Politecnico di Milano

Facoltà di Ingegneria Industriale

Corso di Laurea Specialistica in Ingegneria Spaziale

"Dynamics and Control of a Spacecraft-Manipulator System: Analisys, Simulation and Experiments"

Relatore: Prof. Franco BERNELLI ZAZZERA

Correlatori: Ing. Sabrina CORPINO
Prof. Marcello ROMANO

Edoardo SERPELLONI Matr. 734625

Anno Accademico 2010/2011

To Annachiara, for her love,
To my Parents, for their inconditionate support,
To Marianna, Mario and my amazing nieces,
To all my friends, and in particular to Sara, Cri, Chips, Elena, Bonny, Lore and Ila for their friendship,
To my roommates Sassut, Il Mazzo and Andre.

I also want to thank Professor Bernelli and Professor Romano, that gave me the possibility to live a great experience at the Naval Postgraduate School, from the professional and personal point of view. A particular thank goes to Octavio ,Marco and Alessio, the research team of the Spacecraft Robotics Laboratory at the Naval Postgraduate School.

Contents

List of Figures

List of Tables

List of Symbols

Sommario

La possibilita di avere un manipolatore a bordo di un veicolo spaziale puo espandere enormemente le capacita di un satellite, aprendo nuovi ed interessanti scenari di applicazione. La missione ETSVII della JAXA ha dimostrato la fattibilita del rendez-vous e docking tra due satelliti per mezzo di un braccio robotico. Mentre negli anni passati la ricerca accademica si e' concentrata sullo studio di manipolatori montati su veicoli spaziali non controllati, in questa tesi e' stato studiato il caso concernente il pieno controllo di satellite e braccio robotico.Lo scopo della tesi e' stato l'implementazione di un simulatore Simulink del sistema satellite-manipolatore e la progettazione di un manipolatore planare da montare su un robot flottante allo Spacecraft Robotics Laboratory, alla Naval Postgraduate School, Monterey, California. Particolare attenzione e' stata posta alla modellazione della cinamatica, della dinamica e al controllo del sistema. Il *Direct Path Method* e' stato implementato per calcolare in modo efficiente lo Jacobiano del sistema completo. Per la risoluazione della ridondanza del sistema e' stata testata la *Simple Jacobian Inversion Technique* e un algoritmo di *Joint Limit Avoidance* e' stato implementato. La dinamica del sistema viene calcolata ad ogni passo di integrazione usando il*Newton-Euler Algorithm*, implementato usando il formalismo della *Spatial Algebra* . Questa formulazione permette di studiare il sistema completo come un manipolatore a base fissa, il cui primo link e' il satellite stesso, connesso a terra usando un giunto a 6 DOFs. Nella restrizione bidimensionale il controllo della base e' stato realizzato con una tecnica di Pulse Width Modulation, per poter simulare il reale sistema di controllo di un satellite dotato di attuatori a getto. La tecnica di controllo implementata e' il *Computed Torque Control*, una legge di controllo che permette una *Feedback Linearization*, che significa che, se i parametri del sistema sono noti con accuratezza, il sistema di controllo linearizza il sistema in ciclo chiuso. Infine e' stato progettato e realizzato un prototipo di manipolatore per future attivita sperimentali. Il prototipo realizzato e' un manipolatore planare, composto da giunti di rivoluzione e caratterizzato dall'unione di giunti modulari, realizzati mediante l'utilizzo di una stampante 3D.

Parole Chiave: Robotica Spaziale, Manipolatore abase mobile, Spatial Algebra, Computed Torque Control, Manipulator Design

Abstract

Having a manipulator mounted on a spacecraft can enormously expand the capabilities of a satellite, opening new scenarios of application. JAXA ETS VII mission proved the feasibility of rendez-vous and docking between spacecrafts using a robotic arm. While most of the research in the past years have focused on the study of manipulators mounted on non controlled spacecrafts, in this thesis the case of completely controlled spacecraft and manipulator has been studied. The goal of the thesis is the implementation of a Simulink simulator for a spacecraft-manipulator system and the realization of a planar manipulator mounted on a free flying robot , at the Spacecraft Robotics Laboratory at Naval Postgraduate School,Monterey, California. Particular attention has been posed on the fields of kinematics redundancy solution, dynamics modeling and control techniques.
The *Direct Path Method* has been implemented in order to compute in an efficient way the Jacobian matrix of the complete system. In order to solve the redundancy of the system the *Simple Jacobian Inversion Technique* and a *Joint Limit Avoidance Algorithm* have been implemented. The dynamics of the system is calculated step by step using the *Newton-Euler Algorithm*, implemented using the *Spatial Algebra* formulation. This implementation allows to study the complete system as a fixed base manipulator system, whose first link is the spacecraft itself, fixed to a virtual ground using a 6 DOFs joint. In the 2D case the control of the spacecraft has been realized applying a Pulse Width Modulation in order to simulate the real attitude control techniques based on jet thrusters actuators.
The implemented control technique is the *Computed Torque Control*, a typical *Feedback Linearization Technique*, that means that, if the information on the system parameters and the dynamic model of the system is enough accurate, the control law linearizes the behavior of the system.
Finally a manipulator prototype for future experimental activities has been designed and realized. The realized manipulator, is a planar manipulator, composed by modular revolute joints, produced using a rapid prototyping 3D printer.

Keywords: Space Robotics, Mobile Manipulator, Spatial Algebra, Computed Torque Control, Manipulator Design

Chapter 1

Introduction

1.1 Spacecraft Mounted Manipulators

Nowadays robotic manipulators are extensively used in the industrial and surgical fields. The progresses in the fields of kinematic and dynamic modeling and the design of new , innovative control techniques have allowed a deeper level of automation in activities that were performed only by humans. The industrial field pushed the research on heavy, powerful manipulators, able to manipulate heavy objects or to handle dangerous materials. In the automotive industry most of the assembly phase is now performed using robotic systems. On the other side the surgical field pushed for the development of teleoperation,telepresence and haptics and on the development of extremely high precision control techniques.
Considering the results achieved, the application of robotic arms in the space environment can be considered natural. The space frontier offers a great variety of applications for manipulators: from the repair of damaged structures (also the self-repair), to the refuelling of a satellite (using the manipulator as a fuel pump); from the rendez-vous and docking maneuvers to the manipulation and the orbiting of objects in space.
On the other side the development of a space manipulator represents a huge challenge for the designer: he has to face very strict power, weight and bulk constraints, the computational resources for the management of the arm movements are low and all the components have to be space certified.
Robotic arms have already been applied to the space field in the past both in orbit and for planetary exploration.
This thesis is focused in particular on the case of the on-orbit scenario. the two most outstanding results in this field are represented by the Canadarm (Space Shuttle) and by the JAXA Mission ETS VII.

1.1.1 Canadarm

The Canadrm is a Robotic Manipulator Mounted on the Space Shuttle, in order to deploy and release payloads from its cargo bay.It was launched the first time on November 13, 1981 on the STS-2 Mission. The Canadarm is a six degree of freedom robotic arm, it is 15.2 meters long, and 38 centimeters in diameter. It weights 450 kg. Compared to the total weight of the Space Shuttle (2030 tonns), the manipulator is characterized by an arm to base ratio of 2.21 10^{-4}.

Figure 1.1: The Space Shuttle Canadarm.

At the beginning of its life it could manipulate objects weighing up to 32.5 tonns, successive test and improvements of the control system increased that value to 293 tonns. It is curious to notice that, despite the on-orbit capabilities, the arm can not lift its own weight while it is on the ground on Earth. The Canadarm configuration consists of a display and control panel, including hand controllers for its translational and rotational dynamics and of a manipulator-on board computer interface. The astronauts can see the results of their inputs using the Advanced Space Vision System, close to the controllers.

1.1.2 ETS VII Mission

The ETS VII, also known as KIKU-7 (launched on November 28th, 1997) was the world's first satellite to be equipped with a robotic arm (6 DOFs), and also the first unmanned spacecraft to perform autonomous rendezvous and docking operations successfully. The ETS-VII consisted of two main parts: a chaser satellite and a target satellite. The chaser satellite is the main satellite body, and was named Hikoboshi. A 2 m long robotic arm was attached to this part. The smaller target satellite was named Orihime. The box shaped, complete satellite system weighed 2,860 kg.Three rendezvous and docking operations were carried out with the ETS-VII, which involved placing the target satellite 200 mm away from the chaser and using the robot arm to retrieve and hold it in place. Several other experiments were also carried out with the satellite's robotic components. The experiments on the rendezvous and docking have been completed both autonomously and operated from the ground.

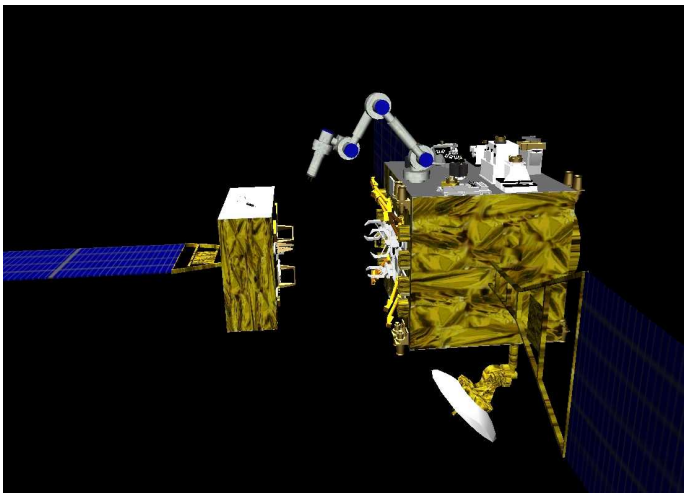


Figure 1.2: ETS VII Mission manipulator.

1.1.3 Manipulators Applications in Space

A manipulator mounted on a spacecraft can improve the number and the quality of the interactions that a spacecraft can perform on the environment, on itself and on other spacecrafts. One of the most attractive applications is the rendez-vous and docking capability (named *berthing*, if performed using a robotic arm). Using a robotic arm could improve the reliability of the operation: in this way the two docking spacecrafts could enter in contact in a more controlled and soft way, avoiding the risks involved in such complex operations.

An extremely attractive idea is to use the manipulator to repair the spacecraft itself and other target spacecrafts. The typical scenario could be the maintenance of a satellite constellation: the presence on each orbit of the constellation of a satellite equipped with tools able to perform accurate analysis of the damages occurred on other satellites could allow to save money for the constellation maintenance. At the same time the capability of analysis and repair could be applied on the spacecraft itself: during long interplanetary missions the capability of the spacecraft to perform analysis on its status could reveal crucial for the success of the mission.
Other possible applications are related to the capability of manipulating objects: during missions finalized to the scientific analysis of sampled objects (for example space debris in LEO) a robotic arm is a possible solution to the critical problem of samples grasping and storing.

1.1.4 Research Focus

Most of the research on spacecraft mounted manipulators is focused on the free floating case: that means that the spacecraft itself is not controlled. In this condition the system becomes extremely challenging to study from the point of view of dynamics: in these case the system is non-holonomic and this causes the birth of dynamic singularities and limited workspaces. These consequences strongly pushed , in the past, the research in the field of space manipulators. On the other side, thinking to the final application on space systems, it can be considered too risky to use a manipulator during proximity operations without controlling the spacecraft to which it is fixed. The high required levels of accuracy and reliability, forces to implement a full control of the spacecraft. Considering the costs and the efforts spent in a mission development, a non controlled spacecraft during extremely delicate missions seems to be unapplicable. With the controlled spacecraft, the system does not show dynamic singularities and can be treated as conventional problem of dynamics modeling and control of a multi-body system. The thesis research is focused on the kinematic and dynamic modeling and control techniques applicable in the case of full control of the spacecraft. The research is finalized to the implementation of a Simulink simulator of the system and to the design of an experimental setup useful for future algorithms' validation. The simulator has to allow the user to define the desirable trajectory of the end effector and, then, it has to compute the joints motion that permits to the whole spacecraft-manipulator system to follow that trajectory. The attention has also been focused on the implementation of an algorithm of joints limits avoidance in order to avoid the problem of possible compenetrations between rigid bodies. In this way the user provides to the system the knowledge about system geometry and bulk, that is translated into constraints for the joint movements. The computation of the dynamic matrices has been implemented using a Spatial Algebra formulation of the Newton Euler Algorithm. In this way it is possible to compact

the formulation of the system's dynamic equations. Another advantage of the Spatial Algebra approach is that in this way it is simple to model the motion of an unconstrainted rigid body in the three dimensional case using the same formulation used to model the other joints of the system. This allows to model the system as a fixed base manipulator, where the first link of this virtual manipulator is the spacecraft itself, fixed to the ground by a 6 DOFs joint. The system is then controlled using a Computed Torque Control Law, that actively uses the dynamic matrices of the system to compute the forces and torques that have to be applied to control the system motion.
After the realization of the simulator, the manipulator itself has been designed. The output of the design phase has been a modular manipulator, composed by modular revolute joints that can be added or removed from the system in order to increase or decrease the number of desired DOFs.

The research main contribution is the implementation of a simple, highly expandible architecture for the simulation of a spacecraft mounted manipulator-system (and, more in general, for the simulation of linear chains of rigid bodies). The spatial algebra allows to reduce the computational cost of the algorithm,and it permits to implement an high number of different joints/constraints (systems with more than ten DOFs can be simulated in a quasi-real-time environment). The implementation can be expanded to create a real multi-body software. The fixed base interpretation permits to implement control laws traditionally applied to fixed base manipulators. The experimental validation on the spacecraft control is a first partial validation of the validity of the implementation.

1.1.5 Thesis Structure

The chapters organization reflects at the same time the logic flux of information in the software and the chronological development of the software.
Chapter 2 presents the results obtained in the past years in the field of manipulators and in particular in the case of spacecraft-manipulators systems.
In the Chapter 3 a complete overview of the developed software is presented, organized in the main block schemes that compose the software.
Chapter 4 introduces the algorithms for the computation of the Jacobian Matrix and the implemented methods for the inverse kinematics solution. Particular attention has been dedicated to the analysis and implementation of a Joint Limit Avoidance Algorithm.
Chapter 5 presents a brief introduction to Spatial Algebra. The basic notions are presented, in order to use this tool in the implementation of a dynamic algorithm function and the modeling of manipulators joints.
Chapter 6 explains the basic notions in manipulators modeling. The implemented joint models are presented and the used reference frames are defined.
In Chapter 7 the implementation of the Newton-Euler Algorithm is presented. The implemenetation has been validated using a simple test case, both in the 2D

and 3D case.
In Chapter 8 the implemented control law is presented and tested using two different test cases: a 4 DOFs planar manipulator on a 3 DOFs spacecraft, with and without thrusters Pulse Width Modulation, and a 4 DOFs spatial manipulator on a full controlled 6 DOFs spacecraft, in the case of full control and control using only a partial computation of the inertia matrix.
Chapter 9 presents the implementation of a virtual environment for the simulation, using the Virtual Reality Toolbox.
Chapter 10 presents the design and realization of the planar manipulator to mount on one of the free flying - free floating robot of the Spacecraft Robotics Laboratory at the Naval Postgraduate School. The first preliminary results obtained applying the developed software to the case of spacecraft control.
The final Chapter 11 summarizes the results obtained in the numerical simulations and the first preliminary results obtained in the control of the spacecraft.

Chapter 2

State of the Art

Nowadays manipulation is an extremely active field of research. The reason of such a success is mostly related to the fact that a lot of fields can offer important applications to robotic arms. In general robotic manipulators have been used to perform operations in dangerous environments (underwater manipulators or manipulators for nuclear reactors maintenance) or situations (for example the US Army's Warrior X700, [1]), or to perform operations in a more safe and precise way (surgery and assembly lines in the automotive field). Also the improvements in the field of humanoid robots pushed the research on manipulation and grasping toward new frontiers.
One of the most emerging field is related to the realization of mobile base manipulators: that means that the manipulator is mounted on a mobile base. The research on mobile manipulators has focused on the cases of wheeled robots, underwater robots and spacecrafts mounted manipulators.
One of the most relevant applications for manipulators mounted on wheeled robots is the field of war robotics with special attention to robots specialized in blastering. In [1], blastering robots performances on the battle field are presented (related to Iraq and Afghanistan wars). These robots can blaster a bomb operating with the on board manipulator: these applications show a particular focus on robust and reliable design of the manipulator itself and a particular attention toward the effectiveness of the remote teleoperation technique used to drive the robot. Non military research is mostly focused on innovative control techniques (in [2], for example, the system's path planning and control is finalized to stay, if possible, as closest as possible to an optimal configuration during its motion) and on the modeling of the effects of the vehicle's suspensions on the system ([3])

The case of a manipulator fixed to an underwater robot can be considered very similar to the case of a spacecraft mounted manipulator: indeed pools are often used in the astronauts training in order to simulate the weightlessness, condition typical of the space environment. On the other side, in some applications, the role of the water's drag force can be critical. The presence of a viscous fluid all

around the robot introduces the effects of an external non linear force acting on each point of the system depending on its orientation and its local velocity and on the fluid mechanical properties and motion status. Because of that, most of the research efforts have been done in order to create models of the water drag force on each link and body of the system able at the same time to perform accurate and fast computations. McMillan, Orin, D.E. and McGhee, R.B. in [4] developed efficient simulations that allowed to compute all the hydrodynamic interactions of the robot with the surrounding fluid, only doubling the computational cost of the same algorithm but without the simulation of the hydrodynamic interactions. Most of the control laws and philosophies used to control underwater manipulators can be applied to the control of a spacecraft-manipulator system (particularly used in this case are the feedback linearization techniques, [5]).
On the other side the real case introduces big uncertainties in the system, such as the currents and waves disturbance effects. These effects can be large and they can strongly influence the system behavior. To face these problems control laws have been developed in order to control the system in a more robust way, facing sudden variations in the environment conditions (in [6] adaptive algorithms have been implemented to face sudden changes in the environment). This represents one of the most critical differences between underwater and space robotics: in the first case the environment can be more dynamic, and act on the system in a more complex and unpredictable way. For the underwater case the possible applications of such systems are related to inspection, maintenance, repair and service work on underwater installations, [5]. As for surgery and mobile ground manipulators a particular attention has been given to teleoperation control techniques ([7], [1]).

The field of spacecraft mounted manipulators modeling and control has been extensively studied in the '80s and early '90s, in particular at Massacchussets Institute of Technology (in particular by S. Dubowsky, E. Papadopoulos and Z. Vafa) , Stanford University (at the Aerospace Robotics Laboratory) and at Tohoku University (in particular K. Yoshida). The research on spacecraft-manipulator systems is quite a new field, usually research was performed on fixed base manipulators, working on low friction tables in order to simulate a microgravity environment. The studied systems can be classified into three different categories, related to the control of the spacecraft itself:

- Full control of both the arm and the spacecraft translation and attitude (*free flying* system)
- Full control of the arm, but spacecraft controlled only in attitude, not in translation
- Full control of the arm, no control of the spacecraft (*free floating* system).

The research has focused for long time mostly on free floating systems. This condition is particularly interesting because it allows to explore particular dynamic features that are not present in the controlled cases. In particular at MIT,

Dubowsky and Vafa developed the Virtual Manipulator approach ([8], [9], [10], [11]), that permits to model a free floating manipulator as a fixed base manipulator. The *Virtual* manipulator is a fixed base manipulator with base in the system's center of mass. If the system's attitude and translation are not controlled and if the system is not subjected to external forces, the center of mass of the system maintains its initial position (if its initial velocity is null). This allows to use the barycenter of the system as a fixed base for a manipulator whose links' geometry is directly related to the geometrical and inertial properties of the system and whose joints rotations are exactly the same as the real joints rotations, in the case of revolute joints. This approach can be also applied to the case of the control only of the attitude of the spacecraft, if performed using actuators that only exchange internal forces with the rest of the system (gyros and reaction wheels are the most used attitude control systems that do not violate this constraint).

Torres and Dubowsky, and then Papadopoulos and Yoshida and Umetani underlined for the free floating case the possible occurrance of dynamic singularities. A free floating system is a non-holonomic mechanical system, so the evolution of the joints motion is directly dependent on the previous motion history of the system. A direct consequence for these systems is the fact that a closed trajectory in the joint space does not produce a closed trajectory in the task-cartesian space. So,some configurations of the manipulator could become singular thanks to the joints motion history. In this case, the Jacobian matrix of the system turns to be singular, and the system is forced to accomplish large joint movements in order to perform small displacements in the task space.

Torres, in [12], [13],[14],[15], defined the *Enhanced Disturbance Map*, that is a tool that allows to map the manipulator workspace using as parameter the attitude disturbance on the spacecraft generated by the motion of the arm itself. He underlined that, given two points A and B both inside the *Reachable Workspace* of the manipulator, where A is the initial end effector position and B is the final desired end effector position, a path from A to B can be found that causes not attitude disturbance on the spacecraft base.

In the field of mobile manipulators kinematics, in [16] and in [17], Papadopoulos and Moosavian developed new algorithms for the system's Jacobian computation in the case of a non fixed base manipulator. In particular the *Barycenter Vector Approach* (presented in [18] together with dynamics modeling and control techniques) and *Direct Path Method* ([17]) have been extensively used. The firsts is more expansive from the computational point of view and it permits to map the desired end effector velocity into the correspondent joints velocity and linear and angular velocity of the system's center of mass. The latter is cheaper from the computational point of view and permits to map the desired end effector velocity into the correspondent joints velocity and linear and angular velocity of

the spacecraft. Both the methods can be easily applied to the case of multiple mounted manipulator.
Most of the research in kinematics is focused on redundancy solution techniques ([19],[20]). A redundant manipulator is a manipulator whose joint space has an higher dimension if compared to the task dimension. This causes to have additional joints variables to exploit in order to accomplish additional and secondary tasks. The research in this field is extremely active because it has a great spectrum of possible applications also in the industrial and medical-surgical field. This class of algorithms permits to implement routines of joint limit avoidance, obstacle avoidance, power limit avoidance etc. In the industrial field non redundant manipulators were preferred: the redundancy greatly complicates the system and it increases the computational cost required to compute the joint trajectories from the desired end effector trajectory. Some routines require complex optimization procedures, that does not allow such algorithms to be directly applied on real systems. Nowadays redundancy is seen as an additional tool in order to generate more accurate and coherent joint motions.
The research has focused in particular on the classes of the Joint Limit ([21] and [22]) and Obstacle Avoidance Algorithms ([23], [24]). The first class of algorithms focuses on finding a redundancy solution that respect the physical, practical, boundaries on the joint motion. This class of algorithms is the only one that has a knowledge of the system configuration: all the other routines can produce joints trajectories that can led to compenetrations of the bodies of the system. Generally the boundaries on the joints are decided by the user himself, that is supposed to have the required knowledge and experience in order to indicate the allowable joint motions. The second class of algorithms focuses on the generation of joint trajectories in order to avoid the contact with obstacles. Obviously not only the end effector is supposed to avoid the obstacles, but also all the joints and the links of the system. Some of these algorithms require the knowledge of obstacles positions, while other implementations allow the system to face unpredictable obstacle positions ([23]).
In the field of dynamics, Featherstone in [25], [26] provided an original contribution in the solution of the direct and inverse dynamics. He devloped a new, original, mathematical tool, named *Spatial Vector Algebra*. This approach is based on the definition of 6D motion and force vectors that contain both the information about the linear and angular velocity and the information about the forces and torques. This allows to greatly compact the equations and the computations volume necessary for the solution of the inverse and direct dynamics. The Spatial Algebra permits to translate the classical algorithms for the computation of dynamics in a more compact and clear way. This permits also to create a constraints/joints library, in order to model all the possible types of interaction between rigid bodies. In a more general view this approach is applicable not only on manipulator systems, but also to parallel systems and systems characterized by closed kinematic

loops. A particular feature of the Spatial Algebra is the modelization of a 6 DoFs joint: this allows to introduce into the system free rigid bodies and, in the case of a spacecraft mounted manipulator, to model the movements of the base as a normal joint of a system fixed to a virtual, hypothetical ground.

A lot of work has been done to explore the topic of teleoperation related to space application. In particular Wang, Liang and Li , in [27], developed the entire software and hardware architecture in order to study teleoperation techniques. Their work revealed to be original also for the implementation of the experimental platform: they were able to simulate the motion of two spacecrafts mounted manipulators (one to each spacecraft), using only fixed base manipulators. Stoll and collaborators, are conducting research on telepresence and teleoperation techniques at the DLR, finalized for future ESA (European Space Agency) missions ([28],[29]).

Experimental Setups

The most advanced laboratory for space manipulators test is the Aerospace Robotics Laboratory at Stanford University. The Laboratory is equipped with three free flyers robots with two manipulators (2 links each) each that float on a granite flat floor. Moreover the laboratory is equipped also with fixed base planar manipulators, floating on a flat surface. One of this manipulators has flexible links, while the other (Macro-Micro) has an original architecture: it is a large 2 links planar manipulator, equipped with an end effector realized by the union of 2 small 2 links planar manipulators. The Tohoku University developed an experimental setup for manipulators testing on free flying robots: their test bed consisted in a free floating robot, equipped with a 2 links planar arm. The testbed was finalized to preliminary tests for the JAXA mission ETS VII. Also the DLR (the German Aerospace Center) developed a system similar to the Tohoku University one. An alternative approach consists in using fixed base manipulators in order to simulate the behavior of spacecraft mounted manipulators, as done by Wang, Liang and Li at University. Many universities and research labs developed free flying tests robots (they are very attractive systems because they permit low cost experimentation of on orbit robotics systems) with application to rendez-vous and docking operations and formation flying (for example Naval Postgraduate School, University of Southern California, Georgia Institute of Technology).

Chapter 3

Overview of the System

In order to handle the problem of dynamics modeling and control of spacecraft mounted manipulators, both a software and an hardware platforms have be implemented. The software platform has to represent the first step in the validation of a new dynamics or control algorithm: the implemented software is supposed to take into account every aspect of the problem under study, such as kinematics, dynamics, control and results visualization. To create such a complete, representative and highly expandible software platform, a Simulink model of the system has been created. In this way it is possible to simulate all the processes from the generations of the joints trajectories, to the dynamics modeling, to the control law implementation. The choice of using Simulink derives from desire to create a software that could be easily enlarged and modified. Moreover, Simulink offers the chance to automatically generate a C code from the model to perform real time operations. Two versions of the software have been developed: a Simulink version, used to generate preliminary results and to perform the first simulation tests on the developed algorithms; and a compiled version, finalized to the generation of an executable file to run in real-time ont the robots on board computer, in order to validate the software results from the experimental point of view.
The hardware implementation in order to validate software results has to be accomplished with the design of a planar manipulator (the experimental test-bed at the Naval Postgraduate School was composed by four free flying robots, but without any manipulator fixed to them). The particular specifications for the manipulator design were related to the development of a modular architecture, in order to easily increase or decrease the tested number of joints. Since the target of thesis is related to the study of manipulators mounted on small satellites, the design of the manipulator itself has to take into account the strong volume and bulk constraints typical of the target application. One of the consequences is the need to accommodate all the electronics drivers inside the manipulator itself, in order to save space on board the satellite.

3.1 Simulator Architecture

Teleoperation Subsystem

This block introduces into the system the input generated by the operator using a joystick and, at the same time, it returns to the operator information about the system status in the case of bilateral teleoperation. The applied scheme consists in the classical unilateral or bilateral teleoperation scheme: the operator commands the end effector motion (providing the desired linear and angular velocity in a cartesian 3D space), while the planning of the trajectory of the rest of the system is completely demanded to system itself. In the case of this thesis the input will be introduced into the system using a joystick device. The input device used by the operator in order to impose the motion to the system is called *master*, while the teleoperated manipulator is called *slave* device. In the case of a planar manipulator a simple joystick represent the most simple solution as master device. In the more complex case of a spatial manipulator the master is usually a replica of the slave manipulator itself and the operator directly moves the end effector to decide the desired motion. Typical applications of this master-slave configuration is in the medical-surgical field.

Predictor

The purpose is to simulate a tipical space mission that involves the use of teleoperated manipulators. In the real case, a delay of second affects the communications beteween thr ground and the satellite. This communication delay strongly decreases the performances of an operator during tele-operations.
A possible way to fix the problem is the design of a *Predictor system*. A predictor consists in the mathematical model of the system, plus an accurate graphic interface. This permits to the operator to work in real time using only the predictor. In this way the performances of the operator are maximized, while the real system is working in time delay. On the other side, in order to minimize the errors between the commanded motion and the obtained motion, great efforts have to be done in order to create a very accurate model of the system. This approach cannot be applied in all these cases where the parameter of the system (masses, inertias, joint friction, flexibility, etc..) are affected by large uncertainties.

Inverse kinematics block

Given the desired motion of the end-effector, the Inverse Kinematics block allows the computation of the joints positions,velocities and accelerations that cause the desired end-effector trajectory. This means that, once the operator has completely determined the end effector motion, this information must be mapped from the cartesian space to the joint space. The inverse kinematics block could be divided

into two different sub-blocks

- *Jacobian calculation*, for the generation of the map from the joint motion to the cartesian end effector motion

- *Inverse mapping and redundancy solution*, for the inversion of the map and the solution of the redundancy of the robot arm.

To summarize, the input of the block is the desired end effector velocity, the output is the set of joint positions, velocities and accelerations that allow the desired end effector position, velocity and acceleration.

Dynamics matrices computation

The canonical equations of motion of a manipulator system is:

$$\mathbf{H}(\mathbf{q})\,\ddot{\mathbf{q}} \;+\; \mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) \;=\; \mathbf{Q} \tag{3.1.1}$$

Most of the trajectory tracking techniques require some knowledge about the system. In particular most of them require the computation of the dynamic matrices of the system, **H** and **C**. This block, using an Inverse Dynamics algorithm allows the computation of the Inertia and Coriolis matrices. The Newton-Euler algorithm has been implemented, in its formulation based on the use of the Spatial Algebra. Using this approach, the system can update the dynamic matrices step by step giving as input the joint position and velocities.

Control Law

Since the final goal is not simply the maintenance of an equilibrium point, but it is the tracking of a given complex real time trajectory , the system can not be linearized. As a consequence of this, a Non Linear Control Law has to be implemented in order to control the complex motion of the manipulator. As exaplained in detail in the following chapters, a Computed Torque control law has been used. Using this law the forces and torques applied on the system are:

$$\tau \;=\; \mathbf{H}(\,\ddot{\mathbf{q}}_{des} \;+\; \mathbf{K}_D\dot{\mathbf{e}} \;+\; \mathbf{K}_P\mathbf{e}) \;+\; \mathbf{C} \tag{3.1.2}$$

This control law operates a *feedback linearization* on the system. This means that if the computed dynamic matrices are very close to the real matrices of the system, applying the control law, the closed loop controlled system reduces to:

$$\ddot{\mathbf{q}} \;=\; \mathbf{u} \;=\; \ddot{\mathbf{q}}_{des} \;+\; \mathbf{K}_D\dot{\mathbf{e}} \;+\; \mathbf{K}_P\mathbf{e} \tag{3.1.3}$$

Forward Dynamics

Forward Dynamics consists in the direct integration of the equations of motion of the system, given the matrices and the forces and torques acting on the system. This block is required only for computer simulations. Indeed in the experimental case the Forward Dynamics role is played by the real system itself.While a large amount of algorithms have been created to perform an efficient calculation of the resulting joint motion, in the software implementation it has been adopted the most simple approach to problem:

$$\ddot{\mathbf{q}} = \mathbf{H}^{-1} (\mathbf{Q} - \mathbf{C}) \tag{3.1.4}$$

$$\dot{\mathbf{q}} = \int_{t_0}^{t} \ddot{\mathbf{q}}\, dt + \dot{\mathbf{q}}_0 \tag{3.1.5}$$

$$\mathbf{q} = \int_{t_0}^{t} \dot{\mathbf{q}}\, dt + \mathbf{q}_0 \tag{3.1.6}$$

Direct Kinematics

The direct kinematics block allows the computation of the actual position, velocity and acceleration of the end effector, given the angular positions, angular rates and angular accelerations of the joints. This allows to compare the desired commanded trajectory with respect to the actual trajectory of the end effector of the manipulator. The Jacobian matrix calculated in the Inverse Kinematics block can be used to perform the reconstruction of the end effector trajectory.

3.2 The Experimental Setup

3.2.1 The Experimental Software Architecture

There are some differences between the software architecture of the Simulator and the architecture of the software implemented on the experimental setup. First of all the software is divided into two main parts: the Teleoperation Module and the software directly executed by the autonomous robot. The human operator works on a Pc using the Teleoperator Module and sends the desired trajectory to the robot. The *Command Sender* block allows to pack data and to send data to the desired IP address (the robot on board pc) using a Wire-less network. The other part of the software is loaded on the on board pc of the robot. The block *Command Receiver* permits to receive the commanded trajectory from the teleoperator and to unpack the sent data. The information about the joint positions, velocities and accelerations (required in order to update the Jacobian matrix and to computed the dynamic matrices) is obtained by on board sensors, not through the integration of the equations of motion as in the simulator case. While the Teleoperation Module is still a Simulink model, the software running on the robot has been compiled into a C code using the Matlab Real Time Workshop and than compiled into a binary executable file.

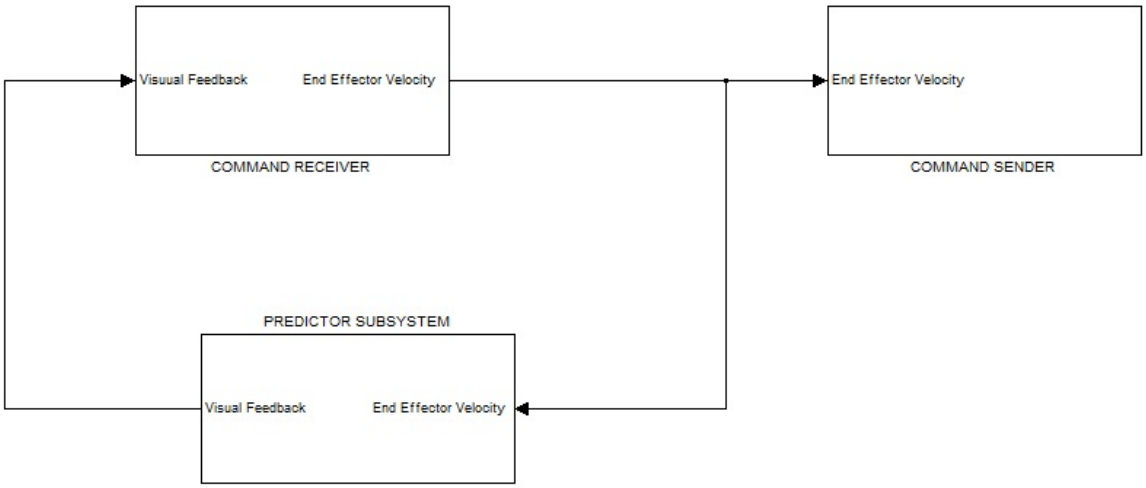


Figure 3.1: Teleoperator Module.

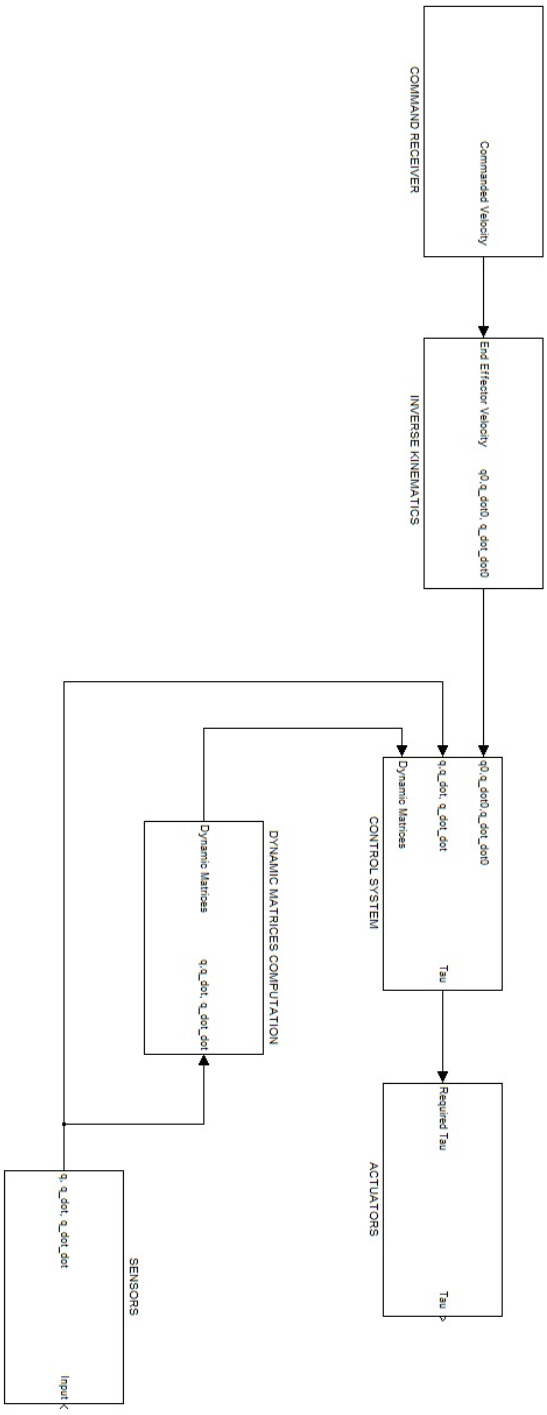


Figure 3.2: Autonomous Robot Module

3.2.2 The Free Flyer Robot

The experimental setup consists in a group of robots floating on a flat floor. This kind of system allows the simulation of the typical lightweigth, no friction condition that a system experiments while it is in orbit.The experimental setup is finalized to the simulation of autonomous rendezvous and docking strategies and algorithms involving a large number of small micro-nanostellites. This represents a unique possibility to test attitude control algorithms in a condition close to the real one, but constrainted to the 2D case, 3 Degrees of Freedom.

Figure 3.3: The Epoxy flat floor.

