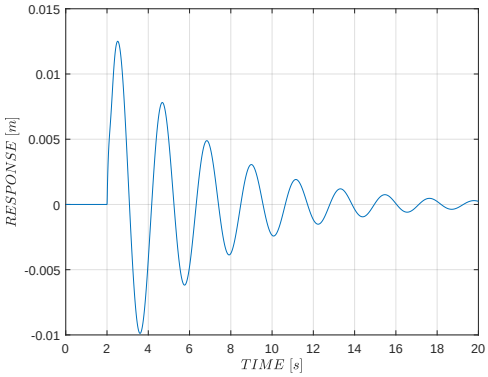
1. At the end of the simulation, each model generates a graph of the response displacements (Figs. 23a and 23b);

2. The graph of the lumped model is compared with that of the flexible beam, which is considered as the reference graph (Fig. 23c);

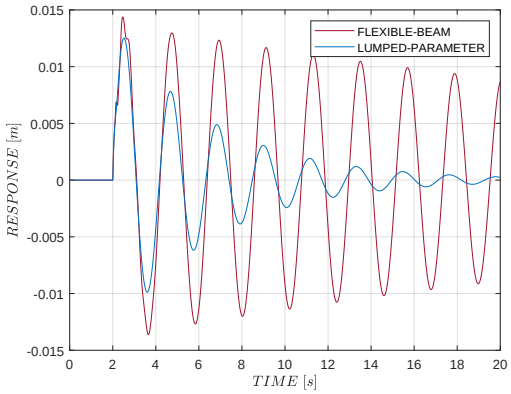
3. Since α directly affects the damping coefficient of the lumped graph, its value is changed until the discrepancy between the two graphs is less than a certain percentage (Figs. 24c).



(a) Flexible-beam model displacement



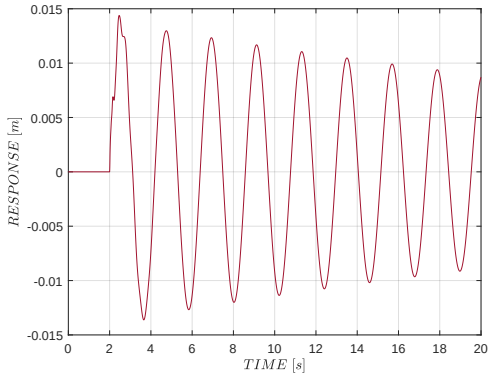
(b) Lumped-parameter model displacement with $\alpha = 0.05$



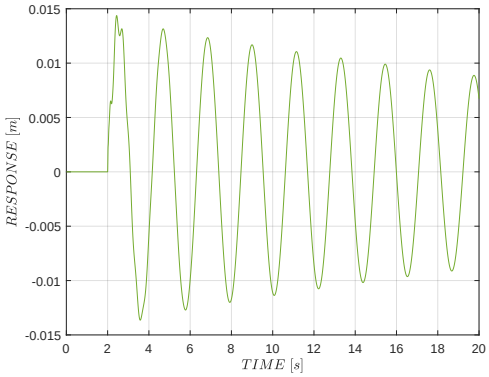
(c) Lumped-parameter model vs Flexible-beam model displacement

Figure 23: *Comparison between models with* $\alpha = 0.05$

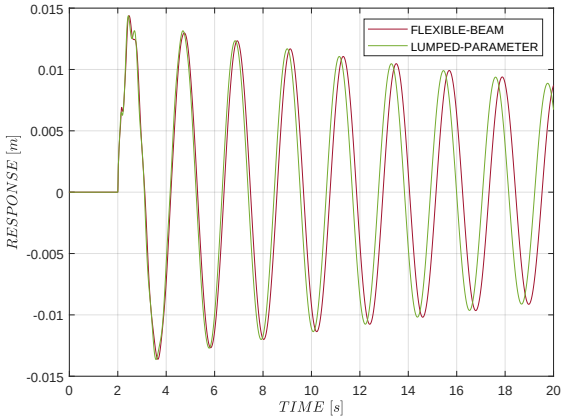
Figs. 23 and 24 show the comparison between the models used to calibrate the constant α. The main difference is that the response of the two lumped models (Figs. 23b and 24b) changes depending on the value of the α constant: with a smaller α, the response is less damped. This attitude is confirmed by the values of the damping coefficient b_R of the two lumped models, which are: $b_R = 1.41 \times 10^3\,\mathrm{N\,s/m}$ with $\alpha = 0.05$ and $b_R = 141.129\,\mathrm{N\,s/m}$ with $\alpha = 0.005$.



(a) Flexible-beam model displacement



(b) Lumped-parameter model displacement with $\alpha = 0.005$



(c) Lumped-parameter model vs Flexible-beam model displacement

Figure 24: *Comparison between models with* $\alpha = 0.005$

3.4 Solver definition

To simulate a dynamic system, its states are calculated at successive time steps over a specified time frame. Time steps are time intervals in which the computation takes place: the size of this time interval is called the step size. The process of computing the states of a model in this manner is known as *solving* the model and this calculation uses the information provided by the model. A wide variety of numerical integration techniques for solving the ordinary differential equations (ODEs) are available in the literature these

equations represent the continuous states of dynamic systems.

Simulink provides a set of solvers, which applies a numerical method to solve a set of ordinary differential equations representing the model. Through this calculation, it determines the time of the next simulation step. In the process of solving the initial value problem, the solver also meets the accuracy requirements specified by the user. A solver can be continuous or discrete. Continuous solvers use numerical integration to calculate the continuous states of a model at the current time step based on the states of previous time steps and state derivatives and rely on individual blocks to calculate the values of the discrete states of the model at each time step. On the other hand, discrete solvers are mainly used to solve purely discrete models: they compute only the next simulation time step for a model. When they perform this calculation, they rely on each model block to update its discrete state. Discrete solvers do not compute continuous states.

Continuous solvers are also divided into fixed-step and variable-step solvers, each of which implements a specific method of solving a specific ODE solution method [64]:

- Fixed-step solvers solve the model using the same step size from the beginning to the end of the simulation and the step size can be specified. In general, decreasing the step size increases the accuracy of the results and the time required to simulate the system.

- Variable step solvers vary the step size during simulation. These solvers reduce the step size to increase the accuracy at certain events during model simulation, such as rapid state changes or zero crossing events. They also increase the step size to avoid taking unnecessary steps when model states change slowly. Calculating the step size increases the computational overhead at each step. However, it can reduce the total number of steps and thus the simulation time required to maintain a given level of accuracy for models with zero crossings, rapidly changing states, and other events that require additional computation.

Since this study is on purpose to contribute to a project at the University of Padua in collaboration with ESA, all models in this Master's thesis are simulated with a fixed-step continuous solver. This is a requirement of ESA in order to be able to use the GNC system on computers representative of those on the mission.

All solvers provided by MATLAB - Simulink follow a similar naming convention: ode, followed by two or three numerals indicating the orders of the solver. Some solvers can solve $stiff$ differential equations and the methods used by them are expressed by the s, t, or tb suffixes. Fig. 25 and Table 7 shows how Simulink classifies solvers.

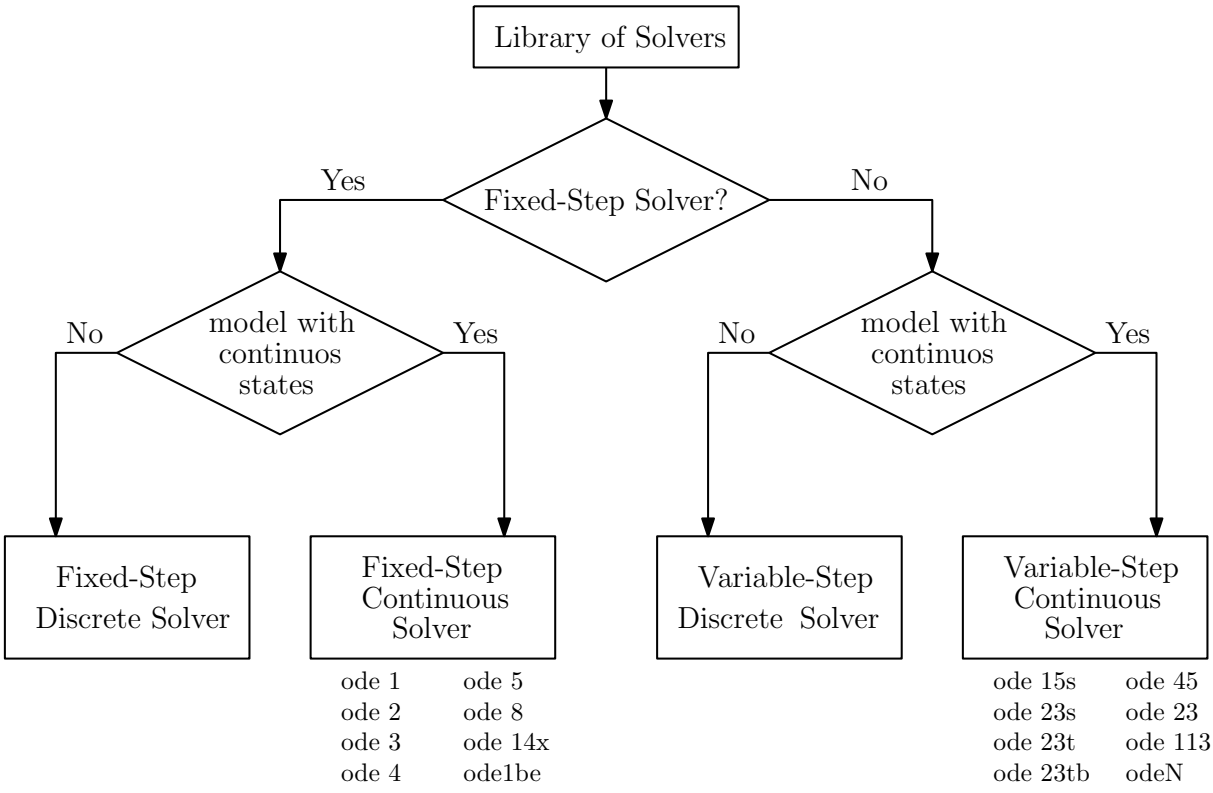


Figure 25: *Classification of solvers in the Simulink library [64]*

Table 7: *Classification of continuous solvers type*

Type of solver	Fixed-step solvers	Variable-step solvers
explicit	ode8	ode45
explicit	ode5	ode23
explicit	ode4	ode113
explicit	ode3	odeN
explicit	ode2	
explicit	ode1	
implicit	ode14x	ode23s
implicit	ode1be	ode23t
implicit		ode23tb
implicit		ode15s

3.4.1 Explicit vs Implicit fixed-step solvers

Another property of continuous solvers is that they can be explicit or implicit (Table 7). An explicit system is represented by the equation:

$$\dot{x} = f(x) \tag{3.28}$$

where for any given value of x, $\dot{x}$ is computed by substituting x in $f(x)$ and evaluating the equation. In general, all fixed-step explicit solvers calculate the next step as:

$$X(n+1) = X(n) + h \cdot dX(n) \tag{3.29}$$

where X is the state, h is the step size, n indicates the current time step and dX is a solver-dependent function that estimates the state derivative. $dX(n)$ is calculated by a particular algorithm using one or more derivative evaluations depending on the order of the method.

On the other hand, an implicit system is represented by the equation:

$$F(\dot{x}, x) = 0 \tag{3.30}$$

where for any value of x, the equation must be solved to calculate $\dot{x}$.

Implicit solvers offer greater stability for oscillatory behavior and they are more efficient than explicit solvers for solving a linearly implicit system. However, implicit solvers are more computationally expensive: at each time step they generate a Jacobian matrix and solve the set of algebraic equations by a Newton-like method. To reduce this additional cost, implicit solvers offer a *Solver Jacobian method* parameter to improve the simulation performance of implicit solvers. There are five options for computing the solver Jacobian method: "auto", Sparse perturbation, Full perturbation, Sparse analytical and Full analytical [65]; the default setting is "auto", which causes Simulink to determine which of the remaining four methods best suits the model, depending on the Number of simulation states and if an analytical Jacobian method is available for all blocks.

Moreover, implicit solvers are specifically designed to solve stiff problems: an ordinary differential equation problem is said to be stiff if the desired solution varies slowly, but there are closer solutions that vary rapidly. The numerical method must therefore take small time steps to solve the system. Stiffness is an efficiency problem and the more stiff a system is, the longer it takes the explicit solver to perform a calculation.

