UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA AEROSPAZIALE

TESI DI LAUREA MAGISTRALE IN INGEGNERIA AEROSPAZIALE

Design of a Robotic Arm for Laboratory Simulations of Spacecraft Proximity Navigation and Docking

RELATORE: PROF. ALESSANDRO FRANCESCONI

LAUREANDO: ANDREA ANTONELLO

ANNO ACCADEMICO 2012-2013

To my families, in Italy and California.
To you, Silvia.

Contents

Abstract

The increasing number of human objects in space has laid the foundation of a novel class of orbital missions for servicing and maintenance. The main goal of this thesis is the development of a robot manipulator for the simulation of close approach orbital maneuvers, with particular attention to docking and capture. There are currently very few facilities able to simulate relative motion between orbiting objects: DLR's EPOS experiment is the leading edge of European research on RvD ground simulations. The 25 m long testing site consists in two industrial anthropomorphic robots that can reproduce docking and berthing scenarios, taking into account dynamic contacts, gravity and even sunlight illumination for utmost realistic simulations. This project tries to propose a viable alternative to these huge and costly RvD structures; the addition of force sensing transducers and the possibility to dynamically scale the simulations makes the manipulator a cheap and portable hardware-in-the-loop testing bench for orbital phenomena. After selecting the most dexterous robotic configuration, the kinematic and dynamic problems were analyzed; a basic PID controller was then implemented and its stability to step response and disturbances successfully verified. An extended simulation campaign, comprising Matlab and SimMechanics environments, confirmed the theoretical models and allowed to reproduce typical rendezvous and docking maneuvers (providing useful data for the sizing). By integrating a force sensor, it was possible to impose and simulate orbital motion and to account for any force disturbance. With information deriving on structural analyses and dynamics extrapolations, a preliminary design was carried out, and led to the translation of the theoretical requirements into the sizing and selection of the structure, the hardware and the actuators. The final robot is able to simulate RvDs inside a spherical working space of 1.3 m radius, with a total mass of just 7.5 kg. This thesis sets the foundations for the physical realization of the arm, which will serve as an innovative platform for a multidisciplinary satellite testing facility.

Chapter 1

Introduction

1.1 A long journey

Even though robotics started to become an important field of nowadays technology during the course of the last decades of the 20^{th} century, it has always been an interesting field of research throughout the history of mankind. Humans have always tried to seek substitutes that would be capable to mimic their actions and behavior. The history of robots has in fact its roots as far back as ancient myths and legends: one of man's greatest goals has been to instill life in their artifacts.

In the Iliad, god Hephaestus created talking mechanical servants out of gold. Heron of Alexandria (10–70 AD) created some mechanical devices at the end of the 1^{st} century AD, including one that reportedly could speak [13]; records show that Aristotle, in his book Politics, speculated that automatons might someday substitute humans in manual labor, thus calling a halt to slavery.

In the 10^{th} century BC, the Cosmic Engine (a 10 m clock tower) had been built in China, featuring bell ringing mannequins and automatic ringing gongs. During the Artuqid dynasty, Al Jazari invented several automatic machines, among which it's noteworthy the first programmable humanoid robot, dated 1206: using a combination of cams and levers, it was capable of moving and playing drums.

It is only with Leonardo da Vinci, however, that we have the first recorded

design of a humanoid robot (1505): a moving mechanical knight, able to move its head and arms, stand up and sit down.

With the advent of the Industrial revolution, the idea of a robot started to be applied: by the 19^{th} century, cloth production was totally automated.

In the literature, it's worth mentioning Mary Shelley's Frankestein, which condenses the dramatic struggle of man in the search for a mechanic replacement of his capabilities.

It is only in the 1920s that the term **robot** was first introduced in the english vocabulary by Czech playwright Karel Capek. The image of the robot as a mechanical artifact starts in the 1940s when fiction writer Isaac Asimov conceived the robot as an automaton of human look but lacking of emotions. Asimov identified the term **robotics** as the science committed to the study of robots, which was founded on three fundamental laws [33]:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given by human beings, except when such orders would conflict with the first law.

3. A robot must protect its own existence, as long as such protection does not conflict with the first or second law.

Use of the industrial robot as a viable manufacturing device started in the 1960s, along with CAD/CAM systems, and characterizes the latest trends in the automation. The principal milestones of modern robot technology are presented below [29]:

1947 - first servoed electric powered teleoperator is developed

1948 - a teleoperator is developed including force feedback

1954 - George Devol designs the first **programmable robot**

1956 - Josh Engelberger buys the rights to Devol's robot and founds the Unimation Company

1961 - the first Unimate robot is installed in New Jersey plant of General Motors

1963 - the first robot vision system is developed

1973 - the first Stanford Arm is developed at Stanford University

1974 - Milacrom introduces the T^3 computer controlled robot

1978 - Unimation develops the PUMA robot

1979 - the SCARA robot design is introduced in Japan

1981 - the first *direct drive* robot is created at Mellon University

1989- chess playing robot HiTech defeats chess master Arnold Denker

1996 - Honda's P2 humanoid robot was first shown

1997 - Sojourner rover performed semi-autonomous operations on Mars

2001 - Canadarm2 was launched into orbit and attached to the ISS

2004 - Cornell University revealed a robot capable of self-replication

1.2 Motivation and state of the art

In the aerospace industry, the applications of robotics have their maximum development. The two main macroares of interest are the Orbital Robotics and the Planetary Rovers [28].

Orbital Robotics comprises the implementation of manipulation and mobility for scenarios such as ISS tasks and satellite servicing. Planetary Rovers address scenarios such as planetary exploration from mobile robot on the surface.

Orbital robotics, due to space environment (radiation, micro-gravity, thermal stresses, etc.) poses unique challenges to robot and robot algorithms, and sets the need for new and innovative autonomous systems.

The design of servicing operations and devices is probably one of the most important research field in space robotics. Servicing operations range from simple inspection to upgrade of components and refuelling [16]. Historical

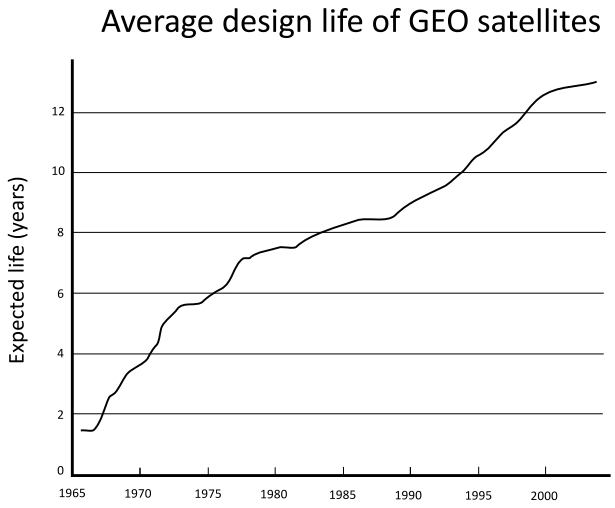


Figure 1.1: Timeline of average GEO satellite design life.

analysis indicates that the combination of the 5% failure rate of launch vehicles coupled with 9% failure rate of satellites during their operational lives will cause the failing of 1/7 of the satellites before the expected end of life (EOL) [31]. Nowadays, the usual approach in trying to avoid these failures is to use proven (usually a synonymous for obsolete) technology and to incorporate massive redundancy. Although the use of proven technology helps to mitigate mission risk, it also has the negative effect of limiting satellite performances.

The increase of costs associated with growing complexity of payloads have led to the need of augmenting satellite design lifetimes in order to obtain a sufficient investment return (Fig 1.1).

One downside of this increased lifetime is the inability to update the hardware and software with modern avionics, in an era governed by "Moore's law"[1]. This slowdown limits the agility of satellite operators in capturing emergent terrestrial markets [21].

All these limitations and the substantial absence of a maniteinance industry for satellites (which is a fundamental part of ground systems such as

[1]Moore law's affirms that there's a doubling of the processing speed of new computer chips every 18÷24 months

automation and aviation industries), are pushing hard for the development of on-orbit servicing (OOS). Among the main operations of OOS, the most important are:

- *Inspecting*: the observation of a space ojected in order to gather information about its status and physical condition are usually the first operations before other OSS activities can take place.

- *Relocating*: this is suitable when the target object has attitude problems and the on-board systems are not able to put the satellite into the correct operational configuration. The relocating of Milstar 3 in its GEO slot, for example, is thought to have saved $1.2 bilion of taxpayers dollars in 1999 [30].

- *Augmenting*: if a satellite has been designed with a modular approach, it is possible to upgrade the hardware to state of art technology as years go by. An astonishing example is the WFPC camera onboard of the Hubble telescope, whose efficiency has been increades by a factor of 180 during four servicing operations (a comparison of the techologies is presented in Fig 1.2).

- *Assembling*: this consists in the merging of mating modules to construct space systems that wouldn't be possible otherwise. This is the procedure that was followed in the case of the International Space Station: the overall structure weighs 200000 kg, whereas the maximum payload capacity is 18300 kg [15].

- *Restoring*: these operations include refueling, docking, station keeping providing, repairing and replacing hardware. In 1984, a refueling operation was successfully carried out, transferring 60 kg of hydrazine between two tanks [7].

The possibility of fixing and refurbishing an out-of-order satelite with unmanned vessels might give rise to a multi millionaire business. NASA estimated the costs for a single Hubble servicing mission at $2 billion. If a robotic servicing satellite was to be sent instead, the economic savings would be enormous, not to mention the avoidance of human losses (which is not an

Figure 1.2: Resolution comparison of WFPC camera from Hubble telescope

unlikely scenario in a manned mission). Nowadays, a lot of space agencies and private companies are pushing in this direction.

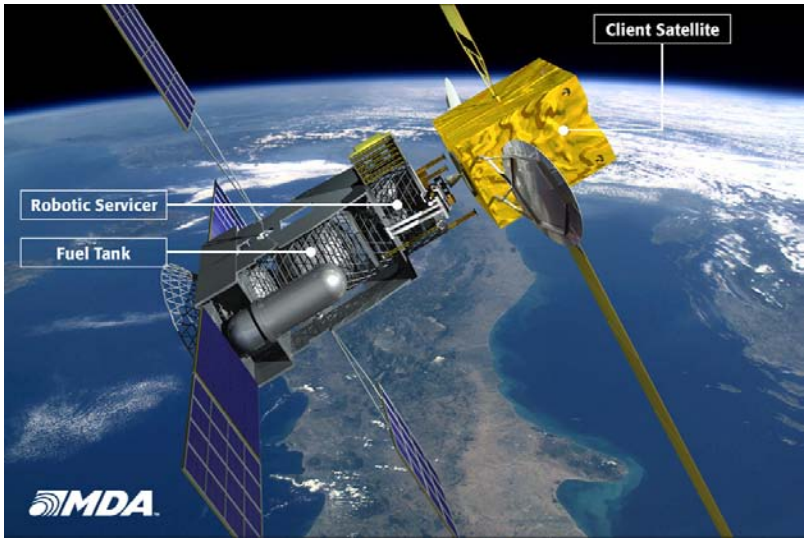
The Canadian aerospace firm MacDonald, Dettwiler and Associates, for example, is developing the Space Infrastructure Servicing (SIS), a spacefracraft for refueling of communication satellites in GEO orbits [27]. SIS is being designed to carry a toolkit able to open most of the ∼40 types of on-orbit fuelling systems. A conceptual design of the system is presented in Fig 1.3 (a).

Intelsat, which owns a 52 communications satellites fleet as of March 2011 [24], has shown a keen interest on the project, founding and sponsoring the inaugural mission with an investment of $280 millions [25].

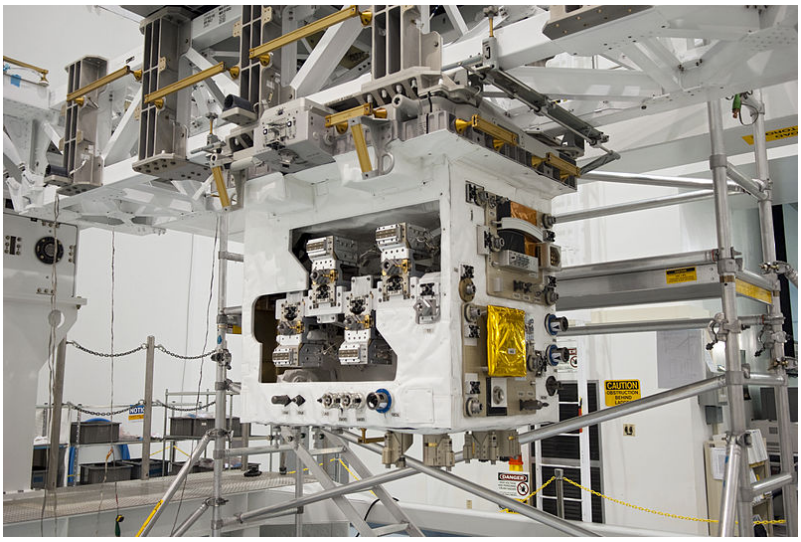
NASA, on the other hand, has already developed and launched a demonstration technology named Robotic Refuelling Mission (RRM). The servicing satellite, Fig 1.3 (b), successfully performed an extensive series of robotically actuated fuel transfer on the ISS (2011) with the aid of the Canadarm manipulator. The long term goal of NASA for this project is to transfer this technology to the commercial market.

Importance of relative attitude operations

It is immediate to notice that, in all the above mentioned operations, the success is strictly linked to the way in which the chaser and the target satellites

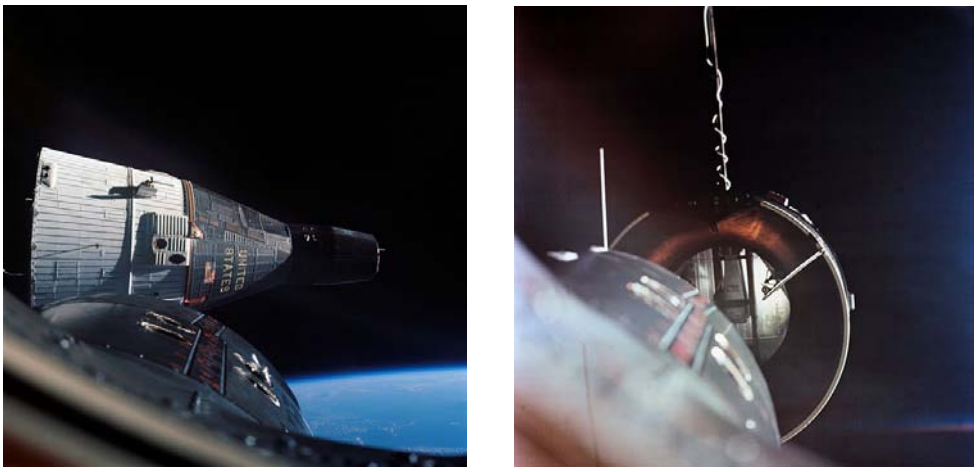


(a)



(b)

Figure 1.3: MDA's Space Infrastucture Servicing concept design (a) and NASA's Robotic Refuelling Mission satellite (b).



(a) (b)

Figure 1.4: Gemini 6's first successfull rendezvous (a) and Gemini 8's first successfull docking (b).

move and interact with respect to each other.

Regardless of the operation to be carried out (a refulling rather than an assembly), it is mandatory to being able to model and control the relative motion.

All the close-approach operations fall under the name of *space-rendezvous*: it follows quite obviously that a precise match of the spacecrafts' orbital velocities is needed, allowing them to remain at a constant distance through orbital station-keeping. Rendezvous may or may not be followed by docking or berthing, which allow a physical contact and create a link between the objects.

The problem of knowning the orbital mechanics of the phenomena hasn't always been obvious. The first rendezvous attemp, for example, was carried out on June 3, 1965, when a Gemini 4 spacecraft was supposed to dock with a spent Titan II upper stage [12]. Astronaut Jim McDivitt tried to manually approach the target, but both him and the ground station engineers had yet to learn the orbital mechanics involved in the process: simply pointing at the target and fire the thrusters won't result in a successfull approach, but will lead to a progressive drift from the target's orbit.

Only on December 15, 1965, Wally Shirra, on board of Gemini 6, suc-

cessfully completed a rendezvous towards Gemini 7, maintaining a station keeping within 30 cm for more than 20 minutes (Fig 1.4 (a)). With Gemini 8, in 1966, Neil Armstrong (now aware of the physical laws involved), was able to perform the first docking with the unmanned Agena Vehicle (Fig 1.4 (b)).

Hardware in the loop (HIL) testing facilities: state of the art

The importance of relative motion for rendezvous and docking operations, calls for an approriate laboratory facility able to reproduce on orbit conditions.

This can be achieved only with a robotic structure that simulates the target and chaser's kinematics and dynamics. There are very few facilities that enable such experiments. One the most important is probably DLR's *European Proximity Operations Simulator* (EPOS) [11].

The original EPOS was designed as a joint-venture between DLR and ESA in the late 1980s, as the need for a rendezvous and docking (RvD) testing facility arised. In that period, in fact, there was a keen interest for an own unmanned vessel for supply and service flights to the ISS. In 1991, the facility started its first tests, and was constituted by three subsystems: a 6 DOF gantry, able to host a 100 kg payload at the end effector, a structure carrying the target object, and an auxiliary illumination system to achieve realistic lighting conditions.

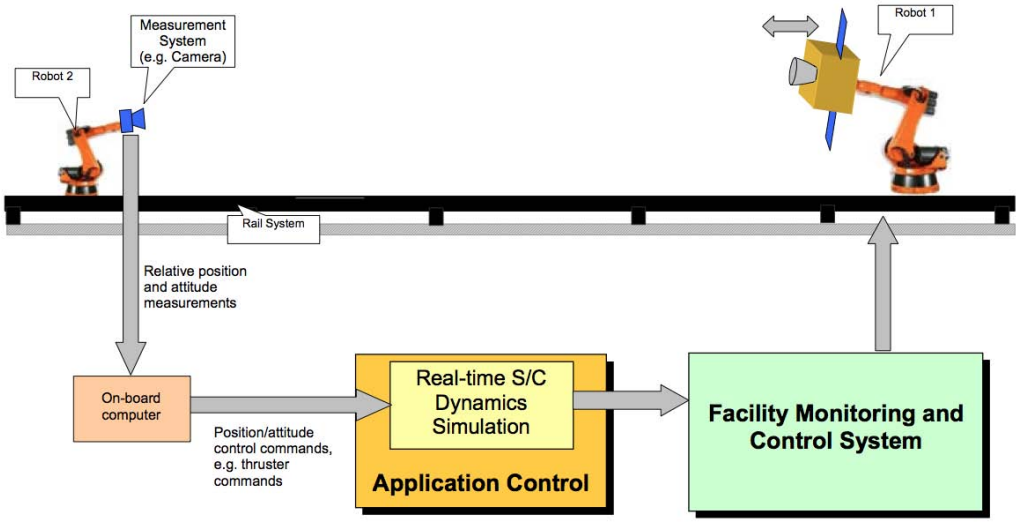
This system served for testing for almost 20 years and was renewed due to the demand for better RvD simulation accuracy. The current facility was built in 2009 and it's a joint effort between the DLR's GSOC, which provided the overall design as well as the orbital mechanics background, and DLR's Robotics and Mechatronics Institute, which contributed to the robotic technology, on behalf of their solid background on the subject.

The approaching vehicles are simulated via two anthropomorphic industrial robots, with the target fixed on the ground and the chaser mounted on a 25 m rail for extra mobility.

An application PC feeds in synchronous trajectories via a Matlab/Simulink interface, and the control and measuring systems allow for a position and an-

(a)

(b)



(c)

Figure 1.5: EPOS RvD simulation facility: laboratory configurations (a), (b) and conceptual operating diagram

gular accuracy of respectively 2 mm and 0.2°. All the trajectory simulation are carried out via an implementation of Clohessy-Wiltshire coordinate system.

Thesis motivation

This thesis focuses on the development and design of a robotic manipulator and its kinematic and dynamic modelling for the reproduction of orbiting operations. The innovative aspect of this structure will be the integration of a force sensing device that will take into account both disturbances and contact forces between the objects. Through a dedicated algorithm, the system is able to compute in real time the consequences of these inputs in terms of trajectory modifications, which are then fed to the *hardware in the loop* (HIL) control system.

Moreover, the software governing the manipulator can be commanded to perform active maneuvers and relocation: as a consequence, this structure can be used as the testing bench for any attitude modification system, providing a faithful, real time simulation of the orbital scenario.

Furthermore, with the aid of dynamic scaling laws, the potentialities of the facility can be exponentially increased: the simulation environment is not longer bounded to be as big as the robot workspace, but could be several orders of magnitude bigger, allowing for the reproduction of otherwise preposterous scenarios in a laboratory environment.

Finally, the robot itself can be used as part of the simulated maneuvers. Berthing operations and uncooperative target docking, for example, can be performed. This latter research field, uncooperative docking, as long with RvD rendezvous and docking operations, are under study at CISAS research center (Padua, Italy) [6]: the manipulator presented in this paper could serve as the main testing facility for the reproduction and the verification of theoretical and numerical analysis.

1.3 Robot preliminary design

1.3.1 Overview

Before embarking in the kinematic and dynamic analysis, it is necessary to identify the main components of a robotic system. Even for a complex architecture, it is always possible to identify a general block diagram [23]:

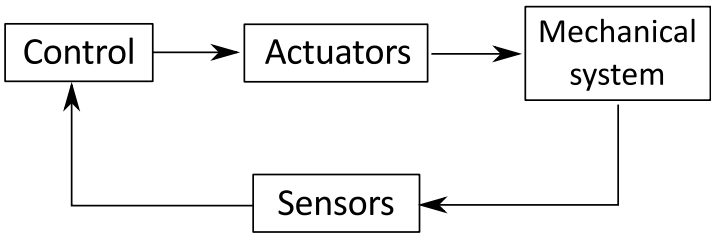


Figure 1.6: Robotic system components.

The core component is the mechanical system, made up of a manipulation apparatus (arms, links, end effectors, artificial hands) and a movement apparatus (wheels, crawlers, legs).

The capability to execute a task is made possible by the actuators block, which provides motion to the manipulation and movement apparatus.

The connection with the outside world is made possible by the presence of sensors, enabling the acquirement of data on the internal status (*proprioceptive* sensors, such as encoders) and on the external status (*exteroceptive* sensors, such as force sensors or vision system)

Finally, the control block permits to make the whole system an harmonious working machine, reading data from the sensors and commanding the actuators with well-tuned control laws.

1.3.2 Mechanical structure

The main distinction between different robots concerns their mechanical structure. That is, the way in which the links are connected and the way in which they move with respect to each other. A robot manipulator is a sequence of rigid bodies (called *links*) which are connected by joints. The

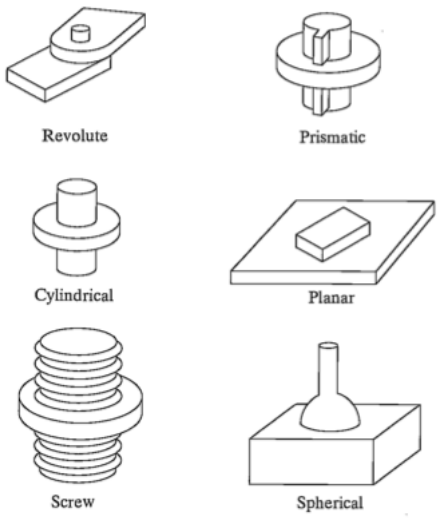


Figure 1.7: Joint configuration types

configuration is most of the times that of an open kinematic chain; usually, at the end of the maniuplator, there is the end-effector, providing the needed dexterity for the execution of tasks.

The mobility is ensured by the presence of joints, which can be of different type and can introduce one or multiples degrees of freedom[2].

Mechanical design considerations when building robots have narrowed the joint choices to two main types: revolute or prismatic. In a revolute joint, the connected bodies rotate with respect to a common axis, whereas in a prismatic joint they slide without rotation. Both of these configurations have a single degree of freedom. When more than one degree of freedom is needed, other less used joint options are available (Fig 1.7).

For simplicity, industrial robots have usually single degree of freedom links. The number of DOF characterizes the mobility of the robot in the operational space: in order to arbitrarily position the end effector in 3D space, 6 DOF are required (excluding for the moment the singularities), 3 being translational and 3 rotational.

When a robot has less than 6 DOF, it will have some limitations on the

[2]Note that, in the special case of singularity, they do not provide any contribution to the overall number of degrees of freedom.

end effector orientation in his working space; when, on the other hand, there are more than 6 DOF, the robot is kinematically redundant, and the same position in space can be obtained via several configurations.

Among the main choices of robot configurations, we will describe the following: cartesian, cylindrical, spherical, SCARA, anthropomorphic [23]. Then, according to our requirements, the most approriate arrangement will be chosen.

Cartesian: This geometry is characterized by three prismatic joints whose axes are reciprocally perpendicular. It is an extremely simple solution and allows to execution of straight motions in the cartesian workspace (which is a rectangular parallelepided). The main drawback is the limited dexterity of the end effector, which has a fixed approaching orientation, Fig 1.8 (a). It is industrially used for handling and assembly of materials and goods.

Cylindrical: In the cylindrical configuration, the first joint is replaced with a revolute connection. Similarly to the cartesian type, here every DOF corresponds to a cartesian variable expressed in cylindrical coordinates; the workspace is a cylinder, Fig 1.8 (b). It has again limited dexterity at the end effector. Commonly used for carrying goods of large dimensions.

Spherical: The cylindrical type is a further modification of the cartesian structure, in which the first two joints have been substituted with revolute joints. Each degree of freedom corresponds to a cartesian variable, here expressed in spherical coordinates. Referring to Fig 1.8 (c), the workspace is a sphere. The main industrial use of these robots is for machining operations.

SCARA: The SCARA (*Selective Compliance Assembly Robot Arm*) geometry is realized by using two revolute and one prismatic joint in a way in which all the joint axes are parallel. The workspace area is pictured in Fig 1.8 (d) and usually depends on the link parameters. It is industrially suitable for vertical assembly.

Anthropomorphic: In this case, all the three joints are revolute, and since the first joint's axis is perpendicular to the ground, it resembles the shape of a human arm (hence its name). This is by far the most dexterous configuration, since it has all revolute joints. The main drawback is that correspondence between cartesian variables and joint variables is lost. The workspace, from Fig 1.8 (e), is approximately a sphere and its industrial application has an extremely wide range.

According to 2012 report of the *International Federation of Robotics* (Fig 1.9), 63% of worldwide installed manipulators are anthropomorphic, 15% are cartesian, 12% are SCARA and 10% are cylindrical type.

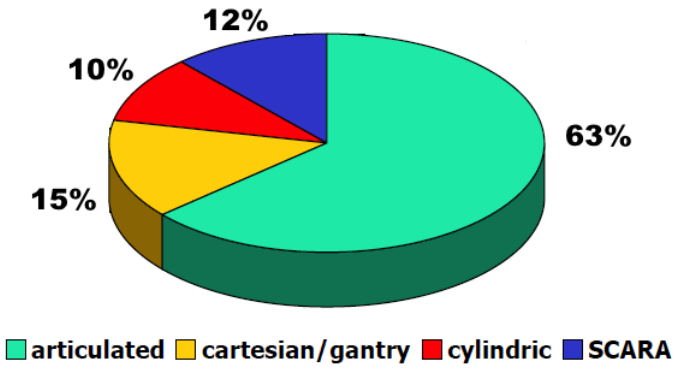
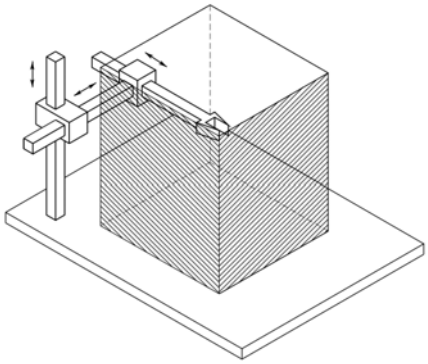


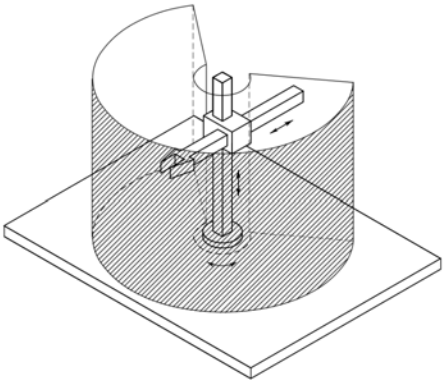
Figure 1.9: Worldwide robot distribution of robots by kinematic configuration type.

From the previous analysis, it follows quite clearly that in our case, since we are looking for the maximum dexterity, the anthropomorphic manipulator seems to be the best choice.

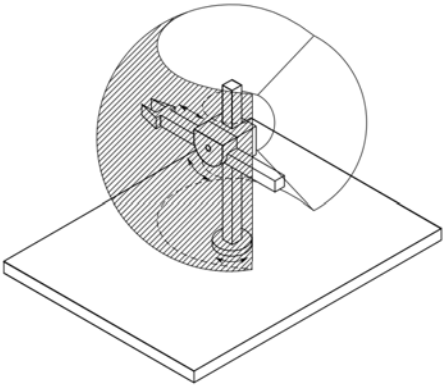
Among the requirements that need to be satisfied in this project, there is the workspace: the manipulator, in fact, has to have sufficient dexterity in a cube whose volume is at least $0.5\,\mathrm{m}\times0.5\,\mathrm{m}\times0.5\,\mathrm{m}$. In the sizing analysis (Chapter 7), the link lengths will be chosen in order to fulfill this requirement.



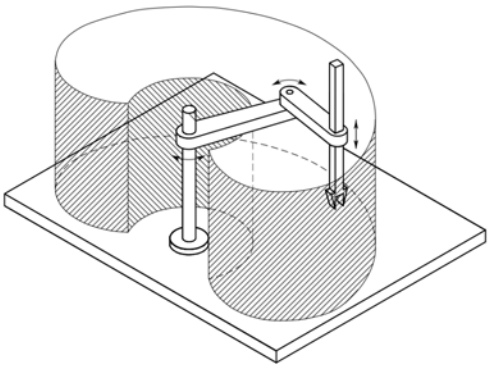
(a) Cartesian



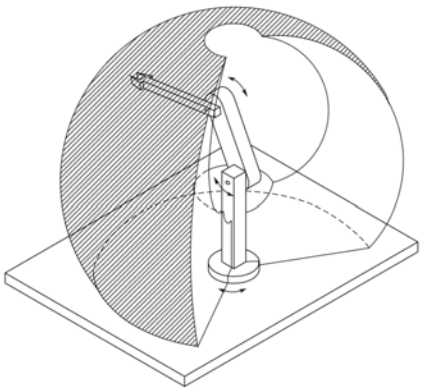
(b) Cylindrical



(c) Spherical



(d) SCARA



(e) Anthropomorphic

Figure 1.8: Main kinematic configurations for manipulators

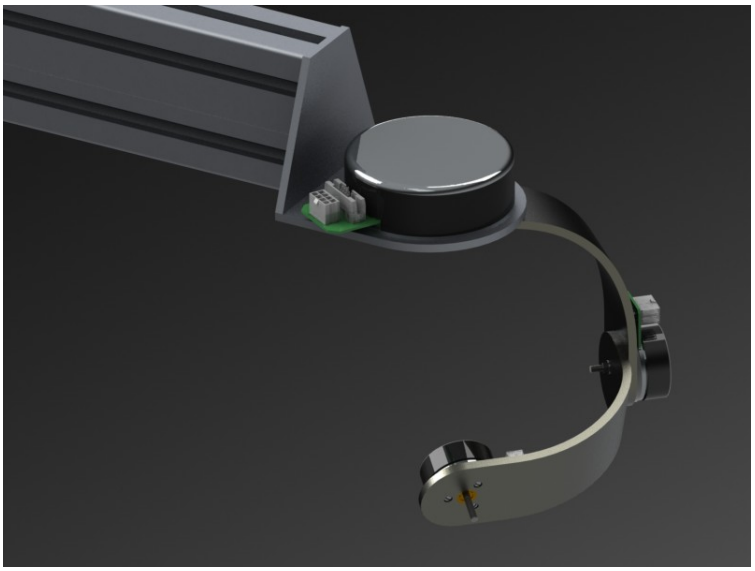


Figure 1.10: End effector custom design.

1.3.3 *End-effector* configuration

Stantis rebus, the manipulator has now only 3 DOF. In order to simulate the motion of an object in 3D space, 3 more degrees of freedom are needed. This is accomplished by adding a compact structure at the end of the last link of the actual kinematic chain. This structure is called end-effector and there exist many choices for its design and configuration.

The need for compactnees usually pose some complications in its design. Moreover, Chapter 2 will explain how some particular end-effector designs may affect positively the kinematic and dynamic analysis: if the axes of the last three joints, in fact, intersect in a point, then Pieper's simplified solution to the kinematic problem can be applied. Such a configuration is commonly known as *spherical wrist*.

However, *spherical wrists* come in several fashions, and various designs have been proposed along the years. The main filter when choosing a design is the analysis of the singular configurations and the angular ranges of the joints.

A custom made end effector was designed for our application, keeping in mind the avoidance of singular configurations and the maximization of the

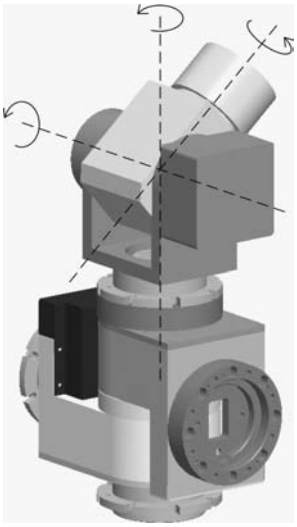


Figure 1.11: Typical industrial end-effector design.

angular range. From Fig 1.10 it can be seen that the first joint (*joint 4*) has theorically[3] no limitations on its angular range. The same thing stands for the second and the third joints (*joint 5* and *6*), which can span all the angles from 0° to 360°.

Notice the particular shape of the second link, which was designed in order to avoid any interference when the axis of *joint 6* is collinear with *joint 4*'s axis; in addition, an adequate gap is present in this configuration to allow the presence of a small object (or a force sensor) at the end of the kinematic chain.

This configuration differs from commonly used industrial end effectors. This is due to the fact that a commercial robot has usually some kind of tool or object at his tip, and a configuration like the one pictured in Fig 1.10 won't allow the presence of voluminous attachments due to the geometrical interferences. Hence, in industrial applications, design configurations like the one in Fig 1.11 [1] are encouraged: a wide mobility of the tip is obtained, though jeopardizing the angular range of the second joint.

[3]For the moment, we ignore the presence of any wire or harness attached to the system, which usually limit the movements.

1.4 Outline

The thesis will cover the main aspects of a manipulator design process; a concise summary of the chapters is provided here below:

- **Chapter 2** - *Kinematics*: This chapter sets the foundations of all the further analyses, providing the tools for the description of the manipulator. Several techniques for relating operational space and joint space variables are presented and compared. This section focuses on the differential kinematics approach, which ultimately leads to the calculation of the Jacobian matrix, one of the most important tool for the analysis of a manipulator. The theoretical description is combined with numerical analyses and trajectory simulations of the different solution approaches.

- **Chapter 3** - *Trajectory definition*: A general overview on the main trajectory techniques is presented, with analyses and numerical simulations of the most common approaches. Specifically, particular care was given to the study of cartesian operational space trajectory planning, and some reference trajectory are implemented with the aid of the kinematic model obtained in the previous chapter.

- **Chapter 4** - *Dynamics*: With the solid background gained from the kinematics chapter, the manipulator analysis is extended to the investigation of dynamic effects. Two approaches are presented, and their advantages/disadvanteges are carefully discussed. After the accurate selection of one of these approaches, in-depth simulations are carried out, combined with critical surveys on the results obtained.

- **Chapter 5** - *Linear feedback control*: In this chapter, a general overview of the ways in which a manipulator can be controlled are explained. Particular attention is given to the *joint space* control techniques. The analysis, on behalf of the linearized model hypotheses, starts by considering the control of a single joint: a design approach for a PD and a PID controller is minutely presented, assisted by numerical simulations of the models. Finally, the extension of the control technique to

a multibody system is discussed.

- **Chapter 6** - *Space trajectory analysis*: In this chapter, the core applications of the manipulator are discussed. From orbital mechanics theory, the power of CW equations is explained and it's applied to the our model. With the aid of equations and block diagrams, several laboratory scenarios are discussed for the simulation of rendzvous maneuvers, disturbances, force contacts and attitude commands; this is followed by an overview on feasible force sensors. Finally, using the models developed for the trajectory simulations, a rendzvous maneuver is implemented, combining the free motion with disturbances and attitude commands.

- **Chapter 7** - *Sizing*: This chapter collects and processes all the data obtained in the previous chapters, and with the aid of geometrical, structural and cost analyses, guides the reader through the design process that finally leads to the physical realization of the structure. The chapter ends with the presentation of rendered images of the final product, combined with drawings and CAD assemblies.

- **Chapter 8** - *Conclusions and future work*: This chapter presents the concluding summary of the thesis, as well as recommendations for future work.

Chapter 2

Kinematics

2.1 Introduction

Kinematics is the study of the motion of a body that considers the object without taking into account the dynamics causing the movement. This branch of robotics accounts for the study of the position and its higher order derivatives[1] (velocity, acceleration, jerk etc). The key for a clear and successful analysis is the correct description of the system in terms of the geometry and its relative motion. The links are numbered starting from the base of the arm, which is fixed and is numbered as *link 0*. The first moving link is *link 1*, and so on, until the last link, which is *link n*.

Usually, at least 6 joints are needed in order to have a complete description of an object in space (corresponding to 6 degrees of freedom). Some robot might have more than 6 DOF, and they are usually referred to as "redundant".
Each link presents several characteristics that need to be considered during the design process, but as long as kinematics is concerned, we only need information about the relationship between the two neighboring joint axes. The links will be treated as rigid bodies.

[1] Taken with respect to time or other variables.

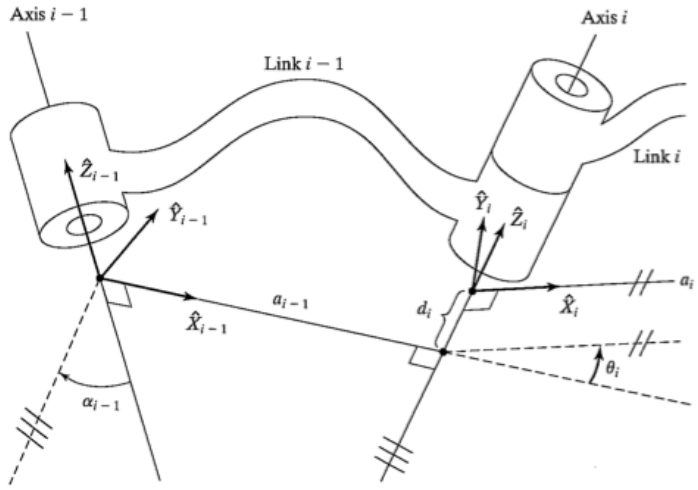


Figure 2.1: Link and joint notation schematic

2.2 Denavit-Hartenberg convention

In order to describe the manipulator and to accomplish not only the kinematic analysis, but also all the further studies, it is necessary to implement a solid and recursive notation. Thus, this convention will be used through the course of the thesis.

The Denavit-Hartenberg convention defines the relative position and orientation of two consecutive links. The problem is to determine the reference frames attached to each link and to compute (possibly with the aid of a standard recursive technique), the coordinate transformations among them. Even if the frames may be arbitrarily chosen, the DH method proposes a series of rules for the definition of these frames. These will be presented with the aid of Fig 2.1.

Referring to Fig 2.1, *joint i-1* and i can be seen. The link in between is here numbered *i-1*. In the literature, the above-mentioned convention is a standard; what might lead to uncertainties and mistakes is the definition of the link frames, which does not appear to be standardized among the main robotics textbooks [4], [8], [23]. Before enumerating the recursive steps, it is mandatory to define some key parameters [8]:

1. The mutual intersection line (a_{i-1}): in spatial geometry, there always exists a well defined distance between two non-parallel lines (let's call them *line 1* and *line 2*). This distance is measured along an axis which is mutually orthogonal to these two lines. This distance can be also interpreted as the radius of a cylinder whose axis is *line 1* and that touches *line 2* (o vice versa). When the two lines are parallel, then the mutual intersection line is not unique, that is, there are infinite parallel lines that satisfy the orthogonality condition.

2. The link twist (α_{i-1}): if we consider the plane generated by axis *i-1* and axis i, then the link twist is the angle between them, measured on this plane, from axis *i-1* to axis i, in the right hand sense around the mutual intersection line.

It can be shown that only these two parameters are necessary to fully describe the relative position between two lines in 3D space. However, we are also interested in how these links are interconnected: two extra parameters can be introduced:

3. The link offset (d_i): this parameter accounts for the distance, along the common axis of two adjoining links, between one link and the next. Referring to Fig 2.1, this offset is the distance between a_{i-1} and a_i, measured along the i axis. Notice that this distance is a signed number: positive if pointing to the positive Z_i axis.

4. The joint angle (θ_i): this parameter describes the amount of rotation about the common axis i, between one link and his neighbor. Referring to Fig 2.1, it is the angle from X_{i-1} to X_i, measured around Z_i.

By knowing these 4 parameters for each link of the robotic chain, it is possible to univocally identify the configuration. The link frames are obtained by following these steps [8]:

- The links are numbered starting from the base, which gets number 0; the first moving body is called link *1*, and the first joint is joint *1*

- The frame attached to link *i-1* will have X_{i-1}, Y_{i-1}, Z_{i-1} axis and Z_{i-1} will be chosen along the axis of joint *i-1*

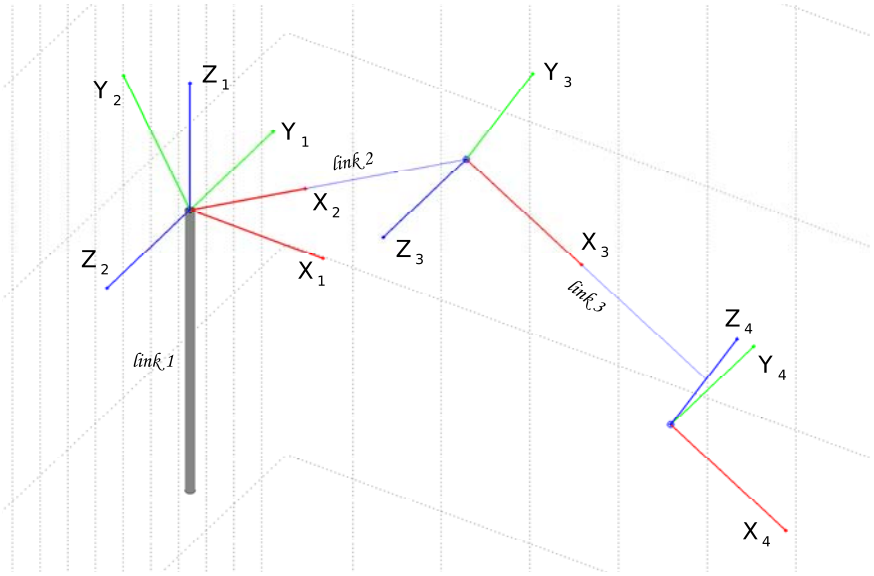


Figure 2.2: Frame configuration obtained via DH procedure.

- Identify the joint axes and the common perpendicular between them. At the point of intersection, assign the link-frame origin
- Assign Z_i axis, which will point along the *i-th* joint axis of rotation
- Assign X_i axis, which will point along the common perpendicular. (In the case of intersecting axes, X_i is chosen so that it's normal to the plane originated by the two axes)
- Assign Y_i by following the right-hand rule
- Chose the first frame so that it matches frame 0 (or base frame) when the first joint variable is zero

The first three frames can now be plotted for a random configuration of the first three links (Fig 2.2)

As far as the end effector is concerned, the frames will have the same origin, and they are oriented as shown in Fig 2.3.

Once all the frames are defined, their characteristic parameters can be stored in a matrix (also called "DH matrix"). In this case, the table is:

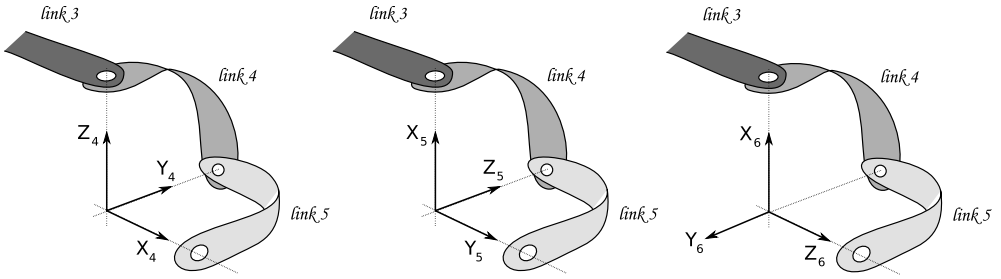


Figure 2.3: Frame configuration for end-effector structure.

Joint (i)	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$\frac{\pi}{2}$	0	-d_2	θ_2
3	0	l_2	-d_3	θ_3
4	-$\frac{\pi}{2}$	l_3	-d_4	θ_4
5	-$\frac{\pi}{2}$	0	0	θ_5
6	-$\frac{\pi}{2}$	0	0	θ_6

Table 2.1: DH matrix containing the parameters for the frame definition.

It's important to note that the table is not a function of the arm configuration, that is, it will not change with the variation of the joint coordinates, but it's a mechanical characteristic of the robot.

A comment should be made upon the d_i column. These values, called offsets, have an important influence on the kinematics calculations. These, in fact, when they are not equal to zero, augment the configuration choices: if we assign a point in space to be reached by the robot, a $d_i \neq 0$ will double the possible $\mathbf{q}$ combinations that can be used. This redundancy has advantages and disadvantages: it might be useful when there's the need to avoid obstacles or singularity conditions (by choosing another arm displacement), but, on the other hand, requires a software that implements the ability to choose among the different options. This, often, might lead to delays in the solution time.

At this point, we defined the frames for the whole system, and we know their orientation with respect to the links. We now wish to define the matrices

that allow transforming from *frame i* to *frame i+1*. This process can be done manually by looking at the DH table and building the transform considering the translation and the rotation between the links.

A way to automate this calculation is to use a recursive process. One of the most common techniques is to evaluate the following matrix, which is DH-parameters dependent.

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

In this case, subtituting the values obtained in Table 2.1, the matrices are:

$$ {}^{0}_{1}\mathbf{T} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}_{2}\mathbf{T} = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & d_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}\mathbf{T} = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & l_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{3}_{4}\mathbf{T} = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & l_3 \\ 0 & 0 & 1 & -d_4 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}\mathbf{T} = \begin{bmatrix} c(\theta_5 - \frac{\pi}{2}) & -s(\theta_5 - \frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s(\theta_5 - \frac{\pi}{2}) & -c(\theta_5 - \frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{5}_{6}\mathbf{T} = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

As we expected, the transforms are a function of the joint variables only. Note that the last two transform-matrices present the same 4^{th} column: this means that the translation with respect to the previous frame is zero, and there is only a rotational transform. This is due to the fact that the same origin was chosen for these frames (Pieper's hypothesis).

2.3 Direct Kinematics

With these matrices computed, we can introduce the direct kinematics problem (DK). Direct kinematics allows for the knowledge of the cartesian position of each link of a kinematic chain once the joint variables $\mathbf{q} = [q_1 \dots q_n]$ are know.

In a manipulator, the most important result that the DK procedure provides is certainly the knowledge of the Cartesian position and orientation of the end effector.

This is done by simply taking the product of the transforms:

$$ {}^{0}_{N}\mathbf{T}(\mathbf{q}) = {}^{0}_{1}\mathbf{T}(q_1)\, {}^{1}_{2}\mathbf{T}(q_2)\ \dots {}^{N-1}_{N}\mathbf{T}(q_N) \tag{2.2} $$

If the cartesian position is needed (in terms of $[p_x, p_y, p_z]$), we recall the general expression of a rototransitional matrix:

$$ {}^{0}_{N}\mathbf{T}(\mathbf{q}) = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & {}^{0}p_{x,N} \\ r_{21} & r_{22} & r_{23} & {}^{0}p_{y,N} \\ r_{31} & r_{32} & r_{33} & {}^{0}p_{z,N} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.3} $$

In this fashion, it is possible to instantly know the position of each joint in the Cartesian space:

$$ {}^{0}\mathbf{p_j} = {}^{0}_{j}\mathbf{T}(1:3, 4) \tag{2.4} $$

Where ${}^{0}\mathbf{p_j}$ is the position of the j-th joint with respect to the origin. The orientation can be obtained in a similar way from from Eq 2.3:

$$ {}^{0}_{j}\mathbf{R}(\mathbf{q}) = {}^{0}_{j}\mathbf{T}(1:3, 1:3) \tag{2.5} $$

2.4 Inverse Kinematics

Inverse kinematics (IK) consists in the solution of the Cartesian-to-joint variables problem. That is, for a given end effector position in 3D space, we want to know the joint variable vector(s) that allows that configuration.

The solution to this problem is way less straightforward than the direct kinematics case, and it is strictly linked to the geometrical configuration of

the manipulator. Not for all cases, in fact, there exists an analytical solution; moreover, for those cases whose analytical solution is available, this is usually difficult and time consuming, since it involves *nonlinear* equations. These might have multiple or even infinite solutions. In some cases, in view of the manipulator kinematics, there might even be no admissible solutions at all.

In general, it can be said that the number of solutions depends on the degrees of freedom but also on the link parameters; for a 6 DOF robot, there might be up to 16 different solutions for a point in the dexterous space[2]. The more non-zero DH parameters, the more solutions available.

The IK problem can be approached with two main strategies: numerical solutions and closed-form solutions. Due to their iterative nature, numerical solutions tend to be time consuming, and there is usually low interest in applying these techniques for kinematic calculations.

In the next sections, we will focus our attention on closed form solutions: they are divided into algebraic and geometric methods. In the first type, the given equations are manipulated into a form for which solution is known (transcendental expressions commonly arise), whereas in the latter the spatial geometry of the arm is decomposed into several plane-geometry problems.

2.4.1 Pieper's solution

Some configurations might provide huge simplifications for the inverse kinmatics problem. A 6 DOF robot, for example, does not have a closed form solution in general. However, if three consecutive axes intersect at a point, then Pieper's solution can be applied [8], [23].

In this thesis, the manipulator has the last three axes intersecting: the origins of frame ${}^{3}_{4}\mathbf{T}$, ${}^{4}_{5}\mathbf{T}$, ${}^{5}_{6}\mathbf{T}$ in fact, are coincident. The merging point can be calculated in base coordinates as:

[2]The *dexterous* space is defined as that volume of space that the robot end-effector can reach with all orientations. The *reachable* workspace, on the other hand, is the volume that the robot can reach in at least one orientation [8].

$$ {}^{0}\mathbf{p_4} = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,{}^{2}_{3}\mathbf{T}\;{}^{3}\mathbf{p_4} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{2.6} $$

from which, using the fourth column of Eq 2.1:

$$ {}^{0}\mathbf{p_4} = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,{}^{2}_{3}\mathbf{T} \begin{bmatrix} a_3 \\ -d_4 s\alpha_3 \\ d_4 c\alpha_3 \\ 1 \end{bmatrix} \tag{2.7} $$

we can also state that:

$$ {}^{0}\mathbf{p_4} = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T} \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix} \tag{2.8} $$

where we defined:

$$ \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix} = {}^{2}_{3}\mathbf{T} \begin{bmatrix} a_3 \\ -d_4 s\alpha_3 \\ d_4 c\alpha_3 \\ 1 \end{bmatrix} \tag{2.9} $$

Using ${}^{2}_{3}\mathbf{T}$ from Eq 2.1, the following expressions for f can be obtained:

$$ \begin{cases} f_1 = a_3 c_3 + d_4 s\alpha_3 s\alpha_3 + a_2 \\ f_2 = a_3 c\alpha_2 s_3 - d_4 s\alpha_3 c\alpha_2 c_3 - d_4 s\alpha_2 c\alpha_3 - d_3 s\alpha_2 \\ f_3 = a_3 s\alpha_2 s_3 - d_4 s\alpha_3 s\alpha_2 c_3 + d_4 c\alpha_2 c\alpha_3 + d_3 c\alpha_2 \end{cases} \tag{2.10} $$

We define also the following parameters:

$$ \begin{cases} g_1 = c_2 f_1 - s_2 f_2 + a_1 \\ g_2 = f_1 c\alpha_1 s_2 + f_2 c\alpha_1 c_2 - f_3 s\alpha_1 - d_2 s\alpha_1 \\ g_3 = f_1 s\alpha_1 s_2 + f_2 s\alpha_1 c_2 + f_3 c\alpha_1 + d_2 c\alpha_1 \end{cases} \tag{2.11} $$

And we can write, with the aid of Eq 2.1, Eq 2.8 and Eq 2.11:

$$ {}^{0}\mathbf{p_4} = \begin{bmatrix} c_1 g_1 - s_1 g_2 \\ s_1 g_1 + c_1 g_2 \\ g_3 \\ 1 \end{bmatrix} \tag{2.12} $$

The square magnitude of ${}^{0}\mathbf{p_4}$, using Eq 2.12 and substituting Eq 2.10, is:

$$ r = f_1^2 + f_2^2 + f_3^2 + a_1^2 + d_2^2 + 2d_2 f_3 + 2a_1(c_2 f_1 - s_2 f_2) \tag{2.13} $$

We define some symplifying parameters:

$$ \begin{cases} k_1 = f_1 \\ k_2 = -f_2 \\ k_3 = f_1^2 + f_2^2 + f_3^2 + a_1^2 + d_2^2 + 2d_2 f_3 \\ k_4 = f_3 c\alpha_1 + d_2 c\alpha_1 \end{cases} \tag{2.14} $$

And we finally state:

$$ \begin{cases} r = (k_1 c_2 + k_2 s_2)2a_1 + k_3 \\ z = (k_1 s_2 - k_2 c_2)s\alpha_1 + k_4 \end{cases} \tag{2.15} $$

The utility of these steps can be appreciated by looking at Eq 2.15: the dependence on θ_1 has been eliminated and the dependence from θ_2 has become much simpler. The first step is to consider the solution for θ_3. We distinguish three cases:

1. If $a_1 = 0$, then $r = k_3$. Since k_3 is a function of θ_3 only, we can obtain a quadratic equation in $tan\frac{\theta_3}{2}$ which yields the solution for θ_3

2. If $s\alpha_1 = 0$, then $z = k_4$. We can obtain a quadratic equation and solve for θ_3

3. If $a_1 \neq 0$ and $s\alpha_1 \neq 0$, we can eliminate with an auxiliary equation s_2 and c_2, and we end up with a 4^{th} degree equation, which will be solved for θ_3

In our particular case, $a_1 = 0$ and we can compute θ_3 referring to the first bullet point. We then focus on the solution of θ_2 and θ_1: this is, however, from Eq 2.12 and Eq 2.15, pretty straightforward.

At this point, we know θ_1, θ_2 and θ_3. Since the three last axis are intersecting, it is possible to compute the remaining angles with the aid of elementary matrix transform algebra.

In this problem, we are given the wanted attitude of the end effector with reference to the base frame, which is ${}^0_6\mathbf{T}_{\mathbf{att}}$. From Pieper's solution, ${}^0_3\mathbf{T}$ can be computed:

$$ {}^0_4\mathbf{T} = {}^0_1\mathbf{T}(q_1)\,{}^1_2\mathbf{T}(q_2)\,{}^2_3\mathbf{T}(q_3) \tag{2.16} $$

The desired orientation, ${}^0_6\mathbf{T}$, differs from the actual orientation ${}^0_3\mathbf{T}$ only due to the action of the last three joints, whose contribution is described by the following matrix:

$$ {}^3_6\mathbf{T}(q_4, q_5, q_6) = {}^0_3\mathbf{T}^{-1}\,{}^0_6\mathbf{T} \tag{2.17} $$

From this matrix, the computation of the angle is pretty straightforward, and we proceed algebraically from the symbolic expression of ${}^3_6\mathbf{T}$, containing the DH parameters and trigonometric functions of q_4, q_5, q_6.

It is important to notice that this method won't produce a single solution vector $\mathbf{q} = [q_1 \ \ldots q_n]$, but due to the properties of trigonometric functions, every solution step will yield two values: each of these then must be used in the next step, will which yields four corresponding solutions and so on. For n degrees of freedom, we will have 2^n solutions. Conceptually, we obtain the tree shaped solution scheme of Fig 2.4. From the related table it can be seen that we end up with 2^n solution vectors (in this case, $n{=}3 \rightsquigarrow 2^3{=}8$).

Some of these solutions, however, are not acceptable and they need to be verified: a way to do this is to insert in matrix 2.1 each of the $\mathbf{q}$ obtained and check if the resulting matrix corresponds to the original ${}^0_6T$ (if computations were to be taken, it will be seen that the 2^n matrices differ only for some sign changes, whereas the absolute value will be the correct ones).

With the aid of a Matlab simulation, the inverse kinematics problem for the first three links of the manipulator was solved. The correct solutions (which are always half of the total mathematical solutions), are plotted in Fig 2.5 for the case in which all the offset are zero *(a)*, and when $d_2 \neq 0$ *(b)*.

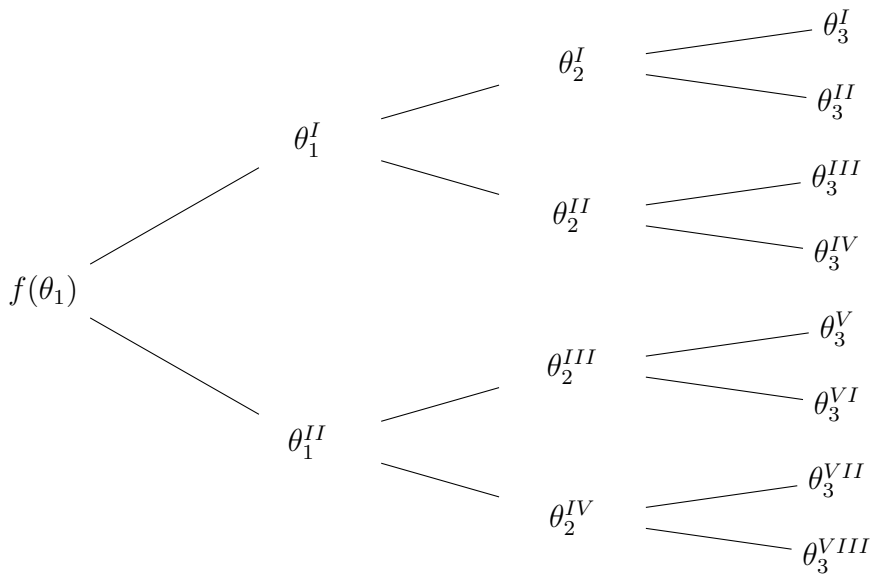


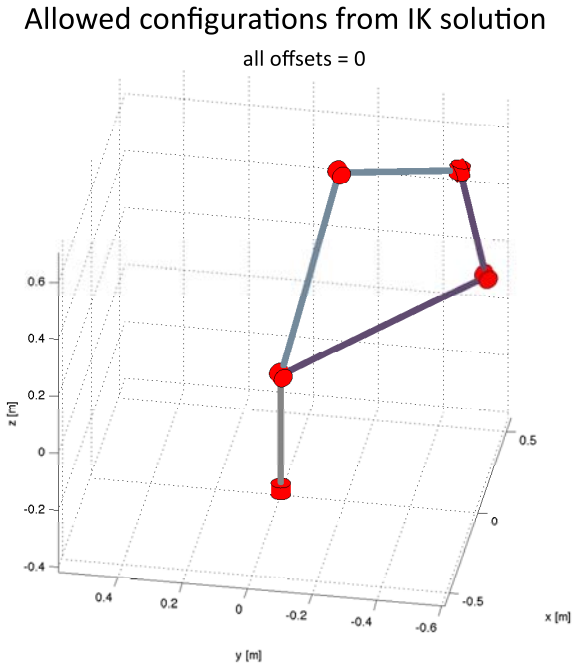
Figure 2.4: Tree diagram of solution procedure for the first 3 joints

Sol #1	θ_1^I	θ_2^I	θ_3^I
Sol #2	θ_1^I	θ_2^I	θ_3^{II}
Sol #3	θ_1^I	θ_2^{II}	θ_3^{III}
Sol #4	θ_1^I	θ_2^{II}	θ_3^{IV}
Sol #5	θ_1^{II}	θ_2^{III}	θ_3^V
Sol #6	θ_1^{II}	θ_2^{III}	θ_3^{VI}
Sol #7	θ_1^{II}	θ_2^{IV}	θ_3^{VII}
Sol #8	θ_1^{II}	θ_2^{IV}	θ_3^{VIII}

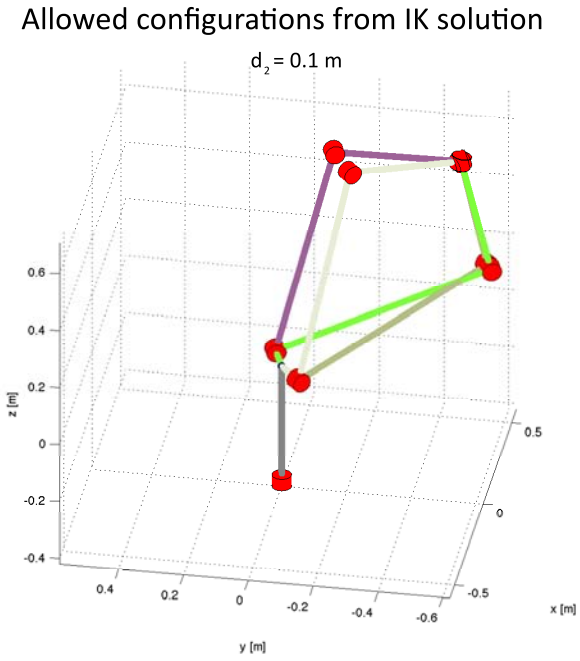
2.4.2 Alternate algebraic solution

Piper's method can be easily coded in any computing software. Some problems, however, might arise due to the need for the symbolic expression of the equations. Matlab, for example, has its own symbolic toolbox, but this stresses enormously the computing power, causing the simulation to run much slower. In general, this is not much of an issue, but when the simulation needs to provide real time solutions (i.e. inputs to actuators), the reduction of computing time is a priority.