The robots float on the flat floor using air pads, that permit to crate a small air layer that supports the weight of the robot. The translational and attitude control is performed using small supersonic jet thrusters, that work using compressed air (as the air pads). Each robot is equipped with 8 fixed position and orientation thrusters in order to fully control the motion of the robot.

A *Pc-104* is the brain of each robot: the on board Pc, based on a Linux architecture manages the execution of the experiments and the collection of data from the sensors. The softwares that these robots execute are Matlab/Simulink codes, compiled into executable files using Real Time Linux. An Indoor Pseudo GPS System can provide to each robot its position on the floor in an inertial reference system with origin in one of the corners of the floor. An on board Fiber Optic Gyro provides to each robot its angular velocity and attitude.

The four robots are called using a colour based system:

- Green
- Yellow
- Red

- Orange

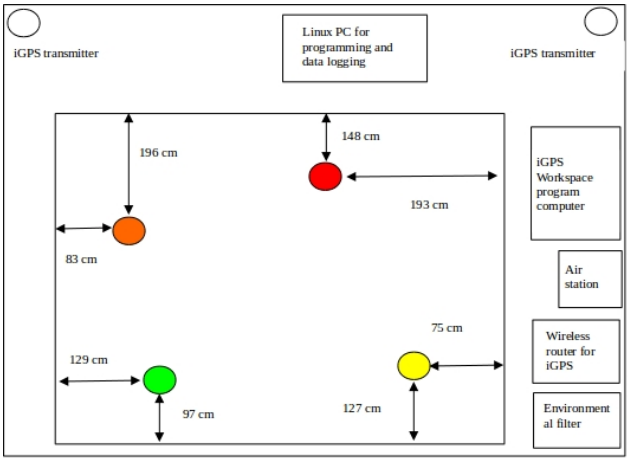


Figure 3.4: Initial positions of the robots.

3.2.3 Manipulator Design

The output of the design phase ia a planar manipulator, made by modular revolute joints. The architecture of the arm is highly flexible: all the joints are exactly the same, and they are connected all at the same way to the precedent and successive joint of the chain. All the driver electronics is accommodated inside the joint and each driver is connected in series to the two adjoining drivers. The joint itself is made by plastic material, modeled using a 3D Printer. Providing a CAD model, the printer can realize the desired structure in a few hours and at relatively low cost. In Fig. 3.5 is presented the designed modular joint:

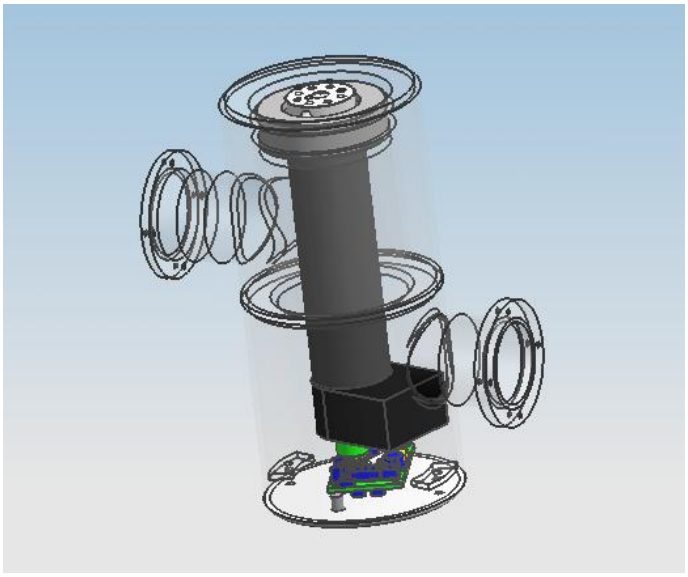


Figure 3.5: Designed joint.

The manipulator developed for the testbed is characterized by the following features:

DOFs	4
Joint type	Revolute
Length	0.52 m
Total Mass	1.6 kg
Aipads	1-2

Chapter 4

Kinematics

4.1 The Kinematic Control Problem

The reference motion of the end effector generated by the human operator is expressed in the Cartesian space. In order to control the base and the manipulator it is necessary to map the information about the end effector position, velocity and acceleration into the joint variables. The end effector position can be written as a non linear function of the joint variables:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{4.1.1}$$

where $\mathbf{x}$ is a $m\ x\ 1$ vector of task variables (ex the end effector position and orientation), $\mathbf{q}$ is a $n\ x\ 1$ vector of joint variables and $\mathbf{f}(\mathbf{q})$ is a non linear vectorial function ($m\ x\ n$) function of the configuration of the system. The 4.1.1 map can be differentiate into:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{4.1.2}$$

where:

$$\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \tag{4.1.3}$$

$\mathbf{J}(\mathbf{q})$ is the $m\ x\ n$ Jacobian matrix of the system. 4.1.2 can be diffrentiated to obtain the realtion for the joint acceleration:

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ \ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \tag{4.1.4}$$

Given the trajectory $\mathbf{x}(t)$ in the Cartesian space of a generic point of the manipulator, the *kinematic control problem* can be formulated in order to find the joint trajectory $\mathbf{q}(t)$ that satisfies $\mathbf{f}(\mathbf{q}(t)) = \mathbf{x}$. The Jacobian formulation reveals to be extremely simple; it is possible to solve the kinematic control problem simply solving:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\,\dot{\mathbf{x}} \tag{4.1.5}$$

It is evident that Equation 4.1.5 can be solved only if $n = m$. This means that the problem can be easily faced only if the manipulator is non redundant.

Definition 1. *A manipulator is said to be* ***redundant*** *if the dimension of the joint space is higher that the dimension of the task space $m \geq n$. Redundancy can be an extremely attractive feature, because it allows the designer to increase the dimension of the task space imposing new conditions and tasks on the system (ex obstacles avoidance, performances optimization,...)*

The main techniques for redundancy solution are:

- Simple Jacobian-Based Techniques
- Gradient Projection Method
- Task Space Augmentation
- Inverse Kinematic Functions

The basics on robot kinematics modeling and Jacobian computation can be found in [30], [19], [20].

4.2 Jacobian Computation

In the last years a lot of algorithms for the Jacobian calculation have been developed in order to reduce drastically the computational cost of the kinematics step, [18], [16]. The method implemented in the thesis is the *Direct Path Method* by A.A. Moosavian, [16]. The method provides low computational cost and can be easily implemented in a symbolic environment. This allows to run the algorithm in two different ways : performing the complete execution of the algorithm at each time step in order to update the Jacobian structure or performing an a priori symbolic solution of the Jacobian matrix, with a simple expression evaluation at each time step. Furthermore the method is easily expandible to the case of multiple manipulators. In this section the expanded algorithm is presented.

4.2.1 The Direct Path Method

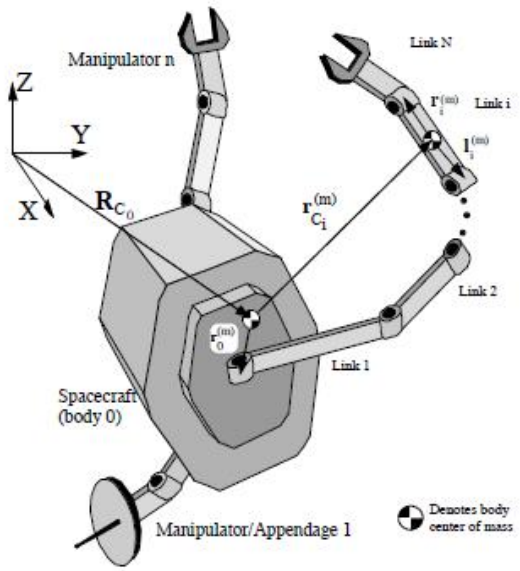


Figure 4.1: System's configuration

The complete algorithm, together with the study on the *Barycenter Vector Approach* can be found in [16]. The method is based on the use of body-fixed geometric vectors. The motion of the spacecraft's center of mass is used to describe the system translation with respect to an inertial frame **XYZ**. The body 0 represents the spacecraft, which is connected to n manipulators, each with N_m links. All the joints are assumed to be revolute, with a single degree of freedom. The joint angles and rates are represented by the K x 1 vectors $\theta = \left(\theta^{(\mathbf{1})}, \theta^{(\mathbf{2})}, ..., \theta^{(\mathbf{n})}\right)^T$, where $\theta^{(\mathbf{m})}$ is an N x 1 column vector, which contains the joint angles of the mth manipulator and $K = \sum_{m=1}^{n} N_m$.
The inertial position of a generic point P in the inertial frame can be written as:

$$\mathbf{R_P} = \mathbf{R_{C_0}} + \mathbf{r_{C_i}} + \mathbf{r_{P/C_i}} \tag{4.2.6}$$

where $\mathbf{R_{C_0}}$ is the inertial position of the spacecraft Center of Mass (CM), $\mathbf{r_{C_i}}$ is the position vector of the CM of the *ith* link respect to the the spacecraft CM and $\mathbf{r_{P/C_i}}$ is the position vector of the point P with respect of the CM of the link *ith*.

$\mathbf{r_{C_i}}$ can be expressed as:

$$\mathbf{r_{C_0}} = \mathbf{0} \tag{4.2.7}$$

$$\mathbf{r_{C_i}^{(m)}} = \mathbf{r_0^{(m)}} + \sum_{k=1}^{i-1} \left(\mathbf{r_k^{(m)}} - \mathbf{l_k^{(m)}}\right) - \mathbf{l_i^{(m)}} \tag{4.2.8}$$

with $m = 1, ..., n$ and $i = 1, ..., N_m$.

The velocity of the point P and its angular velocity are easily obtained by differentiation of the 4.2.6 and 4.2.8, that yields:

$$\dot{\mathbf{R}}^{\mathbf{m}}_{\mathbf{p_i}} = \dot{\mathbf{R}}_{\mathbf{C_0}} + \omega_{\mathbf{0}} \times \mathbf{r}^{\mathbf{m}}_{\mathbf{0}} + \sum_{k=1}^{i-1} \omega^{\mathbf{m}}_{\mathbf{k}} \times \left(\mathbf{r}^{\mathbf{m}}_{\mathbf{k}} - \mathbf{l}^{\mathbf{m}}_{\mathbf{k}}\right) - \omega^{\mathbf{m}}_{\mathbf{i}} \times \left(\mathbf{l}^{\mathbf{m}}_{\mathbf{i}} - \mathbf{r}_{\mathbf{p/C^m_i}}\right) \quad (4.2.9)$$

$$\omega^{\mathbf{m}}_{\mathbf{k}} = \omega_{\mathbf{0}} + \sum_{i=1}^{k} \dot{\theta}^m_i \mathbf{z}^{\mathbf{m}}_{\mathbf{i}} \quad (4.2.10)$$

The linear velocity of the arbitrary point P on the *ith* link of the *mth* manipulator must be rearranged in the form:

$$\begin{pmatrix} \dot{\mathbf{R}}_P \\ \omega^{\mathbf{m}}_{\mathbf{i}} \end{pmatrix} = \mathbf{J}^m_{i,p}\, \mu \quad (4.2.11)$$

where:

$$\mu = \begin{pmatrix} \dot{\mathbf{R}}_{C_0} \\ \omega_{\mathbf{0}} \\ \dot{\theta} \end{pmatrix} \quad (4.2.12)$$

The Jacobian can be obtained based on the 4.18 and 4.2.10 as:

$$\mathbf{J}^m_{i,p} = \begin{bmatrix} \mathbf{1}_{3x3} & \mathbf{J}^m_1 & \mathbf{J}^m_2 \\ \mathbf{0}_{3x3} & \mathbf{1}_{3x3} & \mathbf{J}^m_3 \end{bmatrix} \quad (4.2.13)$$

where:

$$\mathbf{J}^m_1 = -\left[\mathbf{T}^0_0 \mathbf{r}^m_0 + \sum_{k=1}^{i-1} \left[\mathbf{T}^m_k \left({}^k\mathbf{r}^m_k - {}^k\mathbf{l}^m_k\right)\right] - \mathbf{T}^m_i \left({}^i\mathbf{l}^m_i - {}^i\mathbf{r}_{p/C^m_i}\right)\right]^{\times} \quad (4.2.14)$$

$$\mathbf{J}^m_2 = -\sum_{k=1}^{i-1} \left[\mathbf{T}^m_k \left({}^k\mathbf{r}^m_k - {}^k\mathbf{l}^m_k\right)\right]^{\times} \mathbf{E}^m_k + \left[\mathbf{T}^m_i \left({}^i\mathbf{l}^m_i - {}^i\mathbf{r}_{p/C^m_i}\right)\right]^{\times} \mathbf{E}^m_i \quad (4.2.15)$$

$$\mathbf{J}^m_3 = \mathbf{E}^m_i \quad (4.2.16)$$

Where the operator $[]^{\times}$ applied to a generic vector $\mathbf{r}$ means:

$$[\mathbf{r}]^{\times} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \quad (4.2.17)$$

The $\mathbf{T}_0$ and $\mathbf{T}_j^k$ are rotation matrices between body-fixed frame and the inertial frame and $\mathbf{E}_j^k$ is defined as:

$$\mathbf{E}_j^k = \left[\mathbf{0}\ \mathbf{T}_1^{k1}\mathbf{z}_1^k\ ...\ \mathbf{T}_j^{kj}\mathbf{z}_j^k\ \mathbf{0} \right] \tag{4.2.18}$$

where ${}^j\mathbf{z}_j^k = (\ 0\ 0\ 1\ )^T$ is a unit vector along the axis of rotation of the jth joint of the kth manipulator, expressed in its own body-fixed frame.

4.3 Redundancy Solution

4.3.1 Simple Jacobian-based Techniques

Most of the techniques to solve the system's redundancy are based on the inversion of the mapping [19], [20], in order to have the simple map :

$$\dot{\mathbf{q}} = \mathbf{K}(\mathbf{q})\dot{\mathbf{x}} \tag{4.3.19}$$

Where $\mathbf{K}$ is a nxm control matrix. By definition $\mathbf{J}$ is not a square matrix for redundant manipulators, so $\mathbf{J}$ is not invertible. Whitney in 1969 proposed to use the *Moore-Penrose Pseudoinverse* of the Jacobian matrix:

$$\mathbf{K}(\mathbf{q}) = \mathbf{J}^* = \mathbf{J}^{\mathbf{T}}\left(\mathbf{J}\mathbf{J}^{\mathbf{T}}\right)^{-1} \tag{4.3.20}$$

This is a particularly simple and attractive solution. The consequence of this solution of the redundancy is the minimization of the sum of the square of the joint velocities.

4.3.2 Projection techniques

The Gradient Projection techniques could be considered as a generalization of the pseudoinverse methods explained above :

$$\dot{\mathbf{q}} \ = \ \mathbf{J}^*(\mathbf{q})\dot{\mathbf{x}} \ + \ \left[\mathbf{I} - \mathbf{J}^*(\mathbf{q})\mathbf{J}(\mathbf{q})\right]\dot{\mathbf{q}}_\mathbf{0} \tag{4.3.21}$$

where I is the $n\ x\ n$ identity matrix and $\dot{\mathbf{q}}_\mathbf{0}$ is a generic, arbitrary n x 1 velocity vector.

In this way the projection operator $[\ \mathbf{I}\ -\ \mathbf{J}^*\mathbf{J}\ ]$ selects the components of $\dot{\mathbf{q}}_\mathbf{0}$ in the kernel of $\mathbf{J}$. This produce only a joint self motion, without task motion. In this way the desired task to be accomplished remains the same but the choice of the $\dot{\mathbf{q}}_\mathbf{0}$ imposes constraints on the motion of the joints.
The definition of the vector $\dot{\mathbf{q}}_\mathbf{0}$ remains the key point for the resolution of the redundancy. One of the most used methods is the *Gradient Projection Method*, that solves the redundancy minimizing the cost function $h(\mathbf{q})$. Given the cost function $h(\mathbf{q})$, $\dot{\mathbf{q}}_\mathbf{0}$ can be chosen as:

$$\dot{\mathbf{q}}_\mathbf{0} \;=\; \left(\frac{\partial h}{\partial \mathbf{q}} \right) \;=\; \nabla h \tag{4.3.22}$$

The cost function could led to extremely various types of optimization,[20],[19].

Joint limit Avoidance Inverse Kinematics Solution

One of the most implemented types of optimization is the Joint limit avoidance. This technique allows the user to decide the joints limits and to prevent joint angles from exceeding their technological and structural limits. The approach developed by Marchand,Chaumette and Rizzo is particularly attractive because of the simplicity of the developed cost function, [21], [22]. In this approach the joints velocities are:

$$\dot{\mathbf{q}}_\mathbf{d} \;=\; -\lambda \mathbf{e} \tag{4.3.23}$$

where

$$\mathbf{e} \;=\; \mathbf{J}^* \mathbf{e_1} \;+\; \beta \left(\mathbf{I} \;-\; \mathbf{J}^* \mathbf{J} \right) \mathbf{g}^T \tag{4.3.24}$$

$\dot{\mathbf{q}}_\mathbf{d}$ is the output joint velocity, β is a scalar that set the amplitude of the control law, $\mathbf{e}_1$ is and $\mathbf{g}$ is the secondary task defined as:

$$\mathbf{g} \;=\; \frac{\partial h}{\partial \mathbf{q}} \tag{4.3.25}$$

Activation threshold 1.

$$\left(\begin{array}{rcl} \tilde{\mathbf{q}}_{i_{min}} & = & \mathbf{q}_{i_{min}} \;+\; \rho \;(\; \mathbf{q}_{i_{max}} \;-\; \mathbf{q}_{i_{min}} \;) \\ \tilde{\mathbf{q}}_{i_{max}} & = & \mathbf{q}_{i_{max}} \;-\; \rho \;(\; \mathbf{q}_{i_{max}} \;-\; \mathbf{q}_{i_{min}} \;) \end{array} \right) \tag{4.3.26}$$

where $\mathbf{q}_{i_{min}}$ and $\mathbf{q}_{i_{max}}$ are the maximum and minimum joint angles and $0 \;<\; \rho \;<\; 0.5$.

The cost function is defined as:

$$h_s \;=\; \frac{1}{2} \sum_{i=1}^{n} \frac{s_i^2}{\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}}} \tag{4.3.27}$$

where:

$$s_i \;=\; \left[\begin{array}{rl} \mathbf{q}_i - \tilde{\mathbf{q}}_{i_{max}} & if \mathbf{q}_i > \tilde{\mathbf{q}}_{i_{max}} \\ \mathbf{q}_i - \tilde{\mathbf{q}}_{i_{min}} & if \mathbf{q}_i > \tilde{\mathbf{q}}_{i_{min}} \\ 0 & else \end{array} \right] \tag{4.3.28}$$

$\mathbf{g}$ takes the form:

$$\mathbf{g} = \left(\begin{array}{rl} \frac{\beta(q_i - \tilde{q}_{imax})}{q_{imax} - q_{imin}} & if \;\; q_i \;\geq \tilde{q}_{imax} \\ \frac{\beta(q_i - \tilde{q}_{imin})}{q_{imax} - q_{imin}} & if \;\; q_i \;\leq \tilde{q}_{imin} \\ 0 & else \end{array} \right) \tag{4.3.29}$$

and:

$$\frac{\partial \mathbf{g}}{\partial t} = 0 \tag{4.3.30}$$

4.4 Validation

4.4.1 1 Link Free-Flyer

The simplest case that can be used for the validation of the implmentation of the Jacobian matrix is the case of the planar motion of a free flyer with a one-link manipulator in the plane of motion of the spacecraft.

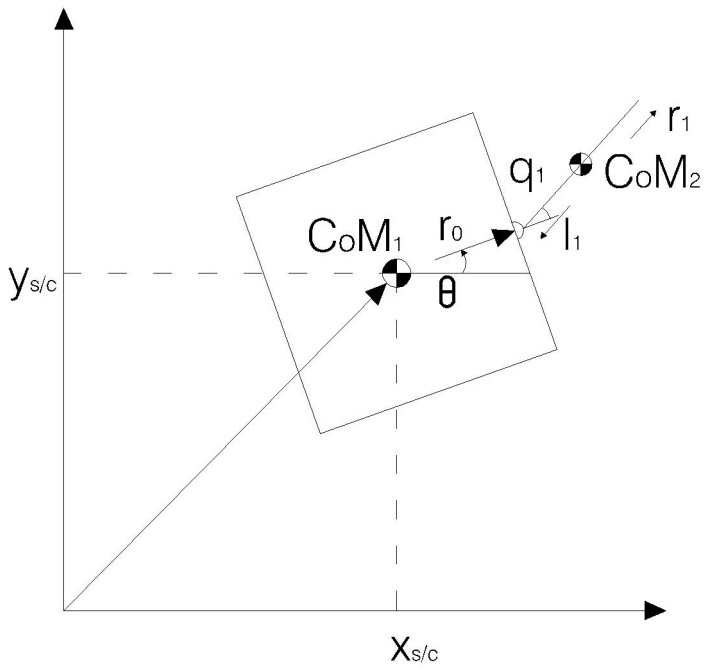


Figure 4.2: Single link arm. 2D case.

This case is attractive because it is easy to write the analytical solution and to compare it to the results of the algorithm. The position of the end effector can be written as:

$$\begin{aligned} x_e &= x_{s/c} + r_o \cos\theta + (r_1 + l_1)\cos(\theta + q_1) \\ y_e &= y_{s/c} + r_o \sin\theta + (r_1 + l_1)\sin(\theta + q_1) \end{aligned} \tag{4.4.31}$$

Where r_0, r_1 and l_1 are the norm of the vectors $\mathbf{r}_o$, $\mathbf{r}_1$ and $\mathbf{l}_1$. Deriving with respect to time:

$$\begin{aligned} \dot{x_e} &= \dot{x}_{s/c} - r_o\dot{\theta}\sin\theta - (r_1 + l_1)\left(\dot{\theta} + \dot{q1}\right)\sin(\theta + q_1) \\ \dot{y_e} &= \dot{y}_{s/c} + r_o\dot{\theta}\cos\theta + (r_1 + l_1)\left(\dot{\theta} + \dot{q_1}\right)\cos(\theta + q_1) \end{aligned} \tag{4.4.32}$$

Rearranging the equation is easy to obtain:

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & -r_o\sin\theta - (r_1 + l_1)\sin(\theta + q_1) & -(r_1 + l_1)\sin(\theta + q_1) \\ 0 & 1 & r_o\cos\theta + (r_1 + l_1)\cos(\theta + q_1) & (r_1 + l_1)\cos(\theta + q_1) \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.4.33}$$

Running the function generated from the above algorithm, in symbolic mode on MATLAB, provided us of the same result.

Spacecraft		
Lenght	l	2 m
Height	h	2 m
Width	w	2 m
Manipulator		
DoFs	n	4
Length	l_m	0.5 m
Joint configuration in the 3D case		y - z - y - y

Table 4.1: System Configuration, Kinematics

4.5 Numerical and graphical validation

It is possible to validate the Kinematics block also checking graphically if the resulting joint motion, applied to the geometrical model of the system produces the effective desired trajectory of the end effector. It is presented here the graphical validation for both the 2D and 3D cases, solving the redundancy with both the Moore-Penrose Pseudo Inverse Matrix and with the Merchand, Chaumette and Rizzo's joint limit avoidance algorithm. For the tests the robot's following configuration has been used:

In the 2D case the task and joint vectors are organized in the following way:

$$\dot{\mathbf{x}}_\mathbf{e} = \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} \quad \dot{\mathbf{q}} = \begin{pmatrix} v_{sx} \\ v_{sy} \\ \omega_s \\ \dot{\mathbf{q}} \end{pmatrix} \tag{4.5.34}$$

In the 3D case the task and joint vectors are organized in the following way:

$$\dot{\mathbf{x}}_\mathbf{e} = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} \quad \dot{\mathbf{q}} = \begin{pmatrix} \mathbf{v}_s \\ \omega_s \\ \dot{\mathbf{q}} \end{pmatrix} \tag{4.5.35}$$

4.5.1 2D Case, Rectilinear x-Trajectory

- Desired motion:

$$\mathbf{x} = \begin{bmatrix} v_x t + x_0 \\ 0 \\ 0 \end{bmatrix} \tag{4.5.36}$$

- Velocity input:

$$\mathbf{v} = \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} \tag{4.5.37}$$

- Initial Conditions

$$\mathbf{x}_0 = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} \tag{4.5.38}$$

The simulation produces the following results in terms of joint motion and path followed:

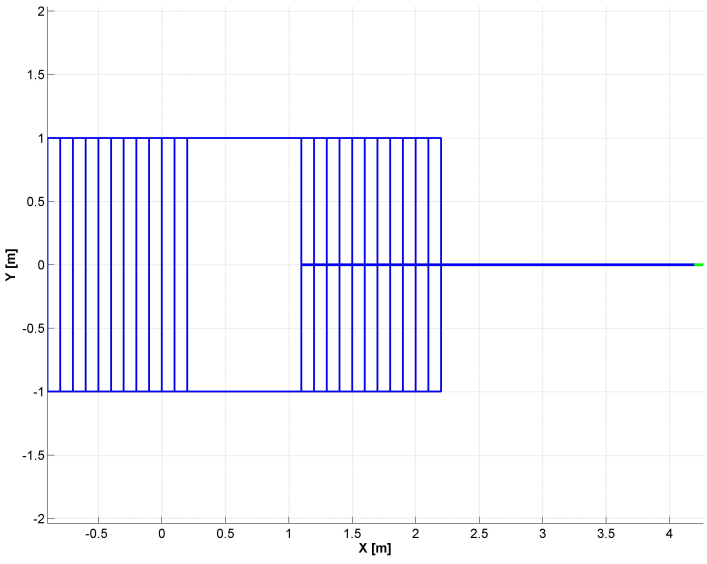


Figure 4.3: Resolved motion, Rectilinear x trajectory.

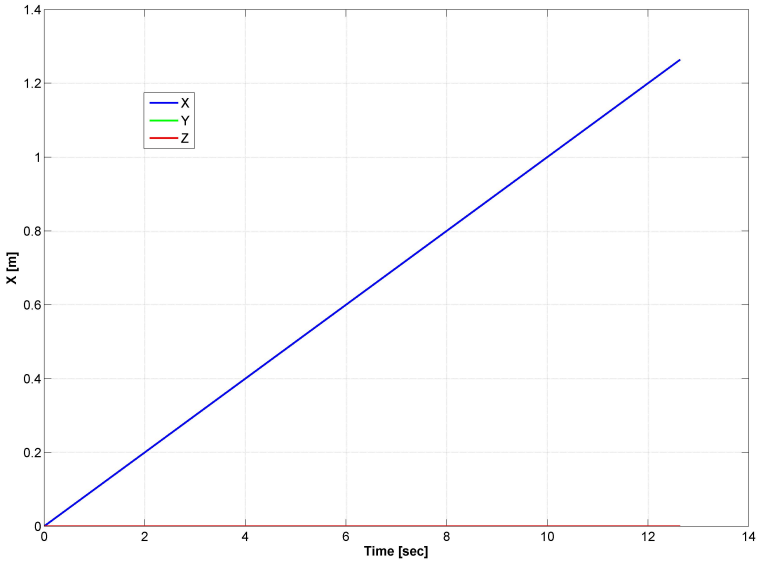


Figure 4.4: Spacecraft CoM Position,Rectilinear x trajectory.

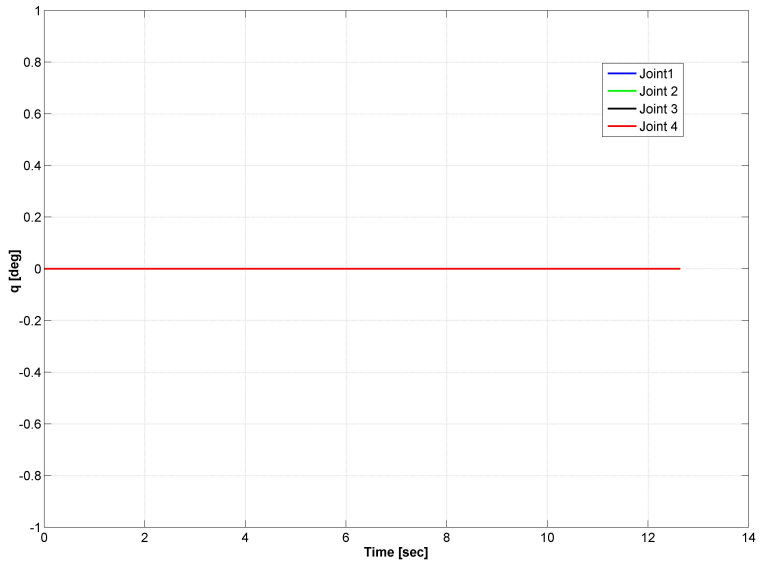


Figure 4.5: Joint Angles, Rectilinear x trajectory.

4.5.2 2D Case, Rectilinear y-Trajcetory

- Desired motion:

$$\mathbf{x} = \begin{bmatrix} 0 \\ v_y t + y_0 \\ 0 \end{bmatrix} \tag{4.5.39}$$

- Velocity input:

$$\mathbf{v} = \begin{bmatrix} 0 \\ v_y \\ 0 \end{bmatrix} \tag{4.5.40}$$

- Initial Conditions

$$\mathbf{x}_0 = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} \tag{4.5.41}$$

The simulation produces the following results in terms of joint motion and path followed:

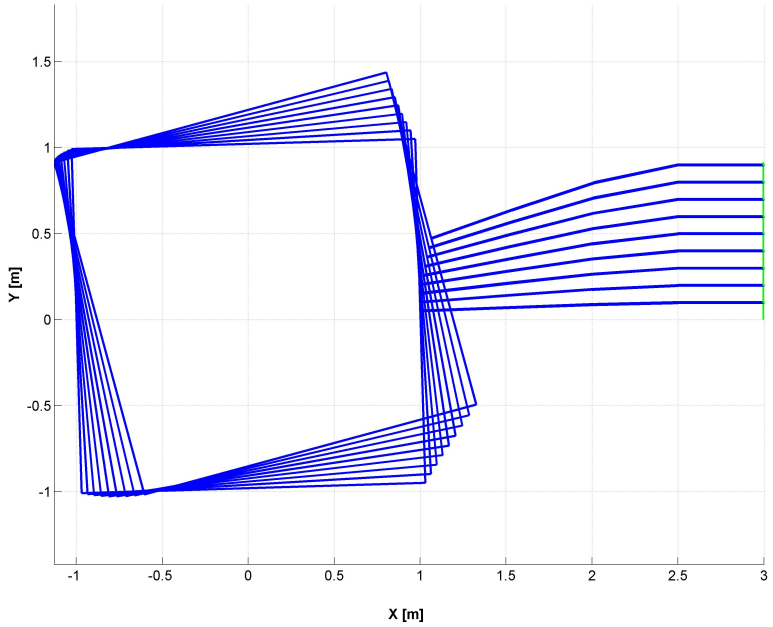


Figure 4.6: Resolved motion, Rectilinear y trajectory.

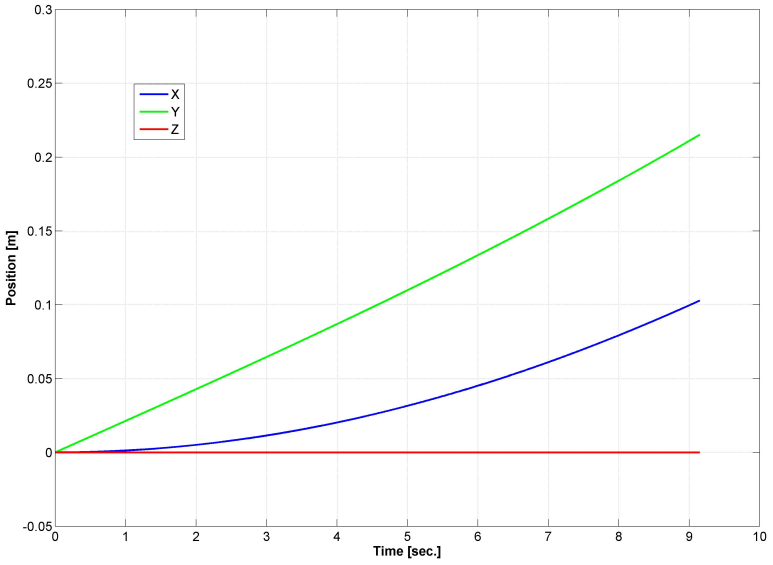


Figure 4.7: Spacecraft CoM Position, Rectilinear y trajectory.

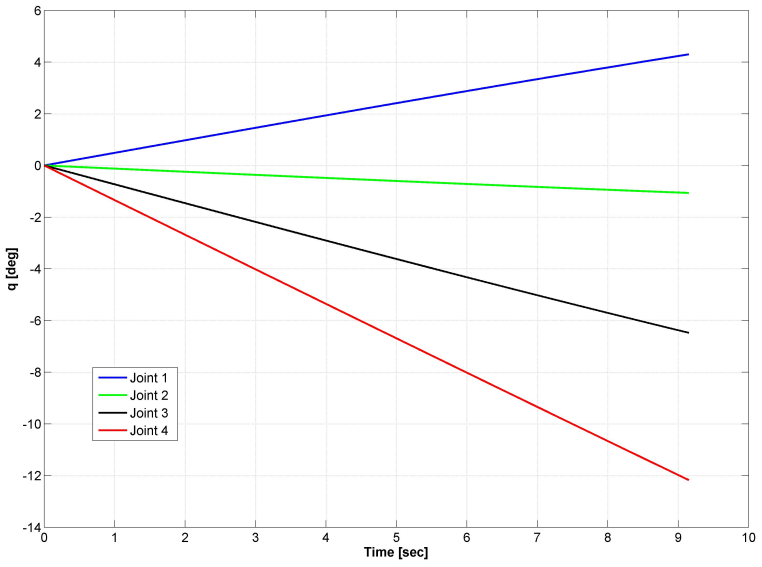


Figure 4.8: Joint Angles, Rectilinear y trajectory.

4.5.3 2D Case, Circular Trajectory

- Input end effector trajectory:

$$\mathbf{x} = \begin{bmatrix} r\ cos(\omega\ t) \\ r\ sin(\omega\ t) \\ \omega\ t \end{bmatrix} \tag{4.5.42}$$

- Velocity input:

$$\mathbf{v} = \begin{bmatrix} -\omega\ r\ sin\,(\omega t) \\ \omega\ r\ cos\,(\omega t) \\ \omega \end{bmatrix} \tag{4.5.43}$$

- Initial Conditions

$$\mathbf{x}_0 = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} \tag{4.5.44}$$

Simple Jacobian Inversion Technique

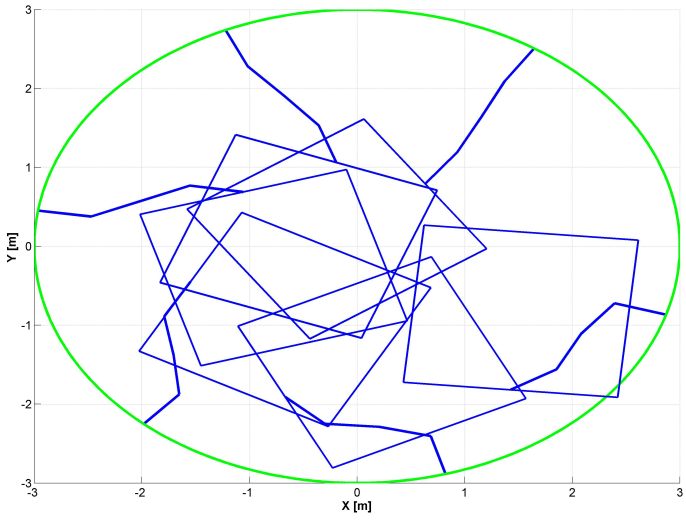


Figure 4.9: Resolved motion, Circular trajectory, Jacobian Pseudo-Inverse.

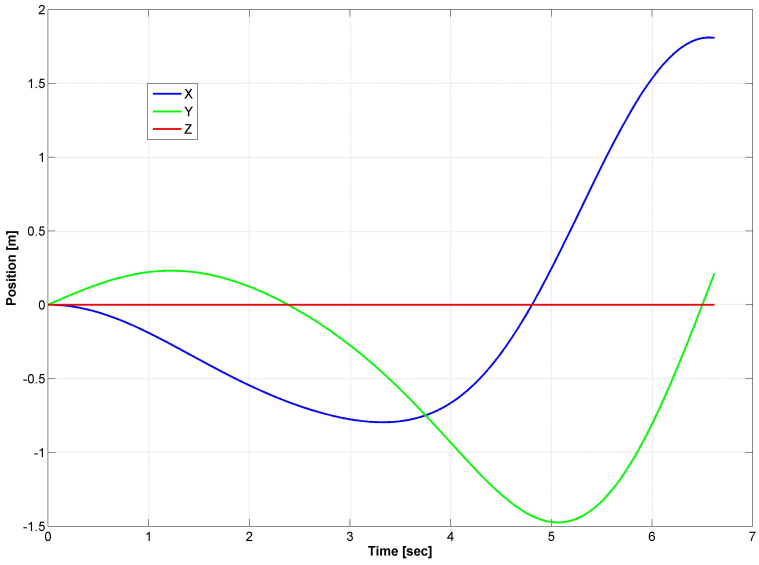


Figure 4.10: Spacecraft Position, Circular trajectory, Jacobian Pseudo-Inverse.

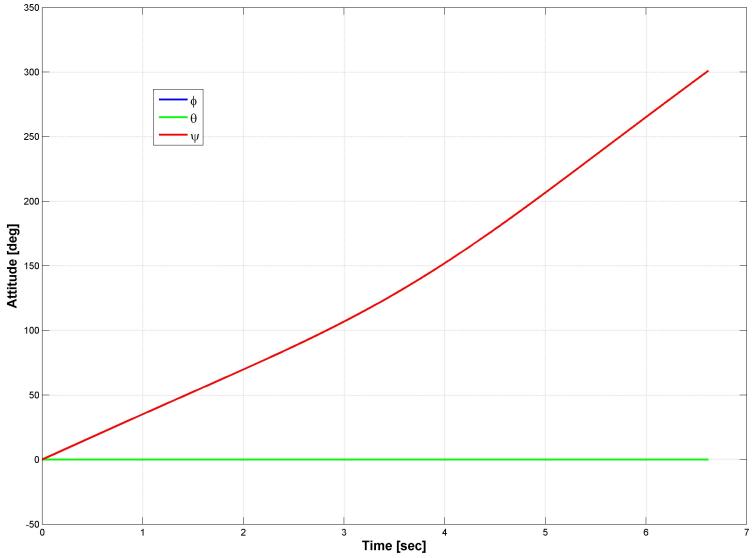


Figure 4.11: Spacecraft Attitude, Circular trajectory, Jacobian Pseudo-Inverse.