Table 8 shows the different integration techniques used to implement all Simulink fixed-step solvers, where the last two solvers are implicit. $ode14x$ uses a combination of Newton's method and extrapolation from the current value to compute the model state at the next time step, as an implicit function of the state and the state derivative at the next time step:

$$X(n+1) - X(n) - h \cdot dX(n+1) = 0 \tag{3.31}$$

It is possible to specify the number of iterations of Newton's method and the order of extrapolation that the solver uses to compute the next value of a model state: the greater

the number of iterations and the order of extrapolation selected, the greater the accuracy obtained. However, this simultaneously results in a greater computational burden for each step. Although it requires more calculations per step than an explicit solver, $ode14x$ is more accurate for a given step size.

The last one is the $ode1be$ solver: a *Backward Euler* type solver [66], that uses a fixed number of Newton iterations and runs into a fixed cost. The $ode1be$ solver is a computationally inexpensive fixed-step alternative to the $ode14x$ solver.

Table 8: *List of fixed-step solvers [67]*

Solver	Integration Technique	Order of accuracy
ode1	Euler's method	First
ode2	Heun's method	Second
ode3	Bogacki-Shampine formula	Third
ode4	Runge-Kutta (RK4) formula	Fourth
ode5	Dormand-Price (RK5) formula	Fifth
ode8	Dormand-Price RK(7)	Eight
ode14x	Newton's method and extrapolation	to be specified
ode1be	Backward Euler	to be specified

4. The Simulation Tool

This chapter provides a description of the entire simulation tool used in this thesis. After a general introduction to the tool structure, a description of the MATLAB scripts that complement Simulink models is provided. Then, an in-depth presentation of these models, shows how the structures of the solar panel and the robotic arm were implemented in Simulink.

4.1 The MATLAB - Simulink environment

The Simulation tool structure consists of two main segments. The first one is implemented in the MATLAB environment and consists in a script which calls several functions in order to initialize the parameters required during the simulation. The second part is implemented in the Simulink environment and performs the numerical integration of the system dynamics.

The objectives of the simulation tool to properly analyze the solar panel and robotic arm can be summarized as follows:

1. implement different models of flexible body methods, specifically the lumped-parameter and flexible-beam methods;

2. reproduce the response of these spacecraft elements when subjected to external loads (solar panel) and during their movement from one point to another (robotic arm);

3. allow comparison between flexible body methods.

Fig. 26 shows the structure of the Simulation tool used to study the solar panel. The Simulink environment consists of two models to study the problem by two different methods. The first one is called "lumped_parameter.slx" in which the model presented in Chapter 3.2 is implemented. The second one is called "flexible_beam.slx" in which the *Flexible Rectangular Beam* block [56] is used to study the behaviour of the solar panel. Each Simulink model is called by the script "launcher.m", which defines the initial parameters of the simulation. Once the outputs have been generated, they are post-processed to perform the FFT (Fast Fourier Transform) [68] of the two models so that the resulting responses from the two methods can be compared.

To achieve this goal for the *solar panel*, the Simulation tool must:

(a) implement each model within Scenario 1 and 2, described in Chapters 2.2.1 and 2.2.2;

(b) simulate the panel displacement caused by the system response to an external load and plot its data and time;

(c) use these information to create a FFT graph for each model.

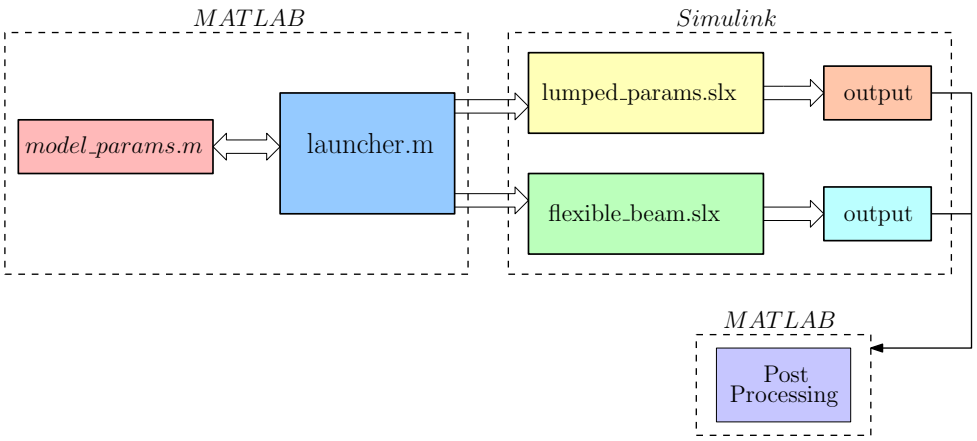


Figure 26: *Simulation tool structure to study the solar panel*

On the other hand, three different MATLAB - Simulink models are used to study the behavior of the *robotic arm* (Fig. 30): two of them implement lumped-parameter and flexible-body methods to the robotic arm links, which are considered as flexible bodies, while the third model implements the robotic arm links as rigid bodies. Since the robotic arm is a 3 DOF planar arm (Fig. 31), it is described in two dimensions (one plane).

Each model describes the robotic arm motion, which must:

(a) track a rectilinear path from the starting point to the final one;

(b) have initial and final velocity and acceleration equal to zero;

(c) employ smooth acceleration and velocity profiles to define the tool motion.

4.2 MATLAB segment

Solar panel

The main script ("launcher.m") that starts the simulation (Fig. 26) contains the system characteristics of both scenarios such as geometric features (mass and inertia), material properties (Young's modulus of elasticity, density and Poisson's ratio) and the modulus of the external load. Other values, such as the unit length of the flexible body (Eq. 3.23), lumped masses (Fig. 28b), the revolute joint stiffness (Eq. 3.26), and damping coefficient (Eq. 3.27) were considered only for the lumped model and have been evaluated in a MATLAB function called "model_params.m". After defining these inputs, the simulation is launched.

The second part of the script evaluates data and time plotted by each model and generates two different Bode diagrams, one for each Fast Fourier Transform (FFT) [68].

Robotic Arm

Robotic arm models are launched from the Simulink segment, while the MATLAB script merely defines their parameters. It also defines the Cartesian coordinates of the start and end position of the the robotic arm End-Effector (EE). All Simulink models contains a MATLAB function that implements the manipulator inverse kinematics to convert the arm position into rotation angles for the arm joints to assume.

4.3 Simulink segment

The two spacecraft elements studied in this Master Thesis are implemented using different Simscape Multibody models. Fig. 27 represents the main model structure of the solar panel, while Fig. 30 represents the general model structure to simulate the robotic arm motion.

4.3.1 Solar panel

The *Simulation Setup* is the block required by Simscape for setting up the simulation, which contains the *Mechanism Configuration*, the *Solver Configuration*, and the *World Frame*. External forces are defined by the *External force and Torque* block [69] that is connected to the subsystem and applies the external load to the end of the panel.

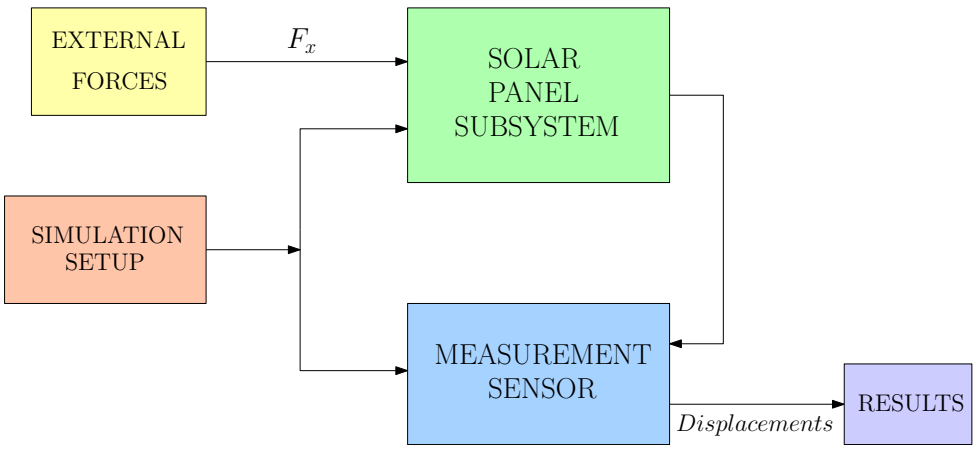
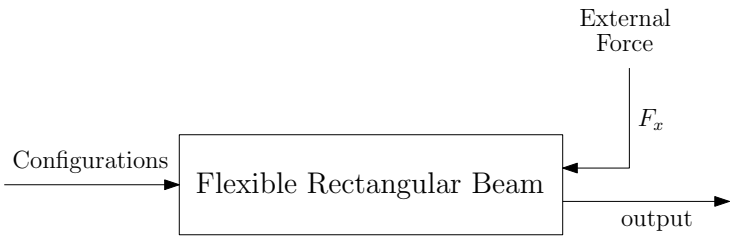


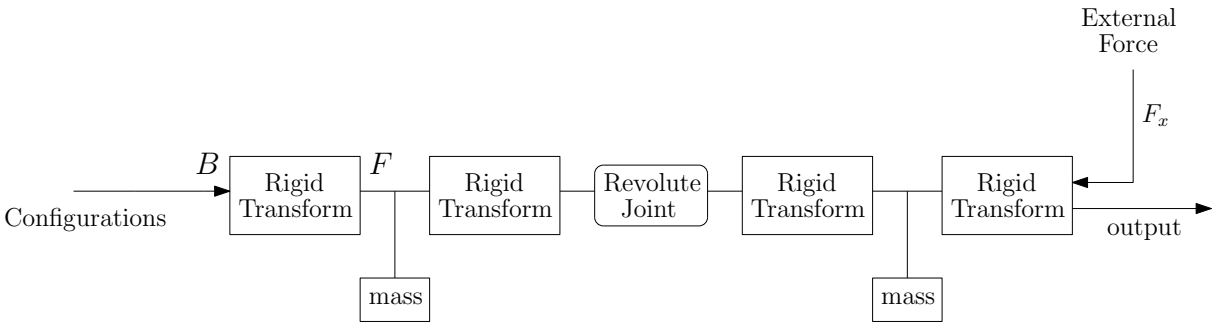
Figure 27: *Simscape model structure of the solar panel*

The solar panel subsystem implements the lumped-parameter and the flexible-beam models. Its input is the external load F_x acting at the end of the solar panel, defined in the "launcher.m". The output is the panel displacements along the axis on which the external force acts. These values are given by the *Measurement sensor*, which contains the *Transform Sensor* block [70]: this block measures the relative spatial relationship between the two ends of the panel.

Fig. 28 shows the block diagrams used within the flexible-beam model and the lumped-parameter model. The *Revolute Joint* [71] (Fig. 28b) provides internal springs k and dampers b, which gives the necessary degrees of freedom for deformation, while the *Rigid Transform* block [72] applies a time-invariant transformation between two frames. This transformation translates the frame of the follower (F) relative to the frame of the base (B) (Fig. 28b). The frames remain fixed with respect to each other during the simulation, moving only as a single unit. This translation is necessary to define the position of the revolute joint and the two mass blocks within the subsystem.



(a) Solar panel block for the flexible-beam model [56]



(b) Solar panel lumped unit for the lumped-parameter model

Figure 28: *Block diagrams for the solar panel subsystem*

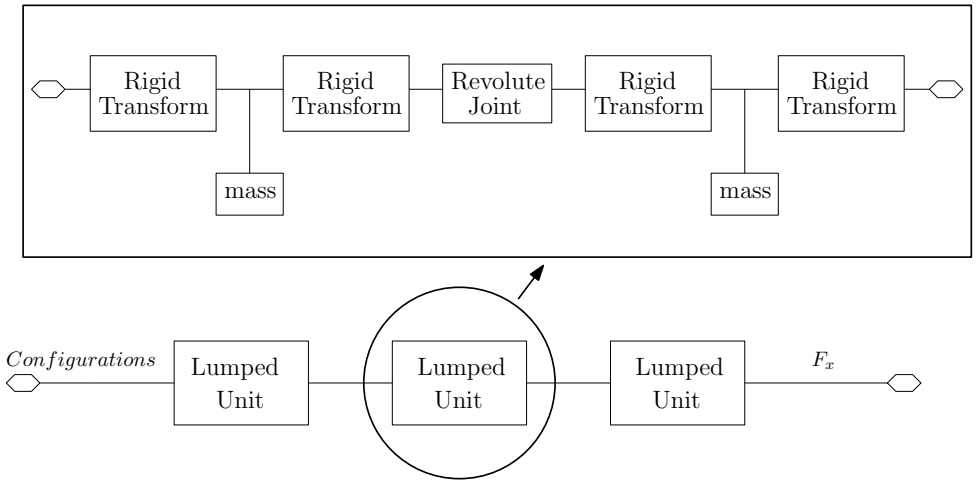


Figure 29: *Example of lumped Simscape diagram with three discretizations*

While the number of elements of the flexible beam model is chosen within the Flexible Rectangular Beam block, the discretization of the lumped model must be adjusted by adding or removing a lumped unit: each unit corresponds to a unit of Number of Elements (Fig. 29).

