POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering

Master's Degree in Mechanical Engineering
ERASMUS+ / PROGRAMME COUNTRIES with UPC-ESEIAAT

Master Thesis

STUDY AND CONSTRUCTION OF AN RCRCR SPATIAL MECHANISM

Home Supervisor:
Prof. Stefano Pastorelli

I.R.I. Supervisor:
Prof. Enric Celaya

U.P.C. - E.S.E.I.A.A.T Supervisor:
Prof. Rita Maria Planas

Candidate:
Giuseppe Notaristefano

April 2019

Working Environment

This thesis has been carried out in the research center *"Institut de Robotica i Informatica Industrial (I.R.I.)"* in cooperation with the *"Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa (U.P.C.-E.S.E.I.A.A.T.)"*, during the mobility period of Erasmus between U.P.C. and Politecnico di Torino.

The Institute has three main objectives: to promote fundamental research in robotics and applied Informatics, to cooperate with the community in industrial technological projects, and to offer scientific education through graduate courses. The institute was divided into four lines: *automatic control, kinematics and robot design, mobile robotics* and *perception and manipulation.*
I have took part in the *kinematics and robot design* group that, carries out fundamental research on design, construction, and motion analysis of complex mechanisms and structures as parallel manipulators, multi-fingered hands, reconfigurable mechanisms, or cooperating robots.

The thesis has been carried out during the Erasmus mobility, for a period of six months entirely spent for the project activities. I have cooperated, for most of the activities, with my host supervisor *Prof. Enric Celaya.* Taking advantages of the research center specialised staff, I have cooperated, for some important analysis with *Prof. Federico Thomas* and the research engineer *P.Grosch* in the *Kinematic and Robot Design Laboratory.* It has been also important the cooperation with *Prof. R.M. Planas* (E.S.E.I.A.A.T. supervisor) for the project defence which took place in the host institution.

UPC Escola Superior d'**Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa**. ESEIAAT

Institut de Robòtica i Informàtica Industrial

Contents

Contents

List of Figures **1**

List of Tables **5**

Abstract **7**

Introduction **9**

1 Study of the RCRCR Mechanism **11**

1.1 State of the art . . . 11

1.2 Description of the mechanism . . . 13

1.3 Computation of the I-O equations . . . 14

1.3.1 Input-Output functions . . . 15

1.3.1.1 Input(θ_5)-Output(θ_1) function . . . 15

1.3.1.2 Input(θ_5)-Output (θ_3) function . . . 15

1.3.1.3 Input(θ_3)-Output (θ_5) function . . . 15

1.4 Fixed parameters to the particular example of study . . . 15

1.5 Analysis with Maple®software . . . 16

1.6 Implementation choiches: assembly mode and driving pair . . . 17

1.6.1 Choice of the assembly mode . . . 17

1.6.2 Choice of the driving pair . . . 18

1.7 Extreme configurations: using an alternative driving pair . . . 18

1.8 Discriminating the two assembly modes . . . 21

1.8.1 Alternative resolution of the fourth-degree equations . . . 22

2 Designing the Links Shape **25**

2.1 Positioning the joints . . . 25

2.1.1 First design: straight links . . . 32

2.1.2 Notation . . . 34

2.2 Collisions detection . . . 34

2.2.1 First step: test for coplanarity of segments . . . 35

2.2.2 Second step: test for interference between coplanar segments . . . 36

2.3 Algorithm implementation . . . 38

2.4 Test results: detected collisions . . . 39

2.4.1 Second design: avoiding collisions . . . 41

2.4.1.1 Avoiding collisions between l_{5-1} and d_{4-4F} . . . 41

2.4.1.2 Avoiding collisions between l_{1-2} and l_{2V-3} . . . 43

2.5 Implementation of the joint envelope . . . 43

3 Construction of the Mechanism 45
3.1 Selection of the scaling factor . . . 45
3.2 Used equipment . . . 46
3.2.1 Printing guidelines . . . 46
3.3 Generic rotational joint model . . . 48
3.4 Generic cylindrical joint model . . . 49
3.5 Generic link model . . . 51
3.6 RCRCR mechanism CAD model . . . 53
3.7 Physical implementation . . . 56
3.7.1 Assembling of the joints . . . 56
3.7.2 Assembling of the links . . . 56
3.7.3 Bonding of the reference surfaces . . . 56
3.7.4 Installation on the base . . . 57
3.7.5 Installation of the motors . . . 58
3.8 Mechanism problems diagnosis . . . 63
3.9 Design improvements . . . 63
3.10 Final mechanism problems diagnosis . . . 65
3.11 Prototype real component . . . 66