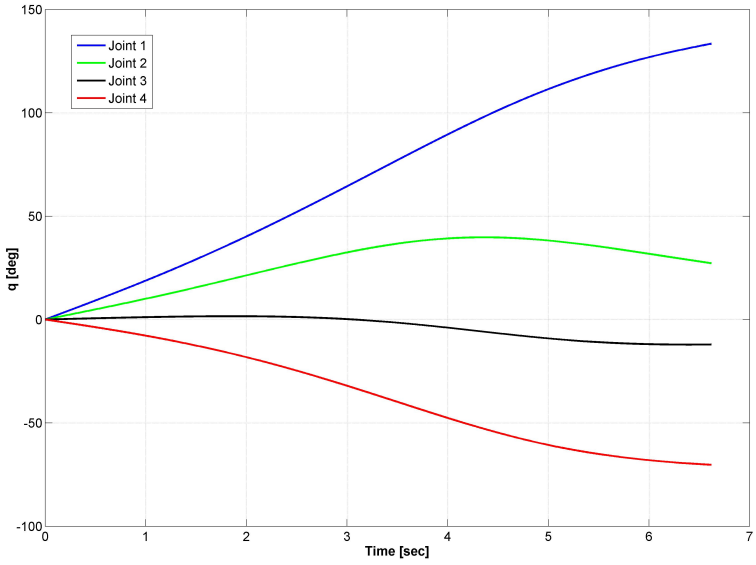


Figure 4.12: Joint Angles, Circular trajectory, Jacobian Pseudo-Inverse.

It is important to underline that this kinematic solution can cause body compenetrations: no information is given about the joints limits or about the bulk of the bodies involved in the motion. Figures 4.9 and 4.12 provide a clear example of this condition: the desired trajectory, combined with the selected solution for the Jacobian's inversion causes rigid body compenentrations. This test underlines the need of a kinematic solution absolutely reliable in terms of safety of the system itself: the pseudoinverse does not take into account the system configuration and the test proofs how dangerous could be for the system a wrong selection in the kinematics inversion algorithm.

Joint Limit Avoidance Algorithm

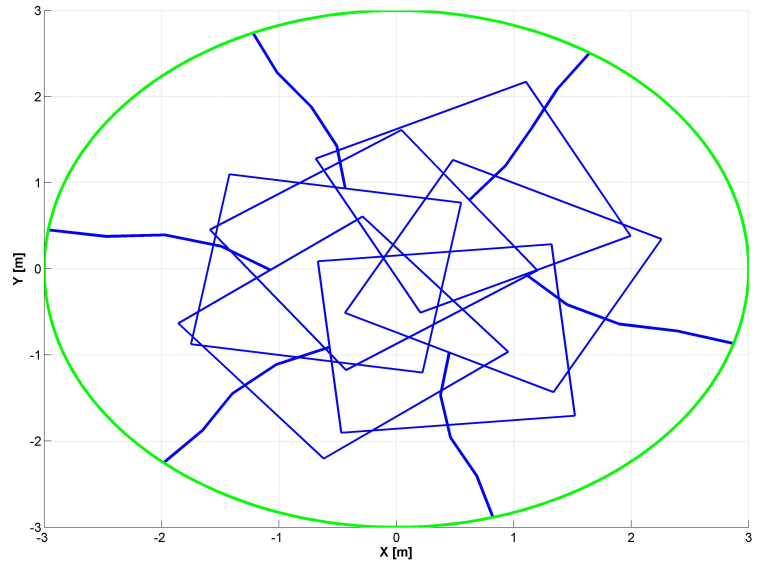


Figure 4.13: Resolved motion, Circular trajectory, Joint Limit Avoidance.

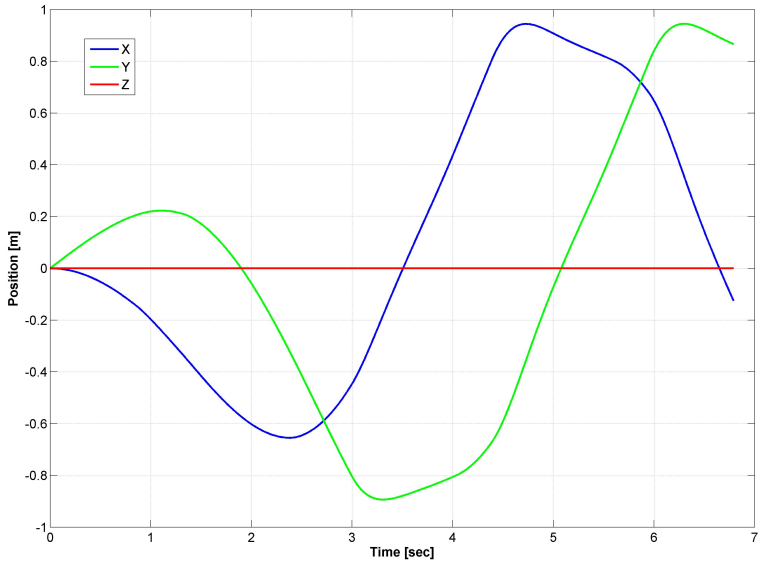


Figure 4.14: Spacecraft Position, Circular trajectory, Joint Limit Avoidance.

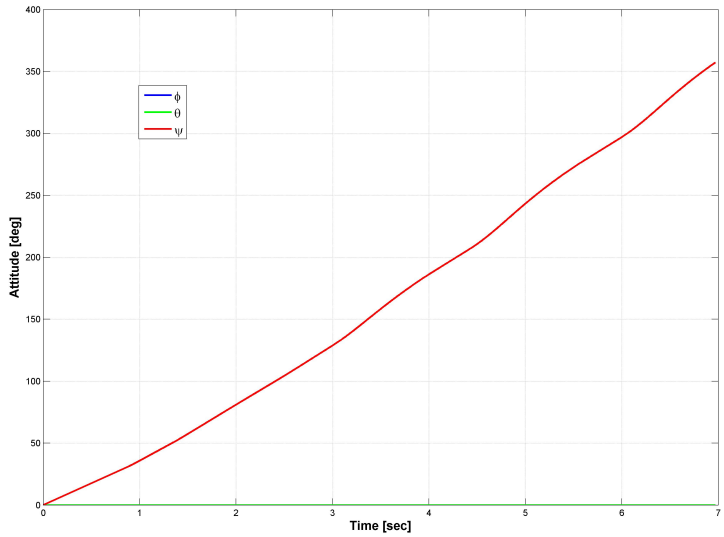


Figure 4.15: Spacecraft Attitude, Circular trajectory, Joint Limit Avoidance.

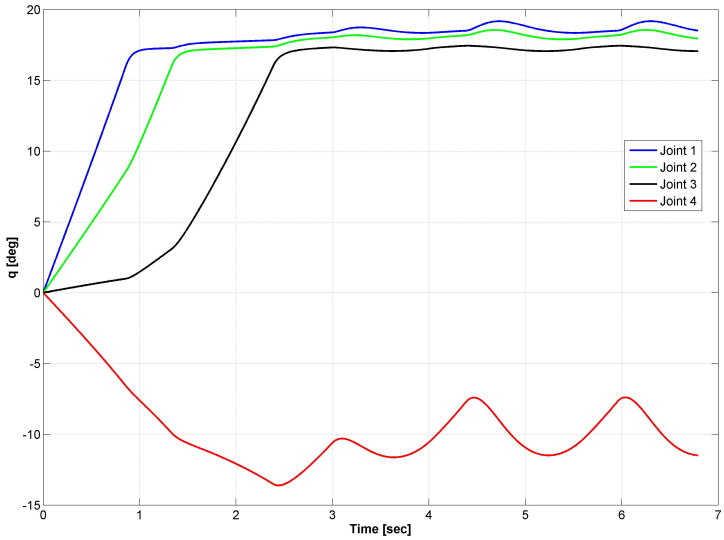


Figure 4.16: Joint Angles, Circular trajectory, Joint Limit Avoidance.

In this case the kinematic algorithm computes the motion taking into account the joint limits provided by the user.Comparing Figure 4.13 with Figure 4.9 it is clear how the Joint Limit Avoidance Algorithm prevents the system from compenetrations. Figure 4.16 shows the position of the joints during the test: while in the previous test the joints motion was computed in order to minimize the total quadratic velocity of the system's DoFs, Figure 4.16 shows that, if one of the joints gets close to its limit, the algorithm prevents it from breaking the constraint. All the system reacts in order to keep all its DoFs inside the boundaries selected by the user.

4.5.4 3D Case, Complex Trajectory

The validation of the full 3D model Kinematics has been done on a complex 3D trajectory defined as:

- Desired trajctory

$$\mathbf{x} = \begin{bmatrix} r\ \cos\omega t \\ r\ \sin\omega t \\ -\frac{1}{k}\cos kt \\ 0 \\ 0 \\ \omega\ t \end{bmatrix} \tag{4.5.45}$$

- Input end effector velocity

$$\mathbf{v} = \begin{bmatrix} -\omega\ r\ \sin\omega t \\ \omega\ r\ \cos\omega t \\ \sin kt \\ 0 \\ 0 \\ \omega \end{bmatrix} \tag{4.5.46}$$

- Initial conditions

$$\mathbf{x}_0 = \begin{bmatrix} r \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.5.47}$$

Joint Limit Avoidance Solution

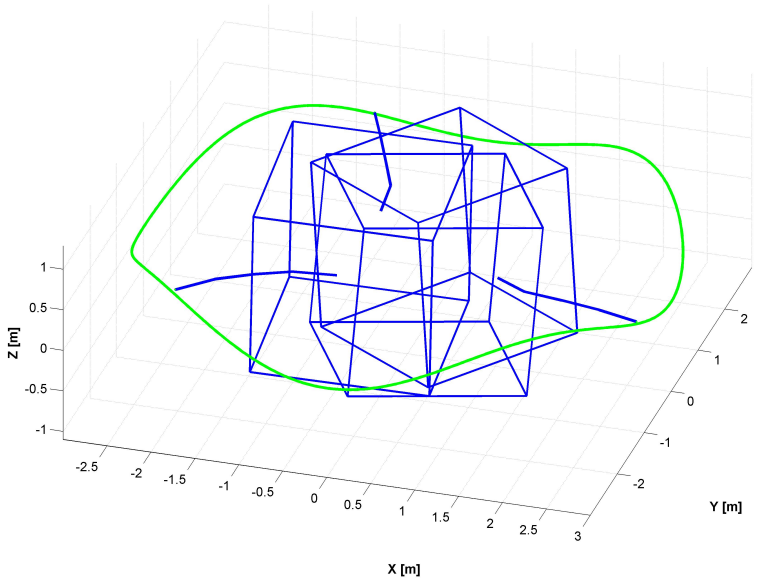


Figure 4.17: Resolved motion, 3D Case.

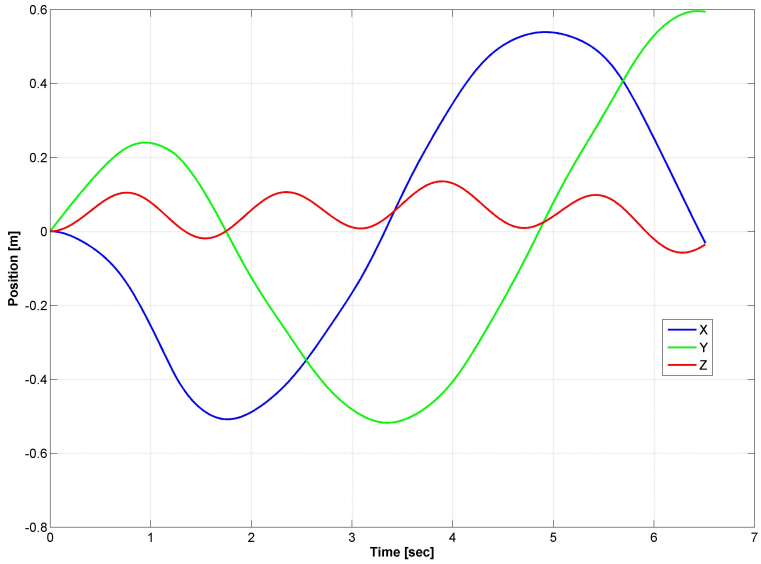


Figure 4.18: Spacecraft Position, 3D Case.

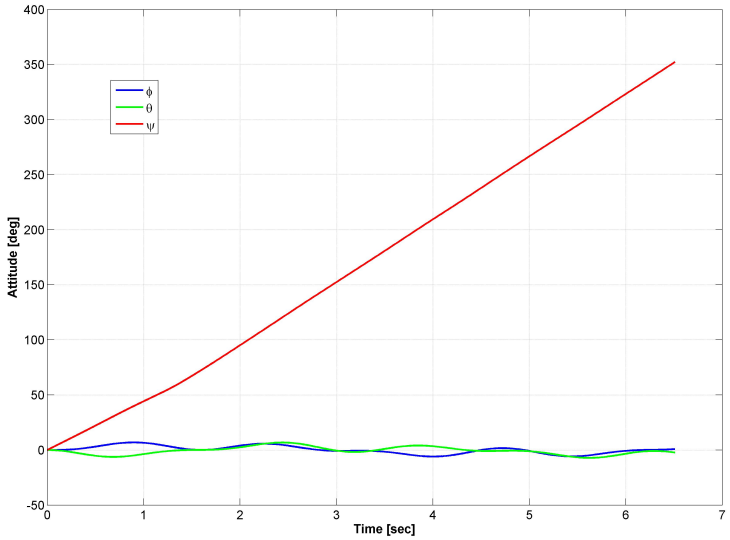


Figure 4.19: Spacecraft Attitude, 3D Case.

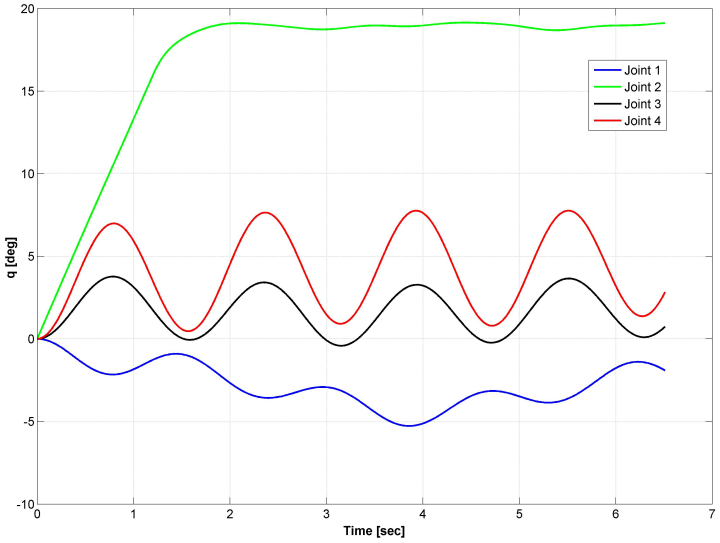


Figure 4.20: Joint Angles, 3D Case

4.5.5 Conclusions

The figures shown, validate the implementation of the inverse kinematics map. The resolved motion, applied to the geometrical model of the system, causes the system to follow the commanded trajectory. The simple Jacobian Pseudo Inverse Solution proved to be poor in terms of reliability of the commanded trajectory: the algorithm is highly susceptible to generate joints trajectories that cause compenetrations in the system. On the other side, the Joint Limit Avoidance implementation proved to be reliable in preventing the system from dangerous configurations. Fig. 4.16 shows how the algorithm limits the motion of a joint if close to its selected boundaries and reacts modifying the motion of the rest of the system. Thinking to the future experimental implementation of the system, this kinematic solution will be critic in order to prevent the systems from damages.

Chapter 5

Spatial Vector Algebra

5.1 Introduction

The study of rigid body linear and angular dynamics algorithms could be expressed in a more compact notation if the spatial vectors are used. A spatial vector is a 6D vector that combines linear and angular aspect of a rigid body motion or force. In this way one spatial vector can do the work of two 3D vectors and one spatial equation can replace the two typical vectorial equations. This algebra allows the engineer to implement quickly rigid body dynamics algorithms and helps in designing new algorithms. The fundamentals of the Space Vector Algebra are presented in the following chapter (the basics concepts can be found in [25] ,[26], [31]).

5.2 Preliminaries

Vectors and Vectors Space 1. *The canonical Linear Algebra defines a vector as a part of a vectorial space. Four vectorial spaces are going to be used in this thesis:*

- R^n *, Coordinate Vector Space*
- E^n *, Euclidean Vector Space*
- M^n *, Spatial Motion Vector Space*
- F^n *, Spatial Force Vector Space*

where n is the dimension of the vectorial space.

The Dual of a Vector Space 1. *Let V be a vectorial space. Its dual space V^*, it is a vectorial space having the same dimension of V and the property that a scalar product is defined between it and V. Duality is a symmetrical relationship:*

if $U = V^*$ then $V = U^*$. Duality is important because the spaces M^n and F^n are dual. This means that a scalar product is defined between motion and force vectors, such that if $\mathbf{m} \in M^6$ and $\mathbf{f} \in F^6$ then $m \cdot f$ represents the power delivered by the force.

5.3 Spatial Velocity

The velocity of the rigid body in figure can be expressed as:

$$\mathbf{v}_P = \mathbf{v}_0 + \omega \times \bar{\mathbf{OP}} \tag{5.3.1}$$

where P is the point of interest, $v\mathbf{v_P}$ is its own velocity, $\mathbf{v_0}$ is the velocity of the fixed point O and $\bar{\mathbf{OP}}$ is the position vector of P relative to O. Two main elements contributes to $\mathbf{v_P}$: the linear translational velocity $\mathbf{v_0}$ and the angular velocity of the rigid body ω about an axis passing through O. Defining a Cartesian reference frame O_{xyz},we define an orthonormal basis $(\mathbf{i}, \mathbf{j}, \mathbf{k}) \subset \mathbf{E}^6$ three directed lines Ox, Oy, Oz, passing through O. ω and $\mathbf{v_0}$ can be expressed in the cartesian frame:

$$\omega \;=\; \omega_x\,\mathbf{i} \;+\; \omega_y\,\mathbf{j} \;+\; \omega_z\,\mathbf{k} \tag{5.3.2}$$

and

$$\mathbf{v}_0 \;=\; v_{Ox}\,\mathbf{i} \;+\; v_{Oy}\,\mathbf{j} \;+\; v_{Oz}\,\mathbf{k} \tag{5.3.3}$$

The aim of the Spatial Algebra is to define a vector $\hat{\mathbf{v}} \in \mathbf{M}^6$ that describe the same motion of ω and $\mathbf{v}_O$. For doing that a Plucker basis must be defined:

Plucker Basis 1. *A Plucker Basis is defined as:*

$$\mathfrak{D}_\circ = (\ \mathbf{d}_{Ox},\ \mathbf{d}_{Oy},\ \mathbf{d}_{Oz},\ \mathbf{d}_x,\ \mathbf{d}_y,\ \mathbf{d}_z\ ) \subset \mathbf{M}^6 \tag{5.3.4}$$

where $\mathbf{d}_{Ox}$, $\mathbf{d}_{Oy}$, $\mathbf{d}_{Oz}$ are unit rotations about the lines Ox , Oy and Oz and $\mathbf{d}_x\mathbf{d}_y\mathbf{d}_y$ are unit translation in the x, yandz directions.

It follows that $\hat{v}$ could be defined as:

$$\hat{\mathbf{v}} \;=\; \omega_x\mathbf{d}_{Ox} \;+\; \omega_y\mathbf{d}_{Oy} \;+\; \omega_z\mathbf{d}_{Oz} \;+\; v_{Ox}\mathbf{d}_x \;+\; v_{Oy}\mathbf{d}_y \;+\; v_{Oz}\mathbf{d}_z \tag{5.3.5}$$

The coordinate vector that represent $\hat{\mathbf{v}}$ in $\mathfrak{D}_\circ$ can be written:

$$\hat{\mathbf{v}}_0 = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ v_{Ox} \\ v_{Oy} \\ v_{Oz} \end{bmatrix} = \begin{bmatrix} \omega \\ \mathbf{v}_0 \end{bmatrix} \tag{5.3.6}$$

5.4 Spatial Force

The reasoning done for the Spatial Velocity can be repeated for the Spatial Force. Given the linear force $\mathbf{f}_0$ and the couple $\mathbf{n}_0$ acting on point O, the force and the couple acting on the point P of the same rigid body is:

$$\begin{aligned} \mathbf{f}_P &= \mathbf{f}_0 \\ \mathbf{n}_P &= \mathbf{n}_0 + \mathbf{f}_0 \times \hat{\mathbf{OP}} \end{aligned} \tag{5.4.7}$$

Repeating the same reasoning as before with the Plucker Basis $\mathfrak{E}_O$:

$$\mathfrak{E}_\circ = (\ \mathbf{e}_x,\ \mathbf{e}_y,\ \mathbf{e}_z,\ \mathbf{e}_{Ox},\ \mathbf{e}_{Oy},\ \mathbf{e}_{Oz}\ ) \subset \mathbf{F}^6 \tag{5.4.8}$$

we can write:

$$\hat{\mathbf{f}} = n_{Ox}\mathbf{e}_x + n_{Oy}\mathbf{e}_y + n_{Oz}\mathbf{e}_z + f_x\mathbf{e}_{Ox} + f_y\mathbf{e}_{Oy} + f_z\mathbf{e}_{Oz} \tag{5.4.9}$$

that can be rearranged as:

$$\hat{\mathbf{f}}_0 = \begin{bmatrix} n_{Ox} \\ n_{Oy} \\ n_{Oz} \\ f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} \mathbf{n}_O \\ \mathbf{f} \end{bmatrix} \tag{5.4.10}$$

5.5 Scalar Product

The scalar product $\mathbf{m}\cdot\mathbf{f} = \mathbf{f}\cdot\mathbf{m}$ connects the two vectorial spaces M^6 and F^6 with the duality relationship. In Space Vector Algebra $\mathbf{f}\cdot\mathbf{f}$ or $\mathbf{m}\cdot\mathbf{m}$ are not defined.

A dual coordinate system on M^6 and F^6 formed by the basis ($\mathbf{d}_1$ ...$\mathbf{d}_6$) and ($\mathbf{e}_1$ ...$\mathbf{e}_6$) for which:

$$\mathbf{d}_i \cdot \mathbf{e}_i = \begin{pmatrix} 1\ if\ i = j \\ 0\ elsewise \end{pmatrix} \tag{5.5.11}$$

5.6 Coordinate Transforms

Let A and B be two Plucker coordinate systems, each defined by the position and the orientation of a Cartesian frame.

M^6 and F^6 Transform Matrix 1. *Let ${}^B\mathbf{X}_A$ and ${}^B\mathbf{X}^*_A$ be the transformation matrices for M^6 and F^6 from the coordinate system A to the coordinate system B. As a consequence of the properties explained in 5.5:*

$$ {}^{B}\mathbf{X}_{A}^{*} \;=\; {}^{B}\mathbf{X}_{A}^{-T} \tag{5.6.12} $$

5.6.1 Rotation

The spatial vector $\hat{\mathbf{m}} \in M^6$ can be represented by the two 3D vectors m and m_O. If anly a rotation is performed from the coordinate system A to B, the transformation matrix ${}^{B}\mathbf{X}_{A}$ can be expressed as:

$$ {}^{B}\mathbf{X}_{A} \;=\; \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \tag{5.6.13} $$

where $\mathbf{E}$ is the classic matrix of rotation for 3D vectors. For F^6 vectors:

$$ {}^{B}\mathbf{X}_{A}^{*} \;=\; \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \tag{5.6.14} $$

5.6.2 Translation

Let O and P be two generic points in space, and let there be a Cartesian frame located in each point, with the same attitude. The transformation matrix from a vector $\hat{\mathbf{m}}_0 \in M^6$, expressed by $\mathbf{m}$ and $\mathbf{m}_0$ to a vector is $\hat{\mathbf{m}}_P \in M^6$, expressed by $\mathbf{m}$ and $\mathbf{m}_P$ is:

$$ \hat{\mathbf{m}}_P \;=\; \begin{bmatrix} \mathbf{m} \\ \mathbf{m}_O \;-\; \mathbf{r} \times \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{r}\,\mathbf{x} & \mathbf{1} \end{bmatrix} \hat{\mathbf{m}}_O \tag{5.6.15} $$

$$ {}^{P}\mathbf{X}_{O} \;=\; \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{r}\,\mathbf{x} & \mathbf{1} \end{bmatrix} \tag{5.6.16} $$

where $\mathbf{r} \;=\; \bar{OP}$.
For F^6 vectors:

$$ {}^{P}\mathbf{X}_{O}^{*} \;=\; \begin{bmatrix} \mathbf{1} & -\mathbf{r}\,\mathbf{x} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{5.6.17} $$

5.6.3 General Transforms

For tranforms involving both translation and rotations the direct transorms are expressed as:

$$ {}^{B}\mathbf{X}_{A} \;=\; \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{r}\times & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ -\mathbf{E}\,\mathbf{r}\times & \mathbf{E} \end{bmatrix} \tag{5.6.18} $$

$$ {}^{B}\mathbf{X}_{A}^{*} \;=\; \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\mathbf{r}\times \\ -\mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{E} & -\mathbf{E}\,\mathbf{r}\times \\ -\mathbf{0} & \mathbf{E} \end{bmatrix} \tag{5.6.19} $$

5.7 Spatial Cross Product

The cross product can be defined in both the spatial motion vector case and the spatial force vector case. The definition of spatial cross product derives from the definition of derivative. Also in the case of spatial algebra, given the spatial force vector $\hat{\mathbf{f}} \in \mathbf{F}^6$ and the spatial motion vector $\hat{\mathbf{m}} \in \mathbf{M}^6$, moving with the velocity $\hat{\mathbf{v}} \in \mathbf{M}^6$, their derivative can be defined as:

$$\begin{aligned} \dot{\hat{\mathbf{m}}} &= \hat{\mathbf{v}}\, x\, \hat{\mathbf{m}} \\ \dot{\hat{\mathbf{f}}} &= \hat{\mathbf{v}}\, x^*\, \hat{\mathbf{f}} \end{aligned} \tag{5.7.20}$$

where x is defined as the spatial motion vector and x^* is defined as the spatial force vector. From the coordinate point of view the cross product can be obained as :

$$\hat{\mathbf{v}}_O\, x = \begin{bmatrix} \omega\, x & \mathbf{0}_3 \\ \mathbf{v}_0\, x & \omega\, x \end{bmatrix} \tag{5.7.21}$$

and

$$\hat{\mathbf{v}}_O\, x^* = \begin{bmatrix} \omega\, x & \mathbf{v}_0\, x \\ \mathbf{0}_3 & \omega\, x \end{bmatrix} = -\,(\hat{\mathbf{v}}_0\, x)^T \tag{5.7.22}$$

5.8 Momentum

A generical rigid body, with center of mass in the point C, moving free in space is characterized by a spataial velocity $\hat{\mathbf{v}}$, composed by its angular and linear velocities ω and $\mathbf{v}_C$. Then its momentum can be described by the two vectors:

$$\begin{aligned} \mathbf{h} &= m\mathbf{v}_C \\ \mathbf{h}_C &= \mathbf{I}_C\omega \end{aligned} \tag{5.8.23}$$

where m and $\mathbf{I}_C$ are the mass of the rigid body and its moment of inertia about its center of mass. The angular momentum, expressed with respect to a generic point O can be written as:

$$\mathbf{h}_O = \mathbf{h}_C + \mathbf{O\vec{C}}x\mathbf{h} \tag{5.8.24}$$

Defining the spatial 6D vector $\hat{\mathbf{h}}_C$ as:

$$\hat{\mathbf{h}}_C = \begin{bmatrix} \mathbf{I}_C\omega \\ m\mathbf{v}_C \end{bmatrix} \tag{5.8.25}$$

the spatial momentum of a rigid body, with respect to a generic point O van be written as:

$$\hat{\mathbf{h}}_O = \begin{bmatrix} \mathbf{1}_3 & O\vec{C}x \\ \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix} \hat{\mathbf{h}}_C \tag{5.8.26}$$

The spatial momentum is a F^6 vector.

5.8.1 Inertia

The spatial inertia $\bar{\mathbf{I}}_C$ of a rigid body, with respect to its center of mass, is:

$$\bar{\mathbf{I}}_C = \begin{bmatrix} \mathbf{I}_C & \mathbf{0}_3 \\ \mathbf{0}_3 & m\ \mathbf{1}_3 \end{bmatrix} \tag{5.8.27}$$

The spatial inertia of a rigid body with respect to a generic point 0 can be easily computed as:

$$\mathbf{I}_O = \begin{bmatrix} \bar{\mathbf{I}}_C\ +\ m\mathbf{c}x\mathbf{c}x^T & m\mathbf{c}x \\ m\mathbf{c}x^T & m\mathbf{1}_3 \end{bmatrix} \tag{5.8.28}$$

where $\mathbf{c}$ is the vector $\vec{OC}$.

Chapter 6

System Modeling

To describe completely a rigid body system it is necessary to provide data about:

- System's topology
- Component parts description (joints and links geometrical and inertial properties)

6.1 Connectivity

The connectivity of a system can be expressed as a graph with the following properties:

- Each node represents a body
- Each arc represents a joint
- The graph is unidirected
- The graph is connected

In the following some examples are provided to describe the graphs related to different configurations of typical manipulators:

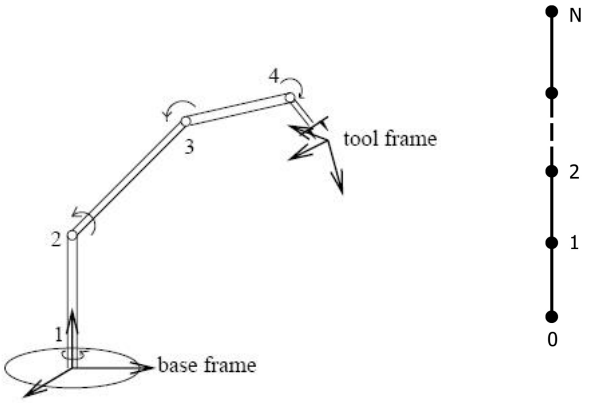


Figure 6.1: Linear manipulator graph.

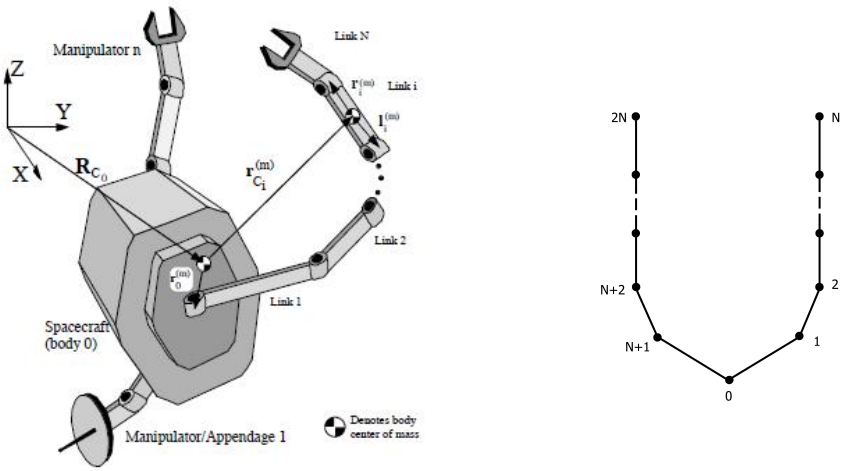


Figure 6.2: Multiple manipulator graph.

A joint is defined as a kinematic constraint between two different bodies (in the case of a manipulator these bodies are called links). It is central in studying mechanical systems to define the *predecessor* and *successor* link with respect to each joint. The definition of the predecessor and successor of a joint defines its *polarity*. It is important to underline that inverting the polarity of a symmetrical joint does not change the allowed type of motion that the joint provides (the case of revolute joints), while a change in the polarity of an unsymmetrical joint causes a complete change in the system. In order to provide the polarity definition of a system's joints the vectors $p(i)$ and $s(i)$ can be used.

Definition 1. *Assuming the base of the manipulator as the system's link* 0*, the vector* $s(i)$ *provides the link successors of the joint* i*, while the vector* $p(i)$ *provides the link predecessors of the joint* i*.*

In the case of a non parallel manipulator the definition of the two vectors is trivial:

$$\mathbf{p} \;=\; [\,0\;1\;2\;3\;...N-1] \tag{6.1.1}$$

$$\mathbf{s} \;=\; [\,1\;2\;3\;4\;...N] \tag{6.1.2}$$

A useful quantity that can be calculated using $p\,(i)$ and $s\,(i)$ is the *parent array* $\lambda\,(i)$, a vector that identifies the parent of each body in a spanning tree. This vector will be extremely useful in the dynamic modeling functions. The parent array is defined as:

$$\lambda\,(i) \;=\; min\,(\,p\,(i)\;,\;s\,(i)\,) \qquad (\,i \;=\; 1...N_B) \tag{6.1.3}$$

Other useful vectors can be built using the information provided by the vectors p and s:

- $k(i)$ (rigid bodies number between the joint i and the base)
- $\mu(i)$ (set of children of the i^{th} body)

In particular in the simulator the user can decide the number of joints and links ,links parameters (mass and inertia), joint type and orientation and position of the first link of the manipulator on the spacecraft. But, despite all the possible combinations of links and joints types, the general architecture of the studied manipulators is:

Figure 6.3: Graph of the studied manipulators.

$$
\begin{aligned}
\mathbf{s} &= \left[\begin{array}{ccccc} 1 & 2 & 3 & \ldots & N \end{array}\right] \\
\mathbf{p} &= \left[\begin{array}{ccccc} 0 & 1 & 2 & \ldots & N-1 \end{array}\right] \\
\lambda &= \left[\begin{array}{ccccc} 0 & 1 & 2 & \ldots & N-1 \end{array}\right] \\
\mathbf{k} &= \left[\begin{array}{ccccc} 1 & 2 & 3 & \ldots & N \end{array}\right] \\
\mu &= \left[\begin{array}{ccccc} 1 & 2 & 3 & \ldots & N \end{array}\right]
\end{aligned}
\tag{6.1.4}
$$

As it will be explained in the next chapters, the kinematics and dynamics algorithms implemented in the simulator can be easily extended to more general configurations of the system (the most interesting one is considered the case of multiple manipulators).

6.2 Body Reference Systems

The reference system labeled as 0 is the inertial reference system. The reference frames labeled as F_{ij} are frames centered at the barycenter of the link between the joint i and j, with the x axis along the link direction. The reference frames labeled as F_i are frames centered in the joint i, with the x axis along the successor link direction. Obviously all the reference frames are orthogonal right handed reference frames.

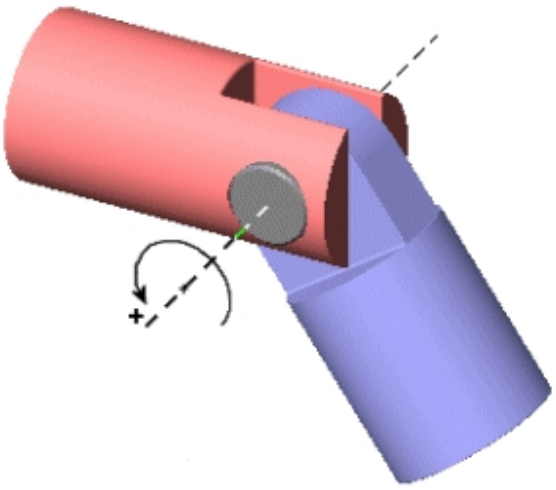
6.3 Joint Models

Defintion 1. *A joint can be seen as a constraint that describes the allowed motion between two reference frame.*

In a complex mechanical systems it is extremely important to model the joints in the best possible way. A complete joint model should provide the description of the transform between the two consecutives joint frames (${}^{s}\mathbf{X}_{p}$), the velocity of the joint $\mathbf{v_J}$ and the motion subspace $\mathbf{S}$. The developed software provides a joint library, in order to allow the user to decide in complete autonomy the configuration of the system. In the following the main joints models are presented. The rotational, prismatic, planar and 6-DoF joints have been implemented. Other joint models are avaiable looking at the literature (for example cylindrical, helical and spherical joints).

6.3.1 Revolute/Hinge Joint

Definition 1. *The revolute Joint is a joint that permit only the rotation of the second reference frame around only one of the axes of the first reference frame.*



The Joint transform for a revolute joint around the i axis is characterized by:

$$\mathbf{E} = rot\,(i) \tag{6.3.5}$$

$$\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.3.6}$$

The Motion Subspace and the Constraint Force Subspace **T** for a revolute joint around the x axis **S** can be represented as:

$$\mathbf{S} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.3.7}$$

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.8}$$

The Motion Subspace and the Constraint Force Subspace **T** for a revolute joint around the y axis **S** can be represented as:

$$\mathbf{S} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.3.9}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.10}$$

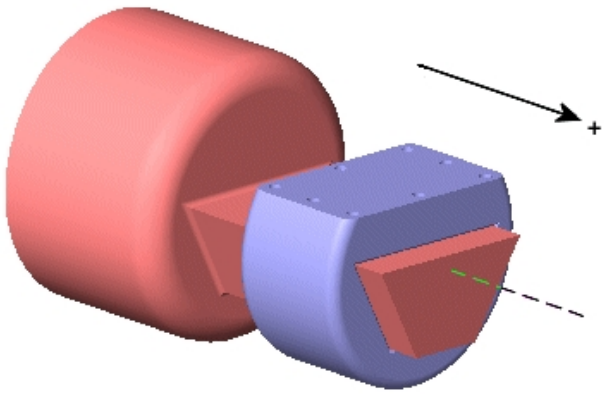
The Motion Subspace and the Constraint Force Subspace **T** for a revolute joint around the z axis **S** can be represented as:

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.3.11}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.12}$$

6.3.2 Prismatic/Sliding Joint

Definition 1. *The Prismatic Joint is a joint that allows only the translational motion along one of the axis of the joint itself.*



The transform for a prismatic joint on the i axis can be modeled as:

$$\mathbf{E} = \mathbf{1}_{3x3} \tag{6.3.13}$$

$$\mathbf{r} = q_1 \, \mathbf{e}_i \tag{6.3.14}$$

where $\mathbf{e}_i$ is the versor of the i^{th} axis of the joint and q_1 is the joint translation in the i^{th} direction.