4.3.2 Robotic arm

As described in the Chapter 4.1, three models were created to analyze the behaviour of robotic arm links.

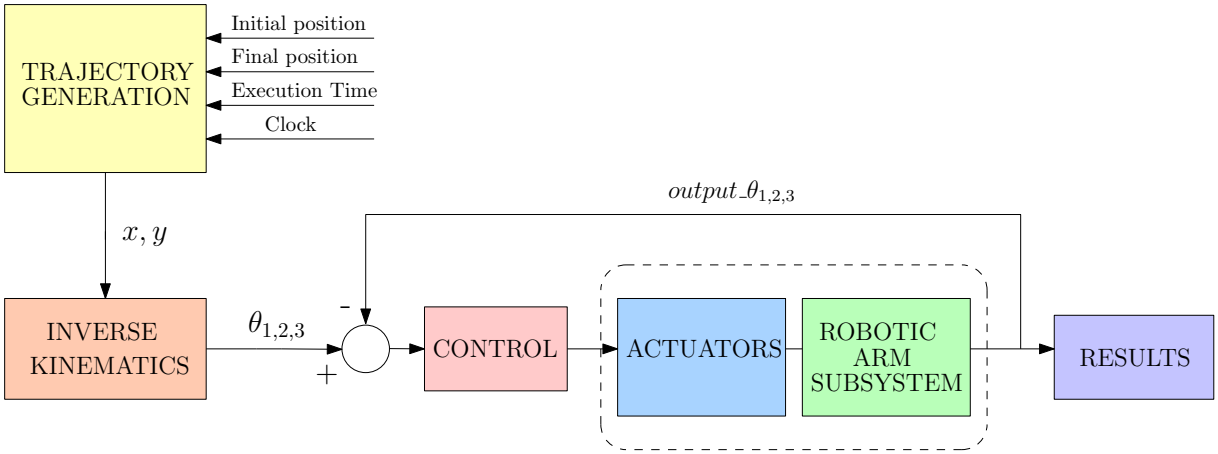


Figure 30: *Simscape model structure of the robotic arm*

Each model is divided into four main sections: the "Trajectory Generation" block, the "Inverse kinematics" block, the "Control" block and the robotic arm system. The latter is divided into two other blocks: the "Actuators" block and the "Robotic Arm Subsystem" block, which includes the flexible body method chosen to simulate the robotic arm motion. Fig. 30 shows the block diagram of the robotic arm model.

Trajectory generation

The first block is the trajectory generation, in which the trajectory is implemented by employing fifth order polynomials. The initial and final position of the EE and the manoeuvre time are specified in the Cartesian space (x, y), where the origin coincides with the first joint of the manipulator. Its task is to have the manipulator hold the EE at the desired initial position from the beginning of the simulation until the time when the maneuver begins; then, after the maneuver is executed, the EE will be held at the final position.

Inverse kinematics

Fig. 31 shows the three-link robotic arm used within this thesis. The input of this block is the motion trajectory assigned by the guide to the EE in terms of position in the Cartesian space (x, y). The goal of inverse kinematics is to determine the joint variables corresponding to the position and orientation of the EE. Solving this problem is important for transforming the motion specifications, assigned to the EE within the trajectory generation block, into the corresponding joint-space variables θ_1, θ_2 and θ_3 [73].

Control

The structure of the robotic arm model (Fig. 30) includes a "control loop" before the "control" block. This loop allows the output value of the robotic arm subsystem to be compared at each time-step with that calculated analytically from the *Inverse kinematics* block.

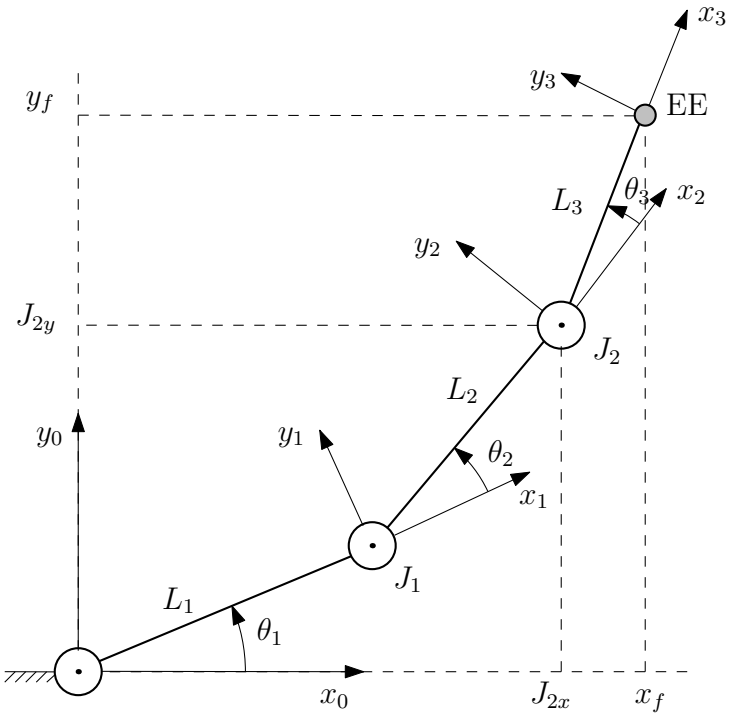


Figure 31: *Three-link planar arm [29]*

The "control" block then implements the obtained error into the Proportional-Integrative-Derivative (PID) controller described in Chapter 2.4, which is a feedback control loop mechanism implemented to control the dynamics of the robotic arm. The output of this block is the torque that the robotic arm joint shall generate to drive the movement of the $i - joint$ in the robotic arm subsystem.

Actuators and Robotic Arm Subsystem

Fig. 32 shows the block diagram of the actuator and the robotic arm subsystem. The actuator is implemented inside the revolute joint [71], which performs rotation and provides output signals for each joint (θ_i, joint velocity v_i, and torque). These signals are used as feedback to control the motion of the manipulator. In a real model, the actuator would be a DC motor. Hence, the "inertia" block represents its mass: without this block, the mass of the motor would not be considered in the model, since the revolute joint represents the action of the motor, but not its mass.

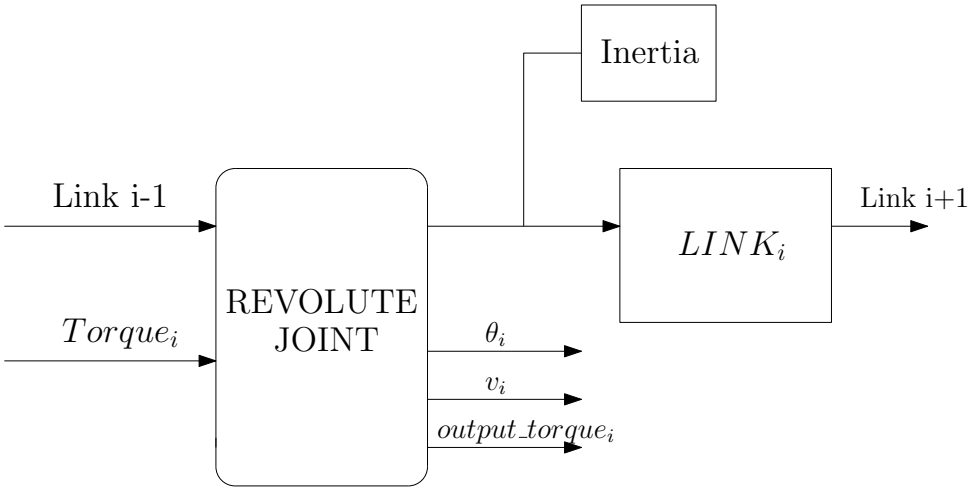


Figure 32: *Simscape representation of the revolute joint and the i link*

The "$Link_i$" represents the subsystem of the robotic arm, and, for each model, a different method is used to create its links:

- the lumped-parameter method, which uses the same techniques described in Chapter 3.3;
- the flexible-beam method, also defined in Chapter 3.2;
- the rigid-body method, where links are created with the "Cylindrical Solid" block [74]. This block generates a rigid body cylinder whose radius and height can be specified by the user.

This structure is repeated for each joint. However, since a three-link robotic manipulator is analyzed, the third link is made as a rigid body cylinder for each model. Note that this manipulator model is a simplification of the robotic manipulator described in Chapter 2.3, which consisted of seven cylindrical links, and the objective of this thesis is not to study the motion of a robotic manipulator, but to analyze how the flexible bodies of the robotic arm affect its motion.

5. Numerical results

This chapter presents the results of all the simulations performed with the simulation tool to find the best numerical method to study the flexible body behaviour, given the specific simulation requirements of this project. For this purpose a comparison between the *Lumped-parameter* and *Flexible-beam* methods is provided: firstly, the response of the solar panel subjected to an external load is studied; secondly, the motion of the robotic arm is simulated and the position error of the end-effector at each time-step is analyzed. The final part of the Chapter analyzes the Simulink solvers used to run the simulation; in particular, an explicit and an implicit fixed-step solver are compared to find the best combination of acceptable results and integration step.

To compare the *Lumped-parameter* and *Flexible-beam* methods, the two case studies of the solar panel and the robotic arm are analyzed separately. Despite this, both flexible bodies are simulated with the same Simulink solver. Each result is reported in the following sections.

5.1 Solar Panel

Table 9 summarizes the characteristics of the system. These values are independent of the discretization of the system and are considered in both lumped-parameter and flexible-beam models, while the damping ratio ζ was considered only for the flexible rectangular beam block.

The external force F_x is chosen to achieve at least 1 cm deformation of the body end (Fig. 33). The *Simulation Time* specify the stopping time of the simulation in seconds and is not equal to the *Execution Time*. For example, running a simulation with a *Simulation Time* of 10 seconds usually does not take 10 seconds. The execution time is the time required by the model to produce the response and depends on many factors, such as model complexity, solver type, step size and system speed.

Discretization is carried out with values between 2 and 4 Numbers of Elements (NOE): after each model simulation, the NOE is increased or decreased to analyze its impact on the system outputs. In addition, the values of the parameters dependent on the NOE are given in Tables 10 and 11.

Table 9: *Solar panel characteristics for both scenario*

	Scenario 1	Scenario 2	Unit
External Force, F_x	50	5×10^3	N
Simulation time	20	14	s
Thickness (x)	0.05	0.03	m
Width (y)	2.1	1	m
Length (z)	10.85	1	m
Mass	90	4	kg
Young's module of Elasticity	3.5×10^9	70×10^9	Pa
Poisson's ratio	0.3	0.3	
Second moment of area	2.1875×10^{-5}	2.25×10^{-6}	m^4
Damping ratio, ζ	0.1	0.1	

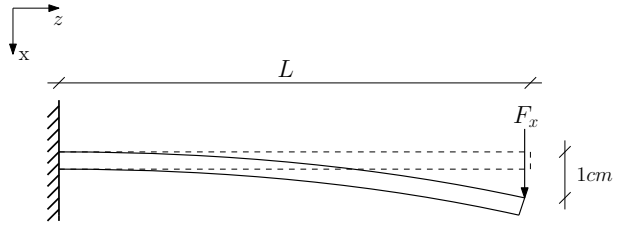


Figure 33: *Simplified flexible body subjected to an external load* F_x

The criteria for comparing the lumped-parameter and the flexible-beam methods with solar panel models are:

- frequency difference, which is taken as the relative error between the two methods;
- execution time.

Since the goal of this thesis is to find a numerically efficient method to model a real-time simulator, it is essential that its execution time is equal to or faster than that of the *Simulation Time*.

Each simulation is also conducted with a different value of solver step-size, from 1×10^{-5} s to 1×10^{-2} s. The smaller the step size, the more accurate results it provides, but at a higher computational cost. In addition, Chapter 5.3 will describe that during the first few simulations an explicit fixed-step solver (*ode*3) was chosen, but then, due to some problems encountered in *Scenario* 2, the use of an implicit fixed-step solver (*ode*1*be*) was chosen upon for both scenarios.

5.1.1 Scenario 1

In Table 10, l_{umped} is the length of a flexible body unit (Eq. 3.23), m_{lumped} is one of the two lumped unit masses (Fig. 29), k_r is the spring coefficient and b_r is the damping coefficient, which depends on α. From the calibration of the proportional constant α in Chapter 3.3.1, its value was chosen as $\alpha = 0.005$ for this scenario. The simulation output provides the panel displacements along the axis on which the force acts. This response is then converted from its original domain (time) to a representation in the frequency domain.