In this section, an alternative method is presented, and it is based on



(a)



(b)

Figure 2.5: Matlab simulation of inverse kinematics problem. Case *(a)* presents no offsets ($d_i = 0 \quad \forall i$), and 2 configurations are allowed. Case *(b)* has $d_2 = 0.1$ and the possible configurations are doubled.

algebraic computations, avoiding any kind of symbolic usage. What has been changed is the calculation of the end effector *position*; the procedure for the calculation of the end effector *orientation* remains the same (Eq 2.16, Eq 2.17).

Given the end effector desired position, we can recall Eq 2.6. The ${}^{0}_{1}\mathbf{T}$, ${}^{1}_{2}\mathbf{T}$, ${}^{2}_{3}\mathbf{T}$ matrix products can be written as a function of the $\mathbf{q}$ coordinates and DH parameters (refer to Eq 2.1). From this matrix, we can extract the needed values step by step by using the inverse trigonometric functions.

Once θ_1 is obtained, we can proceed with θ_2 and θ_3. Recalling the considerations of the previous chapter, since we are solving trigonometric functions, we will have a tree of solutions that will need to be verified. Finally, the procedure for the calculation of the last two coordinates remains unchanged.

2.4.3 Methods comparison

The last method is much faster than the previous one, even after accounting for the extra solution checking time: a test for the computation time has been designed. A random point in the dexterous space of the robot was chosen, and the code for the inverse kinematics calculation was run. The test was carried out in three step: the same piece of code was run 1, 10 and 100 times in a loop[3]. This procedure was repeated identically for the two approaches. In Fig 2.6 the results from the approach explained in Section 2.4.1 are on the right, whereas the results from the technique of Section 2.4.2 are on the left.

It is immediate to notice the disadvantages of the addition of symbolic functions to a computational script: in the third test for example (red bars), the time needed in the symbolic approach is $\sim$50 times more than its numeric equivalent!

[3]For these measurements to be valuable, the number displayed in the plot is an average of 20 values obtained in 20 different tests. For sake of precision, tests were carried out with the aid of Matlab® and a 2.4 GHz Intel Core i5, 4GB RAM laptop computer.

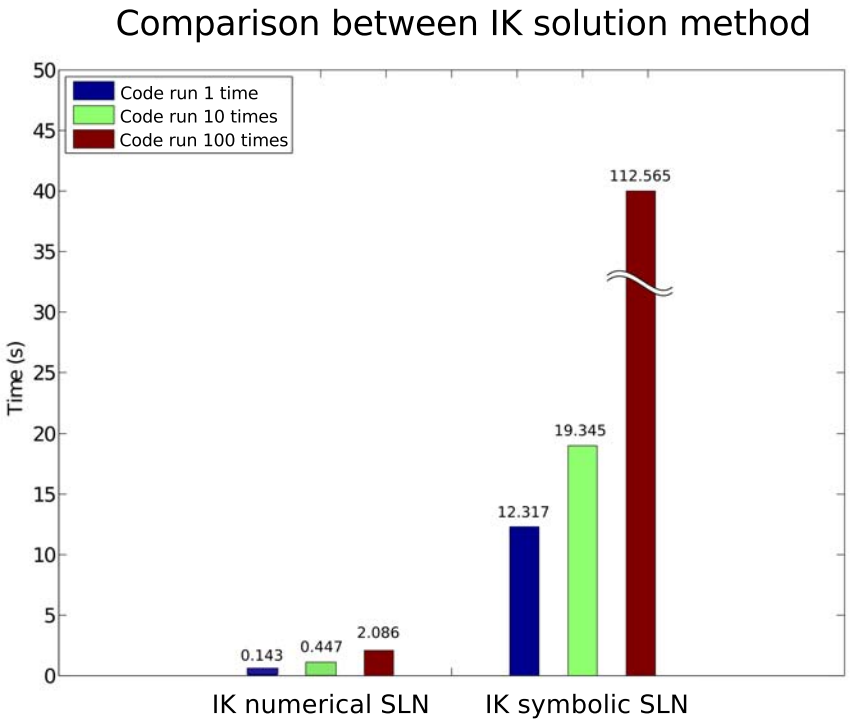


Figure 2.6: Computation time comparison between kinematic analysis approaches. On the left, the alternate albegraic solution, on the right the symbolic solution.

2.5 Differential kinematics

The previous chapter dealt with the relationship between joint variables and end effector position. Equation 2.18 illustrates the relationship between joint velocities and end effector velocities (linear and angular).

$$\mathbf{v} = \mathbf{J}(\bar{q}) \cdot \dot{\mathbf{q}} \tag{2.18}$$

The link between these parameters is provided by the Jacobian matrix. With the knowledge of this matrix and the end effector desired trajectory (expressed in terms of velocities), the kinematic problem can be easily solved: joint velocities can be directly obtained and then, with a numerical integration, also their instant position.

The Jacobian, as can be inferred, represents a fundamental quantity in robotics, and it will be used further in the dynamics analysis. Its calculation can be done in two ways: analytically or geometrically [20]. The analytical computation can be used when the end effector has a minimal representation in the operational space, and it's thus possible to compute the matrix via differentiation of the kinematics equations. The geometric computation, on the other hand, is done by computing the contributions of each joint velocity to the components of the end-effector cartesian linear and angular velocities.

These two different approaches will provide, obviously, the same result, but they have indeed different mapping matrices. In this thesis the geometric approach has been used, due to its recursive fashion that is perfectly suitable for a Matlab routine.

2.5.1 Geometric approach

From solid mechanics, we recall that the velocity of point P belonging to a rigid body moving in 3D space, with respect to frame A, can by expressed as [23], [32]:

$$^{A}V_P = {}^{A}V_B + {}^{A}_{B}R\,{}^{B}V_P \tag{2.19}$$

Where B is a reference matrix fixed to the body. In this case we consider the motion of frame B as a pure translation. If a rotation is present, Eq 2.21 becomes:

$$^{A}V_P = {}^{A}V_B + {}^{A}_{B}R\,{}^{B}V_P + {}^{A}\Omega_B \times {}^{A}_{B}\,{}^{B}P \tag{2.20}$$

Where ${}^{A}\Omega_B$ is the angular velocity of the body with respect to frame A. By using this equation and its derivatives we can approach the Jacobian matrix derivation as well as the dynamics, presented in Chapter 4. For the solution of differential kinematics, the velocities of each link (linear and angular) are needed: a technique called "velocity propagation" will be used in order to obtain a recursive and implementable sequence. We start from the base: *frame 0* will be considered the fixed, reference frame. We define v_i as the linear velocity of the origin of the frame attached to link i; same notation applies to w_i. The superscript on the left of a parameter resprents the frame in which it is expressed.

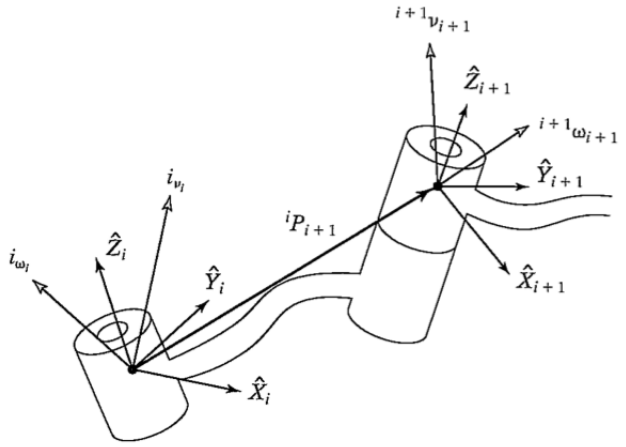


Figure 2.7: Velocity vectors for two adjoining links.

Once the notation issues are cleared, the procedure can be started: since a manipulator is a chain of links, each capable of motion relative to its neighbors, we compute the velocities in order, from the base to the end effector. Referring to Fig. 2.7, velocity of *link i+1* will be the one of *link i* with the addition of a contribution from *joint i+1*. That is:

$$^{i}v_{i+1} = {}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1} \tag{2.21}$$

Where $^{i}P_{i+1}$ is the vector connecting the two links. There is no need to calculate this, since the ${}^{i}_{i+1}\mathbf{T}$ matrices have this information stored in their fourth column. Since we want the velocity of *link i+1* to be expressed in *frame i+1*, we might rearrange the previous equation:

$$^{i+1}v_{i+1} = {}^{i+1}_{i}R\,({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) \tag{2.22}$$

As far as concerns the rotational velocities, we need to note that their addition can take place only when they are expressed in the same reference frame. We can write:

$$^{i}w_{i+1} = {}^{i}w_i + {}^{i}_{i+1}R\,\dot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} \tag{2.23}$$

Where:

$$\dot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} = {}^{i+1}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \tag{2.24}$$

The same equation expressed in *frame i+1* becomes:

$${}^{i+1}w_{i+1} = {}^{i+1}_{i}R\,{}^{i}w_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} \tag{2.25}$$

We can finally write the system of equations that will be used to "propagate" the velocities from $i = 0$, the base frame, to $i = N$, which corresponds to the velocities (linear and angular) of the end effector.

$$\begin{cases} {}^{i+1}v_{i+1} = {}^{i+1}_{i}R\,({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) \\ \\ {}^{i+1}w_{i+1} = {}^{i+1}_{i}R\,{}^{i}w_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{k_{i+1}} \end{cases} \tag{2.26}$$

Let's go back to Eq. 2.18. For a 6 DOF manipulator, it can be rewritten as follows:

$$\begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} = \begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix} \cdot \begin{bmatrix} \dot{\mathbf{q}} \end{bmatrix} \tag{2.27}$$

Where $\mathbf{J}_P$ and $\mathbf{J}_O$ are both 3×6 matrices. The derivation of the Jacobian can be accomplished with several methods. For example, we could differentiate the kinematic equations of the structure. However, looking forward to the dynamics analysis, we will use a technique that will be fundamental to simplify the dynamic analysis. The Jacobian can be written as:

$$\begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix} = \begin{bmatrix} \mathbf{z}_i \times (\mathbf{p}_e - \mathbf{p}_i) \\ \mathbf{z}_i \end{bmatrix} \tag{2.28}$$

Which, in our case, becomes:

$$\begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \times \tilde{\mathbf{p}}_\mathbf{1} & \mathbf{z}_2 \times \tilde{\mathbf{p}}_\mathbf{2} & \mathbf{z}_3 \times \tilde{\mathbf{p}}_\mathbf{3} & \mathbf{z}_4 \times \tilde{\mathbf{p}}_\mathbf{4} & \mathbf{z}_5 \times \tilde{\mathbf{p}}_\mathbf{5} & \mathbf{z}_6 \times \tilde{\mathbf{p}}_\mathbf{6} \\ \mathbf{z}_1 & \mathbf{z}_2 & \mathbf{z}_3 & \mathbf{z}_4 & \mathbf{z}_5 & \mathbf{z}_6 \end{bmatrix} \tag{2.29}$$

Where $\tilde{\mathbf{p}}_\mathbf{i} = \mathbf{p}_e - \mathbf{p}_i$. The elements of the matrix are explained in the following bullet point list [23]:

- $\mathbf{z}_i$ respresents the axis of rotation of *joint i* expressed in the base frame. It is obtained from the third column of the rotation matrix ${}^{0}_{i}\mathbf{R}$:

$$\mathbf{z}_i = {}^{0}_{1}\mathbf{R}\,{}^{1}_{2}\mathbf{R}\,...{}^{i-1}_{i}\mathbf{R}\,\mathbf{z}_0 \tag{2.30}$$

  with $\mathbf{z}_0 = [0\ 0\ 1]^T$ being used to extract the third column of ${}^{0}_{i}\mathbf{R}$.

- $\mathbf{p}_e$ represents the position of the end effector expressed in the base frame. It is a 3×1 vector that can be extracted from $\check{\mathbf{p}}_e$, where $\check{\mathbf{p}}_e$ is calculated from:

$$\check{\mathbf{p}}_e = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,...{}^{N-1}_{N}\mathbf{T}\,\check{\mathbf{p}}_0 \tag{2.31}$$

  in this case, $\check{\mathbf{p}}_0 = [0\ 0\ 0\ 1]^T$ allows for the extraction of the fourth column; clearly, $\mathbf{p}_e$ is given by the first three elements of $\check{\mathbf{p}}_e$.

- $\mathbf{p}_i$ represents the position of the *i-th* joint expressed in the base frame. It is a 3×1 vector that can be extracted from $\check{\mathbf{p}}_i$, where $\check{\mathbf{p}}_i$ is calculated from:

$$\check{\mathbf{p}}_i = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,...{}^{i-1}_{i}\mathbf{T}\,\check{\mathbf{p}}_0 \tag{2.32}$$

Equations 2.29 can be easily implemented in a Matlab code; the value of $\mathbf{J}$ depends on the istantaneous configuration (its symbolic expression is available in the Appendix).

2.5.2 Inverse differential kinematics

As above mentioned, the differential kinematics equation represents a linear mapping between the operational space and the joint space. This fact suggests the possibility to utilize this approach to tackle the inverse kinematics problem. Recalling Eq. 2.18:

$$\mathbf{v} = \mathbf{J}(\bar{q}) \cdot \dot{\mathbf{q}} \tag{2.33}$$

What is interesting from a robotics point of view, is the $\dot{\mathbf{q}}$ vector: this contains information about the motion in the joint space, which are needed when controlling a robot. In order to extract this vector, we can invert the equation:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\bar{q}) \cdot \mathbf{v} \tag{2.34}$$

From this vector, since $\mathbf{v}$ is known from the trajectory planning, we can finally obtain the joint variable position using an integration:

$$\mathbf{q}(t) = \int_0^T \mathbf{q}(t)\, dt \; + \; \mathbf{q}(0) \tag{2.35}$$

The initial position $\mathbf{q}(t = 0)$ needs to be known in order to start the integration. This value can be obtained, for example, with one of the two IK methods presented in Section 2.4.1 and 2.4.2.

Eq 2.35 has to be implemented in the code has a discrete linear expression, using one of the several numerical integration methods available (Euler, Heun, etc.); in this case, Euler method has been implemented: given an integration step, the position at time t_{i+1} is given by:

$$\mathbf{q}(t_{i+1}) = \mathbf{q}(t_i) + \dot{\mathbf{q}}(t_i)\, \Delta t \, \mathbf{q}(t_{i+1}) \, \mathbf{q} = [\, 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \,] \tag{2.36}$$

At this point, it is possible to summarize the procedure with a block diagram (Fig 2.8). If we insert the integration method, then the solution procedure can be represented by the blocks in Fig 2.9. In the diagram, Euler's integration method is implemented.

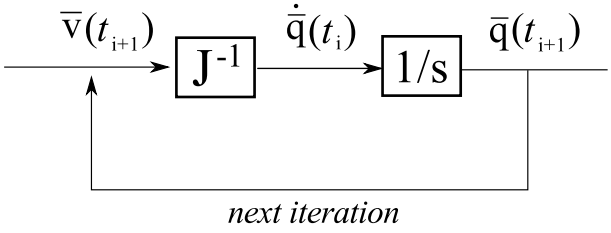


Figure 2.8: Inverse differential kinematics diagram.

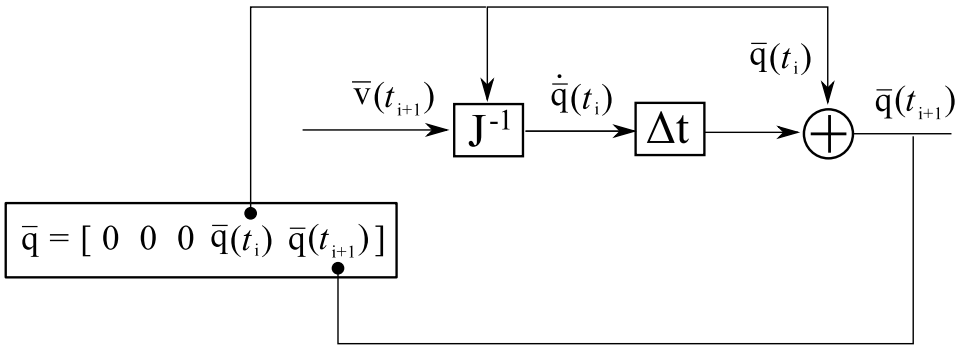


Figure 2.9: Inverse differential kinematics diagram with integration method.

2.5.3 Singularities

Without doubt, the most critical step in obtaining the joint variable values lies in Eq. 2.34. This step, in fact, involves a matrix inversion, which can be successfully carried out only if the determinant is different from zero. When $det(\mathbf{J})=0$, one or more singularities are present: a singularity represents a configuration in which the mobility of the robot is reduced. They can be classified into [23], [8]:

- *Boundary singularities:* these occour when the robot is fully stretched and the desired trajectory is outside of the reachable envelope surface. This singularity can be easily avoided when designing a code: a simple control can be put on theses points of the trajectory.

- *Internal singularities:* these take place inside the workspace when one or more degrees of freedom are lost, that is, when there's a lining up of two or more joint axes. The most critical part of the robot in terms of singularities is the end effector. For this reason, in this paper its design has been carefully planned in order to limit these situations. Refer to Section 1.3.3 for an accurate discussion.

Singularities constitute a serious issue, due to the fact that when $det(\mathbf{J})=0$, the $\dot{\mathbf{q}}$ computations will provide infinite results, which are clearly not acceptable: if the robot is given these results as an input to the motors, then serious trouble may arise.

A way to solve (or, at least, to prevent) this phenomenon is to solve the $det(\mathbf{J})$=0 equation and spot the dangerous cases. It can be seen, unfortunately, that this process is difficult and of no easy solution for generic structures.

For robots with a spherical wrist (our case) it's however possible to split the computation in two parts:

1. Calculation of arm singularities due to the motion of the first links (3 in this case)
2. Calculation of wrist singularities due to the wrist joints (3 in this case)

We can think of the Jacobian as composed by four 3×3 sub-matrices:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_{11}} & \mathbf{J_{12}} \\ \mathbf{J_{21}} & \mathbf{J_{22}} \end{bmatrix} \tag{2.37}$$

Recalling Eq. 2.29, we have that:

$$\mathbf{J_{12}} = \begin{bmatrix} \mathbf{z}_4 \times \tilde{\mathbf{p}}_\mathbf{4} & \mathbf{z}_5 \times \tilde{\mathbf{p}}_\mathbf{5} & \mathbf{z}_6 \times \tilde{\mathbf{p}}_\mathbf{6} \end{bmatrix} \tag{2.38}$$

$$\mathbf{J_{22}} = \begin{bmatrix} \mathbf{z}_4 & \mathbf{z}_5 & \mathbf{z}_6 \end{bmatrix} \tag{2.39}$$

However, since we used Pieper's mechanical simplification, the last three joint axes are intersecting, which yields $\mathbf{J_{12}} = [0 \;\; 0 \;\; 0]$. This means that the overall Jacobian has the structure of a block lower-triangular matrix. Due to this property, the determinant calculation is simplified:

$$det(\mathbf{J}) = det(\mathbf{J_{11}}) det(\mathbf{J_{22}}) \tag{2.40}$$

This form also allows for the immediate decoupling of the two singularity cases: $det(\mathbf{J_{11}})$=0, in fact, accounts for *arm singularities*, whereas $det(\mathbf{J_{22}})$=0 accounts for *wrist singularities*.

As far as concerns *arm singularities*, the determinant is[4]:

$$det(\mathbf{J_{11}}) = -a_2 a_3 c_3 (a_2 c_2 + a_3 c_{23}) \tag{2.41}$$

This equation has two solutions:

[4]The notations $s_{i\ldots j}$, $c_{i\ldots j}$ denote respectively $sin(q_i + \ldots + q_j)$, $cos(q_i + \ldots + q_j)$.

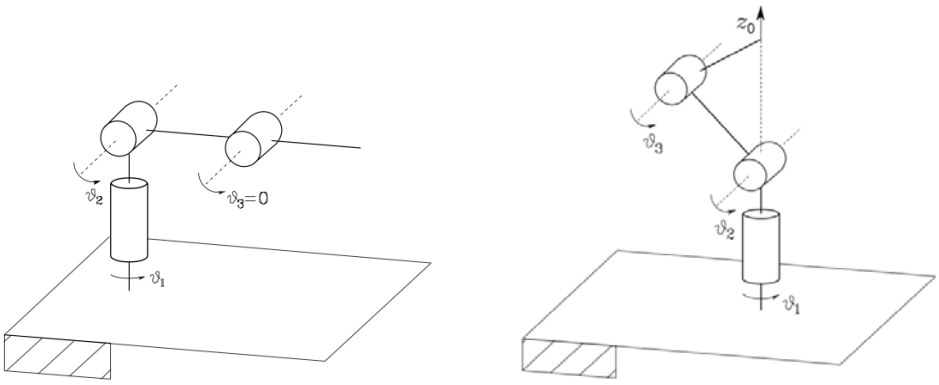


Figure 2.10: Arm singularities: *elbow* and *shoulder*.

1. $sin(\theta_3)$=0 $\rightsquigarrow$ $\boxed{\theta_3 = 0}$, $\boxed{\theta_3 = \pi}$: in this configuration, the manipulator is fully stretched or retracted along *link 2*. This is called **elbow** singularity, Fig 2.10 (a).

2. $a_2c_2 + a_3c_{23} = 0$ $\rightsquigarrow$ $\boxed{\mathrm{p}_x{=}\mathrm{p}_y{=}0}$: in this case the end effector position (that is, the origin of frames ${}^3_4\mathbf{T}$, ${}^4_5\mathbf{T}$, ${}^5_6\mathbf{T}$) lies on the z_0 axis. Singularity arises because in this configuration the variation of θ_1 has no effect on the position of the end-effector: this is called **shoulder** singularity, Fig 2.10 (b).

As far as *wrist singularities* are concerned, the solution can be inferred by taking a close look at the $\mathbf{J_{22}}$ sub-matrix. The determinant is equal to zero when the rows (or the columns) are linearly dependent. Since the rows are the unit vectors describing the orientation of the last three joints' revolution axes, the solution is fairly simple to obtain: when these axes align, the rank of matrix $\mathbf{J_{22}}$ drops from 3 to a value <3, thus providing our solution, $det(\mathbf{J_{22}})$=0.

Axes $\mathbf{z}_4$ and $\mathbf{z}_5$ do not align, no matter what value is given to θ_4. Same thing applies to $\mathbf{z}_5$ and $\mathbf{z}_6$, which are never parallel. The only possible alignment among $\mathbf{z}_4$, $\mathbf{z}_5$ and $\mathbf{z}_6$ occurs when axis $\mathbf{z}_6$ aligns with $\mathbf{z}_4$. This situation is influenced by θ_5 only:

$$\mathbf{z}_4 // \mathbf{z}_5 \rightsquigarrow \boxed{\theta_5 = 0}, \boxed{\theta_5 = \pi} \tag{2.42}$$

2.6 Simulation

Once all the modeling has been completed, it is possible to simulate some trial trajectories. Along with the script for computing the kinematics quantities, a graphical output allows for a better visualization of the problem.

This consists in a 3D animation of a simplified model of the robot. For each time step, a plot with the current configuration is drawn; thus, if the frame rate is adequately high, the serie of images resembles a moving object. On these figures, the predefined trajectory is drawn as well as the actual one, in order to check the correct tracking motion of the end effector. It will be shown how this variance increases dramatically with the time step.

2.6.1 Rectilinear trajectory

The first trajectory to be simulated is a line in space. According the theory presented in Chapter 3, this path needs only the starting and ending points ($\mathbf{p}_i$, $\mathbf{p}_f$) for its complete definition. As far as concerns the motion law, we suppose a 5^{th} degree polynomial with zero acceleration at the extremities.

Moreover, we need to define how the end effector orientation changes during the journey; since we have no particular requirements at this point, we impose the orientation to be coherent to a random attitude frame ($\mathbf{T}_{att}$), described using Euler angles ϕ_{att}, θ_{att}, ψ_{att} (the procedure for obtaining q_4, q_5, q_6 from this approach is explained in Section 2.4.1)

For sake of simplicity, we simulate a line parallel to the x-axis. The parameters used to initialize the code[5] are:

$$\begin{aligned} x_{in} &= [0.4 \;\; 1.1 \;\; 0.2] \quad [m] \\ v_{in} &= [0 \;\; 0 \;\; 0] \quad [m] \\ \psi_{att} &= -20^\circ \\ \theta_{att} &= 90^\circ \\ \phi_{att} &= 45^\circ \end{aligned}$$

[5] A copy of the script is available in the Appendix.

As far as concerns the trajectory, we have:

$$x(t) = \begin{Bmatrix} a_5t^5 + a_4t^4 + a_3t^3 + a_0 \\ 1.1 \\ 0.2 \end{Bmatrix} \tag{2.44}$$

$$\dot{x}(t) = \begin{Bmatrix} 5a_5t^4 + 4a_4t^3 + 3a_3t^2 \\ 0 \\ 0 \end{Bmatrix} \tag{2.45}$$

Where:

$$\begin{aligned} a_0 &= -0.4 \\ a_1 &= 0 \\ a_2 &= 0 \\ a_3 &= -0.008 \\ a_4 &= 0.0012 \\ a_5 &= -4.8 \cdot 10^{-5} \end{aligned}$$

$$t \in [0;\ 10]\ s$$

$$dt = 0.01\ s$$

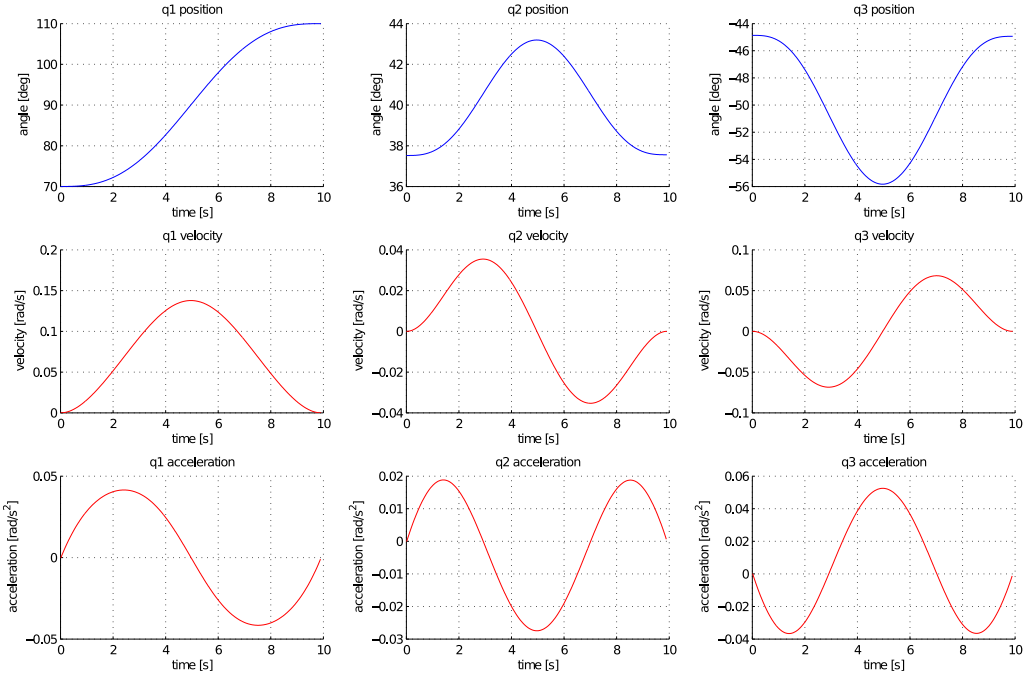
With the following boundary conditions on velocity and acceleration:

$$\dot{x}(0) = \dot{x}(T) = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

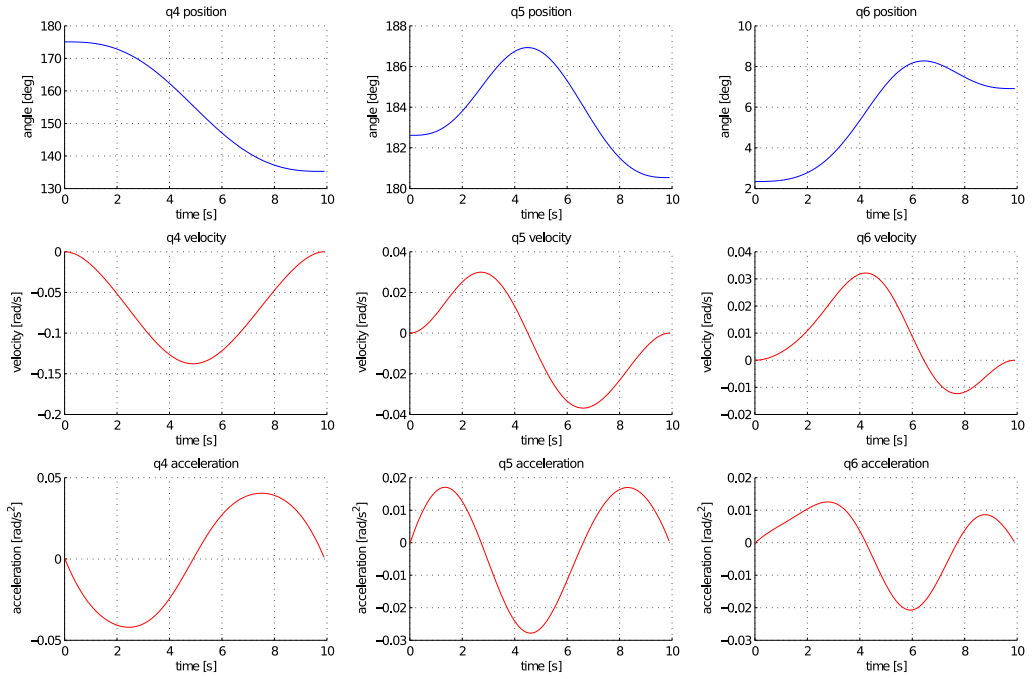
$$\ddot{x}(0) = \ddot{x}(T) = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

Notice that, in the kinematics simulations, no information about the mass and inertial properties of the links are needed. In Fig 2.11 the position, velocity and acceleration of the six joints are plotted.

The last frame of the Matlab animation is proposed in Fig 2.12 *(a)*. The end effector is represented as a concentrated mass, and its orientation is



(a) $q(\mathrm{t})$, $\dot{q}(\mathrm{t})$, $\ddot{q}(\mathrm{t})$ for *joints 1, 2, 3.*



(b) $q(\mathrm{t})$, $\dot{q}(\mathrm{t})$, $\ddot{q}(\mathrm{t})$ for *joints 4, 5, 6.*

Figure 2.11: Kinematics analysis for linear trajectory, T=10 s

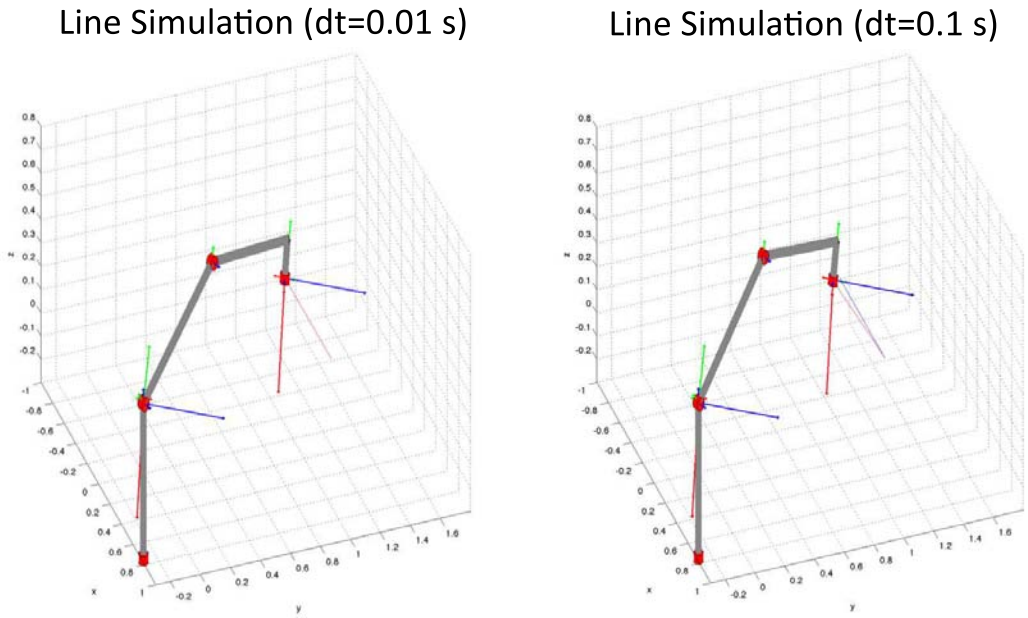


Figure 2.12: Linear trajectory: 3D simulation in Matlab's native environment. Time steps of $dt = 0.01$ s (left) and $dt = 0.1$ s (right)

expressed by the frame attached to it. To verify to correctness of the end effector orientation, the goal frame is plotted in the system's origin: it can be seen that they have the same attitude as predicted. The red line in the figure represents the actual trajectory, which follows very well the ideal path (represented with a blue segment, here hidden by the red line).

The importance of the time step choice can be seen in Fig 2.12 *(b)*: in this case, the only parameter that was changed in the simulation is the timestep, which was increased to $dt = 0.1$ s. In this picture, the divergence between the red and the blue line is visibily increasing with time.

2.6.2 Circular trajectory

For simulating a circular trajectory, we can follow the abovehead procedure. According to the analytical description in Chapter 3, a circular path is characterized by the radius, the center vector, the normal vector (i.e. the perpendicular to the circumference plane) and a starting point.

In this example, the analysis was carried out with the following parameters:

$$\begin{aligned} x_{in} = x_{fin} &= [0.9 \;\; 0.7 \;\; 0.3] \quad [m] \\ x_c &= [0.9 - R \;\; 0.7 \;\; 0.3] \quad [m] \\ \bar{\mathbf{n}} &= [0 \;\; 0 \;\; 1] \\ R &= 0.4\, m \\ \psi_{att} &= 0^\circ \\ \theta_{att} &= 45^\circ \\ \phi_{att} &= 0^\circ \end{aligned}$$

As far as the trajectory is concerned, we utilize a 5^{th} order polynomial for the angular law, that is:

$$\begin{aligned} \Theta(t) =& a_5 t^5 + a_4 t^4 + a_3 t^3 \\ \dot{\Theta}(t) =& 5a_5 t^4 + 4a_4 t^3 + 3a_3 t^2 \\ \ddot{\Theta}(t) =& 20a_5 t^3 + 12a_4 t^2 6 + a_3 t \end{aligned}$$

Where:

$$\begin{aligned} a_0 =& 0 \\ a_1 =& 0 \\ a_2 =& 0 \\ a_3 =& 62.83 \\ a_4 =& -94.25 \\ a_5 =& 37.69 \\ t \in& [0;\, 1]\; s \\ dt =& 0.001\; s \end{aligned}$$

With the following boundary conditions on velocity and acceleration:

$$\dot{x}(0) = \dot{x}(T) = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

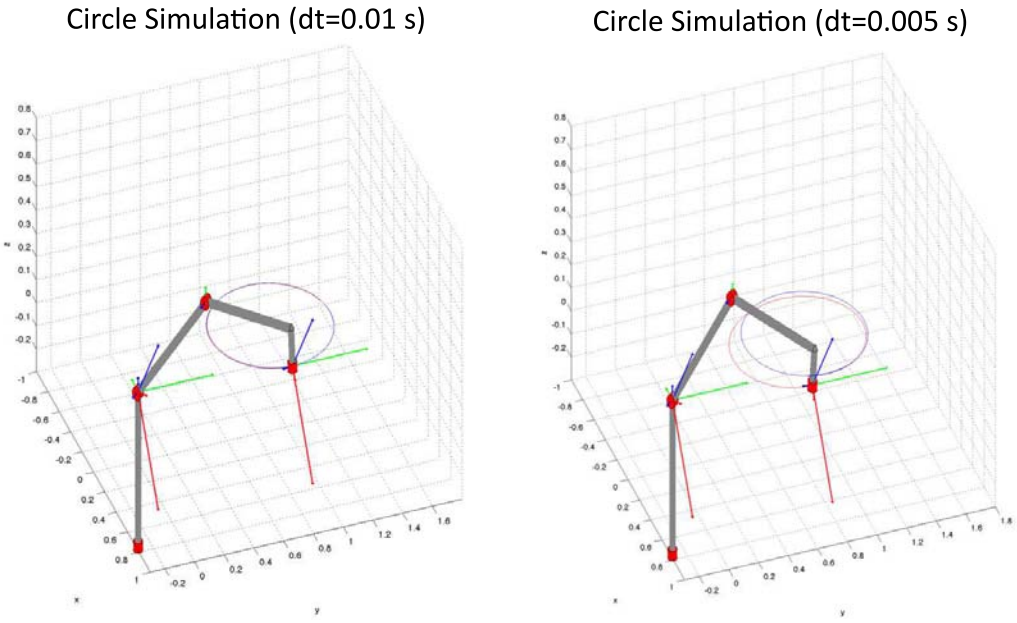


Figure 2.13: Cicular trajectory: 3D simulation in Matlab's native environment. Time steps of $dt = 0.001$ s (left) and $dt = 0.005$ s (right)

$$\ddot{x}(0) = \ddot{x}(T) = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

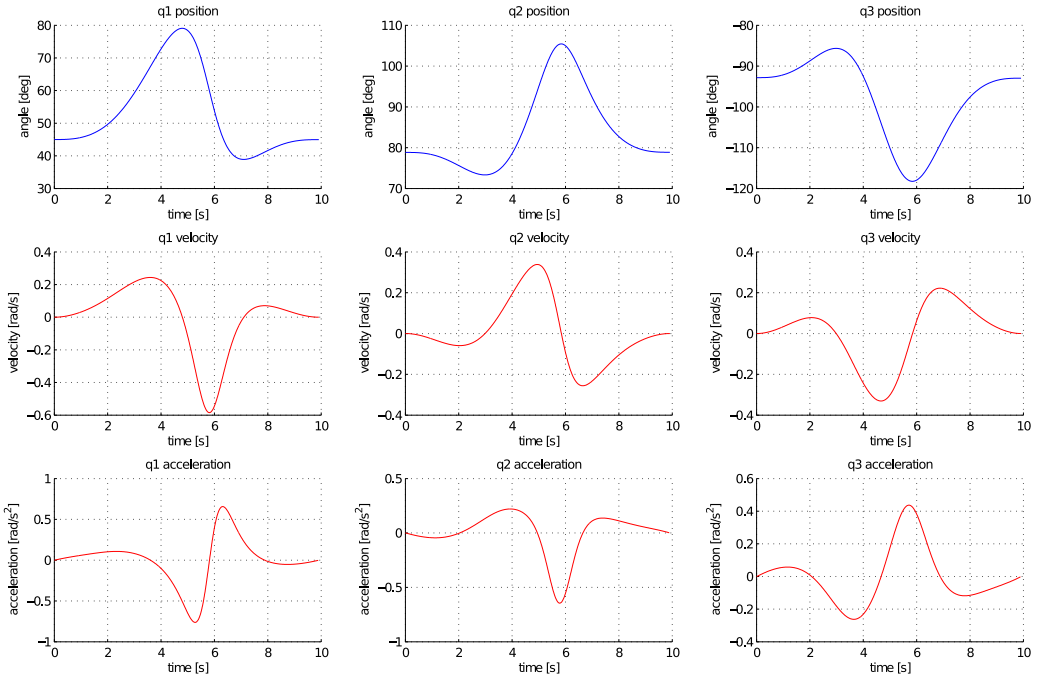
The cartesian position and velocity, then, are:

$$x(t) = \begin{Bmatrix} R\,cos\Theta(t) \\ R\,sin\Theta(t) \\ 0.3 \end{Bmatrix} = \begin{Bmatrix} 0.4\,cos(a_5t^5 + a_4t^4 + a_3t^3) \\ 0.4\,sin(a_5t^5 + a_4t^4 + a_3t^3) \\ 0.3 \end{Bmatrix} \tag{2.50}$$

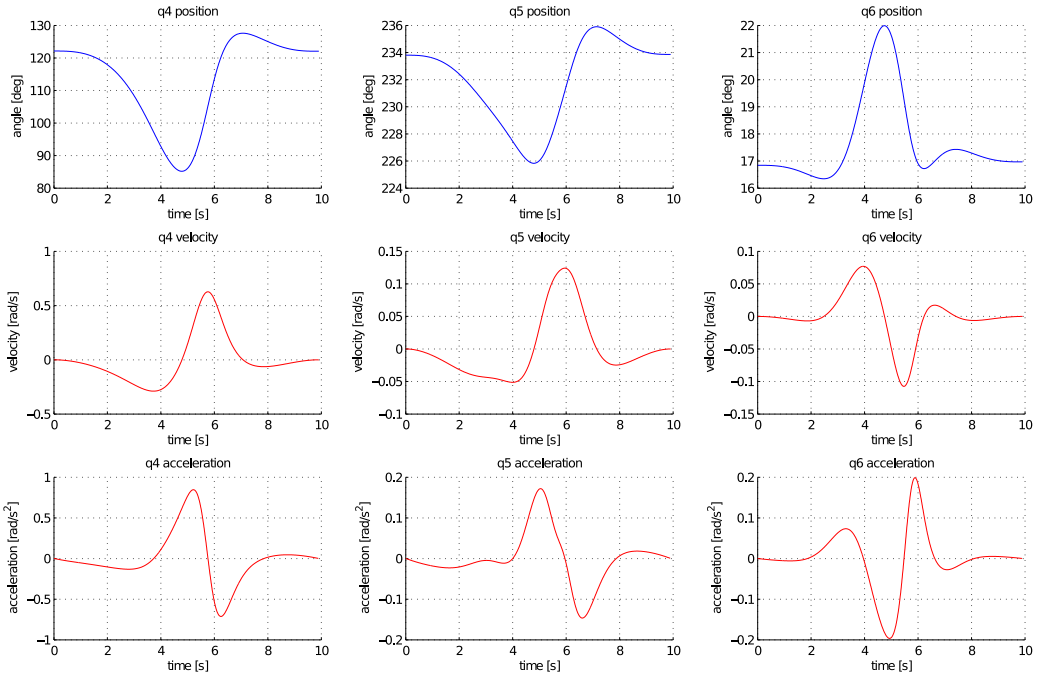
$$\dot{x}(t) = \begin{Bmatrix} -R\,sin\Theta(t) \cdot \dot{\Theta}(t) \\ R\,cos\Theta(t) \cdot \dot{\Theta}(t) \\ 0 \end{Bmatrix} = \begin{Bmatrix} -0.4\,sin(a_5t^5 + a_4t^4 + a_3t^3)(5a_5t^4 + 4a_4t^3 + 3a_3t^2) \\ 0.4\,cos(a_5t^5 + a_4t^4 + a_3t^3)(5a_5t^4 + 4a_4t^3 + 3a_3t^2) \\ 0 \end{Bmatrix} \tag{2.51}$$

In Fig 2.14 the position, velocity and acceleration of the six joints are plotted.

Finally, we propose a screenshot of the animation carried out in this case. As usual, the red trajectory represents the actual position of the end effector, whereas the blue line is the goal trajectory. To stress the importance of the



(a) $q(\mathrm{t})$, $\dot{q}(\mathrm{t})$, $\ddot{q}(\mathrm{t})$ for *joints 1, 2, 3.*



(b) $q(\mathrm{t})$, $\dot{q}(\mathrm{t})$, $\ddot{q}(\mathrm{t})$ for *joints 4, 5, 6.*

Figure 2.14: Kinematics analysis for circular trajectory, T=10 s

step size choice, two simulations are pictured. In the first one, Fig 2.13 *(a)*, step size is $dt = 0.001$; in the second one Fig 2.13 *(b)*, the value was increased to $dt = 0.005$.

Note that also in this case the end effector attitude frame was plotted. In both picture, they are clearly coherent with the predefined attitude frame.

2.7 Model verification: Simulink's *SimMechanics* toolbox.

We here introduce an alternate way of simulating a robotic structure. In the Matlab environment, it is possible to create a SimMechanics^{TM} model. This add-in provides a multibody simulation environment for 3D mechanical systems. The multibody system is modeled using blocks representing bodies, joints, constraints, and force elements; SimMechanics then formulates and solves the equations of motion for the complete mechanical system. Models from CAD systems, including mass, inertia, joint, constraint, and 3D geometry, can be imported into SimMechanics. An automatically generated 3D animation allows for the visualization the system dynamics [26].

The power of the software, in our case, is the ability to cross verify the analytical results. In fact, this can be done without writing any equation for this new model. The only things needed are the physical parameters of the robot and the input values for the motors.

The model can be easily imported from a compatible CAD software (SolidWorks was used in our case), and mass, center of mass and inertial properties are automatically computed and stored for each body part.

As far as concerns the inputs to the motors, SimMechanics joints can be controlled in two ways:

1. *Motion control:* each link is provided with the generalized coordinates $\mathbf{q_i}$ and its derivatives $\dot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_i$.

2. *Torque control:* each link is provided with the instantaneous torque τ_i.

In this chapter, we focus on motion control (refer to Chapter 4 for torque control simulation).

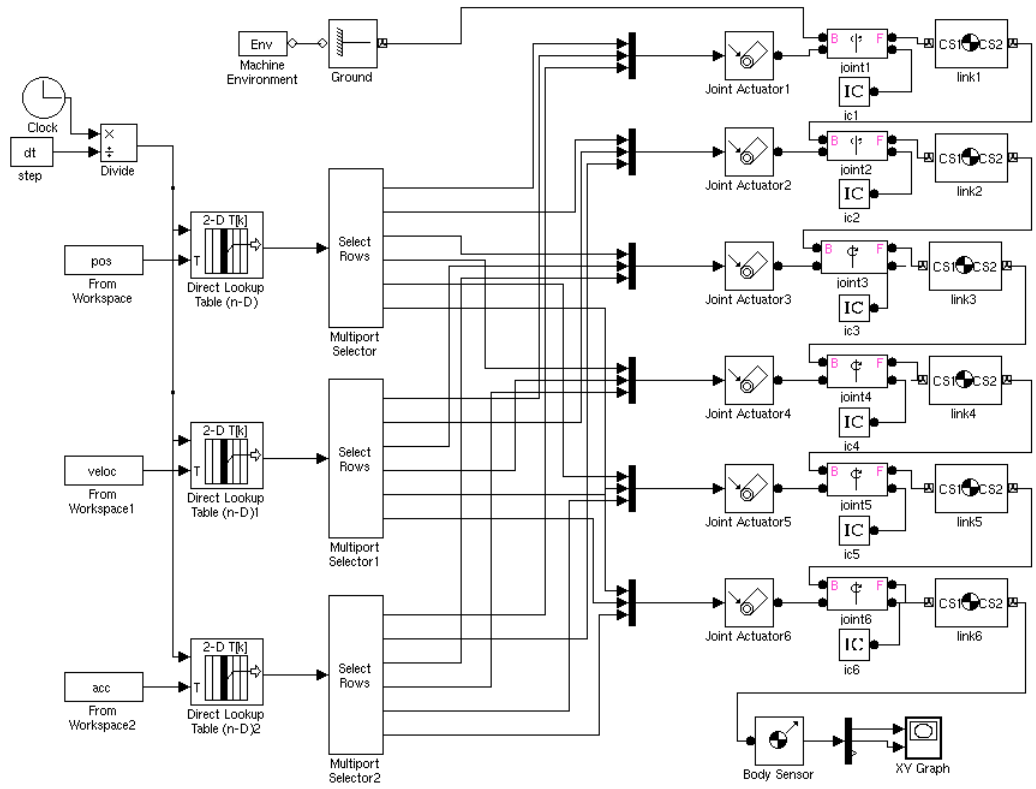


Figure 2.15: *Simulink* block diagram for trajectory analysis and simulation. In this case, the joints are motion controlled.

The motion controlled joint needs $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$ as inputs: these are calculated from the Matlab script and stored in three matrixes that will be extracted from the Simulink environment.

Moreover, SimMechanics provides several built in features for the motion monitoring. For example, virtual sensors can be attached to the joints. What was done in this case, since we are imposing a cartesian trajectory, was to monitor the end effector position with a XY plot of its instantaneous position.

As far as concerns the graphical interface, SimMechanics allows for the import of the CAD drawings of the links. It is possible to obtain very detailed animation, useful to identify intereferences or interface problems. In Fig 2.16 we present some screenshots of the graphical environment.

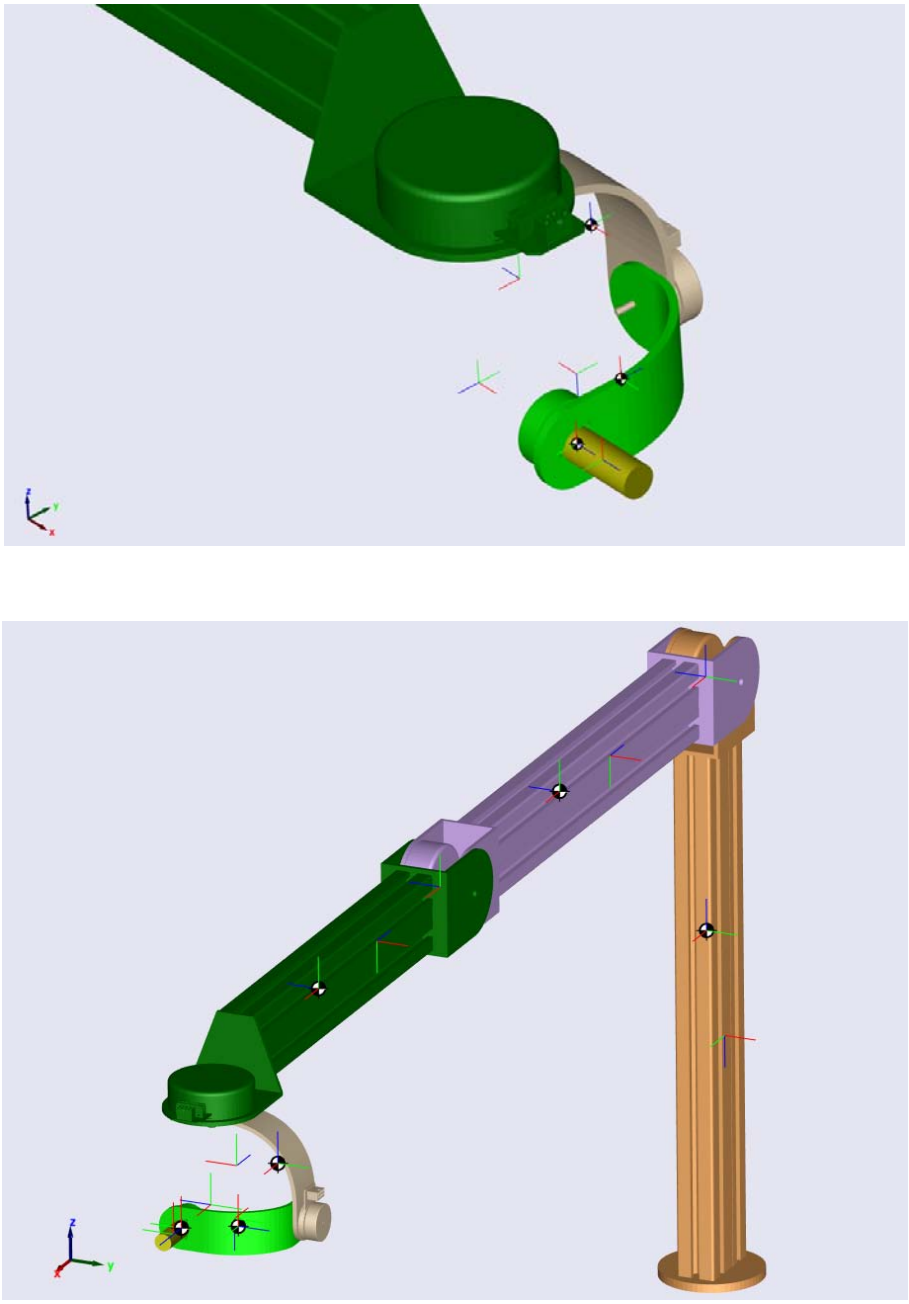
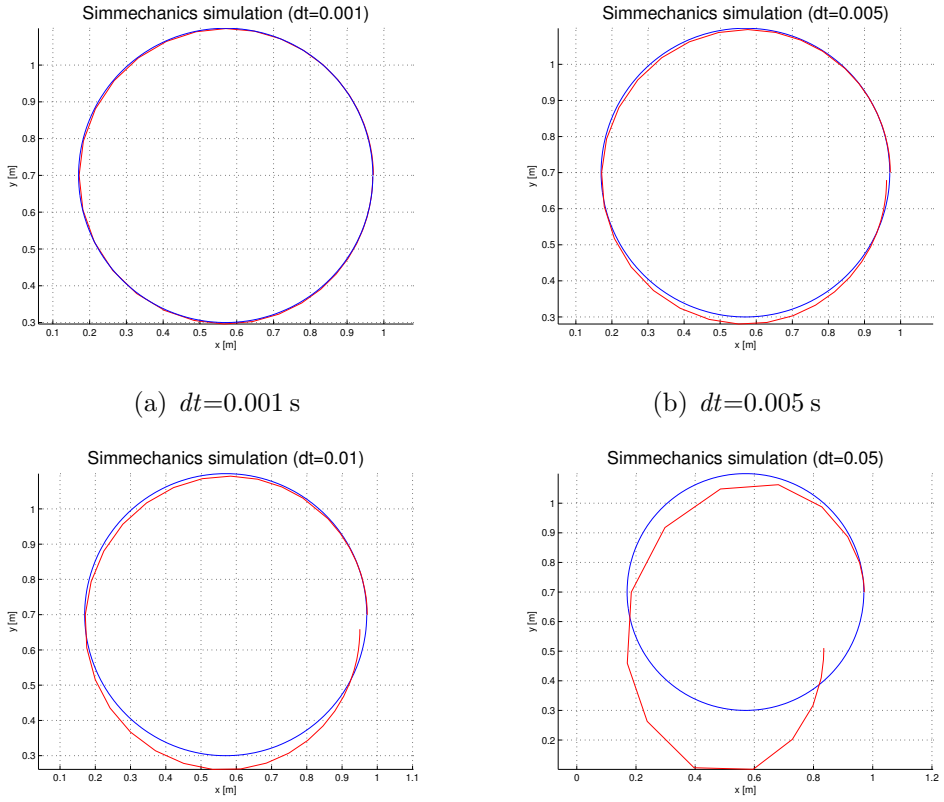


Figure 2.16: *Simmechanics* virtual model of the manipulator: overall view and *end effector* close up.

The Simulink block diagram governing the model is presented in Fig 2.15. The diagram starts with the calculation of the current iteration count by dividing the *clock* variable by the imposed stepsize (whose value is picked automatically from the Matlab model). With this value (i), the *Lookup Table* blocks extract the corresponding *i-th* column from the $6 \times N$ position, velocity and acceleration matrices stored in the workspace. This column is finally divided into its 6 components, which are then fed into the corresponding joint inputs. The solution is obtained with the aid of a variable step solver: *ode45* (Dromand-Price) has been used.

In order to get the end effector position, a *Body Sensor* block is connected to *link 6*. Its output, a 3×1 cartesian position, is then plotted with the aid of a *Scope* block. In Fig 2.17, this graphical output is presented for four different step size choices; the blue line represents the wanted trajectory.



(a) dt=0.001 s

(b) dt=0.005 s

(c) dt=0.01 s

(d) dt=0.05 s

Figure 2.17: Simulink's XY-scope output for circular trajectory. Step sizes used: 0.001 s, 0.005 s, 0.01 s, 0.05 s.

Chapter 3

Trajectory definition

3.1 Introduction

The goal of trajectory planning is to generate the inputs for the control system in order for the end effector to follow a predetermined trajectory. The trajectory is defined with a series of parameters, which depend from the task that the robot is wanted to accomplish.

Once a trajectory is defined in *cartesian space*, then the problem consists in the conversion of this path into the *joint space*. The way in which the path is fed to the motors is the core of this chapter.

Before we start, it is important to notice the difference between *path* and *trajectory*: the *path* describes the locus of points in space (joint or cartesian) that the end effector has to follow, whereas the *trajectory* contains also information on how this path is swept in terms of its derivatives (i.e. velocity and acceleration).

3.2 Main approaches

A manipulator has to be able to move in space from a starting position to an assigned end position. The way in which this task is performed needs to be tuned for the specific machine in order not to approach singularities or motor saturation. Moreover, the laws describing the motion should provide a smooth transition, avoiding vibrations, resonance or shocks.

The laws describing the trajectory of the manipulator are of continuous type. Obviously, it's not possible to give a continuous input to the motors. Usually, only several points are provided: starting and ending points and internal points, which are obtained from the discretization of the continuous laws.

What a trajectory planning algorithm does, is to provide the actuators with a series of discrete inputs; the higher the density of these points in time, the more precise the resulting motion will be. A coarse data flow, in fact, due to the non-linear terms in the direct kinematics equations, might lead to unpredicted motions. Hence, there could be a drift due to accumulated error, and the manipulator motion could diverge, resulting in damages to thing or personal.

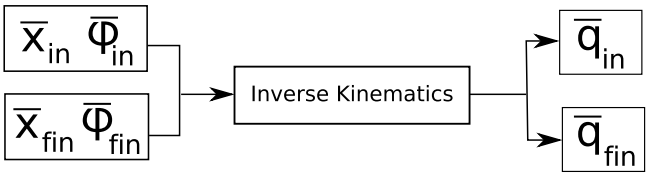
It seems clear that a fine interpolation of the path is the best solution. However, this is true as long as the computing saturation of the CPU is not reached: when this happens, the control of the robot can be lost, creating similar dangers as in the coarse interpolation case.

Though a tradeoff is clearly the best way to proceed, it is interesting to mention some minor changes that can be performed in order to shift forward the CPU limit:

- ◇ upgrading of the computing hardware
- ◇ predilection for numerical algorithms (refer to the IK example in Section 2.4.3)
- ◇ usage of machine code routines (i.e. C++ over Matlab)

3.3 Joint space planning

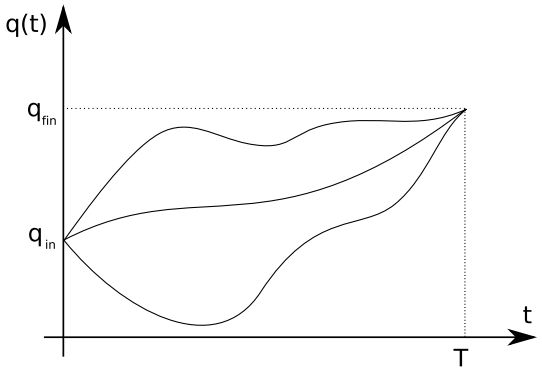
In the joint space planning, the initial and final potion ($\bar{\mathbf{x}}$) of the end effector are known, as long with its orientation ($\bar{\varphi}$). From these, by invoking an inverse kinematic algoritm, it is possible to obtain the initial and final position of the robot in the joint space, that is:



Then, an appropriate law $q(t)$ for each joint needs to be computed. There are several techniques available: the common spirit beyond each of these is to provide a smooth law. We analyze the two main cases: when the start and end points are known and when also some midpoints are given.

Point-to-point motion

In this case, the end effector has to move from a starting position to a final known position in a given time span T. Obviously, infinite curves can solve this problem:



However, we need to impose the continuity of at least the first $q(t)$ derivative. This guarantees that there are no jumps or discontinuites in the velocity profile, which could cause the Jacobian matrix to become singular.