The Motion Subspace and the Constraint Force Subspace for a prismatic joint in the x direction is :

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{6.3.15}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.16}$$

The Motion Subspace and the Constraint Force Subspace for a prismatic joint in the y direction is :

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{6.3.17}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.18}$$

The Motion Subspace and the Constraint Force Subspace for a prismatic joint in the z direction is :

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.3.19}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.3.20}$$

6.3.3 Prismatic and Revolute Joints Conventions

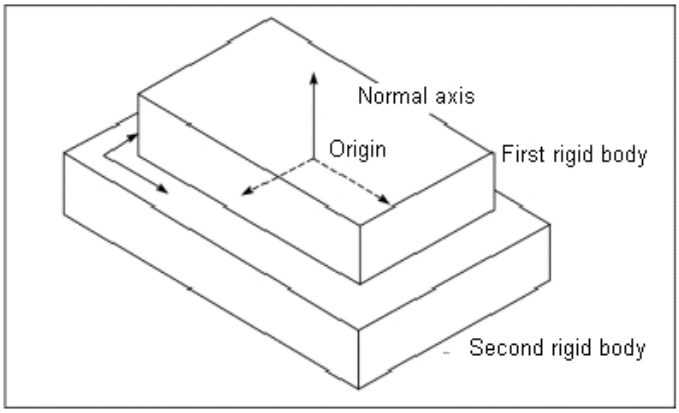
It is extremely important to underline that the definition of the axis of rotation or the definition of direction of motion of a prismatic joint are expressed in the reference system of the *predecessor* link. This can cause difficulties in the definition of such axes, if two adjoining links have no common axes. Indeed in the simulation

tests the initial conditions of the joint angles have been always put to 0 *deg.*, in order to have a full deployied manipulator, simplifying the axes definition.

6.3.4 Planar Joint

Defintion 1. *The Planar Joint is a joint that constraints the motion of the following link upon a plane.*

Therefore the joint model is characterized by three joint coordinates: q_1 is the rotational degree of freedom, q_2 and q_3 are the translation in the joint x and y direction.



The Joint transform ${}^{s}\mathbf{X}_{p}$ can be modeled using:

$$\mathbf{E} = rot_i(q_1) \tag{6.3.21}$$

$$\mathbf{r} = \begin{bmatrix} q_2 \cos q_1 - q_3 \sin q_1 \\ q_2 \sin q_1 + q_3 \cos q_1 \end{bmatrix} \tag{6.3.22}$$

The Motion Subspace and the Constraint Force Subspace can be modeled as:

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.3.23}$$

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3.24}$$

6.3.5 6-DoF Joint

The 6DoF is not a real, physical joint; but it allows to model a rigid body free to move in space using the same conventions and mathematical structure of the classic prismatic and revolute joints. The 6DoF joint model strictly depends on the selected parametrization for the body's attitude. The software implementation uses a classic Euler Angles (sequence 123) parametrization. The joint is characterized by the six joint variables:

$$\mathbf{q} = \begin{pmatrix} \phi \\ \theta \\ \psi \\ s_x \\ s_y \\ s_z \end{pmatrix} \tag{6.3.25}$$

Where ϕ is the rotation around the 3rd axis, θ is the rotation around the 2nd axis and ψ is the rotation around the 1st axis. The translational position of the body is expressed in the body coordinate system. The angular rate, in body coordinates, can be written as function of the Euler angles rates:

$$\begin{aligned} \omega_x &= \cos\psi\cos\theta\dot{\phi} + \sin\psi\dot{\theta} \\ \omega_y &= -\sin\psi\cos\theta\dot{\phi} + \cos\psi\dot{\theta} \\ \omega_z &= -\sin\theta\dot{\phi} + \dot{\psi} \end{aligned} \tag{6.3.26}$$

As a consequence, the $\mathbf{S}$ matrix can be simply modeled as:

$$\mathbf{S} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi & 0 & 0 & 0 & 0 \\ -\sin\psi\cos\theta & \cos\psi & 0 & 0 & 0 & 0 \\ \sin\theta & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3.27}$$

This joint is characterized by a non null $\dot{\mathbf{S}}$ matrix, in particular, deriving the Equations 6.3.26 leads to :

$$\dot{\mathbf{S}} = \begin{bmatrix} -\dot{\psi}\sin\psi\cos\theta - \dot{\theta}\cos\psi\sin\theta & \dot{\psi}\cos\psi & 0 & 0 & 0 & 0 \\ -\dot{\psi}\cos psi\cos\theta + \dot{\theta}\sin\psi\sin\theta & -\dot{\psi}\sin\psi & 0 & 0 & 0 & 0 \\ \dot{\theta}\cos\theta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.3.28}$$

The motion constraint matrix is simply:

$$\mathbf{T} = \mathbf{0}_3 \tag{6.3.29}$$

For both the planar and the 6DoF the translational variables are expressed in the successor link reference frame. In the computer implementation of the joints both the $\mathbf{S}$ and the $\mathbf{T}$ matrices have been extended to the full dimension $[]_{6x6}$ in order to avoid dimension inconsistencies that could arise using Embedded Matlab.

6.3.6 Modeled Systems

Using these joint models it is possible to describe a large number of possible configurations. The simulator gives to the user the possibility to decide completely the architecture, the types and configuration of the joints. Despite the large number of configurations that can be simulated, they can all be grouped into the following model

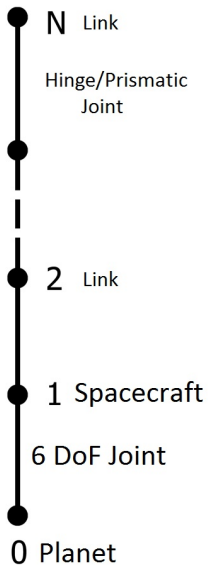


Figure 6.4: Graph of the simulated manipulators.

$$
\begin{aligned}
\mathbf{s} &= \begin{bmatrix} 1 & 2 & 3 & \dots & N \end{bmatrix} \\
\mathbf{p} &= \begin{bmatrix} 0 & 1 & 2 & \dots & N-1 \end{bmatrix} \\
\lambda &= \begin{bmatrix} 0 & 1 & 2 & \dots & N-1 \end{bmatrix} \\
\mathbf{k} &= \begin{bmatrix} 1 & 2 & 3 & \dots & N \end{bmatrix} \\
\mu &= \begin{bmatrix} 1 & 2 & 3 & \dots & N \end{bmatrix}
\end{aligned}
\tag{6.3.30}
$$

The experimental setup, on the other side, has a fixed structure: as previously explained the manipulator is a 4 DoF manipulator composed only by revolute joints, connected to a base whose motion is planar and described by a planar joint.The topology of the experimental setup is presented below:

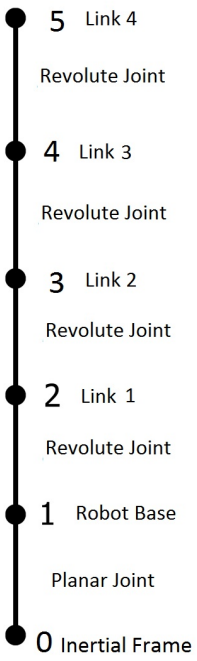


Figure 6.5: Experimental system graph.

$$\begin{aligned}
\mathbf{s} &= \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \\
\mathbf{p} &= \begin{bmatrix} 0 & 1 & 2 & 3 \end{bmatrix} \\
\lambda &= \begin{bmatrix} 0 & 1 & 2 & 3 \end{bmatrix} \\
\mathbf{k} &= \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \\
\mu &= \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}
\end{aligned} \tag{6.3.31}$$

Chapter 7

Dynamics

7.1 Introduction

The canonical equations of motion for a manipulator system (for both industrial manipulator with fixed base and space manipulator with mobile base) can always be organized in the scheme:

$$\mathbf{H}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\,\dot{\mathbf{q}}) = \mathbf{Q} \tag{7.1.1}$$

Where $\mathbf{H}(\mathbf{q},\ model)$ is the inertia matrix of the manipulator system, function of the model parameters and of the joints positions. $\mathbf{C}(\mathbf{q},\ \dot{\mathbf{q}},\ model)$ is the bias force matrix. It contains the Coriolis, gravity and centrifugal forces acting on the system. $\mathbf{C}$ is a function of the model parameters, of the Joints positions and velocities. The matrix $\mathbf{C}$ can be interpreted as the joint forces/torques that produce no acceleration. $\mathbf{Q}$ is the vector of generalized forces: in the case of no external forces/torques the vector contains the actions applied to the joints of the system. The underlined dependencies show the strong non linearity of the equations that model the dynamics of a manipulator system (general basics concepts on robot dynamics can be found in [30], [20], [26] and [25])

Using the spatial algebra formulation it is possible to model the spacecraft itself as one of the joint of a conventional fixed base manipulator. Using the models provided in Chapter 6 for the Planar and 6 DoF joints, it possible to model the allowed degrees of freedom of the spacecraft (3 DoF in the planar experimental case and 6 DOFs in the complete 3D case). The clear advantage of this strategy is that, using this technique, all the possible configurations of spacecrafts plus manipulators can be treated as conventional fixed base manipulator. In this way, all the kinematics, dynamics and control algorithms developed for conventional manipulators can be used also for a mobile manipulator. This condition shares the philosophy of the *Virtual Manipulator* created by Vafa and Dubovsky,[8], [9], [10], [32],[11].

7.2 Inverse and Forward Dynamics

The dynamic resolution of a mechnical system can be organized into two main categories:

- **Inverse Dynanmics**
- **Forward Dynamics**

7.2.1 Inverse Dynamics

The goal of an Inverse Dynamics function is to provide the joint forces and torques that cause a desired system evolution in time. An inverse dynamics function can be schematically presented as:

$$\mathbf{Q} \;=\; ID\;(\mathbf{q},\;\dot{\mathbf{q}},\;\ddot{\mathbf{q}},\;model) \tag{7.2.2}$$

Using an inverse dynamics function it is possible to calculate the matrices **H** and **C**.Indeed it is easy to verify that in the case of $\ddot{\mathbf{q}} \;=\; \mathbf{0}$:

$$\mathbf{C}\,(\mathbf{q},\;\dot{\mathbf{q}}) \;=\; \mathbf{Q} \;=\; ID\;(\mathbf{q},\;\dot{\mathbf{q}},\;\ddot{\mathbf{q}},\;model) \tag{7.2.3}$$

Defining the function ID_δ as:

$$ID_\delta = ID\,(\mathbf{q},\;\dot{\mathbf{q}},\;\ddot{\mathbf{q}},\;model) -\; ID\,(\mathbf{q},\;\dot{\mathbf{q}},\;\delta,\;model) \tag{7.2.4}$$

the *ith* coloumn of the matrix **H** ca be calculated as:

$$\mathbf{H}_i \;=\; ID_\delta\,(\ddot{\mathbf{q}}, \delta,\;model) \tag{7.2.5}$$

where δ is a column vector defined as:

$$\delta \;=\; \begin{pmatrix} \delta_j \;=\; 0 & if\; j = \;i \\ \delta_j \;=\; 1 & if\; j = \;i \end{pmatrix} \tag{7.2.6}$$

7.2.2 Forward Dynamics

The goal of the Forward Dynamics Modeling is to provide the joint acceleration, velocity and position, given the forces and torques on the acting system. A forward dynamics function ca be schematically presented as:

$$\ddot{\mathbf{q}} \;=\; ID\;(\mathbf{Q},\;model) \tag{7.2.7}$$

The joint velocities and positions can be easily obtained simply integrating the joint acceleration. Despite a lot of forward dynamics functions have been presented, the simple process:

$$\ddot{\mathbf{q}} \;=\; \mathbf{H}^{-1}\;(\mathbf{Q} \;-\; \mathbf{C}) \tag{7.2.8}$$

has been used in the implementation.

7.3 Inverse Dynamics Newton-Euler Algorithm

The following implementation of the Newton Euler Algorithm (in joint coordinates and using the Spatial Algebra) can be found in [25] and [26]. In the software the Inverse Dynamics Newton-Euler Algorithm has been implemented in order to compute the dynamic matrices of the system. The knowledge of the Dynamic matrices is necessary in order to:

- perform the direct integration of the equations of motion, in order to simulate the responses of the system;
- compute non linear, model-based control laws.

The function has been implemented using the Spatial Algebra tool: this permits to highly simplify the structure of the function itself, leading to a more compact formulations of the algorithm. At the same time this implementation takes advantages of the developed joint models presented in Chapter 6 Given the joint positions and velocities the function has to:

- Rearrange vector sizes;
- Update the transform matrices between the joint reference frames;
- Calculate the joint velocities and accelerations;
- Calculate the forces/torques allowed by each joint model.

In the following lines the pseudo code of the Newton Euler algorithm expressed in body coordinates is presented:

$$\tau \;=\; Newton - Euler\,(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, Model)$$

$$
\begin{aligned}
& v_0 \;=\; 0 \\
& a_0 1; = \; 0 \\
& for\; i \;=\; 1 \;:\; N \\
& [\,\mathbf{X}_j\; \mathbf{S}_i\; \mathbf{v}_j\; \mathbf{c}_j] \;=\; joint function\,(\; joint - type,\; \mathbf{q}_i,\; \dot{\mathbf{q}}_i) \\
& {}^{i}\mathbf{X}_{\lambda(i)} \;=\; \mathbf{X}_j \mathbf{X}_T \\
& if\; \lambda\,(i) \;\neq\; 0 \\
& {}^{i}\mathbf{X}_0 \;=\; {}^{i}\mathbf{X}_{\lambda(i)}{}^{\lambda(i)}\mathbf{X}_0 \\
& end \\
& \mathbf{v}_{ji} \;=\; \mathbf{S}_i\; \dot{\mathbf{q}}_i \\
& \mathbf{v}_i \;=\; {}^{i}\mathbf{X}_{\lambda(i)}\; \mathbf{v}_{\lambda(i)} \;+\; \mathbf{v}_{ji} \\
& \mathbf{c}_j \;=\; \dot{\mathbf{S}}_i \dot{\mathbf{q}}_i \\
& \mathbf{a}_i \;=\; {}^{i}\mathbf{X}_{\lambda(i)}\; \mathbf{a}_{\lambda(i)} \;+\; \mathbf{S}_i\; \ddot{\mathbf{q}}_i \;+\; \mathbf{c}_j \;+\; \mathbf{v}_i\; x\; \mathbf{v}_l \\
& \mathbf{f}_i \;=\; \mathbf{I}_i\; \mathbf{a}_i \;+\; \mathbf{v}_i\; x^*\; \mathbf{I}_i \mathbf{v}_i \;-\; {}^{i}\mathbf{X}_0^*\; \mathbf{f}_i^x
\end{aligned}
$$

end
$for\ i\ =\ N:-1:1$
$\tau_i\ =\ \mathbf{S}_i^T \mathbf{f}_i$
$if\ \lambda(i)\ \neq\ 0$
$\mathbf{f}_{\lambda(i)}\ =\ \mathbf{f}_{\lambda(i)}\ +\ {}^{\lambda(i)}\mathbf{X}_i^*\ \mathbf{f}_i$
end
end

The presented algorithm is written in joint coordinates: the output vector τ is the vector of joint forces and torques acting at the joint in the local joint reference frame. It easy to understand that this set up is extremely useful: no transformation is required in order to obtain the actual value of the control action to give to the actuators. In the 2D case the vector $\mathbf{q}$ ha been rearranged as:

$$\mathbf{q}_{2D}\ =\ \begin{bmatrix} \theta \\ x_s \\ y_s \\ q_{Joint\ 1} \\ \vdots \\ q_{Joint\ n} \end{bmatrix} \tag{7.3.9}$$

where x_s and y_s are the translation of the base spacecraft (expressed in the body reference system) and θ is the attitude of the spacecraft. In the 3D case the attitude of the spacecraft can be expressed using many different parametrizations, such as Euler Angles, Euler parameters or quaternions. The Euler angles case is presented here:

$$\mathbf{q}_{3D}\ =\ \begin{bmatrix} \phi \\ \theta \\ \psi \\ x_s \\ y_s \\ z_s \\ q_{Joint\ 1} \\ \vdots \\ q_{Joint\ n} \end{bmatrix} \tag{7.3.10}$$

As previously explained the Newton Euler algorithm has been implemented not to solve the inverse dynamics but in order to update step by step the dynamic matrices. One of the peculiarities of this algorithm is the possibility to implement it using recursive techniques. On the other side Matlab Embedded Code does not support recursion, so, the implementation of the software does not use this attractive feature. The dynamic matrices calculation represents the most expansive function of the entire software in terms of computational time. Because of this, it

is really important to optimize the code implementation in both the case of the Simulink Simulator Software and the case of the executable software.

7.3.1 Validation

The Newton-Euler algorithm is an algorithm optimized for the numerical implementation. As a consequence of this, the algorithm cannot be easily implemented with the goal to obtain a symbolic expression for the dynamic matrices. This means that a direct validation of the obtained matrices is not possible. The validity of the implementation has to be verified in different ways. First of all it is mandatory to verify if the inertia matrix is symmetrical and positive definite. After this preliminary demonstration, the conservation of the fundamental physical quantities must be verified. If the system is not subjected to external or dissipative forces it is possible to verify that:

- Matrix $\mathbf{H}$ must be symmetrical positive definite
- The linear momentum is a constant function.
- The angular momentum is a constant function.
- The kinetic energy is constant.
- The barycenter of the entire system moves in rectilinear uniform motion.

Giving to the system non-null initial conditions, the calculated dynamics must be consistent with the above conditions. The check of these conditions represents a good validation for the implementation of dynamic functions. The implementation has been validated using two different system's configurations:

- 4 DoF Planar Manipulator, connected to a 3 DoF spacecraft (7 DoF in total)
- 4 DoF Fully Spatial manipulator, connected to a 6 DoF spacecraft (10 DoF in total)

To verify the conditions regarding the matrix $\mathbf{H}$ can be sufficient to use the Cholesky algorithm to calculate $\mathbf{H}^{-1}$ in order to solve the forward dynamic problem 7.2.8. According to the Spatial Algebra conventions the angular and linear momentum of each link of the system, with respect to the point 0 can be computed as:

$$\mathbf{h_0} = \begin{bmatrix} \mathbf{I}_i & \mathbf{c}_i\, x \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{I}_i & m_i\, \omega_i\, x \\ \mathbf{0}_3 & m_i\, \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \omega_i \\ \mathbf{v}_i \end{bmatrix} \tag{7.3.11}$$

while the kinetic energy can be computed as:

$$T = \frac{1}{2} \begin{bmatrix} \omega_i \\ \mathbf{v}_i \end{bmatrix}^T \begin{bmatrix} \mathbf{I}_i & m_i\, \omega_i\, x \\ \mathbf{0}_3 & m_i\, \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \omega_i \\ \mathbf{v}_i \end{bmatrix} \tag{7.3.12}$$

The following initial conditions have been implemented into the system in the case of a planar manipulator, connected to a planar free flyer robot (experimental setup case):

$$\mathbf{q} = \begin{bmatrix} \theta = 0\,rad \\ s_x = 0\,m \\ s_y = 0\,m \\ q_1 = 0\,rad \\ q_2 = 0\,rad \\ q_3 = 0\,rad \\ q_4 = 0\,rad \end{bmatrix} \tag{7.3.13}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\theta} = 0.7\,rad/s \\ v_x = 0.7\,m/s \\ v_y = 0.7\,m/s \\ \dot{q_1} = 0.7\,rad/s \\ \dot{q_2} = 0.7\,rad/s \\ \dot{q_3} = 0.7\,rad/s \\ \dot{q_4} = 0.7\,rad/s \end{bmatrix} \tag{7.3.14}$$

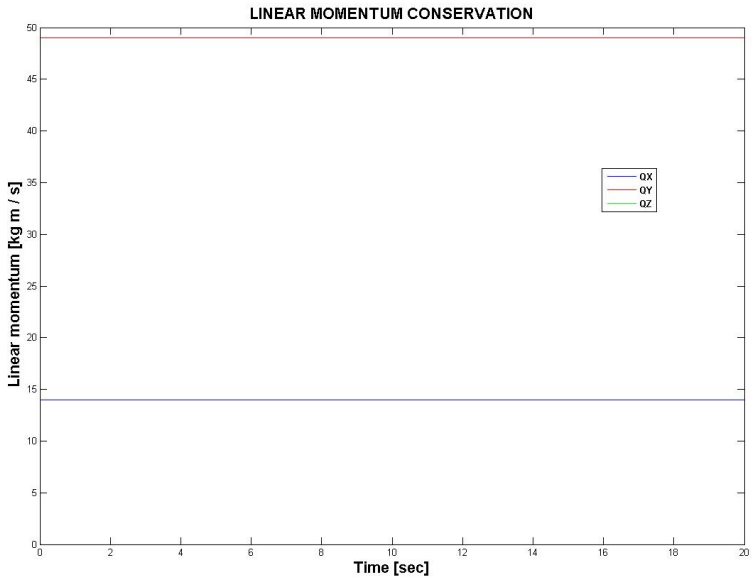


Figure 7.1: Linear Momentum Conservation, 3 DoF spacecraft, 4 DoF planar manipulator

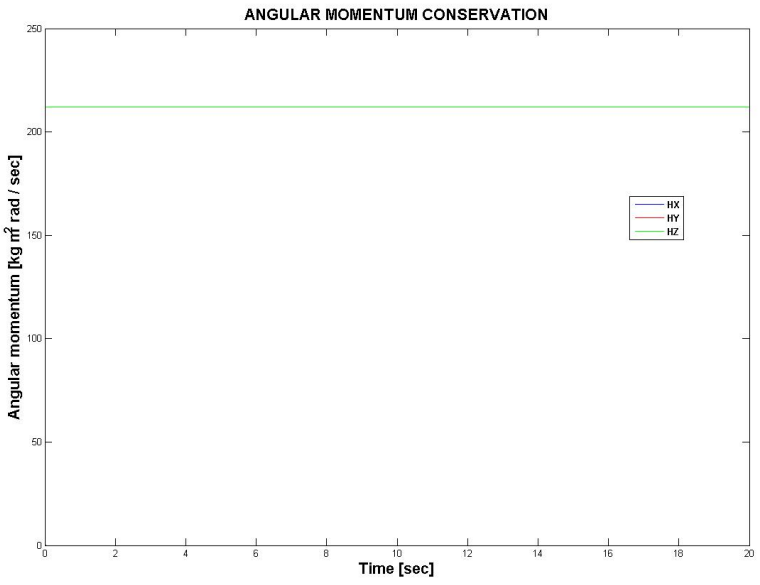


Figure 7.2: Angular Momentum Conservation, 3 DoF spacecraft, 4 DoF planar manipulator

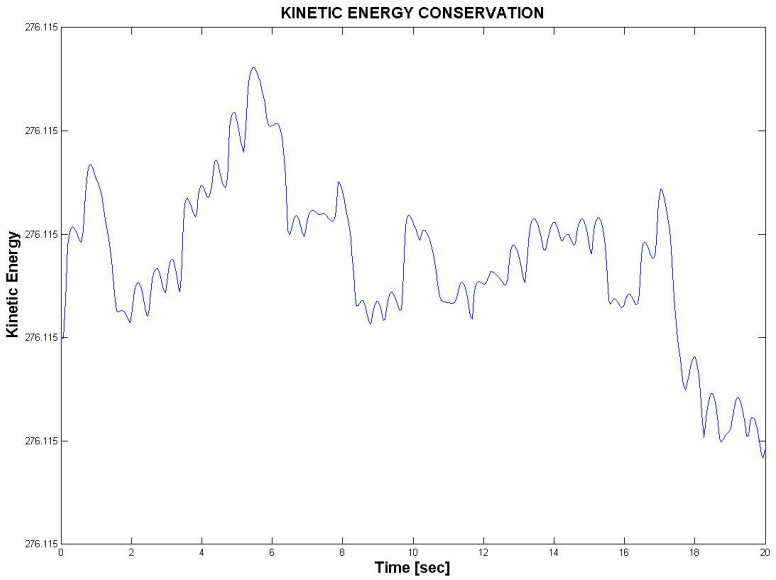


Figure 7.3: Kinetic Energy Conservation, 3 DoF spacecraft, 4 DoF planar manipulator

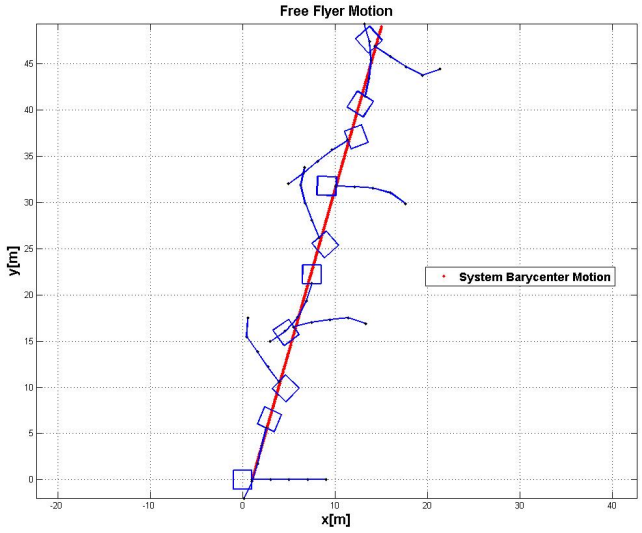


Figure 7.4: System motion, 3 DoF spacecraft, 4 DoF planar manipulator

In the case of completely 3D system the following initial conditions have been assigned to the system:

$$\mathbf{q} = \begin{bmatrix} \phi = 0\ rad \\ \theta = 0\ rad \\ \psi = 0\ rad \\ s_x = 0\ m \\ s_y = 0\ m \\ s_z = 0\ m \\ q_1 = 0\ rad \\ q_2 = 0\ rad \\ q_3 = 0\ rad \\ q_4 = 0\ rad \end{bmatrix} \tag{7.3.15}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\phi} = 0.1\ rad/s \\ \dot{\theta} = 0.1\ rad/s \\ \dot{\psi} = 0.1\ rad/s \\ v_x = 0.1\ m/s \\ v_y = 0.1\ m/s \\ v_z = 0.1\ m/s \\ \dot{q_1} = 0.1\ rad/s \\ \dot{q_2} = 0.1\ rad/s \\ \dot{q_3} = 0.1\ rad/s \\ \dot{q_4} = 0.1\ rad/s \end{bmatrix} \tag{7.3.16}$$

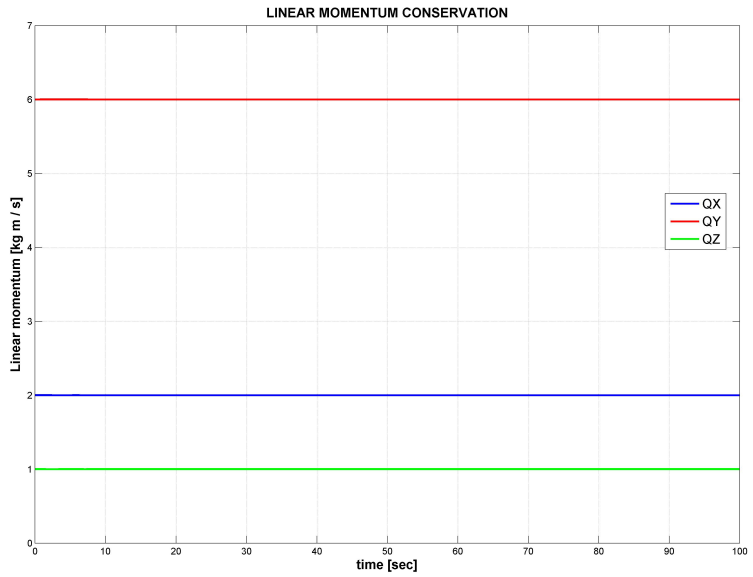


Figure 7.5: Linear Momentum Conservation, 6 DoF spacecraft, 4 DoF manipulator

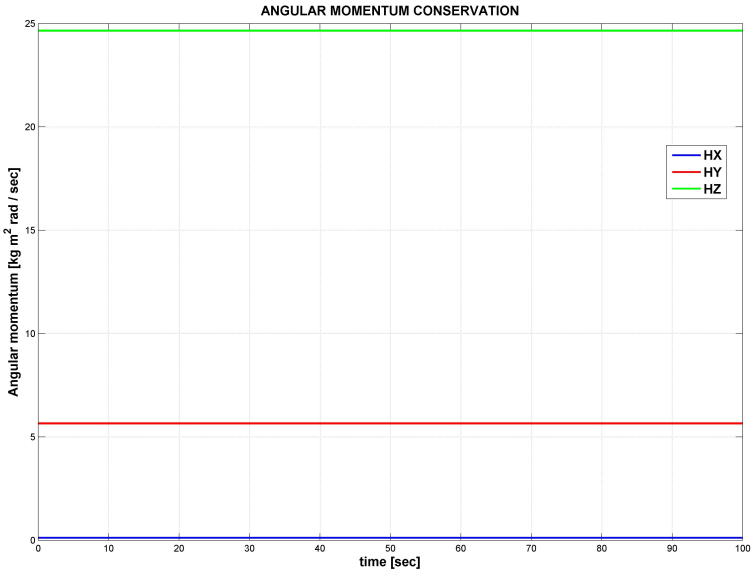


Figure 7.6: Angular Momentum Conservation, 6 DoF spacecraft, 4 DoF manipulator

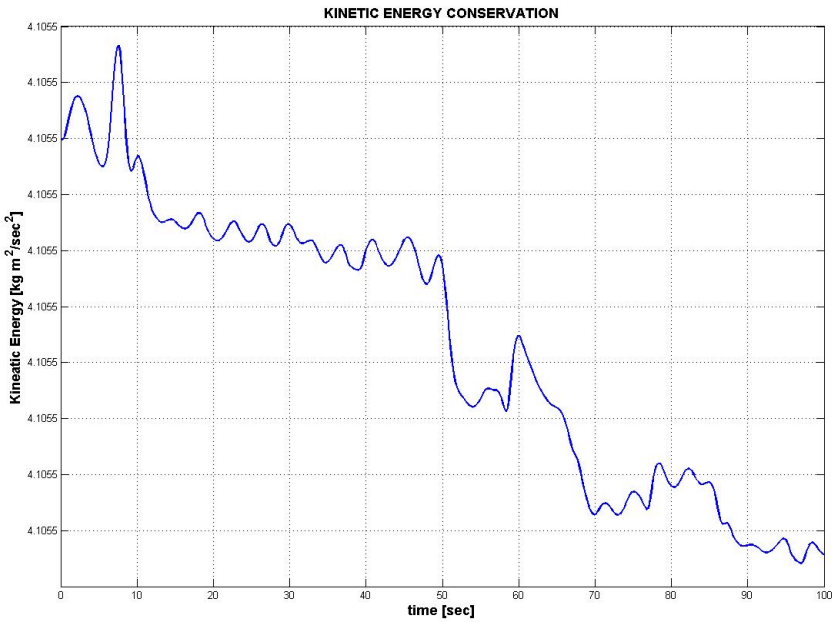


Figure 7.7: Kinetic Energy Conservation, 6 DoF spacecraft, 4 DoF manipulator

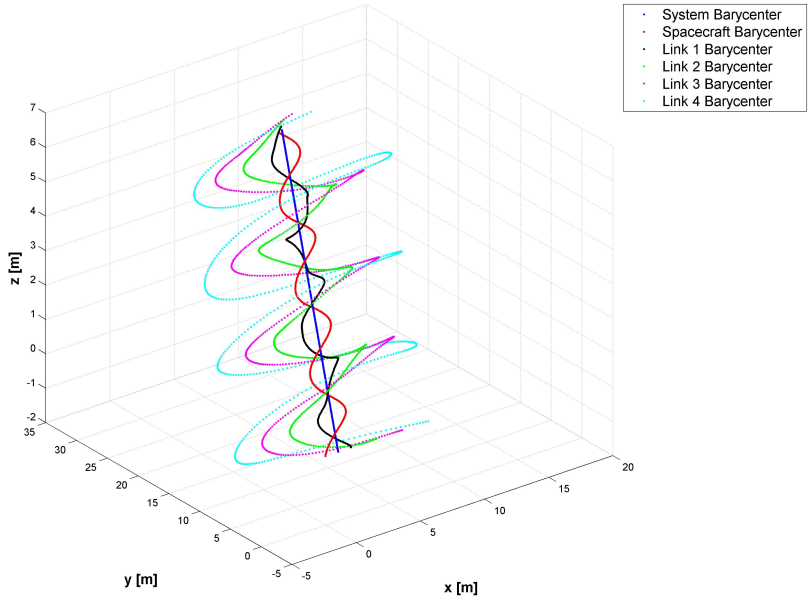


Figure 7.8: System motion, 6 DoF spacecraft, 4 DoF manipulator

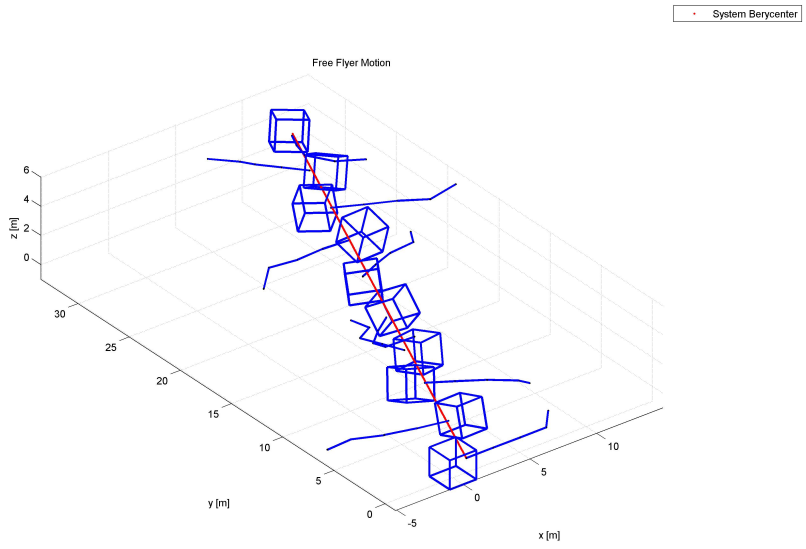


Figure 7.9: System motion visulization, 6 DoF spacecraft, 4 DoF manipulator

7.3.2 Conclusions

The tests performed on the implemented function proved the validity of the approach. In both the 2D and 3D cases, the function computed the dynamics of the system respecting the physical conservation principles. In Fig and the trend of the kinetics energy seems not to be constant, but looking carefully at the scale on the y-axes, the variations of the system's kinetic energy is in the range of the Matlab numerical approximation errors.
This implementation is extremely advantageous because it is based on the the joint reference frame: this means that the user is not forced to rotations and reference changes in order to compute the joint movements and forces and torques in their own reference frame. Moreover the Spatial Algebra permits to implement a large amount of joint models using always the same mathematical formalism and procedures: this is extremely advantageous because it allows treat the spacecraft movements as a normal manipulator joint.

Chapter 8

Control System

The system can be described by the manipulator canonical equation:

$$\mathbf{H}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) = \mathbf{Q} \tag{8.0.1}$$

where the vector $\mathbf{q}$ represents the joint variables, [25]. As previously underlined the system is strongly non linear. The goal of the thesis is related to the application of a trajectory tracking technique, in order to use the manipulator for complex operations. In order to follow a trajectory large movements of the joints themselves could be necessary. In this context the classical linearization techniques are not useful: these approaches are based on the assumption of small excursions of the joint variables and, as a consequence, these techniques can be used effectively only for the disturbance rejection and for the maintenance of an equilibrium position.
The tracking control has to allow large joint movements, breaking the hypothesis at the base of all the linearization techniques. The most useful approach is related to the application of a non linear control technique. In particular the class of the *feedback linearization algorithms* seems extremely attractive for the tracking control of complex non linear systems ([20], [30], [33],[11],[32]).

Definition 1. *A feedback linearization algorithm is a control technique that permits to generate a particular control law that, closing the feedback loop, linearizes the system.*

In the implementation the *Computed Torque Control Law* has been implemented.