Table 10: *Scenario 1 Lumped-parameter method characteristics, which depends on the Number of Elements (NOE)*

	NOE 2	NOE 3	NOE 4	Unit
l_{lumped}	5.425	3.6167	2.7125	m
m_{lumped}	22.5	15	11.25	kg
k_r	1.411×10^4	2.117×10^4	2.822×10^4	N/m
b_r	70.5645	105.8468	141.129	Ns/m

To compare the lumped-parameter model (Lp) to the flexible-beam model (Fb), the relative error percentage between the resulting frequencies is computed (Fig. 34) by considering the flexible-beam model as a benchmark:

$$Error(\%) = \frac{f_{Lp} - f_{Fb}}{f_{Fb}} \cdot 100 \tag{5.1}$$

Fig. 34 shows the error percentage paths of the two methods according to Eq. 5.1. With a Number of Elements of three and four, the frequency of the panel response with the lumped-parameter method is comparable to that of the flexible-beam model, with a relative error lower than 3.5%, while it diverges slightly with two Numbers of Elements ($\simeq$7.5%). This confirms that the greater the Number of Elements, the more accurate the system response.

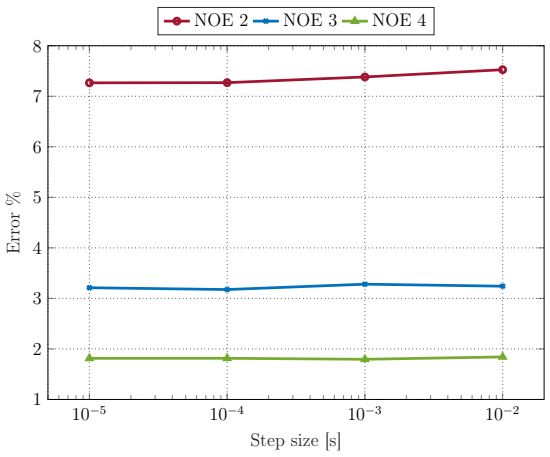
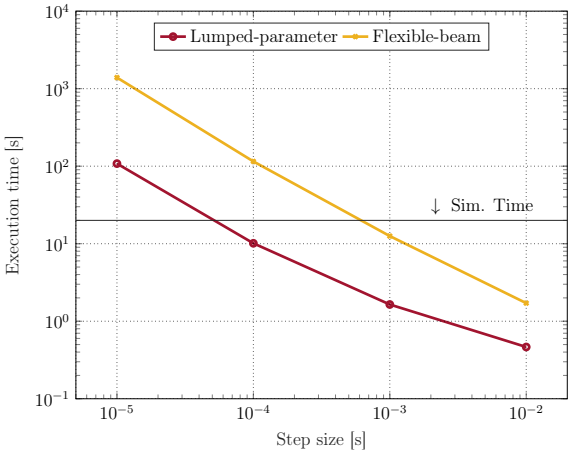
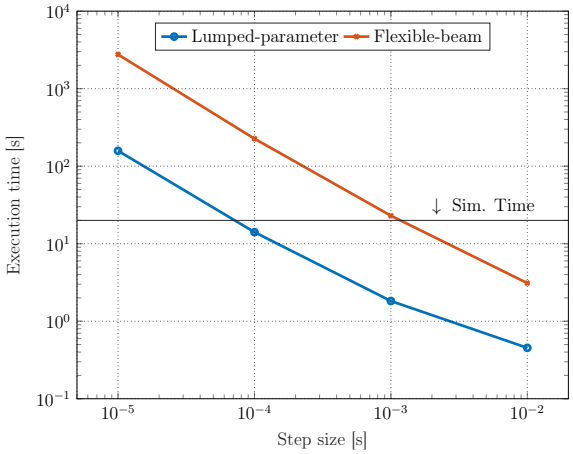


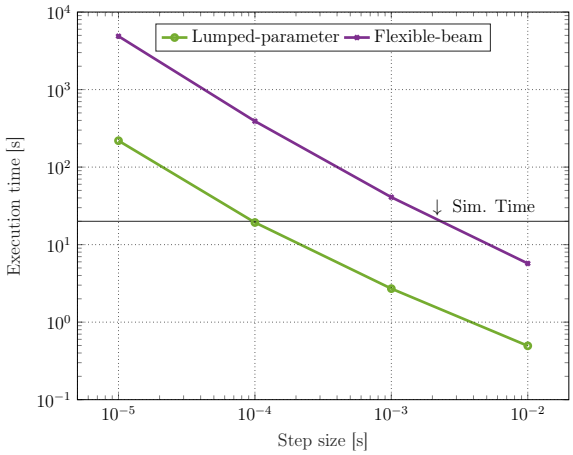
Figure 34: *Scenario 1 frequency error percentage distribution for each Number of Elements*



(a) NOE 2



(b) NOE 3

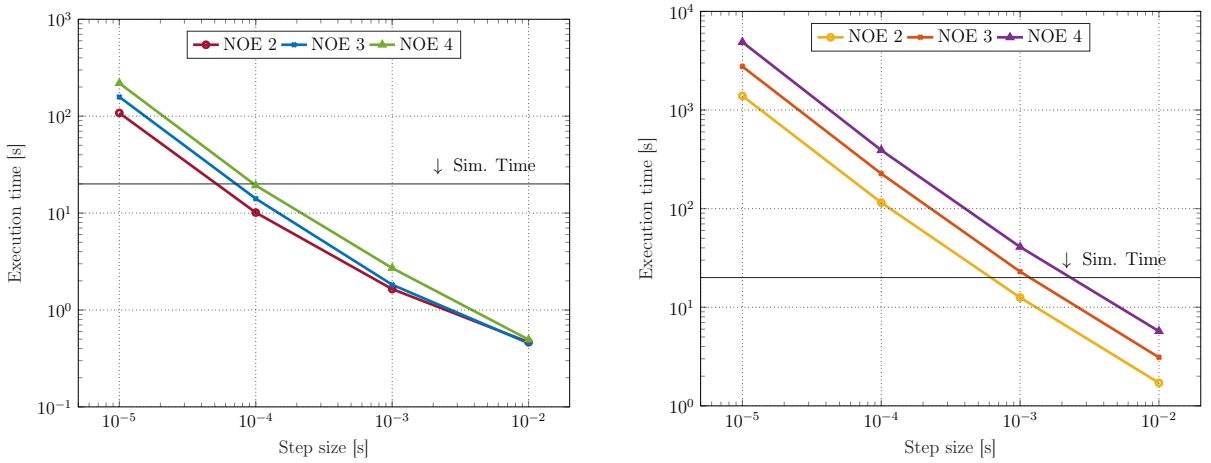


(c) NOE 4

Figure 35: *Scenario* 1 *execution time of Lumped-parameter and Flexible-beam methods based on the Number of Elements, with a* Simulation Time *of* 20*s*

Each simulation of this scenario was calculated with a *Simulation time* of 20 *s*. Fig.35 shows that, compared with the flexible-beam method, the lumped-parameter method shows more results that satisfy the task of having a lower execution time than 20 *s* (step-sizes 1×10^{-4} s, 1×10^{-3} s and 1×10^{-2} s).

From the study of the effects of discretization, it is possible to observe that a larger Number of Elements increases the computational cost of both methods, slowing down the speed of simulation. This behavior is also shown in Fig. 36, where it can be seen that comparing all simulations performed with the two methods, the flexible-beam method always has a higher computational cost.



(a) Lumped-parameter method **(b)** Flexible-beam method

Figure 36: *Scenario* 1 *execution time based on Lumped-parameter and Flexible-beam methods, with a* Simulation Time *of* 20*s*

Table 11: *Scenario 2 Lumped-parameter method characteristics, which depends on the Number of Elements (NOE)*

	NOE 2	NOE 3	NOE 4	Unit
l_{lumped}	0.5	0.333	0.25	m
M_{lumped}	1	0.667	0.5	kg
k_r	3.15×10^5	4.725×10^5	6.3×10^5	N/m
b_r	3.15	4.725	6.3	Ns/m

5.1.2 Scenario 2

Table 11 summarizes the same characteristics of the method described for *Scenario* 1. As it can be seen from this table and Table 9, the solar panel analysed in this scenario has a length lower than the one of the solar panel considered in *Scenario* 1. With a 1-meter long solar panel, the response frequencies are higher (on the order of Hz instead of mHz).

In order to obtain an accurate approximation of the solution, it is necessary to choose a step-size that is small enough ($\leq 1 \times 10^{-4}\, s$) to capture the features of the solution, including any high-frequency components. However, a step-size smaller than $1 \times 10^{-5}\, s$ has a too high computational cost that would lead to excessive execution time, and consequently the model could no longer be used as a real-time simulator. On the other hand, at

higher step-sizes, the high-frequency components are no longer accurately represented in the frequency spectrum because the resulting frequency resolution is coarser. Hence, these frequency components which are still present in the original signal, may appear to have a frequency value of zero in the frequency spectrum. This problem is better described in the final section of this chapter (Chapter 5.3), which explains that the use of an explicit fixed-step solver at those step-sizes brought to the simulation to stop at every run.

The following section reports results provided with an implicit fixed-step solver (*ode1be*), with only three values of step-size.

Scenario 2 results

The lumped-parameter method is modeled with a proportional constant α of 1×10^{-5} for this scenario. Fig. 37, shows the relative error percentage between the frequency of each method computed with Eq. 5.1. As it can be seen, these paths are very different from those in *Scenario* 1, and errors are also higher: from 3% to 14% for a step-size of $1 \times 10^{-4}\, s$ and from 5% to 12% for a step-size of $1 \times 10^{-5}\, s$.

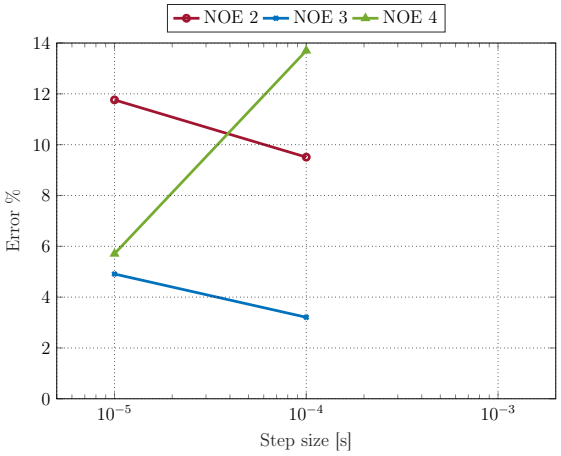
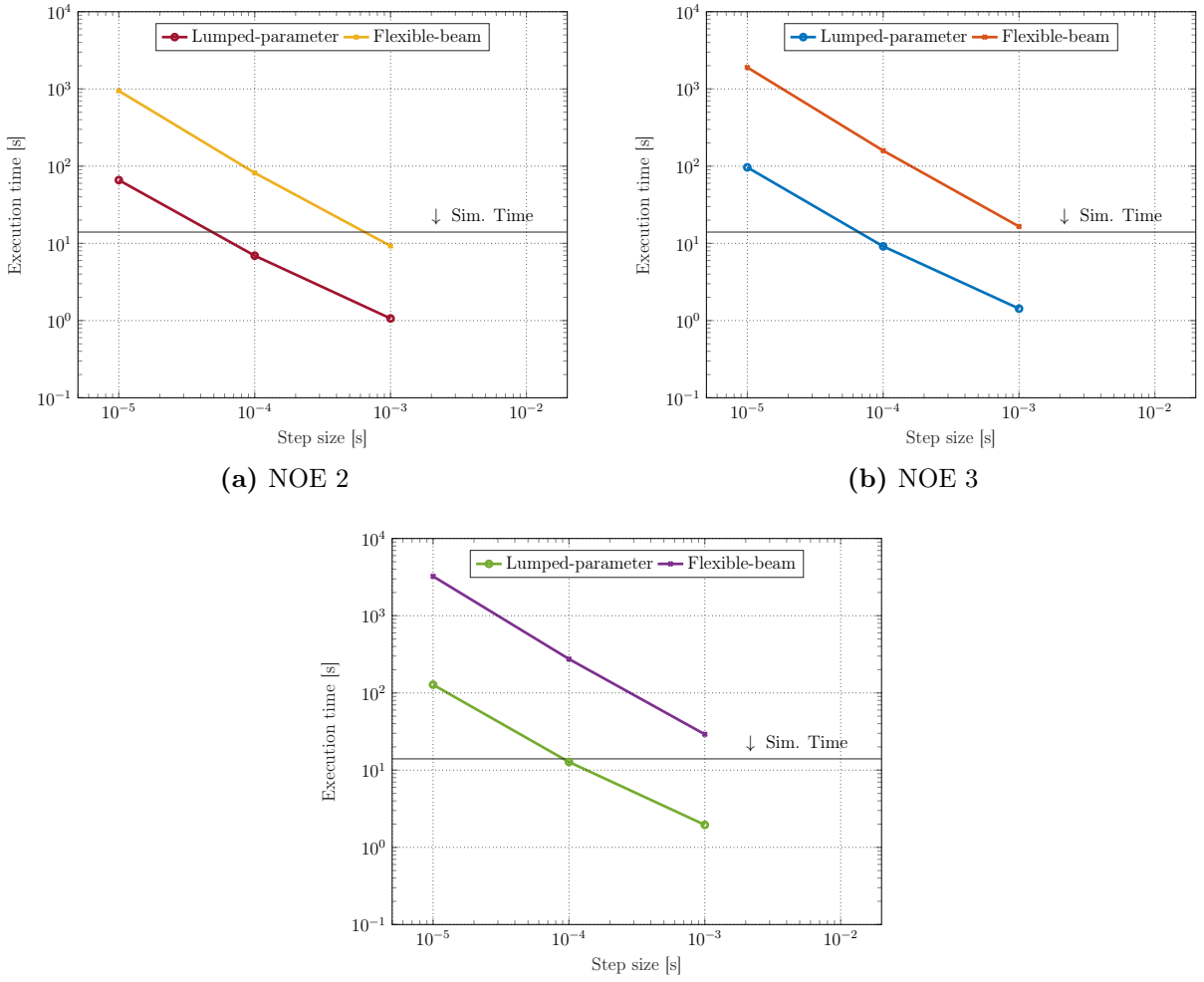