The easiest approach for this goal is to write the generic position law as a polynomial function. The degree of the polynomial needed can be calculated as $n = C - 1$ where C is the number of boundary conditions. In this case, we have that (using $\theta(t)$ as the $q(t)$ variable):

$$\theta(0) = \theta_{in} \qquad \theta(T) = \theta_{fin} \tag{3.1}$$

And, for the velocity profile to be continuous:

$$\dot{\theta}(0) = 0 \qquad \dot{\theta}(T) = 0 \tag{3.2}$$

With 4 boundary conditions, the 3^{rd} degree polynomial can be written as:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.3}$$

And its derivatives:

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{3.4}$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \tag{3.5}$$

The coefficients are easily computed. As an example, the trajectory laws for $\theta(t_0) = \pi,\ \theta(T) = 2\pi,\ T = 10s$ are plotted in the following figure:

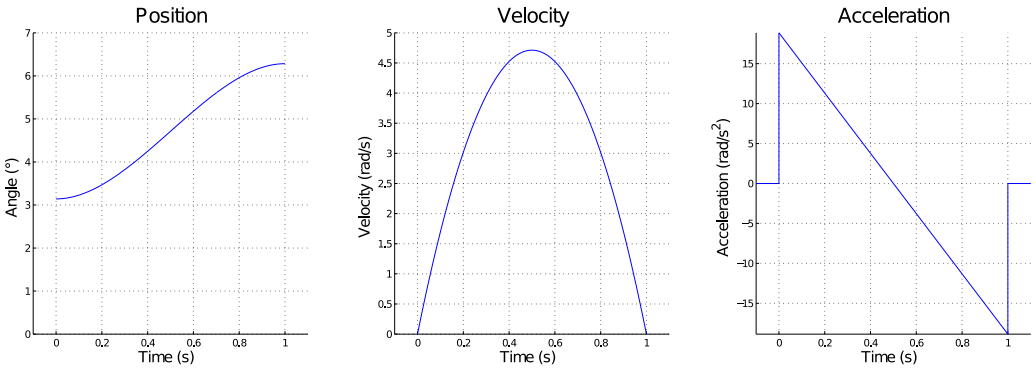


Figure 3.1: Trajectory profiles for a 3^{rd} degree polynomial law.

The position, velocity and acceleration curves have respectively a cubic, parabolic and linear profile. Note that the acceleration has a discontinuity at the beginning and at the end.

When, however, acceleration discontinuities are to be avoided or a precise acceleration profile is needed, the order of the polynomial can be increased to 5. In this case, the function $\theta(t)$ is given by:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{3.6}$$

Note that 2 extra boundary conditions on the initial and final acceleration need to be set:

$$\theta(0) = \theta_{in} \quad \theta(T) = \theta_{fin} \tag{3.7}$$

$$\dot{\theta}(0) = 0 \qquad \dot{\theta}(T) = 0 \tag{3.8}$$

$$\ddot{\theta}(0) = 0 \qquad \ddot{\theta}(T) = 0 \tag{3.9}$$

Using the sample data from the previuos example, the trajectory in this case has the following profiles:

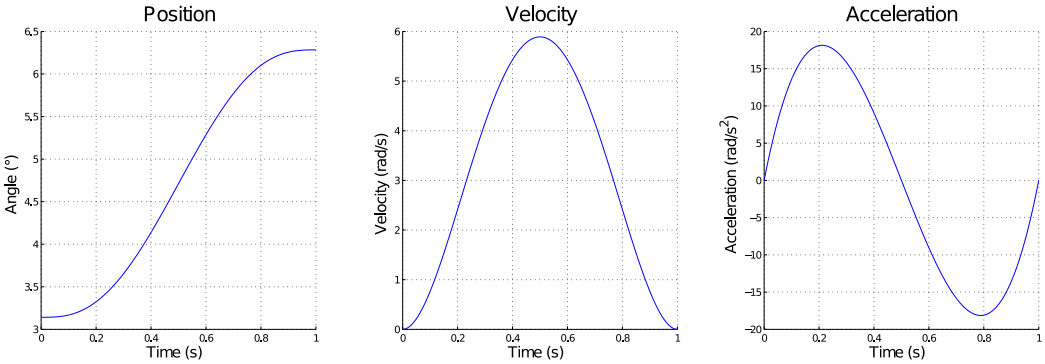


Figure 3.2: Trajectory profiles for a 5^{rd} degree polynomial law.

3.3.1 Point-to-point motion with intermediate via points

From the previous the section, we can extend the problem to the case in which we want the manipulator to be in a specific position at a specific time. That is, we know the position and the orientation in the operational space, and we want to sweep this point(s) without stopping.

The approach to adopt in this case is to analyze the problem piecewisely. The various pieces need to join in a smooth fashion. Each of the "middle stops" will first be converted to the corresponding $\bar{\mathbf{q}}$ vector with the aid of the inverse kinematics routine. The boundary conditions, in this case, are:

$$\theta(t_0) = \theta_{t_0} \qquad \theta(t_f) = \theta_{t_f} \tag{3.10}$$

$$\dot{\theta}(t_0) = \dot{\theta}_{t_0} \qquad \dot{\theta}(t_f) = \dot{\theta}_{t_f} \tag{3.11}$$

From which:

$$\theta(t_0) = a_0 \tag{3.12}$$

$$\dot{\theta}(t_0) = a_1 \tag{3.13}$$

$$\theta(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{3.14}$$

$$\dot{\theta}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 \tag{3.15}$$

With these expressions, we can extract the coefficients for each of the polynomial pieces of the path. Note that, in order to have a continuous trajectory in terms of position and velocity, the position and velocity of the last point of piece *i* need to be equal to the first of piece *i+1*, that is: $\theta_i = \theta_{i+1}$, $\dot{\theta}_i = \dot{\theta}_{i+1}$.

In this case, the main problem is how to calculate the $\dot{\theta}_i$ at the polynomial connection. This can be done using differential kinematics if the cartesian velocity has been specified; if this is not the case, it can be calculated, for example, in order to have a continuous profile of the acceleration.

Obviously, if there's the need for a continuous acceleration profile, this approach can be modified by using a 5^{th} order polynomial, similarly to what was done in the previous section.

3.4 Operational space planning

An operational space trajectory is designed to generate a time sequence of $\mathbf{q}(t)$ joint variables in order to follow a path in the operational space.

The spatial trajectories can be defined in several ways. If we are interested in avoiding certain obstacles (this is the case, for example, of an industrial manipulator for the movement of goods), then only a few path points could be defined. An interpolation, then, takes cares of the other points.

If, on the other hand, we want the manipulator to follow a precise path (i.e. a manipulator for painting or welding), then we are interested in the way in which the journey through the path points is evolving. Our manipulator, since it is needed for trajectory simulations, falls in this second category.

In this case, two main approaches are available: a densification of the path points, in order to limit the *free* interpulated motion, and the definition of an analytical path. It follows that the first approach is more demanding in terms of trajectory definition, since each point needs to be singularly defined, whereas in the second technique path points are easily extrapolated according to the discretization time step.

In both approaches, it is necessary to use the inverse kinematic function to translate the motion specification to the joint space (where actuators operate) [19]. Since this increases the computational burden for trajectory planning, the operations of computing the trajectory and translating it to the joint space are made at a lower frequency with respect to the control frequency. Therefore, it is necessary to interpolate the data before assigning them to the low-level controllers: usually, a simple linear interpolation is adopted.

3.4.1 Predefined analytical path

This approach consists in the definition of *path primitives*, which describe parametrically paths in space. We have that $\mathbf{p}$ is 3×1 vector representing the position of the end effector [23, 8]:

$$\mathbf{p} = f(\mathbf{s}) \tag{3.16}$$

And $\mathbf{p}$ is a function of $\mathbf{s}$, which is the *parametric representation* of path Γ. The latter is usually a continuos function of time, so that we can write:

$$\mathbf{p} = f(\mathbf{s}(t)) \tag{3.17}$$

Once $\mathbf{p}$ is defined, it is also possible to define three important unit vectors for the characterization of the path. They are the *tangent vector*, denoted with $\mathbf{t}$, which is directed along the direction of Γ; the *normal vector*, denoted with $\mathbf{n}$, which lies in the osculating plane and is oriented perpendicularly with respect to the *tangent vector*; the *binomial vector*, finally, denoted with $\mathbf{b}$, completes the right handed frame and is perpendicular to the osculating plane.

Their usefulness lies in their ability to describe the generic path Γ: we can state, in fact:

$$\mathbf{t} = \frac{d\mathbf{p}}{ds} \tag{3.18}$$

$$\mathbf{n} = \frac{1}{\left\| \frac{d^2\mathbf{p}}{ds^2} \right\|} \cdot \frac{d^2\mathbf{p}}{ds^2} \tag{3.19}$$

$$\mathbf{b} = \mathbf{t} \times \mathbf{n} \tag{3.20}$$

With this background, it is possible to introduce two typical trajectories that will be used in the kinematic and dynamic analysis.

Rectilinear path

We consider a rectilinear segment that connects point $\mathbf{p}_i$ to $\mathbf{p}_f$. The position vector can be written as:

$$\mathbf{p}(s) = \mathbf{p}_i + \frac{s}{\|\mathbf{p}_f - \mathbf{p}_i\|} \cdot (\mathbf{p}_f - \mathbf{p}_i) \tag{3.21}$$

Where:

$$s \in [0; \|\mathbf{p}_f - \mathbf{p}_i\|] \tag{3.22}$$

The position of the end effector is thus fully described once the $s = s(t)$ law is defined. Its definition obeys to the same rules presented in the *joint space* analysis section: the best way to choose this function is to use a polynomial with the appropriate degree (usually 3^{rd}, or 5^{th} if specific acceleration requirements need to be satisfied).

As an example, three sweeping laws are analyzed: linear, parabolic and cubic. The parameters to be set are the starting and ending vectors ($\mathbf{p}_f$ and $\mathbf{p}_i$) and the total journey time (T); then, the three time laws are calculated as follows:

- Linear law

$$s(t) = a_0 + a_1 t \tag{3.23}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.24}$$

$$s(T) = \|\mathbf{p}_f - \mathbf{p}_i\| \quad \rightsquigarrow \quad a_1 = \frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T} \tag{3.25}$$

$$s(t) = \frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T} \cdot t \tag{3.26}$$

- Parabolic law

$$s(t) = a_0 + a_1 t + a_2 t^2 \tag{3.27}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.28}$$

$$\dot{s}(0) = 0 \quad \rightsquigarrow \quad a_1 = 0 \tag{3.29}$$

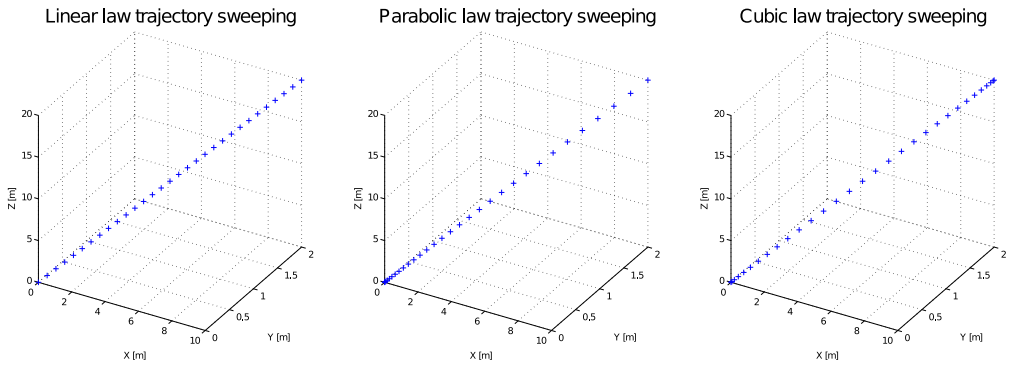


Figure 3.3: Time evolution of three different $s(t)$ trajectory laws for a rectilinear path.

$$s(T) = \|\mathbf{p}_f - \mathbf{p}_i\| \quad \rightsquigarrow \quad a_2 = \frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T^2} \tag{3.30}$$

$$s(t) = \frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T^2} \cdot t^2 \tag{3.31}$$

- Cubic law

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.32}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.33}$$

$$\dot{s}(0) = 0 \quad \rightsquigarrow \quad a_1 = 0 \tag{3.34}$$

$$s(T) = \|\mathbf{p}_f - \mathbf{p}_i\| \quad \rightsquigarrow \quad a_2 = 3\frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T^2} \tag{3.35}$$

$$\dot{s}(T) = 0 \quad \rightsquigarrow \quad a_3 = -2\frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T^3} \tag{3.36}$$

$$s(t) = \frac{\|\mathbf{p}_f - \mathbf{p}_i\|}{T^3} \cdot [3T \cdot t^2 - 2 \cdot t^3] \tag{3.37}$$

An illustrate plot has been created in order to verify the laws: in Fig 3.3, the previous three $s(t)$ expressions are inserted into Eq 3.21 to be evaluated, and a marker is drawn at equally spaced timesteps to show the evolution of the trajectory[1].

[1]For the simulation, the following parameters were used: $\mathbf{p}_1 = [0\ 0\ 0]'$, $\mathbf{p}_2 = [10\ 2\ 20]'$, $T = 10\ s$.

Circular path

For the definition of a circular path, four parameters must be defined:

1. The radius r (which is a scalar)
2. The center $\mathbf{p}_c$ (which is a 3×1 vector)
3. The angular velocity vector $\mathbf{h}$ (which is perpendicular to the circle's plane)
4. The starting path position (which is a point $\mathbf{p}_{in} \in \Gamma$)

We start by analyzing the simplest case, in which the circle is centered in the frame's origin and the path lies on the $x - y$ plane. In this situation, the position vector is given by:

$$\hat{\mathbf{p}}(\theta) = \begin{bmatrix} r \cdot cos\, \theta(t) \\ r \cdot sin\, \theta(t) \\ 0 \end{bmatrix} \tag{3.38}$$

However, we are interested to express $\mathbf{p}$ as a function of the path's law $s = s(t)$. From the definition of radiant:

$$\theta[rad] = \frac{arc}{radius} = \frac{s(t)}{r} \tag{3.39}$$

And we can substitute this result back into Eq 3.38:

$$\hat{\mathbf{p}}(s) = \begin{bmatrix} r \cdot cos\ (s/r) \\ r \cdot sin\ (s/r) \\ 0 \end{bmatrix} \tag{3.40}$$

Which finally yields our goal expression. To extend this result to a generic circle in space, we can write:

$$\mathbf{p}(s) = \mathbf{p}_c + \mathbf{R} \cdot \hat{\mathbf{p}}(s) \tag{3.41}$$

Where $\mathbf{p}_c$ represents the translation of the center, and $\mathbf{R}$ accounts for the rotation of the circle frame. In this latter matrix, $\mathbf{R}$, it is possible to store information on the starting point of the circumference. Once the cartesian

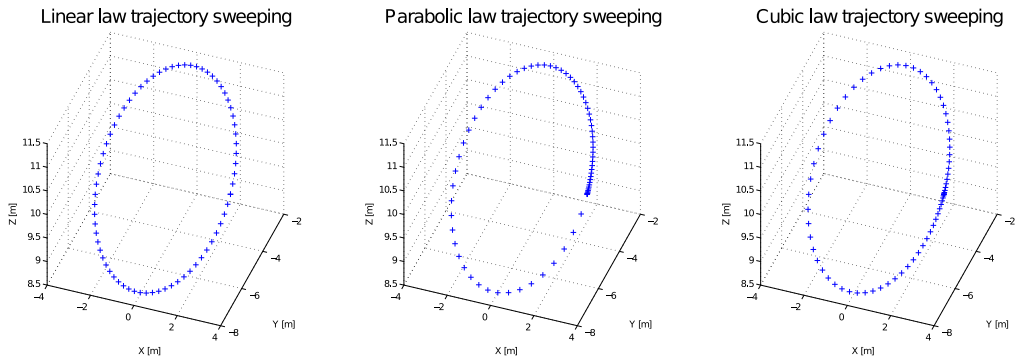


Figure 3.4: Time evolution of three different $s(t)$ trajectory laws for a rectilinear path.

position of this point ($\mathbf{p}_{in}$) and the angular velocity vector $\mathbf{h}$ are known, the new reference frame is composed by the unit vectors $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$:

$$\mathbf{i} = \frac{\mathbf{p}_{in} - \mathbf{p}_c}{\|\mathbf{p}_{in} - \mathbf{p}_c\|} \tag{3.42}$$

$$\mathbf{j} = \frac{\mathbf{h} \times \mathbf{i}}{\|\mathbf{h} \times \mathbf{i}\|} \tag{3.43}$$

$$\mathbf{k} = \frac{\mathbf{h}}{\|\mathbf{h}\|} \tag{3.44}$$

With theese values, the rotation matrix becomes, from its definition:

$$\mathbf{R} = [\ \mathbf{i} \quad \mathbf{j} \quad \mathbf{k}\ ] \tag{3.45}$$

Rewriting Eq 3.41, we finally obtain:

$$\mathbf{p}(s) = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p}_c \\ \hline 0 & 1 \end{array}\right] \cdot \hat{\mathbf{p}}(s) \tag{3.46}$$

$$\mathbf{p}(s) = \left[\begin{array}{ccc|c} i_x & j_x & k_x & p_{cx} \\ i_y & j_y & k_y & p_{cy} \\ i_z & j_z & k_z & p_{cz} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \cdot \left\{\begin{array}{c} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \end{array}\right\} \tag{3.47}$$

Referring to the procedure used in the previous paragraph, we can obtain the linear, parabolic and cubic $s(t)$ path laws in order to simulate a circle trajectory.

- Linear law

$$s(t) = a_0 + a_1 t \tag{3.48}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.49}$$

$$s(T) = 2\pi r \quad \rightsquigarrow \quad a_1 = \frac{2\pi r}{T} \tag{3.50}$$

$$s(t) = \frac{2\pi r}{T} \cdot t \tag{3.51}$$

- Parabolic law

$$s(t) = a_0 + a_1 t + a_2 t^2 \tag{3.52}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.53}$$

$$\dot{s}(0) = 0 \quad \rightsquigarrow \quad a_1 = 0 \tag{3.54}$$

$$s(T) = 2\pi r \quad \rightsquigarrow \quad a_2 = \frac{2\pi r}{T^2} \tag{3.55}$$

$$s(t) = \frac{2\pi r}{T^2} \cdot t^2 \tag{3.56}$$

- Cubic law

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.57}$$

$$s(0) = 0 \quad \rightsquigarrow \quad a_0 = 0 \tag{3.58}$$

$$\dot{s}(0) = 0 \quad \rightsquigarrow \quad a_1 = 0 \tag{3.59}$$

$$s(T) = 2\pi r \quad \rightsquigarrow \quad a_2 = \frac{6\pi r}{T^2} \tag{3.60}$$

$$\dot{s}(T) = 0 \quad \rightsquigarrow \quad a_3 = -\frac{4\pi r}{T^3} \tag{3.61}$$

$$s(t) = \frac{2\pi r}{T^3} \cdot [3T \cdot t^2 - 2 \cdot t^3] \tag{3.62}$$

The trajectory results for some trial parameters[2] are presented in Fig 3.4. A marker is drawn at equally spaced timesteps to show the evolution of the trajectory.

[2]For the simulation, the following parameters were used: $\mathbf{h} = [0\,{-5}\,10]'$, $\mathbf{p}_c = [0\,{-5}\,10]'$, $\mathbf{p}_{in} = [r \;\; -5 \;\; 10]'$, $r = 3$ m, $T = 10\,s$.

3.4.2 Corrected *on the go*

When an analytical trajectory is designed, it is important to remeber that the manipulator, for several reason, might not obey to the predefined laws. If, for example, we want the end effector the follow a goal trajectory (i.e. we need to weld two pieces of metal), and there are some kind of disturbances, the actual path might drift from the theoretical one, resulting in errors and stability problems.

On the other hand, if our goal is to simulate the motion in presence of disturbances (i.e. a RvD manipulator), we need a system that is able to detect these disturbances and to calculate the modified trajectory (the manipulator, otherwise, will keep going approximately on the same path).

Both of these problems can be solved by implementing a software that enables *on the go* corrections, that is, real time modification of the trajectory. For this to be feasible, the control software needs to communicate with the outside world to gain information on position, on disturbances and on the way they are acting, in order to subsequently modify the trajectory. The tools for this "communication" are usually position and force sensors.

The way to perform *on the go* corrections, is to define, first of all, a goal trajectory. Then, if for some reason there's a change in this path, the online sensors are used to extrapolate the cause of this change (a vectorial force, for example) and to compute the new trajectory that has to be imposed. Once the inputs to the motors are calculated, a stable control system will take care of providing the actuators with the appropriate control law.

The details on this technique, applied to RvD simulations, is extensively explained in Chapter 6.

Chapter 4

Dynamics

4.1 Introduction

The analysis of the robot so far has focused on kinematics and positioning problems only.

This chapter deals with the study of the forces required to cause the motion. In order to accomplish this goal, the equations of motion will be presented, and the relationship between the input torque to the motors and the actual structure motion will be analyzed. In the field of robotics, two main approaches are available: the Euler-Newton and the Euler-Lagrange [23, 22]. They both lead to the same unique results, but they are indeed very different, both conceptually and computationally.

4.2 Euler-Lagrange method

Euler-Lagrange method is an energy based approach. With this technique, the equations of motion can be obtained in a systematic way independently of the reference frame. By choosing a set of generalized coordinates describing the link positions (the $\mathbf{q} = [q_1 \ \dots \ q_n]$ are the natural choice), it is possible to defined the Lagragian of the structure [23, 8]:

$$\mathcal{L} = \mathcal{T} + \mathcal{U} \tag{4.1}$$

Where $\mathcal{T}$ and $\mathcal{U}$ are the kinetic and potential energy. Lagrange equations is given by:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) - \left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}\right) = \tau \tag{4.2}$$

Where τ are the generalized forces, or non-conservative forces acting on the links: they are mainly given by the actuator torques and the joint friction torques. From this equation it is possible to examine the relationship between the joint positions and the generalized forces.

However, although the formulation is fairly easy to understand, its implementation is actually very troublesome. The equations of the kinetic and the potential energy are, in fact:

$$\mathcal{T} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}(\mathbf{q})\dot{q}_i\dot{q}_j = \frac{1}{2}\dot{\mathbf{q}}^{\mathbf{T}}\mathbf{B}(\mathbf{q})\dot{\mathbf{q}} \tag{4.3}$$

Where $\mathbf{B}(\mathbf{q})$ represents:

$$\begin{aligned}\mathbf{B}(\mathbf{q}) = \sum_{i=1}^{n}(m_{l_i}\mathbf{J}_{Pi}{}^{T}\mathbf{J}_{Pi} + \mathbf{J}_{Oi}{}^{T}\mathbf{R}_i\mathbf{I}_{l_i}\mathbf{R}_i{}^{T}\mathbf{J}_{O_i} + m_{m_i}\mathbf{J}_{Pm}{}^{T}\mathbf{J}_{Pm} \\ + \mathbf{J}_{O_{mi}}{}^{T}\mathbf{R}_{m_i}\mathbf{I}_{m_i}\mathbf{R}_{m_i}{}^{T}\mathbf{J}_{O_{mi}})\end{aligned} \tag{4.4}$$

And, for the potential energy:

$$\mathcal{U} = \sum_{i=1}^{n}(\mathcal{U}_{l_i} + \mathcal{U}_{m_i}) = -\sum_{i=1}^{n}(m_{l_i}\mathbf{g}_0{}^{T}\mathbf{p}_{l_i} + m_{m_i}\mathbf{g}_0{}^{T}\mathbf{p}_{m_i}) \tag{4.5}$$

These equations do not have an easy solution: Eq 4.3, for example, is highly non linear, and the $\mathbf{B}(\mathbf{q})$ matrix is made up of several nested components that are not well suited for a quick, recursive approach. Moreover, the presence of partial derivatives and the fact that we need to deal with symbolic quantities complicates the problem exponentially. Thus, even tough this approach is good for having a sense of the physics involved in the problem, it does not appear to be a viable method for a real time code simulation.

4.3 Euler-Newton method

Euler-Newton approach is based is based on the balance of all the forces and torques acting on the generic link of the manipulator. The solution of this problem is well suited for a recursive approach, thus making this our choice for the dynamics simulation. The approach starts from the classic Newton's formula [8]:

$$F = m\,\dot{v}_C \tag{4.6}$$

From solid mechanics [32], we can recall that the moment acting on a rotating body of inertia ${}^{C}I$ is given by:

$$N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega \tag{4.7}$$

Where ω is the angular velocity and $\dot{\omega}$ is the angular acceleration. By knowing the trajectory to be followed, we then know the position, velocity and acceleration of the joints (that is, $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$).

With these information and with data about the mass distribution of each joint (mass and inertia tensor), we can calculate the joint torques required at each link. This approach is much more computationally-friendly, and its equations are suited for a simple recursive technique. Thus, in this thesis we analyze dynamics with Euler-Newton method.

4.3.1 The Euler-Newton routine

Equations can be implemented by following Luh-Walker method, developed in 1980 [17]. It is based on two parts: the *outward* and the *inward* iteration. The first part consists on the calculation of ω, $\dot{\omega}$, $\dot{\mathbf{v}}$ $\dot{\mathbf{v}}_{cm}$ for all the links of structure. These computations are "propagated" from *link 1* to *link N* of the chain, hence the name "outward".

Outward **part**

Recalling Section 2.5.1, the propagation of the rotational velocity was obtained:

$${}^{i+1}w_{i+1} = {}^{i+1}_{i}R\,{}^{i}w_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} \tag{4.8}$$

Derivation of rotational velocity implies the derivation of the trasformation matrix as well. If we express the generic rotational velocity as:

$$ {}^{A}w_{C} = {}^{A}w_{B} + {}^{A}_{B}R\,{}^{B}w_{C} \tag{4.9} $$

Its derivative is:

$$ {}^{A}\dot{w}_{C} = {}^{A}\dot{w}_{B} + \frac{d}{dt}({}^{A}_{B}R\,{}^{B}w_{C}) \tag{4.10} $$

Remebering the derivative of a vector expressed in a moving frame, Eq. 4.10 becomes:

$$ {}^{A}\dot{w}_{C} = {}^{A}\dot{w}_{B} + {}^{A}_{B}R\,{}^{B}\dot{w}_{C} + {}^{A}w_{B} \times {}^{A}_{B}R\,{}^{B}w_{C} \tag{4.11} $$

In a similar fashion, we can take the derivative of Eq. 4.8:

$$ {}^{i+1}\dot{w}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{w}_{i} + {}^{i+1}_{i}R\,{}^{i}w_{i} \times \dot{\theta}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} \tag{4.12} $$

The linear acceleration is (from Eq. 2.22):

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\,[\,{}^{i}\dot{v}_{i} + {}^{i}\dot{\omega}_{i} \times {}^{i}P_{i+1} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{i+1})\,] \tag{4.13} $$

The linear acceleration of center of mass of *link i+1*, expressed in *Frame i+1*, is expressed by:

$$ {}^{i}\dot{v}_{C_i} = {}^{i}\dot{v}_{i} + {}^{i}\dot{\omega}_{i} \times {}^{i}P_{C_i} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{C_i}) \tag{4.14} $$

Note that ${}^{i}P_{C_i}$ represents the distance from the i joint to the center of mass of *link i*. As far as concerns the forces and torques acting on the link, we can apply Equations 4.6 and 4.7:

$$ F_{i} = m\,\dot{v}_{C_i} \tag{4.15} $$

$$ N_{i} = {}^{C}_{i}I\dot{\omega}_{i} + \omega_{i} \times {}^{C}_{i}I\omega i \tag{4.16} $$

Summing up, the outward part of the solution process is then constituted by solution of the following set of equations, starting from *i=0* and arriving to *i=N-1*:

$$ \begin{cases} {}^{i+1}\dot{w}_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\dot{w}_{i} + {}^{i+1}_{i}R\,{}^{i}w_{i} \times \dot{\theta}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{k}_{i+1} \\ {}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R\,[\,{}^{i}\dot{v}_{i} + {}^{i}\dot{\omega}_{i} \times {}^{i}P_{i+1} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{i+1})\,] \\ {}^{i}\dot{v}_{C_i} &= {}^{i}\dot{v}_{i} + {}^{i}\dot{\omega}_{i} \times {}^{i}P_{C_i} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{C_i}) \\ F_{i} &= m\,\dot{v}_{C_i} \\ N_{i} &= {}^{C}_{i}I\dot{\omega}_{i} + \omega_{i} \times {}^{C}_{i}I\omega i \end{cases} \tag{4.17} $$

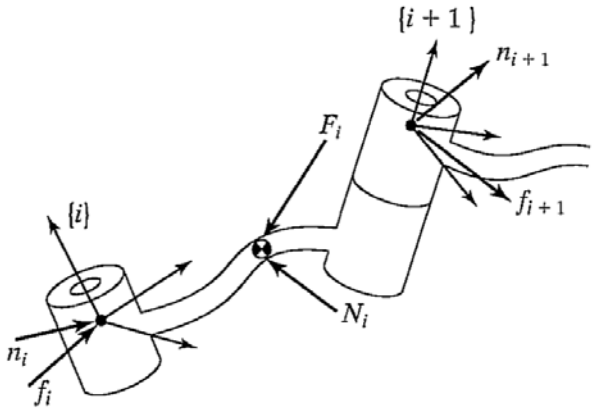


Figure 4.1: Free body diagram of *link i*, with force balance

Inward part

The second part comprises the use of Newton Euler equations (Eq 4.6 and 4.7) to obtain the inertial forces and torques acting on the links' centers of mass. Then, referring to the free body diagram of Fig. 4.1 [8], the force and moment balance equations need to be considered, in order to extract the joint torques.
Every link experiences inertial force and torque in addition to forces and torques exerted on it by the adjoining links. From the free body diagram, the force and torque equilibrium yield the following balance equations:

$$^{i}F_i = {}^{i}f_i - {}_{i+1}^{i}R^{i+1}f_{i+1} \tag{4.18}$$

$$^{i}N_i = {}^{i}n_i - {}^{i}n_{i+1} + (-{}^{i}P_{C_i}) \times {}^{i}f_i - ({}^{i}P_{i+1} - {}^{i}P_i) \times {}^{i}f_{i+1} \tag{4.19}$$

Where the following notation was used:

- f_i is the force exerted by *link i-1* on *link i*
- n_i is the torque exerted by *link i-1* on *link i*

Equation 4.19 can be rearranged with the aid of rotational matrices and the results from Eq. 4.18:

$$^{i}N_i = {}^{i}n_i - {}_{i+1}^{i}R^{i+1}n_{i+1} - {}^{i}P_{C_i} \times {}^{i}F_i - {}^{i}P_{i+1} \times {}_{i+1}^{i}R^{i+1}f_{i+1} \tag{4.20}$$

Reordering Eq. 4.18 and Eq. 4.20, we can finally obtain the iterative expressions needed. This time the index count for the solution will start from N and descreases to 1.

$$\begin{cases} {}^{i}f_i &= {}^{i}F_i + {}^{i}_{i+1}R^{i+1}f_{i+1} \\ {}^{i}n_i &= {}^{i}N_i + {}^{i}_{i+1}R^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R^{i+1}f_{i+1} \\ \tau_i &= {}^{i}f_i^T \, \hat{Z}_i \end{cases} \tag{4.21}$$

Since the calculations are taken from the end effector to the first link, this second part of the routine is called *"inward"*.

Summing up, the script for the torque calculation will be composed of these two parts:

```
% acceleration analysis > Outward iterations

for j=1:6
        W(:,j,i)=R_i(:,:,j)*W(:,j-1,i)+[0 0 q_dot(j,i)]';
        W_d(:,j,i)=R_i(:,:,j)*W_d(:,j-1,i)+cross(R_i(:,:,j)*W(:,j-1,i),[0 0 q_dot(j,
            i)]')+[0 0 q_ddot(j,i)]';
        V(:,j,i)=R_i(:,:,j)*(V(:,j-1,i)+cross(W(:,j-1,i),P_i(:,j)));
        V_d(:,j,i)=R_i(:,:,j)*(V_d(:,j-1,i)+cross(W_d(:,j-1,i),P_i(:,j))+cross(W(:,j
            -1,i),cross(W(:,j-1,i),P_i(:,j))));
        Vc(:,j,i)=V(:,j,i)+cross(W(:,j,i),com(:,j));
        Vc_d(:,j,i)=V_d(:,j,i)+cross(W_d(:,j,i),com(:,j))+cross(W(:,j,i),cross(W(:,j
            ,i),com(:,j)));
        F(:,j,i)=m(j)*Vc_d(:,j,i);
        N(:,j,i)=I(:,:,j)*W_d(:,j,i)+cross(W(:,j,i),I(:,:,j)*W(:,j,i));
end

% acceleration analysis > Inward iterations

for j=5:-1:1
    f(:,j,i)=R_i(:,:,j+1)'*f(:,j+1,i)+F(:,j,i);
    n(:,j,i)=N(:,j,i)+R_i(:,:,j+1)'*n(:,j+1,i)+cross(com(:,j),F(:,j,i))+cross(P_i(:,
        j+1),R_i(:,:,j+1)'*f(:,j+1,i));
    tau(i,j)=n(3,j,i);
end
```

In this script, however, gravity is not considered. This is the case when working in orbit, but on earth we need to deal with gravity, which will constitute the prior force field that needs to be overcome by the motors.
Newton's method, unlike Lagrange's, allows for an easy addition/removal of the gravity contribution: if we want to consider a gravitational field, in fact, we just need to set $\mathbf{v}_0 = \mathbf{g}$, where $\mathbf{g}$ has the magnitude of the gravity vector but points in the opposite direction.
This is equivalent to accelerate the base of the manipulator upward with $\mathbf{g}$ magnitude.

Initial conditions

For both inward and outward iterations, we need some starting conditions. Referring to equations block 4.17, the computation process starts for *i=0*. This means that some of the parameters need to be known: ω_0, $\dot{\omega}_0$, $\dot{\mathbf{v}}_0$. These have to be set in this fashion:

$$\omega_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \dot{\omega}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \dot{\mathbf{v}}_0 = k \cdot \begin{bmatrix} 0 \\ 0 \\ \mathbf{g} \end{bmatrix} \tag{4.22}$$

Where k is 0 if gravity is not considered, 1 if it is considered. Obviously, if the base is connected to ground, ω_0 and $\dot{\omega}_0$ will be zero.
The initial conditions concerning equations block 4.21 are related to the dynamic effects present at the end effector; these effects can be due to impacts/-contacts or to the presence of a tool or a load (i.e. industrial manipulators). In this case, we suppose these components to be zero, that is, we suppose an unloaded robot, subjected only to its dynamics with no external contributions (apart from gravity). Thus, the initial conditions are:

$$^7 f_7 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad ^7 n_7 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.23}$$

In the future development of this project, the end effector will be provided with a force sensor 6.3. This equipment will calculate the vectorial force acting on the wrist, which will then be fed to a routine to compute the perturbed trajectory. Please refer to Chapter 6 for the details.

4.4 Simulation

The routine described in the previous section (made up of an *outward* and an *inward* part) was implemented in a Matlab code. This script can be used for at least two goals:

- ◇ *Control*: instead of controlling the joints with a trajectory ($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$), we can directly control them with the computed torques

- ◇ *Sizing*: the torques obtained from the calculations can be useful for the sizing of the motors. By simulating several typical maneuvers, in fact, we can inferre the average maximum torque for each joint.

Recalling Section 2.6 of the kinematics chapter, it is possible to continue the analysis with the addition of the torque calculation. According to what was done previously, we can simulate the same two trajectories: a line and a circle.

4.4.1 Rectilinear trajectory

In this case, recalling Eq 2.44 and 2.45, the trajectory is defined as:

$$x(t) = \begin{Bmatrix} a_5t^5 + a_4t^4 + a_3t^3 + a_0 \\ 1.1 \\ 0.2 \end{Bmatrix}$$

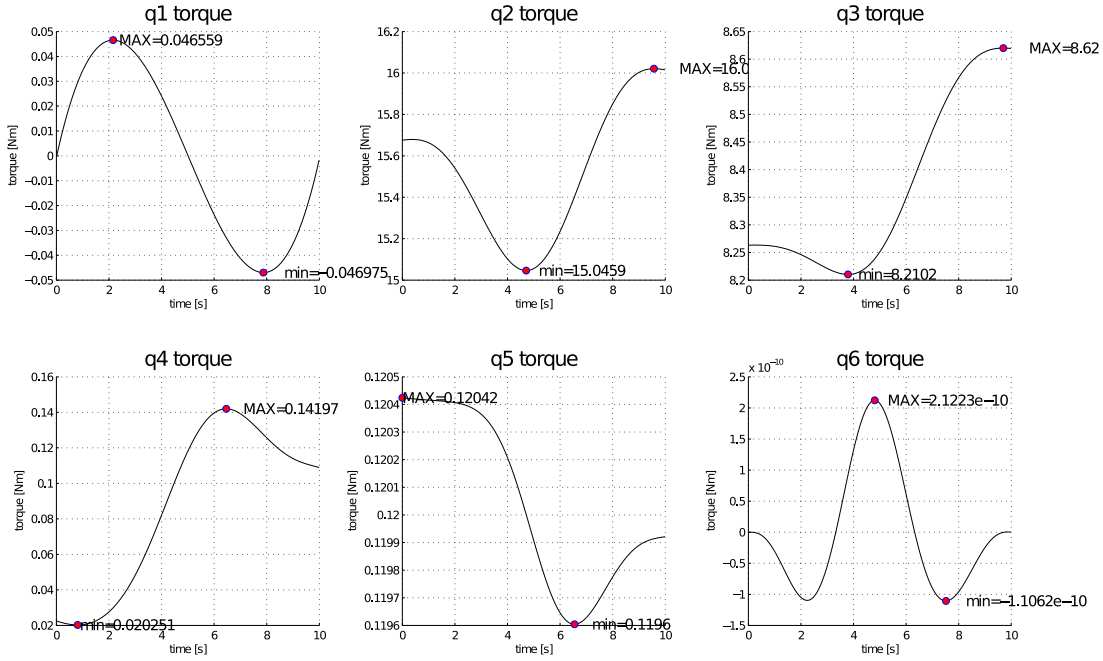
$$\dot{x}(t) = \begin{Bmatrix} 5a_5t^4 + 4a_4t^3 + 3a_3t^2 \\ 0 \\ 0 \end{Bmatrix}$$

Where the coefficients and path parameters are the same of the kinematic simulation. For each iteration, the Matlab code computes vector $\mathbf{q} = [q_1 \dots q_n]$ and its first two derivatives using inverse differential kinematics; then, the *outward-inward* Euler-Newton technique is executed, allowing for the calculation of the joints torques.

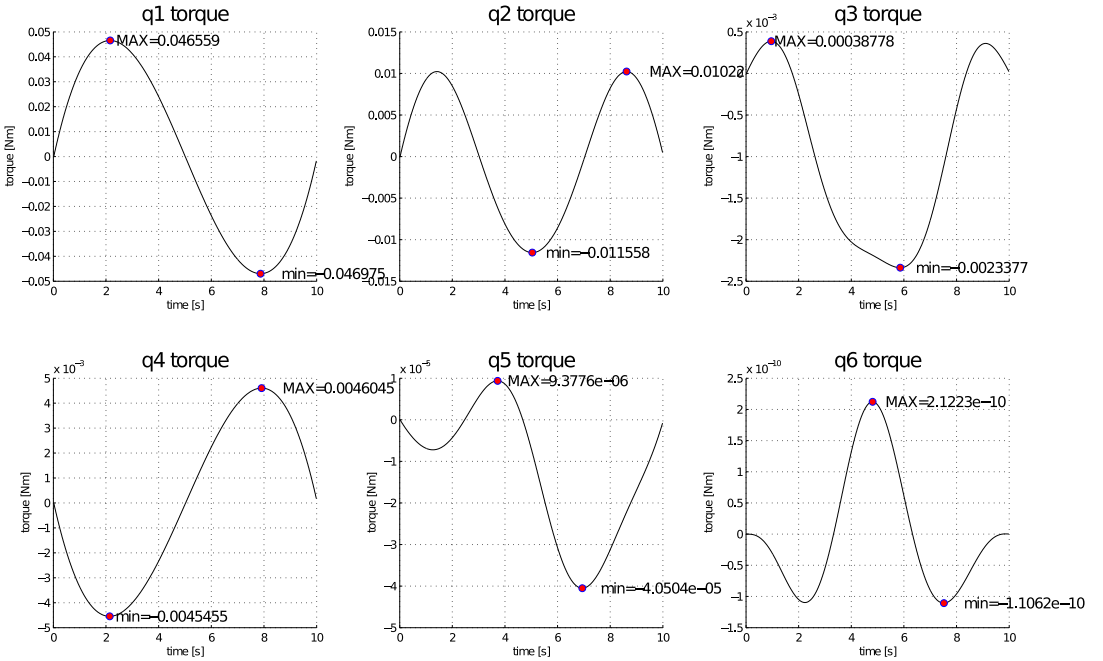
The geometric parameters (mass, center of mass, inertia tensor) of each link are obtained from SolidWorks and are stored in an auxiliary function (refer to Chapter 7 for the complete list of these values).

The simulation here can be divided in two parts: with and without gravity. In the first case, the torques needed will keep into account the static equilibrium as well as the dynamic interactions due to the relative motion of the manipulator; in the latter case, the joints are not required to withstand the weight of the structure, but only the relative dynamics contributions.

If gravity is taken into account, the torques needed for the trajectory of Fig 2.11 are presented in Fig 4.2 (a). On the other hand, if $\mathbf{g} = \mathbf{0}$, the torque behavior in absence of gravity can be seen in Fig 4.2 (b).



(a) Gravity case, *T=10 s.*



(b) No gravity case, *T=10 s.*

Figure 4.2: Rectilinear trajectory simulation. T=10 s

It follows that the torques required depend on the trajectory's time law: the faster the path is swept, the higher the moment that the motors need to exert. The same trajectory, if the timespan becomes $T = 0.1 \cdot T_1 = 1\ s$, requires the torques plotted in Fig 4.3.

The effect that gravity has on the structure's dynamics can be analyzed in Fig 4.4. In these plots, the torques in presence of gravity are represented with solid curves, whereas the dashed lines represent the weightless case. Some interesting information can be inferred from this picture: *joint 1* and *6*, for example, are not affected by the gravitational field. This can be easily explained due to the fact that, for *joint 1*, the joint axis is parallel to the gravity vector; for *joint 6*, we supposed an axialsymmetrical link attached to it, thus not creating any gravity moment.

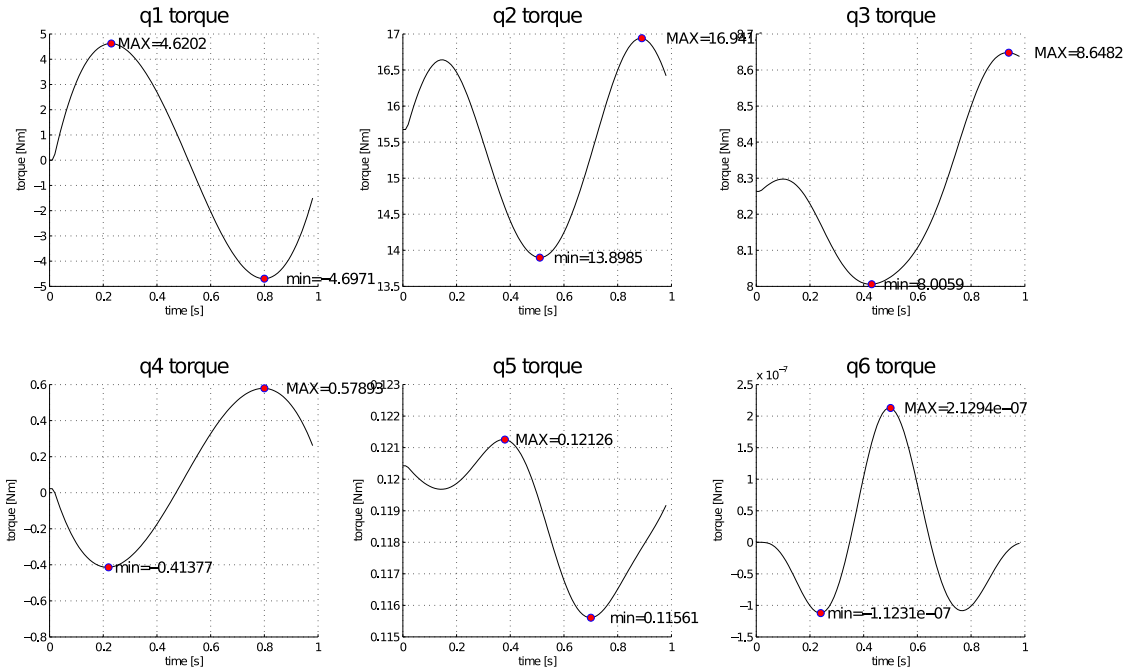
	T=10 s		T=1 s	
	g=9.81	*g=0*	*g=9.81*	*g=0*
τ_{1max}	0.046	0.046	4.62	4.62
τ_{2max}	16.05	0.012	16.94	1.156
τ_{3max}	8.623	$3.87{\cdot}10^{-4}$	8.648	0.037
τ_{4max}	0.442	0.0046	0.578	0.454
τ_{5max}	0.121	$9.42{\cdot}10^{-6}$	0.122	0.004
τ_{6max}	$2.11{\cdot}10^{-10}$	$2.11{\cdot}10^{-10}$	$2.13{\cdot}10^{-7}$	$2.13{\cdot}10^{-7}$

Table 4.1: Linear trajectory: torques needed at each joint for different maneuver conditions. All values have Nm units.

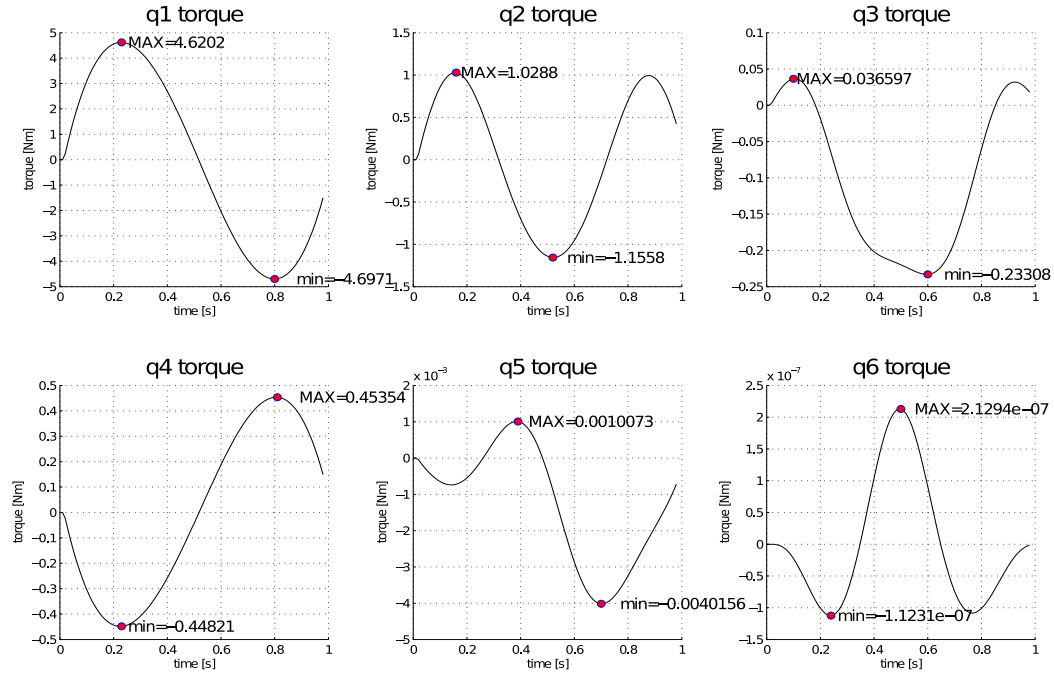
For all the other joints, the torque is gravity dependent, and there is a dramatic difference in the maximum torque values required.

From Table 4.1 we can inferre a key result that will be useful during the sizing process: generally speaking, the dynamic effects induced by the motion, do not have a relevant effect on the torques required at the joints. If we consider the columns that take into account gravity, only *joint 1* and *joint 6* are clearly affected by the movement: for the other joints, a very fast movement (T=1 s) requires not more than an additional 10% of the corresponding torque in the quasi-static case (T=10 s).

The sizing of the motors for *joints 2, 3, 4, 5*, thus, can start from the static analysis. Then, the dynamic effects can be accounted by simply multiplying

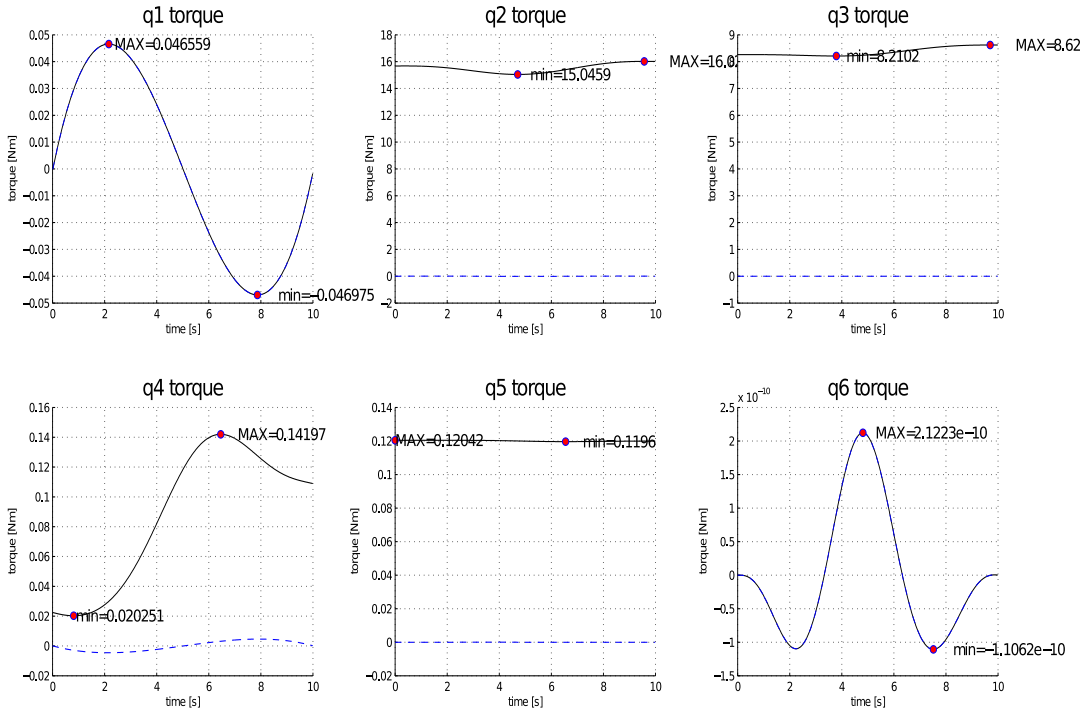


(a) Gravity case, *T=1 s.*

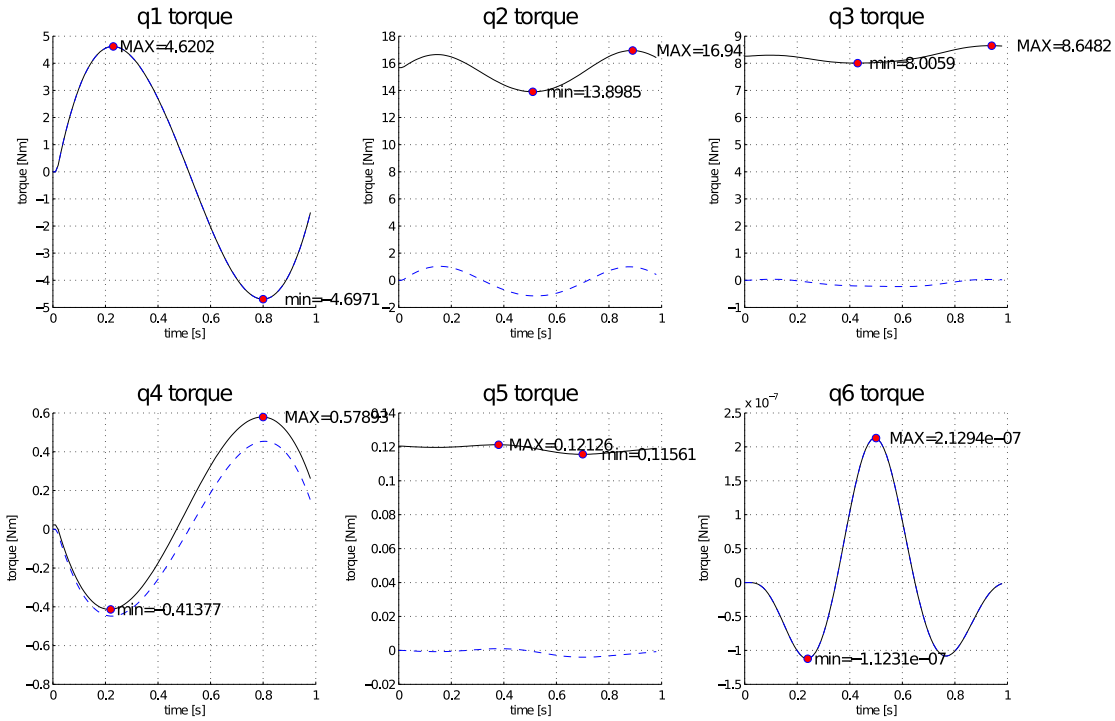


(b) No gravity case, *T=1 s.*

Figure 4.3: Rectilinear trajectory simulation. T=1 s



(a) Torque comparison, *T=10 s* case.



(b) Torque comparison, *T=1 s* case.

Figure 4.4: Gravity influence on torques, cases *T=10 s* and *T=1 s*.

the results from an appropriate safety factor. From the data available, a coefficient ranging from *1.2÷1.3* seems to be a good choice.

This procedure obviously does not work for *joint 1* and *6*: since they are not influenced by gravity, the only sizing parameter is the dynamic contribution. In order to identify their requirements, the only way to proceed is to simulate several typical trajectories; in addition, if we want to have an adequate margin on the motor performances, we can boost the requirements of typical trajectories by decreasing their period.

4.4.2 Circular trajectory

For the circular trajectory, we recall Eq 4.25:

$$x(t) = \begin{Bmatrix} 0.4\, cos(a_5t^5 + a_4t^4 + a_3t^3) \\ 0.4\, sin(a_5t^5 + a_4t^4 + a_3t^3) \\ 0.3 \end{Bmatrix} \tag{4.24}$$

$$\dot{x}(t) = \begin{Bmatrix} -0.4\, sin(a_5t^5 + a_4t^4 + a_3t^3)(5a_5t^4 + 4a_4t^3 + 3a_3t^2) \\ 0.4\, cos(a_5t^5 + a_4t^4 + a_3t^3)(5a_5t^4 + 4a_4t^3 + 3a_3t^2) \\ 0 \end{Bmatrix} \tag{4.25}$$

Where the path parameters are as follows:

$$\begin{aligned} x_{in} = x_{fin} &= [0.5 \;\; 0.7 \;\; 0.3] \quad [m] \\ x_c &= [0.5 - R \;\; 0.7 \;\; 0.3] \quad [m] \\ \bar{\mathbf{n}} &= [0 \;\; 0 \;\; 1] \\ R &= 0.4\, m \\ \psi_{att} &= 0^\circ \\ \theta_{att} &= 45^\circ \\ \phi_{att} &= 0^\circ \end{aligned}$$

And the coefficients are:

$$\begin{aligned} a_0 &= 0 \\ a_1 &= 0 \\ a_2 &= 0 \\ a_3 &= 0.0628 \\ a_4 &= -9.425 \cdot 10^{-3} \\ a_5 &= 3.769 \cdot 10^{-4} \end{aligned}$$

$$t \in [0;\ 10]\ s$$

$$dt = 0.001\ s$$

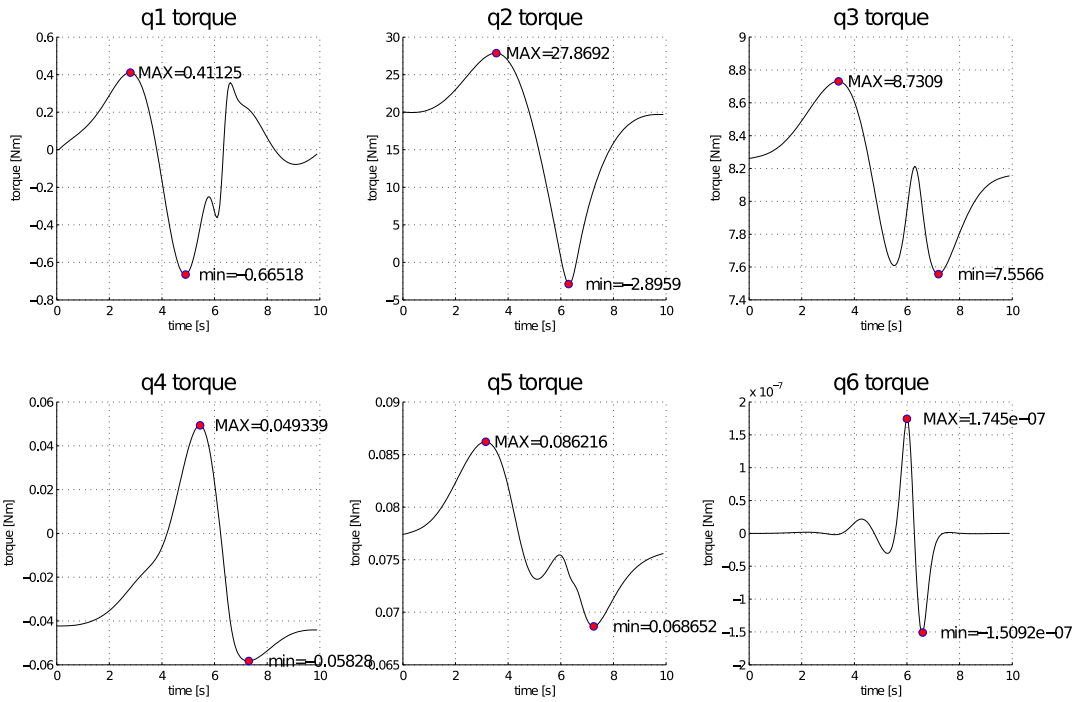
Even in this case, we will divide the simulation in two parts: one with and the other without the account for gravity. For a maneuver time of *10 s*, in the case of g=9.81 ms^{-2}, Fig 4.5 (a) represents the needed torques; if gravity is not considered, the behaviors are pictured in Fig 4.5 (b). If a quicker maneuver is imposed, the plots change into Fig 4.6, where T=3 *s* was used as the period.

Even in this case, it is possible to extrapolate a table containing the torque values of the simulations: the data are stored in Tab 4.2. This table confirms what was stated in the previous paragraph: gravity has a noticeable influence on all the torques apart from *joint 1* and *joint 6*.

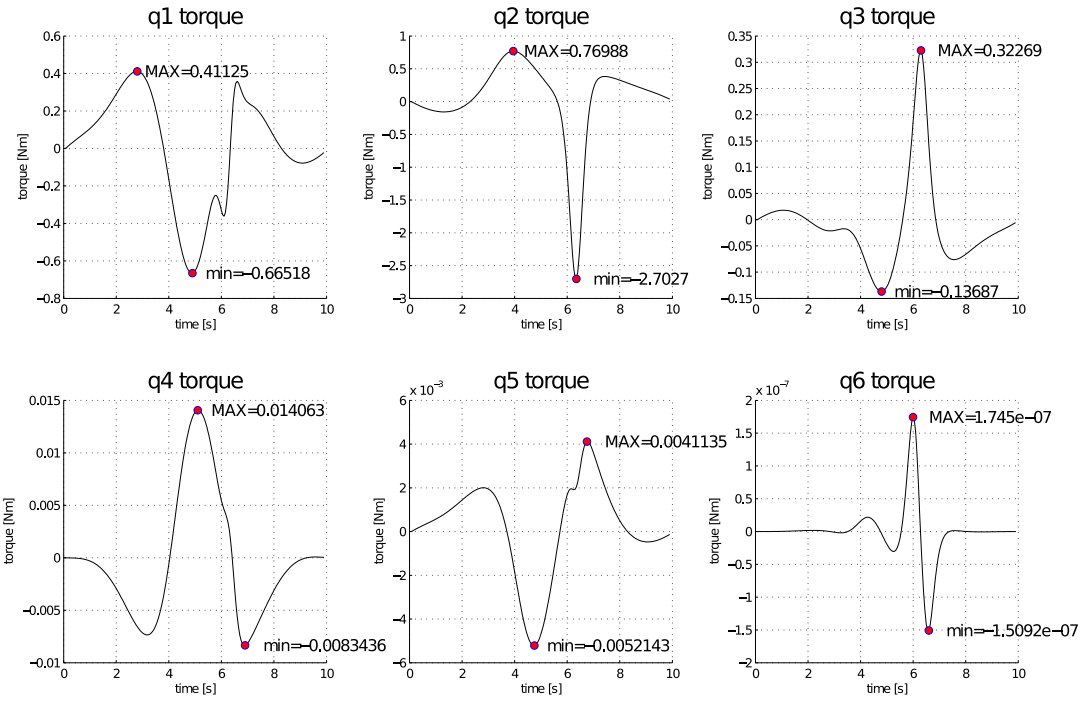
This maneuver, moreover, confirms our choice for the coefficient that will be used for sizing *joint 2, 3, 4, 5*: a value in the 1.2÷1.3 range seems to be the approriate choice.