8.1 Computed Torque Control

The Computed Torque Control is a feedback linearization control technique that requires the complete knowledge of the system properties, in particular the knowledge of the actual inertial and bias matrix $\mathbf{H}$ and $\mathbf{C}$. The forces/torques to apply

to the system are given by the relation, [30],[20]:

$$\mathbf{Q} = \mathbf{Hu} + \mathbf{C} \tag{8.1.2}$$

Applying these generalized forces to the system:

$$\mathbf{H\ddot{q}} + \mathbf{C} = \mathbf{Hu} + \mathbf{C}$$

Doing the elementary calculations the feedback controlled system reduces to:

$$\ddot{\mathbf{q}} = \mathbf{u} \tag{8.1.3}$$

that represents a simple double integrator dynamic system. Using this algorithm the stability properties of the system are completely determined by the control action $\mathbf{u}$. Given the desired acceleration of the joints $\ddot{\mathbf{q}}_{des}$, and defined the error in joint position and velocity:

$$\mathbf{e} = \mathbf{q}_{des} - \mathbf{q} \tag{8.1.4}$$

$$\dot{\mathbf{e}} = \dot{\mathbf{q}}_{des} - \dot{\mathbf{q}} \tag{8.1.5}$$

$$\ddot{\mathbf{e}} = \ddot{\mathbf{q}}_{des} - \ddot{\mathbf{q}} \tag{8.1.6}$$

the most common form for the control action u is:

$$\mathbf{u} = \ddot{\mathbf{q}}_{des} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} \tag{8.1.7}$$

that, applied to the feedback controlled system brings to:

$$\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} = 0 \tag{8.1.8}$$

The error dynamics is now described by a Ordinary Differential Equations System of the second order. The easiest way to select properly the gain matrices is to choose $\mathbf{K}_D$ and $\mathbf{K}_P$ as diagonal matrices. This choice brings to the implementation of a decoupled/independent joint control: each joint, in this configuration, is controlled independently from the others. Choosing diagonal gain matrices the sufficient condition in order t achieve the asymptotic stability of the controlled system is that the gains k_{Pi} and k_{Di} are roots of the fundamental equation:

$$s^2 + k_{Di}s + k_{Pi} = 0 \tag{8.1.9}$$

A possible choice is to interpret the gains as:

$$k_{Pi} = \omega_i^2 \tag{8.1.10}$$

$$k_{Di} = 2\zeta_i\omega_i \tag{8.1.11}$$

Using this expression for the control action $\mathbf{u}$, the commanded forces and torques on the system are:

$$\mathbf{Q} = \mathbf{H}\left(\ddot{\mathbf{q}}_{des} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}\right) + \mathbf{C} \tag{8.1.12}$$

The method can be summarized using a classical block scheme :

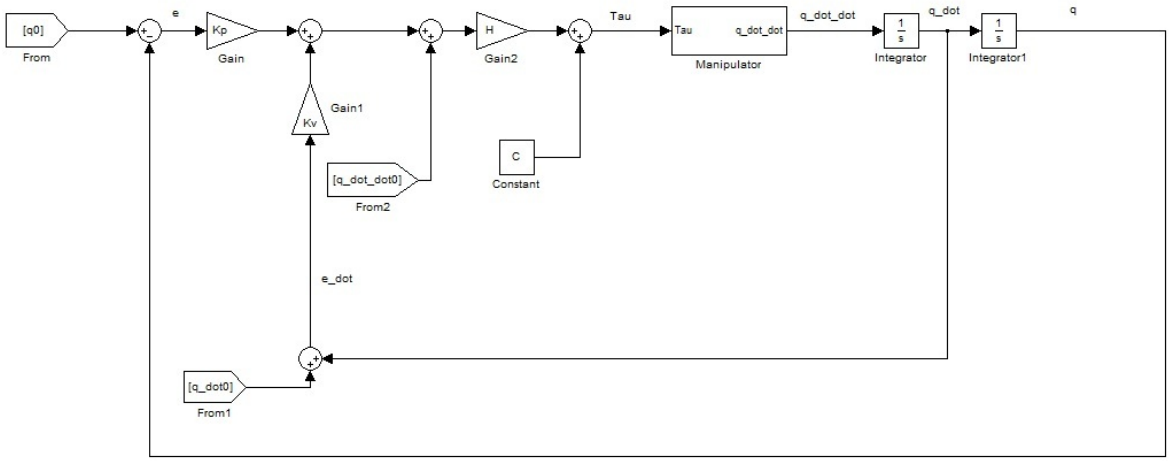


Figure 8.1: Computed Torque method block scheme.

The Computed Torque method is extremely attractive because it permits to work with a linearized system. On the other side it requires to calculate step by step the dynamic matrices $\mathbf{H}$ and $\mathbf{C}$. This could be extremely challenging in real-time applications. A lot of techniques have been developed to reduce the computational cost of the algorithm: this class of algorithms is called *Computed Torque-Like Control* (*Variable Structure Compensation*, *Independent Joint Compensation*, ecc). The Spatial Algebra formulation of the Newton Euler Algorithm for Inverse Dynamics revealed to be, on the other side, enough fast to be processed in real time. As mentioned above, the application of the Computed Torque Control implies the deep knowledge of the parameters of the entire system in order to have the most possible accurate computation of $\mathbf{H}$ and $\mathbf{C}$. The perfect elimination of the non linearities can be done only in the computer simulation, where the matrices involved in the dynamics and in the control law are the same. In the practical case the computed matrices contain errors and uncertainties with respect to the real matrices (the matrices used to compute the control forces are generated by the Newton Euler Dynamic algorithm, using the joint variables measured by the sensors, while the matrices of the motion equations are the real matrices of the robot system). In this condition the non linearities cannot be deleted and the effect of this is a deviation from the ideal feedback linearization behavior:

$$\mathbf{H}\ddot{\mathbf{q}} + \mathbf{C} = \tilde{\mathbf{H}}(\mathbf{u}) + \tilde{\mathbf{C}} \tag{8.1.13}$$

$$\tilde{\mathbf{H}}\mathbf{u} - \mathbf{H}\ddot{\mathbf{q}} + \left(\tilde{\mathbf{C}} - \mathbf{C}\right) = 0 \tag{8.1.14}$$

$$\tilde{\mathbf{H}}(\ddot{\mathbf{q}}_{des} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}) - \mathbf{H}\ddot{\mathbf{q}} + \left(\tilde{\mathbf{C}} - \mathbf{C}\right) = 0 \tag{8.1.15}$$

where $\mathbf{H}$ and $\mathbf{C}$ are the matrices of the real system, while $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{C}}$ are the computed dynamic matrices. A possible way to reduce the computational cost of the Computed Torque Control consists in computing just the diagonal of the $\mathbf{H}$ matrix, decreasing the computational cost of the algorithm from n^2 to n, where n is the number of the system DoFs. This technique can reveal particularly useful in practical implementation characterized by an high number of DoFs. In this case the control law can be written as:

$$\mathbf{Q} \;=\; Diag\,(\mathbf{H})\;(\ddot{\mathbf{q}}_{Des} \;+\; \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}) \;+\; \mathbf{C} \tag{8.1.16}$$

The closed loop controlled system can no more be reduced to a double integrator system:

$$\mathbf{H}\ddot{\mathbf{q}} = \;\; Diag\,(\mathbf{H})\;(\ddot{\mathbf{q}}_{Des} \;+\; \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}) \tag{8.1.17}$$

8.1.1 Spacecraft Control

The Computed Torque Control has been used to compute both the torques required at the manipulator joints and the forces and torque required by the spacecraft. The joints of a space manipulator are usually equipped with electric motors that allow the application of continuous torque profiles, the spacecraft is equipped with jet thrusters, that can provide only two level of thrust: 0 or T_{max}. As a consequence, the provided continuous thrust profile has to be converted in a sequence of impulses *on-off* to be sent to the thrusters controller.
First of all it is necessary to map the computed control actions (2 forces and 1 torque in the 2D case, 3 forces and 3 torques in the 3D case) from the cartesian space to the thrusters space.

$$\mathbf{f} \;=\; \mathbf{M}\mathbf{t} \tag{8.1.18}$$

where $\mathbf{f}$ is the vector of the 3 or 6 force and torque profiles for the control of the spacecraft, $\mathbf{t}$ is a vector of $n_t \; x \; 1$ (where n_t is the number of thrusters) with the continuous force and torque profiles of each thruster and $\mathbf{M}$ is the mapping matrix. In order to obtain the force profile for each thruster it is necessary to invert the Equation 8.1.18.

$$\mathbf{t} \;=\; \mathbf{M}^*\mathbf{f} \tag{8.1.19}$$

where $\mathbf{M}^*$ is the Moore-Penrose Pseudo-Inverse Matrix of the matrix $\mathbf{M}$. In the 2D case, given the thrusters distribution:

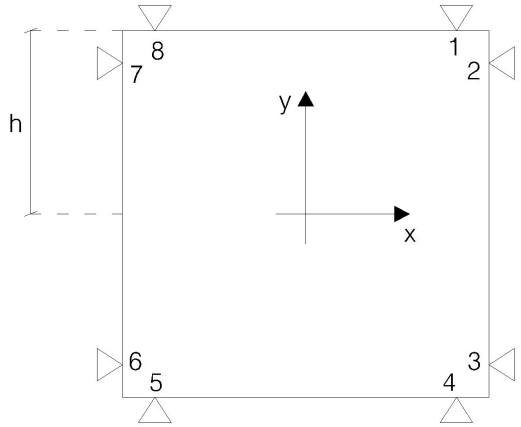


Figure 8.2: Thrusters distribution and mapping.

$$\begin{aligned} m_\theta &= ht_2 - ht_3 + ht_4 - ht_5 + ht_6 - ht_7 + ht_8 - ht_1 \\ f_x &= -t_2 - t_3 + t_6 + t_7 \\ f_y &= t_4 + t_5 - t_8 - t_1 \end{aligned} \tag{8.1.20}$$

Where h is the semi-lenght of the spacecraft side.

$$\begin{bmatrix} m_\theta \\ f_x \\ f_y \end{bmatrix} = \begin{bmatrix} -h & h & -h & h & -h & h & -h & h \\ 0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{bmatrix} \tag{8.1.21}$$

$$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{bmatrix} = \begin{bmatrix} -\frac{1}{8h} & 0 & -\frac{1}{4} \\ \frac{1}{8h} & -\frac{1}{4} & 0 \\ -\frac{1}{8h} & -\frac{1}{4} & 0 \\ \frac{1}{8h} & 0 & \frac{1}{4} \\ -\frac{1}{8h} & 0 & \frac{1}{4} \\ \frac{1}{8h} & \frac{1}{4} & 0 \\ -\frac{1}{8h} & \frac{1}{4} & 0 \\ \frac{1}{8h} & 0 & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} m_\theta \\ f_x \\ f_y \end{bmatrix} \tag{8.1.22}$$

This procedure permits to compute the thrust profile for each thruster.

As already explained the thrusters cannot give a continuous thrust profile: they can provide only a fixed amount of thrust or the null thrust. This condition forces to convert the continuous thrust profile in a series of impulses using a *Pulse Width Modulation Technique*. Using this technique it is possible to produce a control action that provides the system with the same force impulse of the continuous computed action. Defining the force impulse as:

$$\mathbf{I}(t) \;=\; \int_{t_0}^{t} \mathbf{F}\, dt \tag{8.1.23}$$

The PWM technique is base on the equality, on the sampling time:

$$\int_{t_0}^{t} \mathbf{F}\, dt \;=\; \mathbf{F}_{max} \Delta t_{PWM} \tag{8.1.24}$$

Manipulating this expression, it is clear that the Δt required to produce the necessary control action is:

$$\Delta t_{PWM} \;=\; \frac{\int_{t_0}^{t} \mathbf{F}\, dt}{\mathbf{F}_{max}} \tag{8.1.25}$$

The simulator implements only a PWM technique, that means that the sampling frequency of the thrusters controller is fixed.

8.2 Sensors

As shown in 3.2 the feedback loop is based on the measures of the joint variables:

- spacecraft translation and linear velocity
- spacecraft attitude and angular velocity
- manipulator joint position and angular velocity.

in order to compute the error on the reference input and to compute the dynamic matrices. While the values of these variables are available in the software simulation, in the real case sensors are required in order to measure the values needed for the feedback loop.

8.2.1 Spacecraft Attitude and Angular Velocity

KVH DSP-3000 Fiber Optic Gyro

The angular velocity of the spacecraft base is measured using a fiber optic gyro KVH DSP-3000. This sensor permit to obtain the measure of the angular velocity

of the base to which it is solidal, measuring the delay time between two optic signal traveling through the gyro itself. It is easy to compute the spacecraft attitude as:

$$\theta \;=\; \int_{t_0}^{t} \omega \; dt \;+\; \theta_0 \tag{8.2.26}$$

A Simulink block has been developed at the Naval Postgraduate School (in the NPS Toolbox) in order to interface the sensor with the Simulink Model. The Gyro communicates with the on-board pc using a Serial Port, through the communication protocol **rs232**, using a *baudrate* of 38400 $\frac{bits}{s}$

Metris iGPS Position Measurement

The position of the spacecraft in an inertial reference system is measured using an Indoor Pseudo GPS System. The System provides to the on board Pc the measures of the x and y coordinate of the barycenter of the spacecraft in the inertial reference system and communicates the measures to the on board pc using a Wire-less link. As explained above, the control system and the Newton Euler function for the computation of the dynamic matrices require the position of the spacecraft center of mass, but expressed in body coordinates. So, it is necessary to perform the rotation:

$$\begin{aligned} x_s \;&=\; x_m \cos\theta_m \;+\; y_m \sin\theta_m \\ y_s \;&=\; -\, x_m \sin\theta_m \;+\; y_m \cos\theta_m \end{aligned} \tag{8.2.27}$$

Also for the GPS system the NPS Toolbox provides the Simulink blocks needed in order to interface the sensor with the system model.

8.3 Control System Performances

In the following section the performances of the control system in the virtual environment are presented. Three cases are compared: the commanded variable, the controlled variables in the case of continuous forces and torques and the case of forces and torques modulated using the PWM technique. In the sections below the time hystory of the controlled joint variables is presented in the 2D and 3D cases. The absolute error with respect to the reference input is then presented. It was preferred to show the absolute error than the relative error in order to underline the real accuracy of the system in tracking the selected end effector trajectory.

8.3.1 2D Case - Continuous Force and PWM Control

In this section the simulation results of the control system for the planar case are provided both for the case of continuous spacecraft forces and torque and the

case of the application of the Pulse Width Modulation. The tracked trajectory is a circumference of radius equal to the distance from the spacecraft centeringof mass to the end effector, in the full deployed manipulator configuration. The parameters of the tested system are:

Spacecraft		
Length	l	.2 m
Height	h	.2 m
Width	w	.2 m
Mass	M	10 kg
Moment of Inertia along x	I_x	6.66 10^{-2} kgm^2
Moment of Inertia along y	I_y	6.66 10^{-2} kgm^2
Moment of Inertia along z	I_z	6.66 10^{-2} kgm^2
Manipulator		
DoFs	n	4
Length	l_m	0.2 m
Mass	m	0.4 kg
Moment of Inertia along x	I_{xm}	0 kgm^2
Moment of Inertia along y	I_{ym}	8.1 10^{-3} kgm^2
Moment of Inertia along z	I_{zm}	8.1 10^{-3} kgm^2

Table 8.1: Systm Configuration, 2D Case

The selected gains are:

$$K_P = \begin{bmatrix} 80 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \tag{8.3.28}$$

$$K_D = \begin{bmatrix} 2\sqrt{80} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2\sqrt{80} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2\sqrt{80} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\sqrt{100} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\sqrt{100} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} \end{bmatrix} \tag{8.3.29}$$

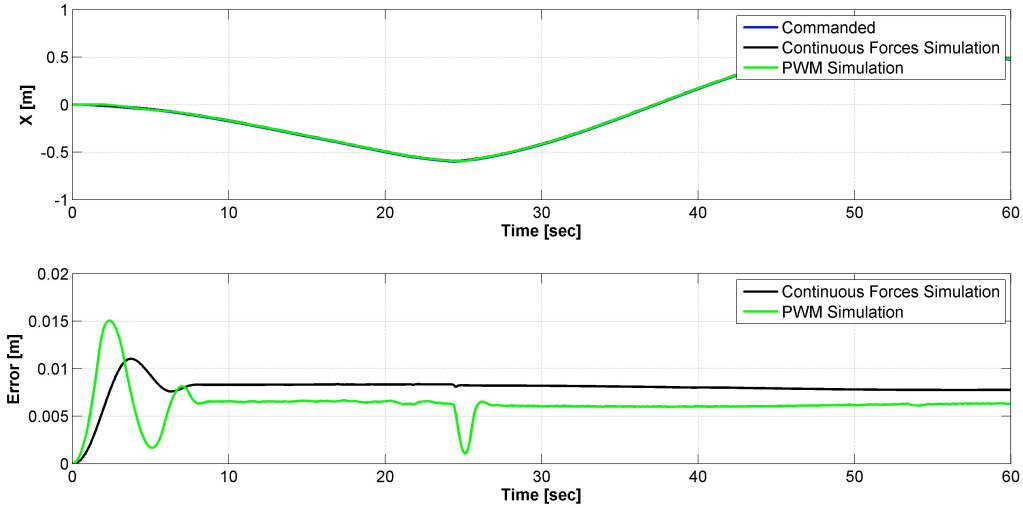


Figure 8.3: Position of the spacecraft along the x-direction.

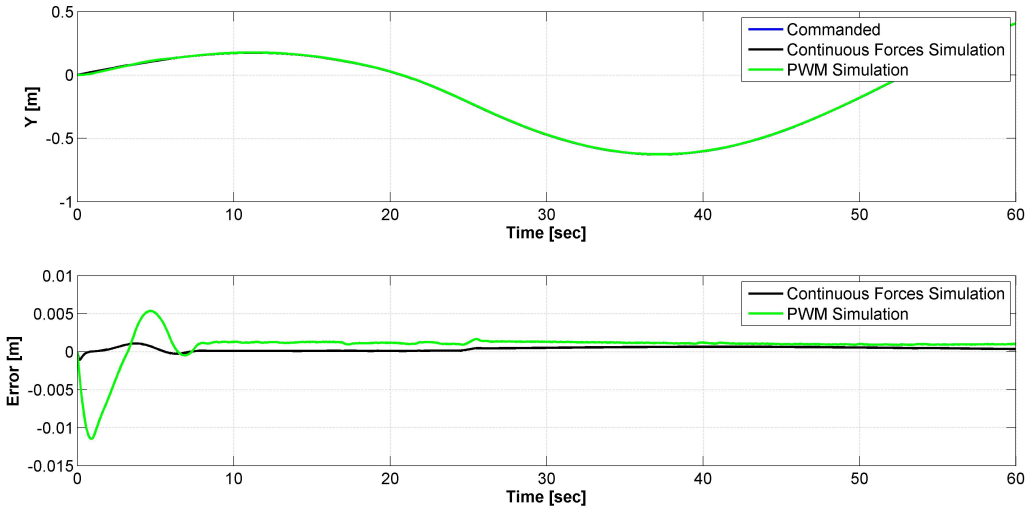


Figure 8.4: Position of the spacecraft along the y-direction.

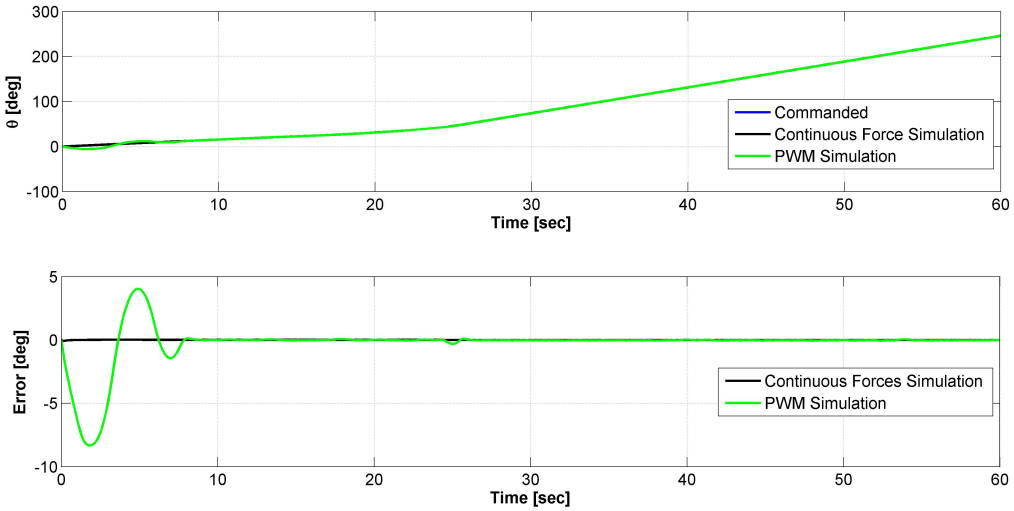


Figure 8.5: Spacecraft attitude.

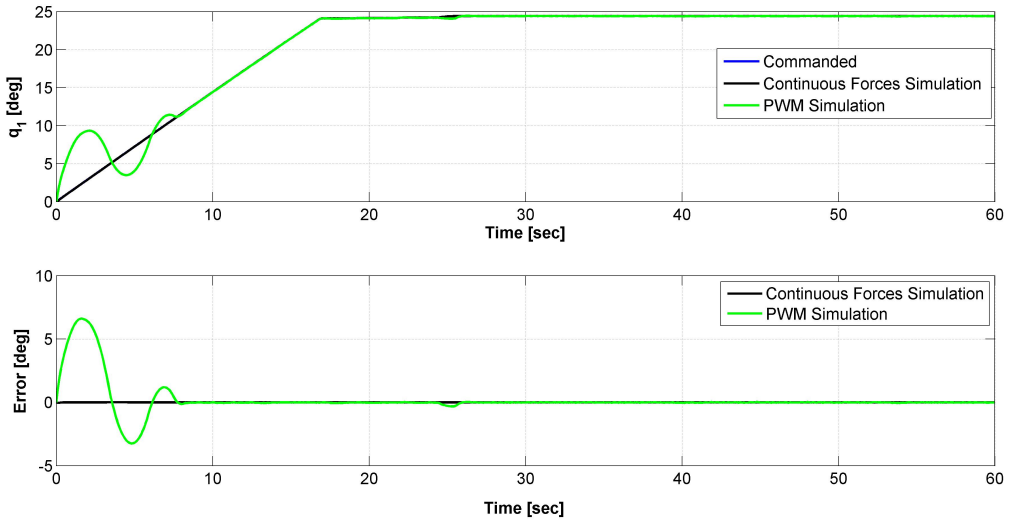


Figure 8.6: Joint 1 position.

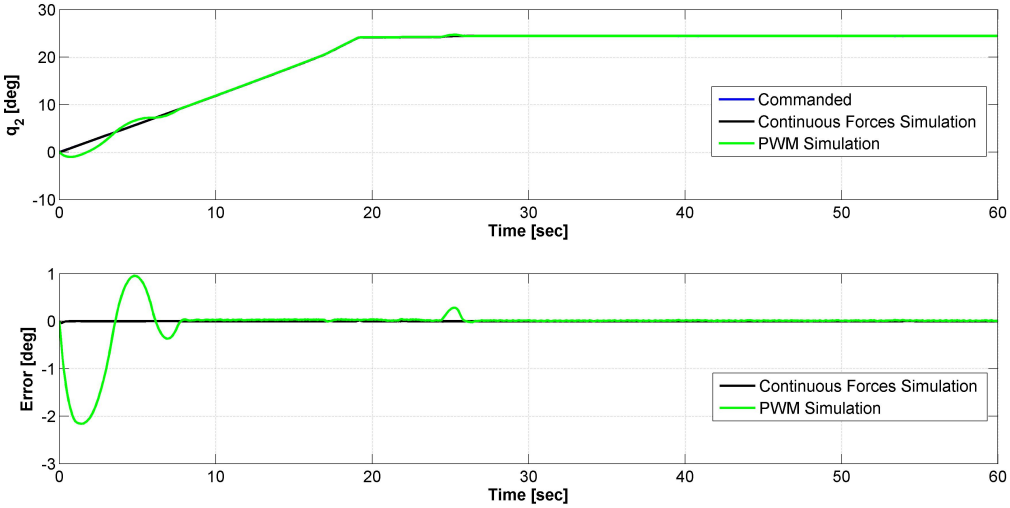


Figure 8.7: Joint 2 position.

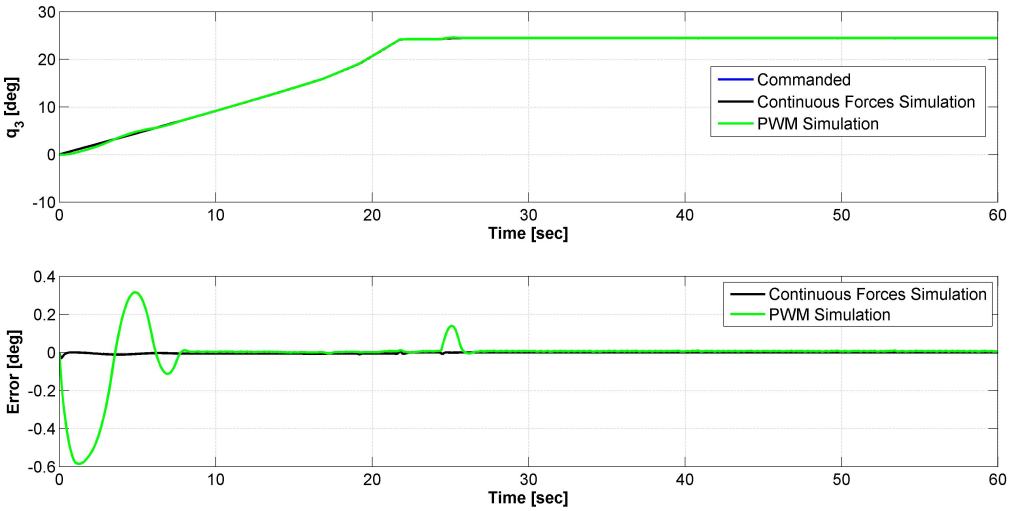


Figure 8.8: Joint 3 position.

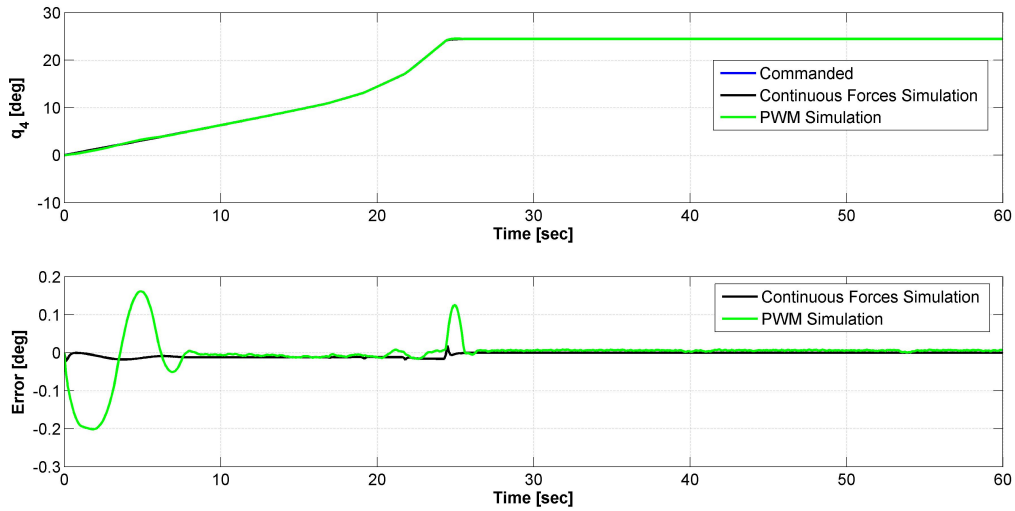


Figure 8.9: Joint 4 position.

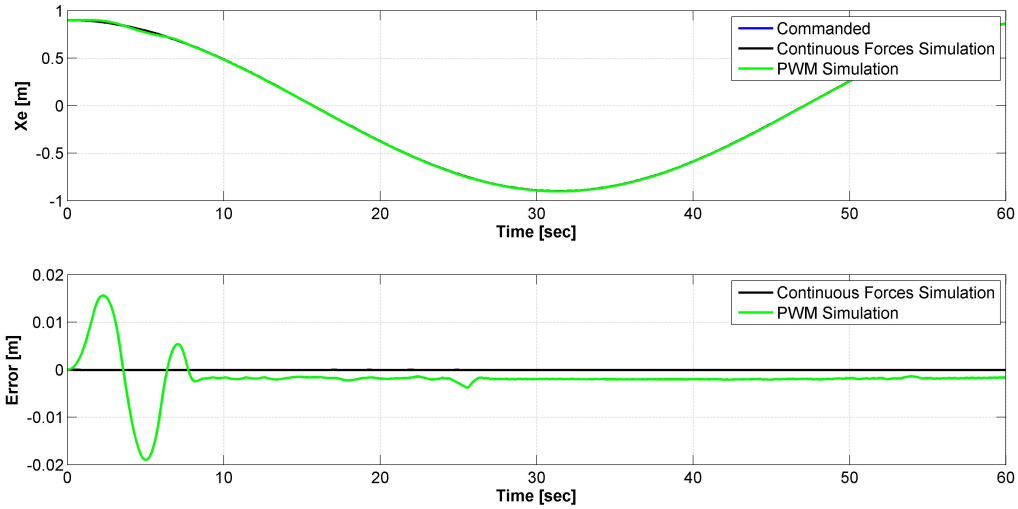


Figure 8.10: Position of the end effector along the x-direction.

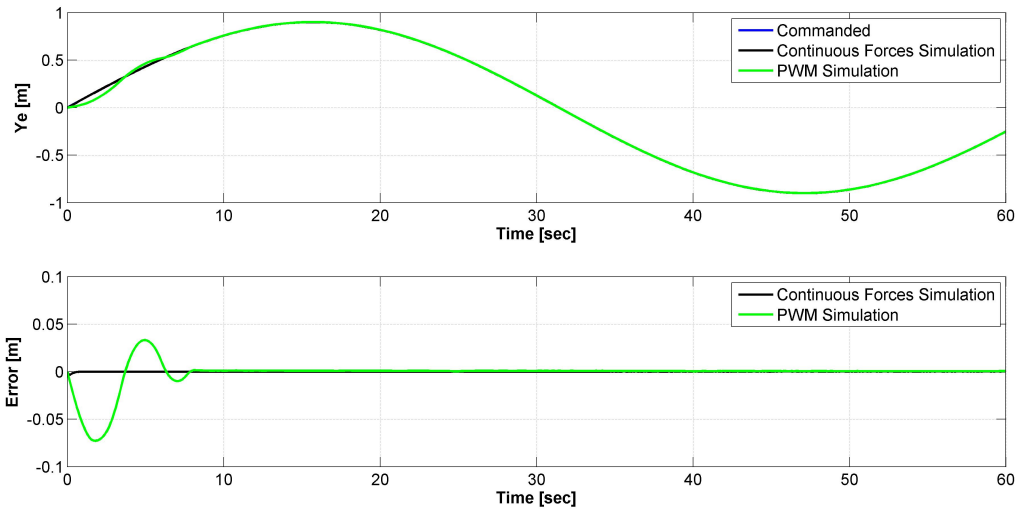


Figure 8.11: Position of the end effector along the y-direction.

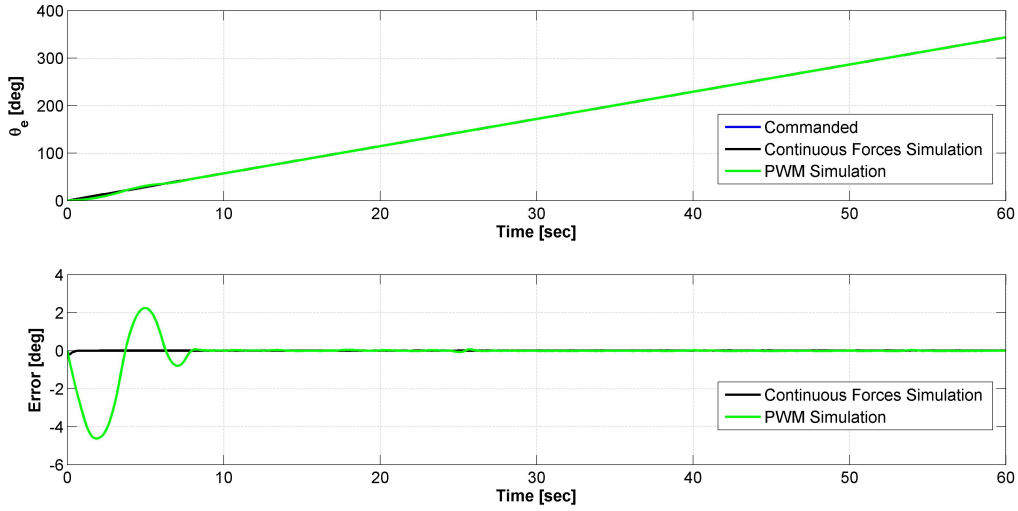


Figure 8.12: End-Effector orientation.

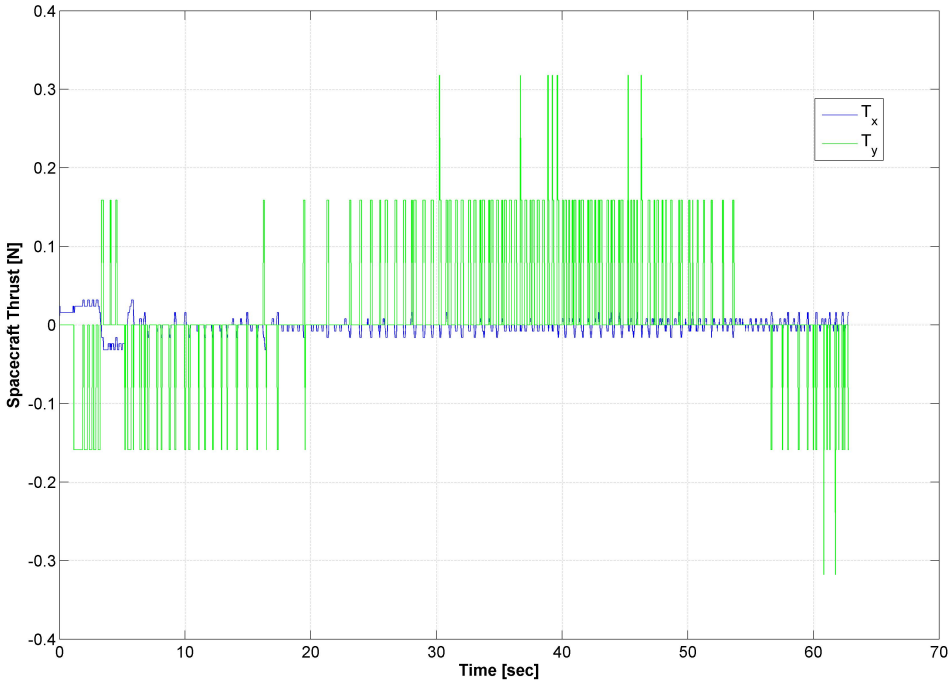


Figure 8.13: Scacecraft Thrust Forces.

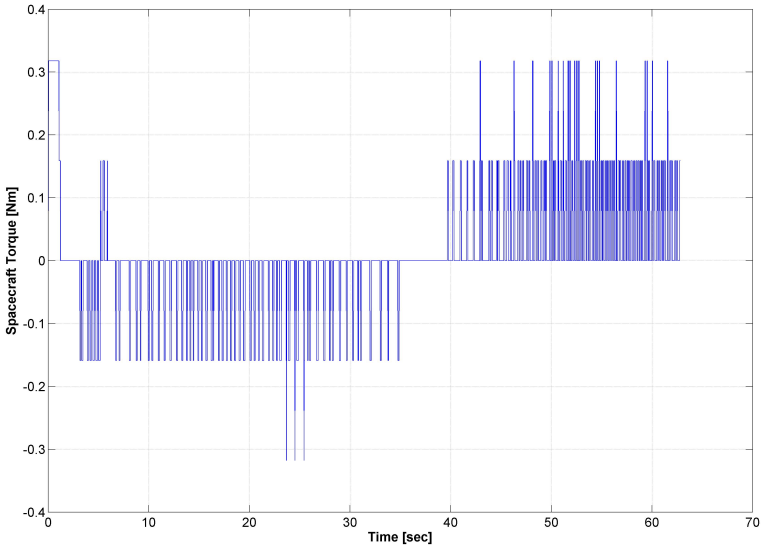


Figure 8.14: Scacecraft Torque.

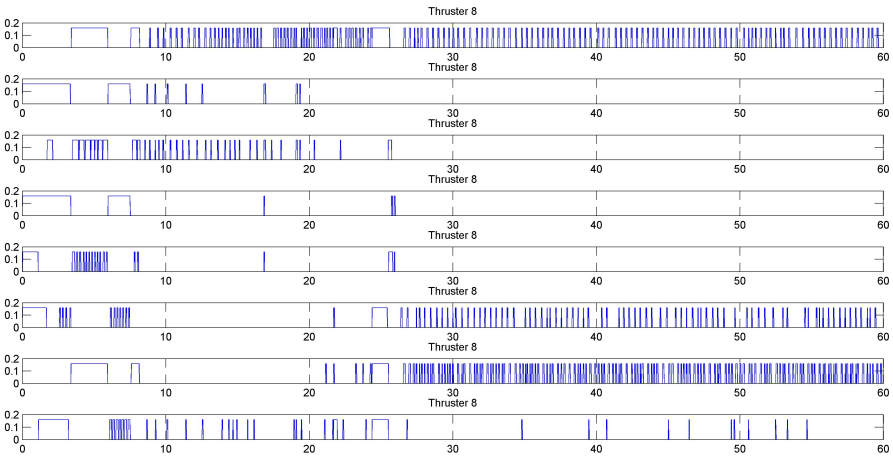


Figure 8.15: Thrusters impulse profile.

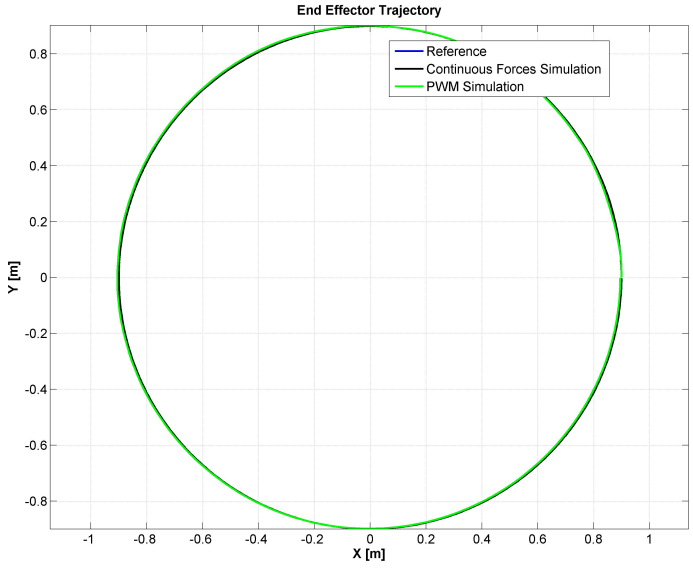


Figure 8.16: End Effector resulting trajectory.