Figure 37: *Scenario* 2 *error percentage distribution for each Number of Elements*

Note that for a step-size of $1 \times 10^{-3}\, s$, it may appear that there are no frequency values in the frequency spectrum for either method, but, as explained before, high-frequency components are still present in the original signal. Thus, in fact, the execution time for that step-size is still reported. In addition, it can be seen that the relative error rate with two and three Numbers of Elements decreases when the step size is increased, although the simulation result should be less accurate with a larger step size. Probably, with such high frequencies, the step-size $1 \times 10^{-4}\,$s is not low enough for the simulator to generate reliable results: NOE 4 has the highest error percentage with that step size, even though with a large Number of Elements the simulation should be more accurate (Chapter 3.2.2).

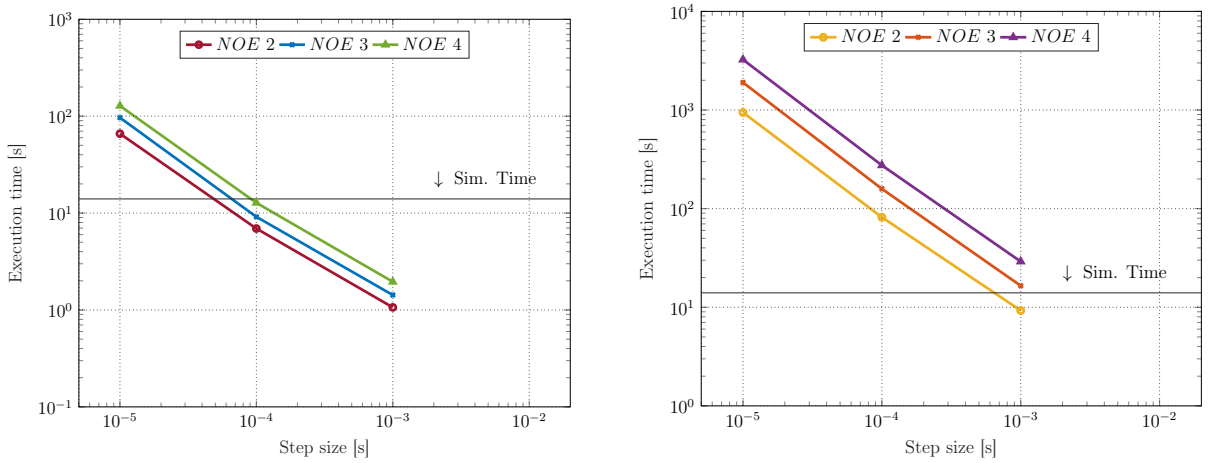


(a) NOE 2

(b) NOE 3

(c) NOE 4

Figure 38: *Scenario* 2 *execution time of Lumped-parameter and Flexible-beam methods based on the Number of Elements, with a* Simulation Time *of* 14 *s*



(a) Lumped-parameter method **(b)** Flexible-beam method

Figure 39: *Scenario* 2 *execution time based on Lumped-parameter and Flexible-beam methods, with a* Simulation Time *of* 14 *s*

Each simulation is computed with a *Simulation Time* of 14 *s*. From the comparison of the two methods execution time (Figs. 38 and 39), it can be seen that the lumped-parameter method still obtains the most results with less execution time than the *Simulation Time* and, in general, has the lowest computational cost. However, Fig. 37 shows that this method is not very accurate when used in this scenario.

5.1.3 Methods comparison

All *Frequency* and *Execution Time* results shown in previous figures, are summarized in Tables 12 and 13. By observing all models frequencies, it can be seen that those of the lumped-parameter method depend much more on the Number of Elements than those of the flexible-beam method, which always maintain a value in the range of 452.1 mHz to 454.1 mHz for *Scenario* 1, and 107.04 Hz to 112.21 Hz for *Scenario* 2.

In addition, simulations with circled Execution Time values for both methods are considered valid to be used in a real-time simulator; as a matter of fact, these values are less than the *Simulation Time* of 20 s for *Scenario* 1 and 14 s for *Scenario* 2.

Table 12: *Comparison of the models main results for Scenario* 1

NOE	Step Size [s]	Lumped-parameter *frequency* [mHz]	Flexible-beam *frequency* [mHz]	Relative Error Percentage (%)
2	1×10^{-5}	487.062	454.070	7.27
	1×10^{-4}	487.043	454.037	7.27
	1×10^{-3}	487.388	453.882	7.38
	1×10^{-2}	486.343	452.304	7.53
3	1×10^{-5}	468.481	453.905	3.21
	1×10^{-4}	468.450	454.032	3.18
	1×10^{-3}	468.375	453.494	3.28
	1×10^{-2}	466.796	452.139	3.24
4	1×10^{-5}	462.055	453.823	1.81
	1×10^{-4}	462.018	453.783	1.81
	1×10^{-3}	461.755	453.613	1.79
	1×10^{-2}	460.436	452.100	1.84

NOE	Step Size [s]	Lumped-parameter *Execution Time* [s]	Flexible-beam *Execution Time* [s]	*Simulation Time* of 20 s
2	1×10^{-5}	107.95	$1.4 \times 10^3 \simeq 23min$	
	1×10^{-4}	10.15	115.15	
	1×10^{-3}	1.65	12.55	
	1×10^{-2}	0.465	1.72	
3	1×10^{-5}	157.89	$2.76 \times 10^3 \simeq 46min$	
	1×10^{-4}	14.1	226.24	
	1×10^{-3}	1.83	22.97	
	1×10^{-2}	0.45	3.1	
4	1×10^{-5}	219.01	$4.88 \times 10^3 \simeq 81min$	
	1×10^{-4}	19.3	391.28	
	1×10^{-3}	2.71	40.88	
	1×10^{-2}	0.5	5.72	

Table 13: *Comparison of the models main results for Scenario 2*

NOE	Step Size [s]	Lumped-parameter *frequency* [Hz]	Flexible-beam *frequency* [Hz]	Relative Error Percentage (%)
2	1×10^{-5}	119.663	107.07	11.76
	1×10^{-4}	101.536	112.205	9.51
	1×10^{-3}	-	-	-
3	1×10^{-5}	112.300	107.044	4.91
	1×10^{-4}	115.448	111.860	3.21
	1×10^{-3}	-	-	-
4	1×10^{-5}	113.148	107.040	5.71
	1×10^{-4}	96.538	111.863	13.70
	1×10^{-3}	-	-	-

NOE	Step Size [s]	Lumped-parameter *Execution Time* [s]	Flexible-beam *Execution Time* [s]	*Simulation Time* of 14 s
2	1×10^{-5}	65.98	$944.14 \simeq 15min$	
	1×10^{-4}	6.93	81.77	
	1×10^{-3}	1.06	9.26	
3	1×10^{-5}	96.49	$1.90 \times 10^3 \simeq 31min$	
	1×10^{-4}	9.15	158.80	
	1×10^{-3}	1.43	16.52	
4	1×10^{-5}	127.78	$3.24 \times 10^3 \simeq 54min$	
	1×10^{-4}	12.76	274.88	
	1×10^{-3}	1.96	29.10	

5.2 Robotic arm

The implemented robotic arm models are described in Chapter 4.3.2 and its characteristics are shown in Table 14. The first two robotic manipulator links (L_1 and L_2) are considered as flexible bodies with a square section of 2.5×10^{-3} m^2, while the third link L_3 is considered as a rigid body cylinder, since it is the shortest. This version of the manipulator is considered to compare the lumped-parameter and the flexible-beam methods applied to the first two robotic arm links, along with a third version featuring three rigid-body links.

Table 14: *3 DOF Robotic arm characteristics*

	Value	Unit
First link length, L_1	1	m
Second link length, L_2	1	m
Third link length, L_3	0.2	m
Initial position (x,y)	(2,0)	m
Final position (x,y)	(0.5,1.5)	m
Simulation Time	10	s
Density	2700	$\mathrm{kg\,m^{-3}}$
Young's module of Elasticity	70	GPa
Poisson's ratio	0.3	

For the two flexible links, the rectangular shape was chosen to use the lumped-parameter method with the same procedure described in Chapter 3.3, while their square section was chosen to have comparable dimensions with those of the cylindrical link, already included in the default manipulator model, described in Chapter 2.3.

Criteria for this comparison are:

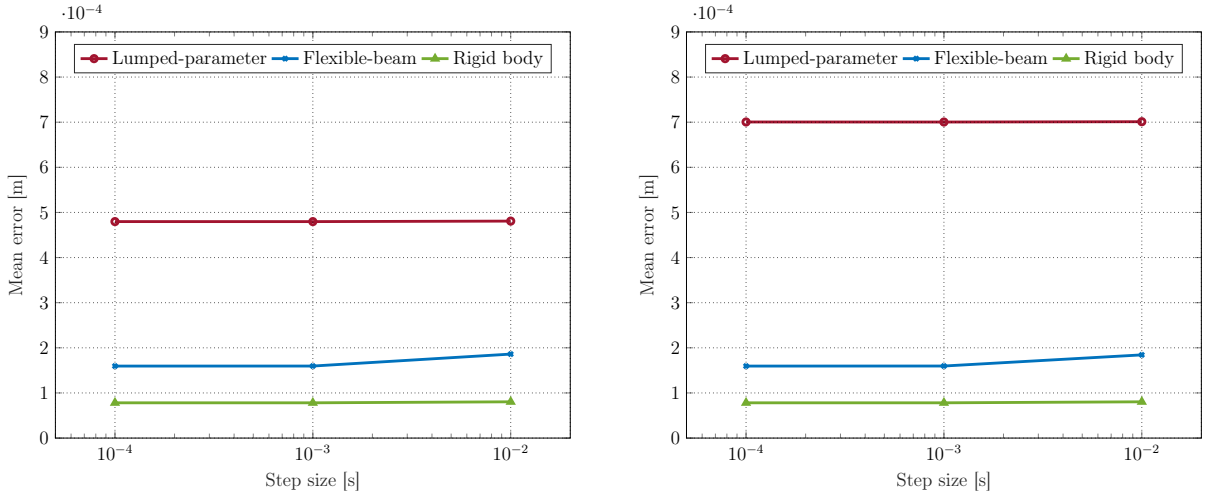
- Mean error between the instantaneous position of the end-effector and its position set by the robotic arm subsystem at each time step;
- Execution time, which has to be equal to or less than the *Simulation Time* of 10 s.

The instantaneous position of the end-effector is computed analytically in MATLAB and is taken as the reference value, while the position set by the robotic arm subsystem depends on the accuracy of the flexible body method used to model the two links.

5.2.1 End-Effector position error

The End-Effector mean position error of the three considered models is shown in Fig.40 and consists of subtracting, for each simulation step, the EE position computed analytically by the *Trajectory Generation* block (Chapter 4.3.2) from its position at the output of the robotic arm subsystem. At the end of each simulation, the mean position error and its standard deviation are taken into account to compare the three flexible-body models. It should be noted that this error is different from the relative error used to analyze the solar panel behavior: while that was the percentage of the relative frequency error between the two flexible body methods, this is a position error computed separately for each method.

Since the rigid-body model has the same number of elements in all simulations, its presence is only for comparison with other models; it can also be noted that its values are the lowest ($\simeq 0.78 \times 10^{-4} m$). While the flexible-beam position error has a comparable path with both Numbers of Elements, the position error of the lumped-parameter model is higher with three Numbers of Elements (Fig. 40).