Together with the previous simulation, we can gather some information on the intensity of the torques at *joint 1* and *6* for their sizing. As far as concerns *joint 1*, even for a very quick maneuver, we do not exceed 10 Nm: in the case of the line, 4.62 Nm were needed, and in the circular path we arrive a 6.516 Nm. By applying a reasonable safety factor, we can chose a motor in the 10÷12 Nm range.

As far as concerns *joint 6*, the torque required is very small: for the fastest trajectory simulated, the calculations yield a value of $7.23 \cdot 10^{-6}$ Nm. However, we are in the approxiamtion of an axialsymmetric body connected

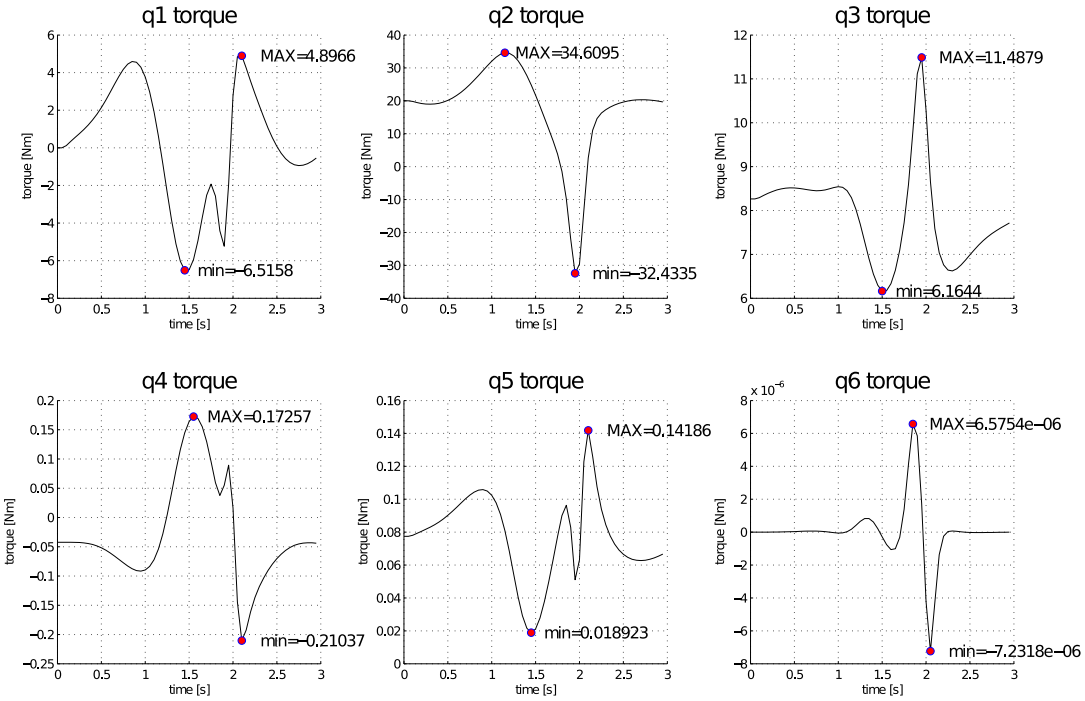


(a) Gravity case, *T=10 s.*

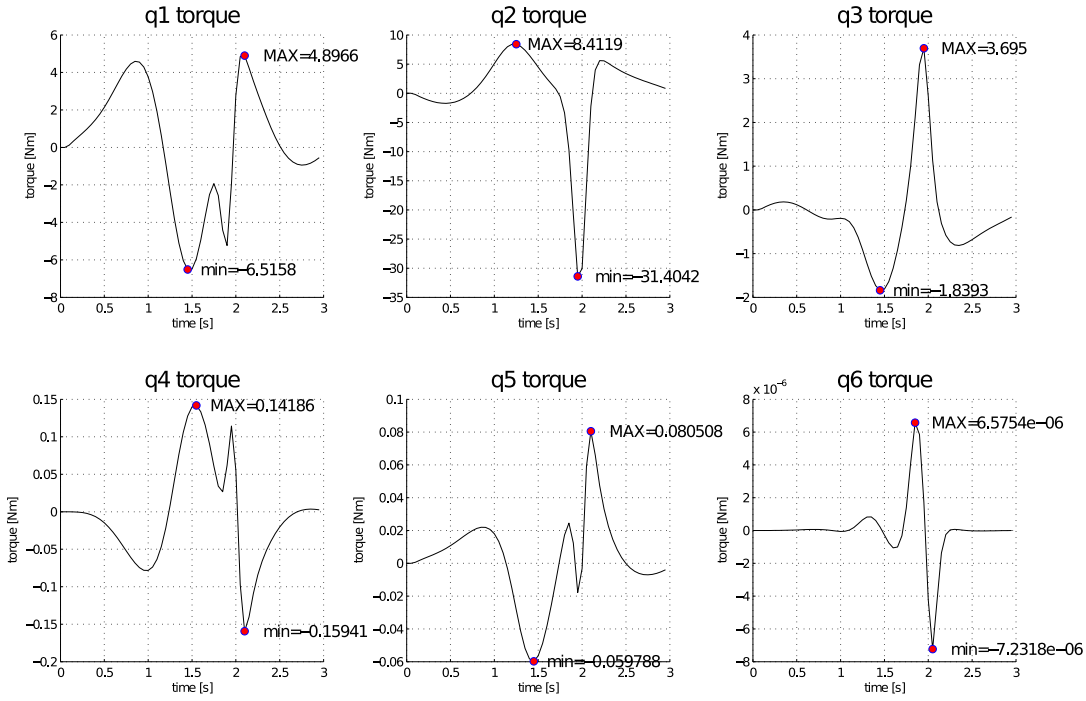


(b) No gravity case, *T=10 s.*

Figure 4.5: Circular trajectory simulation. T=10 s



(a) Gravity case, *T=3 s*.



(b) No gravity case, *T=3 s*.

Figure 4.6: Circular trajectory simulation. T=3 s

	T=10 s		T=3 s	
	g=9.81	*g=0*	*g=9.81*	*g=0*
τ_{1max}	0.411	0.411	6.516	6.516
τ_{2max}	27.869	2.703	34.610	31.404
τ_{3max}	8.731	0.323	11.488	3.695
τ_{4max}	0.058	0.014	0.211	0.159
τ_{5max}	0.086	$5.21{\cdot}10^{-3}$	0.142	0.081
τ_{6max}	$1.74{\cdot}10^{-7}$	$1.74{\cdot}10^{-7}$	$7.23{\cdot}10^{-6}$	$7.23{\cdot}10^{-6}$

Table 4.2: Circular trajectory: torques needed at each joint for different maneuver conditions. All values have Nm units.

to the shaft whose axis of giration is coincident with the shaft's axis. Since other bodies, non necessarily axialsymmetric, might be attached to it for testing, and due to possible misalignments between the axis, the torque required could be bigger. In order to stay away from saturation, we can think of increasing the requirements: a commercial motor in the $0.1 \div 0.2$ Nm range appears to be more than sufficient to withstand misaligments and (limited) extra weight.

4.5 Model verification: Simulink's *SimMechanics* toolbox.

The results obtained from the Matlab analysis need to be cross-verified. In order to do this, a Simulink model was built, upgrading the kinematics one presented in Chapter 2, Fig 2.15.

Recalling Simulink's environment, we recall that there exist two options for the joint control. We now want to use the *torque control*. The block diagram is presented in Fig 4.7.

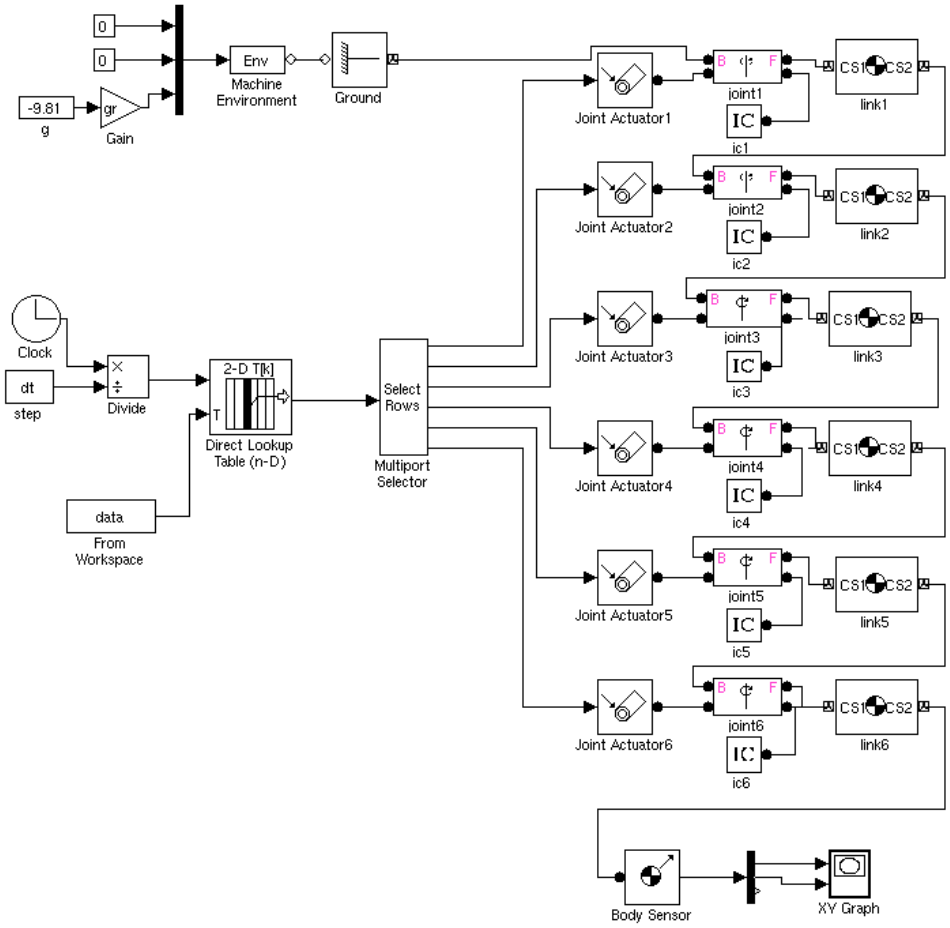


Figure 4.7: *Simulink* block diagram for trajectory analysis and simulation. In this case, the joints are torque controlled.

The Matlab routine saves a $N \times n_{it}$ matrix[1], containing, for each iteration, the torques computed with Euler-Newton approach. Simulink then extracts the torque values with the aid of a *Lookup table* and feeds them into the joint inputs. Note that in upper left hand corner there's a block structure that checks if the Matlab simulation was run with or without gravity.

Similarly to what was done in the kinematic case, the correctness of the results can be verified either with an animation or with the plot of the tra-

[1] N stands for the degrees of freedom, whereas n_{it} represents the iteration count, which can be obtained from $\frac{T_{sim}}{\Delta t}$, with T_{sim} being the simulation time and Δt the step size.

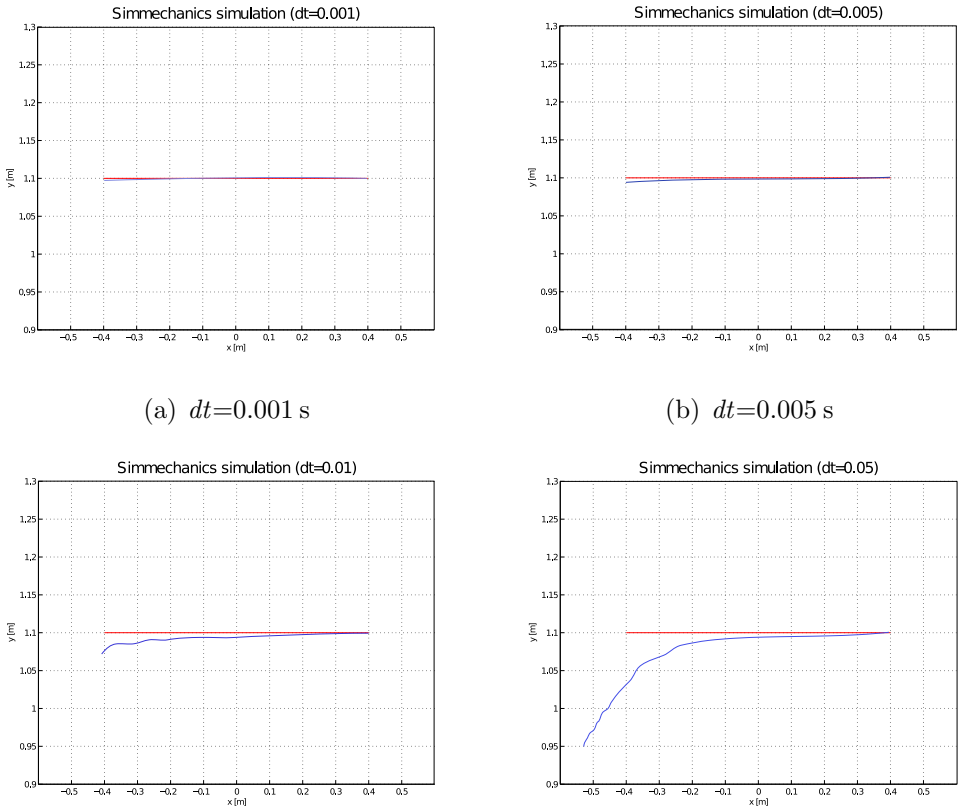
jectory in a $2D$ graph ($x-y$, $x-z$ or $y-z$). In the model, we verify the data in both ways.

Each link needs to be fully described in terms of its mass, center of mass, inertia tensor and frame of references. Refer to Chapter 7 for a complete list of the parameters.

The XY trajectory plots obtained with Simulink's SimMechanics tool are presented in Fig 4.8 (for the linear path) and Fig 4.9 (circular path). It is interesting to note how this model is strongly affected by the simulation conditions. That is, a small change in the time step size affects dramatically the computed motion; moreover, a change in the solution technique (i.e. *ode45*, *ode23*, *ode113*) might lead to divergence. This is a behavior that was not particularly marked in the kinematics analysis.

A likely explanation is connected with the probable uncertainties of the overall simulation model. The inertial parameters fed into each link are computed with the aid of SolidWorks, which surely introduces a certain degree of approximation. Plus, the dynamics of the manipulator is being simulated with Simulink's internal simulation engine, which might present some discrepancies with our iterative equations.

From Fig 4.9, it is straightforward to notice the relationship between the time step size and the accuracy of the simulated trajectory. Notice how, for the same step size, the simulation with gravity is more affected by drifting: this can be explained if we consider the solution procedure: the solver acquires the discrete simulation points and interpolates among them with a specific algorithm (*ode45*, *ode23*, *ode113* etc). In the case where gravity is considered, the *blank* intervals among points are vexed by very high torques if compared to the weightless case; this, therefore, leads to a faster drift from the theoretical trajectory and will eventually drive the simulation to divergence.



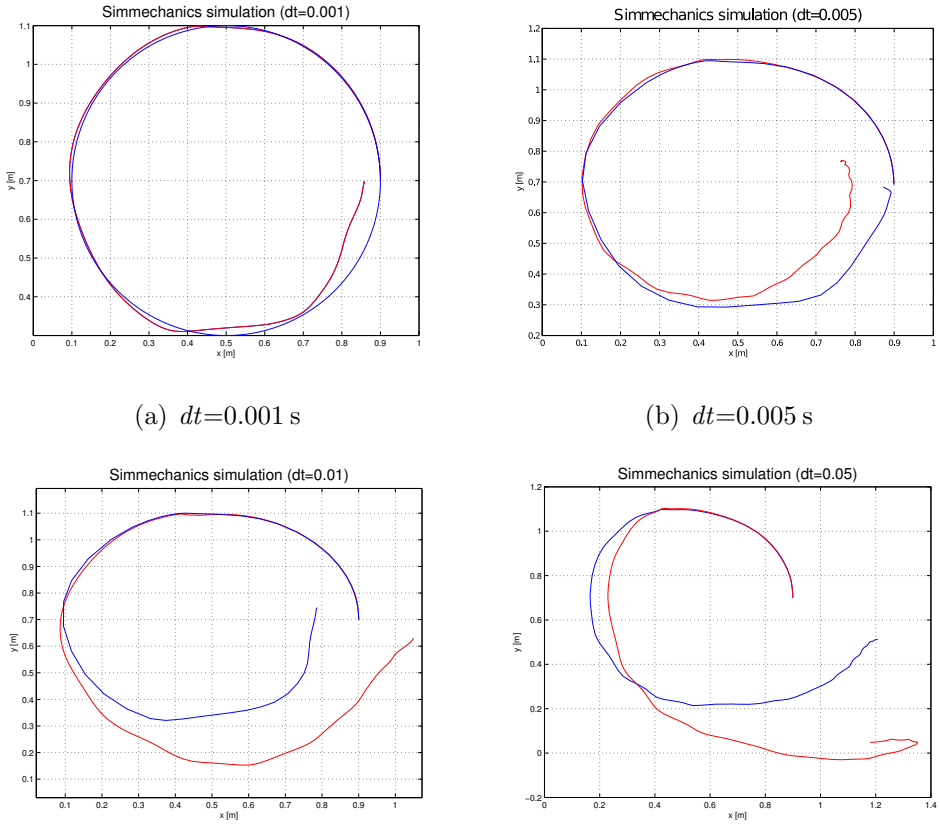
(a) dt=0.001 s

(b) dt=0.005 s

(c) dt=0.01 s

(d) dt=0.05 s

Figure 4.8: *Simmechanics* trajectory simulation for T=*10 s*. Blue line represents the case with g=*0*, whereas red line accounts for g=*9.81* m/s^{-2}.



(a) dt=0.001 s

(b) dt=0.005 s

(c) dt=0.01 s

(d) dt=0.05 s

Figure 4.9: *Simmechanics* trajectory simulation for T=*10 s*. Blue line represents the case with g=*0*, whereas red line accounts for g=*9.81* m/s^{-2}.

Chapter 5

Linear Feedback Control

In the previous chapters, trajectory planning techniques have been presented which allow the generation of the inputs to the motion system. The problem of controlling a robot can be formulated as that to determine the time history of the generalized forces (forces or torques) to be developed by the joint actuators, in order to guarantee the execution of the predefined tasks.

The problem of motion control of a manipulator is the topic of this chapter. Several techniques are available, and their main distinction is due to the way the operate: joint space or operational space. The most common techniques, due to their simplicity, belong to the first category and can be further divided according to the approach taken towards the dynamic contributions. In the following paragraph, both families of control are presented, with a particular focus on joint space procedures [29, 4, 22, 23].

5.1 Joint space control

In joint space techniques, the control is focused on the *q(t)* values so that the actual motion track the reference inputs. These inputs are calculated from the desired trajectory with the aid of an inverse kinematics procedure (refer to Chapter 2 for an overview of the various IK approaches).

However, this solution has the drawback that a joint space control does not have effects on the operational space variables, which are controlled in an open-loop fashion through the manipulator mechanical structure. It follows

quite clearly that any uncertainty in the structure (backlash, play, stiffness) or any dicrepancy between the calculated geometric data and the actual ones causes a decrease of the accuracy on the operational space trajectory.

In approaching the control design, it is fundamental to frame the problem properly. In a control system design process, in fact, several parameters are required in order to model (and also simplify) the procedure. First of all, it is mandatory to know the mechanical design of the structure (the control of a cartesian manipulator, for example, would be substantially different from the control of an anthropomorphic one).

Furthermore, the way the motion is transferred through the joints has its influence as well; if the motors, for example, are coupled with high-ratio reduction gears, it is possible to linearize the problem. This means that the analysis of the joints can take advantage of the effects superposition, and the solution is dramtically simplified. The disadvantage of this approximation is that all the nonlinear effects (such as friction, backlashes, elasticity) might affect the performances of the control.

5.1.1 Decentralized control

In our case, all the motors[1] feature a reduction gear, whose transmission ratio is relatively high. In this condition, the linear approximation can take place. A control of this type is often reffered to as *decentralized control* [29], since each linked is analyzed as a SISO independent system.

We recall, from the dynamic analysis, the differential equations describing the motion of a n degrees of freedom robot [32].

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = \tau \tag{5.1}$$

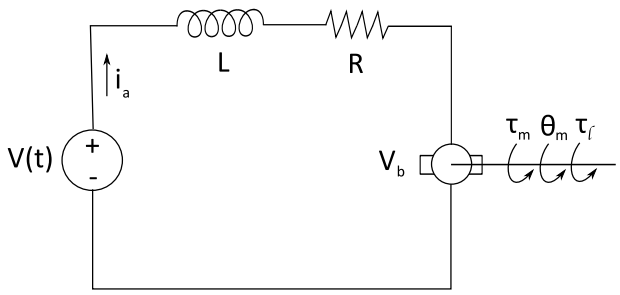
This represents the dynamics of an interconnected chain of bodies when a generalized force τ is acting. This torque is produced by an actuator, which can be electric, hydraulic or pneumatic.

Even if this expression is quite complicated, it does not take into considerations several dynamic effects which are present in the real case. Friction,

[1] Except from *joint 6*, whose contribution to the coupling effect is nonetheless very small.

flexibility and backlash, for example, are accounted. In this chapter only the dynamics of permanent DC motors will be treated, but the analysis can be further extended to other families of actuators.

An armature controlled DC motor presents the following electric diagram [29]:



Due to the presence of a movable rotor inside the stator (which creates a radial magnetic flux Φ), if there is a current i_a flowing, there will be a torque on the rotor:

$$\tau_m = K_1 \, \Phi \, i_a \tag{5.2}$$

When the rotor starts to rotate, however, an electromagnetic field arises (back **emf**), trying to oppose the current flow in the conductor. This can be expressed with:

$$V_b = K_2 \, \Phi \, \omega_m \tag{5.3}$$

The differential equation for the armature current is:

$$L \frac{di_a}{dt} + R \, i_a = V - V_b \tag{5.4}$$

Since the flux Φ is constant, we can rewrite τ_m as (where K_m is the torque constant of the motor):

$$\tau_m = K_1 \, \Phi \, i_a = K_m \, i_a \tag{5.5}$$

From Eq 5.3, with K_b being the back **emf** constant, we have:

$$V_b = K_2 \, \Phi \, \omega_m = K_b \, \omega_m = K_b \frac{d\theta_m}{dt} \tag{5.6}$$

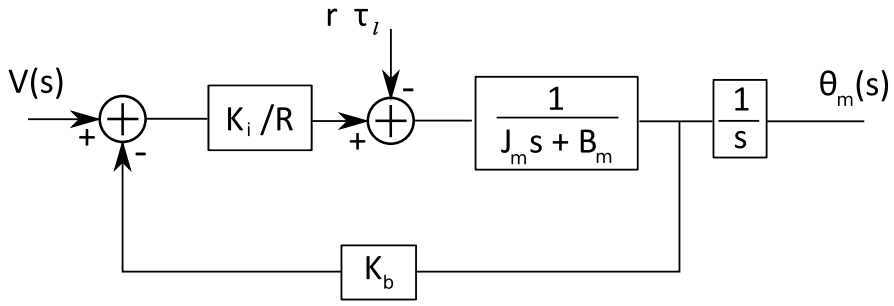


Figure 5.1: Block diagram of DC motor.

When the motor stalls, the corresponding torque is denoted with τ_0; evaluating Eq 5.4 for V_b=0 and $\frac{di_a}{dt}$=0:

$$V_r = R\, i_a = \frac{T\, \tau_0}{K_m} \tag{5.7}$$

Which yields:

$$K_m = \frac{R\, \tau_0}{V_r} \tag{5.8}$$

If we couple the motor with a gear train, the differential equation describing the assemby is:

$$J_m\, \frac{d^2\theta_m}{dt^2} + B_m\, \frac{d\theta_m}{dt} = \tau_m - r\, \tau_l \tag{5.9}$$

Where $I_m = I_a + I_g$, that is, the sum of the actuator and the geartrain inertias. The torque at the output of the gear is τ_l. The block diagram of the DC motor is pictured in Fig 5.1.

At this point, we can switch from the time domain to the Laplace domain, and rewrite Eq 5.6 and Eq 5.9 as:

$$(Ls + R)\, I_a(s) = V(s) - K_b\, s\, \Theta_m(s) \tag{5.10}$$

$$(J_m\, s^2 + B_m\, s)\, \Theta_m(s) = K_i\, I_a(s) - r\, \tau_l(s) \tag{5.11}$$

It is straightforward to obtain the transfer function between the armature voltage $V(s)$ and the angle $\Theta_m(s)$ (imposing $\tau_l = 0$):

$$\frac{\Theta(s)}{V(s)} = \frac{K_m}{s[(L\, s + R)(J_m\, s + B_m) + K_b\, K_m]} \tag{5.12}$$

If $V = 0$, the transfer function between the torque τ_l and $\Theta_m(s)$ is:

$$\frac{\Theta(s)}{\tau_l(s)} = \frac{-r\,(L\,s + R)}{s[(L\,s + R)(J_m\,s + B_m) + K_b\,K_m]} \tag{5.13}$$

Dividing everything by R and assuming that the electrical constant L/R is much smaller than the mechanical constant J_m/B_m, we approximate the previous expressions to:

$$\frac{\Theta(s)}{V(s)} = \frac{K_m/R}{s\left(J_m\,s + B_m + \dfrac{K_b\,K_m}{R}\right)} \tag{5.14}$$

And:

$$\frac{\Theta(s)}{\tau_l(s)} = -\frac{r}{s\left(J_m\,s + B_m + \dfrac{K_b\,K_m}{R}\right)} \tag{5.15}$$

Returning for a moment to the time domain, Eq 5.15 and 5.16 can be expressed, using the superposition of the effects, with the following differential linear equation:

$$J_m\,\ddot{\theta}_m(t) + (B_m + K_b\,K_m/R)\,\dot{\theta}_m(t) = (K_m/R)\,V(t) - r\,\tau_l(t) \tag{5.16}$$

At this point, we need to provide further assumptions and simplifications in order to obtain the solution. Since the output of the gear is directly connected to the link, then the generalized coordinate q_i is given by (with r_i being the *i-th* reduction ratio):

$$q_i = r_i\,\theta_{mi} \tag{5.17}$$

It follows that the torques given by the actuators and the load torques of the actuators share the following relationship:

$$\tau_{li} = \tau_i \tag{5.18}$$

Finally, the equations of motion of the manipulator become:

$$\sum_{j=1}^{n} d_{ji}(\mathbf{q})\ddot{q}_j + \sum_{j,k=1}^{n} c_{jki}(\mathbf{q})\dot{q}_j\dot{q}_k + g_i(\mathbf{q}) = \tau_i \tag{5.19}$$

$$J_m\,\ddot{\theta}_{m_i} + (B_m + K_b\,K_m/R)\,\dot{\theta}_{m_i} = (K_m/R)\,V_i - r_i\,\tau_{l_i} \tag{5.20}$$

If we take a closer look to the last two equations, we can note that the first one represents the nonlinear inertial, Coriolis, centripetal and gravitational coupling contributions due to the motion of the robot, whereas the second one models the actuator dynamics.

If we have to control this kind of system, a first good consideration would be to treat the nonlinear term τ_i as a disturbance entering into Eq 5.20: this is extremely convenient, since Eq 5.20 is linear.
After this substitution, however, the term $r_i^2 d_i i(\mathbf{q})$ appears in the coefficient $\ddot{\Theta}_{m_i}$, which hence becomes:

$$J_m + r_i^2 d_i i(\mathbf{q}) \tag{5.21}$$

That is, this coefficient is configuration dependent. For the purpose of the control, however, we can approximate this value with an effective value, called **effective inertia** J_{eff}. For the moment, we can suppose J_{eff} to be the simple mean average between the value of the inertia at its minimum (J_{min}) and at its maximum (J_{max}), that is:

$$J_{eff} = \frac{J_{min} + J_{max}}{2} \tag{5.22}$$

We also define B_{eff} as:

$$B_{eff} = B_m + K_b \, K_m / R \tag{5.23}$$

$$K = \frac{K_m}{R} \tag{5.24}$$

Thus, Eq 5.20 becomes:

$$J_{eff} \, \ddot{\theta}_{m_i} + B_{eff} \, \dot{\theta}_{m_i} = K \, V_i - r_i \, d_i \tag{5.25}$$

In which d_i is taken as a disturbance and is made up by:

$$d_i := \sum_{j \neq i}^{n} d_{ji} \ddot{q}_j + \sum_{j,k}^{n} c_{jik} \dot{q}_j \dot{q}_k + g_i \tag{5.26}$$

Translating this result into a block diagram, we finally obtain the scheme of Fig 5.2, which is clearly an open loop system.

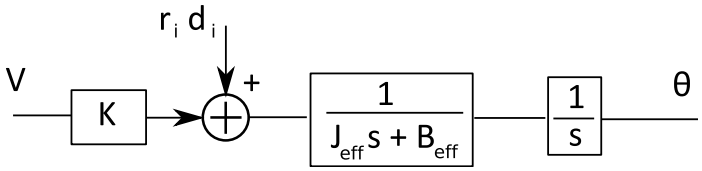


Figure 5.2: Open loop block diagram of manipulator link.

5.1.2 Design of the PD compensator

Once the equations describing the system are obtained, the open loop transfer function in the Laplace domain is immediately obtained [10]:

$$s^2 \, J_{eff} \, \Theta(s) + s \, B_{eff} \, \Theta(s) = K \, V(s) - r \, D(s) \tag{5.27}$$

The input $V(s)$ can be substituted by a PD control law:

$$V(s) = K_p \, [\Theta_r(s) - \Theta(s)] - s \, K_d \, [\Theta(s)] \tag{5.28}$$

Where $\Theta_r(s)$ is the reference command that needs to be followed by the system. Combining these two expressions, we get:

$$\Theta_m(s) = \frac{K \, K_p}{\alpha(s)} \, \Theta_r(s) - \frac{r}{\alpha(s)} \, D(s) \tag{5.29}$$

With $\alpha(s)$ being the characteristic equation:

$$\alpha(s) = J_{eff} \, s^2 + (B_{eff} + K \, K_d) \, s + K \, K_p \tag{5.30}$$

The feedback control loop can then be described by the block in Fig 5.3.

Stability analysis

From the characteristic equation, it is possible to use Routh-Hurwitz criterion in order to analyze the stability of the system [10].

We recall that a stable system needs to have no poles in right hand section of the Re-Im plane. In other words, the real part *Re* has to be smaller or equal to zero.

Routh-Hurwitz criterion states that *the number of roots of the characteristic equation $\alpha(s)$ with Re≥0 is equal to the number of changes in sign of*

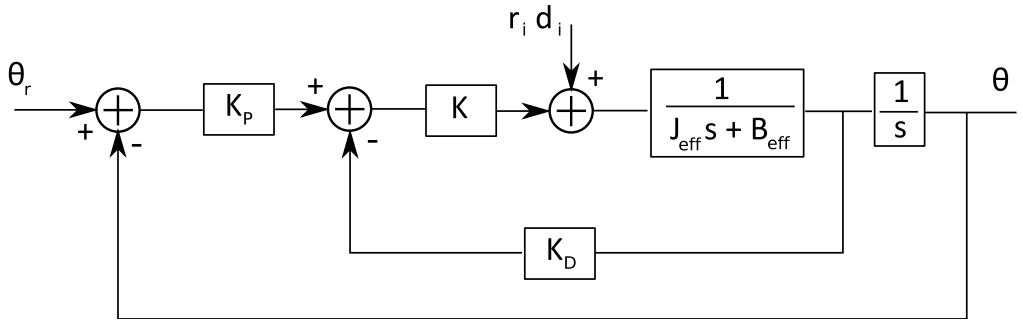


Figure 5.3: Block diagram of PD control system.

the first column of the Routh array [10]. In our case, we have a second order system whose generic equation can be written as:

$$q(s) = a_0 + a_1\, s + a_2\, s^2 \tag{5.31}$$

It follows that the Routh array is:

$$\begin{array}{c|cc} s^2 & a_2 & a_0 \\ s^1 & a_1 & 0 \\ s^0 & b_1 & 0 \end{array} \tag{5.32}$$

Where:

$$b_1 = \frac{a_1\, a_0 - (0)\, a_2}{a_1} = -\frac{1}{a_1} \cdot \begin{vmatrix} a_2 & a_0 \\ a_1 & 0 \end{vmatrix} = a_0 \tag{5.33}$$

Hence, the first column is made up by the coefficients of the equation: for stability to be satisfied, they must all be positive (or all negative). In our case, therefore, the system will be stable as long as the values of K_p and K_d are positive (all the other parameters are already positive).

Tracking error

From the block diagram, the tracking error can be expressed as:

$$e(s) = \Theta_r(s) - \Theta(s) \tag{5.34}$$

$$e(s) = \frac{J_{eff}\, s^2 + (B_{eff} + K\, K_d)\, s}{\alpha(s)}\, \Theta_r(s) + \frac{r}{\alpha(s)}\, D(s) \tag{5.35}$$

For a step reference input and a constant disturbance:

$$\Theta_r(s) = \frac{\Theta_r}{s} \tag{5.36}$$

$$D(s) = \frac{D}{s} \tag{5.37}$$

The tracking error can be calculated by applying the final value theorem:

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} s \cdot e(s) = \frac{-r\ D}{K\ K_p} \tag{5.38}$$

The system is a *Type 1*: that is, for a step input, there will be a steady state error that increases with the gear reduction ratio and (obviously) with the magnitude of the disturbance; on the other hand, it can be reduced by increasing the proportional gain K_p.
Note that $D(s)$ won't be necessarely constant; nevertheless, this simplified model gives a good description of the physical system.

Performances and tuning

Since we have a second order system, the response will influenced by the natural frequency ω and the damping ratio ξ of the closed loop characteristic equation, which can be rewritten as:

$$s^2 + 2\xi\omega\ s + \omega^2 \tag{5.39}$$

$$s^2 + \frac{(B_{eff} + K\ K_d)}{J_{eff}}\ s + \frac{K\ K_p}{J_{eff}} \tag{5.40}$$

The proportional and derivate gains can be extrapolated and yield:

$$K_p = \frac{\omega^2\ J_{eff}}{K} \qquad K_d = \frac{2\,\xi\,\omega\ J_{eff} - B_{eff}}{K} \tag{5.41}$$

We notice that these gains depend only on the choice of ω and ξ. The damping ratio affects the oscillatory response, and it's usually chosen $\xi = 1$ in order to have a critically damped system, i.e. the fastest response with no oscillations. Thus, the overall response of the system will depend only on the value of the natural frequency ω.

We provide an example of this procedure referring to *link 1*. Its effective inertia can be inferred by calculating the inertias in the minimum and maximum case. The minimum case occurs when the arm is perpendicular to the ground, the maxium occurs when the arm is fully extended.

link	J_{link}	$J_{min,\,tot}$	$J_{max,\,tot}$	J_{eff}	Unit
1	$2.623{\cdot}10^{-3}$	0.102	3.682	1.89	kg·m^2

As far as concerns the motor constants, from the EC90 motor's datasheet we can write:

$$K_m = 0.217\ Nm\ A^{-1}$$

$$K = \frac{K_m}{R} = 0.0943\ Nm\ V^{-1}$$

$$K_b = 4.61\ rad\ s^{-1}\ V^{-1}$$

$$B_m = 2.05 \cdot 10^{-2}\ \ kg\ m^2\ s^{-1}$$

$$r = 43$$

For a step input of $\theta_r = 10°$, and with no disturbances ($d = 0$), the system shows, for various values of ω, the characteristics reported in Table 5.1. The corresponding behavior is plotted in Fig 5.4. Note that, since we set

$\omega[rad/s]$	K_p	K_d	e_{ss} [%]	settling time [s]
1	20.04	35.25	0	4
2	80.17	75.34	0	2
4	320.72	155.53	0	1
8	$1.28{\cdot}10^3$	$3.16{\cdot}10^2$	0	0.5

Table 5.1: Step response parameters for system with no disturbances.

$D(s) = 0$, the error at steady state is zero. The same system for a disturbance input of $D = 1$, is characterized by the values in Table 5.2. The step response, in this case, in pictured in Fig 5.5.

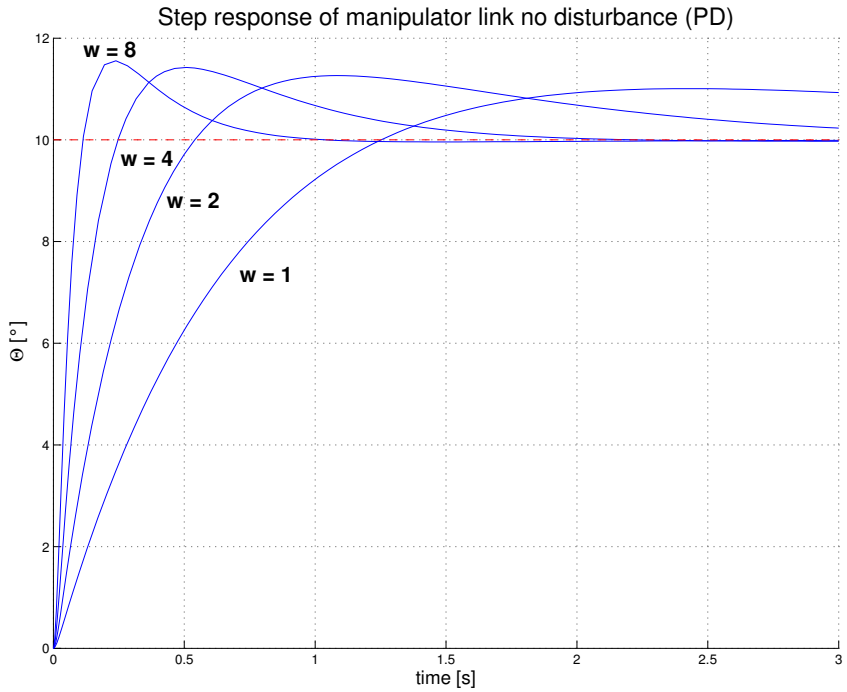


Figure 5.4: Time response of the system (with zero disturbances) for different ω.

$\omega[rad/s]$	K_p	K_d	$e_{ss}[\%]$	settling time [s]
1	20.04	35.25	127.53	4
2	80.17	75.34	56.88	2
4	320.72	155.53	14.24	1
8	$1.28{\cdot}10^3$	$3.16{\cdot}10^2$	3.55	0.5

Table 5.2: Step response parameters for system with disturbances.

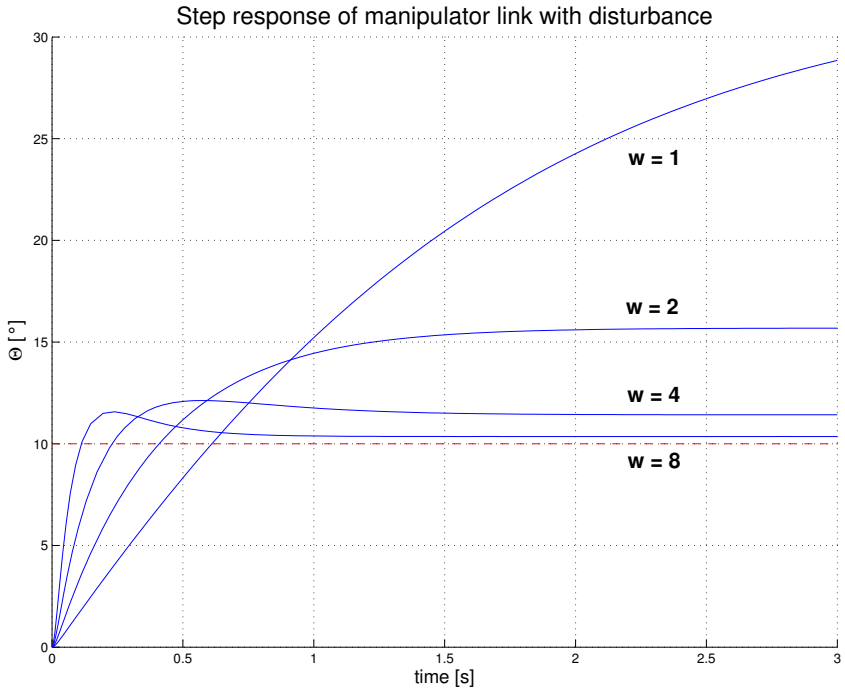


Figure 5.5: Time response of the system with nonzero disturbance for different ω.

In this last figure, it is evident that the steady state error is not longer zero. The amplitude of the drift will depend on the intesity of the disturbance $D(s)$ and on the value of the gains: in the picture, in fact, the disturbace was kept constant while the gain were increased (due to the increase in ω): the higher the gains, the smaller the steady state errors.

Apart from disturbances, it is important to remember that the motors can't provide infinite velocity and acceleration, and they have a so called *saturation limit*: beyond this point, an increase in the requested input will not be satisfied, since the motor will provide the maximum output constantly.

This shows the limited capability of the gains: there is a point beyond which a further increase of the gains won't result in a faster response; on the contrary, it will create an higher overshoot.

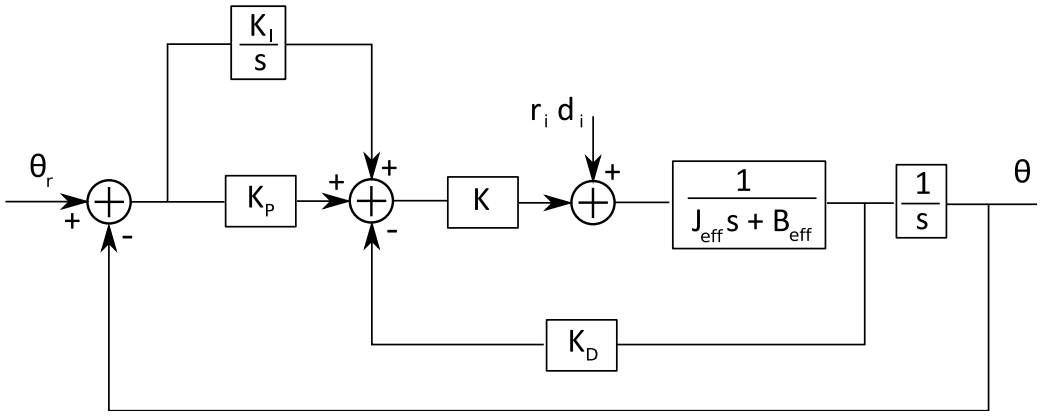


Figure 5.6: Block diagram of PDI control system.

5.1.3 Design of the PID compensator

In the previous section, we were able to acknowledge the limits of the PD controller for our particular system: it is sensitive to external disturbances and in order to limit the steady state error, high gains are needed. On the other hands, motor saturation limits the values of these gains.

An interesting upgrade to the previous system would be the addition of an integral term to the PD compensator law $C(s)$:

$$C(s) = K_p + K_d\, s + \frac{K_i}{s} \tag{5.42}$$

As far as concerns the closed loop expression, Eq 6.30 gets modified into:

$$\Theta_m(s) = \frac{K_d\, s^2 + K_p\, s + K_i}{\beta(s)}\, \Theta_r(s) - \frac{r\, s}{\beta(s)}\, D(s) \tag{5.43}$$

The characteristic equation, in this case, is the following 3^{rd} order polynomial:

$$\beta(s) = J_{eff}\, s^3 + (B_{eff} + K\, K_d)\, s^2 + K\, K_p\, s + K\, K_i \tag{5.44}$$

And the modified block diagram is pictured in Fig 5.6. Note the addition of the feedforward integral part.

Stability analysis

Even in this case, the stability can be inferred from the characteristic equation by using Routh-Hurwitz criterion. For a third order system, whose

generic equation is:

$$q(s) = a_0 + a_1\, s + a_2\, s^2 + a_3\, s^3 \tag{5.45}$$

We have that the Routh array is:

$$\begin{array}{c|cc} s^3 & a_3 & a_1 \\ s^2 & a_2 & a_0 \\ s^1 & b_1 & 0 \\ s^0 & c_1 & 0 \end{array} \tag{5.46}$$

Where:

$$b_1 = \frac{a_2\, a_1 - a_0\, a_3}{a_2} \qquad c_1 = \frac{b_1\, a_0}{b_1} = a_0 \begin{vmatrix} a_2 & a_0 \\ a_1 & 0 \end{vmatrix} = a_0 \tag{5.47}$$

It follows that, since stability occurs when all the elements of Routh matrix's first column are positive, the system is stable if:

$$a_2\, a_1 > a_0\, a_3 \tag{5.48}$$

Hence:

$$K_i < \frac{(B_{eff} + K\, K_d)}{J_{eff}} \cdot K_p \tag{5.49}$$

Performances and tuning

It is possible to integrate the PD case with the integral part of the controller. Since the value of K_i is not already determined by the equations, we can impose a semi-random value that satisfies Eq 5.49. For example:

$$K_i = 0.05 \cdot \frac{(B_{eff} + K\, K_d)}{J_{eff}} \cdot K_p \tag{5.50}$$

With all the parameters set, we can first of all simulate the case in which no disturbances are present. The parameters in this case are described in Table 5.3. The time response for the different frequency choices is presented in Fig 5.7. It can be seen that the behavior and the parameters are the same of the PD case.

The power of the PID controller arises when the motion with disturbances is analyzed. For the usual disturbance $D = 1$, in fact, we obtain the data in

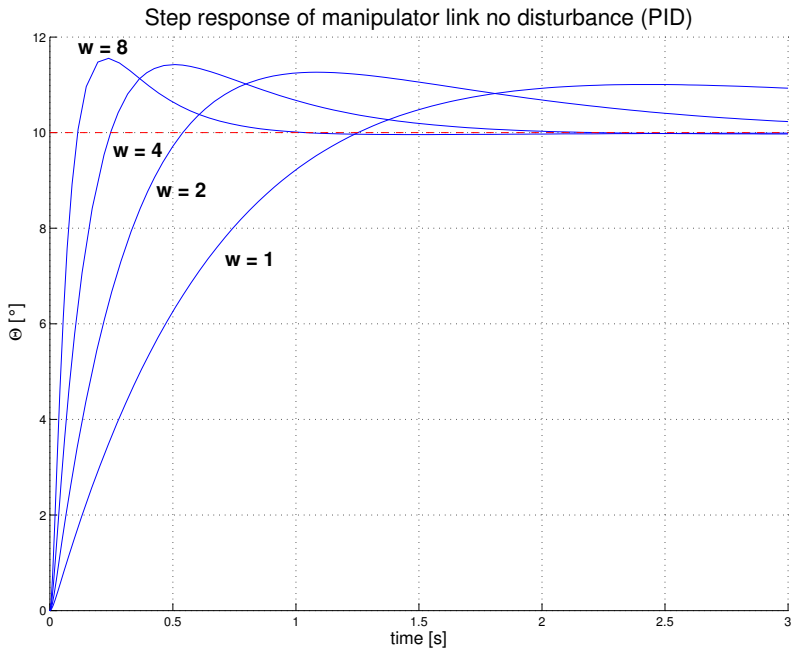


Figure 5.7: Time response of the PID system with zero disturbance for different ω.

$\omega[rad/s]$	K_p	K_d	$e_{ss}[\%]$
1	20.04	35.25	127.53
2	80.17	75.34	56.88
4	320.72	155.53	14.24
8	$1.28{\cdot}10^3$	$3.16{\cdot}10^2$	3.55

Table 5.3: Step response parameters for system with disturbances.

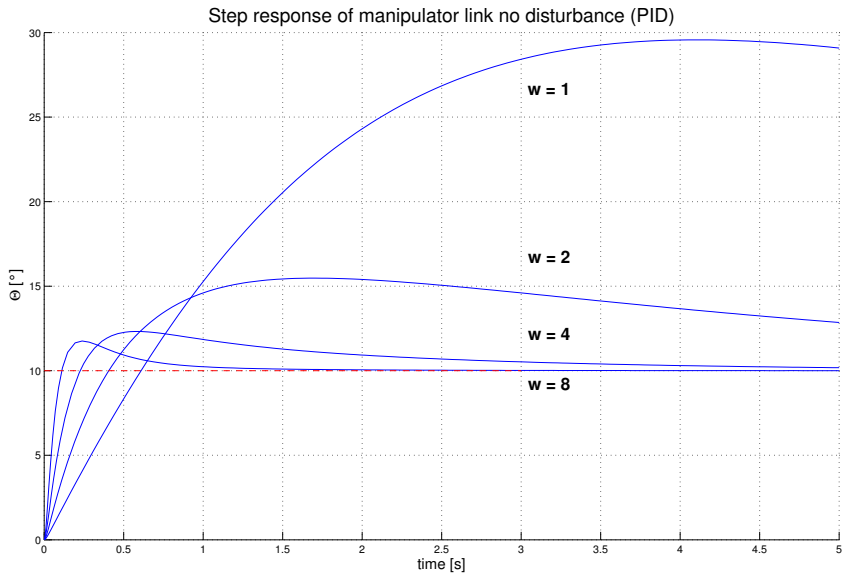


Figure 5.8: Time response of the PID system with nonzero disturbance for different ω.

Table 5.4. The time response to the step is presented in Fig 5.8 and it can be seen that, after the overshoot, the angle tends to θ_r. The error at steady state, hence, is zero.

$\omega[rad/s]$	K_p	K_d	$e_{ss}[\%]$
1	20.04	35.25	0
2	80.17	75.34	0
4	320.72	155.53	0
8	$1.28{\cdot}10^3$	$3.16{\cdot}10^2$	0

Table 5.4: Step response parameters for system with disturbances.

This yields an important result: the addition of the integral part to the control law is mandatory if we are looking for a system which is able to reject external disturbances (that are always present).

5.1.4 Extension to a multibody system

The previous analysis of the PD and PID control system concerned the control of a single link. In the case of a multibody structure, as in our case, the problem can be solved by invoking the linearity of the model (if the hypoteses on the high gear ratio and the slow dynamics are fulfilled [4]).

This extension is fairly straightforward, since any dependency among the bodies has been removed. Hence, every link will be modeled by following the approach explained in the previous paragraphs, and the gains will be tuned in order to obtain the best overall performances.

With these simplifications, the only actual way to verify the performances of system would be to simulate a control and to post-analyze the results; this is due, first of all, to the fact that the inertia seen by each link has been approximated with the effective inertia J_{eff}, even if this parameter is clearly configuration dependent.

Moreover, the input disturbance, that should take into account all the nonlinear effects, cannot be known exactly at each step, and an educated guess on its value has to be made, introducting another relevant source of uncertainties. Note that, again, the slower the dynamics of the object, the better this simplified model will control the system.

Adaptive control

A cunning refinement of this system would be the introduction of an adaptive control. In such a technique, the gains of the controller are not fixed, but they change according to the manipulator configuration.

Using a model with a single averaged value J_{eff} of the inertia, in fact, results in some obvious disadvantages. The control is well tuned when J_{real} is equal to J_{eff}, but when we approach the maximum or the minimum inertia, unpleasant consequences arise: if $J_{real} > J_{eff}$, the manipulator will have a time delay; if $J_{real} < J_{eff}$, the motion will be jerky.

One technique that can be implemented in order to avoid these side effects, is to identify several working configurations and to calculate the effective inertias in those subcases. Then, with these data, it is possible to calculate the optimal gains for ech subcase, which will be then stored in a

look-up table.

During the simulation, a sensing system will detect in which of the working configurations the robot is and will extrapolate the computed optimal gains from the table. It is clear that this is still an approximation, but it's way less coarse than the original approach, thus limiting the delay and jerk phenomena.

5.2 Operational space control

The operational space control enables the manipulator to obtain a greater degree of precision in the cartesian space: that is, the end effector position is actively controlled and is not longer a byproduct of the accuracy with which the geometry of manipulator is known.

However, this global approach requires a greater complexity; notice, in fact, that the inverse kinematics algorithm is now embedded into the feedback control loop. This slows down the algorithm and requires higher computational perfomances. Moreover, the abovementioned advantage on the end effector position presents actually an obvious limit. The measurement of the cartesian variables, in fact, is not always[2] performed directly, but via the application of the direct kinematic algorithm to the encoders' readings.

Hence, since this technique does not clear the need of a having good knowledge of the robot parameters, it doesn't makes sense to go to the trouble of implenting such a complicated and CPU consuming control law. The need for the extra computing power to run the model at a sufficient rate might not be worthwhile.

The most common industrial robots, for economic reasons, do not use this technique: instead, present-day manipulators are controlled with very simple control laws that generally are just error driven.

For all the above reasons, in this section we are just going to introduce the schematics of the principal control blocks without diving too deep into the details, leaving any further analysis to the appropriate references [23, 22, 4, 8].

[2]This is not valid if there exists a cartesian sensor which avoids the need for the direct kinematics transformation: for example, cameras or vision sensors.

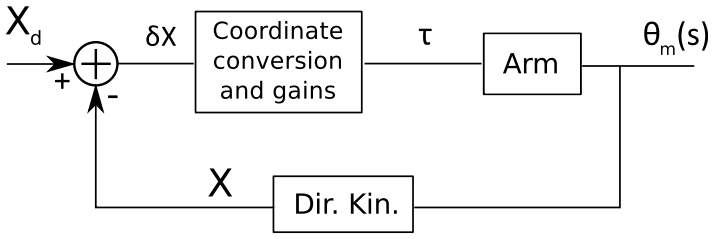


Figure 5.9: Block diagram of a general cartesian based control loop.

5.2.1 An overview

The general scheme of a control based on operational space techniques is presented in Fig 5.9. The input to the close loop block is not longer the generalized coordinate $q(t)$, but it's simply the cartesian trajectory needed.

Thus, all the cartesian transformation into the joint space variables need to be performed inside the loop; this is an important drawback, that results in a lower sampling frequency if compared to joint based controls, degrading the stability and the disturbance-rejection ability of the loop [8].

Note that, even if we are talking about a cartesian control, the conversion to the joint space is necessary at some point for the calculation of the joint torques.

The procedure starts from the reading of the position sensors in order to gain information on the $\theta(t)$ values. With the aid of simple direct kinematics, the angles are converted to the actual position of the end effector. By knowing the goal coordinates X_d, we can obtain cartesian errors δX:

$$\delta X = X_d - X$$

From these errors, with the aid of a coordinate conversion and some appropriate gains, the torques are then computed and provided to the joints.

It is quite clear that the most important part of the block diagram in Fig 5.9 is the *"Coordinate conversion and gains"* block. In the literature, there are several ways of practically implementing this block [23, 22, 4, 8].

One the most common strategies, recalling the theory from Section 2.5, is to use a Jacobian-type algorithm. If the time step is sufficiently small, in

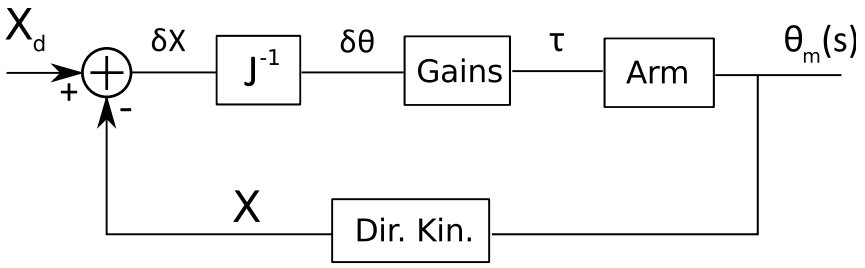


Figure 5.10: The *inverse-Jacobian* cartesian control block.

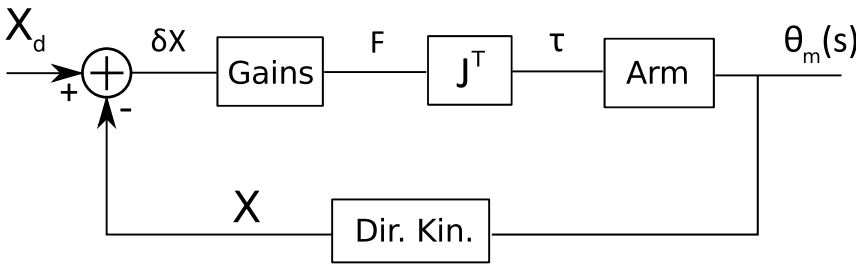


Figure 5.11: The *transpose-Jacobian* cartesian control block.

fact, we can map the cartesian error δX into the corresponding displacement $\delta\theta$ in the joint space. The $\delta\theta$ errors are then multiplied by the appropriate gains to compute the torques that will presumably reduce the errors. This approach takes the name of **inverse-Jacobian controller** and is presented in Fig 5.10 [8].

Another viable solution, that is strictly related to the previous one, is presented in Fig 5.11. In this case, we compute the cartesian error δX and we multiply it by a gain block to obtain a force vector F in the cartesian space. We can think of this force as the vector that, if applied to the end effector, would reduce the error δX. From this vector, we can again use the differential kinematics theory to obtain the solution: F gets multiplied by the transpose of Jacobian, J^T, and the torques are obtained. This approach is referred to as **transpose-Jacobian controller**.

Although the block diagrams look neat and simple, the exact dynamics of these systems is very complicated. It has been shown that both schemes

will work, meaning that it is possible, with the approriate gains, to make the loop stable. This partial success, however, is obscured by the need for adaptive laws: it is not possible to choose some fixed gains and have fixed closed loop poles in all the points of the workspace. The dynamics of these controllers, in fact, is influenced by the arm configuration.

Chapter 6

Space trajectory analyis

In this chapter we analyze the ultimate goal of the manipulator: the need to simulate spacecraft proximity navigation and docking. The previous chapters dealt with the general analysis of the manipulator, which does not actually introduce any innovative argument.

The insteresting application of the robot is to be able to simulate, on a ground laboratory, what happens in space. Thus, the motion needs to account for the effect of the orbital environment, and the trajectory definition must consider the non inertial frame of reference.

Moreover, the most challenging goal of the project is to provide the robot with instrument and software that allows for a real time interaction with the outside world. The dynamics of the contact bewteen two approaching satellites, for example, might create impulsive forces that have a relevant effects on the trajectory. The software, with the aid of a force sensing device, will extrapolate from the transducer the corrected trajectory, and the manipulator will simulate the motion along this new path.

Thereby, it is immediate to foresee the powerful applications of this piece of equipment: apart from docking maneuvers, it is possible to simulate any kind of reaction control system and examine their dynamic and kinematic effects. Nonetheless, it is a powerful test bench for control systems and attitude architectures.

In this chapter, we introduce the theory behind the trajectory simulation and depict all the main applications of the system. A brief overview on force

sensors is then presented and the results of a Matlab simulation campaign are finally discussed.

6.1 Orbital mechanics review

The main goal of the manipulator is to simulate docking and approach bewteen orbiting satellites. In order to simulate these situations on the ground, the modeling needs to faithfully reproduce the dynamic conditions.

When the relative motion of two objects is analyzed, it is fundamental to understand the physics of the space in which they are immersed.

In orbital mechanics, the motion is usually described with reference to an inertial frame, fixed to the center of attraction (center of the planet), so that Newton equation can be used [9]:

$$\mathbf{F}_{net} = m \cdot \mathbf{a}_{abs} \tag{6.1}$$

In laboratory, however, this motion can be simulated only as seen from a relative frame. That is, given two objects (a target and a chaser), it is possible to define the target (for exampe), as the center of the free falling, rotating non inertial frame of reference, considered fixed with respect to the lab ground.

With this approach, rendevous manouvers can be reproduced: it is fundamental, however, to recognize that in this new relative system, we need to take into account the transformation of relative velocity and acceleration.

6.1.1 Relative motion in orbit

In close approach maneuvers, generally, one object (the *target*) is passive and non-maneuvering, whereas the other (the *chaser*), is active and trying to approach the target.

Referring to Fig 6.1, the position of the target in the geocentric frame is given by r_0. The target represents also the origin of the moving frame, whose x axis is along the r_0 direction, y points in the local horizon of the target's orbit and z is chosen to complete a right handed frame.