8.3.2 3D Case: Full Computed Torque Control and Diagonal H Matrix Control

In this section the simulation results for the full 3D case are presented. In particular the results for two different control laws are presented: the case of full computation of the H matrix (that leads to a linear system closing the feedback loop) and the case of the control law generated using only the diagoal of the H matrix (obviously the Dynamics is computed using the full matrix). In this simulation the forces and torques acting on the spacecraft are continuous. The parameters of the tested system are:

Spacecraft		
Length	l	2 m
Height	h	2 m
Width	w	2 m
Mass	M	16 kg
Moment of Inertia along x	I_x	10.667 kgm^2
Moment of Inertia along y	I_y	10.667 kgm^2
Moment of Inertia along z	I_z	10.667 kgm^2
Manipulator		
DoFs	n	4
Length	l_m	0.5 m
Mass	m	1 kg
Moment of Inertia along x	I_{xm}	0 kgm^2
Moment of Inertia along y	I_{ym}	0.083 kgm^2
Moment of Inertia along z	I_{zm}	0.083 kgm^2
Joint Configuration		$y - z - y - y$

Table 8.2: System Configuration, 3D Case

The selected gains are:

$$K_P = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \tag{8.3.30}$$

$$K_D = \begin{bmatrix} 2\sqrt{20} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2\sqrt{20} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2\sqrt{20} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\sqrt{80} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\sqrt{80} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\sqrt{80} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2\sqrt{100} \end{bmatrix} \tag{8.3.31}$$

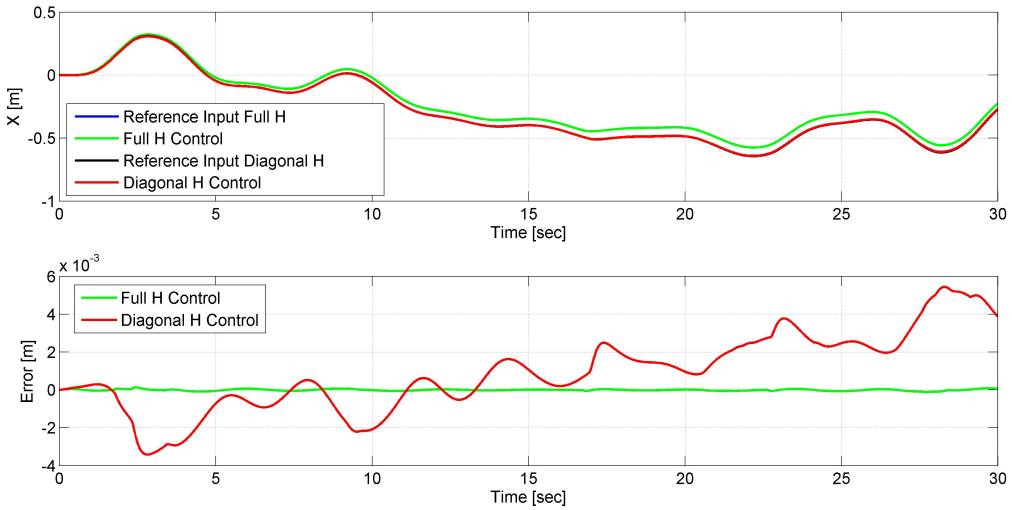


Figure 8.17: Position of the spacecraft along the x-direction.

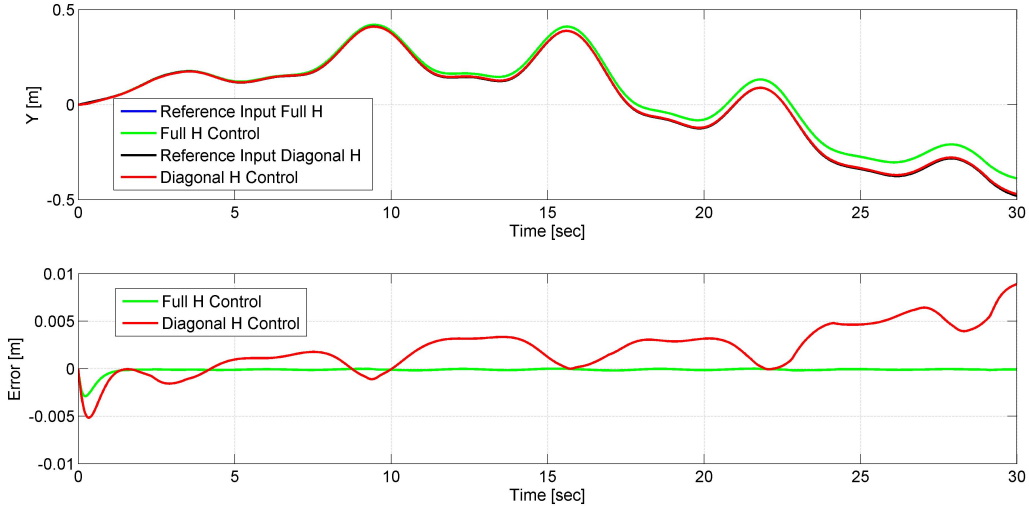


Figure 8.18: Position of the spacecraft along the y-direction.

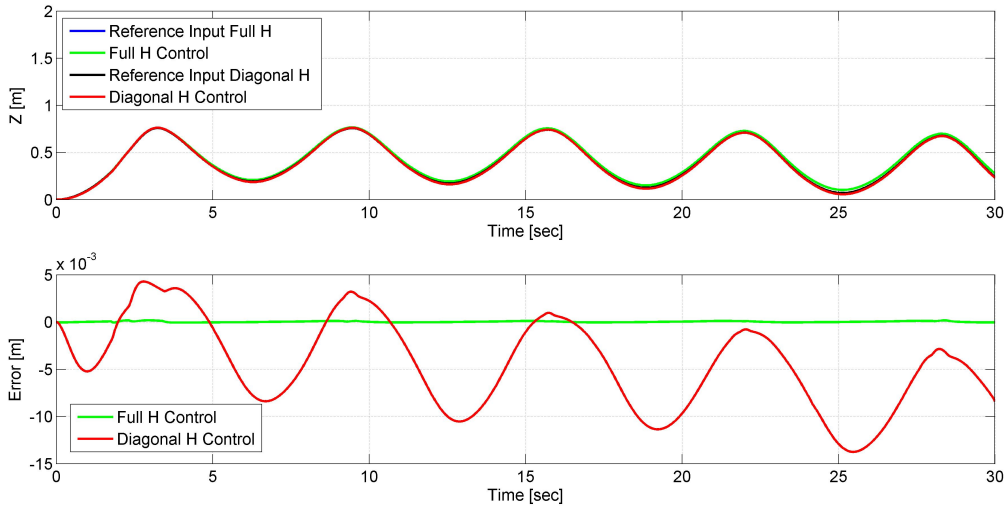


Figure 8.19: Position of the spacecraft along the z-direction.

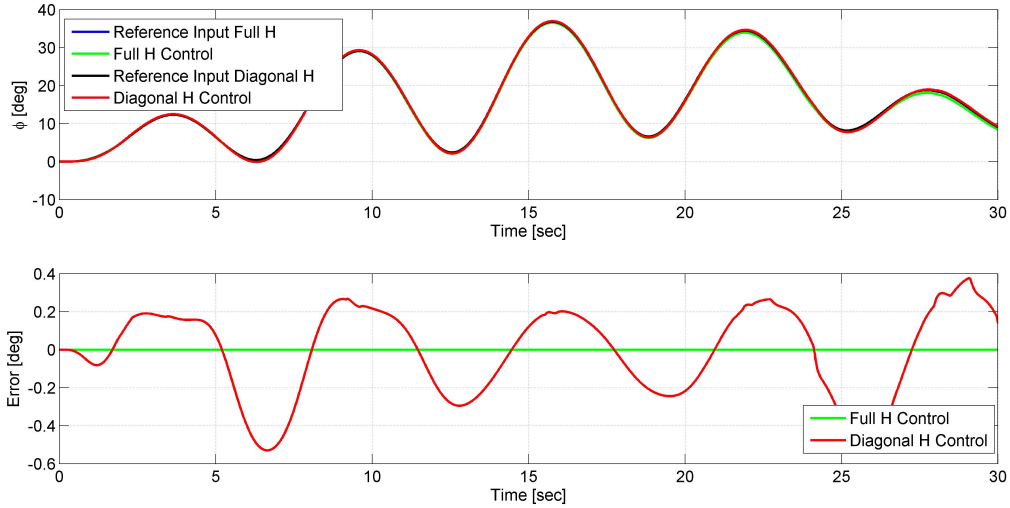


Figure 8.20: Spacecraft Euler Angle ϕ.

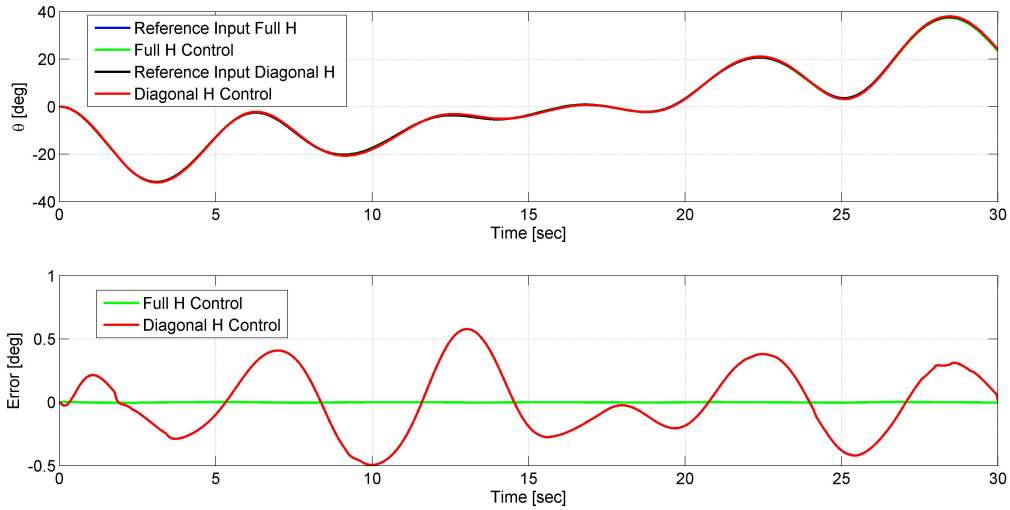


Figure 8.21: Spacecraft Euler Angle θ.

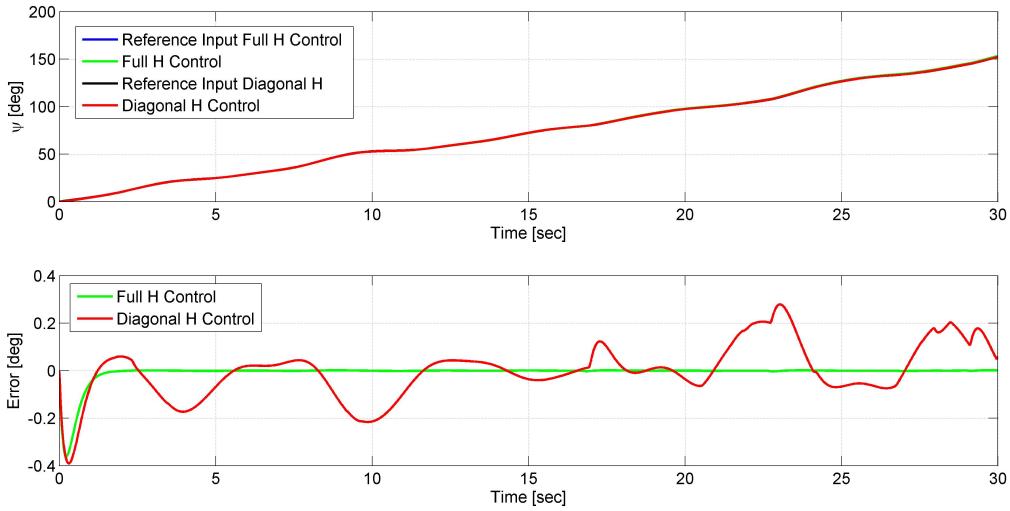


Figure 8.22: Spacecraft Euler Angle ψ.

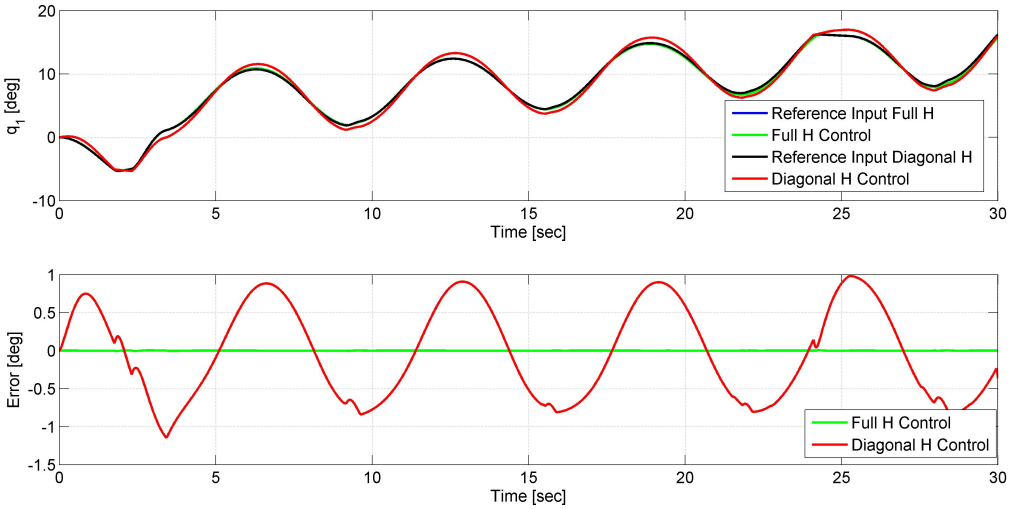


Figure 8.23: Joint 1 position.

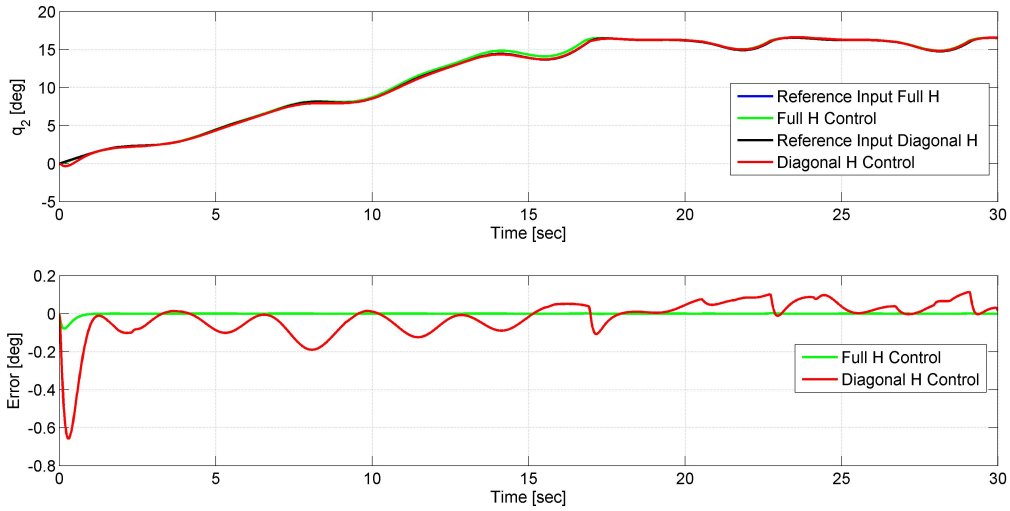


Figure 8.24: Joint 2 position.

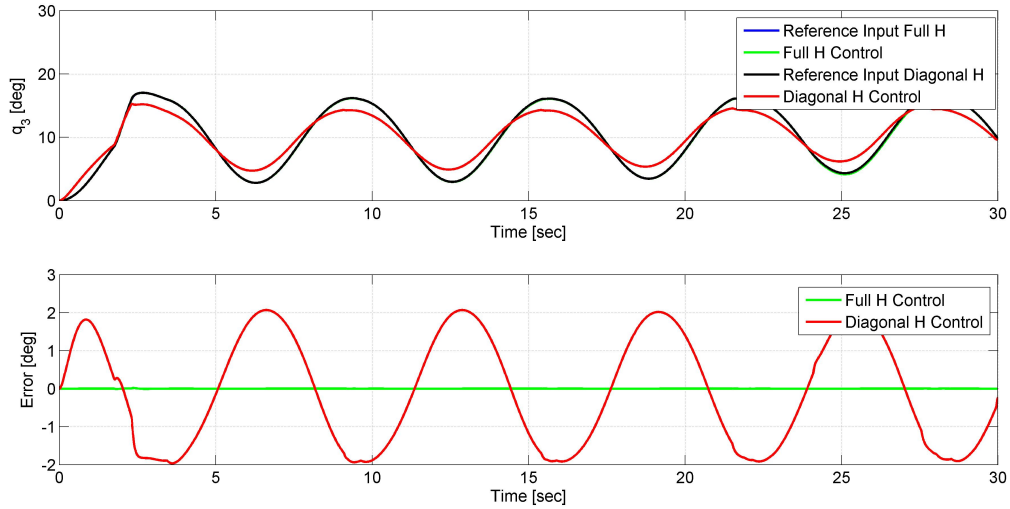


Figure 8.25: Joint 3 position.

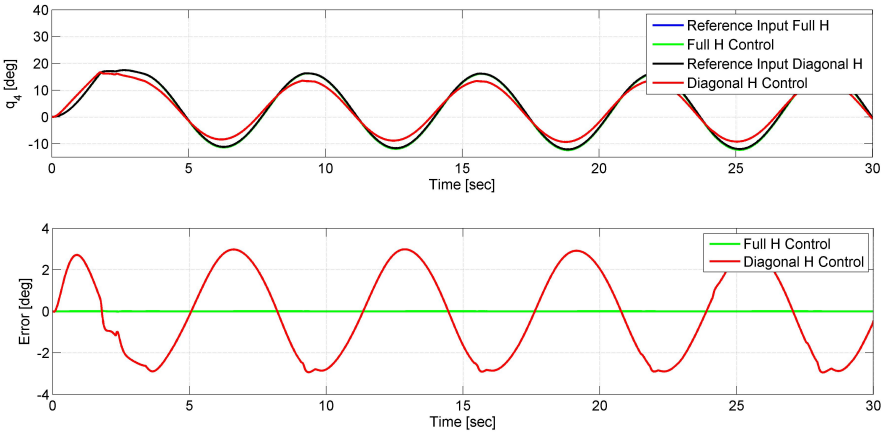


Figure 8.26: Joint 4 position.

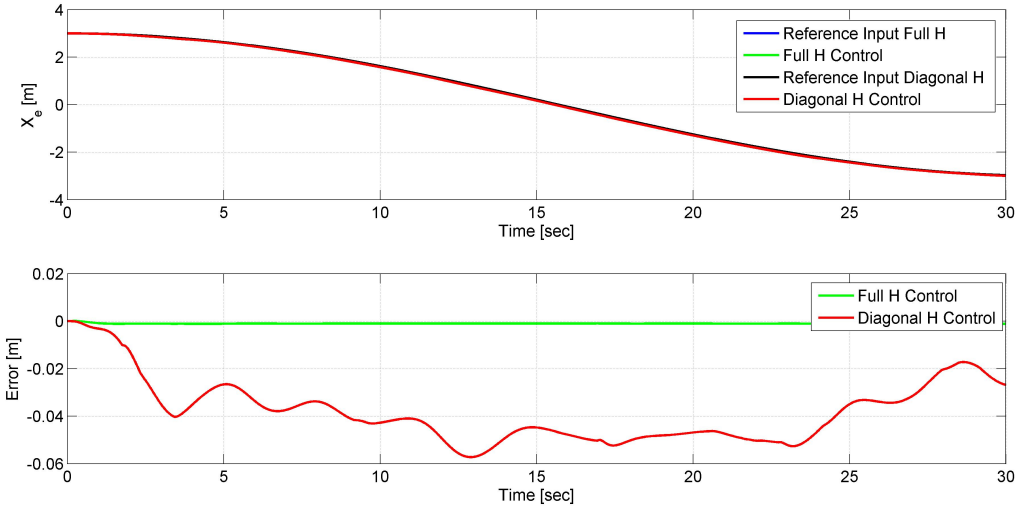


Figure 8.27: Position of the end effector along the x-direction.

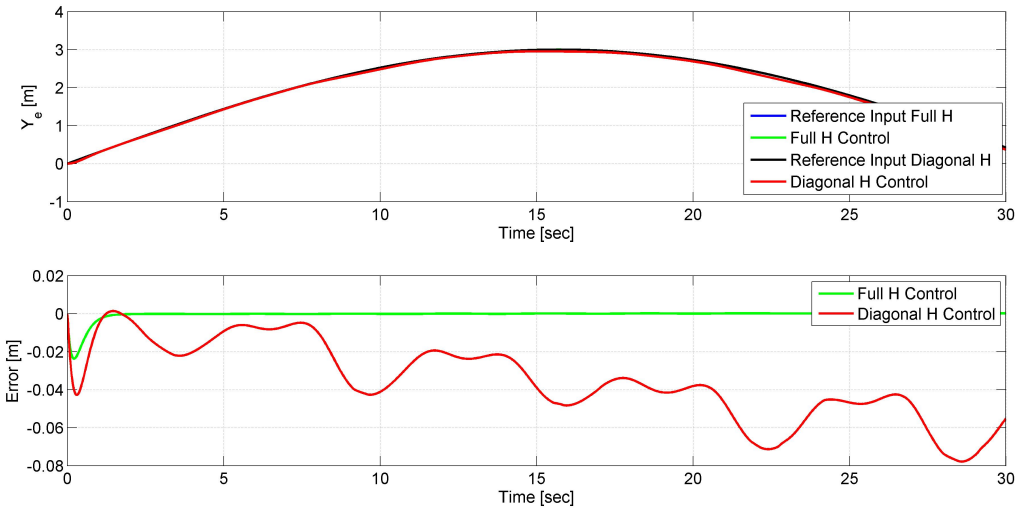


Figure 8.28: Position of the end effector along the y-direction.

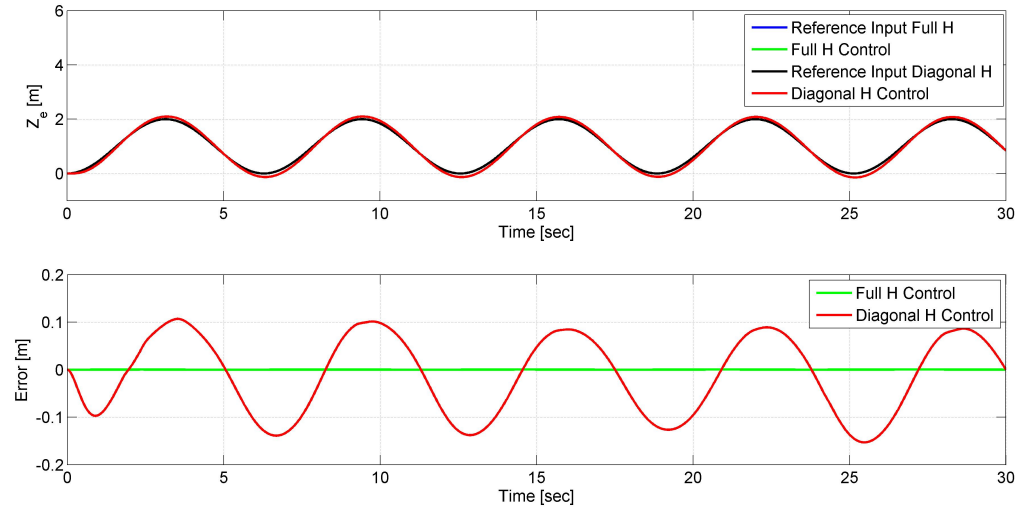


Figure 8.29: Position of the end effector along the z-direction.

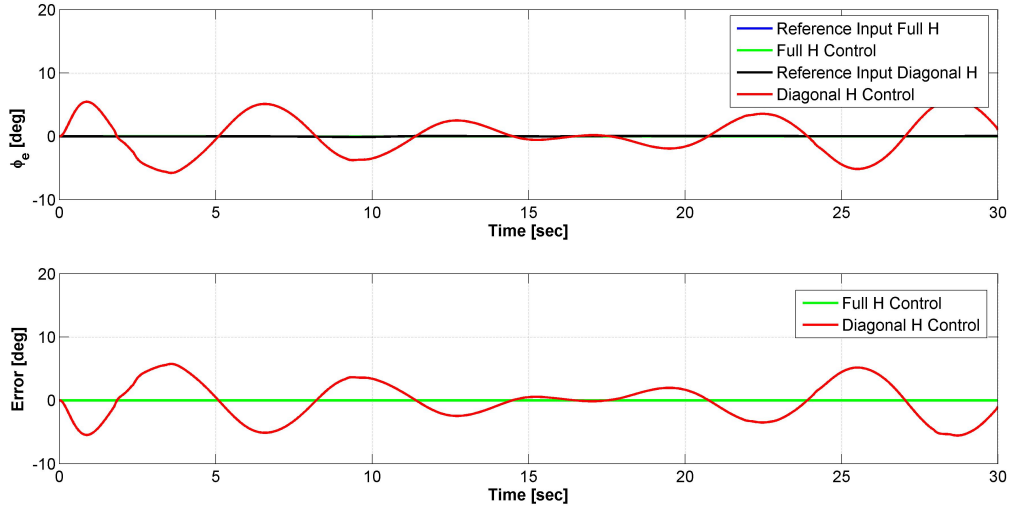


Figure 8.30: End Effector Euler Angle ϕ.

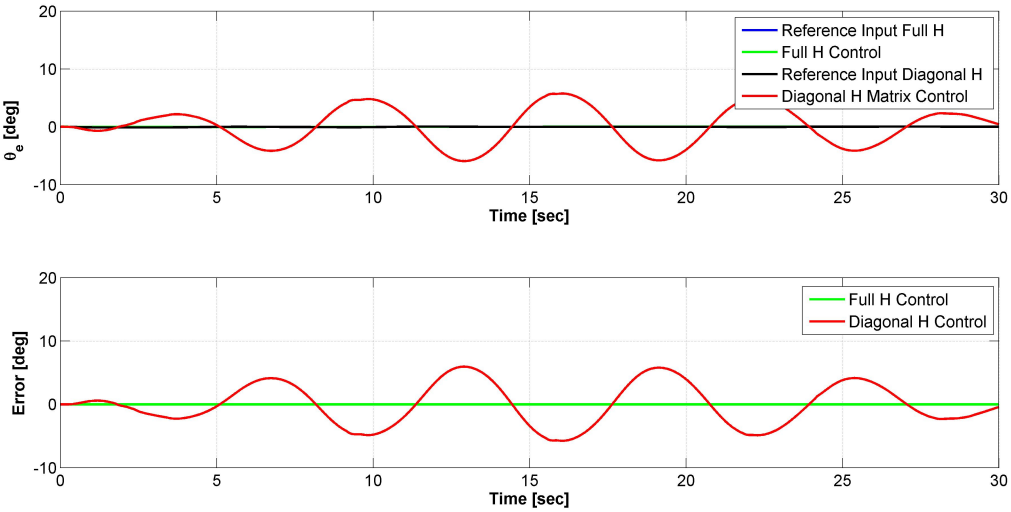


Figure 8.31: End Effector Euler Angle θ.

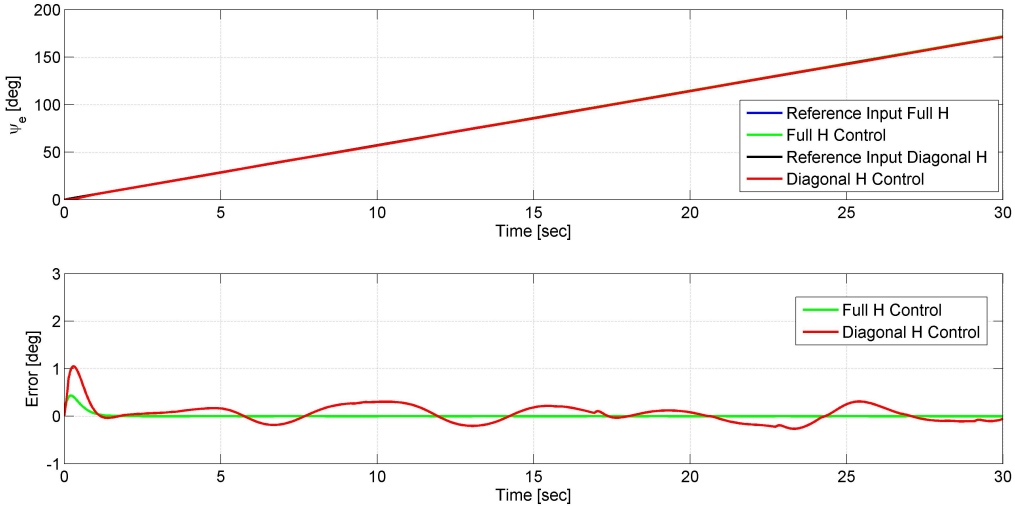


Figure 8.32: End Effector Euler Angle ψ.

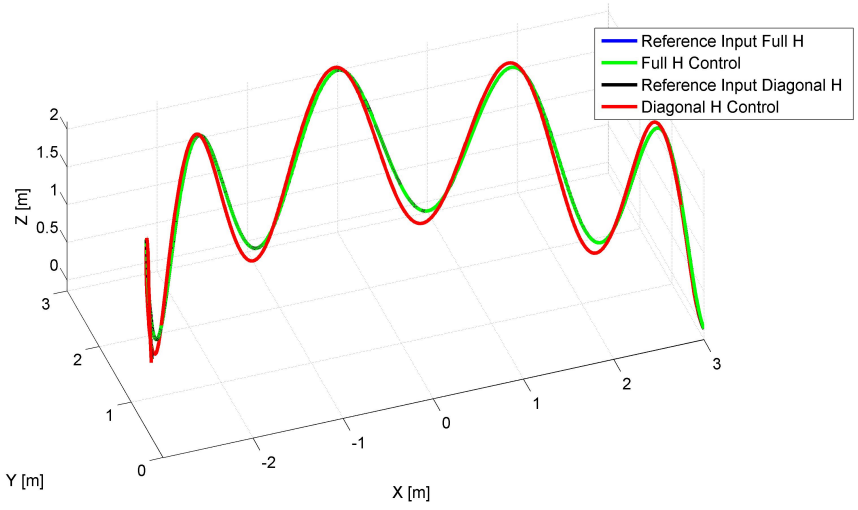


Figure 8.33: End Effector resulting trajectory.

8.3.3 Conclusions

The test cases present the results of the control of the system using the Computed Torque Control technique. Both the 2D and 3D cases have been simulated and analyzed.
As expected, the case of full computation of the inertia matrix and perfect feedback linearization, the control system assures optimal performances. In this case

the closed loop system is a linear system and the selection of the gains in accord with the condition in Equation 8.1.9 is the sufficient condition in order to provide asymptotic stability to the control.
Other more challenging conditions have been tested, in order to estimate the robustness of the control law. The Pulse Width Modulation of the thrusters has been tested in the 2D case in order to test the same conditions of the experimental testbed. As shown in Figure 8.10,8.11 and 8.12 also with the PWM control of the base the system remains asymptotically stable , assuring good performances in terms of error with respect to the reference input. The control in the case of PWM base control is affcted by a larger error in the first transitory phase (in Figures 8.6,8.7,8.8,8.9).Despite that the control system quickly arrives to a status of asymptotic stability. Figure 8.16 shows that the end-effector is tracking correctly the desired input trajectory.
The case of control using anly the diagonal of the **H** matrix reveales to be more complex to face: the error in the tracked trajectory is considerable and it is a consequence of the great amount of information neglected. On the other side Figure 8.33 shows that, despite the error, the system is reproducing the correct shape of the end effector input trajectory: if the mission phase requirements are enough low, this technique can be taken into account, especially if the reduction in the computational cost is considered a primary objective.
The analysis on computational cost revealed a linear dependance between the total compuatational time and the number of degrees of freedom of the manipulator (in addition to the 6 DOFs of the spacecraft).

Chapter 9

Virtual Reality Model

9.1 Operator Performances

The developed system allows the simulation of teleoperation techniques using a joystick. During a space mission the human operator is supposed to control the movements of the manipulator while the satellite is orbiting around the Earth. An artificial time delays can simulate the intrinsic commmunication delays experienced during such operations. As reported in (mettere reference) the communication delay can counts a few seconds. In this scenario it is almost impossible for a human operator to control a system without a real-time or quasi-real-time feedback from the system itself. Mettere gli studi visti in giro. A possible way to face time delays is to create a graphical predictor of the system. In this way the operator can see in real-time the simulated/predicted response of the system. The operator, in this way, is controlling the system basing his inputs only on the system's simulated response: the simulator model should be the more accurate as possible, in order to minimize the difference between the predicted operator control and the real applied control. This technique has been used in the operations of the ETS VII mission.

9.2 3D Graphical Model

In order to create a valid graphical representation of the system, a Virtual World has been created using the Matlab Virtual Reality Toolbox. The Virtual Reality Toolbox allows to interface a .WRL script to a Simulink block. In particular, given the .WRL script it allows the user to indicate the variables of the system: in this way each port of the corresponding Simulink block is linked to variables inside the virtual system. In the studied scenario the variables are the angular positions of the joints and the position and attitude of the spacecraft. The result of the graphical modeling is shown in the figures below for the 2D case (the environment is a representation of the Naval Postgraduate School Spacecraft Robotics

Laboratory, indeed the 2D version of the software will be used as a simulation for future laboratory tests):

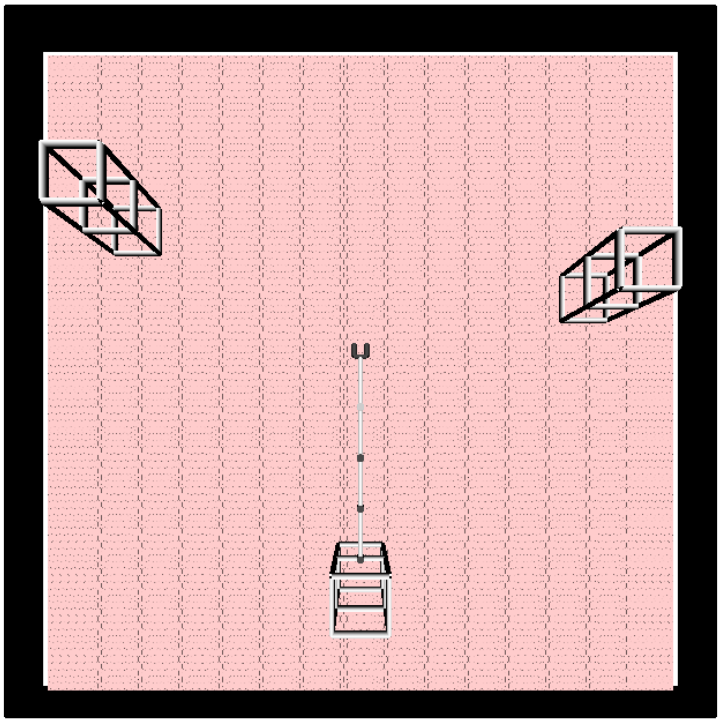


Figure 9.1: NPS SRL Virtual model.

The operator, using the joystick, has the full control of the end effector motion of the robot equipped with the robotic arm. In this scenario the robot can move inside the black fance, interacting with the environment (giving to the operator tactile/vibration feedbacks in case of contacts). The VRML (Virtual Reality Modeling Language) allows the creation of a full 3D model, with the possibility for the designer to add material, colour and shape features to the system. From this perspective the system can be highly improved in the future: modeling contacts and collisions between objects and the effects of the direct manipulation of objects could improve the effectiveness of the simulation itself. In the figure below the robot and the environment models are presented in detail:

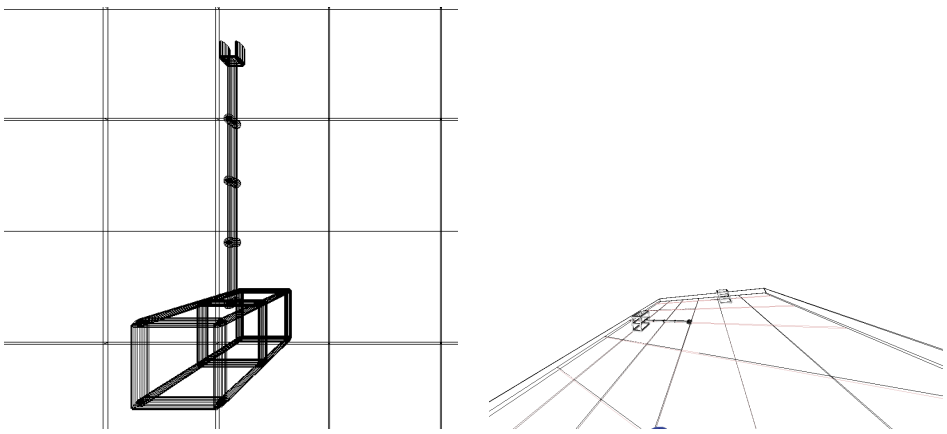


Figure 9.2: Virtual World.