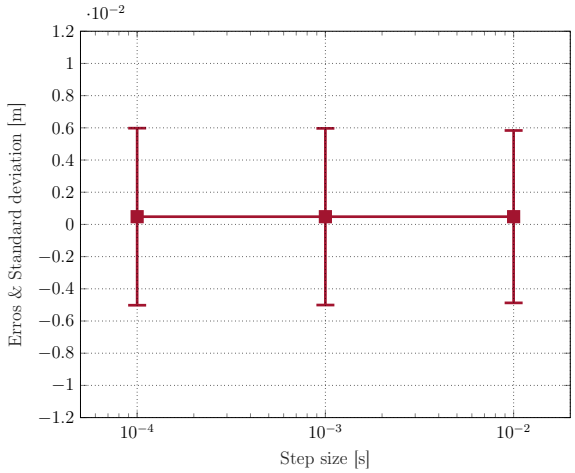
(a) NOE 2 **(b)** NOE 3

Figure 40: *Mean error between the instantaneous position of the end-effector and the position set by the guide*

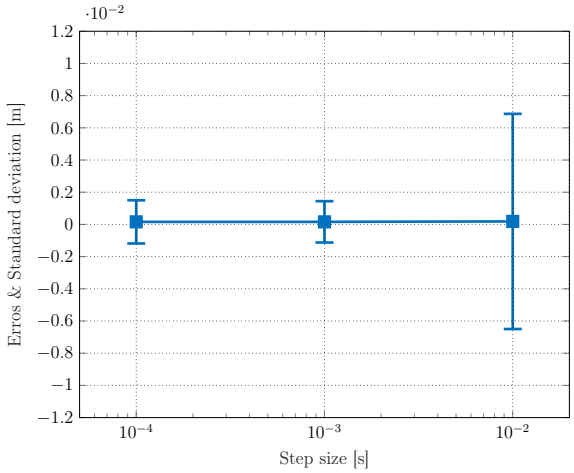
However, this error still remain on the order of 1×10^{-4}, which corresponds to an EE position error of half a millimeter, while a real model of the robotic arm works on the order of centimeters. Therefore, a margin of safety is provided for the lumped-parameter method and the quality of its performance can be based on its execution time.

Another measure for comparing End-Effector position errors is the standard deviation (STD), which is a measure of the amount of variation or dispersion of a set of values. Standard deviation is considered as a measure of uncertainty: a low STD indicates that values tend to be closer to the mean of the set, while a high STD indicates that values are spread out over a wider range [75]. For each model, the standard deviation was computed in the MATLAB script of the robotic arm with the following syntax [76]: $S = std(E)$, where E is a vector containing the model position errors at each time step.

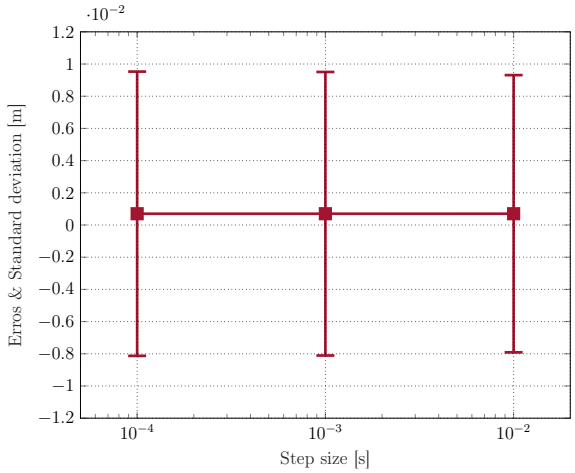
Fig. 41 shows the STD values of each method at different Numbers of Elements. Each figure has the same values on the X-axis and Y-axis, and the line connecting each point is the mean error path shown in Fig. 40. Moreover, STD values of the flexible beam models increase by $\approx 5.65 \times 10^{-3} m$ between step $1 \times 10^{-3}\, s$ and $1 \times 10^{-2}\, s$, while the lumped-parameter models have a stable STD value for both NOE. However, the lumped model values have an higher dispersion ($\pm\, 5.5 \times 10^{-3}\, m$ and $\pm\, 8.8 \times 10^{-3}\, m$ with NOE 2 and NOE 3 respectively). This means that during the motion of the robotic arm, the lumped-parameter method is less accurate in describing the behavior of its flexible bodies with both numbers of elements, while the flexible-beam method provides less accurate values only with a step-size of $1 \times 10^{-2}\, s$.



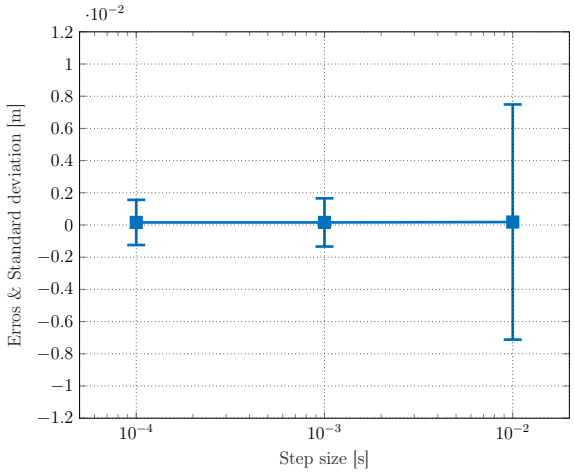
(a) NOE 2, lumped-parameter method



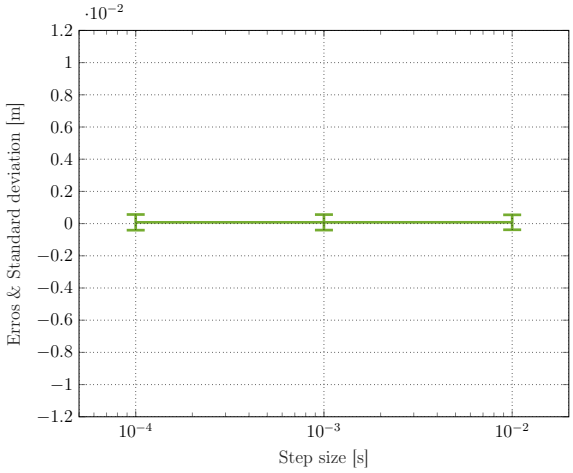
(b) NOE 2, flexible-beam method



(c) NOE 3, lumped-parameter method

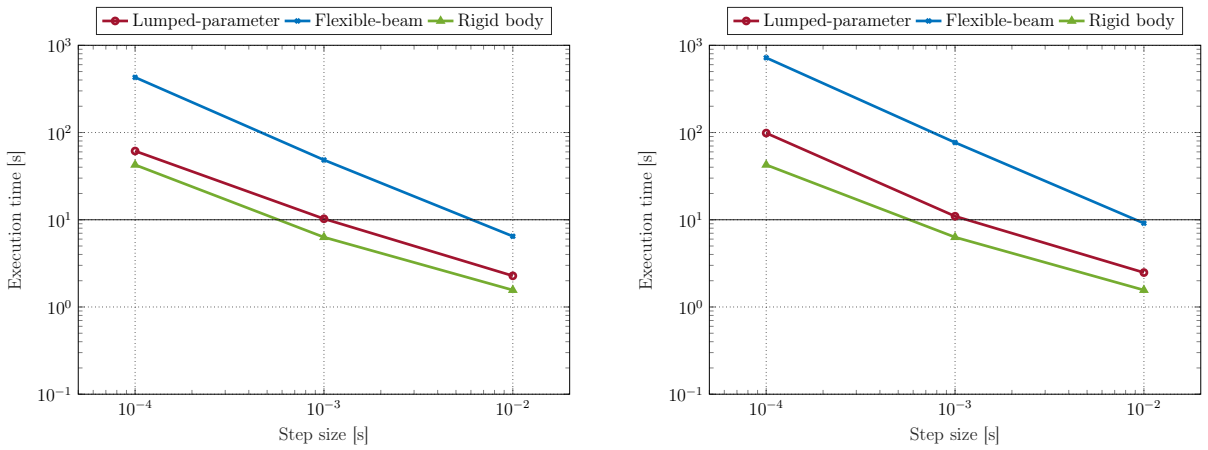


(d) NOE 3, flexible-beam method



(e) Rigid-body method

Figure 41: *Error & Standard deviation (STD) of all models*



(a) NOE 2 **(b)** NOE 3

Figure 42: *Execution time based on the models number of elements*

To simulate the robotic arm motion, the *Simulation Time* for each model is $10s$. As it can be seen in Fig. 42, the Execution Time follows a comparable path for each discretization. The rigid-body is the model with the best combination of high accuracy and short execution time, but it does not consider the robotic arm links as flexible bodies, so its results are not a viable solution for the flexible bodies theory. When considering the two flexible-body methods, the lumped-parameter method is overall the one with the shortest execution time.

5.2.2 Methods comparison

All results of the robotic arm End-effector position error are summarized in Tables 15, 16 and 17, while execution times are summarized in Tables 18 and 19.

Both flexible body models have the same number of simulation with a Execution Time less than the *Simulation Time*, although the lumped-parameter method has two Execution Time values at a step size of $1 \times 10^{-3}\, s$ (10.26 s and 10.98 s with NOE 2 and NOE 3 respectively), which are remarkably close to the *Simulation Time*.

Table 15: Lumped-parameter models end-effector position error

NOE	Step Size [s]	Mean error $[1 \times 10^{-4}]$	St. deviation $[1 \times 10^{-3}]$
2	1×10^{-4}	4.80	5.50
	1×10^{-3}	4.80	5.50
	1×10^{-2}	4.81	5.40
3	1×10^{-4}	7.00	8.80
	1×10^{-3}	7.00	8.80
	1×10^{-2}	7.01	8.60

Table 16: Flexible-beam models end-effector position error

NOE	Step Size [s]	Mean error $[1 \times 10^{-4}]$	St. deviation $[1 \times 10^{-3}]$
2	1×10^{-4}	1.60	1.30
	1×10^{-3}	1.60	1.30
	1×10^{-2}	1.86	6.70
3	1×10^{-4}	1.60	1.40
	1×10^{-3}	1.60	1.50
	1×10^{-2}	1.84	7.30

Table 17: Rigid-body models end-effector position error

NOE	Step Size [s]	Mean error $[1 \times 10^{-4}]$	St. deviation $[1 \times 10^{-3}]$
-	1×10^{-4}	0.78	0.49
	1×10^{-3}	0.78	0.48
	1×10^{-2}	0.80	0.46

Table 18: Execution Time for Lumped-parameter and Flexible-beam models

NOE	Step Size [s]	Lumped-parameter *Execution Time* [s]	Flexible-beam *Execution Time* [s]	*Simulation Time* = 10 s
2	1×10^{-4}	61.35	429.01	
	1×10^{-3}	10.26	48.47	
	1×10^{-2}	2.28	6.45	
3	1×10^{-4}	98.56	$719.61 \simeq 12min$	
	1×10^{-3}	10.98	76.82	
	1×10^{-2}	2.48	9.12	

Table 19: Execution Time for the Rigid-body models

NOE	Step Size [s]	Rigid-body *Execution Time* [s]
-	1×10^{-4}	42.58
	1×10^{-3}	6.31
	1×10^{-2}	1.56

5.3 Solver test

The analyses conducted as part of this thesis serve to support the design of the FES simulator (Chapter 2.1), thus the first simulations were conducted with *ode*3, the same explicit fixed-step solver also used for the FES. This solver integration technique is the Bogacki-Shampine method for the numerical solution of ordinary differential equations, which is a Runge-Kutta method of order three, so it uses approximately three evaluations of functions per step [77]. This solver did not generate any issues with solutions of *Scenario* 1, unlike in *Scenario* 2 (Chapter 5.1.2).

Since the frequency of a system is inversely proportional to the length of the element, the short solar panel in *Scenario* 2 generated high frequency responses that could only be calculated with small step sizes. Hence, the use of an explicit fixed-step solver at high step-sizes resulted in an error, i.e., the state of the derivative when the load was applied was not finite, creating a singularity in the solution. Since this error occurred only after a certain value of the panel frequency, it was necessary to figure out what was the threshold length associated with that frequency value for each Number of Elements.

In order to accomplish this task, a solver test was conducted on a solar panel model with the same characteristics described in Table 9 for *Scenario* 2. Both lumped-parameter and flexible-body methods were used to realize the panel, with a Number of Elements varying from 2 to 4. The test consists of a simulation, launched with the solver *ode*3 and a step size of $1 \times 10^{-3}\, s$, that computes the same model with a smaller length each time until it is stopped due to the derivation error.