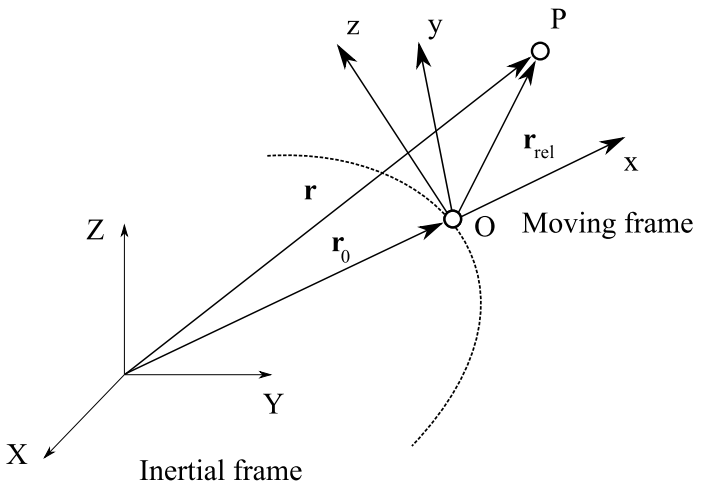


Figure 6.1: Absolute and relative position vectors

In order to analyze the motion, we recall the formulas for relative velocity and acceleration [9]:

$$\mathbf{v} = \mathbf{v}_0 + \mathbf{v}_{rel} + \boldsymbol{\Omega} \times \mathbf{r}_{rel} \tag{6.2}$$

$$\mathbf{a} = \mathbf{a}_0 + \mathbf{a}_{rel} + \dot{\boldsymbol{\Omega}} \times \mathbf{r}_{rel} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{r}_{rel}) + 2\,\boldsymbol{\Omega} \times \mathbf{v}_{rel} \tag{6.3}$$

In these equations, the terms $\boldsymbol{\Omega}$ and $\dot{\boldsymbol{\Omega}}$ need to be computed. The angular moment of the orbit can be calculated as:

$$\mathbf{h} = \mathbf{r}_0 \times \mathbf{v}_0 = (\mathbf{r}_0\,\Omega)\,\hat{\mathbf{z}} = r_0^2\,\boldsymbol{\Omega} \tag{6.4}$$

From which, the angular velocity of the moving frame is:

$$\boldsymbol{\Omega} = \frac{\mathbf{r}_0 \times \mathbf{v}_0}{r_0^2} \tag{6.5}$$

As far as concerns the acceleration $\dot{\boldsymbol{\Omega}}$, we derive the previous equation:

$$\dot{\boldsymbol{\Omega}} = \frac{1}{r_0^2}(\dot{\mathbf{r}}_0 \times \mathbf{v}_0 + \mathbf{r}_0 \times \dot{\mathbf{v}}_0) - \frac{2}{r_0^3}\dot{r}_0(\mathbf{r}_0 \times \mathbf{v}_0) \tag{6.6}$$

Which yields (recalling that $\dot{\mathbf{r}}_0 \times \mathbf{v}_0 = 0$ and $\mathbf{r}_0 \times \dot{\mathbf{v}}_0 = 0$[1]):

$$\dot{\boldsymbol{\Omega}} = -\frac{2}{r_0}\dot{r}_0\boldsymbol{\Omega} \tag{6.7}$$

[1]Note that $\dot{\mathbf{r}}_0 \times \mathbf{v}_0 = \mathbf{v}_0 \times \mathbf{v}_0 = 0$. As far as concerns the second equation: $\dot{\mathbf{v}}_0 = -\frac{\mu}{r_0^3}\mathbf{r}_0$. Hence: $\mathbf{r}_0 \times \dot{\mathbf{v}}_0 = \mathbf{r}_0 \times \left(-\frac{\mu}{r_0^3}\mathbf{r}_0\right) = -\frac{\mu}{r_0^3}(\mathbf{r}_0 \times \mathbf{r}_0) = \mathbf{0}$.

Finally, since $\dot{r}_0 = \mathbf{v}_0 \cdot \mathbf{r}_0 / r_0$:

$$\dot{\mathbf{\Omega}} = -\frac{2(\mathbf{r}_0 \times \mathbf{v}_0)}{r_0^2}\mathbf{\Omega} \tag{6.8}$$

By placing equations Eq 6.5 and 6.8 inside Eq 6.2 and 6.3, one can calculate the relative velocity and accelerations of an object measured along the frame centered in the target.

Linearization

We start by recalling that the inertial acceleration of the chaser is given by:

$$\ddot{\mathbf{r}} = -\mu\frac{\mathbf{r}}{r^3} \tag{6.9}$$

From this, since $\mathbf{r} = \mathbf{r}_0 + \mathbf{r}_{rel}$, we can write that:

$$\ddot{\mathbf{r}}_{rel} = -\ddot{\mathbf{r}}_0 - \mu\frac{\mathbf{r}_0 + \mathbf{r}_{rel}}{r^3}. \tag{6.10}$$

Referring to Fig 6.1, if $\mathbf{r}_{rel}$ is much smaller than $\mathbf{r}_0$ (which is the case of a close approach maneuver), then Eq 6.10 can be linearized as follows:

$$\ddot{\mathbf{r}}_{rel} = -\frac{\mu}{r_0^3}\left[\mathbf{r}_{rel} - \frac{3}{r_0^2}(\mathbf{r}_0 \cdot \mathbf{r}_{rel})\mathbf{r}_0\right] \tag{6.11}$$

Since $\mathbf{R} = R\,\hat{\mathbf{k}}$ and $\mathbf{r}_{rel} = \delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}} + \delta z\hat{\mathbf{k}}$, we can further simplify Eq 6.11:

$$\ddot{\mathbf{r}}_{rel} = -\frac{\mu}{r_0^3}(-2\delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}} + \delta z\hat{\mathbf{k}}) \tag{6.12}$$

To avoid confusion, note that this is the linearized acceleration of the chaser with respect to the geocentrical frame; our goal, on the other hand, is to obtain the motion equations with reference to the target centered frame. This means plugging Eq 6.12 into Eq 6.3.
Omitting the tedious algebraic calculations, we can write the final expression for the relative acceleration:

$$\begin{aligned}\delta\mathbf{a}_{rel} = &-\frac{\mu}{r_0^3}(-2\delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}} + \delta z\hat{\mathbf{k}}) - \frac{2(\mathbf{V}\cdot\mathbf{r}_0)h}{r_0^4}(\delta y\hat{\mathbf{i}} - \delta x\hat{\mathbf{j}}) \\ &+ \frac{h^2}{r_0^4}(\delta x\hat{\mathbf{i}} + \delta y\hat{\mathbf{j}}) - 2\frac{h}{r_0^2}(\delta\dot{x}\hat{\mathbf{j}} - \delta\dot{y}\hat{\mathbf{i}})\end{aligned} \tag{6.13}$$

Its components are then:

$$\begin{cases} \delta\ddot{x} - \left(\dfrac{2\mu}{r_0^3} + \dfrac{h^2}{r_0^4}\right)\delta x + \dfrac{2(\mathbf{V}\cdot\mathbf{r}_0)h}{r_0^4}\delta y - 2\dfrac{h}{r_0^2}\delta\dot{y} = 0 \\ \delta\ddot{y} - \left(\dfrac{\mu}{r_0^3} - \dfrac{h^2}{r_0^4}\right)\delta y - \dfrac{2(\mathbf{V}\cdot\mathbf{r}_0)h}{r_0^4}\delta x + 2\dfrac{h}{r_0^2}\delta\dot{x} = 0 \\ \delta\ddot{z} + \dfrac{\mu}{r_0^3}\delta z = 0 \end{cases} \tag{6.14}$$

Clohessy-Wiltshire equations

Equations 6.14 describe the relative motion of the chaser in the target frame, which has a generic elliptical orbit around the center body. If this orbit is circular, then:

$$\mathbf{V}\cdot\mathbf{r}_0 = 0 \qquad h = \sqrt{\mu\, r_0} \tag{6.15}$$

And Eq 6.14 become:

$$\begin{cases} \delta\ddot{x} - 3\dfrac{\mu}{r_0^3}\delta x - 2\sqrt{\dfrac{\mu}{r_0^3}}\delta\dot{y} = 0 \\ \delta\ddot{y} + 2\sqrt{\dfrac{\mu}{r_0^3}}\delta\dot{x} = 0 \\ \delta\ddot{z} + \dfrac{\mu}{r_0^3}\delta z = 0 \end{cases} \tag{6.16}$$

These are called Clohessy-Wiltshire (CW) equations and they are relatively simple to solve. With a simple analytical integration, we can obtain the velocity and the position equations:

$$\begin{cases} \delta\dot{x} = 3nsin(nt)\delta x_0 + cos(nt)\delta\dot{x}_0 + 2sin(nt)\delta\dot{y}_0 \\ \delta\dot{y} = 6n[cos(nt) - 1]\delta x_0 - 2sin(nt)\delta\dot{x}_0 + [4cos(nt) - 3]\delta\dot{y}_0 \\ \delta\dot{z} = -nsin(nt)\delta z_0 + cos(nt)\delta\dot{z}_0 \end{cases} \tag{6.17}$$

$$\begin{cases} \delta x = [4 - 3cos(nt)]\delta x_0 + \dfrac{sin(nt)}{n}\delta\dot{x}_0 + \dfrac{2}{n}[1 - cos(nt)]\delta\dot{y}_0 \\ \delta y = 6[sin(nt) - nt]\delta x_0 + \delta y_0 + \dfrac{2}{n}\delta[cos(nt) - 1]\delta\dot{x}_0 + \dfrac{1}{n}[4sin(nt) - 3nt]\delta\dot{y}_0 \\ \delta z = cos(nt)\delta z_0 + \dfrac{1}{n}sin(nt)\delta\dot{z}_0 \end{cases} \tag{6.18}$$

In order to improve the relative motion analysis, the handling of CW equations is made easier with the a matrix notation. First of all, we define:

$$\delta\mathbf{r}(t) = \begin{Bmatrix} \delta x(t) \\ \delta y(t) \\ \delta z(t) \end{Bmatrix} \qquad \delta\mathbf{v}(t) = \begin{Bmatrix} \delta\dot{x}(t) \\ \delta\dot{y}(t) \\ \delta\dot{z}(t) \end{Bmatrix} \tag{6.19}$$

Whose corresponding initial values, for $t = 0$ are:

$$\delta\mathbf{r}_0 = \begin{Bmatrix} \delta x_0 \\ \delta y_0 \\ \delta z_0 \end{Bmatrix} \qquad \delta\mathbf{v}_0 = \begin{Bmatrix} \delta\dot{x}_0 \\ \delta\dot{y}_0 \\ \delta\dot{z}_0 \end{Bmatrix} \tag{6.20}$$

Then, the position and velocity of the chaser at instant t is given by:

$$\begin{Bmatrix} \delta\mathbf{r}(t) \\ \delta\mathbf{v}(t) \end{Bmatrix} = \begin{bmatrix} \mathbf{\Psi}_{rr}(t) & \mathbf{\Psi}_{rv}(t) \\ \mathbf{\Psi}_{vr}(t) & \mathbf{\Psi}_{vv}(t) \end{bmatrix} \cdot \begin{Bmatrix} \delta\mathbf{r}_0 \\ \delta\mathbf{v}_0 \end{Bmatrix} \tag{6.21}$$

Or:

$$\{\delta\mathbf{r}(t)\} = [\mathbf{\Psi}_{rr}(t)]\,\{\,\delta\mathbf{r}_0\} + [\mathbf{\Psi}_{rv}(t)]\,\{\,\delta\mathbf{v}_0\} \tag{6.22}$$

$$\{\delta\mathbf{v}(t)\} = [\mathbf{\Psi}_{vr}(t)]\,\{\,\delta\mathbf{r}_0\} + [\mathbf{\Psi}_{vv}(t)]\,\{\,\delta\mathbf{v}_0\} \tag{6.23}$$

Where:

$$\mathbf{\Psi}_{rr}(t) = \begin{bmatrix} 4 - 3cos(nt) & 0 & 0 \\ 6[sin(nt) - 1] & 1 & 0 \\ 0 & 0 & cos(nt) \end{bmatrix} \tag{6.24}$$

$$\mathbf{\Psi}_{rv}(t) = \frac{1}{n}\begin{bmatrix} sin(nt) & 2[1 - cos(nt)] & 0 \\ [cos(nt) - 1] & [4sin(nt) - 3nt] & 0 \\ 0 & 0 & sin(nt) \end{bmatrix} \tag{6.25}$$

$$\mathbf{\Psi}_{vr}(t) = \begin{bmatrix} 3nsin(nt) & 0 & 0 \\ 6n[cos(nt) - 1] & 0 & 0 \\ 0 & 0 & -nsin(nt) \end{bmatrix} \tag{6.26}$$

$$\mathbf{\Psi}_{vv}(t) = \begin{bmatrix} cos(nt) & 2sin(nt) & 0 \\ -2sin(nt) & 4cos(nt) - 3 & 0 \\ 0 & 0 & cos(nt) \end{bmatrix} \tag{6.27}$$

6.2 CW equations: main applications

Once the equations describing the orbital relative motion of two (or more) object is known from Clohessy Wiltshire expressions, it is possible to sketch out some interesting applications for the robotic manipulator.

6.2.1 Relative free motion simulation

The first, immediate application of CW equations is the free motion simulation. The starting point is to define the target position in the operational space (which will be chosen to maxime the robot dexterity in the neighborhood area); then, the direction of x and y is chosen for the target-centered frame (a good choice would be to choose the orbital plane $x - y$ parallel to the base plane $x_0 - y_0$).

This fully defines the CW environment and, consequentely, the ${}^{0}_{cw}\mathbf{T}$ transformation matrix between this frame and the base robot frame. The robot, in fact, will be provided trajectory information with respect to his frame of reference: the conversion from CW coordinates to the manipulator coordinates can be written as:

$${}^{0}\delta\mathbf{r} = {}^{0}_{cw}\mathbf{T} \cdot {}^{cw}\delta\mathbf{r} \tag{6.28}$$

$${}^{0}\delta\mathbf{r} = \left[\begin{array}{c|c} \mathbf{R}(\alpha, \beta, \gamma) & \mathbf{p}_{cw} \\ \hline \mathbf{0} & 1 \end{array}\right] \cdot {}^{cw}\delta\mathbf{r} \tag{6.29}$$

To start the simulation, information on the orbit are need (since it's a circular orbit, we actually need to know only the altitude); moreover, a starting position and velocity has to be set (that is, vectors $\mathbf{r}_0$ and $\mathbf{v}_0$). At this point, the solution block diagram can be easily inferred:

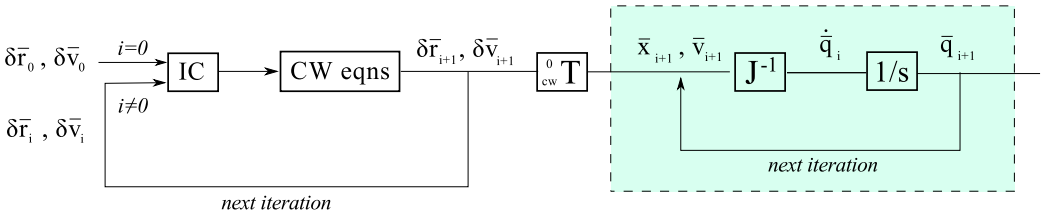


Figure 6.2: Block diagram for the free relative motion simulation.

The diagram is composed essentially by three parts: the first one comprises the CW equations routine, the second consists in the CW frame to base frame conversion and the third one (boxed in green) takes care of the conversion from operational space to joint space (using the inverse differential kinematics, Section 2.5).

Notice that, since the last box does not change among the different cases presented in this chapter, we will further omit it (even if, obviously, it is always present).

6.2.2 Relative motion with quasi-constant disturbances

The motion of the chaser can be subjected to several types of external forces. Their behavior with respect to time might be constant, impulsive or semi-periodic. CW equations allow for the modeling of the motion when disturbances are present.

In order to take them into account correctly, it is fundamental to understand their way of action. Specifically, the duration of these disturbances changes their insertion spot into the CW expressions.

A very fast-acting force, for example, can be approximated with an impulse. Thus, we can suppose that throughout the dynamic phenomenon, the position of the object doesn't change. On the contrary, for a low frequency disturbance, this approximation doesn't stand. In this latter case, the CW

equations can be written as:

$$\begin{cases} \delta\ddot{x} - 3\dfrac{\mu}{r_0^3}\delta x - 2\sqrt{\dfrac{\mu}{r_0^3}}\delta\dot{y} &= \dfrac{F_x}{m_c} \\ \delta\ddot{y} + 2\sqrt{\dfrac{\mu}{r_0^3}}\delta\dot{x} &= \dfrac{F_y}{m_c} \\ \delta\ddot{z} + \dfrac{\mu}{r_0^3}\delta z &= \dfrac{F_z}{m_c} \end{cases} \tag{6.30}$$

Where the F_x, F_y, F_z terms are the disturbance components in the CW frame. Note that, since the equations are expressed in terms of acceleration, the forces need to be divided by the mass of the chaser.

The block diagram, in this case, has the following arrangement:

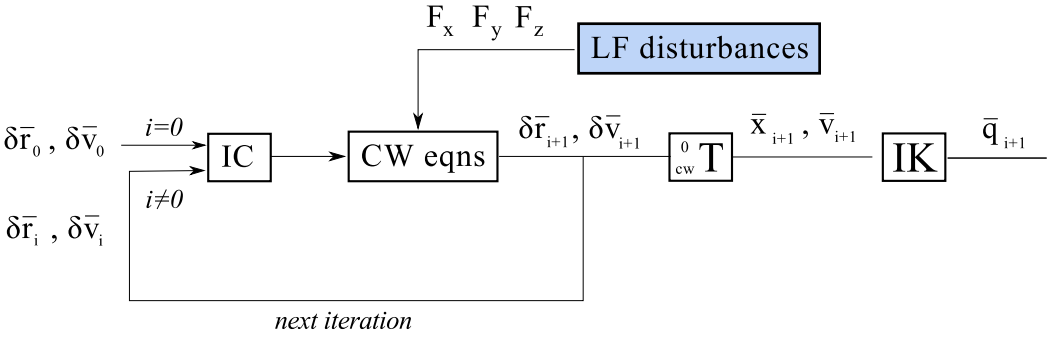


Figure 6.3: Block diagram for the relative motion with quasi-constant disturbances.

6.2.3 Relative motion with impulsive disturbances

When the force acts very quickly, for example in the case of an impact, we can approximate its effect to that of an impulse. In physics, an impulse J is defined as:

$$J = \int_{t_1}^{t_2} F \cdot dt \tag{6.31}$$

If the impulse is acting on a body of mass m, the change in linear velocity is given by:

$$\Delta v = \int_{t_1}^{t_2} \frac{|F|}{m} \cdot dt \tag{6.32}$$

For this first approach, we suppose the force as acting on the object's center of mass, so that no tumbling motion is induced. The sensor will provide the components of the force decomposed along a certain frame of refence fixed with respect to the sensor.

First of all, we need to transform these three components (F_x, F_y and F_z) into Clohessy-Wiltshire frame. In order to do this, we first transform the vector into base frame coordinates (we denote with S the force sensor's frame of reference):

$$^0\mathbf{F} = {}^0_S\mathbf{T}\ {}^S\mathbf{F} \tag{6.33}$$

The transformation from base coordinates to the CW frame is:

$$^{cw}\mathbf{F} = {}^{cw}_0\mathbf{T}\ {}^0\mathbf{F} \tag{6.34}$$

The cumulative transformation is then:

$$^{cw}\mathbf{F} = {}^{cw}_0\mathbf{T}\ {}^0_S\mathbf{T}\ {}^S\mathbf{F} \tag{6.35}$$

Once the frame transformation is completed, the impulse calculation can be carried out. The force data with respect to time will tipically resemble a peak, whose area, recalling Eq 6.31, is the magnitude of the impulse. A numerical integration can be easily implemented to calculate the area underneath the curve.

Having the impulse value, we can immediately obtain the $\Delta\mathbf{v}$: from Eq 6.32, in fact, since the mass is constant (we suppose no mass flow leaving or entering the object), the $\Delta\mathbf{v}$ is simply the impulse J divided by the satellite's mass m. This procedure is graphically illustrated in Fig 6.4.

The three computed components of the $\Delta\mathbf{v}$ will be inserted into Eq 6.37. At time $t = 0^-$, the position $\delta\mathbf{r}_0^-$ and velocity $\delta\mathbf{v}_0^-$ are known. At istant $t = 0^+$, the position doesn't change, whereas there's the velocity jumps to the new value. In formulas, we have:

$$\delta\mathbf{r}_0^+ = \delta\mathbf{r}_0^- \tag{6.36}$$

$$\delta\mathbf{v}_0^+ = \delta\mathbf{v}_0^- + \Delta v_0 \tag{6.37}$$

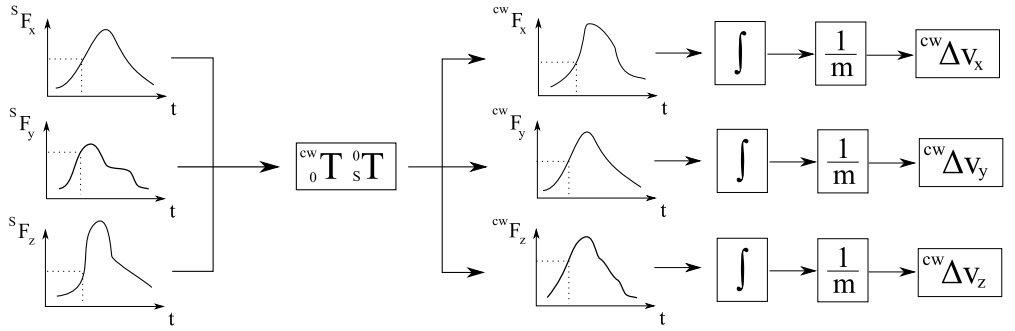


Figure 6.4: Δv components computation from force sensor acquistions.

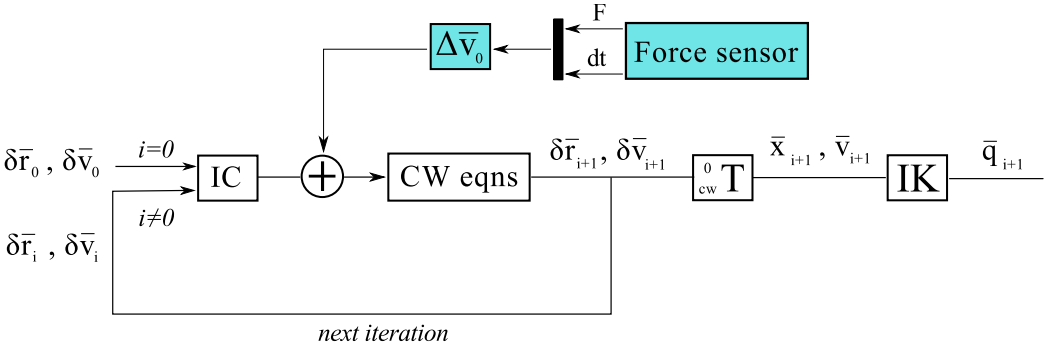


Figure 6.5: Block diagram for the relative motion with impulse disturbances.

Thus, the trajectory is affected by the impulse and this is modeled with the modification of Eq 6.21, which becomes:

$$\begin{Bmatrix} \delta\mathbf{r}(t)^{+} \\ \\ \delta\mathbf{v}(t)^{+} \end{Bmatrix} = \begin{bmatrix} \mathbf{\Psi}_{rr}(t) & \mathbf{\Psi}_{rv}(t) \\ \\ \mathbf{\Psi}_{vr}(t) & \mathbf{\Psi}_{vv}(t) \end{bmatrix} \cdot \begin{Bmatrix} \delta\mathbf{r}_0 \\ \\ \delta\mathbf{v}_0^{+} \end{Bmatrix} \tag{6.38}$$

Where the change on the velocity initial conditions is denoted by the $+$ superscript. The conditions on the position, as said, do not change. The block diagram of this case is presented in Fig 6.5.

6.2.4 Relative motion with ADCS control

The Clohessy-Wilthsire model, due to its elegance in the force/impulses analysis, can be used also to simulate an attitude control system.

That is, we can design an approach maneuver with the knowledge of the initial conditions and the time needed for the rendez-vous. This consists in the calculation of two impulses, at the start and at the end of the trajectory, that will then translated into vectorial forces (for example, which thruster to turn on and for how long).

Moreover, the model can be used for the opposite goal: that is, for a given thrust/reaction, the new trajectory is computed and immediately simulated. This is particularly useful if *on-the-go* corrections have to be made, or if disturbances drift away the trajectory from the planned one. This second aspect of the problem, however, is a direct application of the diagram presented in the previous section, where the force sensor is substituted by the (known) thrust force provided to the ADCS.

As far as concerns the maneuver planning (also known as **two impulses rendezvous**), we start with the definition of the t_f, the time required for the maneuver; this, obviously, is a function of the requirements and is limited by the reaction control system performances.

Thus, in time t_f, we want the chaser to start from initial position A and arrive at target position B, which, recalling the CW model, is also the origin of the local frame of reference. Hence, from Eq 6.22:

$$\{\mathbf{0}\} = [\mathbf{\Psi}_{rr}(t_f)]\,\{\,\delta\mathbf{r}_0\} + [\mathbf{\Psi}_{rv}(t_f)]\,\{\,\delta\mathbf{v}_0^+\} \tag{6.39}$$

Which yields:

$$\{\delta\mathbf{v}_0^+\} = -[\mathbf{\Psi}_{rv}(t_f)]^{-1}[\mathbf{\Psi}_{rr}(t_f)]\,\{\,\delta\mathbf{r}_0\} \tag{6.40}$$

This provides the velocity needed at the beginning, after the first thrust. The corresponding $\Delta\mathbf{v}$ is then:

$$\Delta\mathbf{v}_0 = \delta\mathbf{v}_0^+ - \delta\mathbf{v}_0^- \tag{6.41}$$

The second thrust happens as soon as the target is reached, and is calculated by setting to zero the velocity of the chaser. Since we know the

trajectory, we also know the velocity at the end (i.e. $t = t_f$), which is given by:

$$\left\{\delta \mathbf{v}_f^-\right\} = \left[\mathbf{\Psi}_{vr}(t_f)\right]\left\{\,\delta \mathbf{r}_0\right\} + \left[\mathbf{\Psi}_{vv}(t_f)\right]\left\{\,\delta \mathbf{v}_0^+\right\} \tag{6.42}$$

From Newton's dynamic formula, a velocity with the same magnitude but opposite direction is needed to bring the chaser to rest at point B:

$$\Delta \mathbf{v}_f = \delta \mathbf{v}_f^+ - \delta \mathbf{v}_f^- = \mathbf{0} - \delta \mathbf{v}_f^- = -\delta \mathbf{v}_f^- \tag{6.43}$$

Some doubts may arise from these calculations: how come are differences between relative velocities being used if the delta-v are actually absolute velocities? This can be easily explained by taking a look at the relative velocity formulas:

$$\mathbf{v}^- = \mathbf{v}_0^- + \mathbf{v}_{rel}^- + \mathbf{\Omega}^- \times \mathbf{r}_{rel}^- \tag{6.44}$$

$$\mathbf{v}^+ = \mathbf{v}_0^+ + \mathbf{v}_{rel}^+ + \mathbf{\Omega}^+ \times \mathbf{r}_{rel}^+ \tag{6.45}$$

Since the maneuver is impulsive, the state of motion does not change: $\mathbf{v}_0^- = \mathbf{v}_0^+$ and $\mathbf{\Omega}^- = \mathbf{\Omega}^+$. For the same reason, the position remains unchanged, and $\mathbf{r}_{rel}^- = \mathbf{r}_{rel}^+$. It yields that:

$$\mathbf{v}^+ - \mathbf{v}^- = \mathbf{v}_{rel}^+ - \mathbf{v}_{rel}^- \tag{6.46}$$

$$\Delta \mathbf{v} = \Delta \mathbf{v}_{rel} \tag{6.47}$$

The block diagram pictured in Fig 6.6 represents the working model of this system. However, this model could also be integrated with all the previous block diagrams. In this way, the system will be able to accomplish a maneuver, to perform *on-the-go* corrections and to take into account constant and impulsive disturbances all at the same time.

6.3 Force sensor

In the previous section, the theory beyond the trajectory simulation was analyzed. Moreover, several applications of this powerful tool were discussed. Most of times, a force sensor was introduced as a mean for the robot to interface with the external world.

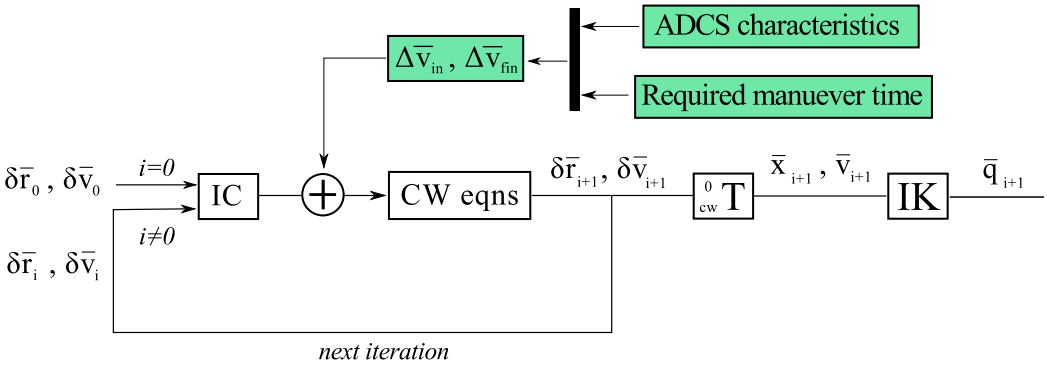


Figure 6.6: Block diagram for the free relative motion simulation.

Nowadays, there exist several kinds of dynamic sensor. The main techonlogies, with their advantages and disadvatages, are described in Table 6.1. Among the numerous choices, it is possible to narrow our search if we estabilish some requirements: first of all, most of the sensors presented in the table can measure forces along a single axis. In this application, however, we need a multi-axis sensor in order to obtain the components vectorial force acting on the object.

If we apply this *filter*, we are left with *Hall sensors* (which are sensitive to normal and shear forces), *piezoelectric sensors* (which can measure longitudinal, transversal and shear loads) and *strain gauges* (which sense forces and torques along multiple axes).

We choose a strain gauge-based sensor, due to its relative low cost and handling simplicity. A good choice, for example, would be one of the products from the *Multi-Axis Sensors* series by ATI [3].

Along with our requirements, these sensors measure all the six components of force and torque in compact and rugged transducer. This is made possible by the presence of 12 strain gauges positioned along three different axes. When a strain gauge, (properly glued to the surface of the beam), is subject to a force, its length slightly changes, changing its resistance as well. An active circuit is needed to read this change in terms of a voltage, and usually a Wheatstone's bridge is used.

It is important to remember that Wheatstone's bridge may presents dif-

Technology	Type	Advantages	Disadvantages
Mechanical	Whisker	Simple, robust	Bad resolution
	Mechanical displacement	Simple, robust, cheap	Limited resolution
	Pneumatic sensor	Simple, robust, cheap	Can't measure force
	Tactile sensor	No AD conversion	Prone to damage
Capacitive		Good sensitivity, robust	Complex circuit, noise
Strain gauges	Metal strain gauges	Robust	Temperature drift
	Semiconductor gauges	Linear, low hysteresis	Temperature drift
Piezoresistive	Conductive elastometers	Shapeable	Creep, memory
	Carbon fibers	Shapeable	Noise @ low loads
Piezoelectric		Dynamic range, durability	Fragile, complex
Pyroelectric		Dynamic range, durability	Difficult to model
Optical	Opto-mechanical	Repeteability	Creep, hysteresis
	Fiber-optic	Low noise	Complex
	Photoelasticity	Linear	Complicated optics
	Marker tracking	No interconnects	Requires PC
Magnetic	Hall effect	Shapeable	Measure in one direction
	Magnetoelastic	Wide dynamic range	Affected by noise
Ultrasonic		Dynamic range, durability	Impedance problems
Electrochemical		Low sensitivity	

Table 6.1: Advantages and disadvantages of different sensor technologies

Figure 6.7: The ATI Nano-17 6-axis transducer.

ferent configurations (full bridge, 1/2 bridge, 1/4 bridge) according to the requirements: precision, temperature drift avoidance, cost. The readings are then elaborated by the transducer default software, which provides the force and torque vector components in an orthogonal cartesian reference frame (which is clearly a body fixed frame). Hence, the components provided by the software interface do not refer to an absolute frame, and the probe position needs to be taken into account.

For the selection of the proper transducer, a maximum force requirement has to be set. Typically, we can suppose that the contact forces won't be higher than $\pm 50\,\mathrm{N}$ along each of the axes. A good option, for example, would be the ATI Nano 17 (Fig 6.7). Among its main features, it is the smallest commercially available 6-axes tranducer and presents a very fine resolution. It has silicon strain gages that provide a signal 75 times stronger than conventional foil gages. This signal is the amplified, resulting in near-zero noise distortion.

Without diving too much into the details, in Table 6.2 we present the sensor's main characteristics.

Single-axis overload		
$F_{x,y}$	±250	N
F_z	±480	N
$T_{x,y}$	±1.6	Nm
T_z	±1.8	Nm
Resonant frequency		
$F_{x,y,z}$	7200	Hz
$T_{x,y,z}$	7200	Hz
Physical specifications		
mass	0.00907	kg
diameter	17	mm
height	15	mm

Table 6.2: ATI Nano-17 main characteristics.

6.4 Matlab simulation

6.4.1 Rendezvous maneuver

By using the equation developed in the kinematics chapter, it is possible to simulate the rendevous and approach with very minor changes to the original script.

First of all, we might want to define some starting parameters: one of key parameters is the choice of the CW frame position and orientation. A wise choice would be to avoid setting the origin coincident with the manipulator's base frame origin: this is due to the fact that the manipulator has limited dexterity in this region; instead, there should be an adequate offset (we'll name this as **CW**) between the origins.

As far as concerns the orientation, the natural choice would be to use a CW frame whose axes are parallel to those of the base frame (Fig 6.8): in this way, the computation of the parameters it's faster and the motion it's easier to visualize (the horizontal ground plane corresponds to the virtual orbital plane). In our case, for example, a good choice as far as concerns the

offset would be:

$$\mathbf{CW} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \ [m] \tag{6.48}$$

The rotation matrix between the base frame and the CW frame was chosen as:

$$^{CW}_{0}\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.49}$$

That is, the CW frame is simply translated from the base frame. We then need to determine the orbit characteristics; let's imagine, for example, an approach bewteen our chaser and the International Space Station. The orbital parameters for the ISS are[2]:

$$R = 390\ km$$

$$n = \frac{V}{R} = \sqrt{\frac{\mu}{R^3}} = 1.134 \cdot 10^{-3}\ \ rad/s$$

Where n is the mean motion of the orbit and it's the only orbital parameter needed in the CW approach. Next, we need a starting point for the maneuver and a total journey time. We impose, for example:

$$^{CW}X_{in} = \begin{bmatrix} -0.4 \\ -0.5 \\ 0 \end{bmatrix} \ [m] \tag{6.50}$$

$$^{0}X_{in} = ^{CW}\ X_{in} + \mathbf{CW} = \begin{bmatrix} 0.6 \\ -0.5 \\ 0 \end{bmatrix} \ [m] \tag{6.51}$$

With a total maneuver time of t_f=300 s, we can calculate the initial and final maneuver impulses as follows:

$$\{\mathbf{0}\} = [\mathbf{\Psi}_{rr}(t_f)]\ \{\ \delta\mathbf{r}_0\} + [\mathbf{\Psi}_{rv}(t_f)]\ \{\ \delta\mathbf{v}_0^+\} \tag{6.52}$$

$$\{\delta\mathbf{v}_0^+\} = -[\mathbf{\Psi}_{rv}(t_f)]^{-1}[\mathbf{\Psi}_{rr}(t_f)]\ \{\ \delta\mathbf{r}_0\} \tag{6.53}$$

[2]We suppose the orbit to be circular; in reality, the ISS is on an elliptical orbit with h_p=388 km and h_a=401 km [14].

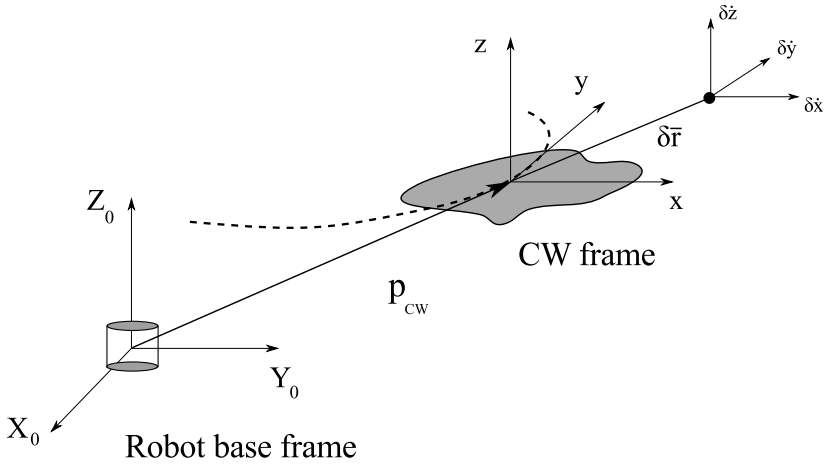


Figure 6.8: CW environment definition with respect to the robot base frame

Which yields:

$$\{\delta \mathbf{v}_0^+\} = \begin{bmatrix} 8.82 \cdot 10^{-7} \\ 2.07 \cdot 10^{-6} \\ 0 \end{bmatrix} \; [km/s] \tag{6.54}$$

From which, if we suppose a zero velocity at the beginning:

$$\Delta v_0 = \delta \mathbf{v}_0^+ \tag{6.55}$$

The final impulse can be calculated by setting to zero the final velocity:

$$\Delta \mathbf{v}_f = \delta \mathbf{v}_f^+ - \delta \mathbf{v}_f^- = \mathbf{0} - \delta \mathbf{v}_f^- = -\delta \mathbf{v}_f^- \tag{6.56}$$

Hence:

$$\left\{\delta \mathbf{v}_f^-\right\} = [\mathbf{\Psi}_{vr}(t_f)] \{ \delta \mathbf{r}_0\} + [\mathbf{\Psi}_{vv}(t_f)] \{ \delta \mathbf{v}_0^+\} \tag{6.57}$$

And finally:

$$\left\{\delta \mathbf{v}_f^-\right\} = \begin{bmatrix} -1.76 \cdot 10^{-6} \\ -1.16 \cdot 10^{-6} \\ 0 \end{bmatrix} \; [km/s] \tag{6.58}$$

The trajectory, along with the corresponding Δv, is pictured in Fig 6.9. The path has been drawn recalling that the motion in described by the following equations:

$$\{\delta \mathbf{r}(t)\} = [\mathbf{\Psi}_{rr}(t)] \{ \delta \mathbf{r}_0\} + [\mathbf{\Psi}_{rv}(t)] \{ \delta \mathbf{v}_0^+\} \tag{6.59}$$

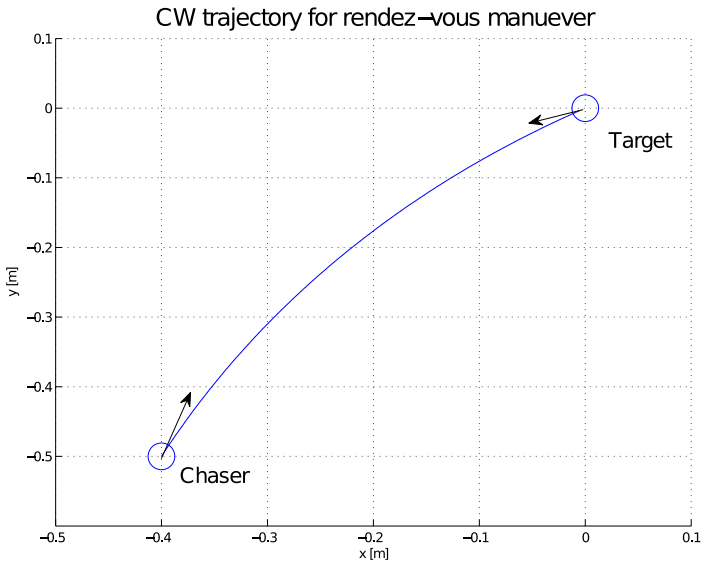


Figure 6.9: Chaser trajectory in CW coordinates for a rendezvous maneuver.

$$\{\delta\mathbf{v}(t)\} = [\mathbf{\Psi}_{vr}(t)]\,\{\,\delta\mathbf{r}_0\} + [\mathbf{\Psi}_{vv}(t)]\,\{\,\delta\mathbf{v}_0^+\} \tag{6.60}$$

Once the initial conditions $\delta\mathbf{r}_0$ and $\delta\mathbf{v}_0^+$ are known, the trajectory is fully defined. This leads to a straightforward simulation with the aid of the manipulator: it is sufficent, in fact, to feed the output of equations Eq 6.59 and 6.60 into the previously designed Matlab and Simulink control environments. The Matlab simulation screenshot is provided in Fig 6.10.

6.4.2 Rendez-vous maneuver with impulsive disturbance

The natural extension of this procedure is to implement a trajectory modification algorithm in order to account for the presence of forces. We will focus on impulsive effects only.

By using the same trajectory parameters as before, we can suppose that an object collides with the chaser after $t = T_{imp}$ from the start. This generates an impulsive force $\mathbf{F}$ that it's translated into a change in trajectory. Firstly,

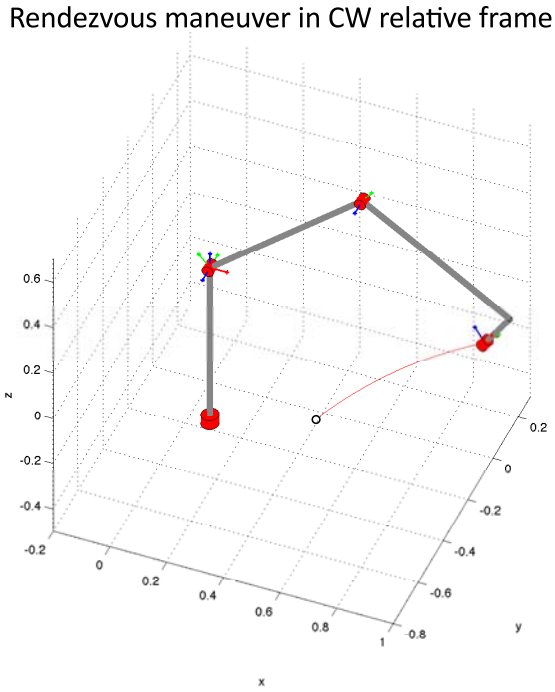


Figure 6.10: Rendezvous simulation in the CW relative frame using Matlab environment.

we define:

$$\mathbf{F} = \begin{bmatrix} -10 \\ 10 \\ 0 \end{bmatrix} \ [N] \tag{6.61}$$

in the hypothesis that this vector is already expressed in the CW frame of reference. The impact time can be chosen as:

$$T_{imp} = 0.5 \cdot t_f = 150 \ s \tag{6.62}$$

The simulation starts normally, but as soon as the sensor records the impact and integrates the data to obtain the vectorial force, there's the need for the appropriate new path simulation. The impact generates (we impose a satellite mass of 5 kg, a contact time of 0.1 s and square wave impulse profile):

$$\Delta v_{imp} = \frac{1}{m} \cdot \mathbf{F} \cdot dt = \frac{1}{5} \cdot \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \cdot 0.1 = \begin{bmatrix} -0.02 \\ 0.02 \\ 0 \end{bmatrix} \ [m/s] \tag{6.63}$$

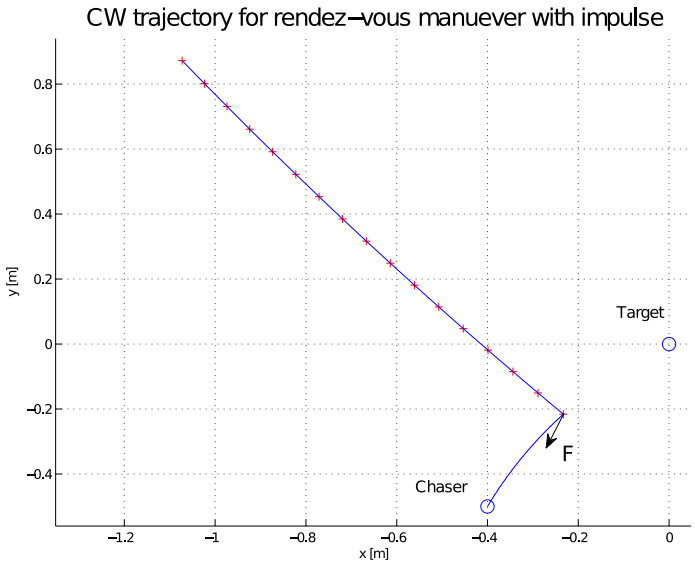


Figure 6.11: Chaser trajectory in CW coordinates for a rendezvous maneuver.

The change in the trajectory, once the Δv is known, consists in the calculation of the new $\delta\mathbf{r}_0$ and $\delta\mathbf{v}_0$. Hence, the position where the impact occours is:

$$\left\{\delta\mathbf{r}^-_{imp}\right\} = \left[\mathbf{\Psi}_{rr}(T_{imp})\right]\left\{\,\delta\mathbf{r}_0\right\} + \left[\mathbf{\Psi}_{rv}(T_{imp})\right]\left\{\,\delta\mathbf{v}_0\right\} \tag{6.64}$$

$$\left\{\delta\mathbf{r}^-_{imp}\right\} = \begin{bmatrix} -0.233 \\ -0.216 \\ 0 \end{bmatrix} \; [m] \tag{6.65}$$

And the velocity before the impact is:

$$\left\{\delta\mathbf{v}^-_{imp}\right\} = \left[\mathbf{\Psi}_{vr}(T_{imp})\right]\left\{\,\delta\mathbf{r}_0\right\} + \left[\mathbf{\Psi}_{vv}(T_{imp})\right]\left\{\,\delta\mathbf{v}_0\right\} \tag{6.66}$$

$$\left\{\delta\mathbf{v}^-_{imp}\right\} = \begin{bmatrix} 1.34 \cdot 10^{-6} \\ 1.69 \cdot 10^{-6} \\ 0 \end{bmatrix} \; [km/s] \tag{6.67}$$

The value of $\delta\mathbf{r}^+_{imp}$ is equal to $\delta\mathbf{r}^-_{imp}$ (we assume an instantaneous action of the force), whereas $\delta\mathbf{v}^+_{imp}$ is given by:

$$\delta\mathbf{v}^+_{imp} = \Delta v_{imp} + \delta\mathbf{v}^-_{imp} \tag{6.68}$$

$$\delta\mathbf{v}^{+}_{imp} = \begin{bmatrix} -2\cdot 10^{-5} \\ 2\cdot 10^{-5} \\ 0 \end{bmatrix} + \begin{bmatrix} 1.34\cdot 10^{-6} \\ 1.69\cdot 10^{-6} \\ 0 \end{bmatrix} = \begin{bmatrix} -1.86\cdot 10^{-5} \\ 2.17\cdot 10^{-5} \\ 0 \end{bmatrix} [km/s] \tag{6.69}$$

Which yields the new equations of motion:

$$\left\{\delta\mathbf{r}^{+}(t)\right\} = [\mathbf{\Psi}_{rr}(t)]\left\{\delta\mathbf{r}^{-}_{imp}\right\} + [\mathbf{\Psi}_{rv}(t)]\left\{\delta\mathbf{v}^{+}_{imp}\right\} \tag{6.70}$$

$$\left\{\delta\mathbf{v}^{+}(t)\right\} = [\mathbf{\Psi}_{vr}(t)]\left\{\delta\mathbf{r}^{-}_{imp}\right\} + [\mathbf{\Psi}_{vv}(t)]\left\{\delta\mathbf{v}^{+}_{imp}\right\} \tag{6.71}$$

The resulting trajectory is pictured in Fig 6.11. It can be seen that the impulse has completely deviated the chaser from its approach trajectory. The last frame of the Matlab animation is reported in Fig 6.12.

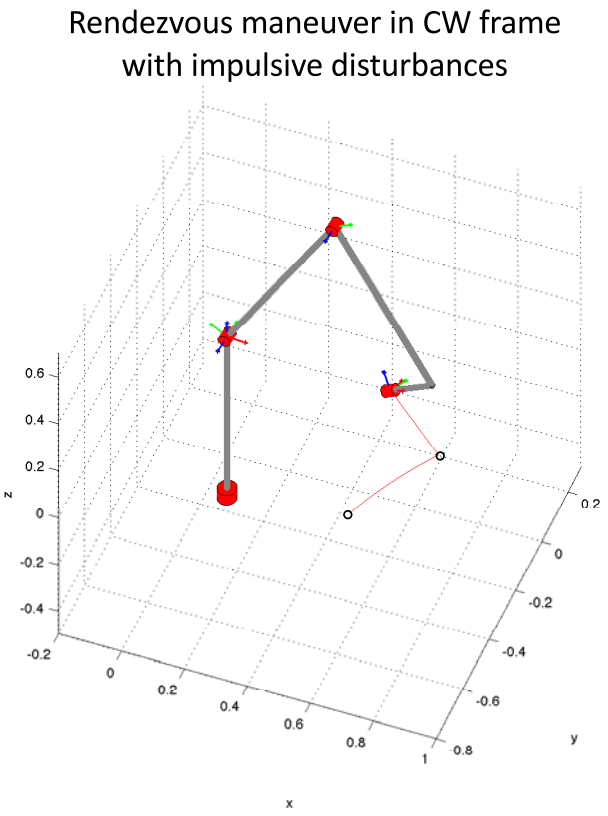


Figure 6.12: Rendezvous simulation with impulse disturbance in the CW relative frame using Matlab environment.

6.4.3 Rendezvous approach with *on the go* corrections

Finally, it is possible to simulate an actively controlled system, which allows for the continuous correction of the path in case of disturbances. What is needed, in fact, is to recompute the chasing trajectory and, in turn, the Δv that needs to be provided to the ADCS.

By contiuning with the previous example, we want the chaser to get back in the approaching path; naturally this path would be different from the one computed at the beginning of the journey, and a new set of parameters needs to be calculated.

In a real case situation, there will be a certain time delay between the drifting event and the shot of the new Δv, and we can't assume the $\delta\mathbf{r}$ vector to be constant anymore. Let's suppose a time delay between the impact and the reaction of T_{delay}=10 s; in this time span, the position and the velocity constants become (recalling Eq 6.70 and 6.71):

$$\{\delta\mathbf{r}_{shot}\} = [\mathbf{\Psi}_{rr}(T_{delay})]\,\{\,\delta\mathbf{r}^{-}_{imp}\} + [\mathbf{\Psi}_{rv}(T_{delay})]\,\{\,\delta\mathbf{v}^{+}_{imp}\} \tag{6.72}$$

$$\{\delta\mathbf{v}_{shot}\} = [\mathbf{\Psi}_{vr}(T_{delay})]\,\{\,\delta\mathbf{r}^{-}_{imp}\} + [\mathbf{\Psi}_{vv}(T_{delay})]\,\{\,\delta\mathbf{v}^{+}_{imp}\} \tag{6.73}$$

$$\left\{\delta\mathbf{r}^{-}_{shot}\right\} = \begin{bmatrix} -0.417 \\ 3.11\cdot 10^{-3} \\ 0 \end{bmatrix}\ [m] \tag{6.74}$$

$$\left\{\delta\mathbf{v}^{-}_{shot}\right\} = \begin{bmatrix} -1.82\cdot 10^{-5} \\ 2.21\cdot 10^{-5} \\ 0 \end{bmatrix}\ [km/s] \tag{6.75}$$

The new approaching trajectory is then calculated as usual (assume a total maneuver time of $t^{*}_{f} = 150$ s):

$$\{\mathbf{0}\} = [\mathbf{\Psi}_{rr}(t^{*}_{f})]\,\{\,\delta\mathbf{r}^{-}_{shot}\} + [\mathbf{\Psi}_{rv}(t^{*}_{f})]\,\{\,\delta\mathbf{v}^{+}_{shot}\} \tag{6.76}$$

$$\{\delta\mathbf{v}^{+}_{shot}\} = -[\mathbf{\Psi}_{rv}(t^{*}_{f})]^{-1}[\mathbf{\Psi}_{rr}(t^{*}_{f})]\,\{\,\delta\mathbf{r}^{-}_{shot}\} \tag{6.77}$$

$$\{\delta\mathbf{v}^{+}_{shot}\} = \begin{bmatrix} 2.84\cdot 10^{-6} \\ 4.56\cdot 10^{-7} \\ 0 \end{bmatrix}\ [km/s] \tag{6.78}$$

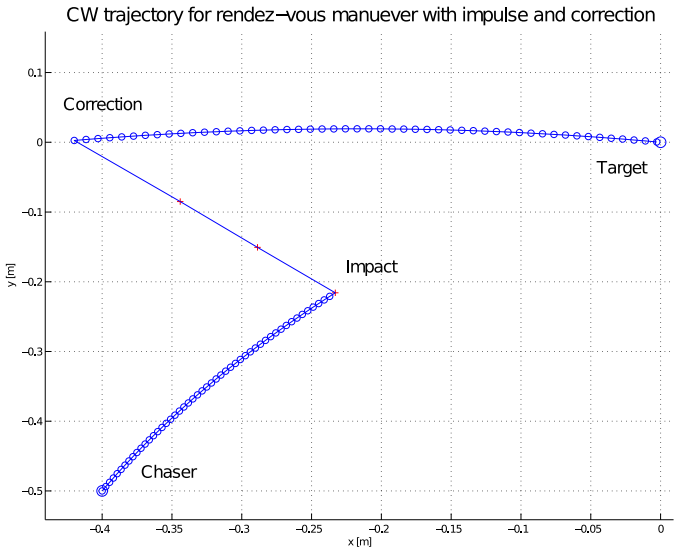


Figure 6.13: Chaser trajectory in CW coordinates for a rendezvous maneuver with *on the go* corrections.

Which yields the first impulse:

$$\Delta\mathbf{v}_0^* = \delta\mathbf{v}_{shot}^+ - \delta\mathbf{v}_{shot}^- = \begin{bmatrix} 2.11 \cdot 10^{-5} \\ -2.16 \cdot 10^{-5} \\ 0 \end{bmatrix} [km/s] \tag{6.79}$$

As far as concerns the final impulse, we set to zero the final velocity:

$$\Delta\mathbf{v}_f^* = \delta\mathbf{v}_f^{*+} - \delta\mathbf{v}_f^{*-} = \mathbf{0} - \delta\mathbf{v}_f^{*-} = -\delta\mathbf{v}_f^{*-} \tag{6.80}$$

$$\left\{\delta\mathbf{v}_f^{*-}\right\} = [\mathbf{\Psi}_{vr}(t_f^*)]\left\{\delta\mathbf{r}_{shot}^-\right\} + [\mathbf{\Psi}_{vv}(t_f^*)]\left\{\delta\mathbf{v}_{shot}^+\right\} \tag{6.81}$$

$$\Delta\mathbf{v}_f^* = \begin{bmatrix} -2.71 \cdot 10^{-6} \\ 4.91 \cdot 10^{-7} \\ 0 \end{bmatrix} [km/s] \tag{6.82}$$

The position and velocity laws for this final trajectory are then:

$$\left\{\delta\mathbf{r}^{*+}(t)\right\} = [\mathbf{\Psi}_{rr}(t)]\left\{\delta\mathbf{r}_{shot}^-\right\} + [\mathbf{\Psi}_{rv}(t)]\left\{\delta\mathbf{v}_{shot}^+\right\} \tag{6.83}$$

$$\left\{\delta\mathbf{v}^{*+}(t)\right\} = [\mathbf{\Psi}_{vr}(t)]\left\{\delta\mathbf{r}_{shot}^-\right\} + [\mathbf{\Psi}_{vv}(t)]\left\{\delta\mathbf{v}_{shot}^+\right\} \tag{6.84}$$

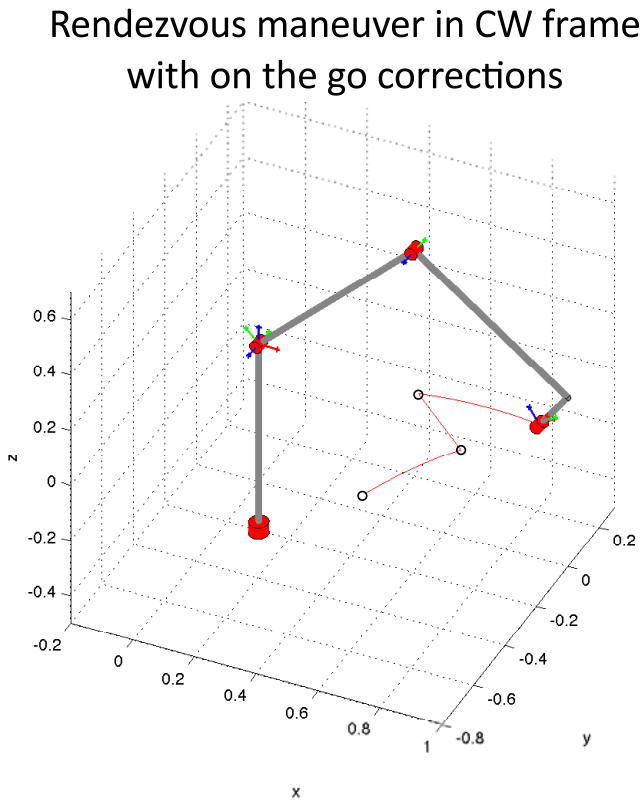


Figure 6.14: Rendezvous simulation with *on the go* corrections in the CW relative frame using Matlab environment.

The plot of the maneuver is presented in Fig 6.13. Obviously, this procedure can be performed each time a disturbance is sensed, and the corrected trajectory is easily computed. The manipulator graphic simulation yields the plot in Fig 6.14.

In this last case, we also provide the plot of the required torques for the maneuver (Fig 6.15). As expected, the behavior of the torques presents noticeable spikes in presence of the impact and of the attitude modification. This is due to the fact the trajectory has a discontinuity in these locations, and thus the accelerations needed for this repentine path change show a very steep slope. However, the maximum torques required are in the range of the motors (refer to Chapter 7 for details on the actuators), and the maneuver success will depend only on the control system capability and on the motors

response time.

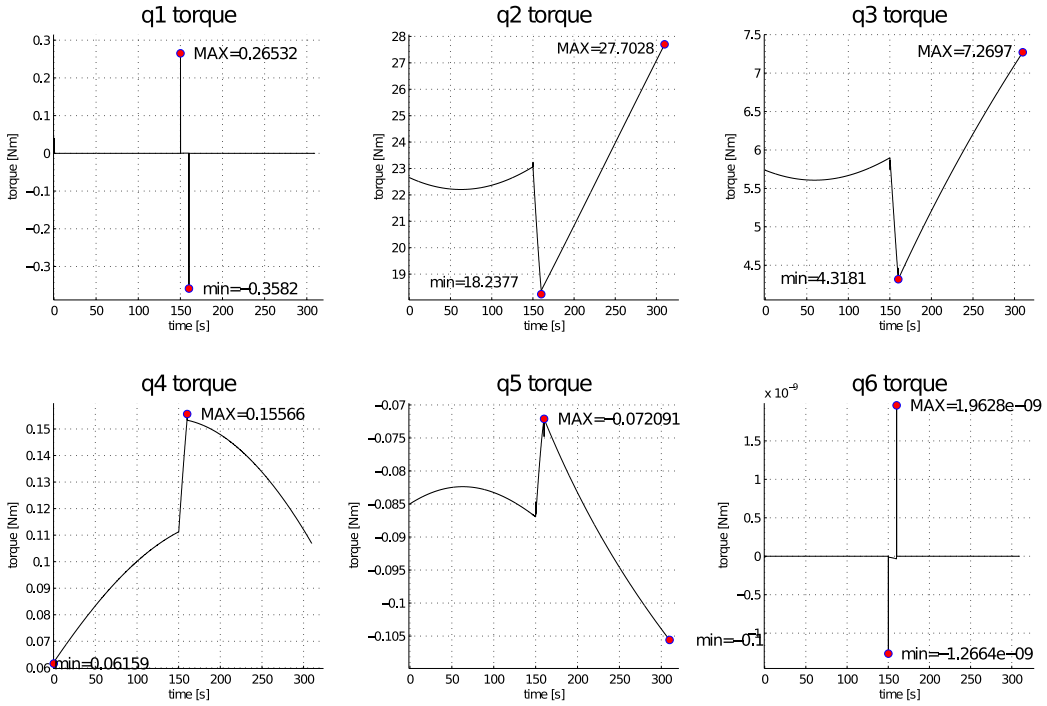


Figure 6.15: The torques required at each joint for the maneuver presented in Fig 6.13.

Chapter 7

Sizing

7.1 Introduction

In this chapter, the sizing approach is described. This process is made up of several steps, and most of the time these are not indipendent from each other: that is, the variation of one of the conditions might have important consequences on the others.