In the virtual world used for the visualization of the results of the Pulse Width Modulation control technique of the spacecraft a feature related to the thrusters firing sequence has been implemented. During that visualization, each thruster is represented as a sphere fixed on the side of the robot itself. Exploiting the capabilities of the Virtual Reality Toolbox, these spheres can turn from black to red when the thruster they represent is supposed to fire. The figures 9.2 and 9.2 show two moments of a simulation:

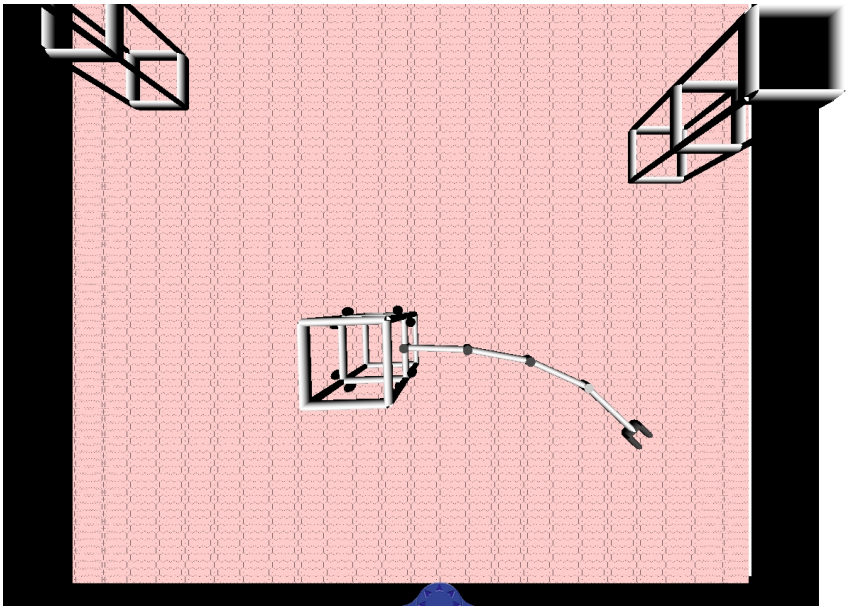


Figure 9.3: Thrusters off.

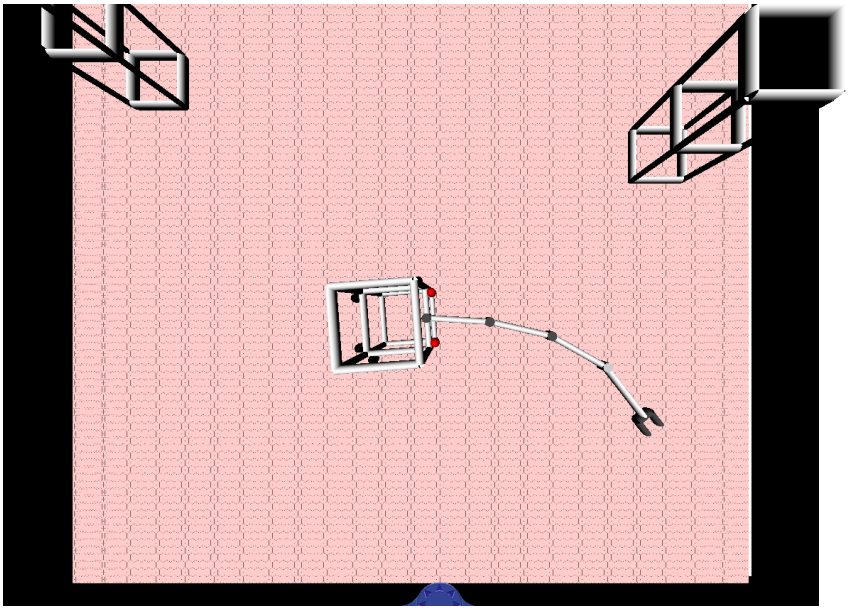
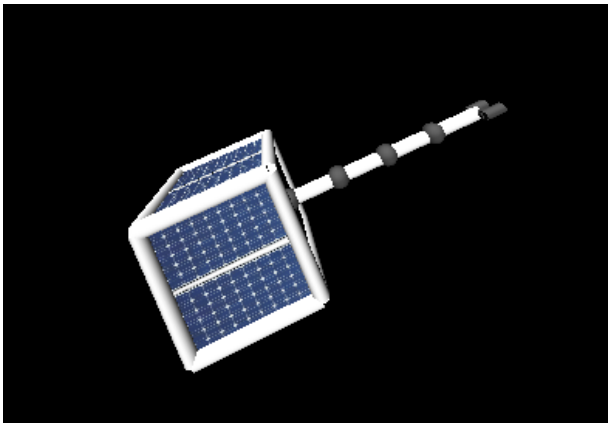
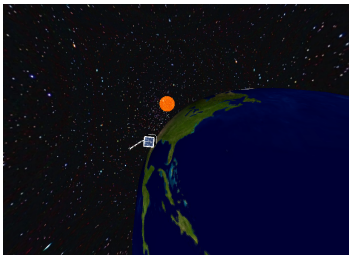


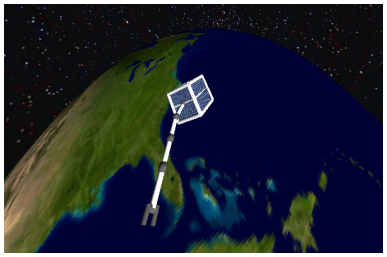
Figure 9.4: Thrusters firing.

Also for the full 3D case a Virtual World has been implemented. While for the 2D case, the computational effort required by the simulator is not extremely high, the operator can visualize in real time the controlled system, for the 3D case the required computational time does not allow the user to have a fluid real time animation of the system behavior. Because of that, the user decides the motion of the system using just the visual representation of the system's kinematics, that represents the reference input of the control system. The simulator can then perform the full-control simulation and show off line the evolution of the controlled system.
The virtual world for the 3D case is based on the same architecture of the previously explained .WRL file, but setted in order to take the inputs for the control of all the degrees of freedom of the spacecraft. The most useful configuration of the World is the one exposed in figure:



A more detailed and scenographic world has been implemented, with the visualization of some celestial bodies:





Chapter 10

Manipulator Realization

10.1 Project Requirements and Constraints

The Specification and constraints to the project can be summarized in the following table:

Weight	$\leq$2 kg
DoFs	$\geq$ 2
Length	$\geq$ 40 cm
Modularity	Yes
Electronics	Inside the arm if possible

In the design phase of such a system there are constraints that can not be written in a quantitative form. Some desirable characteristics of the system are:

- **Modularity**: the system has to be designed in order to easily increase or decrease the number of degree of freedom, in case of future needs.
- **Easy Assembly**: the design phase has to take into account the assembly phase
- **Wires Bulk**: the wires can have significant lengths and can represent a serious obstacle to the movement of the joints. They have to be accommodated inside the arm itself.

10.2 Components Selection

10.2.1 Servo Motors

The torques required to move the manipulator are in the order of the 10 mNm. On the other side, in order not to put limits to future applications in the field

of force control and objects manipulation, motors with higher torque limit have been selected. In particular it is important to underline that, since the motor are torque (current) controlled, the selection of an high torque motor does not affect the capability of the motor itself to provide low torques. The resolution in providing low torques is driven by the capability of the electronics to provide low level of current. The selected motors are Brushed DC Motors, with the following specifications:

Electric Motor		
Weight	W	114 g
Nominal Voltage	U_n	12 V
Output Power	$P_{2\,max}$	22.1 W
Friction Torque	M_R	1.7 mNm
Stall Torque	M_H	114.6 mNm
Torque Constant	k_M	18.57 mNm
Speed up to	$n_{e\,max}$	6000 rpm
Torque up to	$M_{e\,max}$	21 mNm
Current up to (thermal limits)	$I_{e\,max}$	1.34 A
Gear-head		
Gear ratio		16:1
Encoder		
Lines per revolution	N	500

Table 10.1: Servo Motors Data.

The motors affect in a strong way the final weight of the joint (the motor represent the of the total joint weight).

10.2.2 Drive Electronics

The drive electronics has to be the smallest as possible, in order to be accommodated inside the joint itself. At the same time it has obviously to provide the required current resolution in order to control the output current in the most accurate way. The selected driver is an All Motion EZSV10, with the following features:

Supply input	12 V to 40 V 1.5 A
Dimensions	24 mm x 35 mm x 15.24 mm
Control Modes	Position-Velocity-Torque
Encoder interface	Quadrature encoder, max frequency 4MHz
Electronics	Inside the arm if possible
Communication interface	RS232, RS485, USB

Table 10.2: Electronics Data

This driver model is equipped with an EEPROM memory, that can be used to store an initialization program. According to the driver model command set, the following lines are executed each time the driver is turned on:

$$\textbf{nb38400z0mxR} \tag{10.2.1}$$

where n is the address of the driver, $b38400$ sets the *baud rate* (bits per second) of the communication link, $z0$ initializes the encoder position and mx sets the maximum output current ($0 \leq m \leq 100$, where 100 is the current peak allowed by the driver). The driver communicates with the on board Pc using a serial port with the communication protocol **rs232**. An S-Function has been implemented in order to communicate with the drivers and the motors from Simulink. In this way it is very easy to generate a code for real time applications with the capability of commanding a group of servo motors.

10.3 Realization Process

In this chapter the manipulator design and assembly phase and the obtained geometrical, mass and inertial properties are explained in detail. The realization process can be divided into:

- **Specifications and project constraint analysis**
- **CAD Design of the Joint**
- **Joint Prototype realization and testing**
- **Realization of the complete manipulator**

10.3.1 CAD Design

The design of the Manipulator has been performed using the Software NX5, a classical Cad system of the SolidWorks family. The prototype has been realized in a polymeric material, using a Rapid Prototyping 3D printer.

Modularity Constraint

One of the biggest constraints in the design was represented by the need to have a reconfigurable manipulator. This means that the realized prototype has to allow the researchers to future expansions in terms of degrees of freedom of the system and orientation of the joints axes. The accomplishment of this task can be realized through a careful design of the revolute joints of the manipulator. The joint represents a great challenge from the design point of view: it has to contain the electric motor, all the mechanisms that allow the movements, the electronics to control the motors and the wires. These constraints increase the level of complexity of the system. The proposed design is presented in the Figures 10.1 and 10.2 :

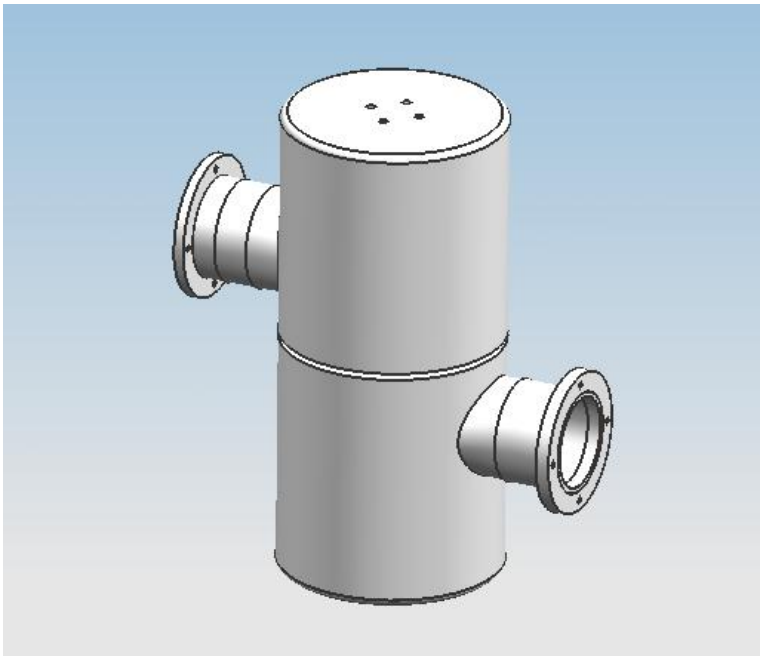


Figure 10.1: Proposed design for the joint realization

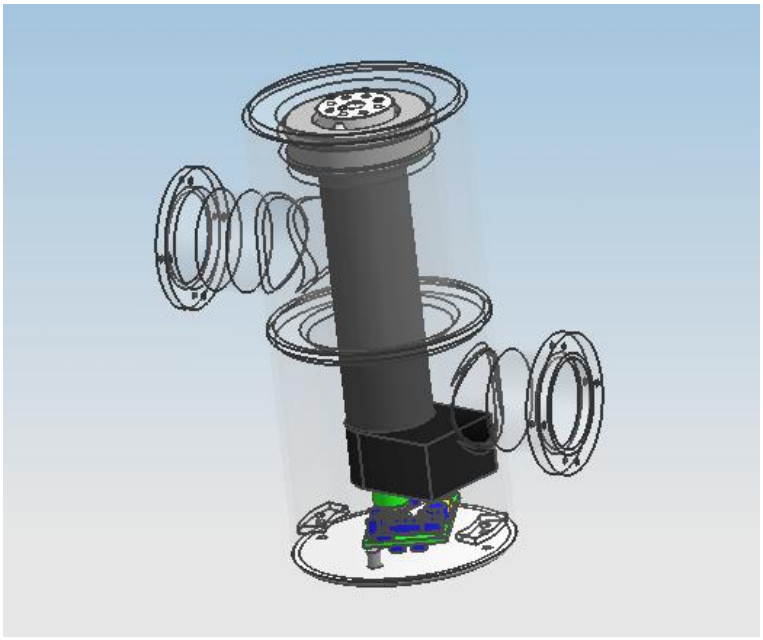


Figure 10.2: Joint internal disposition.

The system structure is composed by two parts: the bottom part of the joint, that accommodates the driver electronics end the electric motor, and the joint top part, that accommodates the mechanisms. The joint bottom part is presented in the Figures 10.3(a) and 10.3(b):

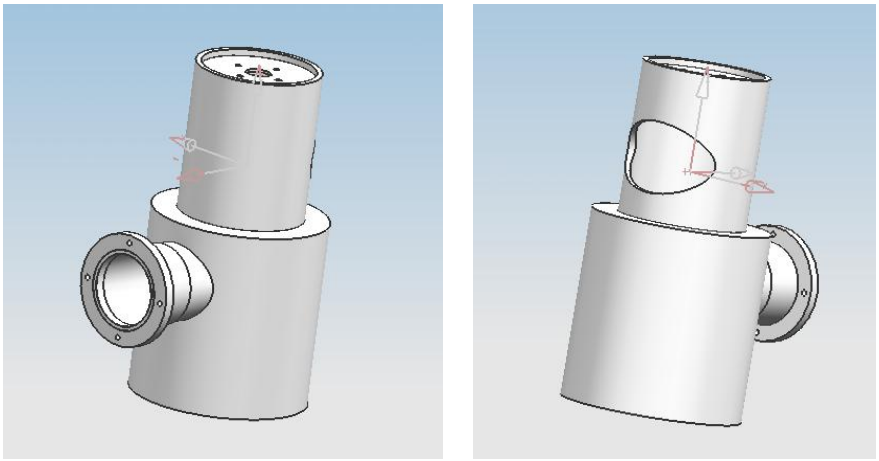


Figure 10.3: Joint bottom part front and rear.

The big opening on the rear side of the part is dedicated to the accommodations of the motor and driver wires.
The joint top part connects the bottom to the rest of the manipulator and accommodates a screw hub, in order to fix the motor shaft to the rest of the manipulator, and a bearing, to provide a larger, low friction, supporting area.

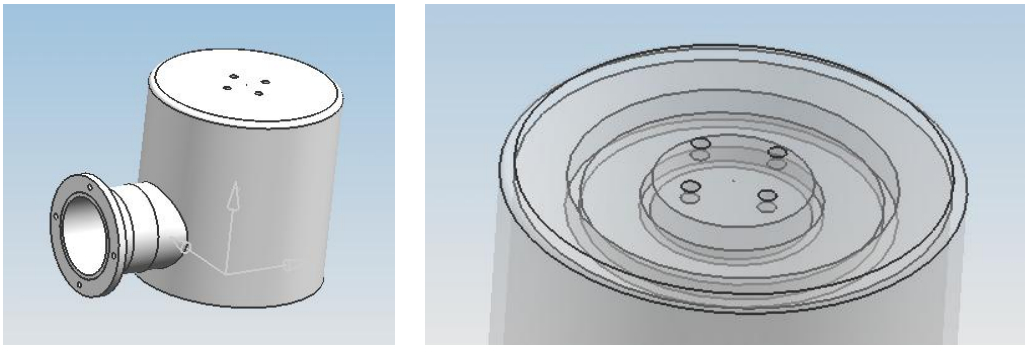


Figure 10.4: Joint top part.

The Figure 10.4(b) underlines the internal structure, designed to accommodate the superior side of the bearing and the screw hub. The exploded view of the joint helps understanding the assembly procedure and the accommodation of each component inside the structure.

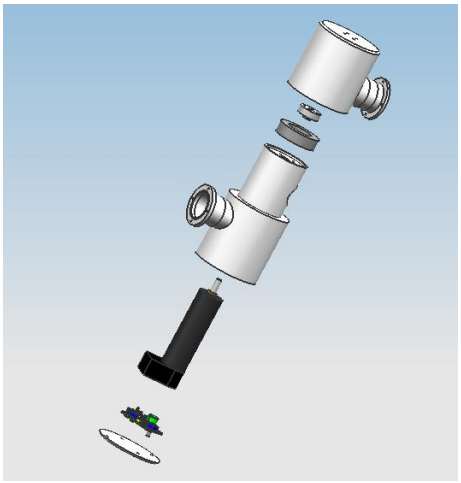


Figure 10.5: Joint exploded view.

Thank to the modular design the manipulator can be built simply connecting the top link with bottom links: this configuration allows a great simplicity in the assembly phase and, at the same time, the possibility to customize the joint orientation in the 3D space, enabling the assembly of a complete spatial manipulator. The Figure 10.6 presents the virtual model of the designed 4 DoF planar manipulator.

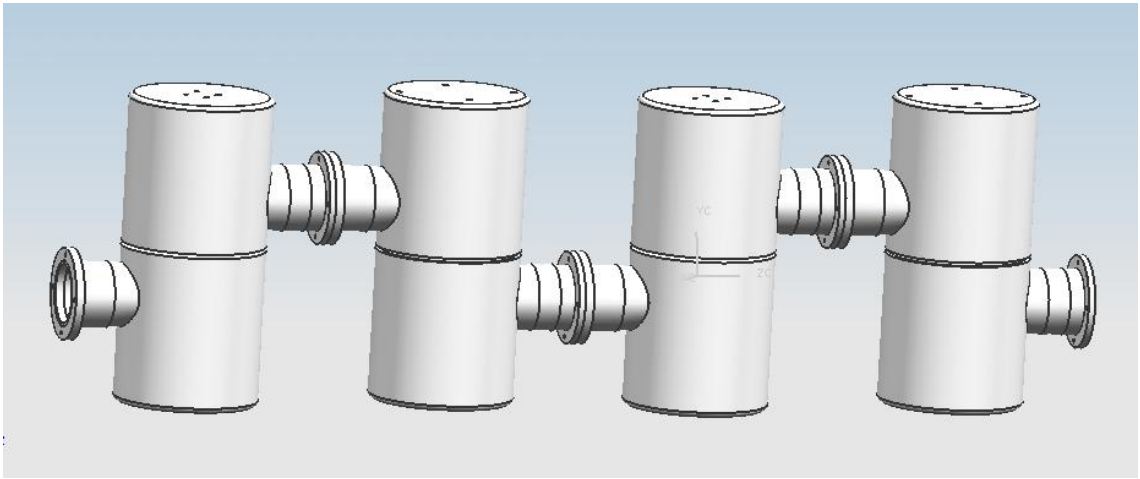


Figure 10.6: CAD view of the designed manipulator.

10.3.2 Joint Realization and Assembly

After the joint design phase, a joint prototype has been realized in order to test the validity of the design. The figure below presents the realized joint and each step necessary in order to mount it and the whole manipulator:

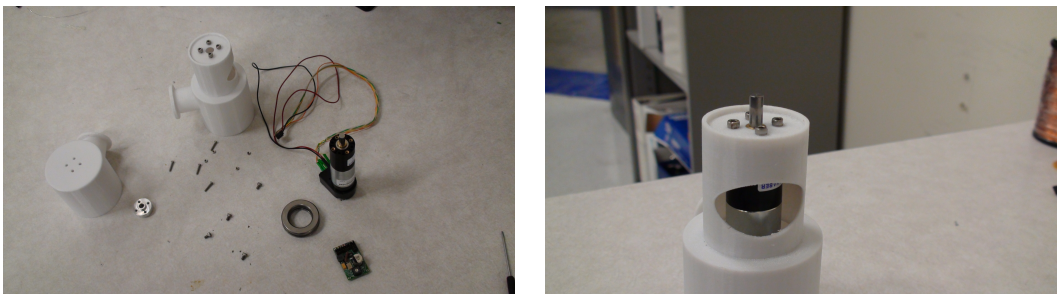


Figure 10.7: Components presentation and first assembly phase.

Figure 10.8: Second and third assembly phases.

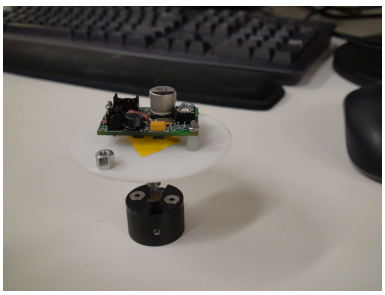


Figure 10.9: Electronics accomodation .

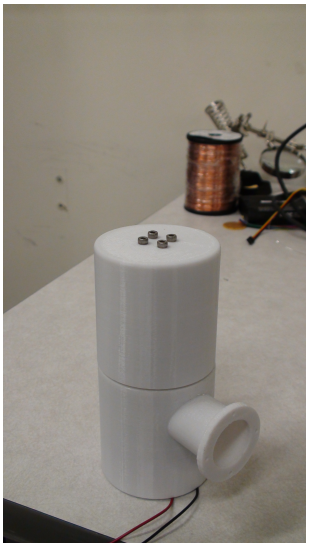


Figure 10.10: Complete Joint.

Coupling together the designed joints, changing the polarity of the joint itself each time, it is possible to create the entire manipulator. The figure below shows the designed manipulator, used in order to validate the theoretical and numerical results produced by the simulator.

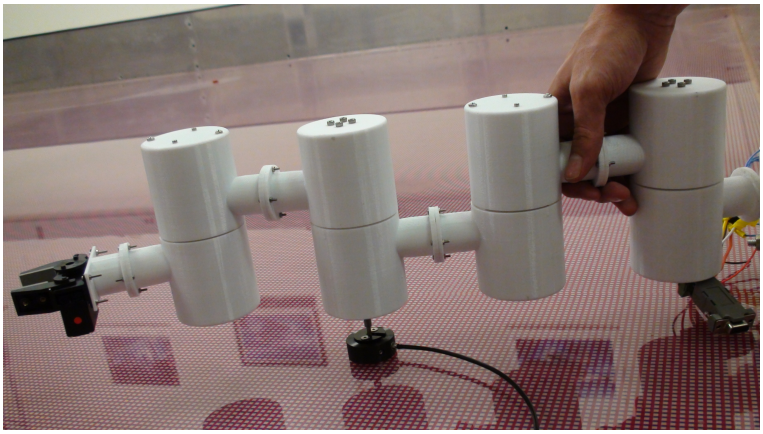


Figure 10.11: Final Design.

Final Manipulator Data	
Total Weight	1.6 kg (with 4 Joints)
Joint Weight	0.360 kg
Modularity	Yes
Length	52 cm (with 4 Joints)
Electronics	Inside the arm
Wires	Inside the arm

Table 10.3: Manipulator Data.

The final configuration of the free flying robot is presented in the Figure 10.12:

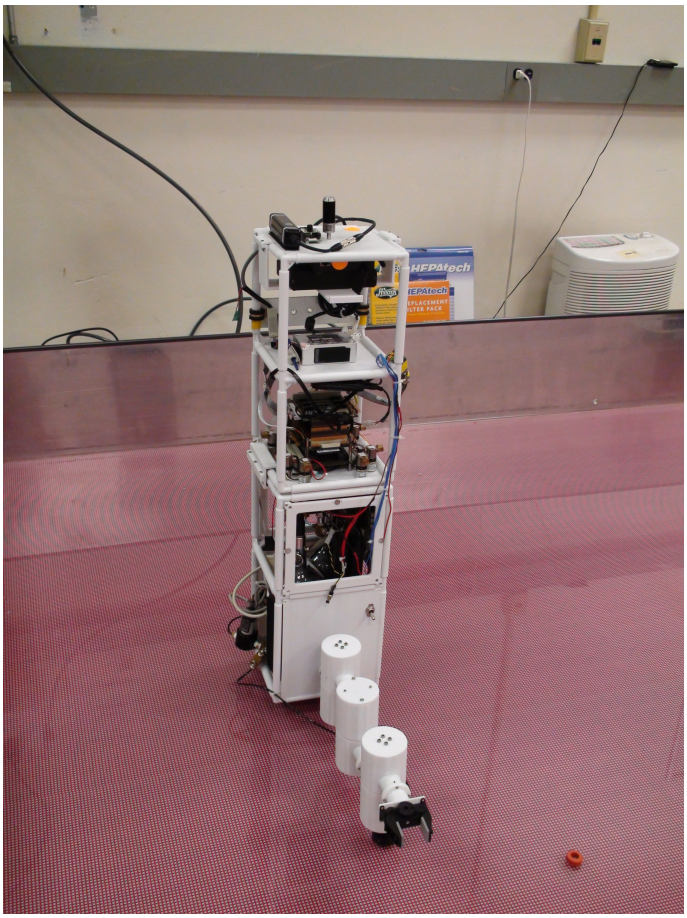


Figure 10.12: Final system configuration.

10.4 Preliminary Experimental Results: Spacecraft Control

In the following sections the preliminary results obtained in the control of the spacecraft, using the software derived from the simulator architecture, are presented. The software has been tested for the two most simple cases : rectilinear trajectory of the manipulator along the x and y direction of the inertial space. The control of the manipulator has been turned off. In the following Figures, as in the previous ones, the error on the controlled system is an absolute error, not a relative one. This beacuse in this phase of the tests the goal was to produce an evaluation of the real maximum error in the spacecraft positioning.

10.4.1 X-Trajectory

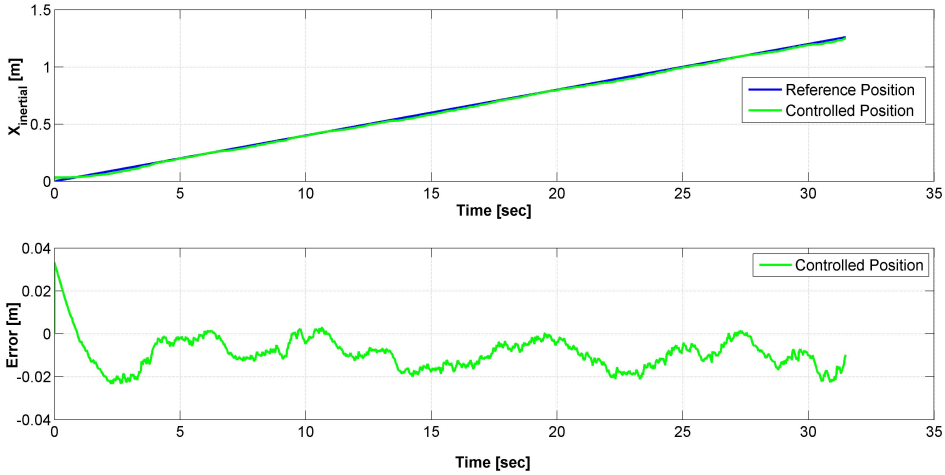


Figure 10.13: Motion in the x direction.

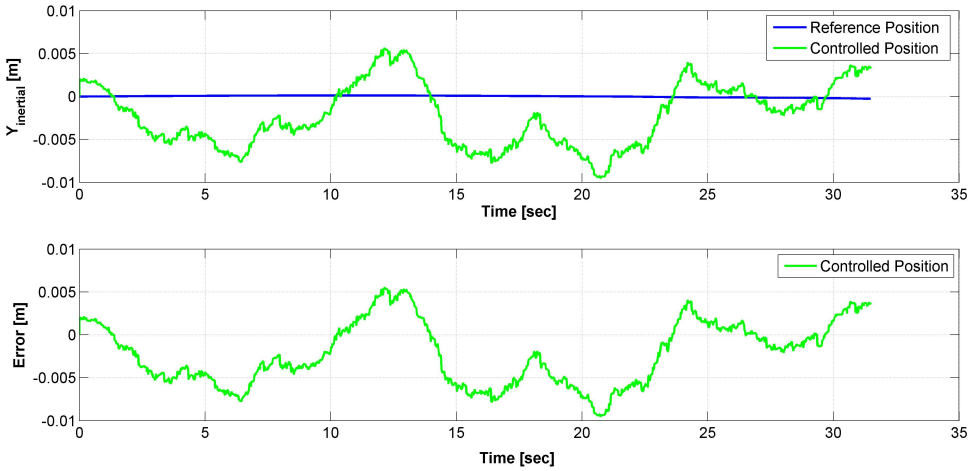


Figure 10.14: Motion in the y direction.

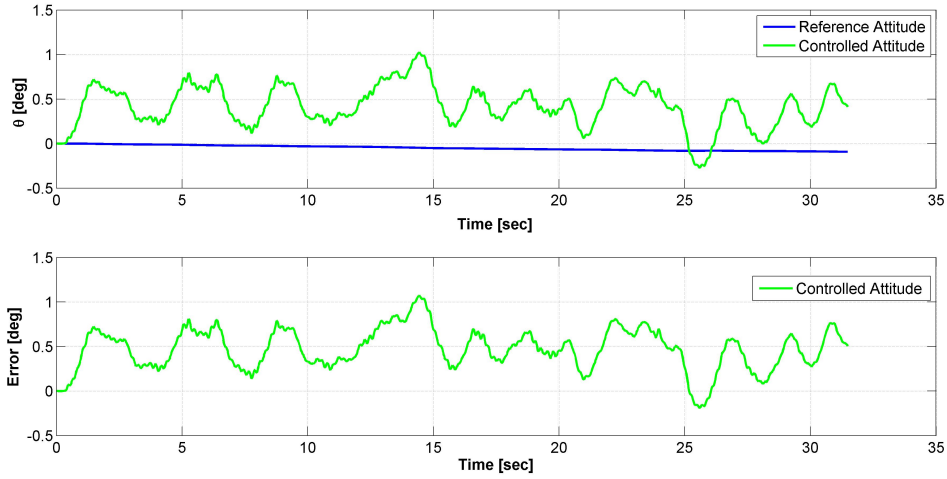


Figure 10.15: Attitude Control.

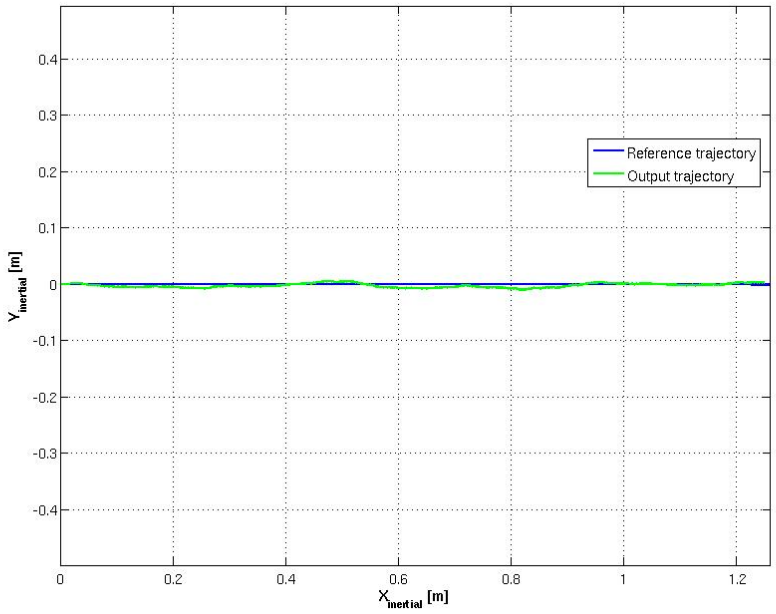


Figure 10.16: Followed trajectory.

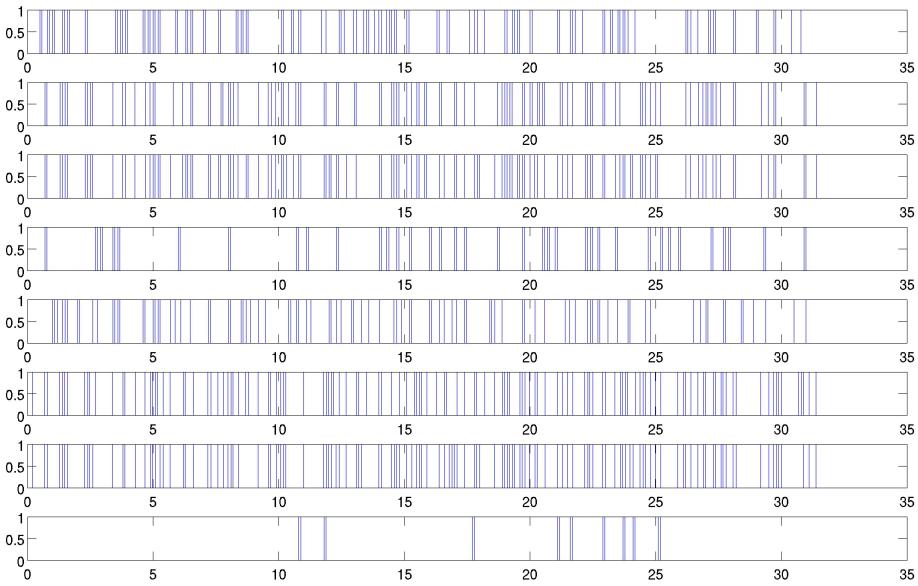


Figure 10.17: Thrusters command.

10.4.2 Y-Trajectory

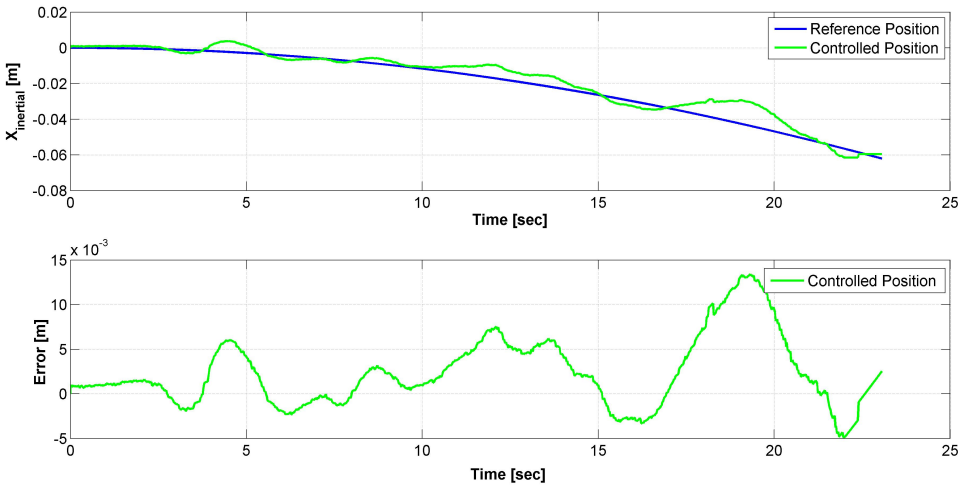


Figure 10.18: Motion in the x direction.

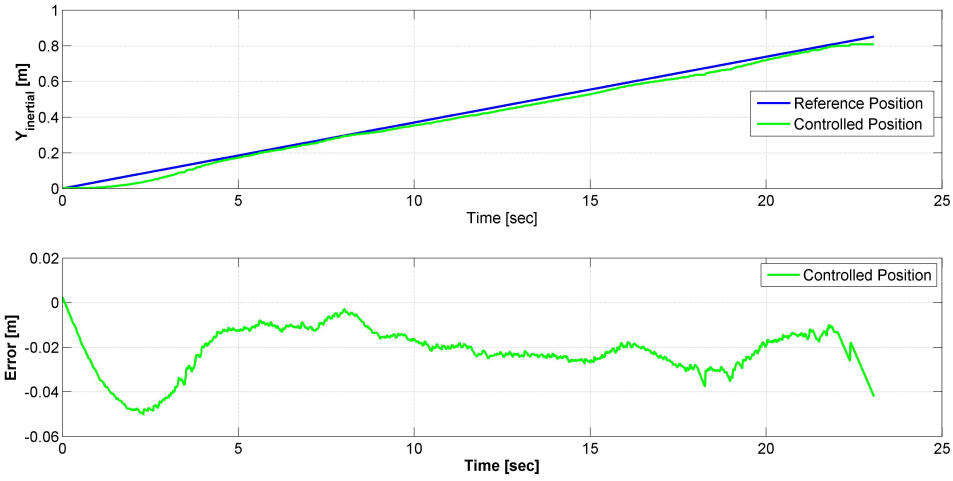


Figure 10.19: Motion in the y direction.

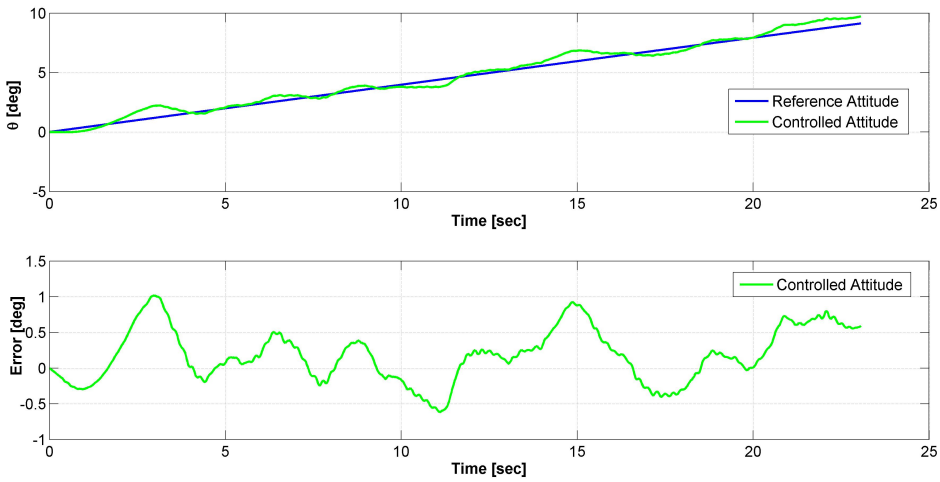


Figure 10.20: Attitude Control.