Table 20: *Solver test for 2 Numbers of Elements*

Length [m]	Lumped-parameter frequency [Hz]	Flexible-beam frequency [Hz]	Relative Error Percentage (%)
1.6	3.42	3.25	5.13
1.5	3.83	3.66	4.54
1.4	4.41	4.16	5.99
1.3	5.08	4.91	3.39
1.2	5.83	5.75	1.45
1.1	499.96	6.83	7217.07
1.0	2.74	1.64	66.67

Table 21: *Solver test for 3 Numbers of Elements*

Length [m]	Lumped-parameter frequency [Hz]	Flexible-beam frequency [Hz]	Relative Error Percentage (%)
2.2	1.75	1.67	4.99
2.1	1.92	1.84	4.55
2.0	2.08	2.079	0.01
1.9	2.33	2.25	3.70
1.8	2.58	2.58	0.01
1.7	499.96	2.83	17547.04
1.6	1.17	0.25	64.11

Table 22: *Solver test for 4 Numbers of Elements*

Length [m]	Lumped-parameter frequency [Hz]	Flexible-beam frequency [Hz]	Relative Error Percentage (%)
2.8	1.08	1.08	7.0×10^{-4}
2.7	1.17	1.17	1.0×10^{-4}
2.6	1.25	1.25	9.0×10^{-4}
2.5	1.33	1.33	1.0×10^{-4}
2.4	0.58	0.365	58.82
2.3	0.17	0.09	89.47
2.2	0.17	0.09	90.00

Tables 20, 21 and 22 show the panel length at which each simulation stops, along with the frequency of the two methods and the relative error between them. In these tables, values of the minimum length (L_{min}) and the maximum frequency (f_{max}) reached for the three Numbers of Elements are circled. It can be seen that the frequency increases as the panel length decreases until a length at which the maximum frequency, which probably

causes the derivative error, is reached. Also, frequency values after f_{max} have a relative error $> 50\%$, which means that at those lengths the model cannot adequately simulate the dynamic response.

To solve this problem, the solar panel of *Scenario* 2 was analyzed with the *ode*1*be* implicit fixed-step solver (Chapter 3.4.1), which uses a Backward Euler integration technique [66]. This solver successfully overcame the derivation error and computed all simulations at each step-size. However, step-sizes that generated the error with the explicit solver failed to accurately capture the panel dynamics for simulations performed with the implicit solver: this attitude does not necessarily imply that at those step sizes the frequency value is zero. Increasing the step-size of a solver can result in a loss of accuracy in the estimation of the frequency content of a signal, particularly if the signal contains high-frequency components, such as those in *Scenario* 2.

When using numerical solvers to estimate the frequency content of a signal, such as with a Discrete Fourier Transform (DFT) (Chapter 3.2.1), the frequency resolution is determined by the number of samples in the time domain and the sampling rate. Increasing the step-size of a solver effectively decreases the number of samples in the time domain, which reduces the frequency resolution and can lead to a loss of accuracy in the estimation of the signal frequency content. If the solver step-size is increased too much, the resulting frequency resolution can become so coarse that high-frequency components are no longer accurately represented in the frequency spectrum. In this case, the high-frequency components may appear to have a frequency value of zero in the frequency spectrum, even though they are present in the original signal (Fig. 39). Therefore, it is important to choose an appropriate step-size when using numerical solvers to estimate the frequency content of a signal, to ensure that the frequency resolution is sufficient to accurately represent the frequency content of the signal.

6. Conclusions

This Master Thesis had a main goal: to find a numerically efficient method with adequate execution time to be used in a real-time simulator and suitable for studying the behavior of satellite flexible bodies. To achieve this task a Simulation Tool was developed to model these bodies and reproduce their dynamics.

The Simulation Tool was implemented in the MATLAB - Simulink environment, and its development was presented in detail in Chapter 4. The two spacecraft elements studied within this thesis were a solar panel and a 3 DOF robotic arm. The objective was to create and compare several models of these two spacecraft flexible bodies, each implemented by using Simscape Multibody with two different mathematical representations of flexible bodies: the *Lumped-parameter* and the *Flexible-beam* method. The Tool offers the possibility to reproduce the response of the solar panel when subjected to an external load and to simulate the robotic arm motion from one point to another. Furthermore, each spacecraft element is modeled with different Number of Elements (NOE) and implemented with different simulation step-sizes in order to compare the accuracy and execution time of the two flexible body methods. The solar panel model accuracy was analyzed by computing the Relative error (RE) percentage, which is the ratio of the absolute error of the two method frequencies to the frequency value of the flexible-beam method. On the other hand, the robotic arm model accuracy was expressed by finding the End-Effector position error and its standard deviation. This error is the mean difference between the

instantaneous position of the End-Effector and its position set by the robotic arm model subsystem, after a movement from an initial point to an end point. Thus, finally, for both spacecraft elements, the execution time was tracked after each simulations and compared to the *Simulation Time*, which is the time specified by the simulation in seconds. The goal here was for the execution time of each model to be equal to or less than the *Simulation Time*, so that the model could be used for modeling a real-time simulator. Tables within Chapters 5.1.3 and 5.2.2 include all these simulations results.

The study showed that the dynamic response of the solar panel was affected by various factors, particularly its geometry: by reducing the length of the solar panel from 10.85 m to 1 m in *Scenario* 2, the response frequencies increased from the mHz to the Hz. At high step-sizes the number of samples in the time domain decreases and thus the frequency resolution is reduced; consequently, these high-frequency components were no longer accurately represented in the frequency spectrum, resulting in a loss of information for a step-size of $1 \times 10^{-3}\,s$. On the other hand, the solar panel dynamic response could be predicted with reasonable accuracy for *Scenario* 1 and the lumped-parameter method was able to simplify the complexity of the flexible structure while still capturing its essential dynamic behavior. To model a real-time simulator, the relative error (RE) percentage should be at least less than 5%: by comparing the frequencies of the lumped-parameter method with those of the flexible-beam method, the RE with three and four Numbers of Elements was $\approx 3.23\%$ and $\approx 1.81\%$ respectively for all step-sizes, while it was $\approx 7.36\%$ for two Numbers of Elements. In addition, it proved to be 91% faster on average than the flexible-beam method to simulate the solar panel response (Table 12).

However, the same accuracy considerations could not be made when analyzing the robotic arm. The End-Effector position error computed with the lumped-parameter method gave a mean error ≈ 3 times higher than that of the flexible-beam method with two Numbers of Elements, and ≈ 5 times higher with three Numbers of Elements. In addition, the mean position error for the lumped-parameter model increased by 46% with a larger Number of Elements, while that of the flexible-beam model gave comparable

results with both Numbers of Elements. The Standard deviation (SD) also reflected the same attitudes for both methods: lumped-parameter model solutions showed greater dispersion with all step-size values than those of the flexible-beam model, although the latter method presented an high dispersion with a step size of $1 \times 10^{-2}\, s$ for each Number of Elements (Tables 15 and 16). Even though the analyses conducted on the behavior of the robotic arm have shown that the lumped-parameter method has higher errors than the flexible-beam method, they still remain on the order of 1×10^{-4}, which corresponds to an end-effector position error of half a millimeter, while a real model of the robotic arm works on the order of centimeters. Therefore, a margin of safety is provided for the lumped-parameter method and the quality of its performance can be based on its execution time, which remains 76% faster with two Numbers of Elements and 84% faster with three Numbers of Elements than the flexible-beam method.

Through these analyses, it can be seen that these flexible bodies exhibit complex behaviors that must be carefully considered in their design, operation and control. The solver analyses also demonstrated the importance of using the right solver for the model to be simulated: due to the shorter length of the solar panel from one scenario to the next, a fixed-step explicit solver was unable to compute the panel frequencies after a certain length, until a minimum length was reached after which the simulation was stopped. Tables in Chapter 5.3 show that for each Number of Elements the frequencies increase steadily with decreasing panel length to a value that corresponds to the maximum panel frequency f_{max}. All values after this maximum frequency cannot be considered suitable to describe the dynamic behavior of the solar panel because of their high relative error rate values: well over 55% for all frequencies after f_{max}. The fixed-step implicit solver *ode1be* was the right solver to overcome this problem: the results for *Scenario* 1 remained the same even though the solver was changed, and for *Scenario* 2 no more errors occurred leading the termination of the simulation. However, the implicit solver failed to accurately capture the panel dynamics for simulations performed at high step-sizes ($\geq 1 \times 10^{-3}\, s$): frequency values of these step-sizes were not displayed or were not consistent if compared

with the values of smaller step-sizes. This attitude justifies obtaining more accurate results with *Scenario* 1, where significantly lower frequencies were involved.

This problem did not occur for the robotic arm models because the results analyzed were position errors after a defined movement and not frequency responses after a received impulse. Despite this, the implicit solver *ode*1*be* was still used to simulate the motion of the robotic arm.

In conclusion, the research revealed that the dynamic behavior of flexible bodies can be modeled by various methods and that the appropriate modeling approach depends on the specific problem and the required accuracy of the results. The lumped-parameter method of flexible bodies has proved to be a valuable tool for the design, analysis, and optimization especially for its simulation speed, which makes it suitable for modeling real-time simulators. Future research in this area, may focus on developing more accurate and efficient modeling techniques and control strategies, along with solving challenges associated with the design and operation of flexible-body satellites, such as the trade-off between stiffness and weight, thermal management, and compatibility with other subsystems. For example, the lumped-parameter method can be extended to model more complex robotic manipulators and solar panel structures, such as flexible arrays, and to study the effect of their deformation on power output.

Bibliography

[1] J. P. Davis, J. P. Mayberry, and J. P. Penn. "On-orbit servicing: Inspection, repair, refuel, upgrade, and assembly of satellites in space". In: *The Aerospace Corporation report* (Apr. 2019).

[2] Borna Monazzah Moghaddam and Robin Chhabra. "On the guidance, navigation and control of in-orbit space robotic missions: A survey and prospective vision". In: *Acta Astronautica* 184 (2021), pp. 70–100. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2021.03.029.

[3] *On-Orbit Satellite Servicing Study*. NASA Project Report. National Aeronautics and Space Administration (NASA). Oct. 2010.

[4] Paulina Swiatek et al. "Design Principles for Sustainable Close Proximity Operations". In: (2019). URL: https://www.hou.usra.edu/meetings/orbitaldebris2019/orbital2019paper/pdf/6177.pdf.

[5] Alex Ellery, Joerg Kreisel, and Bernd Sommer. "The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth". In: *Acta Astronautica* 63.5 (2008), pp. 632–648. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2008.01.042.

[6] Joseph H. Saleh, Daniel E. Hastings, and Dava J. Newman. "Spacecraft Design Lifetime". In: *Journal of Spacecraft and Rockets* 39.2 (2002), pp. 244–257. DOI: 10.2514/2.3806. eprint: https://doi.org/10.2514/2.3806.

[7] Mak Tafazoli. "A study of on-orbit spacecraft failures". In: *Acta Astronautica* 64.2 (2009), pp. 195–205. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2008.07.019.

[8] HubbleSite. *Enabling Science Through New Technologies*. 2009. URL: https://hubblesite.org/mission-and-telescope/servicing-missions.

[9] European Space Policy Istitute (ESPI). *In-Orbit Servicing: Challenges and Implications of an Emerging Capability*. URL: https://www.espi.or.at/wp-content/uploads/espidocs/ESPI.

[10] L. Bitetti et al. "Reliability Model Supporting Satellite Life Extension and Safe Disposal". In: *2018 Annual Reliability and Maintainability Symposium (RAMS)*. 2018, pp. 1–6. DOI: 10.1109/RAM.2018.8463107.

[11] Ram S. Jakhu, Yaw Otu M. Nyampong, and Tommaso Sgobba. "Regulatory framework and organization for space debris removal and on orbit servicing of satellites". In: *Journal of Space Safety Engineering* 4.3 (2017), pp. 129–137. ISSN: 2468-8967. DOI: https://doi.org/10.1016/j.jsse.2017.10.002.

[12] David A. Whelan et al. "DARPA Orbital Express program: effecting a revolution in space-based systems". In: *Small Payloads in Space*. Ed. by Brian J. Horais and Robert J. Twiggs. Vol. 4136. International Society for Optics and Photonics. SPIE, 2000, pp. 48–56. DOI: 10.1117/12.406656.

[13] Walker J. "Fact Sheet: Defense Advanced Research Projects Agency." In: Orbital Express, 2007, March.

[14] Putbrese et al. "When will on-orbit servicing be part of the space enterprise?" In: *Acta Astronauta* (2016). ISSN: 0094-5765. URL: http://hdl.handle.net/1721.1/118431.

[15] Vicki Cox. *Mission Extension Vehicle: Breathing Life Back Into In-Orbit Satellites*. 2020. URL: https://news.northropgrumman.com/news/features/mission-extension-vehicle-breathing-life-back-into-in-orbit-satellites.

[16] Jason Rainbow. *Northrop's MEV-2 servicer closing in on Intelsat-10-02 docking attempt*. 2021. URL: https://spacenews.com/mev-2-servicer-closing-in-on-intelsat-10-02-docking-attempt/.