The problems presented in this chapter, in fact, cannot be solved analitically, and the solution is not unique either. Several combinations of the parameters satisfy the requirements fixed at the beginning. It is however pos-

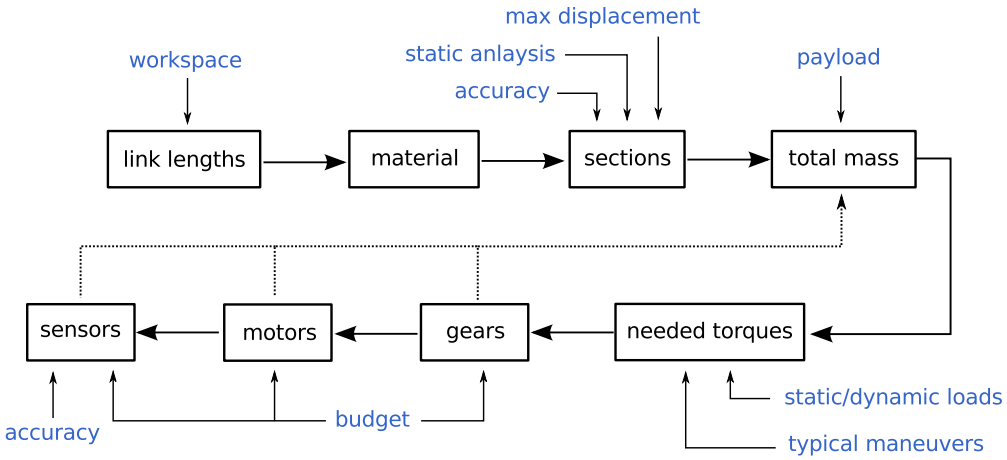


Figure 7.1: Block diagram representing the iterative sizing process.

sible to design a generic block diagram, presented in Fig 7.1, that represents the logic process on which the sizing procedure is based. In the picture, the blue writings represents the requirements, and are needed for the calculation of the different steps, whereas the dotted lines are the parameters that need to be assumed and then back-checked in a *trial-error* type of procedure.

7.2 Link design

The process starts from the requirements presented in Chapter 2. The robot needs to have a working space of at least $0.5 \times 0.5 \times 0.5$ m. Obviously, infinite configurations would satisfy this requirement, and some initial values need to be set to solve the indetermination. We can, for example, use the following lengths for the first three links:

$$l_1 = 0.7\ m \qquad l_2 = 0.7\ m \qquad l_3 = 0.6\ m \tag{7.1}$$

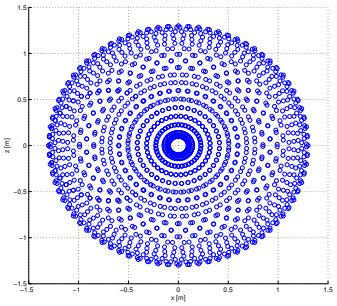
With this configuration, the working space requirement is fully satisfied. This can be inferred by plotting a cloud of points for all the positions reached by the end effector. These points are obtained from the direct kinematics by varying discretely θ_1, θ_2 and θ_3.

Since a single figure won't be easy to understand, three separate plots are presented: in the first one, θ_1=0 and θ_2, θ_3 sweep the [-π; π] interval; in the second one, θ_2=0 and θ_1, $\theta_3 \in [-\pi; \pi]$; in the last one, θ_3=0 and θ_1, $\theta_2 \in [-\pi; \pi]$.

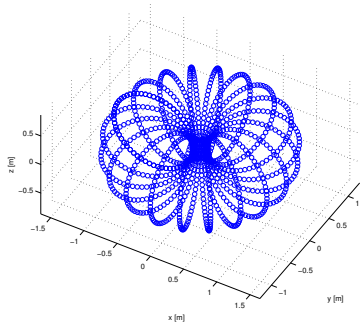
The sizing of the end effector is actually more delicate, since there shall not be any interference between the last three links. Plus, the space occupied by an eventual test object has to be considered. In Fig 7.3, a preliminary design of the end effector configuration is shown (refer to Section 1.3.3 for an accurate analysis of the problem).

Note that, in these configurations, the offsets between the links, identified as d_i in Denavit-Hartenberg's notation, are all zero except from d_4.
This choice can be explained from a computational point of view: apart from d_4, which needs to be different from zero in order to have a 3 axes intersection at the end effector[1], the other offsets are zeroed. This limits the inverse

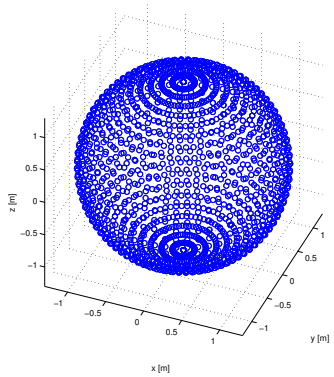
[1] In order to satisfy Pieper's approach, Sec. 2.4.1.



(a) $\theta_1 = 0$, θ_2, $\theta_3 \in (-\pi; \pi)$



(b) $\theta_2 = 0$, θ_1, $\theta_3 \in (-\pi; \pi)$



(c) $\theta_3 = 0$, θ_1, $\theta_2 \in (-\pi; \pi)$

Figure 7.2: Manipulator workspace analysis.

kinematics solution, thus speeding up the code.

As far as concerns the links, we now have their lengths. What is needed are the geometric properties and the material properties, which will then provide further information about the weight, the inertia and the stiffness.

7.2.1 Material choice

The material choice is fairly straightforward: what is needed is a material with a high resistance-to-weight (RtoW) ratio. Weight saving is a must in order to limit the size and cost of the motor. From this standpoint, steel could be a good choice in terms of performances, but its high density doesn't

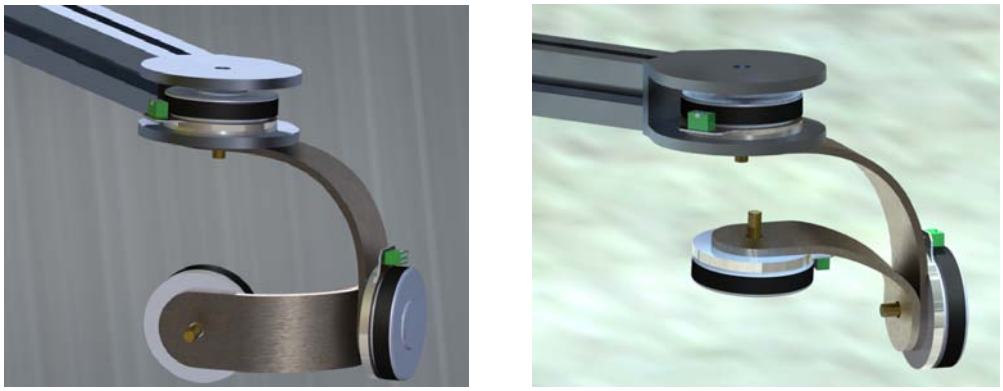


Figure 7.3: *End effector* preliminary design.

make it a suitable material for this application.

Probably the best solution, in terms of resistance-to-weight ratio, would be a composite material, such as carbon fiber or fiberglass. However, this is not the ideal choice for a preliminary design. Moreover, these materials need to be custom made, with an obvious steep increase in the price. Aluminium is definitely the most favorable solution: it presents a good RtoW ratio, it is very easy to machine and there are a lot of section choices which are relatively cheap due to the simple process (extrusion) used in their production.

7.2.2 Load analysis

When working with slender bodies, it is key to worry about their rigidity in order to avoid bending and buckling. In this case, due to the way loads are applied, it is unlikely for buckling to occur for *link 2* and *link 3*, whereas bending could be a serious issue. Moreover, a flimsy structure might cause vibrations and disturbances. *Link 1*, on the other hand, could be subjected to both buckling and bending.

To avoid these phenomena, proper sections need to be chosen. A first approach to the problem could be the analysis of the symplified 2D structure when the arm is fully extended. The diagram in Fig 7.4 schematizes the problem.

In the figure, F_M is the weight of the motor and F_{pay} is the weight of the payload. The latter accounts not only for the object attached at the tip of the

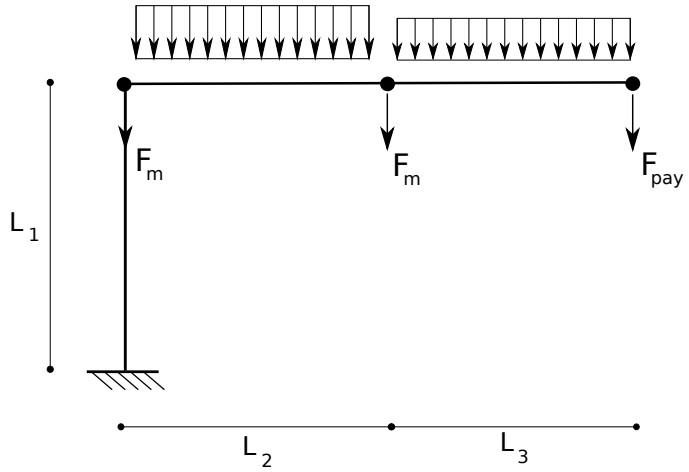


Figure 7.4: Simplified model of the robot structure.

end effector, but also for the weight of the last three links. The distributed loads represent the weight of the arms and do not have necessarily the same magnitude.

This structure can be easily analyzed analitically in order to obtain the moment, shear and normal force distribution along the links. Although we haven't defined the parametrs yet, a plot of the general behavior can be obtained and is presented in Fig 7.5.

It is clear that *link 1* is subjected to the highest load, in terms of moment (which is constant along its span) and normal force. The normal force acting

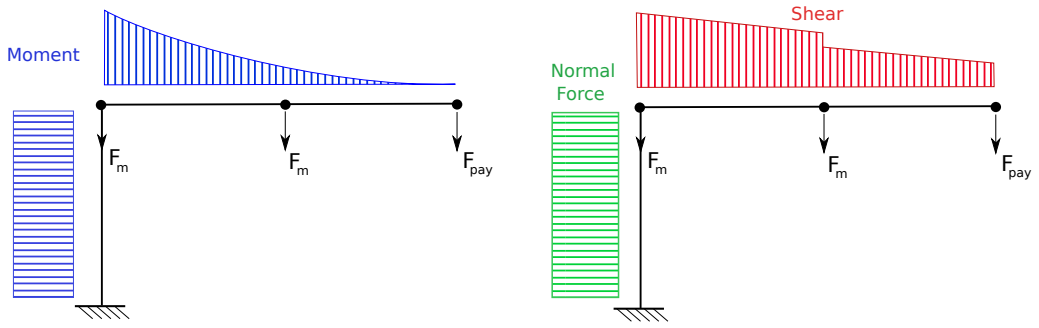


Figure 7.5: Moment, shear and normal force behavior of robot's simplified structure.

on this first link may create buckling: this situation will be further analyzed in section 7.2.3.

Moving on towards the end effector, we can see that *link 2* is subjected to a bending moment that has the highest value at *joint 1* and decreases in a parabolic fashion until the end effector. The shear force acting on *link 2* is linear.

The presence of a concentrated force at *joint 3*, induced by the weight of the motor, changes the slope of the moment profile, which keeps decreasing till zero at the end effector. On the other hand, there is a discontinuity of the shear profile due to the concentrated load F_m (note that this "jump" is equal to the value of this force), which descreases linearly from *joint 2* to the end effector, where its value is F_{pay}.

The sizing of the links, once the length is known, might start from the displacement analysis. It is possible to set a requirement on the maximum vertical displacement[2] in the worst case configuration: this happens when the manipulator is fully stretched (Fig 7.4).

With a simple analytical procedure, this approach can lead to the calculation of the minimum moment of inertia I_x required for the links' sections. The commercial link choice is then quite straightforward.

This approach, however, needs some assumptions, and a trial/error procedure has to be used:

1. The motor and payload weights are estimated, with an appropriate safety factor (e.g. 1.3/1.5)

2. The links' linear weight (kg/m) are estimated according to the common extruded profiles properties. Again, an appropriate safety factor is used

3. A maximum tip displacement requirement is set

4. The displacement analysis is executed: this will yield the product $E \cdot I$

5. Knowing the material properties (and E), the value of I is obtained

[2]We will ignore the horizontal displacement analysis. That is, we suppose the links' sections to be axialsymmetric and the loads to be applied along the shear center [18].

6. A profile having this I and the previously estimated linear weight is searched among the commercially available sections

7. If commercial profiles present higher weight for that I, the estimated weight has to be increased. Another solution could be the relaxation of the tip displacement requirement. Analysis is executed again with these modifications.

8. If commercial profiles present lower weight for that I, the estimated weight has to be descreased. Another solution could be to set a tighter requirement on the tip displacement. Analysis is executed again with these modifications.

9. If there exists commercial profiles with the parameters used, then the problem is solved and the procedure ends.

After the definition of the first-try parameters, we need to calculate the displacement. From beam theory [18], we recall the formula relating the moment and the curvature it produces on a beam:

$$\begin{cases} M_x = -EI_{xy}u'' - EI_{xx}v'' \\ M_y = -EI_{yy}u'' - EI_{xy}v'' \end{cases} \tag{7.2}$$

Since we assume to work with symmetric profiles, $I_{xy} = 0$. Moreover, in our 2D model, $M_y = 0$. We are left with the formula for the vertical displacement:

$$v''(x) = -\frac{M_x(x)}{EI_{xx}} \tag{7.3}$$

Integrating twice this formula, we can obtain the displacement of the beam as a function of x. The main problem is to obtain the $M_x(x)$ function. This can be easily accomplished remembering that we are analyzing a linear elastic problem, and the superposition of effects is value.

Therefore, the problem in Fig 7.4 can be decomposed in three parts (we assume the distributed weigth to be constant, that is, *link 2* and *link 3* have the same profile): we obtain the three cases presented in Fig 7.6. The computation of the moments derives from static equilibrium, and yields, for

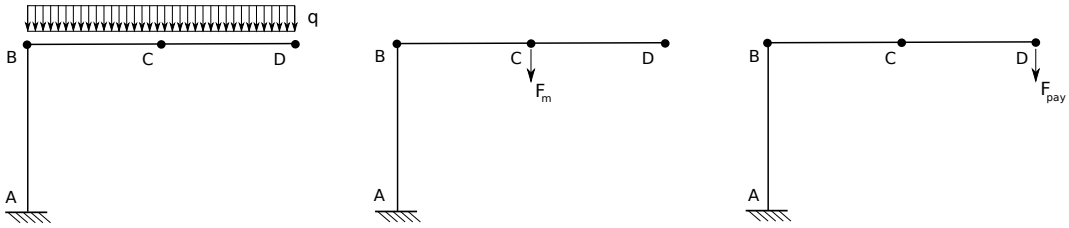


Figure 7.6: Load decomposition for structural analysis.

the three cases (referring to Fig 7.4, we set the x axis as starting from *joint 2* and going rightwise, and $L = L_2 + L_3$):

$$M_1(x) = \frac{q}{2}(L-x)^2 \qquad for \quad 0 \leq x \leq L \tag{7.4}$$

$$M_2(x) = F_m(L-x) \qquad for \quad 0 \leq x \leq L_2 \tag{7.5}$$

$$M_3(x) = F_{pay}(L-x) \qquad for \quad 0 \leq x \leq L \tag{7.6}$$

The moment diagrams are presented in Fig 7.7. From these expressions, we can obtain the corresponding displacements [18]:

Case 1

$$EIv_1'' = -M_1(x) \tag{7.7a}$$

$$EIv_1'' = -\frac{q}{2}(L-x)^2 \tag{7.7b}$$

$$EIv_1' = -\frac{q}{2}\left[L^2x + \frac{x^3}{3} - Lx^2\right] + C_1 \tag{7.7c}$$

$$EIv_1 = -\frac{q}{2}\left[L^2\frac{x^2}{2} + \frac{x^4}{12} - L\frac{x^3}{3}\right] + C_1x + C_2 \tag{7.7d}$$

From which, using the assumption of a fixed constraint at *joint 2*, $v'(x=0) = 0$ and $v'(x=0) = 0$. Thus, $C_1 = C_2 = 0$. The final expression for $v_1(x)$ is:

$$v_1(x) = -\frac{q}{24\,EI}\left[6L^2x^2 + x^4 - 4Lx^3\right] \tag{7.8}$$

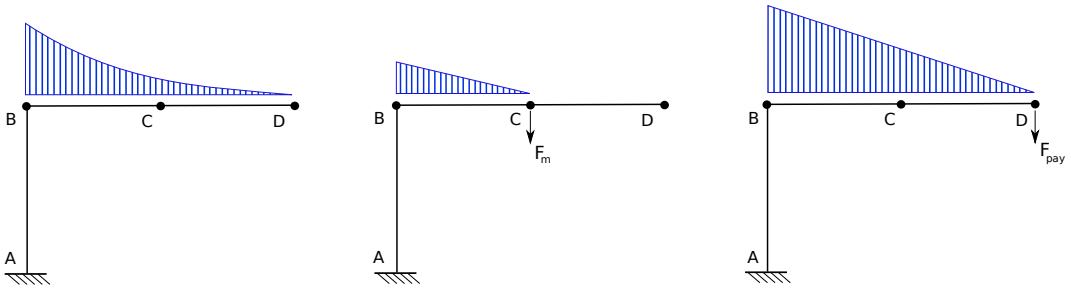


Figure 7.7: Moment diagrams for the three decomposed cases.

Case 2

$$EIv_2'' = -M_2(x) \tag{7.9a}$$
$$EIv_2'' = -F_m(L_2 - x) \tag{7.9b}$$
$$EIv_2' = -F_m\left(L_2 x - \frac{x^2}{2}\right) + C_1 \tag{7.9c}$$
$$EIv_2 = -F_m\left(L_2\frac{x^2}{2} - \frac{x^3}{6}\right) + C_1 x + C_2 \tag{7.9d}$$

From which, using the assumption of a fixed constraint at joint 2, $v'(x=0) = 0$ and $v'(x=0) = 0$. Thus, $C_1 = C_2 = 0$. The final expression for $v_2(x)$ is:

$$v_2(0 \le x \le L_2) = -\frac{F_m}{6\,EI}\left[3L_2x^2 - z^3\right] \tag{7.10}$$

Since the load is effective till $x = L_2$, the slope of the curvature past L_2 will remain constant, and the deformed curve will be a segment. Since we can calculate the value of the displacement and its derivative in L_2, the line equation[3] yields the formula for $v_2(x)$ when $L_2 \le x \le L$:

$$v_2(L_2 \le x \le L) = -\frac{F_m}{6\,EI}\left[3L_2^2x - L_1^3\right] \tag{7.11}$$

[3]We can write the line equation as $v_2(x) = v_2'(L_2)(x - L_2) + v_2(L_2)$, where $v_2(L_2) = -\frac{F_m}{3EI}L_1^3$ and $v_2'(L_2) = -\frac{F_m}{2EI}L_1^2$.

Case 3

$$EIv_3'' = -M_3(x) \tag{7.12a}$$
$$EIv_3'' = -F_{pay}(L - x) \tag{7.12b}$$
$$EIv_3' = -F_{pay}\left(Lx - \frac{x^2}{2}\right) + C_1 \tag{7.12c}$$
$$EIv_3 = -F_{pay}\left(L\frac{x^2}{2} - \frac{x^3}{6}\right) + C_1 x + C_2 \tag{7.12d}$$

From which, using the assumption of a fixed constraint at *joint 2,* $v'(x = 0) = 0$ and $v'(x = 0) = 0$. Thus, $C_1 = C_2 = 0$. The final expression for $v_2(x)$ is:

$$v_3(x) = -\frac{F_{pay}}{6\,EI}\left[3Lx^2 - z^3\right] \tag{7.13}$$

Putting together the three case, we can obtain the equation describing the total displacement as the sum of v_1, v_2, v_3:

$$v(x) = \begin{cases} \dfrac{1}{6\,EI}\left\{q\left[6L^2x^2 + x^4 - 4Lx^3\right] + \right. \\ \left. -4\left[F_m(3L_2x^2 - z^3) + F_{pay}(3Lx^2 - z^3)\right]\right. & \text{if } 0 \le x \le L_2 \\ \\ \dfrac{1}{6\,EI}\left\{q\left[6L^2x^2 + x^4 - 4Lx^3\right] + \right. \\ \left. -4\left[F_m(3L_2^2x - L_1^3) + F_{pay}(3Lx^2 - z^3)\right]\right. & \text{if } L_2 \le x \le L \end{cases} \tag{7.14}$$

Finally, the tip displacement can be obtained by evaluating Eq 7.14 for $x = L$. This yields:

$$v_{tip} = -\frac{qL^4}{8EI} - \frac{F_m L_1^2(3L - L_1) + 3F_{pay}L^3}{6EI} \tag{7.15}$$

The requirement set for the displacement is v_{tip}. Since we are looking for the value of I, Eq 7.15 can be rewritten extracting EI:

$$EI = -\frac{1}{v_{tip}}[3qL^4 + 4F_m L_1^2(3L - L_1) + 12F_{pay}L^3] \tag{7.16}$$

The parameter E, Young's modulus, is material dependent and is known: for aluminium, $E = 70$ GPa. We are left with I, which can be easily computed. The relation between the requirement v_{tip} and the minimum I needed

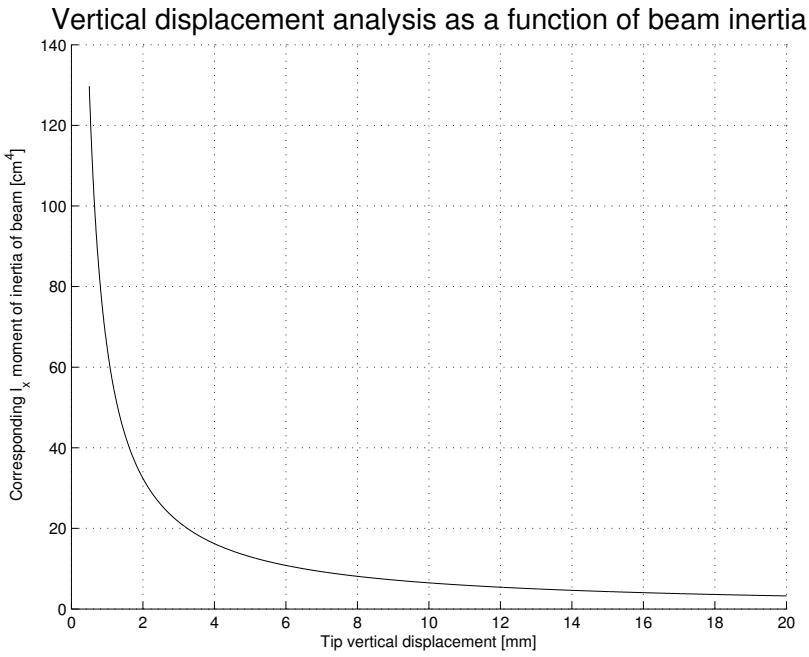


Figure 7.8: Vertical displacement versus I_x value.

can be expressed with a plot (Fig 7.8). It can be seen that the smaller the requirement for the displacement becomes, the steeper the rise in I is.

The procedure now is quite simple: with the requirements and the given values for the loads, I is computed. From commercial available profiles, the sections with a similar I are located, and the mass for linear meter is compared to the one supposed at the beginning (q).

If these values are close to each other, say, in a $\pm 10\%$ range, the assumptions were good and the sections are readily available. If, on the other hand, this is not happening, we need to re-iterate the process. Two parameters can be changed: the requirement on the displacement and/or the weight for unit length q. If, for example, the computed I is typical of profiles with higher mass per unit length, the simulation will be repeated increasing the presumed q. This tuning will finally provide a compatible solution. In our

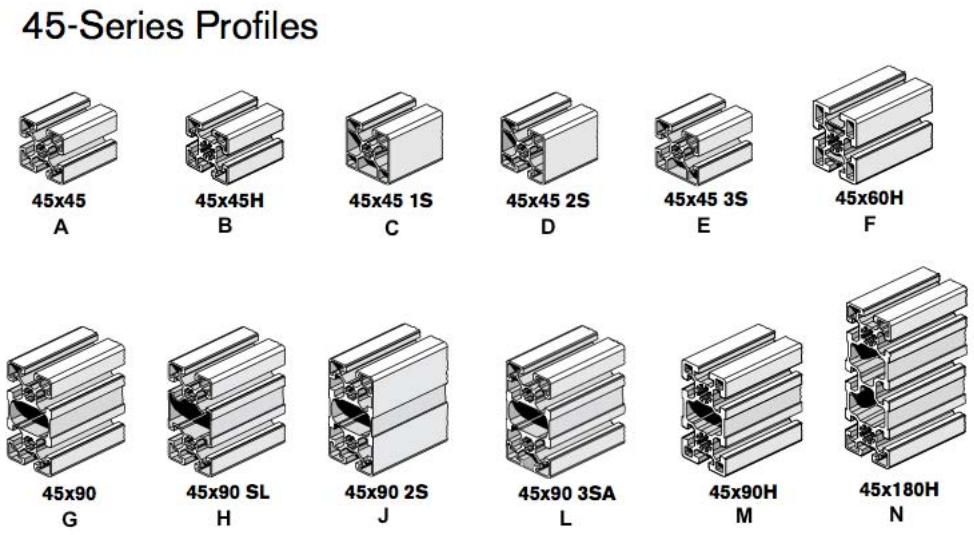


Figure 7.9: Typical extruded aluminium profiles. Data gathered from *boschrexroth.us*.

case, the sizing parameters were chosen as follows:

$$\begin{aligned} F_m &= 1\ kg \\ F_{pay} &= 2\ kg \\ E &= 70\ GPa \\ q &= 2\ \ kg/m \\ v_{tip} &= 5\ mm \end{aligned}$$

With these data, Eq 7.16 yields the value for the x moment of inertia:

$$I_x = 11.27\ \ cm^4$$

From the internet [2], we can find some typical extruded profiles (Fig 7.9). The main properties corresponing to these sections are gathered in Table 7.1.

From the table, it can be seen that, for profiles with a linear weight similar to the one we chose ($2\ kg/m$), the x moment of inertia satisfies the computed value. In our case, sections **B, C, D, E** might fulfill our needs. An interesting

	A	B	C	D	E	F	G	H	J	L	M	N
I_x $[cm^4]$	11	13.8	11.8	11.5	11.8	37.2	81.8	73.4	85.6	87.2	124.6	766.7
I_y $[cm^4]$	11	13.8	12.1	11.4	11.7	22.7	23.1	18.1	38.1	38.8	32.8	57.3
mass [kg/m]	1.5	2.0	1.6	1.6	1.6	2.9	3.0	2.44	3.24	3.2	4.15	6.90

Table 7.1: Geometric characteristic of the extruded profiles pictured in Fig 7.9.

choice would be also section H, which is relatively lightweight and, due to its bigger dimensions, could facilitate the building and interfacing with the motors (recall that the motor height, in fact, has to be augmented with the length of the transmission gear).

For **H**, however, we need to the change the parameters of the simulations: weight requirement will increase from 2 *kg/m* to ∼2.5 *kg/m*. Since we have the moment of inertia of the section, we solve equation Eq 7.15 for the maximum displacement. This yields:

$$v_{tip}^{H} = 0.83\, mm \tag{7.17}$$

Which shows an interesting improvement in the stiffness of the structure. The main drawback is given by the extra torque that the motors have to provide for equilibrium. With these data, *link 2* and *link 3* are fully defined.

The next step is to define the characteristics of *link 1*. This link is subjected to both bending and compression. We can treat this case with a buckling analysis.

7.2.3 Buckling analysis

Link 1 can be analyzed as a column subjected to a loading combination (Fig 7.10). There are, in fact, a normal loading component P_N, given by:

$$P_N = q(L_2 + L_3) + 2\, F_m + F_{pay} \tag{7.18}$$

And a moment, which can be calculated (in the worst case condition), as:

$$M = F_m L_2 + F_{pay}(L_2 + L_3) + q\frac{{L_2}^2}{2} + qL_3(L_2 + \frac{L_3}{2}) \tag{7.19}$$

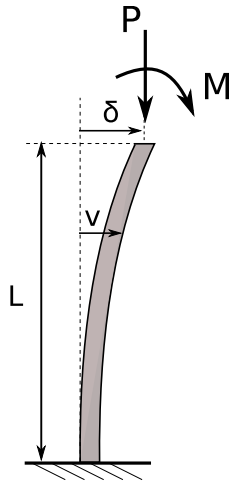


Figure 7.10: Free body diagram for *link 1*.

Under this loading, it is possible to calculate the deformed profile of the column. We recall the governing equation from bending theory [18]:

$$EI\frac{d^2v}{dx^2} = -\mathcal{M} \tag{7.20}$$

From which, having this case $\mathcal{M} = P_N v + M$:

$$EI\frac{d^2v}{dx^2} = -[P_N v + M] \tag{7.21}$$

$$\frac{d^2v}{dx^2} + \frac{P_N}{EI}v = -\frac{M}{EI} \qquad M' = -\frac{M}{EI} \tag{7.22}$$

$$\frac{d^2v}{dx^2} + \frac{P_N}{EI}v = M' \tag{7.23}$$

Where the last one is a second order linear non homogeneous differential equation; its solution v is composed by a combination of the homogenous solution v_h and of a particular one v_{part}:

$$v = v_h + v_{part} \tag{7.24}$$

The homogeneous solution can be obtain recalling the armonic oscillator equation:

$$v''(x) + k^2 v(x) = 0 \tag{7.25}$$

Where:

$$k^2 = \frac{P_N}{EI} \tag{7.26}$$

From Eq 7.25, the homogenoeous solution is [5]:

$$v(x) = A\, sin(kx) + B\, cos(kx) \tag{7.27}$$

The particular solution is obtained immediately if the characteristic equation[4] is considered, and yields:

$$v_{part} = \frac{M'}{k^2} \tag{7.28}$$

The complete solution is then:

$$v(x) = A\, sin(kx) + B\, cos(kx) + \frac{M'}{k^2} \tag{7.29}$$

The coefficients A and B can be obtained from the boundary conditions of the problem. In this case, since the lower end is considered fixed and the upper free, we have that $v = 0$ for $x = 0$ and $\frac{dv}{dx} = 0$ for $x = 0$. The coefficients then become:

$$A = 0 \qquad B = -\frac{M'}{k^2} \tag{7.30}$$

Equation 7.29 is useful to obtain the deformation of the column under the combined load. However, to structurally dimension the link, we can use the secant formula. Suppose the two ends of the column are pinned. Then the boundary conditions become $v = 0$ for $x = 0$ and $v = 0$ for $x = L$. This yields the following conditions:

$$0 = B + \frac{M'}{k^2} \quad \rightsquigarrow \quad B = -\frac{M'}{k^2} \tag{7.31}$$

$$0 = A\, sin(kL) - \frac{M'}{k^2} cos(kL) + \frac{M'}{k^2} \tag{7.32}$$

$$A\, sin(kL) = \frac{M'}{k^2}[cos(kL) - 1] \tag{7.33}$$

From which, recalling the duplication trigonometric formula:

$$sin(kL) = 2sin\left(\frac{kL}{2}\right) cos\left(\frac{kL}{2}\right) \tag{7.34}$$

[4]The characteristic equation, in this case, is given by $s^2 + k^2 = 0$.

$$1 - cos(kL) = 2sin^2\left(\frac{kL}{2}\right) \tag{7.35}$$

Plugging these relations into Eq 7.33:

$$A\left[2sin\left(\frac{kL}{2}\right)cos\left(\frac{kL}{2}\right)\right] = -\frac{M'}{k^2}\left[2sin^2\left(\frac{kL}{2}\right)\right] \tag{7.36}$$

$$A = -\frac{M'}{k^2}\,tan\left(\frac{kL}{2}\right) \tag{7.37}$$

Finally, Eq 7.29 becomes:

$$v(x) = -\frac{M'}{k^2}\left[tan\left(\frac{kL}{2}\right)sin(kx) + cos(kx) - 1\right] \tag{7.38}$$

The maximum displacemente is obtained for $x = L$

$$v_{max} = -\frac{M'}{k^2}\left[tan\left(\frac{kL}{2}\right)sin(kL) + cos(kL) - 1\right] \tag{7.39}$$

Using the secant, we can write:

$$v_{max} = -\frac{M'}{k^2}\left[sec\left(\frac{kL}{2}\right) - 1\right] \tag{7.40}$$

We note that v_{max} becomes infinite when the argument of the secant tends to $\frac{\pi}{2}$:

$$\frac{kL}{2} = \sqrt{\frac{P_N}{EI}} \cdot \frac{L}{2} = \frac{\pi}{2} \tag{7.41}$$

This expression yields the value of P for which v_{max} becomes unacceptably large, that is, the *critical load*. Solving for P, we get:

$$P_{cr} = \frac{\pi^2 EI}{L_e^2} \tag{7.42}$$

Where L_e is the effective legth and depends on the constraint configuration (refer to Table 7.2). Thus, the *critical load* in this case becomes:

$$P_{cr} = \frac{\pi^2 EI}{4L^2} \tag{7.43}$$

With this formula, we can solve the $y-z$ problem, where the x axis belongs to the manipulator arm plane, the y is perpendicular to it and z is directed along *link 1*.

Fixtures	L_e
Pinned-Pinned	L
Fixed-Free	2 L
Fixed-Fixed	0.5 L
Pinned-Fixed	0.7 L

Table 7.2: Effective lengths for different constraint configurations

Equation 7.43, however, is not sufficient for the calculation of the $x-z$ behavior, since a bending moment is acting as well. The maximum stress σ_{max} is compressive and composed by two components:

$$\sigma_{max} = \frac{P_N}{A} + \frac{\hat{M}}{I}c \tag{7.44}$$

Where $\hat{M} = M + P_N\, v_{max}$. Rearranging all the previous expressions, we end up with:

$$\sigma_{max} = \frac{P_N}{A} + \left[\frac{M}{I} + \frac{P_N\, v_{max}}{I}\right] \cdot c \tag{7.45a}$$

$$\sigma_{max} = \frac{P_N}{A} + \left\{\frac{M}{I} + \frac{P_N}{I} \cdot \frac{M}{P_N}\left[sec\left(\sqrt{\frac{P_N}{EI}} \cdot \frac{L}{2}\right) - 1\right]\right\} \cdot c \tag{7.45b}$$

$$\sigma_{max} = \frac{P_N}{A} + \left[\frac{M}{I} \cdot sec\left(\sqrt{\frac{P_N}{EI}} \cdot \frac{L}{2}\right)\right] \cdot c \tag{7.45c}$$

Where c is the distance measured from the centroid of the profile section to the outer skin. We can finally proceed with the calculations for the thesis' case. We start by defining and calculating the needed parameters:

$$q = 2.44\ \ kg/m$$
$$P_N = 52.2\ \ N$$
$$M = 47.19\ \ Nm$$

From Eq 7.43, the minimum value for I required to withstand $y-z$ buckling is:

$$I = \frac{52.2 \cdot 4 \cdot 0.7^2}{\pi^2 \cdot 70 \cdot 10^9} = 1.1 \cdot 10^{-10}\, m^4 = 0.011\, cm^4 \tag{7.46}$$

Obviously, this requirement is easily met even with the thinner sections available. It might be more interesting to set some requirements on the maximum displacement v_{max}. A starting value could be $v_{max} \leq 1\,mm$. Solving Eq 7.40 for I yields:

$$I = \frac{P_N}{E \cdot \left\{ \frac{2}{L} \cdot acos \left[\frac{1}{1+\frac{P_N}{M} v_{max}} \right] \right\}^2} = 4.13\,cm^4 \tag{7.47}$$

From Table 7.1 we can see that even the smaller profile, the **A** model, is suitable for this application, whose specifics are:

$$I_x = I_y = 11\ cm^4$$
$$h = w = 45\ mm$$
$$q = 1.5\ kg/m$$
$$A = 5.70\ cm^2$$

For sake of precision, it is also possible to verify under which condition this profile will fail. The critical load for profile **A**, for $x - y$ buckling, is:

$$P_{cr,y} = \frac{\pi^2 EI}{4L^2} = \frac{\pi^2 \cdot 70 \cdot 10^9 \cdot 11 \cdot 10^{-8}}{4 \cdot 0.7^2} = 38.7\,kN \tag{7.48}$$

As far as concerns the $y - z$ behavior, we recall Eq 7.45:

$$\sigma_{max} = \frac{P_N}{A} + \left[\frac{M}{I} \cdot sec \left(\sqrt{\frac{P_N}{EI}} \cdot \frac{L}{2} \right) \right] \cdot c \tag{7.49}$$

In this case, we set $\sigma_{max} = \sigma_Y$: we want to find the load $P_{cr,x}$ that causes yielding. Since $\sigma_Y = 165\,MPa$ for aluminum, we write:

$$\sigma_Y = \frac{P_{cr,x}}{0.00057} + \left[\frac{47.19}{11 \cdot 10^{-8}} \cdot sec \left(\sqrt{\frac{P_{cr,x}}{70 \cdot 11 \cdot 10}} \cdot \frac{0.7}{2} \right) \right] \cdot \frac{0.045}{2} \tag{7.50}$$

Solving numerically for $P_{cr,x}$ (with Newton-Rapson technique, for example), we get:

$$P_{cr,x} = 25.7\ kN \tag{7.51}$$

This means that, since $P_{cr,x} < P_{cr,y}$, the link will fail due to the moment and compression combination on plane $y - z$. This is intuitive since the loading conditiong is higher in this case.

Note that the maximum stress when failing occurs is:

$$\sigma_{max} = \frac{P_{cr,x}}{A} = 45.12 \ MPa \tag{7.52}$$

$$\sigma_{max} = 45.12 \ MPa \ll \sigma_Y = 165 \ MPa \tag{7.53}$$

That is, failing occurs much before yielding: this is due to the slenderness of the column. By decreasing the slenderness ratio (i.e. the ratio between the height and the width of the element), it can be shown that the failure mode shifts from buckling to yielding.

Reassuming the reults obtained in this section, the links' sizing parameters are:

	length $[m]$	area $[cm^2]$	$I_x[cm^4]$	$I_y[cm^4]$	q $[kg/m]$	mass $[kg]$
link 1	0.7	5.70	11	11	1.5	1.05
link 2	0.7	9.04	73.4	18.1	2.44	1.71
link 3	0.6	9.04	73.4	18.1	2.44	1.46

It is important to note that the sizing of the link is actually oversized for static stability: however, designing a robot's structure just to avoid yielding is not the way to proceed here. In fact, apart from yielding, there's the need to avoid vibrations and resonance: a flismy structure, although statically adequate, might be excited at very low frequencies. Oversizing the links beyond the static safety limits seems to be the best way to shift the natural frequency of the system to higher, less dangerous ranges.

Moreover, since the inertial characteristics of *link 1* do not influence the performances of the structure, we can think of using section **H** for *link 1* too. The extra weight, in fact, will not affect any of the motors: on the other hand, we obtain a much stiffer structure, way less prone to resonance and vibrations than the one having section **A** for *link 1*.

At this point, the first three links are fully dimensioned. As far as concerns the last three links, the approach used is different. Since their lengths are fairly small, they won't be subjected to relevant moments. The main problem, thus, would be the geometric definition (yielding of the material and bending/buckling can be easily ignored).

Since, referring to Fig 7.3, the profiles need to be curved, we can't no longer use a standard extruded profile; the section will have to be custom made. The profiles have been designed in SolidWorks, using a thickness of $5mm$. The last link, which is supposed to host the force sensor, has been approximated with an axialsymmetric cylinder.

By knowing the specific weight of the material, the mass was computed. In the following table these parameters are presented. More details on the geometry are presented in the *Physical parameters* section.

	mass $[g]$
link 4	230
link 5	190
link 6	210

7.3 Motor choice

7.3.1 Motor types

There are several types of motors available for robotic purposes, which can be divided according to different characteristics.

The main distinction is based on the kind of power used, AC or DC. AC motors are generally used in high power, single or multi phase industrial systems where a constant rotational rate is usually needed. The motors used in robotics are an evolution of these actuators, with powers ranging from 10W to 10kW, and are mainly DC powered.

The key parameters to seek for when choosing a motor are:

- ◇ high *power-to-weight* ratio
- ◇ low mass and inertia
- ◇ high rotational speed
- ◇ low backlash (e.g. high precision)

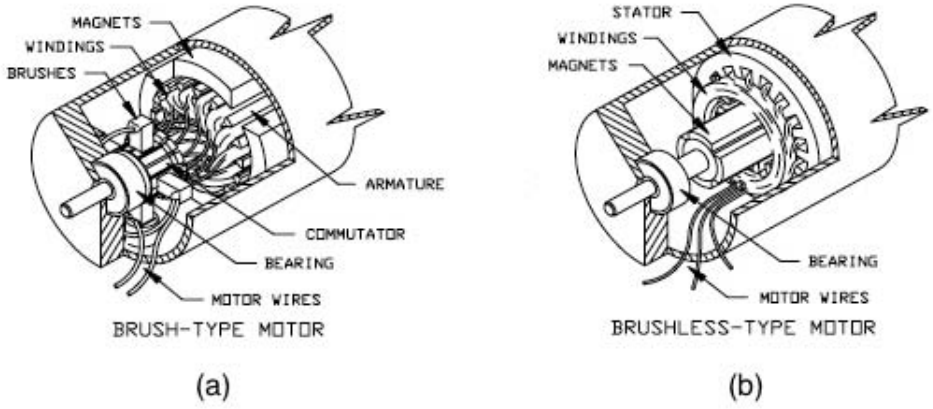


Figure 7.11: Brushed and brushless DC motor schematics.

◇ low torque ripple

◇ *(if available)* accurate built-in sensors

The most common DC electric actuators can be further divided into two classes: permanent-magnet DC servomotors and brushless DC servomotors.

The first type, Fig 7.11 (a), consists in a stator coil that creates the magnetic flux, since it is a permanent magnet made out of a ferromagnetic material; the rotor, which is again made out of a ferromagnetic material, is covered by an armature, composed by current carrying windings. Brushes between the rotating armature and the external winding are used for the commutation.

In the brushless type, Fig 7.11 (b), the rotor (in ferromagnetic material) generates the magnetic flux, whereas the fixed external armature (stator) has polyphase windings. The commutation is provided by a position sensor placed on the shaft, which generates the feed sequence for the windings.

It is crystal clear that in the latter case, because of the absence of physical contact between the rotor and the stator, the performances are definitely superior. First of all, with no contact, the mechanical losses due to friction are minimized. The elimination of brushes eliminates also the electric loss due to voltage drops at the contact of brushes and plates. Moreover, with

no contact there is also less material wear, and the motor life is increased.

This choice, however, comes with its **disadvantages**: brushless motors are more expensive, and they usually require a more sophisticated control algorithm.

This first mental process allows for the identification of the kind of motor to be used in this application. Due to the superior performances and the limited disadvantages, we'll stick with a brushless DC motor. What is needed next are the *operative requirements* for each of the actuators.

7.3.2 Requirements

The motor choice starts with the analysis of torques required in the **worst operational case**. Two contributions sum up:

1. the torques needed for supporting the weight of the structure[5]
2. the torques needed for withstanding the dynamic effects

For most applications, the first contribution is much more relevant than the second one. The ideal approach is to study the static case and then add a correction factor chosen accordingly to the typical torques calculated in the dynamics simulations.

We start with the analysis of the **static** situation. The motors loads depend on the configuration, that is, on the values of the generalized coordinates $\bar{\mathbf{q}} = [q_1 \ \dots q_n]$. However, it is immediate to note that θ_1 does not have any relevance in changing the configuration's loads. For the same reason, θ_6 has no influence as well.

We are left with θ_2, θ_3, θ_4 and θ_5, whose variations change the torques needed for static equilibrium. The situation can be further simplified by noting that the end effector can be considered as a single body, thus arriving at the conclusion that only θ_2 and θ_3 are playing a significant role in the variation of the static torque. (With this simplification, the static torque analysis for *joint 1 and 6* is momentarily ignored).

With the values obtained from the analysis presented in Section 7.2, we can obtain the moments that need to be provided at *joints 2* and *joint 3* in

[5] We suppose a gravity field different from zero.

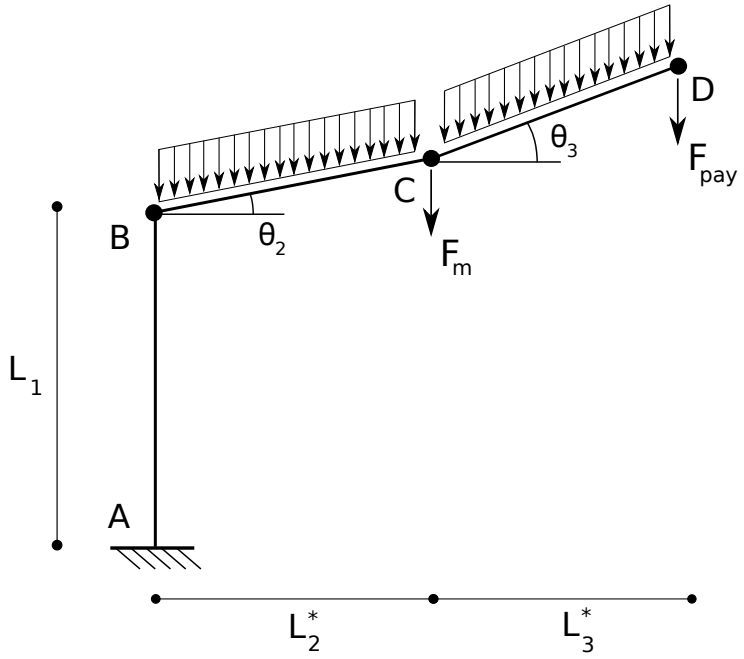


Figure 7.12: Simplified model of robot's structure for nonzero joint angles

order to have a static equilibrium. Referring to Fig 7.12, we can write:

$$M_2(\theta_2, \theta_3) = q \cdot L_2^* \cdot \frac{L_2^*}{2} + F_m L_2^* + F_{pay}(L_2^* + L_3^*) + q_3^* \cdot \left(L_2^* + \frac{L_3^*}{2} \right)$$

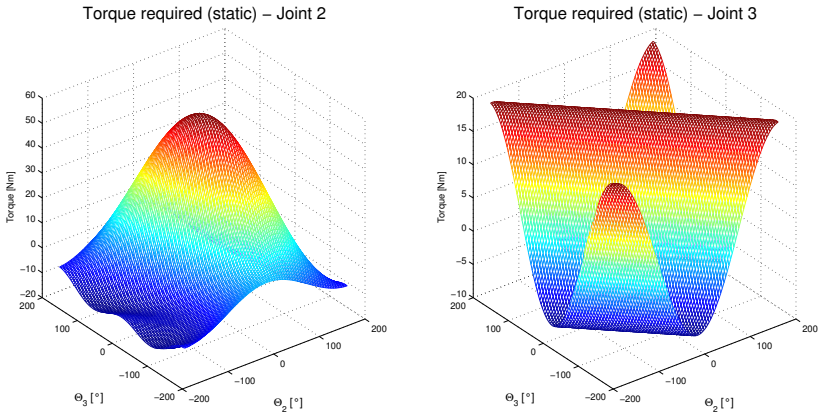
$$M_3(\theta_2, \theta_3) = q \cdot L_3^* \cdot \frac{L_3^*}{2} + F_{pay} L_3^*$$

Where:

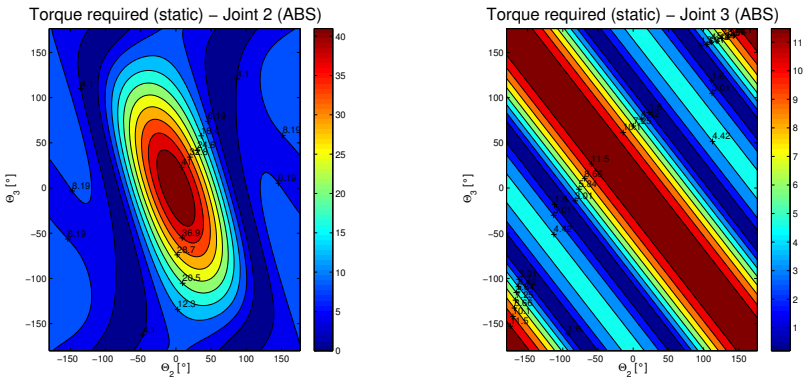
$$L_2^* = L_2 \cdot cos(\theta_2)$$
$$L_3^* = L_3 \cdot cos(\theta_2 + \theta_3)$$

This calculation can be extended from the simple discrete case to a continuous analysis of all the possible configurations for *link 2* and *3*: by varying θ_2 and θ_3, two 3D plots are obtained, one representing on the z-axis the torque τ_2 as a function of θ_2 and θ_3, and the other representing τ_3 as a function of θ_2 and θ_3. The results are presented in Fig 7.13.

The peaks (both positive and negative) represent the worst loading conditions, and the choice of the motor starts here.



(a) Surface plot: τ_2 on the left, τ_3 on the right



(b) Contour plot: τ_2 on the left, τ_3 on the right

Figure 7.13: Static torque analysis for *joint 2* and *3*

When the arm is fully stretched, there is a maximum of the torque to be provided at *joint 2*. For the addition of the dynamic contribution, we can refer to the plots and the tables in Section 4.4. The torques required in the dynamic case are in the $+10 \div 12\%$ range, confirming the assumption that dynamic loading constitutes a relatively small percentage of the total value.

A good design strategy would be to multiply the maximum values obtained from Fig 7.13 by a dynamic correction coefficient of 1.2. Moreover, in addition to the dynamic effects, we need to take into account also friction and all the additional hardware and harness weight needed in the installation.

An additional safety factor (SF) of 1.1 can be reasonably introduced. Besides, this allows to still have some authority margin when the critical conditions are reached.

As far as *joint 1* is concerned, it can be seen that, ideally, there are no torque requirements for static equilibrium: since the axis of rotation is parallel to the gravity vector, the vectorial moment acting on *joint 1* due to gravity can not be handled by *joint 1*. Consequently, an appropriate bearing attached to *joint 1* will be needed in order to hadle this load.

In this case, the only torque contribution for the motor sizing is given by inertial and dynamic loads, which can be inferred again from the torque tables of Section 4.4. A sizing torque of 10 Nm seems to be a reasonable choice. We again multiply this value by the usual safety factor SF.

The last three joints are clearly not subjected to high torque values. *Joint 4* experiences the maximum static torque in the (unlikely) case in which *link 3* is parallel to the $\mathbf{z_0}$ direction and *link 4* is perpendicular to *link 3*. Maximum torque for *joint 5* happens when *link 5* and its axis of revolution lie on the ground plane.

Recalling the link masses from Section 7.2 and assuming 300 g as the average weight of the motors, the maximum moments are calculated with the following formulas:

$$M_4 = \frac{L_4}{2} \cdot (m_4 \cdot g) + L_4 \cdot [(m_5 + m_6 + m_{pay} + 2 \cdot m_M) \cdot g] \tag{7.55}$$

$$M_5 = \left[\frac{L_5}{2} \cdot m_5 + L_5 \cdot m_M + \left(L_5 + \frac{L_6}{2} \right) \cdot (m_6 + m_{pay}) \right] \cdot g \tag{7.56}$$

Joint 6 maximum torque takes into account only dynamic related torques: as long as *link 6* and the objects connected to it are axial-symmetric, there are no acting torques for static equilibrium.

From Section 7.2 we can see that for the fastest trajectory simulated, a torque value of $7.23 \cdot 10^{-6}$ Nm is needed. However, we are in the approximation of an axialsymmetric body connected to the shaft whose axis of giration is coincident with the shaft's axis. Since other bodies, non necessarily axial-symmetric, might be attached to it for testing, and due to possible misalignments between the axis, the torque required could be bigger. In order to stay

away from saturation, we can think of increasing the requirements: a commercial motor in the $0.1 \div 0.2$ Nm range appears to be more than sufficient to withstand misaligments and (limited) extra weight.

The following table summarizes the maximum torque required by each motor (all values are in Nm):

	Joint 1	*Joint 2*	*Joint 3*	*Joint 4*	*Joint 5*	*Joint 6*
Static torque	/	40.3	12.9	1.27	1.34	/
Dynamic correction	10	48.4	15.5	1.53	1.61	0.2
SF correction	12	53.2	17.1	1.68	1.77	0.25

With the aid of these simple static formulas, this procedure allows for the selection of the motors in terms of torques. There are no problems in finding a motor that satisfies the torque requirement for *joint 6*. Problems arise when looking at the other joints. The maximum output torque for an average commercial motor is usually on the order of $\sim$1 Nm: a transmission gear is obviously needed.

This kind of device allows to up-size/down-size a motor in terms of torque; the consequence is a proportional change in the revolution speed, according to the following laws:

$$\dot{\theta}_{out} = \frac{1}{\eta} \cdot \dot{\theta}_{in} \tag{7.57}$$

$$\tau_{out} = \eta \cdot \tau_{in} \tag{7.58}$$

From which:

$$\dot{\theta}_{in} \cdot \tau_{out} = \dot{\theta}_{out} \cdot \tau_{in} \tag{7.59}$$

Where η is the gear ratio. However, this is an ideal law: in the real case mechanical losses are present. Clearly, a good performing transmission, in terms of efficiency, is what we are looking for. The reduction ratio should be as high as possible, as long as the minimum rotational speed requirement is satisfied. Moreover, a high reduction rate means usually a longer series of reduction stages, which decrease the efficiency and increase the backlash. This, however, is not true for certain kind of gears, such as the **harmonic drives** (also know as "strain wave gearing").

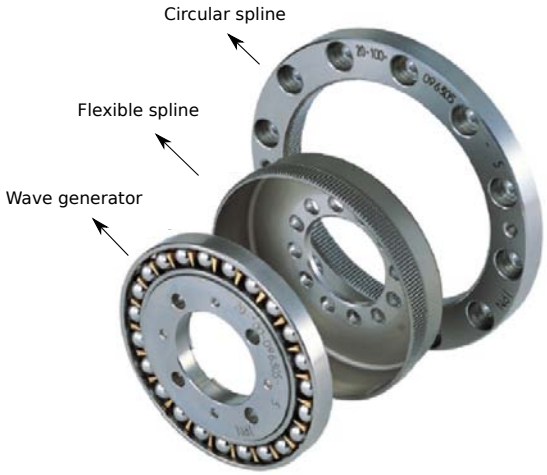


Figure 7.14: Harmonic gear schematic [34]

These devices are built by taking advantage of the metal elasticity. They are composed by three main elements (refer to Fig 7.14): a wave generator, a flex spline and a circular spline. The flex spline sides are very thin, but the bottom is thick and rigid and can be attached to a shaft (output shaft). The wave generator is tightly secured inside the flex spline, and when it is rotating, the spline deforms to the shape of an ellipse.

Because of this shape, the teeth of the spline mesh only with two parts of the outer circular ring: the key point of the design is that there are two/three fewer teeth in the flex spline. This means that for an entire spin of the input shaft (connected to the wave generator), the output shaft (secured to the flex spline) rotates only of the angular range corresponding to those two/three teeth. That is, extremely high reduction ratio are possible (up to *300:1*), coupled with good resolution, high torque capability, compactness, light weight and no backlash[6].

Practically, with this kind of gear, very small (and cheap) motors would be necessary to produce high torques, thus making this the best performing transmission device any robot could desire. In this thesis, harmonic gear

[6]Backlash, which arises from the imperfect meshing of gears, can be defined as the maximum angular motion of the output gear when the input gear remains fixed.

drives were immediately taken into account. Unfortunately, all these qualities come with a big drawback: price. For a *100:1* harmonic drive gear, in fact, unit price is on the k$ range, which abundantly overcomes the available budget.

Among other kind of gear drives, planetary gears represent a good alternative. Although their performances dwarf if compared to harmonic drives, their quality-to-price ratio is extremely high, and the backlash is still very limited.

7.3.3 Hardware selection

As far as concerns the choice of the motor, the Maxon EC 90 seems to be a viable choice for *joint 1, 2* and *3*: it is relatively small and presents very good performances. Its main characteristics are presented in Table 7.3 (the complete datasheet is available in the Appendix):

As far as concerns *joint 4* and *5*, smaller motors can be used. From the Maxon® catalog, for example, the EC 45 is a good option, since it is very cheap and lightweight.

Motor data	**EC 90**	**EC 45**
Nominal voltage [V]	48	36
Nominal speed [rpm]	1640	3210
Nominal torque [mNm]	494	69.5
Max efficiency [%]	85	81
Weight [g]	648	88
Price [$]	~ 230	~ 80

Table 7.3: Maxon EC 90 and EC 45 data

The required reduction ratio for the motors is computed as follows: first of all, we define ξ_{gear} as the efficiency of the gearbox. From the Maxon datasheets, average efficiency of a planetary gearbox ranges from 60 to 90%; for the first-try calculation, we can suppose and efficiency $\xi_{gear} = 0.75$. We have that:

$$\tau_{out} = \tau_{in} \cdot \eta \cdot \xi_{gear} \tag{7.60}$$

$$\eta = \frac{\tau_{out}}{\tau_{in} \cdot \xi_{gear}} \tag{7.61}$$

From which, using the Maxon EC 90 for *link 1, 2* and *3* and Maxon EC 45 for *link 4* and *5*:

$$\eta_{link\,1} = \frac{12}{0.494 \cdot 0.75} = 32.4 \tag{7.62a}$$

$$\eta_{link\,2} = \frac{53.2}{0.494 \cdot 0.75} = 143.6 \tag{7.62b}$$

$$\eta_{link\,3} = \frac{17.1}{0.494 \cdot 0.75} = 46.2 \tag{7.62c}$$

$$\eta_{link\,4} = \frac{1.68}{0.059 \cdot 0.75} = 37.9 \tag{7.62d}$$

$$\eta_{link\,5} = \frac{1.77}{0.059 \cdot 0.75} = 40 \tag{7.62e}$$

$$\tag{7.62f}$$

From these calculation, we can obtain the (commercial) values of the gear ratios needed: 150 for *link 2*, 50 for *link 3* and 40 for *link 1, 4* and *5*. From the Maxon catalog, Table 7.4 can be compiled.

Gearhead type	Model	Stages	Ratio	Weight	Efficiency	Price	*Joint*
GP 52C	223097	4	150:1	920g	78%	420$	2
GP 52C	223090	3	53:1	620g	83%	411$	3
GP 52C	223089	3	43:1	620g	83%	411$	1, 4, 5

Table 7.4: Gear ratio selection, Maxon datasheet.

The choice of the specific motors for *joints 6* is not of dramatic importance, and since the requirements in term of torque are all less than 0.5 Nm, the reduction gear is not needed.

Thus, relatively cheap actuators can be used. An important requirement, in this case, is to find a motor with a built-in position sensor (e.g. encoder), or at least a motor that allows for the installation of a third party sensor. For example, the Maxon EC 90 could be used.

This motor overcomes the requirements of *joint 6*: in this case however, since price is not a big concern, it is better to have a set of identic motors instead of a specific motor for each joint. Having the same model standardizes

the configuration process and the control loop. Moreover, if test samples have to be attached to the end effector, the excess torque available would be certainly useful.

7.4 Final data

The following data are the same used in the dynamics chapter for the Matlab and Simulink simulations, Section 4.4. The inertia tensors, due to the relative complexity of the structures, are calculated with the aid of SolidWorks *"Mass properties"* tool. In the CAD model, supporting metal structures were designed to hold the motors in place; since this is still not the definitive building design, we omit their corresponding mechanical analysis.

The frames used in this calculation are centered in the center of mass of the body (link+motor+supporting structure) and their axes are parallel and equi-oriented to the ones of the link frames defined with Denavit-Hartenberg's process.