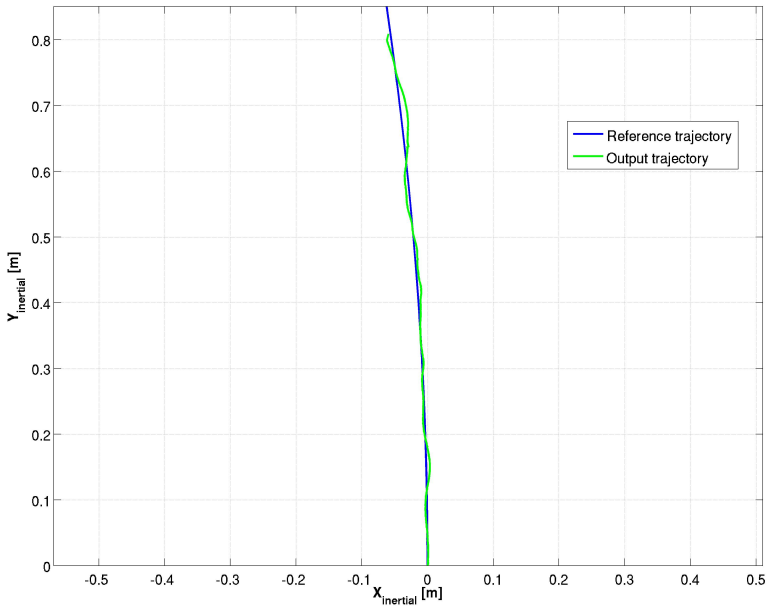


Figure 10.21: Followed trajectory.

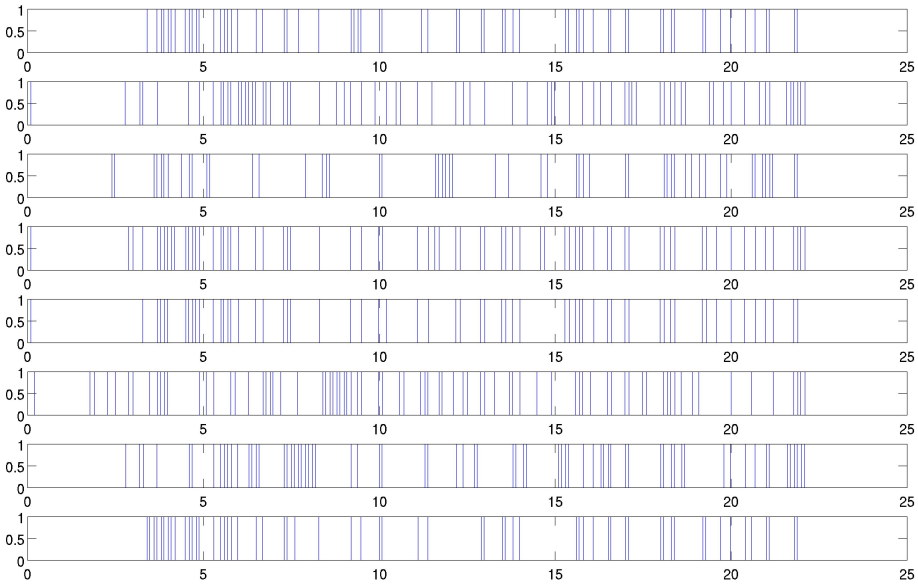


Figure 10.22: Thrusters command.

10.4.3 Conclusions

The control of the spacecraft is extremely accurate. As shown by Fig. 10.13 , 10.14, 10.18 and 10.19 , the spacecraft position tracking error is in the order of the centimeter, sometimes millimiters. These preliminary results confirm the validity of the approach in the system modeling. The tested case was the case of full control law, with full H matrix computation. The Spatial Algebra approach revealed to be extremely useful in order to allow the application of such an expensive control law to a real time system. The computed control law could, indeed, be challenging to implement for real time applications. The errors in the controlled spacecraft position are due to:

- Errors in system's mass parameters
- Errors in the process of derivation of spacecraft's position
- Floor imperfections and dust.

The errors in the attitude, as shown in Fig. 10.15 and 10.20, are in the order of the degree. The errors can be due to:

- Errors in system's computed inertial properties

- Numerical errors in the angular velocity integration process
- Floor imperfections and dust.

Chapter 11

Conclusions

11.0.4 Software Implementation

The thesis presented step by step the development of a software simulator created in order to test the kinematics, dynamics and control of a spacecraft mounted manipulator as a support for future experimental tests. Each sub-block of the system has been validated, using direct or indirect methods. The simulation tests provided in the previous chapters, prove the validity of the implemented approach in the dynamics modeling and the robustness of the implemented Computed Torque Control Law. Indeed, while in the case of perfect feedback we expected to obtain good results, the case of diagonal **H** matrix control law can be considered a strong test for the robustness of the system. The results provided in this case prove not only the capability of the control law to face non perfect linearization but, at the same time, it is an additional proof of the validity of the dynamic modeling. The errors found in this case are higher than the ideal case, but, at the same time, they can be considered acceptable for tests and operations where an extremely high accuracy is not the principal objective.
The Spatial Algebra approach, as already explained, proved to be a useful tool in order to build an extremely general multi body dynamics function. The use of this tool introduced in the system an high level of abstractness, but on the other side, it allowed to implement an algorithm that can be easily expanded to model more complex systems. The choice of modeling the system as a fixed base manipulator allows to use the analysis and control tools developed for fixed base manipulators : the implemented Joint Limit Avoidance kinematics algorithm, for example, has been implemented and validated in the case of fixed base manipulators.

The computational performances in 2D case can support the real time implementation of the software, while the full 3D case (with Joint Limit Avoidance Algorithm and full computation of the **H** matrix) cannot run in real time: as underlined before, the full 3D case brings a strong increase in the number of the DOFs of the system and in the number of operations to perform (in particular all the transformations and integrations of the Euler Angles that are not necessary

in the 2D case).

11.0.5 Future Developments

The software is now configured in order to model for both the kinematics and dynamics, linear chains of rigid bodies, connected together by revolute, prismatic, 3DOFs or 6DOFs joints. No external forces and torques are applied to the system. The implementation can be developed into three main directions:

- Kinematics and Dynamics Modeling
- Integration in the system of a module of Orbital Mechanics
- Control Techniques and Artificial Intelligence

The first possibility consists in a further development of the kinematics and dynamics modeling algorithms in order to study multiple manipulators systems and systems characterized by closed kinematic loops. Both the Direct Path Method for kinematics and the Newton Euler Algorithm for dynamics can be generalized to more complex and general systems. Also the joint models library can be expanded with different new joints and constraints models. The second development direction concerns the implementation inside the software of an Orbital Mechanics Simulator. The implementation of the Newton Euler Algorithm allows the introduction of external forces on each body of the system. This feature can be easily exploited in order to introduce in the system the effects of the orbital motion: the system can be tested under the disturbances produced by the Gravity Gradient Torque, Magnetic Torque, Solar Pressure Torque and Atmospheric Drag Torque. The addition of this feature will permit to simulate more realistic and complex operations. In this scenario a further improvement could be the modeling of the contacts between the end effector and other bodies: this could allow to study the system from the perspective of the manipulation capabilities of the robotic arm. The architecture of the system can also be expanded in the field of control and artificial intelligence. Now the system takes the input trajectory directly from an external operator; a possible improvement for the system can be the introduction of on board autonomy.

11.1 Experimental Tests

As shown in chapter 10 the manipulator has been completely realized and the S-Function for the communication from the on board PC to the drivers has been written and validated. As explained, the design of the revolute joint allows the implementation of a modular manipulator : the only limitation to the number of joints is related to the delay in the electronics. From the mechanical point of view

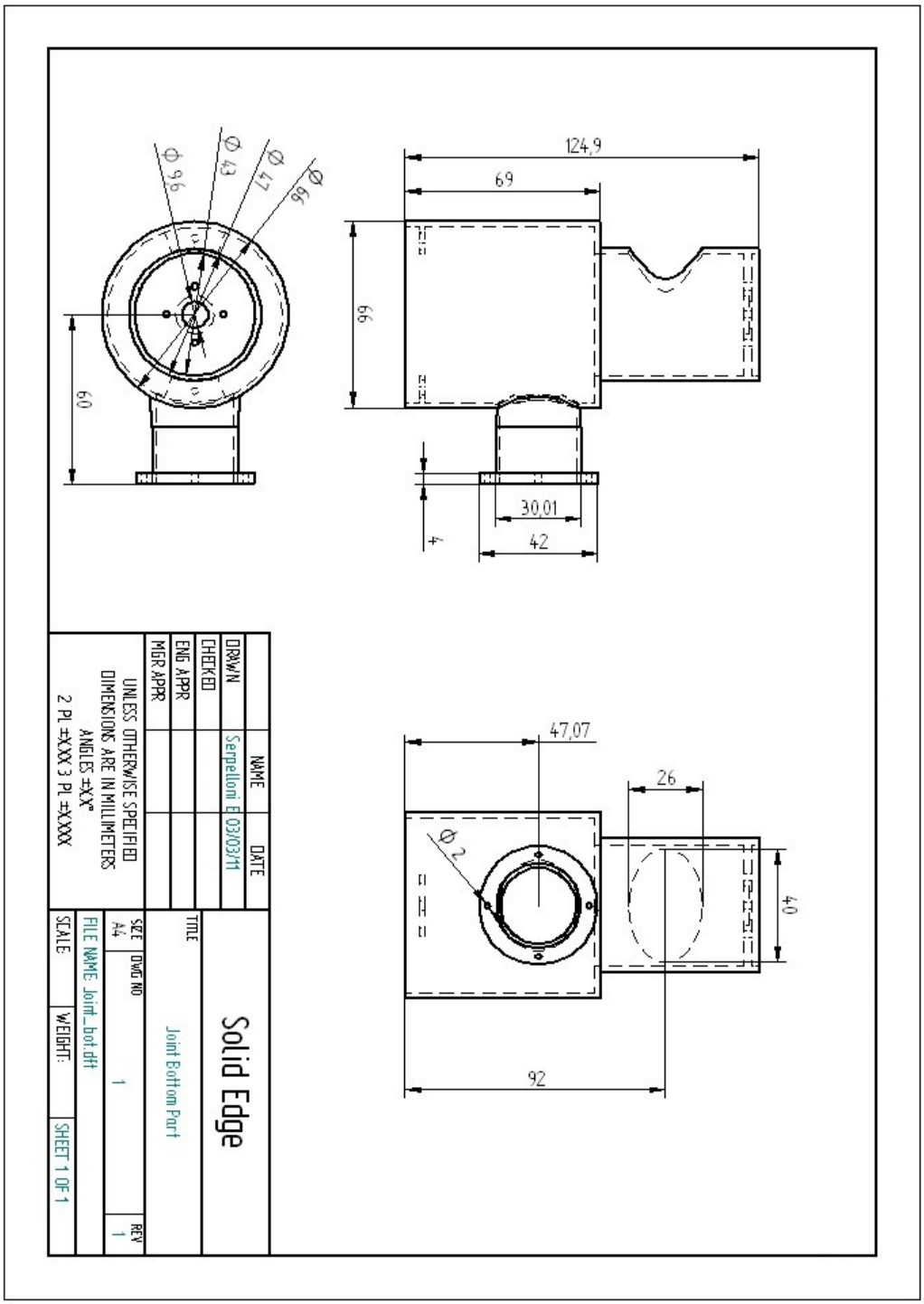
each joint is identical to the others: the manipulator can be virtually expanded to each desired number of DOFs. This modularity is supported by the software itself, that has no limits in the number of DOFs of the system that has to be tested. Also from the point of view of the design philosophy the designed manipulator respects the specifications: all the wires and the electronics are accommodated inside the structure of the joint, this feature could be critical in the case of nano-satellites applications: the space inside the body of the satellite is extremely limited, and it is important to find a rational way to organize the disposition of the elements of the system.

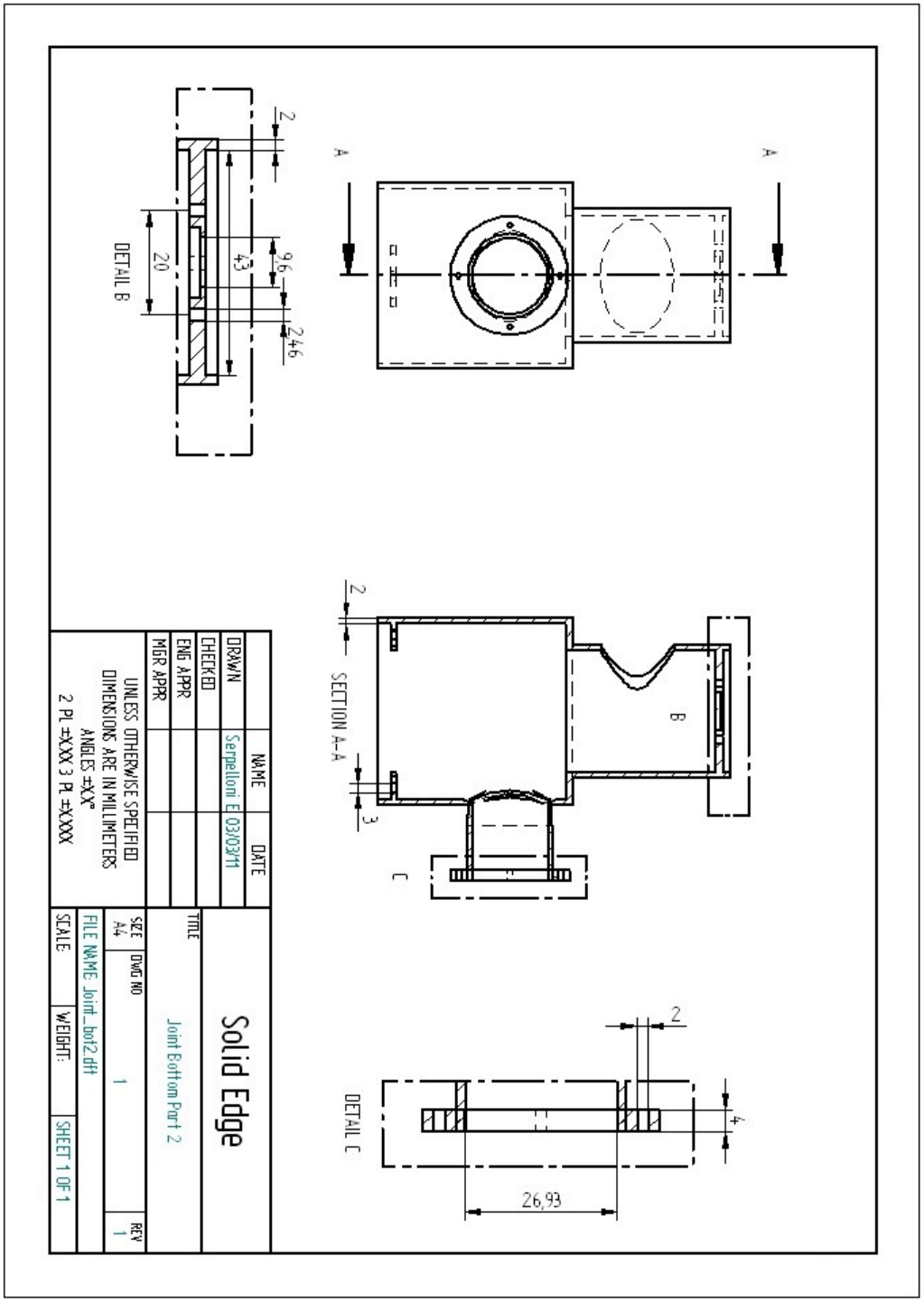
11.1.1 Future Develpments

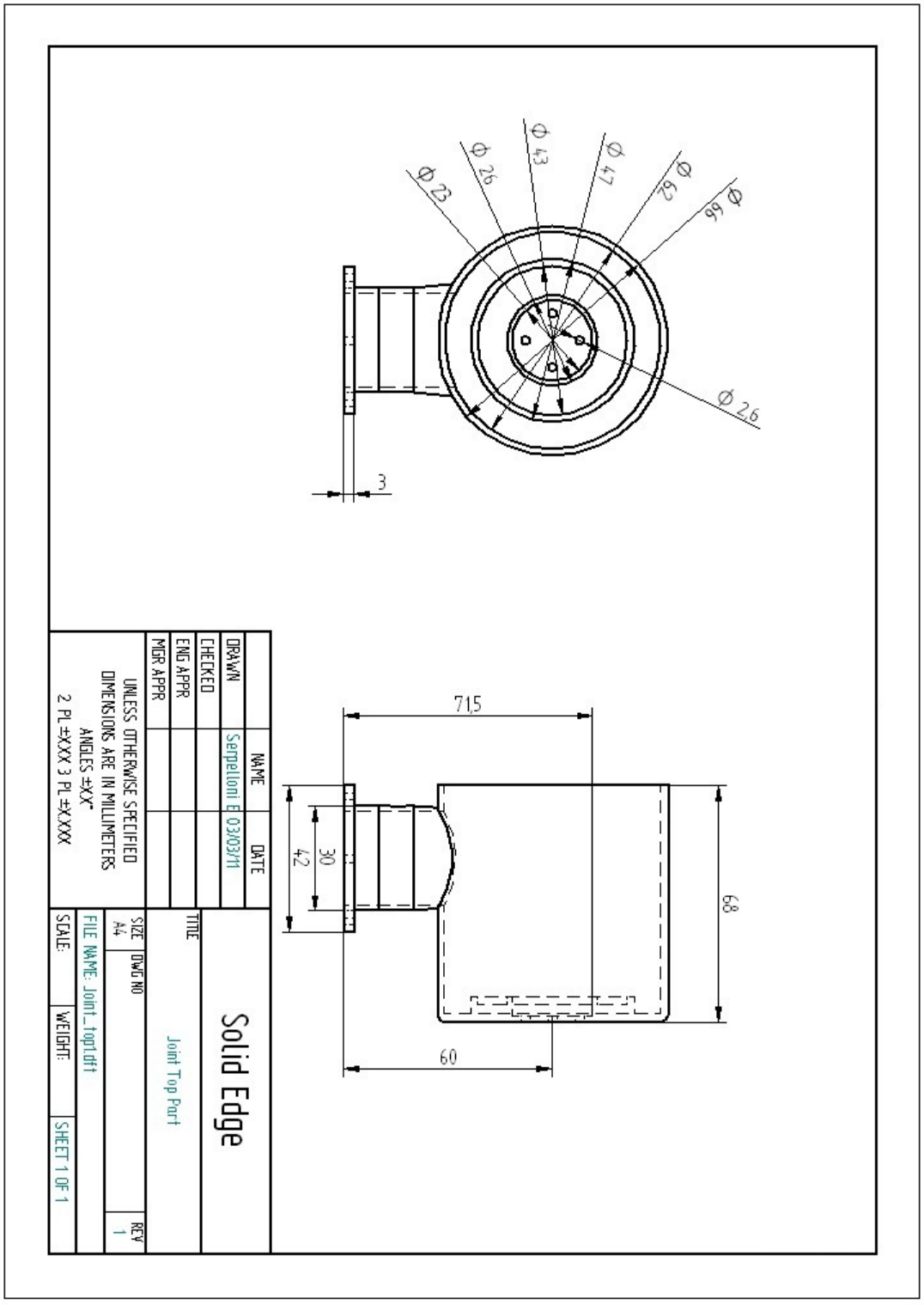
The manipulator and the S-functions for rs232 communication are now ready for experimentation. In order to validate the software from an experimental perspective the whole simulator has to be compiled in a C code. The operation has already been completed and tested for the base of the robots. The integration of the control of the arm will be more complex: a particular attention has to be put on the electronics delays. The configuration of the arm drivers forces to study in a careful way the delays problem deriving from the electronics. Moreover it is necessary to derive the encoder feedback and the other sensors measurements in order to obtain the joints velocities, necessary for the computation of the control law. This operation can be risky: in order to limit the errors it will be necessary to implement low pass filters, to avoid the derivation of the sensors noise.

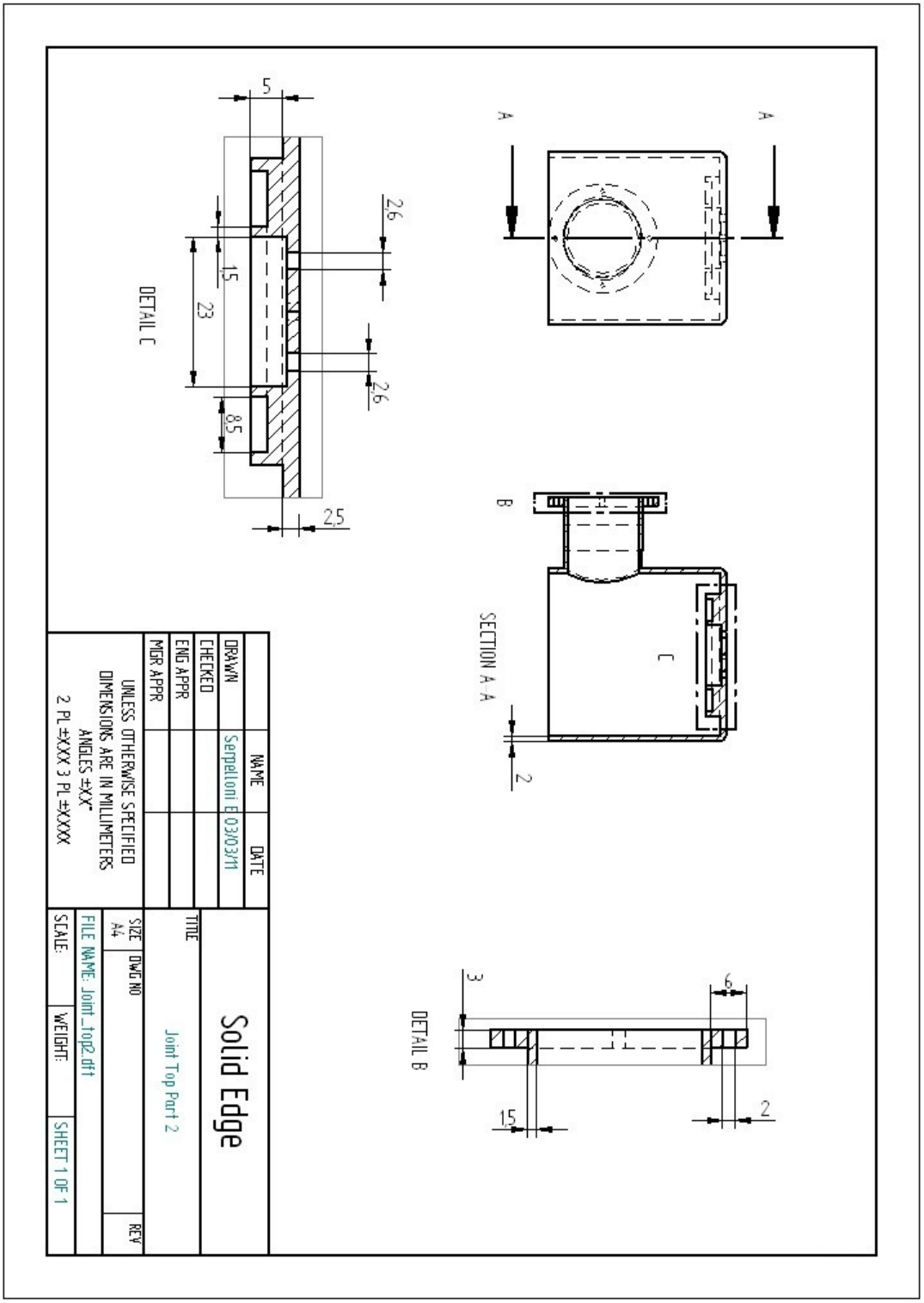
Appendix A

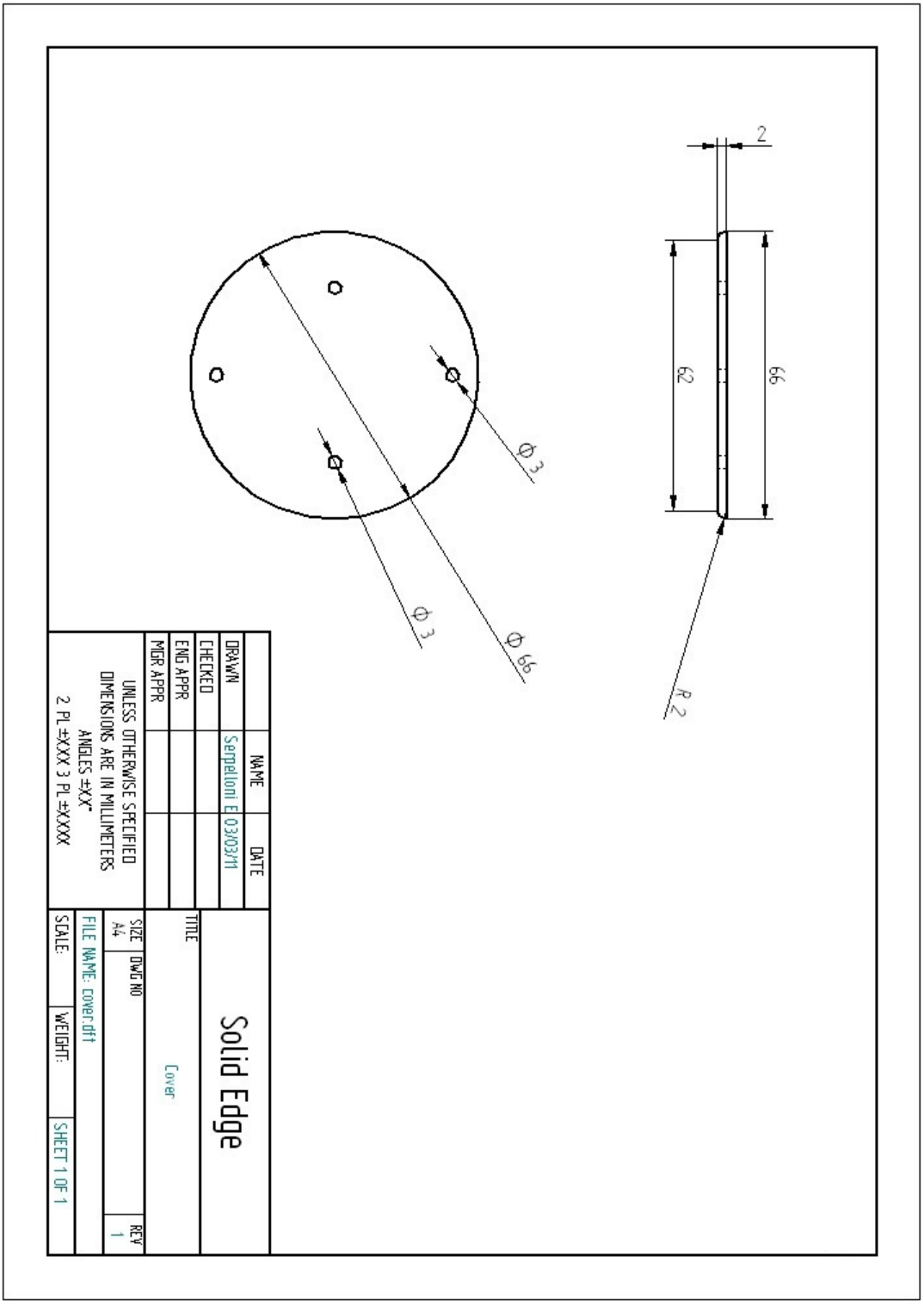
In this Appendix the joint drawings are presented. The drawings have been done using the Solid Edge software.
The first two drwaings represents the bottom part of the joint, where the electronics, the motor and the wires are accommodated.
The second two drawings represent the top part of the joint, where the bearing and the screw hub are fixed. The last drawing is the representation of the cover, whose goal is to close the bottom part of the joint and to give support to the electronics.


Ø 96
Ø 43
Ø 47
Ø 66
60
124,9
69
99
30,01
42
4
47,07
26
Ø 2
40
92
Solid Edge
DRAWN
Serpelloni E
03/03/11
NAME
DATE
CHECKED
ENG APPR
MGR APPR
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN MILLIMETERS
ANGLES ±X°
2 PL ±XXX 3 PL ±XXX
TITLE
Joint Bottom Part
SIZE
A4
DWG NO
1
REV
1
SCALE:
WEIGHT:
SHEET 1 OF 1
FILE NAME Joint_bot.dft


A
A
DETAIL B
2
20
43
9,6
2,46
SECTION A-A
2
3
B
C
DETAIL C
2
4
26,93
Solid Edge
TITLE
Joint Bottom Part 2
DRAWN
CHECKED
ENG APPR.
MGR APPR.
NAME
DATE
Serpelloni E. 03/03/11
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN MILLIMETERS
ANGLES ±XX°
2 PL ±XXX 3 PL ±XXXX
SIZE
A4
DWG NO
1
REV
1
FILE NAME Joint_bot2.dft
SCALE
WEIGHT:
SHEET 1 OF 1


Ø 23
Ø 26
Ø 43
Ø 47
Ø 62
Ø 66
Ø 26
3
715
42
30
68
60
Solid Edge
Joint Top Part
Serpelloni B 03/03/11
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN MILLIMETERS
ANGLES ±XX°
2 PL ±XXX 3 PL ±XXXX
FILE NAME: Joint_top1.dft


A
A
5
26
15
23
26
8.5
25
DETAIL C
B
SECTION A-A
C
2
3
6
15
2
DETAIL B
Solid Edge
NAME
DATE
DRAWN
Sempeloni E
03/03/11
CHECKED
ENG APPR
MGR APPR
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN MILLIMETERS
ANGLES ±XX°
2 PL ±XXX 3 PL ±XXXX
TITLE
Joint Top Part 2
SIZE
A4
DWG NO
REV
FILE NAME Joint_top2.dft
SCALE:
WEIGHT:
SHEET 1 OF 1


2
62
66
R 2
Ø 3
Ø 3
Ø 66
NAME
DATE
DRAWN
Serpelloni E 03/03/11
CHECKED
ENG APPR
MGR APPR
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN MILLIMETERS
ANGLES ±XX°
2 PL ±XXX 3 PL ±XXXX
Solid Edge
TITLE
Cover
SIZE
A4
DWG NO
REV
1
FILE NAME: cover.dft
SCALE:
WEIGHT:
SHEET 1 OF 1

Bibliography

[1] Singer P.W. *Wired for War: The Robotics Revolution and Conflict in the 21st Century*. The Penguin Press, 2009.

[2] Yamamoto Y., Yun X. Coordinating Locomotion and Manipulation of a Mobile Manipulator. In *IEEE Transactions on Automatic Control*, volume 39, June 1994.

[3] Hootsman N.A.M., Dubowsky S. Large Motion Control of Mobile Manipulators Including Vehicle Suspension Characteristics. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2336–2341, April 1999.

[4] McMillan S., Orin D.E., McGhee R.B. Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator. In *IEEE Transactions on System, Man and Cybernetics*, volume 25, pages 1194 – 1206, August 1995.

[5] Schjolberg I., Fossen T. Modelling and Control of Underwater Vehicle-Manipulator Systems. In *Proc rd Conf on Marine Craft Maneuvering and Control*, pages 45–57, 1994.

[6] Mahesh H., Yuh J., Lakshmi R. A Coordinated Control of an Underwater Vehicle and Robotic Manipulator. *Journal of Field Robotics*, 8(3):339–370, June 1991.

[7] Ballantyne G.H. Tobotic Surgery, Telerobotics Surgery, Telepresence and Telementoring. *Surgical Endoscopy and Other Interventional Results*, July 2022.

[8] Vafa Z. The Kinematics and Dynamics of Space Manipulators: The Virtual Manipulator Approach. *The International Journal of Robotics Research*, 9, August 1990.

[9] Vafa Z., Dubowsky S. On the Dynamics of Manipulators in Space using the Virtual Manipulator Approach. In *IEEE International Conference on Robotics and Automation*, pages 579–585, May 1987.

[10] Vafa Z., Dubowsky S. A Virtual Manipulator Model for Space Robotic Systems. In *Proceedings of the Workshop on Space Telerobotics*, volume 3, pages 342–344, 1987.

[11] Dubowsky S., Papadopoulos E. The Kinematics, Dynamics, and Control of Free-Flying and Free-Floating Space Robotic Systems. In *IEEE Transactions on Robotics and Automation*, volume 9, pages 531–543, October 1993.

[12] Torres M.A. The Disturbance Map and Minimization of Fuel Consumption During Space Manipulator Maneuvers. Master's thesis, Massachusetts Institute of Technology, 1989.

[13] Dubowsky S., Torres M.A. Minimizing Attitude Control Fuel in Space Manipulator Systems.

[14] Dubowsky S Torres MA. Path Planning for Space Manipulators to Minimize Spacecraft Attitude Disturbances.

[15] Torres M.A. Minimizing Spacecraft Attitude Disturbances in Space Manipulator Systems.

[16] Moosavian A.A., Papadopoulos E. On the Kinematics of Multiple Manipulator Space Free-Flyers and their Computation. *Journal of Robotic Systems*, 1998.

[17] Moosavian A. A., Papadopoulos E. Dymamics and Control of Space Free-Fyers with Multiple Manipulators.

[18] Papadopoulos E. *On the Dynamics and Control of Space Manipulators*. PhD thesis, Massachusetts Institute of Technology, October 1990.

[19] Siciliano B. Kinematic Control of Redundant Robot Manipulators: A Tutorial. *Journal of Intelligent and Robotics Systems*, pages 201–212, 1990.

[20] Siciliano B., Khatib O. *Handbook of Robotics*. Springer-Verlag New York, 2007.

[21] Chaumette F., Marchand E. A Redundancy-Based Iterative Approach for Avoiding Joint Limits: Application to Visual Servoing. In *IEEE Transactions on Robotics and Automation*, volume 17, pages 719–730, October 2001.

[22] Marchand E., Rizzo A., Chaumette F. Avoiding Robot Joint Limits and Kinematic Singularities in Visual Servoing. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 1, pages 297–301, August 1996.

[23] Kathib O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*.

[24] Maciejewsky A., Klein C. Obstacle Avoidance for Kinematically Redundant Manipulators in Dinamically Varying Environments.

[25] Featherstone R. *Rigid Body Dynamics Algorithms*. Springer, 2008.

[26] Featherstone R., Orin D. Robot Dynamics: Equations and Algorithms. In *IEEE International Conference in Robotics and Automation*, volume 1, pages 826–834, 2000.

[27] Xequian W., Wenfu X., Bin L., Cheng L.,. General Scheme of Teleoperation for Space Robot. In *Proceedings of the 2008 Conference on Advanced Intelligent Mechatronics*, July 2008.

[28] Soll E., Ulrich W., Artigas J., Preusche C., Kremer P., Hirzinger G., Letschnik J., Pongrac H. Ground Verification of the Feasibility of Telepresent On-Orbit Servicing. *Journal of Field Robotics*, 26(3):287–307, 2009.

[29] Soll E., Letschnik J., Ulrich W., Artigas J., Kremer P., Preusche C., Hirzinger G. On-Orbit Servicing, Exploration and Manipulation Capabilities of Robots in Space. *IEEE Robotics and Automation Magazine*, pages 29–33, December 2009.

[30] Siciliano B., Sciavicco B., Villani L., Oriolo G.,. *Robotics. Modelling, Planning and Control*. Springer, 2009.

[31] Featherstone R. A Beginner's Guide to 6-D Vectors. *IEEE Robotics and Automation Magazine*, 17(4):88–99, December 2010.

[32] Dubowsky S., Papadopoulos E. On the Nature of Control Algorithms for Free-Floating Space Manipulators. In *IEEE Transactions on Robotics and Automation*, volume 7, pages 750–758, December 1991.

[33] Dubowsky S., Papadopoulos E. On the Nature of Control Algorithms for Space Manipulators. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1102–1108, May 1990.

[34] Dubowsky S., Torres M.A. Path Planning for Space Manipulators to Minimize Spacecraft Attitude Disturbances. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2522–2528, April 1991.

[35] Dubowsky S., Papadopoulos E. Coordinated Manipulator/Spacecraft Motion Control for Space Robotic Systems. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1696–1701, April 1991.

[36] Xu Y., Kanade T., editor. *Space Robotics: Dynamics and Control*. The Springer International Series in Engineering and Computer Science.

[37] Yoshida K.,. Experimental Study on the Dynamics and Control of a Space Robot with Experimental Free-Floating Robot Satellite EFFORTS Simulators. *Advanced Robotics*, 9(6):583–602, 1995.

[38] Yoshida K., Umetani Y. Control of Space Free-Flying Robots. In *Proceedings of the 29th Conference on Decision and Control*, December 1990.

[39] Yoshida K., Umetani Y. Resolved Motion Rate Control of Space Manipulators with Generalized Jacobian Matrix. *IEEE Transaction on Robotics and Automation*, 5(3), June 1999.

[40] Hootsmans N., Dubowsky S., Mo P. The Experimental Performance of a Mobile Manipulator Control Algorithm. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, volume 3, pages 1948–1954, 1992.

[41] Nanchev D., Umetani Y., Yoshida K. Analysis of a Redundant Free-Flying Spacecraft/Manipulator System. *IEEE Rransactions on Robotics and Automation*, 8(1), February 1992.

[42] Umetani Y., Yoshida K. Workspace and Manipulability Analysis of Space Manipulator. *Transactions of the Society of Instrument and Control Engineers*, E-1(1), 2001.

[43] Marchesi M. Angrilli F., Bettanini C. On Ground Experiments of Free-Flyer Space Robot Simulator in Intervention Missions. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics and Automation in Space*, June 2001.

[44] Agrawal S.K., Grimella R., Desmier G. Optimal Workspace Designs of Free-Floating Planar Manipulators. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, November 1991.

[45] Yoshida K.,. Space Robot Dymamics and Control: To Orbit, From Orbit, and Future.

[46] Umetani Y., Yoshida K. Workspace and Manipulability Analysis of Space Manipulator. *Transactions of the Society of Instrument and Control Engineers*, 2001.

[47] Romano M., Friedman D., Shay T. Laboratory Experimentation of Autonomous Spacecraft Approach and ocking to a Collaboratory Target. *Journal of Spacecraft and Spacecraft and Rockets*, 1, 2007.

[48] Romano M., Friedman D., Shay T. Ad-hoc Wireless Networking and Shared Computation based upon Linux for Autonomous Multi-Robots Systems. *AIAA Journal of Aerospace Computing, Information and Communication*, 6, 200.