[17] Daniel Hastings and Carole Joppin. "Evaluation of the Value of the Flexibility Offered by On-orbit Servicing: Case of Satellite Upgrade". In: *AIAA Space 2003 Conference & Exposition*. DOI: 10.2514/6.2003-6366. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2003-6366.

[18] Donald J. Kessler and Burton G. Cour-Palais. "Collision frequency of artificial satellites: The creation of a debris belt". In: 83.A6 (June 1978), pp. 2637–2646. DOI: 10.1029/JA083iA06p02637.

[19] Nodir Adilov, Peter J. Alexander, and Brendan M. Cunningham. "An economic "Kessler Syndrome": A dynamic model of earth orbit debris". In: *Economics Letters* 166 (2018), pp. 79–82. ISSN: 0165-1765. DOI: https://doi.org/10.1016/j.econlet.2018.02.025.

[20] National Aeronautics and Space Administration (NASA). "Orbital Debris Quarterly News". In: (2009). URL: https://www.orbitaldebris.jsc.nasa.gov/newsletter/pdfs/odqnv13i2.pdf.

[21] Wang Dongfang, Pang Baojun, and Xiao Weike. "GEO space debris environment determination in the earth fixed coordinate system". In: *PR China* (2017). URL: https://conference.sdo.esoc.esa.int/proceedings/sdc7/paper/304/SDC7-paper304.pdf.

[22] Darren S. McKnight and Frank R. Di Pentino. "New insights on the orbital debris collision hazard at GEO". In: *Acta Astronautica* 85 (2013), pp. 73–82. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2012.12.006.

[23] R. Jehn, V. Agapov, and C. Hernández. "The situation in the geostationary ring". In: *Advances in Space Research* 35.7 (2005). Space Debris, pp. 1318–1327. ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2005.03.022.

[24] G.J. Dittberner, J.S. Huth M.L. Fudge, and D.S. McKnight N.L. Johnson. "Examining simplifying assumptions of probability of collisions in LEO". In: *Proceedings of the First European Conference on Space Debris*. 1993, pp. 485–489.

[25] Donald J. Kessler et al. "Aerospace: Collision avoidance in space: Proliferating payloads and space debris prompt action to prevent accidents". In: *IEEE Spectrum* 17.6 (1980), pp. 37–41. DOI: 10.1109/MSPEC.1980.6330357.

[26] Markus Wilde, Jan Harder, and Enrico Stoll. "Editorial: On-Orbit Servicing and Active Debris Removal: Enabling a Paradigm Shift in Spaceflight". In: *Frontiers in Robotics and AI* 6 (2019). ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00136.

[27] Stephen Ornes. "Bringing down the trash". In: *Physics World* 25.06 (June 2012), p. 28. DOI: 10.1088/2058-7058/25/06/38.

[28] G. Borelli et al. "Preparation of enabling space technologies and building blocks: GNC and Robotic Arm Combined Control. TN1: Scenario definition and GNC requirements". In: (2021).

[29] Bruno Siciliano et al. *Robotics Modelling, Planning and Control*. Springer, 2009.

[30] Andrea Antonello. "Design of a robotic arm for laboratory simulations of spacecraft proximity navigation and docking". 2016.

[31] Jamshed Iqbal, R Ul Islam, Hamza Khan, et al. "Modeling and analysis of a 6 DOF robotic arm manipulator". In: *Canadian Journal on Electrical and Electronics Engineering* 3.6 (2012), pp. 300–306.

[32] Carlo Menon et al. "Free-Flying Robot Tested on Parabolic Flights: Kinematic Control". In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 28 (July 2005), pp. 623–630. DOI: 10.2514/1.8498.

[33] Zhang X and Liu J. "Autonomous trajectory planner for space telerobots capturing space debris under the teleprogramming framework". In: (2017). DOI: 10.1177/1687814017723298.

[34] Wenfu Xu et al. "Hybrid modeling and analysis method for dynamic coupling of space robots". In: *IEEE Transactions on Aerospace and Electronic Systems* 52.1 (2016), pp. 85–98. DOI: 10.1109/TAES.2015.140752.

[35] Zhai Guang, Zheng Heming, and Bin Liang. "Attitude Dynamics of Spacecraft with Time-Varying Inertia During On-Orbit Refueling". In: *Journal of Guidance, Control, and Dynamics* 41.8 (2018), pp. 1744–1754. DOI: 10.2514/1.G003474.

[36] M. Oda and Y. Ohkami. "Coordinated control of spacecraft attitude and space manipulators". In: *Control Engineering Practice* 5.1 (1997), pp. 11–21. ISSN: 0967-0661. DOI: https://doi.org/10.1016/S0967-0661(96)00202-X.

[37] Zong L., Luo J., and Wang M. et al. "Parameters concurrent learning and reactionless control in post-capture of unknown targets by space manipulators". In: *Nonlinear Dynamics* 96.1 (2019), pp. 443–457. DOI: https://doi.org/10.1007/s11071-019-04798-w.

[38] R. LONGMAN, R. LINDBERG, and M. ZEDD. "Satellite mounted robot manipulators - New kinematics and reaction moment compensation". In: *Guidance, Navigation and Control Conference*. DOI: 10.2514/6.1985-1885.

[39] Evangelos Papadopoulos and Steven Dubowsky. "Coordinated manipulator/spacecraft motion control for space robotic systems." In: *ICRA*. 1991, pp. 1696–1701.

[40] Canadian Space Agency. *Canadarm, Creation of the first Canadian robotic arm, flight history, comparison with Canadarm2 and Canadarm3*. URL: https://www.asc-csa.gc.ca/eng/canadarm/.

[41] Bruce A. Aikenhead, Robert G. Daniell, and Frederick M. Davis. "Canadarm and the space shuttle". In: *Journal of Vacuum Science & Technology A* 1.2 (1983), pp. 126–132. DOI: 10.1116/1.572085.

[42] Michael Hiltz et al. "Canadarm: 20 years of mission success through adaptation". In: *International Symposium on Artificial Intelligence, Robotics and Automation*. JSC-CN-6877. 2001.

[43] Juergen Telaar et al. "Coupled control of chaser platform and robot arm for the e. deorbit mission". In: *10th Int. ESA conference on Guidance Navigation and Control Systems (GNC), May 2017*. 2017, p. 4.

[44] Z. Pavanello et al. "Combined Control and Navigation Approach to the Robotic Capture of Space Vehicles". In: (2021). URL: https://hdl.handle.net/11311/1189477.

[45] Gunter D. Krebs. "OSAM 1 (Restore-L)". In: *Gunter's Space Page* (2023). URL: https://space.skyrocket.de/doc_sdat/osam-1.htm.

[46] ESA. "The RemoveDebris ADR Mission: Preparing for an International Space Station Launch". In: 7 (2017). URL: https://conference.sdo.esoc.esa.int/proceedings/sdc7/paper/173.

[47] Chris Blackerby et al. "ELSA-d: An In-Orbit End-of-Life Demonstration Mission". In: *Proc. Int. Astronaut. Congr. IAC*. Vol. 6. 2018, p. 43644.

[48] Mario E. Salgado, Graham C. Goodwin, and Stefan F. Graebe. *Control System Design*. Ed. by Pearson College Div. 2000. URL: http://caaelotel.elo.utfsm.cl/home/wp-content/uploads/Control-System-Design-SalgadoGoodwinGraebe.pdf.

[49] "Identification and Model Reduction Techniques". In: *Practical PID Control*. London: Springer London, 2006, pp. 165–207. ISBN: 978-1-84628-586-8. DOI: 10.1007/1-84628-586-0_7. URL: https://doi.org/10.1007/1-84628-586-0_7.

[50] S. Miller et al. *Modeling Flexible Bodies with Simscape Multibody Software*. URL: https://it.mathworks.com/campaigns/offers/model-flexible-bodies.html.

[51] Rached El Fatmi and Hatem Zenzri. "On the structural behavior and the Saint Venant solution in the exact beam theory: Application to laminated composite beams". In: *Computers Structures* 80.16 (2002), pp. 1441–1456. ISSN: 0045-7949. DOI: https://doi.org/10.1016/S0045-7949(02)00090-1. URL: https://www.sciencedirect.com/science/article/pii/S0045794902000901.

[52] Stefano Lenci. *Lezioni di MECCANICA STRUTTURALE*. Ed. by Pitagora Editrice Bologna. 2004, pp. 243–268.

[53] Alexander Mielke. "Saint-Venant's problem and semi-inverse solutions in nonlinear elasticity". In: *Archive for Rational Mechanics and Analysis* 102 (1988), pp. 205–229.

[54] J. Thomas and B.A.H. Abbas. "Finite element model for dynamic analysis of Timoshenko beam". In: *Journal of Sound and Vibration* 41.3 (1975), pp. 291–299. ISSN: 0022-460X. DOI: https://doi.org/10.1016/S0022-460X(75)80176-3. URL: https://www.sciencedirect.com/science/article/pii/S0022460X75801763.

[55] Alessandro Tasora. "Euler-bernoulli corotational beams in chrono:: Engine". In: *Chrono:: Engine technical documentation* (2016).

[56] MathWorks. *Flexible Rectangular Beam*. URL: https://it.mathworks.com/help/sm/ref/flexiblerectangularbeam.html.

[57] Xuli Han and Xiao Guo. "Cubic Hermite interpolation with minimal derivative oscillation". In: *Journal of Computational and Applied Mathematics* 331 (2018), pp. 82–87. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2017.09.049. URL: https://www.sciencedirect.com/science/article/pii/S0377042717304910.

[58] W. Khalil and M. Gautier. "Modeling of mechanical systems with lumped elasticity". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 4. 2000, 3964–3969 vol.4. DOI: 10.1109/ROBOT.2000.845349.

[59] James W. Cooley, Peter A. W. Lewis, and Peter D. Welch. "The Fast Fourier Transform and Its Applications". In: *IEEE Transactions on Education* 12.1 (1969), pp. 27–34. DOI: 10.1109/TE.1969.4320436.

[60] James W Cooley and John W Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[61] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898718027. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718027.

[62] Lizhe Tan and Jean Jiang. "Chapter 2 - Signal Sampling and Quantization". In: *Digital Signal Processing (Third Edition)*. Ed. by Lizhe Tan and Jean Jiang. Third Edition. Academic Press, 2019, pp. 13–58. ISBN: 978-0-12-815071-9. DOI: https://doi.org/10.1016/B978-0-12-815071-9.00002-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128150719000026.

[63] Edmund Lai. "2 - Converting analog to digital signals and vice versa". In: *Practical Digital Signal Processing*. Ed. by Edmund Lai. Oxford: Newnes, 2003, pp. 14–49. ISBN: 978-0-7506-5798-3. DOI: https://doi.org/10.1016/B978-075065798-3/50002-3. URL: https://www.sciencedirect.com/science/article/pii/B9780750657983500023.

[64] MathWorks. *Choose a solver*. URL: https://it.mathworks.com/help/simulink/ug/choose-a-solver.html.

[65] MathWorks. *Choose a Jacobian Method for an Implicit Solver*. URL: https://it.mathworks.com/help/simulink/ug/choose-a-jacobian-method-for-an-implicit-solver.html.

[66] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[67] MathWorks. *Fixed Step Solvers in Simulink*. URL: https://it.mathworks.com/help/simulink/ug/fixed-step-solvers-in-simulink.html.

[68] MathWorks. *fft - Fast Fourier Transform*. URL: https://it.mathworks.com/help/matlab/ref/fft.html.

[69] MathWorks. *External Force and Torque*. URL: https://it.mathworks.com/help/sm/ref/externalforceandtorque.html.

[70] MathWorks. *Transform Sensor*. URL: https://it.mathworks.com/help/sm/ref/transformsensor.html.

[71] MathWorks. *Revolute Joint*. URL: https://it.mathworks.com/help/sm/ref/revolutejoint.html.

[72] MathWorks. *Rigid Transform*. URL: https://it.mathworks.com/help/sm/ref/rigidtransform.html.

[73] Bruno Siciliano et al. *Robotics Modelling, Planning and Control*. Springer, 2009, pp. 91–94.

[74] MathWorks. *Cylindrical Solid*. URL: https://it.mathworks.com/help/sm/ref/cylindricalsolid.html.

[75] J Martin Bland and Douglas G Altman. "Statistics notes: measurement error". In: *Bmj* 312.7047 (1996), p. 1654.

[76] MathWorks. *std - standard deviation*. URL: https://it.mathworks.com/help/matlab/ref/std.html#d124e1492879.

[77] P. Bogacki and L.F. Shampine. "A 3(2) pair of Runge - Kutta formulas". In: *Applied Mathematics Letters* 2.4 (1989), pp. 321–325. ISSN: 0893-9659. DOI: https://doi.org/10.1016/0893-9659(89)90079-7. URL: https://www.sciencedirect.com/science/article/pii/0893965989900797.