- Assembly 1: *link, motor, support structure, hardware fasteners*

$$m_1 = 2259.34\, g \tag{7.63}$$

$$\overrightarrow{Com}_1 = \begin{bmatrix} -0.13 \\ -1.54 \\ -300.08 \end{bmatrix} mm \tag{7.64}$$

$$I_1 = \begin{bmatrix} 1.289 \cdot 10^8 & 4.951 \cdot 10^3 & -8.773 \cdot 10^4 \\ 4.951 \cdot 10^3 & 1.299 \cdot 10^8 & -1.041 \cdot 10^6 \\ -8.773 \cdot 10^4 & -1.041 \cdot 10^6 & 2.622 \cdot 10^6 \end{bmatrix} g \cdot mm^2 \tag{7.65}$$

- Assembly 2: *link, motor, support structure, hardware fasteners*

$$m_2 = 2424.06\, g \tag{7.66}$$

$$\overrightarrow{Com}_2 = \begin{bmatrix} 392.94 \\ -0.12 \\ 1.43 \end{bmatrix} mm \tag{7.67}$$

$$I_2 = \begin{bmatrix} 3.078 \cdot 10^6 & -9.561 \cdot 10^4 & 1.134 \cdot 10^6 \\ -9.561 \cdot 10^4 & 1.539 \cdot 10^8 & -4.981 \cdot 10^3 \\ 1.134 \cdot 10^6 & -4.981 \cdot 10^3 & 1.546 \cdot 10^8 \end{bmatrix} \; g \cdot mm^2 \tag{7.68}$$

- Assembly 3: *link, motor, support structure, hardware fasteners*

$$m_3 = 2076.64 \, g \tag{7.69}$$

$$\overrightarrow{Com_3} = \begin{bmatrix} 330.04 \\ -6.30 \\ 0 \end{bmatrix} \; mm \tag{7.70}$$

$$I_3 = \begin{bmatrix} 2.761 \cdot 10^6 & -3.754 \cdot 10^6 & 0 \\ -3.754 \cdot 10^6 & 9.971 \cdot 10^7 & 0 \\ 0 & 0 & 1.004 \cdot 10^8 \end{bmatrix} \; g \cdot mm^2 \tag{7.71}$$

- Assembly 4: *link, motor, support structure, hardware fasteners*

$$m_4 = 182.86 \, g \tag{7.72}$$

$$\overrightarrow{Com_4} = \begin{bmatrix} 0 \\ 81.77 \\ 60.82 \end{bmatrix} \; mm \tag{7.73}$$

$$I_4 = \begin{bmatrix} 7.399 \cdot 10^5 & 0 & 0 \\ 0 & 4.335 \cdot 10^5 & -3.101 \cdot 10^5 \\ 0 & -3.101 \cdot 10^5 & 3.739 \cdot 10^5 \end{bmatrix} \; g \cdot mm^2 \tag{7.74}$$

- Assembly 5: *link, motor, support structure, hardware fasteners*

$$m_5 = 158.69 \, g \tag{7.75}$$

$$\overrightarrow{Com_5} = \begin{bmatrix} 0 \\ 50.17 \\ 55.71 \end{bmatrix} \; mm \tag{7.76}$$

$$I_5 = \begin{bmatrix} 4.269 \cdot 10^5 & 0 & 0 \\ 0 & 1.363 \cdot 10^5 & -1.421 \cdot 10^5 \\ 0 & -1.421 \cdot 10^5 & 3.481 \cdot 10^5 \end{bmatrix} g \cdot mm^2 \tag{7.77}$$

- Assembly 6: *link, motor, support structure, hardware fasteners*

$$m_6 = 41.39\, g \tag{7.78}$$

$$\overrightarrow{Com_6} = \begin{bmatrix} 0 \\ 0 \\ 105.25 \end{bmatrix} mm \tag{7.79}$$

$$I_6 = \begin{bmatrix} 9.714 \cdot 10^3 & 0 & 0 \\ 0 & 9.714 \cdot 10^3 & 0 \\ 0 & 0 & 2.118 \cdot 10^3 \end{bmatrix} g \cdot mm^2 \tag{7.80}$$

Finally, the assembled structure has the following characteristics:

- ◇ Total weight: $\sim$7.5 kg
- ◇ Maximum extension of the arm: $\sim$1.3 m
- ◇ Maximum height of the arm: $\sim$2.2 m

7.5 SolidWorks renders

In this section we present the graphical rendering of the manipulator. First, each link assembly (link, motor, support structure, hardware fasteners) is displayed; subsequently, the whole structure is analyzed and rendered in different positons and configurations.

The simulations take into account that material properties and the manifacturing processes for their realization. In order to host the motors, auxiliary *U-shaped* support structures were designed.

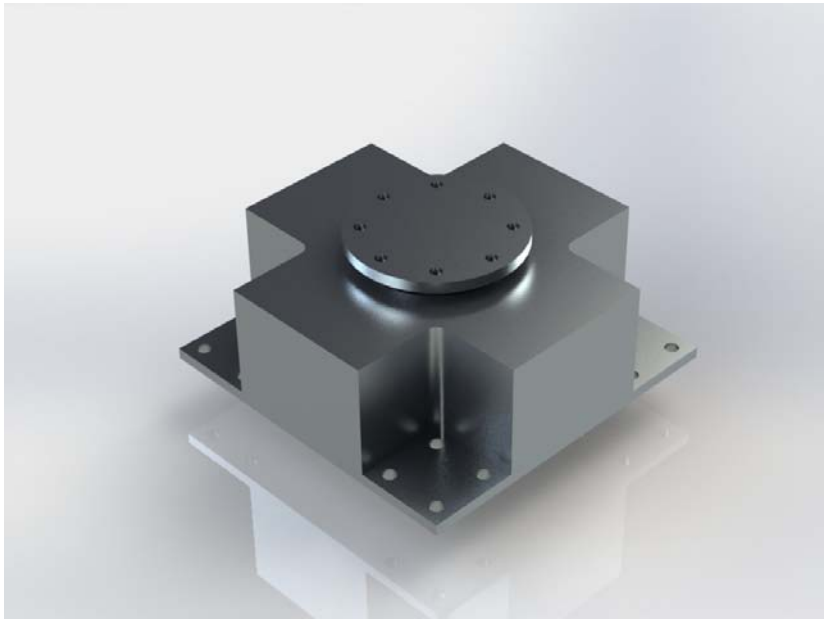


Figure 7.15: Base structure.

Figure 7.16: *Link 1* assembly.

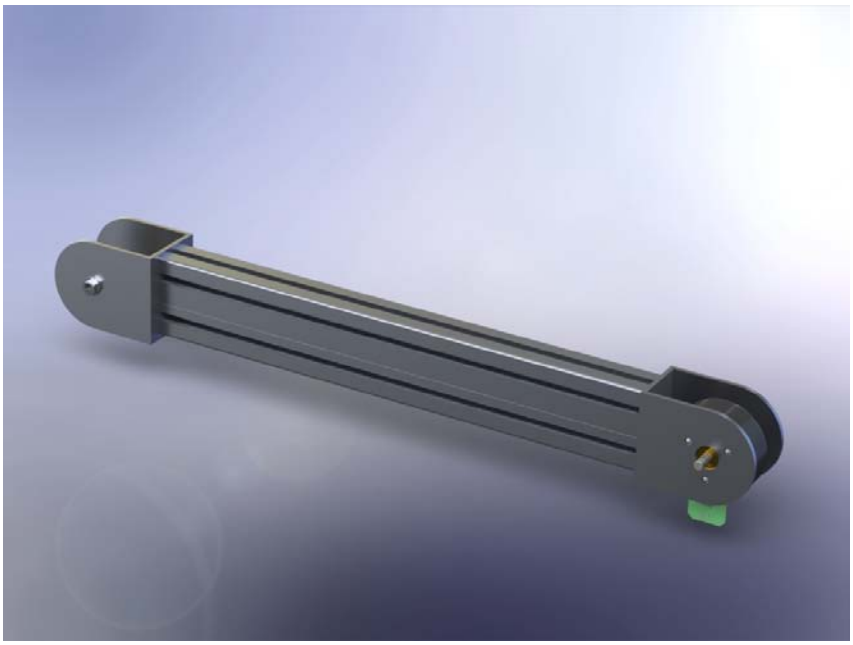


Figure 7.17: *Link 2* assembly.

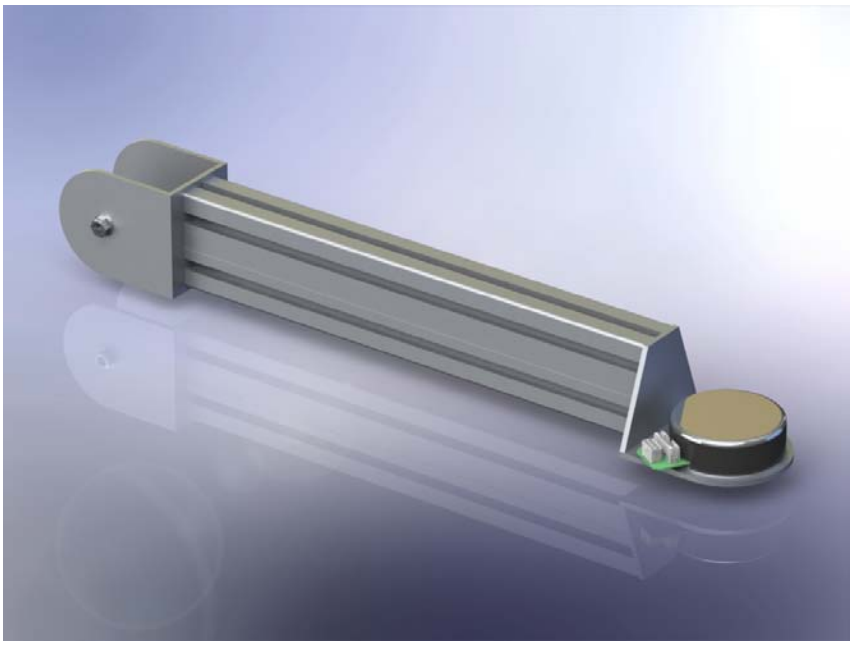


Figure 7.18: *Link 3* assembly.

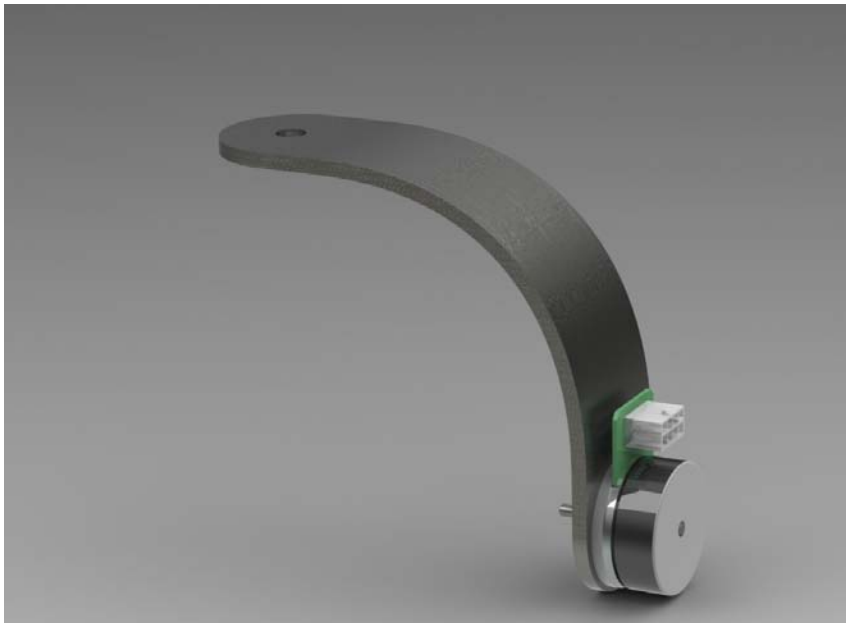


Figure 7.19: *Link 4* assembly.

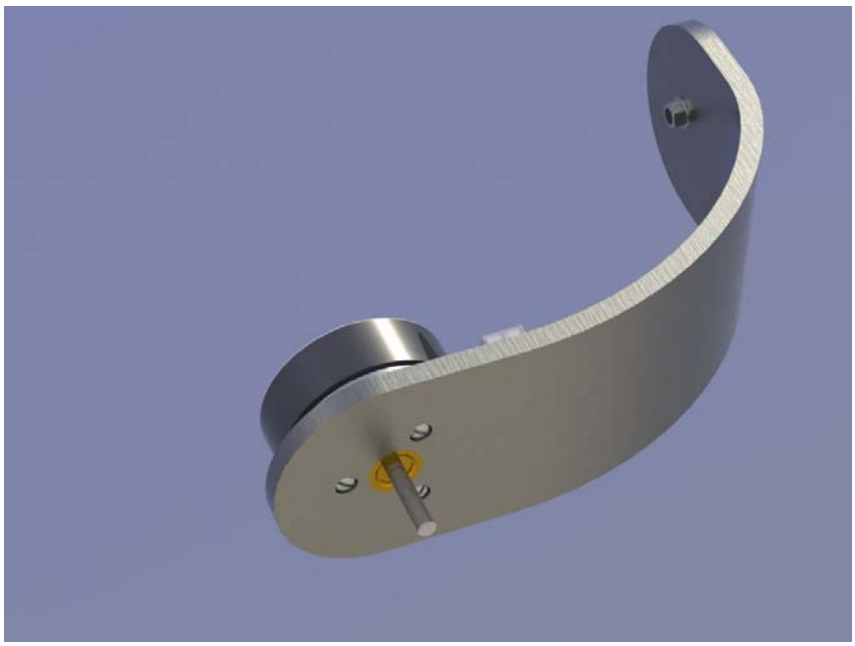


Figure 7.20: *Link 5* assembly.

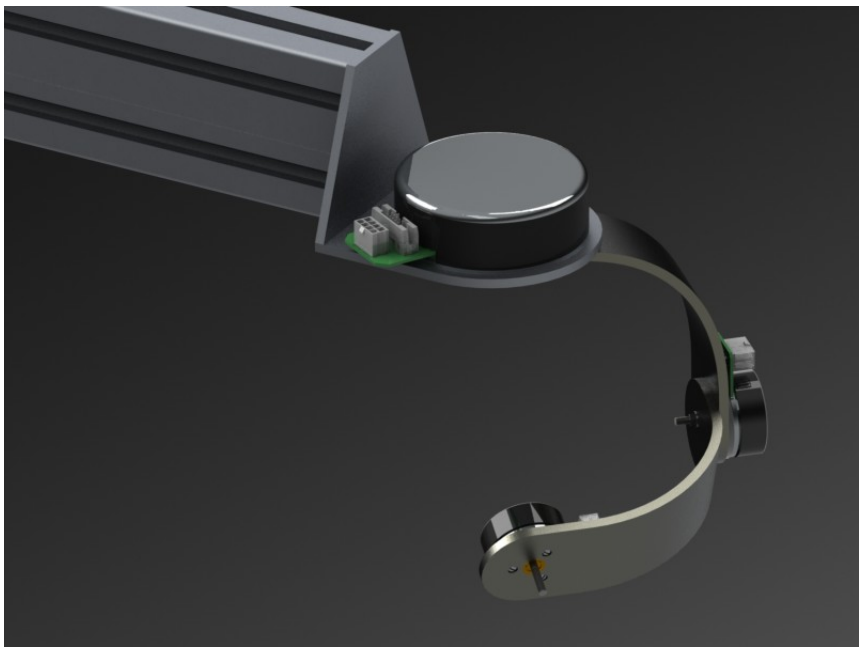


Figure 7.21: *End effector* assembly.

Figure 7.22: *End effector* assembly, rear view.

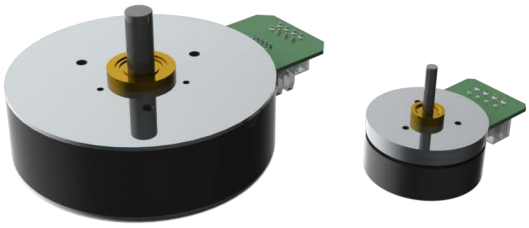


Figure 7.23: The actuators: EC 90 on the left, EC 45 on the right.

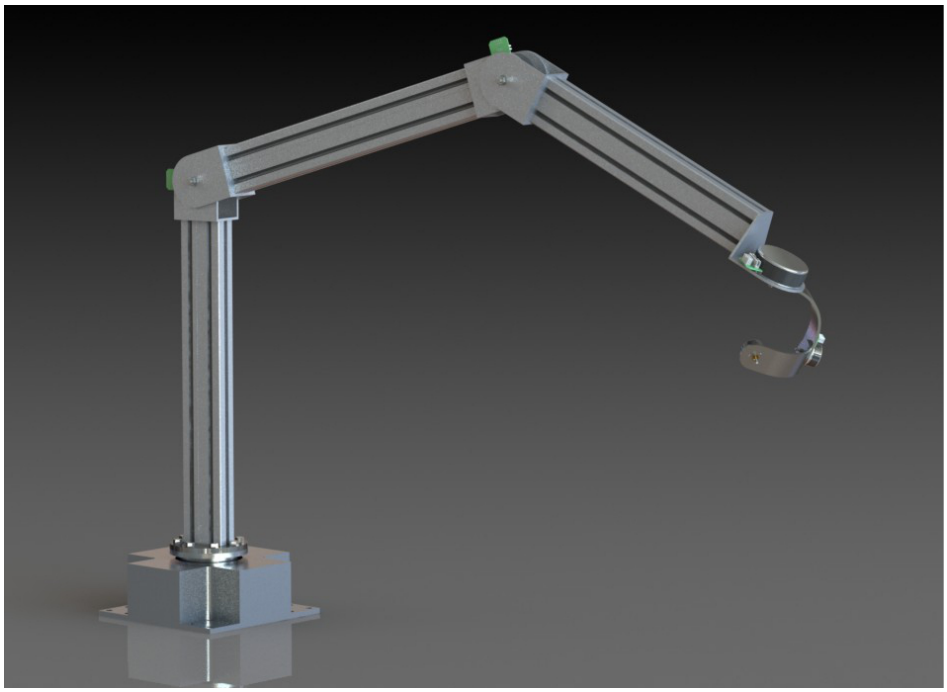
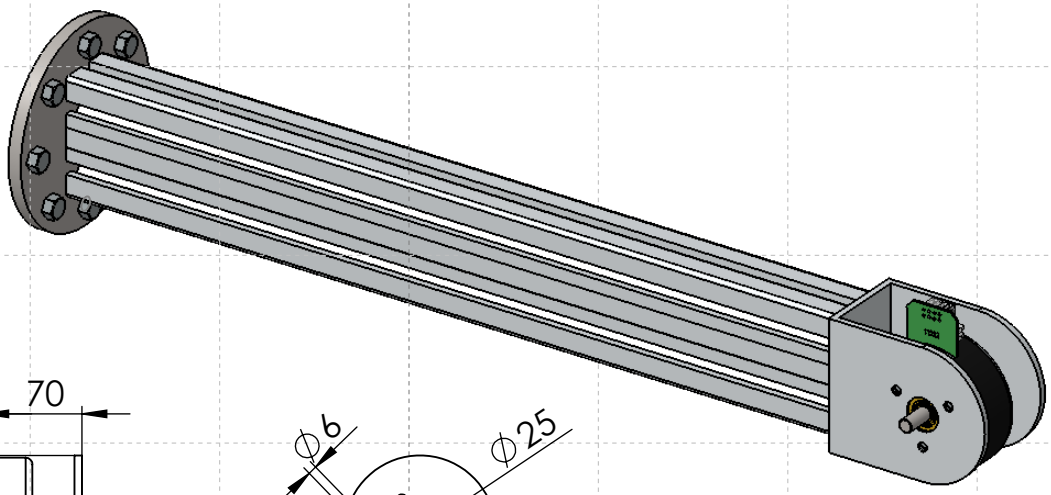
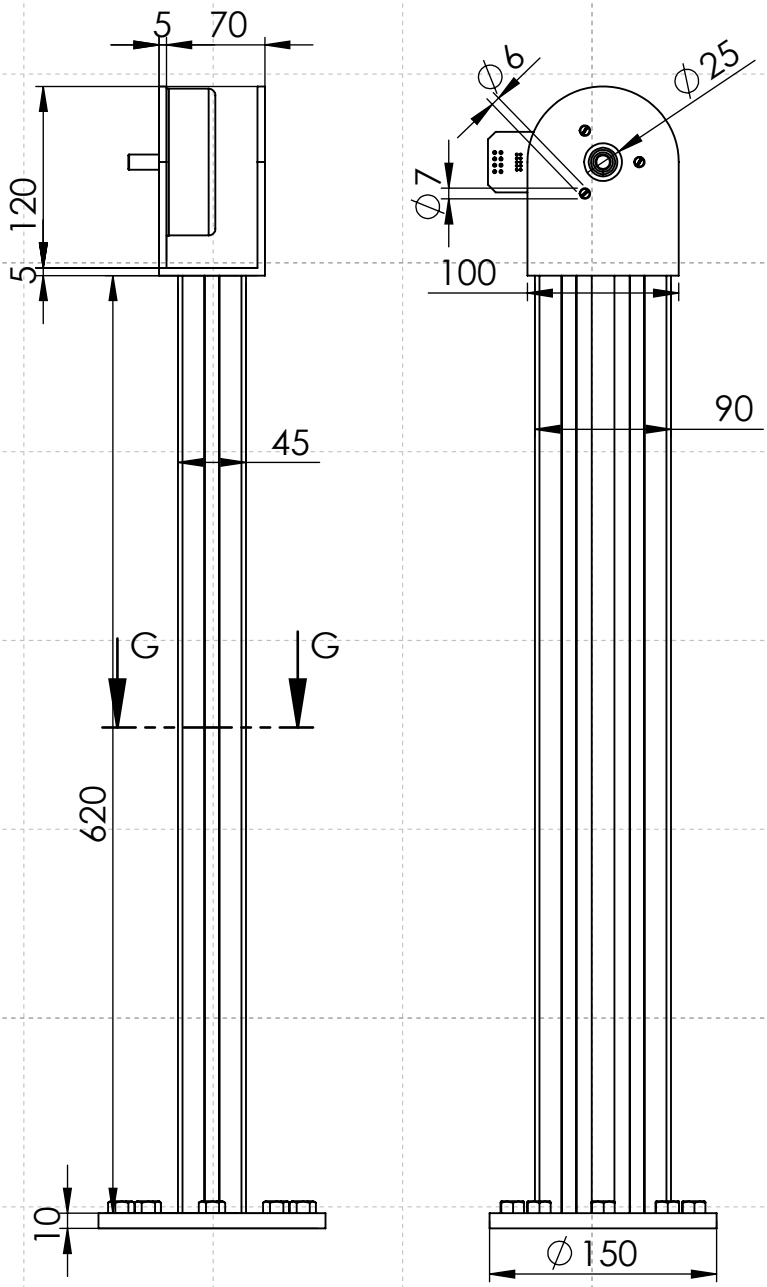
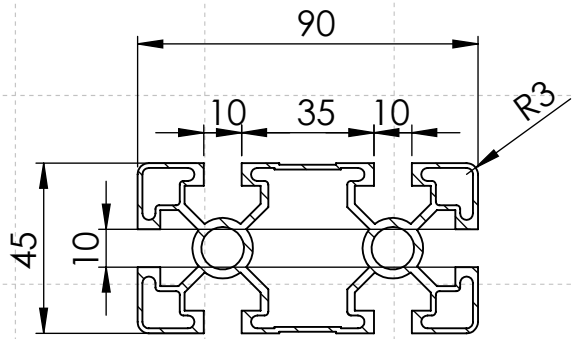


Figure 7.24: Manipulator side view.

Figure 7.25: Manipulator closeup.







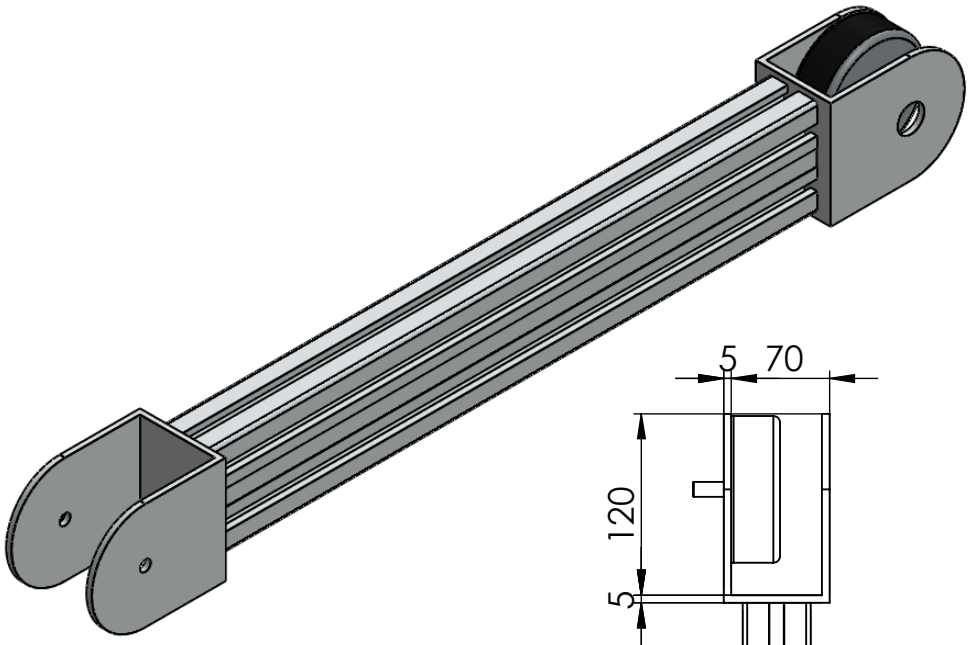
SEZIONE G-G

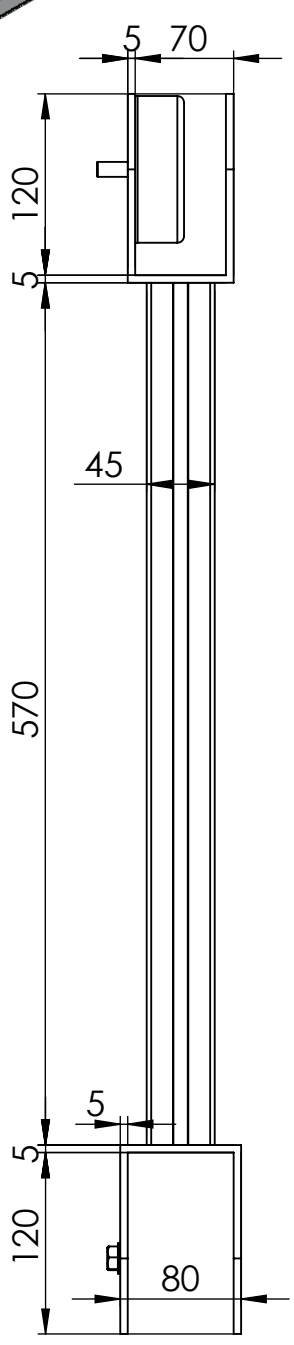
SCALA 1 : 2

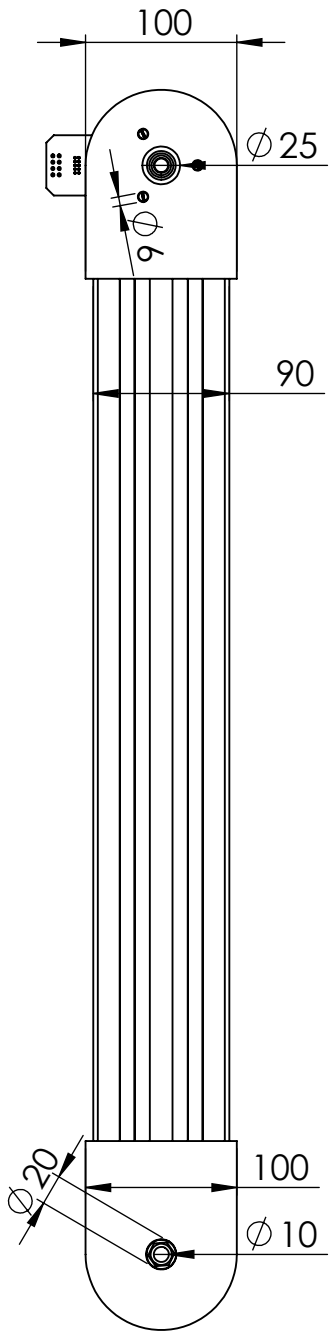
SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: ANGOLARE:	FINITURA: none		SBAVATURA E INTERRUZIONE DEI BORDI NETTI		NON SCALARE IL DISEGNO	REVISIONE

	NOME	FIRMA	DATA			
DISEGNATO	A. A.		27/09/13			
VERIFICATO						
APPROVATO						
FATTO						
QUALITA'						

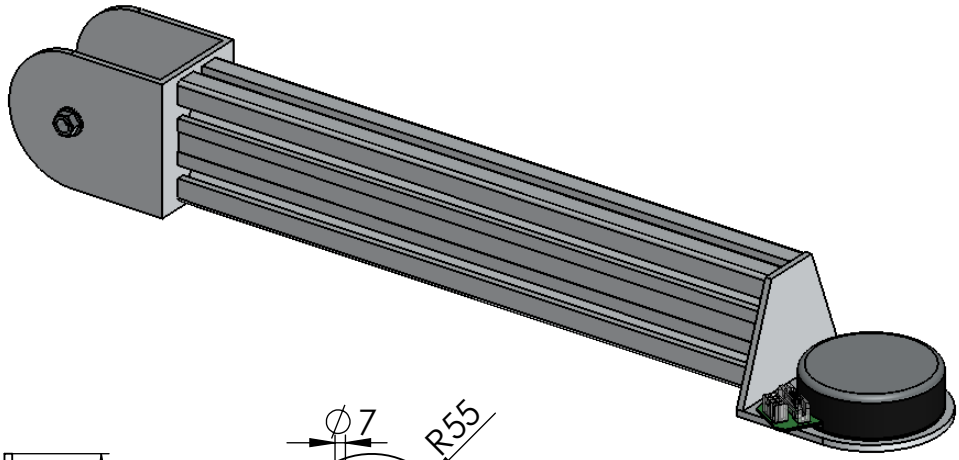
MATERIALE: Alluminium	TITOLO: Assembly 1		
	N. DISEGNO Drawing 1		A4
-	SCALA:1:20	FOGLIO 1 DI 1	

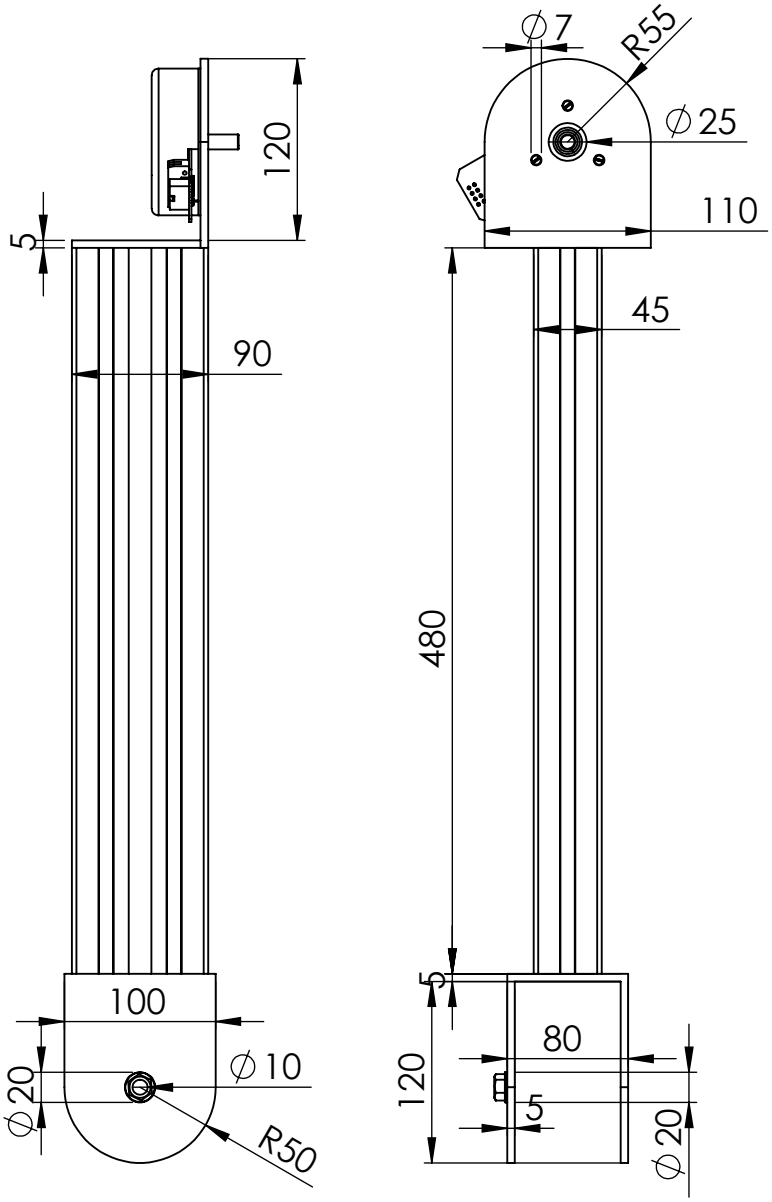






SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: ANGOLARE:	FINITURA: none		SBAVATURA E INTERRUZIONE DEI BORDI NETTI	NON SCALARE IL DISEGNO	REVISIONE	
	NOME	FIRMA	DATA	TITOLO:		
DISEGNATO	A. A.		27/09/13	Assembly 2		
VERIFICATO						
APPROVATO						
FATTO						
QUALITA'				MATERIALE:	N. DISEGNO	A4
				Alluminium	Drawing 2	
				-	SCALA:1:20	FOGLIO 1 DI 1





SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: ANGOLARE:	FINITURA: none	SBAVATURA E INTERRUZIONE DEI BORDI NETTI	NON SCALARE IL DISEGNO / REVISIONE

	NOME	FIRMA	DATA
DISEGNATO	A. A.		27/09/13
VERIFICATO			
APPROVATO			
FATTO			
QUALITA'			

TITOLO:

Assembly 3

MATERIALE: Alluminium

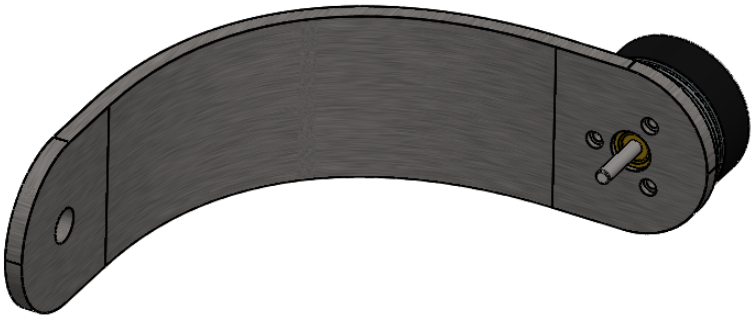
N. DISEGNO

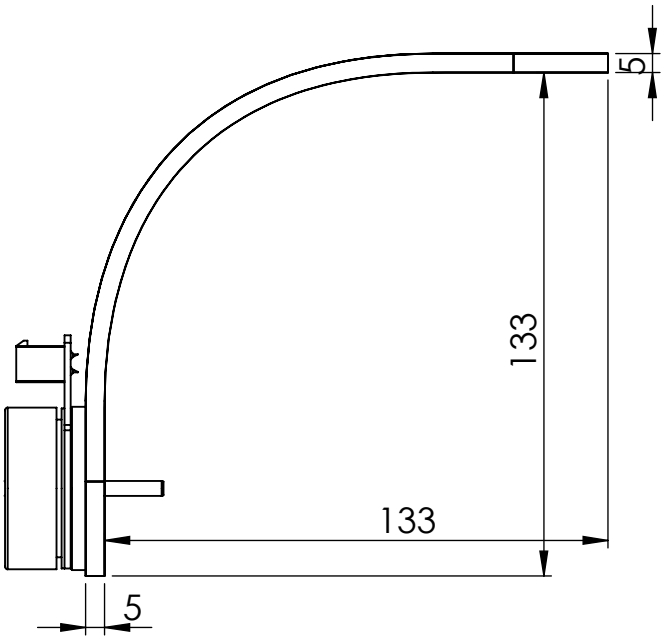
Drawing 3

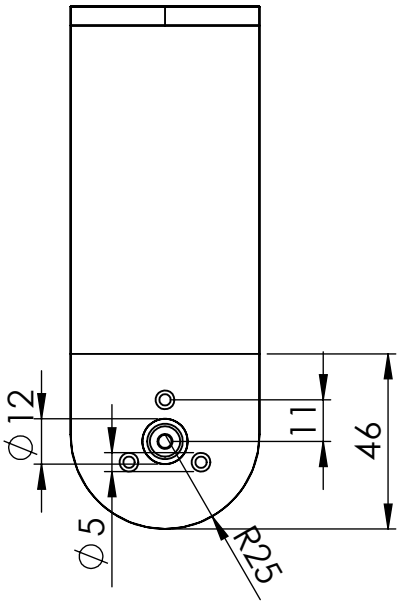
A4

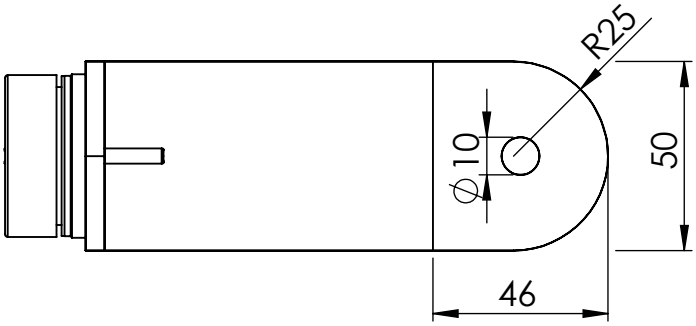
SCALA:1:20

FOGLIO 1 DI 1

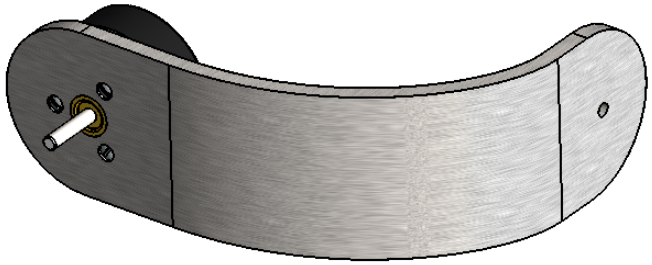


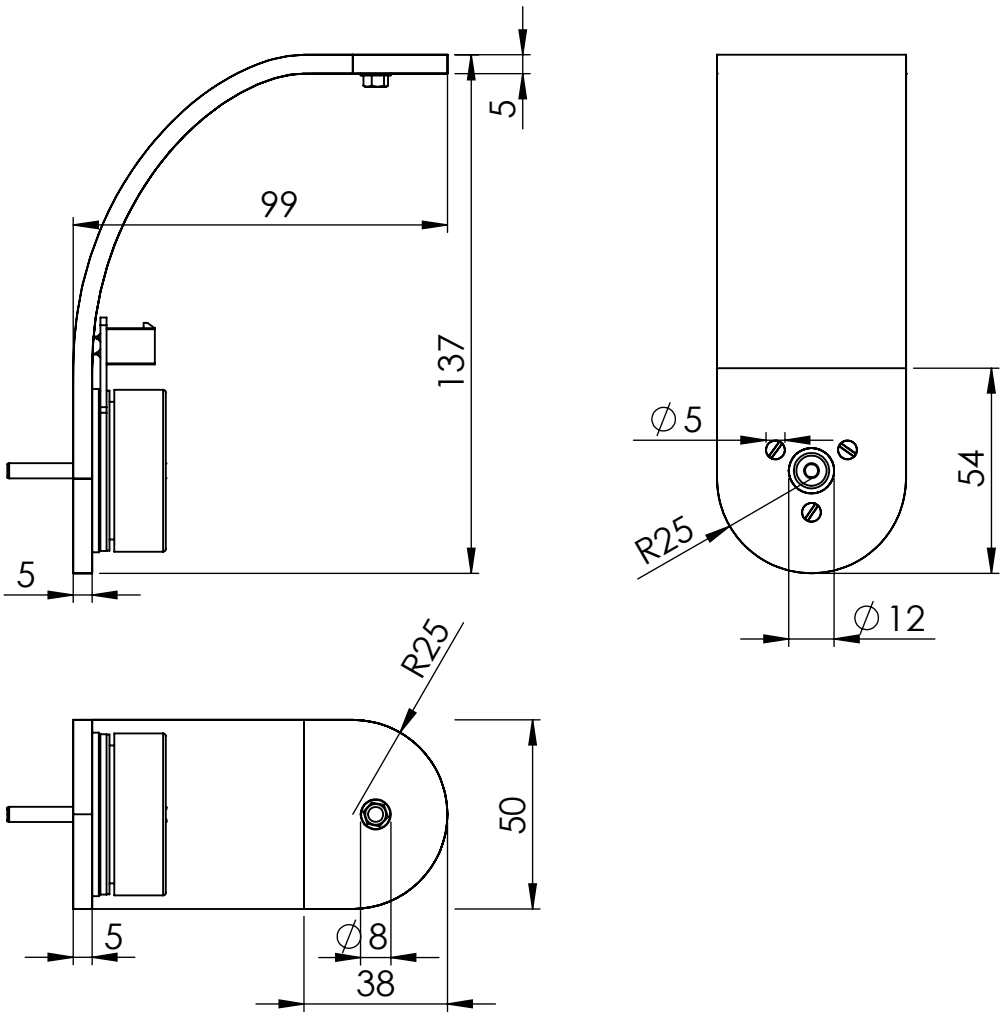






SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: ANGOLARE:	FINITURA: none		SBAVATURA E INTERRUZIONE DEI BORDI NETTI		NON SCALARE IL DISEGNO	REVISIONE
	NOME	FIRMA	DATA		TITOLO:	
DISEGNATO	A. A.		27/09/13		Assembly 4	
VERIFICATO						
APPROVATO						
FATTO						
QUALITA'				MATERIALE: Alluminium	N. DISEGNO Drawing 4	A4
				-	SCALA:1:5	FOGLIO 1 DI 1





SE NON SPECIFICATO: QUOTE IN MILLIMETRI FINITURA SUPERFICIE: TOLLERANZE: LINEARE: ANGOLARE:	FINITURA: none		SBAVATURA E INTERRUZIONE DEI BORDI NETTI		NON SCALARE IL DISEGNO	REVISIONE
	NOME	FIRMA	DATA		TITOLO:	
DISEGNATO	A. A.		27/09/13		Assembly 5	
VERIFICATO						
APPROVATO						
FATTO						
QUALITA'				MATERIALE: Alluminium	N. DISEGNO Drawing 5	A4
				-	SCALA:1:5	FOGLIO 1 DI 1

Chapter 8

Conclusions and future work

Due to the growing number of artificial orbiting objects, space missions for servicing purposes have recently been the subject of attention from industries and agencies [16]. Consequently, there is the need for a simulation facility that allows for reproduction of on-orbit close approaches. The main goal of this thesis is the development of a robot manipulator for the simulation of orbital maneuvers, with particular attention to docking and capture.

As was discussed in the introduction, this work split into several sections: kinematics, dynamics, space trajectory planning, linear control and final sizing.

With reference to the corresponding chapters, it can be stated that the kinematics was carried out successfully, leading to the calculation of the Jacobian matrix, providing the capability of relating joint and cartesian variables. The simulations showed a perfect correspondence with the imposed trajectory laws. The extension of the analysis to the dynamics resulted in a simple, fast reliable algorithm for the computation of the generalized forces involved in the motion. This enabled a wide campaign of trajectory simulations, which were compared with the theoretical model, showing the powers and the limits of Newton's approach.

The core part of this thesis, consisting in the space trajectory analysis, was successfully studied and simulated, and provided an extremely flexible model, able to take into consideration several kinds of modifications to the free orbital relative motion: it was possible, in fact, to simulate rendezvous,

disturbances, impacts and attitude maneuvers.

Above all, the addition of a force sensor to the end effector tip makes the manipulator an active simulation facility: the initial approach trajectory is calculated and imposed by the controller, but in presence of disturbances, the sensing transducer (which can be real, i.e. for contact simulation, or fictious, i.e. for the simulation of ADCS systems) allows for real time, online modifications of the orbital trajectory.
Trial simulations consisting in the combination of some of these effects, proved the correctness of the Matlab model, validating our code and underlining the power of this innovative flexible approach.

The general overview on the linear control of the robot provided a simplified approach to the problem, and a PID control technique was designed. Analyses confirmed the stability of the system to step response as well as to external disturbances. Additionaly, the PID controller, unlike the PD version, revelead a null-error steady state behavior even in the presence of disturbances.

The sizing process, finally, combined all the previous analyses with physical and geometrical parameters. Commercial link sections were chosen after examining bending and buckling problems. A static 3D load analysis, coupled with extrapolation of the dynamic loads from the trajectory simulations, lead to the choice of appropriate actuators and their corresponing gear boxes.

The future development of this thesis will consist in a further refinement of the aspects that time didn't allow to treat properly. For example, an extension of the control scheme might be implemented: a broad campaign of simulations has to be performed in order to estimate the proper gains of the controllers. In addition, we can think of implementing an adaptive control in order to increase the positioning accuracy and to limit delays and jerks. This will ultimately lead to the selection of the sensors and electronics hardware. The dynamic model, after the control sensors and the wiring harness are known, can be further polished by taking into account these extra components.

Obvious modifications will then be made to the CAD model, designing all the interfaces for the connections and for the housing of motors and sensors. Moreover, bearing and fastening hardware selection will be required.

Once all these issues have been cleared, construction and testing will start. Due to its extreme flexibility, the robot will serve as a test bench for multidisciplinary simulations. From docking to rendezvous, from cooperative to uncooperative capture, this manipulator will be the leading character of an advanced testing facility.

Bibliography

[1] G. Acaccia and L. Bruzzone. "A modular robotic system for industrial applications A modular robotic system for industrial applications A modular robotic system for industrial applications". In: *Assembly Automation* 28.2 (2008), pp. 151–162.

[2] *Aluminium Framing Catalog*. URL: `http://www.boschrexroth-_us.com/country_units/america/united_states/sub_websites/brus_dcl/Products`.

[3] *ATI Nano 17 Force Transducer*. URL: `http://www.ati-_ia.com/products/ft/ft_models.aspx?id=Nano17`.

[4] T. Bajd and M. Mihelj. *Robotics*. Springer Verlag, 2010.

[5] M. Bertsch, R. Dal Passo, and L. Giacomelli. *Analisi Matematica*. McGraw-Hill, 2007.

[6] *CISAS - Unipd*. URL: `http://cisas.unipd.it/Attivita.php`.

[7] M. Collins, A. Aldrich, and G. Lunney. *STS 41-G National Space Transportation Systems Program Mission Report*. Tech. rep. Johnson Space Center: NASA, 1984.

[8] J. Craig. *Intorduction to Robotics: Mechanics and Control*. Prentice Hall, 2005.

[9] H. D. Curtis. *Orbital Mechanics for Engineering Students*. Elsevier, 2010.

[10] R. C. Dorf and Bishop R. H. *Modern Control Systems*. 12th. Prentice Hall, 2012.

[11] *European Proximity Operations Simulator (EPOS) - DLR*. URL: `http://www.weblab.dlr.de/rbrt/OOS/EPOS/EPOS.html`.

[12] *Gemini 7*. URL: `http://en.wikipedia.org/wiki/Gemini_7`.

[13] *History of Robots*. URL: `http://en.wikipedia.org/wiki/History_of_robots`.

[14] *International Space Station*. URL: `http://it.wikipedia.org/wiki/Stazione_Spaziale_Internazionale`.

[15] S. Isakowitz, J. Hopkins, and J. Hopkins. *International Reference Guide to Space Launch Systems*. 4th. AIAA, 2004.

[16] A. Long, M. Richards, and D. Hastings. "On-Orbit Servicing: A New Value Proposition for Satellite Design and Operation". In: *Journal of Spacecraft and Rockets* 44.4 (July–August 2007).

[17] J. Y. S. Luh, M. W. Walker, and R. P. Paul. "On-Line Computational Scheme for Me- chanical Manipulators". In: *ASME Journal of Dynamic Systems, Measurement, and Control* (1980).

[18] T. H. G. Megson. *An introduction to Aircraft Structural Analysis*. Elsevier, 2010.

[19] C. Melchiorri. *Trajectory Planning for Robot Manipulators, Part 4*. URL: `http://www-_lar.deis.unibo.it/people/cmelchiorri/Files_Robotica/FIR_07_Traj_4.pdf`.

[20] J. Pires. *Industrial Robots Programming*. Springer Verlag, 2007.

[21] J. Saleh et al. "To Reduce or Extend a Spacecraft Design Lifetime". In: *Journal of Spacecraft and Rockets* 43.1 (2006), pp. 207–217.

[22] L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators*. Springer Verlag, 2010.

[23] L. Sciavicco and B. Siciliano. *Robotics: Modelling, Planning and Control*. Springer Verlag, 2010.

[24] P. B. Selding. *Intelsat Signs Up for MDA's Satellite Refueling Service*. URL: http://www.spacenews.com/article/intelsat-_signs-_satellite-_refueling-_service.

[25] P. B. Selding. *MDA, Intelsat Scrap In-orbit Servicing Deal*. URL: http://www.spacenews.com/article/mda-_intelsat-_scrap-_orbit-_servicing-_deal.

[26] *SimMechanics 2 User Guide*. The MathWorks, Inc. 2007.

[27] *Space Infrastructure Servicing*. URL: http://en.wikipedia.org/wiki/Space_Infrastructure_Servicing.

[28] *Space Robotics Technical Committee*. URL: http://ewh.ieee.org/cmte/ras/tc/spacerobotics/.

[29] M. Spong. *Robot dynamics and control*. Wiley, 1989.

[30] B. Sullivan. "Technical and Economic Feasibility of Telerobotic On Orbit Satellite Servicing". PhD thesis. University of Maryland, 2005.

[31] B. Sullivan and D. Akin. "A Survey of Serviceable Spacecraft Failures". In: *AIAA Paper* (2001).

[32] K. R. Symon. *Mechanics*. Addison-Wesley, 1971.

[33] *The Robot*. URL: http://en.wikipedia.org/wiki/Robot.

[34] *What are Gearmotors?* URL: http://www.motioncontroltips.com/2011/10/04/gearing/.

List of Figures

List of Tables

Appendix A

Matlab Scripts

We report in this section the Matlab scripts used in the simulation. The main script is called `main.m`; the functions invoked during the simulations (i.e. for matrix inversion, for direct and inverse kinematic solving, for drawings, etc.) are:

- `bending_analysis.m`: for the analysis of bending in the structure
- `buckling_analysis.m`: for the analysis of buckling for *link 1*
- `Cylinder.m`: drawing purposes
- `direct_kin.m`: for the direct kinematics calculation
- `disframe3.m`: draws the coordinate frames
- `geom.m`: stores the inertial parameters
- `inv_kin.m`: for the inverse kinematics calculation
- `jaco.m`: for the calculation of the Jacobian
- `jaco_syms.m`: for the calculation of the symbolic Jacobian
- `rot_matrix.m`: obtains the rotational matrices
- `rot_matrix_sym.m`: obtains the rotational matrices in symbolic form
- `rpy_wrist.m`: solves the end effector orientation

◇ `sim_starter.m`: calculates the initial conditions for the simulations

◇ `static_analysis.m`: for the analysis of static loading in the structure

In the next pages it is possible to find the complete Matlab scripts.

```
% main.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

clear all
figure
global l1 l2 l3 d1 d2 d3 d4 d a alp att Tatt

sim=0; % 1=ok simulation, 0=no simulation
tic

format long

% gravity selection
gr=1;  % 1=ok gravity, 0=no gravity
g=9.81;

T=5; % simulation period
dt=0.05; % simulation step

% target
l1=0.664;
l2=0.720;
l3=0.625;
d1=0.0;
d2=0.0;
d3=0.0;
d4=0.160;
alp=[0 pi/2 0  pi/2  pi/2  pi/2];
a=[0 0 l2 l3 0 0];
d=[0  d2  d3  d4 0 0];

% initial situation computation
R=0.4; % radius of circle
x_or(:,1)=[0.5 0.7 0.3 0 0 0]; % initial position
xfin=x_or(:,1);
x_dot(:,1)=[0 0 0 0 0 0]'; % initial velocity
att=[0.1 45 0.1]; % angle in degrees

[th]=sim_starter(x_or);

q(:,1)=[th]'; % radians
q_dot=[];

[T01 T12 T23 T34 T45 T56]=rot_matrix(a, th, alp, d); % gets the rotational matrixes

sys=[T^5/32 T^4/16 T^3/8 pi; 5*T^4 4*T^3 3*T^2 0; 20*T^3 12*T^2 6*T 0];
sol=sys(1:3,1:3)\sys(:,4);
aa=sol(1);
bb=sol(2);
cc=sol(3);

t=0;
```

```
i=1;

ex_t=toc;

% plot initialization
if sim==1
    fig=figure;
    set(fig,'units','normalized','outerposition',[0 0 1 1]);
    view(30,30);
end

% geometric infos
[I,m,com]=geom;

while t<T
    cla
    xlabel('x');
    ylabel('y');
    zlabel('z');
    % step i
    tic
    if i>1;
        th=aa*t^5+bb*t^4+cc*t^3;
        thd=5*aa*t^4+4*bb*t^3+3*cc*t^2;
        thdd=20*aa*t^3+12*bb*t^2+6*cc*t;
        x_dot(:,i)=[R*sin(th)*thd;
            R*cos(th)*thd;
            0;
            0.0;
            0.0;
            0.0];
    end

    Jin=jaco(q(:,i)); % Jacobian
    q_dot(:,i+1)=Jin*x_dot(:,i);
    q_ddot(:,i+1)=(q_dot(:,i+1) q_dot(:,i))/dt;
    q(:,i+1)=q(:,i)+q_dot(:,i+1)*dt;
    [T01 T12 T23 T34 T45 T56]=rot_matrix_jaco(q(:,i)); % gets the rotational matrixes

    T21=T12';
    T32=T23';
    T43=T34';
    T54=T45';
    T65=T56';

    [t01 t12 t23 t34 t45 t56]=rot_matrix_jaco(q(:,i+1)); % gets the rotational matrixes
    t03=t01*t12*t23;
    t04=t01*t12*t23*t34;

    t06=eye(4);
    t06(1:3,1:3)=[rpy(att(1),att(2),att(3))];
    t06(:,4)=t34(:,4);
    t36=t03'*t06;
    [th4 th5 th6]=rpy_wrist_23J(t36);
    q(4:6,i+1)=[th4 th5 th6];

    R01=T01(1:3,1:3);
    R12=T12(1:3,1:3);
    R23=T23(1:3,1:3);
    R34=T34(1:3,1:3);
    R45=T45(1:3,1:3);
    R56=T56(1:3,1:3);
    R10=R01';
    R21=R12';
    R32=R23';
    R43=R34';
    R54=R45';
    R65=R56';

```

```
126     % acceleration analysis > Outward iteration
127
128     P_i=[T01(1:3,4) T12(1:3,4) T23(1:3,4) T34(1:3,4) T45(1:3,4) T56(1:3,4)];
129     R_i(:,:,1)=R10;
130     R_i(:,:,2)=R21;
131     R_i(:,:,3)=R32;
132     R_i(:,:,4)=R43;
133     R_i(:,:,5)=R54;
134     R_i(:,:,6)=R65;
135
136     Vd_0=g*[0 0 gr]';
137
138     W(:,1,i)=R_i(:,:,1)*[0 0 q_dot(1,i)]';
139         W_d(:,1,i)=[0 0 q_ddot(1,i)]';
140         V(:,1,i)=[0 0 0]';
141         V_d(:,1,i)=Vd_0;
142         Vc(:,1,i)=[0 0 0]';
143         Vc_d(:,1,i)=Vd_0;
144         F(:,1,i)=m(1)*Vc_d(:,1,i);
145         N(:,1,i)=I(:,:,1)*W_d(:,1,i)+cross(W(:,1,i),I(:,:,1)*W(:,1,i));
146
147     for j=1:6
148         if j>1
149             W(:,j,i)=R_i(:,:,j)*W(:,j-1,i)+[0 0 q_dot(j,i)]'; % angular velocity
150             W_d(:,j,i)=R_i(:,:,j)*W_d(:,j-1,i)+cross(R_i(:,:,j)*W(:,j-1,i),[0 0 q_dot(j,
                    i)]')+[0 0 q_ddot(j,i)]'; % angular velocity dot
151             V(:,j,i)=R_i(:,:,j)*(V(:,j-1,i)+cross(W(:,j-1,i),P_i(:,j))); % linear
                    velocity of link
152             V_d(:,j,i)=R_i(:,:,j)*(V_d(:,j-1,i)+cross(W_d(:,j-1,i),P_i(:,j))+cross(W(:,j
                    -1,i),cross(W(:,j-1,i),P_i(:,j))));
153             Vc(:,j,i)=V(:,j,i)+cross(W(:,j,i),com(:,j));
154             Vc_d(:,j,i)=V_d(:,j,i)+cross(W_d(:,j,i),com(:,j))+cross(W(:,j,i),cross(W(:,j
                    ,i),com(:,j)));
155             F(:,j,i)=m(j)*Vc_d(:,j,i);
156             N(:,j,i)=I(:,:,j)*W_d(:,j,i)+cross(W(:,j,i),I(:,:,j)*W(:,j,i));
157         end
158     end
159
160     % acceleration analysis > Inward iteration
161
162     f(:,7,i)=[0 0 0]';
163     n(:,7,i)=[0 0 0]';
164
165     f(:,6,i)=F(:,6,i);
166     n(:,6,i)=N(:,6,i)+cross(com(:,6),F(:,6,i));
167     tau(i,6)=n(3,6,i);
168
169     for j=5:-1:1
170         f(:,j,i)=R_i(:,:,j+1)'*f(:,j+1,i)+F(:,j,i);
171         n(:,j,i)=N(:,j,i)+R_i(:,:,j+1)'*n(:,j+1,i)+cross(com(:,j),F(:,j,i))+cross(P_i(:,
                j+1),R_i(:,:,j+1)'*f(:,j+1,i));
172         tau(i,j)=n(3,j,i);
173     end
174
175
176     check(:,:,i)=T01*T12*T23*T34*T45*T56;
177
178     %% Plot
179     if sim==1
180         % Plot
181         Or=[0 0 0 1]';
182         P_check=T01*T12*T23*T34*Or;
183
184         disframe3(T01,0.06);
185         disframe3(T01*T12,0.06);
186         disframe3(T01*T12*T23,0.06);
187         disframe3(T01*T12*T23*T34,0.06);
188         disframe3(T01*T12*T23*T34*T45,0.06);
189         disframe3(T01*T12*T23*T34*T45*T56,0.5);
190         disframe3(Tatt,0.5);
```

```
191
192         P(:,2)=T01*Or; % origin
193         plot3(P(1,2),P(2,2),P(3,2),'o')
194         plot3([0 P(1,2)],[0 P(2,2)],[0 P(3,2)],'b')
195
196         P(:,3)=T01*T12*Or; % origin
197         P3=T01*T12*[l2 0 0 1]';
198         P4=T01*T12*[l2 0  d3 1]';
199         plot3(P(1,3),P(2,3),P(3,3),'o')
200         plot3([P3(1) P(1,3)],[P3(2) P(2,3)],[P3(3) P(3,3)],'r')
201
202         P(:,4)=T01*T12*T23*Or; % origin
203         plot3(P(1,4),P(2,4),P(3,4),'o')
204         P5=T01*T12*T23*[l3 0 0 1]';
205         plot3([P(1,4) P3(1)],[P(2,4) P3(2)],[P(3,4) P3(3)],'k')
206         plot3([P(1,4) P5(1)],[P(2,4) P5(2)],[P(3,4) P5(3)],'k')
207
208         P(:,6)=T01*T12*T23*T34*Or; % origin
209         plot3(P(1,6),P(2,6),P(3,6),'o')
210         plot3([P(1,6) P5(1)],[P(2,6) P5(2)],[P(3,6) P5(3)])
211
212         real_traj(:,i)=[P(1,6) P(2,6) P(3,6)]';
213
214         % actuators
215         Dz=[0 0 0.02 1]'; % actuators height
216         mDz=[0 0  0.02 1]'; % actuators height
217
218         Pz(:,3)=T01*T12*Dz; % origin
219         Pzz(:,3)=T01*T12*(mDz); % origin
220
221         Pz(:,4)=T01*T12*T23*Dz; % origin
222         Pzz(:,4)=T01*T12*T23*(mDz); % origin
223
224         Pz(:,5)=T01*T12*T23*T34*Dz; % origin
225         Pzz(:,5)=T01*T12*T23*T34*(mDz); % origin
226
227         Cylinder(P(1:3,1)+Dz(1:3) [0 0 l1]',P(1:3,1) Dz(1:3) [0 0 l1]',0.03,30,'r',1,0)
228         Cylinder(Pz(1:3,3),Pzz(1:3,3),0.03,30,'r',1,0)
229         Cylinder(Pz(1:3,4),Pzz(1:3,4),0.03,30,'r',1,0)
230         Cylinder(Pz(1:3,5),Pzz(1:3,5),0.03,30,'r',1,0)
231
232         % links
233         Cylinder(P(1:3,2),[0 0  l1]',0.01,30,[0.52 0.52 0.52],1,0)
234         Cylinder(P(1:3,2),P(1:3,3),0.01,30,[0.52 0.52 0.52],1,0)
235         Cylinder(P(1:3,3),P3(1:3),0.01,30,[0.52 0.52 0.52],1,0)
236         Cylinder(P3(1:3),P(1:3,4),0.01,30,[0.52 0.52 0.52],1,0)
237         Cylinder(P(1:3,4),P5(1:3),0.01,30,[0.52 0.52 0.52],1,0)
238         Cylinder(P(1:3,6),P5(1:3),0.01,30,[0.52 0.52 0.52],1,0)
239
240         grid on
241         axis([ 0.3 1  0.7 0.8  0.3 0.8])
242         axis square
243         hold on
244
245         % real trajectory
246         plot3(real_traj(1,:),real_traj(2,:),real_traj(3,:),'r')
247     end
248
249     i=i+1;
250     t=t+dt;
251     time(i)=t;
252     ex_t(i)=toc;
253
254     if sim==1
255         pause(0.001)
256     end
257 end
258
259
260 % Data storing
```

```
261
262 ic1=q(1,1);
263 ic2=q(2,1);
264 ic3=q(3,1);
265 ic4=q(4,1);
266 ic5=q(5,1);
267 ic6=q(6,1);
268
269 format long
270 taut=tau';
271 save('my_file.mat','tau');
272
273 t1 = [0:dt:T]';
274 m = taut;
275 M = repmat(m,[1 1 length(t1)]);
276 data.time=t1;
277 data.signals.values = M;
278 data.signals.dimensions=[size(taut,1) size(taut,2)];
279
280 t1 = [0:dt:T]';
281 m = q;
282 M = repmat(m,[1 1 length(t1)]);
283 pos.time=t1;
284 pos.signals.values = M;
285 pos.signals.dimensions=[size(q,1) size(q,2)];
286
287 t1 = [0:dt:T]';
288 m = q_dot;
289 M = repmat(m,[1 1 length(t1)]);
290 veloc.time=t1;
291 veloc.signals.values = M;
292 veloc.signals.dimensions=[size(q_dot,1) size(q_dot,2)];
293
294 t1 = [0:dt:T]';
295 m = q_ddot;
296 M = repmat(m,[1 1 length(t1)]);
297 acc.time=t1;
298 acc.signals.values = M;
299 acc.signals.dimensions=[size(q_ddot,1) size(q_ddot,2)];
300
301
302 % final plot
303 hold on
304 [T01 T12 T23 T34]=rot_matrix_jaco(q(:,1)); % gets the rotational matrixes
305 Or=[0 0 0 1]';
306
307 xlabel('x [m]')
308 ylabel('y [m]')
309 zlabel('z [m]')
310
311 P_check=T01*T12*T23*T34*Or;
312
313 disframe3(T01,0.06);
314 disframe3(T01*T12,0.06);
315 disframe3(T01*T12*T23,0.06);
316 disframe3(T01*T12*T23*T34,0.06);
317
318 P(:,2)=T01*Or; % origin
319 plot3(P(1,2),P(2,2),P(3,2),'o')
320 plot3([0 P(1,2)],[0 P(2,2)],[0 P(3,2)],'b')
321
322 P(:,3)=T01*T12*Or; % origin
323 P3=T01*T12*[l2 0 0 1]';
324 P4=T01*T12*[l2 0 d3 1]';
325 plot3(P(1,3),P(2,3),P(3,3),'o')
326 plot3([P3(1) P(1,3)],[P3(2) P(2,3)],[P3(3) P(3,3)],'r')
327
328 P(:,4)=T01*T12*T23*Or; % origin
329 plot3(P(1,4),P(2,4),P(3,4),'o')
330 P5=T01*T12*T23*[l3 0 0 1]';
```

```
plot3([P(1,4) P3(1)],[P(2,4) P3(2)],[P(3,4) P3(3)],'k')
plot3([P(1,4) P5(1)],[P(2,4) P5(2)],[P(3,4) P5(3)],'k')

P(:,6)=T01*T12*T23*T34*Or; % origin
plot3(P(1,6),P(2,6),P(3,6),'o')
plot3([P(1,6) P5(1)],[P(2,6) P5(2)],[P(3,6) P5(3)])

fig=figure;
set(fig,'units','normalized','outerposition',[0 0 1 1]);

semilogy(ex_t)
grid on
hold on
xlabel('iteration');
ylabel('execution time [s]');
title('Execution time')
semilogy(ex_t,'o')

fig=figure;
set(fig,'units','normalized','outerposition',[0 0 1 1]);

lt=length(time) 2;

%% Positions
subplot(4,6,1)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q1 position')
plot(time,q(1,:)*180/pi)

subplot(4,6,2)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q2 position')
plot(time,q(2,:)*180/pi)

subplot(4,6,3)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q3 position')
plot(time,q(3,:)*180/pi)

subplot(4,6,4)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q4 position')
plot(time,q(4,:)*180/pi)

subplot(4,6,5)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q5 position')
plot(time,q(5,:)*180/pi)

subplot(4,6,6)
grid on
hold on
xlabel('time [s]');
ylabel('angle [deg]');
title('q6 position')
```

```
401 plot(time,q(6,:)*180/pi)
402
403 %% Velocities
404
405 subplot(4,6,7)
406 grid on
407 hold on
408 xlabel('time [s]');
409 ylabel('velocity [rad/s]');
410 title('q1 velocity')
411 plot(time,q_dot(1,:),'r')
412
413 subplot(4,6,8)
414 grid on
415 hold on
416 xlabel('time [s]');
417 ylabel('velocity [rad/s]');
418 title('q2 velocity')
419 plot(time,q_dot(2,:),'r')
420
421 subplot(4,6,9)
422 grid on
423 hold on
424 xlabel('time [s]');
425 ylabel('velocity [rad/s]');
426 title('q3 velocity')
427 plot(time,q_dot(3,:),'r')
428
429 subplot(4,6,10)
430 grid on
431 hold on
432 xlabel('time [s]');
433 ylabel('velocity [rad/s]');
434 title('q4 velocity')
435 plot(time,q_dot(4,:),'r')
436
437 subplot(4,6,11)
438 grid on
439 hold on
440 xlabel('time [s]');
441 ylabel('velocity [rad/s]');
442 title('q5 velocity')
443 plot(time,q_dot(5,:),'r')
444
445 subplot(4,6,12)
446 grid on
447 hold on
448 xlabel('time [s]');
449 ylabel('velocity [rad/s]');
450 title('q6 velocity')
451 plot(time,q_dot(6,:),'r')
452
453 %% Accelerations
454 subplot(4,6,13)
455 grid on
456 hold on
457 xlabel('time [s]');
458 ylabel('acceleration [rad/s^2]');
459 title('q1 acceleration')
460 plot(time(1:lt),q_ddot(1,(1:lt)),'r')
461
462 subplot(4,6,14)
463 grid on
464 hold on
465 xlabel('time [s]');
466 ylabel('acceleration [rad/s^2]');
467 title('q2 acceleration')
468 plot(time(1:lt),q_ddot(2,(1:lt)),'r')
469
470 subplot(4,6,15)
```

```
grid on
hold on
xlabel('time [s]');
ylabel('acceleration [rad/s^2]');
title('q3 acceleration')
plot(time(1:lt),q_ddot(3,(1:lt)),'r')

subplot(4,6,16)
grid on
hold on
xlabel('time [s]');
ylabel('acceleration [rad/s^2]');
title('q4 acceleration')
plot(time(1:lt),q_ddot(4,(1:lt)),'r')

subplot(4,6,17)
grid on
hold on
xlabel('time [s]');
ylabel('acceleration [rad/s^2]');
title('q5 acceleration')
plot(time(1:lt),q_ddot(5,(1:lt)),'r')

subplot(4,6,18)
grid on
hold on
xlabel('time [s]');
ylabel('acceleration [rad/s^2]');
title('q6 acceleration')
plot(time(1:lt),q_ddot(6,(1:lt)),'r')


%% Torques
subplot(2,3,1)
grid on
hold on
xlabel('time [s]');
ylabel('torque [Nm]');
title('q1 torque','Fontsize',18)
plot(time((1:lt)),tau((1:lt),1),'k')
%plot(time((1:lt)),tau2((1:lt),1),' ')
[ym,xm]=min(tau(:,1));
[yM,xM]=max(tau(:,1));
plot(time(xm),ym,'o','markerfacecolor','r')
plot(time(xM),yM,'o','markerfacecolor','r')
text(1.1*time(xm),ym,['min=' num2str(ym)],'Fontsize',13)
text(1.1*time(xM),yM,['MAX=' num2str(yM)],'Fontsize',13)

subplot(2,3,2)
grid on
hold on
xlabel('time [s]');
ylabel('torque [Nm]');
title('q2 torque','Fontsize',18)
plot(time(1:lt),tau((1:lt),2),'k')
%plot(time((1:lt)),tau2((1:lt),2),' ')
[ym,xm]=min(tau(:,2));
[yM,xM]=max(tau(:,2));
plot(time(xm),ym,'o','markerfacecolor','r')
plot(time(xM),yM,'o','markerfacecolor','r')
text(1.1*time(xm),ym,['min=' num2str(ym)],'Fontsize',13)
text(1.1*time(xM),yM,['MAX=' num2str(yM)],'Fontsize',13)

subplot(2,3,3)
grid on
hold on
xlabel('time [s]');
ylabel('torque [Nm]');
title('q3 torque','Fontsize',18)
plot(time(1:lt),tau((1:lt),3),'k')
```

```
%plot(time((1:lt)),tau2((1:lt),3),' ')
[ym,xm]=min(tau(:,3));
[yM,xM]=max(tau(:,3));
plot(time(xm),ym,'o','markerfacecolor','r')
plot(time(xM),yM,'o','markerfacecolor','r')
text(1.1*time(xm),ym,['min=' num2str(ym)],'Fontsize',13)
text(1.1*time(xM),yM,['MAX=' num2str(yM)],'Fontsize',13)

subplot(2,3,4)
grid on
hold on
xlabel('time [s]');
ylabel('torque [Nm]');
title('q4 torque','Fontsize',18)
plot(time(1:lt),tau((1:lt),4),'k')
%plot(time((1:lt)),tau2((1:lt),4),' ')
[ym,xm]=min(tau(:,4));
[yM,xM]=max(tau(:,4));
plot(time(xm),ym,'o','markerfacecolor','r')
plot(time(xM),yM,'o','markerfacecolor','r')
text(1.1*time(xm),ym,['min=' num2str(ym)],'Fontsize',13)
text(1.1*time(xM),yM,['MAX=' num2str(yM)],'Fontsize',13)
```

```
% bending_analysis.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

clear all
close all
clc

g=9.81;
w=2.5; % 3 kg/m
q=w*g;
m_pay=3; % kg
m_mot=1; % kg
Fpay=g*m_pay;
Fm=g*m_mot;
L1=0.7;
L2=0.6;
L=L1+L2;

EI=1;

z=0:0.01:L;

%% link 1
% q/24*(6*L^2*z^2 4*L*z^3+z^4)
hold on
grid on
plot(z, q/24*(6*L^2*z.^2 4*L*z.^3+z.^4),'r')

%% link 2
% Fm/6*(3*L1*z^2 z^3)
% Fm/6*(3*L1^2*z L1^3)
z1=0:0.01:L1;
z2=L1:0.01:L;
plot(z1, Fm/6*(3*L1*z1.^2 z1.^3),'b')
plot(z2, Fm/6*(3*L1^2*z2 L1^3),'b')

%% link 3
% Fpay/6*(3*L*z^2 z^3)
plot(z, Fpay/6*(3*L*z.^2 z.^3),'k')
syms x
max_disp=5*10^3;
k=solve( q*L^4/(8*(x)) Fm*L1^2/(6*x)*(3*L L1) Fpay/(2*x)*L^3+max_disp);
E=7*10^10;
```

```
I=double(k/E)*10^(8); % in cm^4
I=75*10^(8); % in cm^4

%% plot displacement as a function of Ix
figure
grid on
hold on
x=0.5:0.01:20; % in mm
plot(x,(double((3*q*L^4 + 12*Fpay*L^3 + 12*Fm*L*L1^2   4*Fm*L1^3)./(24.*x.*10^ 3))./E)
    .*10^8)
title('Vertical displacement analysis as a function of beam inertia','fontsize',18)
xlabel('Tip vertical displacement [mm]','fontsize',10)
ylabel('Corresponding I_{x} moment of inertia of beam [cm^4]','fontsize',10)
```

```
% buckling_analysis.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

vmax=1; % mm
vm=vmax*10^ 3;

Pn=52.2;
M=470.18;
L=0.7;
E=70*10^9; % Pa

I=Pn/E*(2/L*acos((1+Pn/M*vm)^ 1))^(2);

I1=11*10^(8);
A=5.7*10^(4);
c=45*(10^ 3)*0.5;
sgy=165*10^6;

syms P
Pcr_x=pi^2*(E*I1)/(4*L^2);
Pcr_y=solve( sgy+P/A+(M/I1/(cos(sqrt(P/(E*I1))*L/2)))*c);
```

```
% Cylinder.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights to the scripts owner

function  Cylinder(X1,X2,r,n,cyl_color,closed,lines)
%
% This function constructs a cylinder connecting two center points
%
% Usage :
% [Cylinder EndPlate1 EndPlate2] = Cylinder(X1+20,X2,r,n,'r',closed,lines)
%
% Cylinder        Handle of the cylinder
% EndPlate1       Handle of the Starting End plate
% EndPlate2       Handle of the Ending End plate
% X1 and X2 are the 3x1 vectors of the two points
% r is the radius of the cylinder
% n is the no. of elements on the cylinder circumference (more > refined)
% cyl_color is the color definition like 'r','b',[0.52 0.52 0.52]
% closed=1 for closed cylinder or 0 for hollow open cylinder
% lines=1 for displaying the line segments on the cylinder 0 for only
% surface
%
% Typical Inputs
% X1=[10 10 10];
% X2=[35 20 40];
% r=1;
```

```
29 % n=20;
30 % cyl_color='b';
31 % closed=1;
32 %
33 % NOTE: There is a MATLAB function "cylinder" to revolve a curve about an
34 % axis. This "Cylinder" provides more customization like direction and etc
35
36
37 % Calculating the length of the cylinder
38 length_cyl=norm(X2 X1);
39
40 % Creating a circle in the YZ plane
41 t=linspace(0,2*pi,n)';
42 x2=r*cos(t);
43 x3=r*sin(t);
44
45 % Creating the points in the X Direction
46 x1=[0 length_cyl];
47
48 % Creating (Extruding) the cylinder points in the X Directions
49 xx1=repmat(x1,length(x2),1);
50 xx2=repmat(x2,1,2);
51 xx3=repmat(x3,1,2);
52
53 % Drawing two filled cirlces to close the cylinder
54 if closed==1
55     hold on
56     EndPlate1=fill3(xx1(:,1),xx2(:,1),xx3(:,1),'r');
57     EndPlate2=fill3(xx1(:,2),xx2(:,2),xx3(:,2),'r');
58 end
59
60 % Plotting the cylinder along the X Direction with required length starting
61 % from Origin
62 Cylinder=mesh(xx1,xx2,xx3);
63
64 % Defining Unit vector along the X direction
65 unit_Vx=[1 0 0];
66
67 % Calulating the angle between the x direction and the required direction
68 % of cylinder through dot product
69 angle_X1X2=acos( dot( unit_Vx,(X2 X1) )/( norm(unit_Vx)*norm(X2 X1)) )*180/pi;
70
71 % Finding the axis of rotation (single rotation) to roate the cylinder in
72 % X direction to the required arbitrary direction through cross product
73 axis_rot=cross([1 0 0],(X2 X1) );
74
75 % Rotating the plotted cylinder and the end plate circles to the required
76 % angles
77 if angle_X1X2~=0 % Rotation is not needed if required direction is along X
78     rotate(Cylinder,axis_rot,angle_X1X2,[0 0 0])
79     if closed==1
80         rotate(EndPlate1,axis_rot,angle_X1X2,[0 0 0])
81         rotate(EndPlate2,axis_rot,angle_X1X2,[0 0 0])
82     end
83 end
84
85 % Till now cylinder has only been aligned with the required direction, but
86 % position starts from the origin. so it will now be shifted to the right
87 % position
88 if closed==1
89     set(EndPlate1,'XData',get(EndPlate1,'XData')+X1(1))
90     set(EndPlate1,'YData',get(EndPlate1,'YData')+X1(2))
91     set(EndPlate1,'ZData',get(EndPlate1,'ZData')+X1(3))
92
93     set(EndPlate2,'XData',get(EndPlate2,'XData')+X1(1))
94     set(EndPlate2,'YData',get(EndPlate2,'YData')+X1(2))
95     set(EndPlate2,'ZData',get(EndPlate2,'ZData')+X1(3))
96 end
97 set(Cylinder,'XData',get(Cylinder,'XData')+X1(1))
98 set(Cylinder,'YData',get(Cylinder,'YData')+X1(2))
```

```
set(Cylinder,'ZData',get(Cylinder,'ZData')+X1(3))

% Setting the color to the cylinder and the end plates
set(Cylinder,'FaceColor',cyl_color)
if closed==1
    set([EndPlate1 EndPlate2],'FaceColor',cyl_color)
else
    EndPlate1=[];
    EndPlate2=[];
end

% If lines are not needed making it disapear
if lines==0
    set(Cylinder,'EdgeAlpha',0)
end
```

```
% direct_kinematics.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

clc
clear all
close all

l1=0.664;
l2=0.720;
l3=0.625;
d1=0.0;
d2=0.0;
d3=0.0;
d4=0.160;
th =[0 pi/6  pi/3 0 0 0];

alp=[0 pi/2 0  pi/2  pi/2  pi/2];
a=[0 0 l2 l3 0 0];
d=[0  d2  d3  d4 0 0];

[T01 T12 T23 T34 T45 T56]=rot_matrix_23J(a, th, alp, d); % gets the rotational matrixes

disframe3(T01,0.3);
disframe3(T01*T12,0.3);
disframe3(T01*T12*T23,0.3);
disframe3(T01*T12*T23*T34,0.3);
disframe3(T01*T12*T23*T34*T45,0.1);
disframe3(T01*T12*T23*T34*T45*T56,0.61);

Or=[0 0 0  1]';

hold on
grid on
axis equal
xlabel('x [m]')
ylabel('y [m]')
zlabel('z [m]')

P(:,2)=T01*Or; % origin
plot3(P(1,2),P(2,2),P(3,2),'o')
plot3([0 P(1,2)],[0 P(2,2)],[0 P(3,2)])

P(:,3)=T01*T12*Or; % origin
P3=T01*T12*[l2 0 0 1]';
P4=T01*T12*[l2 0  d3 1]';
plot3(P(1,3),P(2,3),P(3,3),'o')
plot3([P3(1) P(1,3)],[P3(2) P(2,3)],[P3(3) P(3,3)])

P(:,4)=T01*T12*T23*Or; % origin
plot3(P(1,4),P(2,4),P(3,4),'o')
```

```
P5=T01*T12*T23*[l3 0 0 1]';
plot3([P(1,4) P3(1)],[P(2,4) P3(2)],[P(3,4) P3(3)])
plot3([P(1,4) P5(1)],[P(2,4) P5(2)],[P(3,4) P5(3)])

P(:,6)=T01*T12*T23*T34*Or; % origin
plot3(P(1,6),P(2,6),P(3,6),'o')
plot3([P(1,6) P5(1)],[P(2,6) P5(2)],[P(3,6) P5(3)])


% actuators
Dz=[0 0 0.02 1]'; % actuators height
mDz=[0 0 0.02 1]'; % actuators height

Pz(:,3)=T01*T12*Dz; % origin
Pzz(:,3)=T01*T12*(mDz); % origin

Pz(:,4)=T01*T12*T23*Dz; % origin
Pzz(:,4)=T01*T12*T23*(mDz); % origin

Pz(:,5)=T01*T12*T23*T34*Dz; % origin
Pzz(:,5)=T01*T12*T23*T34*(mDz); % origin

% motors
Cylinder(P(1:3,1)+Dz(1:3) [0 0 l1]',P(1:3,1) Dz(1:3) [0 0 l1]',0.03,30,'r',1,0)
Cylinder(Pz(1:3,3),Pzz(1:3,3),0.03,30,'r',1,0)
Cylinder(Pz(1:3,4),Pzz(1:3,4),0.03,30,'r',1,0)
Cylinder(Pz(1:3,5),Pzz(1:3,5),0.03,30,'r',1,0)

% links
Cylinder(P(1:3,2),[0 0 l1]',0.01,30,[0.52 0.52 0.52],1,0)
Cylinder(P(1:3,2),P(1:3,3),0.01,30,[0.52 0.52 0.52],1,0)
Cylinder(P(1:3,3),P3(1:3),0.01,30,[0.52 0.52 0.52],1,0)
Cylinder(P3(1:3),P(1:3,4),0.01,30,[0.52 0.52 0.52],1,0)
Cylinder(P(1:3,4),P5(1:3),0.01,30,[0.52 0.52 0.52],1,0)
Cylinder(P(1:3,6),P5(1:3),0.01,30,[0.52 0.52 0.52],1,0)
disframe3(T01,0.3);
disframe3(T01*T12,0.3);
disframe3(T01*T12*T23,0.3);
disframe3(T01*T12*T23*T34,0.3);

view(30,30)
```

```
% disframe3.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

function [x1,y1,z1]=disframe3(m, L)

x=L;
y=L;
z=L;
x1=m*[x,0,0,0]';
y1=m*[0,y,0,0]';
z1=m*[0,0,z,0]';
X1=m(1,4);
Y1=m(2,4);
Z1=m(3,4);

h=line([X1 X1+x1(1,1)], [Y1 Y1+x1(2,1)], [Z1 Z1+x1(3,1)]);
k=line([X1 X1+y1(1,1)], [Y1 Y1+y1(2,1)], [Z1 Z1+y1(3,1)]);
l=line([X1 X1+z1(1,1)], [Y1 Y1+z1(2,1)], [Z1 Z1+z1(3,1)]);

set(h,'LineWidth',1.5)
set(k,'LineWidth',1.5)
set(l,'LineWidth',1.5)
set(h,'Marker','o')
set(k,'Marker','o')
```

```
set(l,'Marker','o')
set(h,'Markersize',2)
set(k,'Markersize',2)
set(l,'Markersize',2)
set(h,'Color','r')
set(k,'Color','g')
```

```
% geom.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

function [I,m,com]=geom

% geometric information about the arm

I(:,:,1)=[0.128911203050000     4.95069000000000e 06     8.77328200000000e 05;
4.95069000000000e 06     0.129902859470000         0.00104133814000000;
 8.77328200000000e 05     0.00104133814000000     0.00262285717000000];

I(:,:,2)=[0.00307791577000000     9.56111000000000e 05     0.00113495196000000;
 9.56111000000000e 05     0.153927371740000         4.98120000000000e 06;
0.00113495196000000       4.98120000000000e 06     0.154686818120000];

I(:,:,3)=[0.00276059347000000     0.00375355660000000     2.86400000000000e 08;
 0.00375355660000000     0.0997111819000000        1.43000000000000e 09;
 2.86400000000000e 08     1.43000000000000e 09     0.100361734600000];

I(:,:,4)=[0.000739954430000000   7.60000000000000e 10     2.09000000000000e 09;
 7.60000000000000e 10     0.000433548980000000     0.000310060540000000;
 2.09000000000000e 09     0.000310060540000000     0.000373994180000000];

I(:,:,5)=[0.000426889310000000   6.00000000000000e 11     7.90000000000000e 10;
6.00000000000000e 11     0.000136362730000000     0.000142111750000000;
 7.90000000000000e 10     0.000142111750000000     0.000348042310000000];

I(:,:,6)=[9.71437000000000e 06   0         0;
0         9.71437000000000e 06     0;
0         0         2.11854000000000e 06];


com1=[ 0.000130000000000000       0.00154000000000000     0.300080000000000]';
com2=[0.392940000000000  0.000120000000000000    0.00143000000000000]';
com3=[0.330040000000000  0.00630000000000000     0]';
com4=[0  0.0817700000000000       0.0608200000000000]';
com5=[0  0.0501700000000000       0.0557100000000000]';
com6=[0  0         0.105250000000000]';

com=[com1 com2 com3 com4 com5 com6]; % center of mass coordinates in meters

m=[2.259340
    2.424060
    2.076640
    0.182860
    0.158690
    0.041390];
```

```
% inv_kin.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

function [THf]=inv_kin(x,y,z,d,alp,a)

l2=a(3);
```

```
10 l3=a(4);
11 d2=d(2);
12 d3=d(3);
13 d4=d(4);
14 if sqrt(x^2+y^2+z^2)>l2+l3
15 disp('target out of reach')
16 return
17 end
18 th=zeros(8,3);
19
20 % for TH3
21 % d2^2 + 2*d2*d3 + d3^2 + d4^2 + 2*sin(th3)*d4*l2 + l2^2 + 2*cos(th3)*l2*l3 + l3^2
22
23 A=2*l2*d4;
24 B=2*l2*l3;
25 C=(x^2+y^2+z^2) (d2+d3)^2 (d4^2+l2^2+l3^2);
26 th3_1=atan2(B,A)+asin(C/(sqrt(A^2+B^2)));
27 th3_2=atan2(B,A)+pi asin(C/(sqrt(A^2+B^2)));
28 th(1:4,3)=th3_1;
29 th(5:8,3)=th3_2;
30
31 % for TH2
32 % l2*sin(th2)   d4*cos(th2)*cos(th3) + l3*cos(th2)*sin(th3) + l3*cos(th3)*sin(th2) + d4*
       sin(th2)*sin(th3)
33 % (l2+l3*cos(th3)+d4*sin(th3))*sin(th2) + (l3*sin(th3)) d4*cos(th3))*cos(th2) = C
34
35 for i=1:4:8
36     A=l2+l3*cos(th(i,3))+d4*sin(th(i,3));
37     B=l3*sin(th(i,3)) d4*cos(th(i,3));
38     C=z;
39     if abs(C/(sqrt(A^2+B^2)))<1
40         th(i:i+1,2)=atan2(B,A)+asin(C/(sqrt(A^2+B^2)));
41         th(2+(i:i+1),2)=atan2(B,A)+pi asin(C/(sqrt(A^2+B^2)));
42     else
43         th(i:i+1,2)=1111;
44         th(2+(i:i+1),2)=1111;
45     end
46 end
47
48 % for TH1
49 % x= l2*cos(th1)*cos(th2)   d3*sin(th1)   d2*sin(th1) + l3*cos(th1)*cos(th2)*cos(th3) +
       d4*cos(th1)*cos(th2)*sin(th3) + d4*cos(th1)*cos(th3)*sin(th2)   l3*cos(th1)*sin(th2)
       *sin(th3)
50 % x= ( d3 d2)*sin(th1) + (l2*cos(th2) + l3*cos(th2)*cos(th3) + d4*cos(th2)*sin(th3) + d4
       *cos(th3)*sin(th2)   l3*sin(th2)*sin(th3))*cos(th1)
51
52 for i=1:2:8
53     A= d3 d2;
54     B=l2*cos(th(i,2)) + l3*cos(th(i,2))*cos(th(i,3)) + d4*cos(th(i,2))*sin(th(i,3)) + d4
           *cos(th(i,3))*sin(th(i,2))   l3*sin(th(i,2))*sin(th(i,3));
55     C=x;
56     if abs(C/(sqrt(A^2+B^2)))<1
57         th(i,1)=atan2(B,A)+asin(C/(sqrt(A^2+B^2)));
58         th(i+1,1)=atan2(B,A)+pi asin(C/(sqrt(A^2+B^2)));
59     else
60         th(i,1)=1111;
61         th(i+1,1)=1111;
62     end
63 end
64
65 for i=1:8;
66
67     [T01 T12 T23 T34]=rot_matrix(a, [th(i,:) 0 0 0], alp, d); % gets the rotational
           matrixes
68
69     Or=[0 0 0   1]';
70     P_check=T01*T12*T23*T34*Or;
71     if P_check(2)*y<0
72         continue
73     else
```

```
74          THf(i,:)=th(i,:);
75      end
76  end
77
78  THf(all(THf==0,2),:)=[];
```

```
1   % jaco.m
2   %
3   % Andrea Antonello   antonela@uci.edu
4   % Master's thesis
5   % Universtity of Padova   All rights reserved
6
7   function [Jinv] = jaco(th)
8
9   global l1 l2 l3 d1 d2 d3 d4
10
11  t1=th(1);
12  t2=th(2);
13  t3=th(3);
14  t4=th(4);
15  t5=th(5);
16  t6=th(6);
17  Jinv=[see appendix for jacobian expression]
18
19  end
```

```
1   % jaco_sym.m
2   %
3   % Andrea Antonello   antonela@uci.edu
4   % Master's thesis
5   % Universtity of Padova   All rights reserved
6
7   clc
8   clear all
9   close all
10  syms l1 l2 l3 l4 l5 d1 d2 d3 d4 d5 t1 t2 t3 t4 t5 t6
11
12  th=[t1 t2 t3 t4 t5 t6];
13  a=[0 0 l2 l3 0 0];
14  d=[0  d2  d3  d4 0 0];
15  [T01 T12 T23 T34 T45 T56]=rot_matrix_sym(a, th, 1, d);
16
17  R01=T01(1:3,1:3);
18  R12=T12(1:3,1:3);
19  R23=T23(1:3,1:3);
20  R34=T34(1:3,1:3);
21  R45=T45(1:3,1:3);
22  R56=T56(1:3,1:3);
23
24  z0=[0 0 1]';
25  z1=R01*z0;
26  z2=R01*R12*z0;
27  z3=R01*R12*R23*z0;
28  z4=R01*R12*R23*R34*z0;
29  z5=R01*R12*R23*R34*R45*z0;
30  z6=R01*R12*R23*R34*R45*R56*z0;
31
32  p0=[0 0 0 1]';
33  pe=T01*T12*T23*T34*T45*T56*p0;
34  p1=T01*p0;
35  p2=T01*T12*p0;
36  p3=T01*T12*T23*p0;
37  p4=T01*T12*T23*T34*p0;
38  p5=T01*T12*T23*T34*T45*p0;
39  p6=T01*T12*T23*T34*T45*T56*p0;
40
41  pe=pe(1:3);
```

```
42 p0=p0(1:3);
43 p1=p1(1:3);
44 p2=p2(1:3);
45 p3=p3(1:3);
46 p4=p4(1:3);
47 p5=p5(1:3);
48 p6=p6(1:3);
49
50
51 J1=[cross(z1,pe p1) cross(z2,pe p2) cross(z3,pe p3) cross(z4,pe p4) cross(z5,pe p5)
       cross(z6,pe p6)];
52 J2=[z1 z2 z3 z4 z5 z6];
53 J=[J1; J2];
54 Jinv=inv(J);
```

```
1  % rot_matrix.m
2  %
3  % Andrea Antonello   antonela@uci.edu
4  % Master's thesis
5  % Universtity of Padova   All rights reserved
6
7  function [T01 T12 T23 T34 T45 T56]=rot_matrix(a, th, alp, d)
8
9  th(5)=th(5) pi/2;
10
11 for i=1:6
12     T(:,:,i)=[cos(th(i))  sin(th(i)) 0 a(i)
13         sin(th(i))*cos(alp(i)) cos(th(i))*cos(alp(i))  sin(alp(i))  sin(alp(i))*d(i)
14         sin(th(i))*sin(alp(i)) cos(th(i))*sin(alp(i)) cos(alp(i)) cos(alp(i))*d(i)
15         0       0       0       1];
16 end
17
18 T01=T(:,:,1);
19 T12=T(:,:,2);
20 T23=T(:,:,3);
21 T34=T(:,:,4);
22 T45=T(:,:,5);
23 T56=T(:,:,6);
24 end
```

```
1  % rot_matrix_sym.m
2  %
3  % Andrea Antonello   antonela@uci.edu
4  % Master's thesis
5  % Universtity of Padova   All rights reserved
6
7  function [T01 T12 T23 T34 T45 T56]=rot_matrix_sym(a, th, alp, d)
8
9  alp=[0 sym('pi')/2 0  sym('pi')/2  sym('pi')/2  sym('pi')/2];
10 th(5)=th(5) pi/2;
11
12 for i=1:6
13     T(:,:,i)=[cos(th(i))  sin(th(i)) 0 a(i)
14         sin(th(i))*cos(alp(i)) cos(th(i))*cos(alp(i))  sin(alp(i))  sin(alp(i))*d(i)
15         sin(th(i))*sin(alp(i)) cos(th(i))*sin(alp(i)) cos(alp(i)) cos(alp(i))*d(i)
16         0       0       0       1];
17 end
18
19 T01=T(:,:,1);
20 T12=T(:,:,2);
21 T23=T(:,:,3);
22 T34=T(:,:,4);
23 T45=T(:,:,5);
24 T56=T(:,:,6);
25
26 end
```

```
% rpy_wrist.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

function [th4 th5 th6]=rpy_wrist(T)

% angles in degrees
% roll, pitch, yaw angles wrt fixed frame

% ROLL: rot about x
% PITCH: rot about y
% YAW: rot about z

% T34 =
% [  cos(th4),  sin(th4), 0,   2/5]
% [         0,         0, 1,  1/10]
% [  sin(th4),  cos(th4), 0,     0]
% [         0,         0, 0,     1]
%
% T45 =
% [  cos(th5   pi/2),  sin(th5   pi/2), 0, 0]
% [                0,                0, 1, 0]
% [  sin(th5   pi/2),  cos(th5   pi/2), 0, 0]
% [                0,                0, 0, 1]
%
% T56 =
% [  cos(th6),  sin(th6), 0, 0]
% [         0,         0, 1, 0]
% [  sin(th6),  cos(th6), 0, 0]
% [         0,         0, 0, 1]
%

ans=[acos( T(2,3))+pi/2;
    2*pi acos( T(2,3))+pi/2];

Y(1:4,2)=(ans(1));
Y(5:8,2)=(ans(2));

ans1=[acos( T(2,1)/sin(Y(1,2)  pi/2));
    2*pi acos( T(2,1)/sin(Y(1,2)  pi/2))];
ans2=[acos( T(2,1)/sin(Y(5,2)  pi/2));
    2*pi acos( T(2,1)/sin(Y(5,2)  pi/2))];

Y(:,3)=[ans1; ans1; ans2; ans2];

ans3=[acos( T(1,3)/sin(Y(1,2)  pi/2));
    2*pi acos( T(1,3)/sin(Y(1,2)  pi/2))];
ans4=[acos( T(1,3)/sin(Y(5,2)  pi/2));
    2*pi acos( T(1,3)/sin(Y(5,2)  pi/2))];

Y(:,1)=[ans3(1); ans3(1); ans3(2); ans3(2); ans4(1); ans4(1); ans4(2); ans4(2);];


for i=1:8
    if sin(Y(i,1))*sin(Y(i,2)   pi/2)*T(3,3)>0 && sin(Y(i,3))*sin(Y(i,2)   pi/2)*T(2,2)
        >0
    thh(i,:)=Y(i,:);
    end
end

thh(all(thh==0,2),:)=[];

for i=1:size(thh,1)
    for j=1:3
    if thh(i,j)>2*pi
        thh(i,j)=thh(i,j) 2*pi;
    end
```

```
    end
end

% choice of solution
if abs(thh(1))<abs(thh(2))
    thh=thh(1,:);
else
    thh=thh(2,:);
end


th4=thh(1);
th5=thh(2);
th6=thh(3);
```

```
% sim_starter.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

function [th]=sim_starter(x_or)

global l1 l2 l3 d1 d2 d3 d4 d a alp att Tatt

[THf]=inv_kin2(x_or(1),x_or(2),x_or(3),d,alp,a); % Inverse kinematic: yields th1, th2,
    th3
ind=1;
th1=double(THf(ind,1));
th2=double(THf(ind,2));
th3=double(THf(ind,3));
th=[th1,th2,th3,0,0,0];

[T01 T12 T23 T34]=rot_matrix_23J(a, th, alp, d); % gets the rotational matrixes
T03=T01*T12*T23;
T04=T01*T12*T23*T34;
T06=eye(4);
T06(1:3,1:3)=[rpy(att(1),att(2),att(3))];
Tatt=T06;
T06(:,4)=T34(:,4);
T36=T03'*T06;
[th4 th5 th6]=rpy_wrist_23J(T36);
th=[th1 th2 th3 th4 th5 th6];
```

```
% static_analysis.m
%
% Andrea Antonello   antonela@uci.edu
% Master's thesis
% Universtity of Padova   All rights reserved

% start
clc
clear all
figure
global l1 l2 l3
% parameters definition
g=9.8062; % gravity
n=100;
l1=0.7; % length of arm 1 [m]
l2=0.7; % length of arm 2 [m]
bl2=0.0; % back length of arm 2 [m]
l3=0.4; % length of arm 3 [m]
a1=0; % first angle, we suppose first arm perpendicular to ground
a2v=180:360/n:180; % vector of angles
a3v=180:360/n:180; % vector of angles

% material properties
```

```
rho1=2.44; % profile density [kg/m]
rho2=2.44; % profile density [kg/m]
rho3=2.44; % profile density [kg/m]
m1=l1*rho1; % mass of arm 1 [kg]
m2=l2*rho2; % mass of arm 2 [kg]
bm2=abs(bl2)*rho2; % mass of back arm 2 [kg]
m3=l3*rho3; % mass of arm 3 [kg]
P=0.7; % payload [kg]
Pm2=0.2; % motor weight [kg]
Pm3=0.2; % motor weight [kg]
Pcw=0; % counterweight [kg]
Fax=(m1+m2+m3+P+Pm2+Pm3)*g/(10^3);

% joints static torque calculation
for i=1:n
    a2=a2v(i);
    for j=1:n
        a3=a3v(j);
        x(i,j)=a2;
        y(i,j)=a3;
        M3(i,j)=(P+0.5*m3)*g*l3^2*cosd(a2+a3+a1);
        M2(i,j)=m2*g*0.5*l2^2*cosd(a2+a1)+Pm3*g*l2^2*cosd(a2+a1)
        (bm2*g*0.5*bl2*cosd(a2+a1)) (Pcw*g*bl2*cosd(a2+a1))+
        (l2*cosd(a2+a1)+0.5*l3^2*cosd(a2+a3+a1))*m3*g+(l2^2*cosd(a2+a1)+l3*cosd(a2+a3+a1
            ))*P*g;
        z2(i,j)=M2(i,j);
        z3(i,j)=M3(i,j);
    end
end

M2_max=(max(max(M2))); % max value of torque @ joint 2
M3_max=(max(max(M3))); % max value of torque @ joint 3

% command window print
fprintf('LENGTHS \n')
disp(['length arm 1 = ' num2str(l1) ' m']);
disp(['length arm 2 = ' num2str(l2) ' m']);
disp(['length arm 3 = ' num2str(l3) ' m']);

fprintf('\nMASSES \n')
disp(['mass arm 1 = ' num2str(m1) ' kg']);
disp(['mass arm 2 = ' num2str(m2) ' kg']);
disp(['mass arm 3 = ' num2str(m3) ' kg']);
disp(['mass payload (wrist) = ' num2str(P) ' kg']);
disp(['mass motor 2 = ' num2str(Pm2) ' kg']);
disp(['mass motor 3 = ' num2str(Pm3) ' kg']);

fprintf('\nTORQUES \n')
disp(['Max torque @ joint 2 = ' num2str(M2_max) ' Nm']);
disp(['Max torque @ joint 3 = ' num2str(M3_max) ' Nm']);


% static deflection, max load case
syms x2 x3
E=69*10^9; % alluminium [Pa]
I2=22.9*10^(8); % Ixx
I3=11.54*10^(8); % Ixx
y2=Pm3*g*x2^2/(6*E*I2)*(3*l2 x2)+(m3*g*(l2+l3/2)
+P*g*(l2+l3))*x2^2/(2*E*I2)+rho2*g*x2^2/(24*E*I2)*(x2^2+6*l2^2 4*l2*x2);
y3=rho3*g*x3^2/(24*E*I3)*(x3^2+6*l3^2 4*l3*x3)+P*g*x3^2/(6*E*I3)*(3*l3 x3);

figure
grid on
hold on
m=100;
xx2=0:l2/m:l2;
xx3=0:l3/m:l3;

defl1=[xx2; subs(y2, 'x2', xx2)];
defl2=[xx3; subs(y3, 'x3', xx3)];
```

```
93 plot(defl1(1,:),defl1(2,:)*10^3)
94 plot(defl2(1,:)+defl1(1,m+1),(defl2(2,:)+defl1(2,m+1))*10^3)
95 plot(defl1(1,m+1),defl1(2,m+1),'o')
96 xlabel('Length [m]')
97 ylabel('Deflection [mm]')
```

Appendix B

Jacobian expression

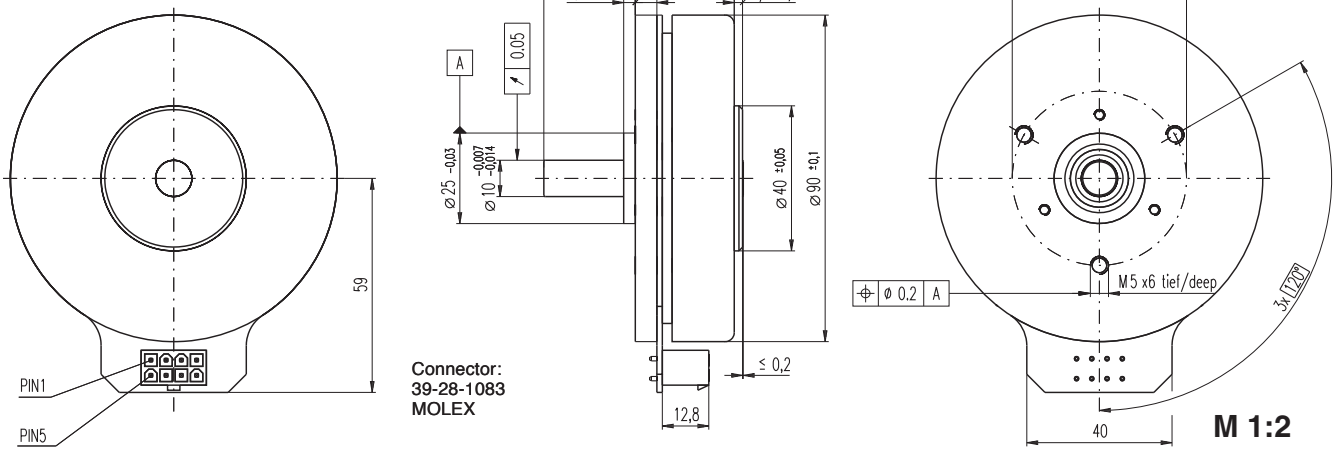
We report in this section the analytical expression of the Jacobian matrix for reference purposes.

```
1
2  [ l3*(sin(t1)*sin(t2)*sin(t3)   cos(t2)*cos(t3)*sin(t1))   d4*(cos(t2)*sin(t1)*sin(t3) +
       cos(t3)*sin(t1)*sin(t2))   d2*cos(t1)   d3*cos(t1)   l2*cos(t2)*sin(t1),

       cos(t1)*(l3*(cos(t2)*sin(t3) + cos(t3)*sin(t2))   d4*(cos(t2)*cos(t3)   sin(t2)*sin
      (t3)) + l2*sin(t2)), cos(t1)*(d4*(cos(t2)*cos(t3)   sin(t2)*sin(t3))   l3*(cos(t2)*
      sin(t3) + cos(t3)*sin(t2))), 0, 0, 0]
3
4
5  [ d4*(cos(t1)*cos(t2)*sin(t3) + cos(t1)*cos(t3)*sin(t2))   l3*(cos(t1)*sin(t2)*sin(t3)
       cos(t1)*cos(t2)*cos(t3))   d2*sin(t1)   d3*sin(t1) + l2*cos(t1)*cos(t2),

       sin(t1)*(l3*(cos(t2)*sin(t3) + cos(t3)*sin(t2))   d4*(cos(t2)*cos(t3)   sin(t2)*sin
      (t3)) + l2*sin(t2)), sin(t1)*(d4*(cos(t2)*cos(t3)   sin(t2)*sin(t3))   l3*(cos(t2)*
      sin(t3) + cos(t3)*sin(t2))), 0, 0, 0]
6
7
8  [0, sin(t1)*(d4*(cos(t2)*sin(t1)*sin(t3) + cos(t3)*sin(t1)*sin(t2))   l3 *(sin(t1)*sin(
      t2)*sin(t3)   cos(t2)*cos(t3)*sin(t1)) + d3*cos(t1) + l2*cos(t2)*sin(t1)) + cos(t1)
      *(d4*(cos(t1)*cos(t2)*sin(t3) + cos(t1)*cos(t3)*sin(t2))   l3*(cos(t1)*sin(t2)*sin(
      t3)   cos(t1)*cos(t2)*cos(t3))   d3*sin(t1) + l2*cos(t1)*cos(t2)), sin(t1)*(d4*(cos
      (t2)*sin(t1)*sin(t3) + cos(t3)*sin(t1)*sin(t2))   l3*(sin(t1)*sin(t2)*sin(t3)   cos(
      t2)*cos(t3)*sin(t1))) + cos(t1)*(d4*(cos(t1)*cos(t2)*sin(t3) + cos(t1)*cos(t3)*sin(
      t2))   l3*(cos(t1)*sin(t2)*sin(t3))   (cos(t1)*cos(t2)*cos(t3))), 0, 0, 0]
9
10
11 [0, sin(t1), sin(t1),   cos(t1)*cos(t2)*sin(t3)   cos(t1)*cos(t3)*sin(t2), sin(t4)*(cos(
      t1)*sin(t2)*sin(t3)   cos(t1)*cos(t2)*cos(t3))   cos(t4)*sin(t1), sin(t5   pi/2)*(
      sin(t1)*sin(t4) + cos(t4)*(cos(t1)*sin(t2)*sin(t3)   cos(t1)*cos(t2)*cos(t3))) + cos
      (t5   pi/2)*(cos(t1)*cos(t2)*sin(t3) + cos(t1)*cos(t3)*sin(t2))]
12
13
14 [0,  cos(t1),  cos(t1),   cos(t2)*sin(t1)*sin(t3)   cos(t3)*sin(t1)*sin(t2), cos(t1)*cos
      (t4) + sin(t4)*(sin(t1)*sin(t2)*sin(t3)   cos(t2)*cos(t3)*sin(t1)), cos(t5   pi/2)*(
      cos(t2)*sin(t1)*sin(t3) + cos(t3)*sin(t1)*sin(t2))   sin(t5   pi/2)*(cos(t1)*sin(t4)
         cos(t4)*(sin(t1)*sin(t2)*sin(t3)   cos(t2)*cos(t3)*sin(t1)))]
15
16
17 [ 1, 0, 0; cos(t2)*cos(t3)   sin(t2)*sin(t3), sin(t4)*(cos(t2)*sin(t3) + cos(t3)*sin(t2
      )),   cos(t5   pi/2)*(cos(t2)*cos(t3)   sin(t2)*sin(t3))   cos(t4)*sin(t5   pi/2)*(
      cos(t2)*sin(t3) + cos(t3)*sin(t2))]
```

Appendix C

Datasheets

EC 90 flat ∅90 mm, brushless, 90 Watt



Stock program
Standard program
Special program (on request)

			Order Number	
	with Hall sensors		323772	244879
Motor Data				
	Values at nominal voltage			
1	Nominal voltage	V	24.0	48.0
2	No load speed	rpm	3190	2080
3	No load current	mA	539	130
4	Nominal speed	rpm	2650	1640
5	Nominal torque (max. continuous torque)	mNm	387	494
6	Nominal current (max. continuous current)	A	5.39	2.12
7	Stall torque	mNm	4670	4530
8	Starting current	A	66.2	20.9
9	Max. efficiency	%	83	85
	Characteristics			
10	Terminal resistance phase to phase	Ω	0.363	2.30
11	Terminal inductance phase to phase	mH	0.264	2.50
12	Torque constant	mNm / A	70.5	217
13	Speed constant	rpm / V	135	44.0
14	Speed / torque gradient	rpm / mNm	0.697	0.466
15	Mechanical time constant	ms	22.3	14.9
16	Rotor inertia	gcm²	3060	3060

Specifications

Thermal data

17	Thermal resistance housing-ambient	1.89 K / W
18	Thermal resistance winding-housing	2.99 K / W
19	Thermal time constant winding	52.6 s
20	Thermal time constant motor	281 s
21	Ambient temperature	-40 ... +100°C
22	Max. permissible winding temperature	+125°C

Mechanical data (preloaded ball bearings)

23	Max. permissible speed	5000 rpm
24	Axial play at axial load < 15 N	0 mm
	> 15 N	0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	12 N
27	Max. force for press fits (static)	150 N
	(static, shaft supported)	8000 N
28	Max. radial loading, 7.5 mm from flange	30 N

Other specifications

29	Number of pole pairs	12
30	Number of phases	3
31	Weight of motor	648 g

Values listed in the table are nominal.

Connection

Pin 1	Hall sensor 1
Pin 2	Hall sensor 2
Pin 3	4.5 ... 24 VDC
Pin 4	Motor winding 3
Pin 5	Hall sensor 3
Pin 6	GND
Pin 7	Motor winding 1
Pin 8	Motor winding 2

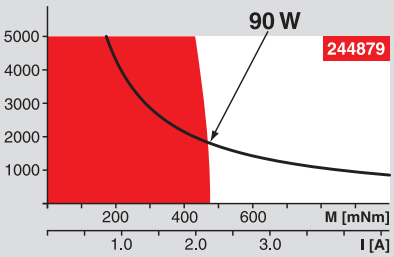
Wiring diagram for Hall sensors see page 29

Cable

Connection cable Universal, L = 500 mm	**339380**
Connection cable zu EPOS, L = 500 mm	**354045**

Operating Range

n [rpm]



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

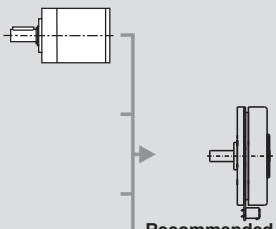
Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅52 mm
4 - 30 Nm
Page 244

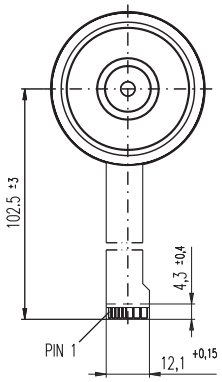


Recommended Electronics:

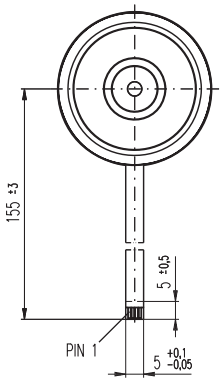
DECS 50/5	Page 288
DEC 50/5	289
DECV 50/5	295
DEC 70/10	295
EPOS 24/5	303
EPOS 70/10	303
EPOS P 24/5	306
Notes	**20**

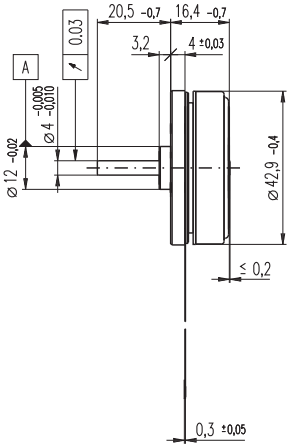
EC 45 flat ∅45 mm, brushless, 30 Watt

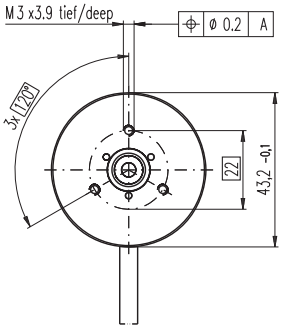
A with hall sensors



B sensorless







M 1:2

Stock program
Standard program
Special program (on request)

			Order Number							
	A with Hall sensors		200142		339281		339282			
	B sensorless			200189		339283		339284		
Motor Data										
	Values at nominal voltage									
1	Nominal voltage	V	12.0	12.0	24.0	24.0	36.0	36.0		
2	No load speed	rpm	4370	4360	4370	4370	4760	4760		
3	No load current	mA	151	150	75.3	75.2	56.9	56.9		
4	Nominal speed	rpm	2860	2820	2850	2840	3210	3210		
5	Nominal torque (max. continuous torque)	mNm	59.0	54.3	58.8	57.5	70.6	69.5		
6	Nominal current (max. continuous current)	A	2.14	2.00	1.07	1.05	0.893	0.882		
7	Stall torque	mNm	255	219	253	243	380	369		
8	Starting current	A	10.0	8.57	4.96	4.77	5.38	5.22		
9	Max. efficiency	%	77	76	77	77	81	81		
	Characteristics									
10	Terminal resistance phase to phase	Ω	1.20	1.40	4.84	5.04	6.70	6.9		
11	Terminal inductance phase to phase	mH	0.560	0.560	2.24	2.24	4.29	4.29		
12	Torque constant	mNm / A	25.5	25.5	51.0	51.0	70.6	70.6		
13	Speed constant	rpm / V	374	374	187	187	135	135		
14	Speed / torque gradient	rpm / mNm	17.6	20.6	17.8	18.5	12.8	13.2		
15	Mechanical time constant	ms	17.1	19.9	17.2	17.9	12.4	12.8		
16	Rotor inertia	gcm²	92.5	92.5	92.5	92.5	92.5	92.5		

Specifications

	Thermal data		
17	Thermal resistance housing-ambient		4.23 K / W
18	Thermal resistance winding-housing		4.57 K / W
19	Thermal time constant winding		13.2 s
20	Thermal time constant motor		186 s
21	Ambient temperature		-40 ... +100°C
22	Max. permissible winding temperature		+125°C
	Mechanical data (preloaded ball bearings)		
23	Max. permissible speed		10000 rpm
24	Axial play at axial load	< 5.0 N	0 mm
		> 5.0 N	typ. 0.14 mm
25	Radial play		preloaded
26	Max. axial load (dynamic)		4.8 N
27	Max. force for press fits (static)		50 N
	(static, shaft supported)		1000 N
28	Max. radial loading, 7.5 mm from flange		5.5 N
	Other specifications		
29	Number of pole pairs		8
30	Number of phases		3
31	Weight of motor		88 g

Values listed in the table are nominal.

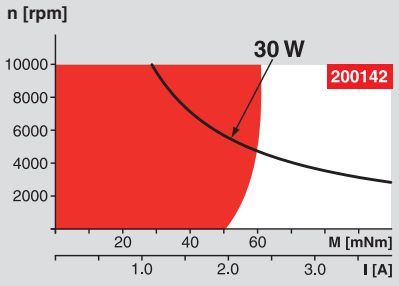
Connection	**with hall sensors**	**sensorless**
Pin 1	4.5 ... 18 VDC	Motor winding 1
Pin 2	Hall sensor 3*	Motor winding 2
Pin 3	Hall sensor 1*	Motor winding 3
Pin 4	Hall sensor 2*	neutral point
Pin 5	GND	
Pin 6	Motor winding 3	
Pin 7	Motor winding 2	
Pin 8	Motor winding 1	

*internal pull-up (7 … 13 kΩ) on pin 1
Wiring diagram for Hall sensors see page 29

Adapter	**Order Number**	**Order Number**
see p. 308	220300	220310
Connector	**Article number**	**Article number**
AMP	1-487951-1	487951-4
MOLEX	52207-1190	52207-0490
MOLEX	52089-1110	52089-0410

Pin for design with Hall sensors:
FPC, 11 pole, pitch 1.0 mm, top contact style

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

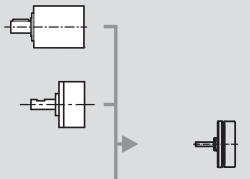
Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 240

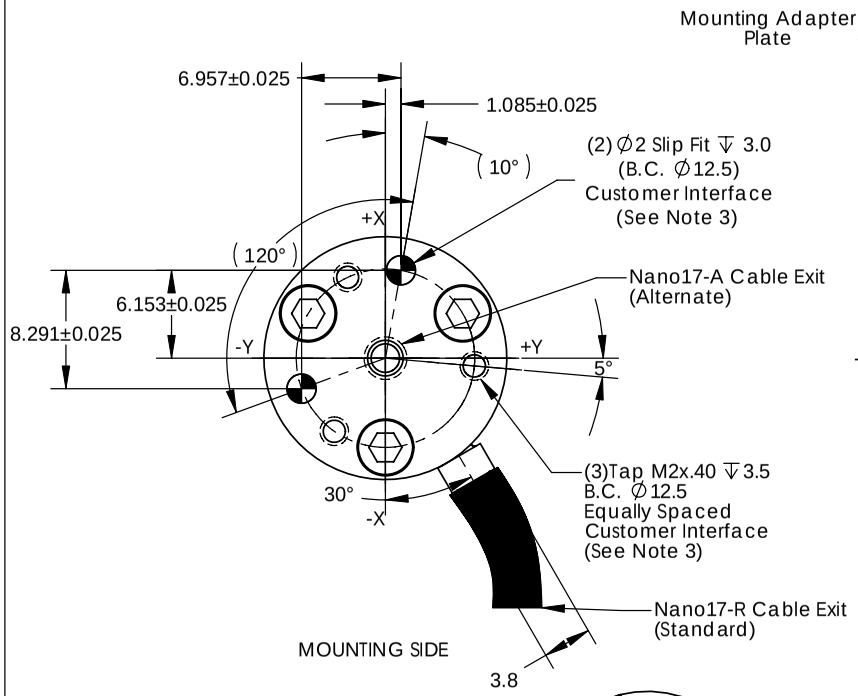
Spur Gearhead
∅45 mm
0.5 - 2.0 Nm
Page 242



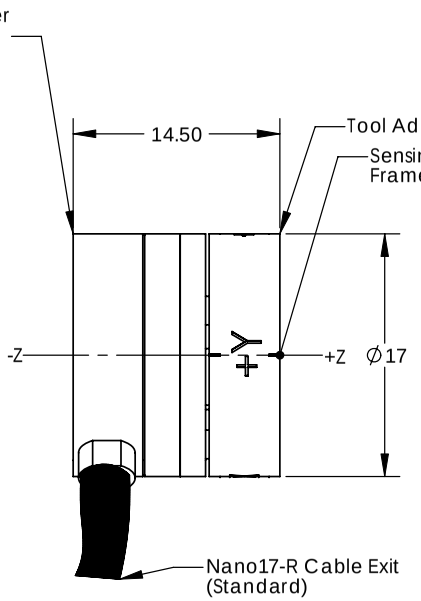
Recommended Electronics:

DECS 50/5	Page 288
DEC 24/3	289
DEC 50/5	289
DEC Module 24/2	289
DECV 50/5	295
EPOS2 Module 36/2	302
EPOS 24/1	302
EPOS2 24/5	303
EPOS P 24/5	306
Notes	**20**

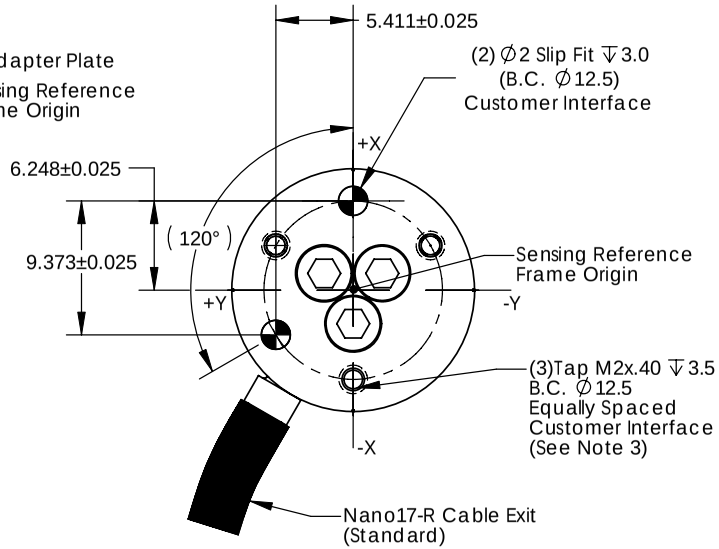
Rev.	Description	Initiator	Date
13	Eco 7635; Add -E cable exit	CCS	3/15/2010



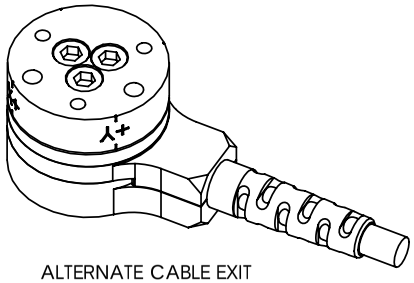
MOUNTING SIDE



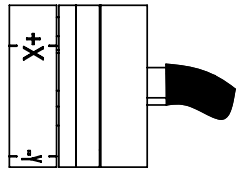
SIDE VIEW



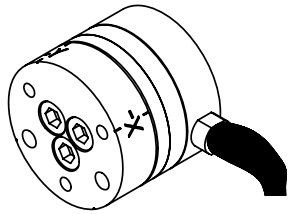
TOOL SIDE



ALTERNATE CABLE EXIT
NANO17-E



ALTERNATE CABLE EXIT
NANO17-A



NANO17-R ISOMETRIC VIEW

Notes:
1. Mounting and Tool Adapter made of Aluminum.
   Transducer made of hardened Stainless Steel.
2. WARNING: DO NOT LOOSEN OR REMOVE INTERFACE PLATES
   OR CABLE FITTING DUE TO POTENTIAL DAMAGE.
3. DO NOT EXCEED INTERFACE DEPTH, MAY CAUSE DAMAGE.
4. Connector (not shown) has 17mm diameter and is 67.5mm long.

NOTES: UNLESS OTHERWISE SPECIFIED

DO NOT SCALE DRAWING. DRAWN IN SOLIDWORKS.
ALL DIMENSIONS ARE IN MILLIMETERS.

3rd ANGLE PROJECTION

ATI INDUSTRIAL AUTOMATION

1031 Goodworth Drive, Apex, NC 27539, USA
Tel: +1.919.772.0115 Email: info@ati-ia.com
Fax: +1.919.772.8259 www.ati-ia.com
ISO 9001 Registered Company

DRAWN BY:	B. Digeso
CHECKED BY:	D. Perry
WEIGHT LBS:	
ASSEMBLY REF:	

TITLE: Nano17 Transducer

SCALE	SIZE	DRAWING NUMBER	REVISION
3:1	B	9230-05-1073	13

PRODUCT RELEASE # " DATE: SHEET 1 OF 1