4 Implementation of the Motors 75
4.1 Learning the use of the Dynamixel Motors . . . 75
4.1.1 Brief description of the motors environment . . . 76
4.1.2 Motor model: Dynamixel AX-12A . . . 77
4.1.3 Control Table and functioning type of the motor Dyamixel AX-12A . 78
4.1.4 Feed-Forward position control:point-to-point motion in the joint space 79
4.1.5 Further considerations . . . 83
4.2 Installed Motor: Robotis Dynamixel MX-64T . . . 87
4.2.1 Control Table and functioning type of the motor Dyamixel MX-64T . 88
4.2.2 Operating mode chosen: position control mode . . . 90
4.2.3 Considerations on the motor behaviour . . . 93
4.2.4 Feed-forward position control implemented: linear condition . . . 97
4.2.5 Feed-forward position control implemented: test on the mechanism . . 100

5 Cost Analysis 103

Conclusion 105

Attachment 107
Appendix A . . . 107
Appendix B . . . 109
Appendix C . . . 121
Appendix D . . . 130
Appendix E . . . 148

Acknowledgement 155

Bibliography 157

List of Figures

1.1 *Example of a CAD model of the RCRCR mechanism with the chosen fixed parameters [1].* 12
1.2 *RCRCR mechanism with general proportions (adapted from [15]).* 13
1.3 *I-O functions involving the three rotational pair variables θ_1, θ_3, θ_5.* 16
1.4 *Relationship θ_5 with the displacement s_2 and s_4.* 17
1.5 *Plot $\theta_5 - \theta_5$.* 19
1.6 *Plot $\theta_3 - \theta_5$.* 20
1.7 *Plot $\theta_1 - \theta_5$.* 20
1.8 *Plotting the solutions with different colours.* 22
1.9 *Relationship between θ_5 and θ_1 obtained with the alternative method.* 23

2.1 *Reference frame origins.* 26
2.2 *$\theta_5 - \theta_1$ plot.* 26
2.3 *Trajectory of O_2 wrt the O_1.* 29
2.4 *Trajectory of O_{2V} wrt the O_1.* 29
2.5 *Trajectory of O_3 wrt the O_1.* 30
2.6 *Trajectory of O_4 wrt the O_1.* 30
2.7 *Trajectory of O_{4V} wrt the O_1.* 31
2.8 *Trajectory of O_5 wrt the O_1.* 31
2.9 *All the trajectories of the reference frames wrt O_1.* 32
2.10 *Link O_5 to O_1.* 33
2.11 *RCRCR mechanism, wire frame representation, in a generic instant.* 33
2.12 *Two generic segments in the space.* 35
2.13 *Crossing segments.* 36
2.14 *No crossing segments.* 36
2.15 *Crossing segments in a plane.* 37
2.16 *The four different area for crossing segments in a plane.* 37
2.17 *No Crossing segments in a plane.* 38
2.18 *The four different area for no-crossing segments in a plane.* 38
2.19 *Collisions 1,between the links $l_{5-1} \leftrightarrow d_{4-4F}$.* 40
2.20 *Collisions 2, between the links $l_{1-2} \leftrightarrow l_{2V-3}$.* 41
2.21 *Changes to the link l_{5-1}.* 42
2.22 *Changes to the Link 12.* 43
2.23 *Simulation with the joint envelopes.* 44

3.1 *CAD part, in a non optimized printing orientation.* 47
3.2 *Layer by layer model, in a non optimize printing orientation.* 47
3.3 *Model in a optimized printing orientation.* 48
3.4 *Rotational pair, section view.* 48

3.5 *Rotational pair, isometric views.* 49
3.6 *Cylindrical pair, Igus-DryLin ®Linear Plain Bearings-RJUM-01-12* and Igus-DryLin ®shaft-AWMP-12 [11]. 50
3.7 *Cylindrical pair, section view.* 50
3.8 *Cylindrical pair, isometric view.* 51
3.9 *Cylindrical pair,the two half part of the cover.* 51
3.10 *Generic link between two joints.* 52
3.11 *Generic link between two joints, chunks mate.* 52
3.12 *CAD reference frame.* 53
3.13 *RCRCR CAD assembly.* 54
3.14 *RCRCR CAD assembly,particular l_{4-5}.* 55
3.15 *Links assembly, reference surfaces.* 56
3.16 *Bounding reference surfaces.* 57
3.17 *Base of the mechanism.* 57
3.18 *Installation on the base, joint J_5.* 58
3.19 *Deny rotation pin, joint J_1 (right) and J_5 (left).* 59
3.20 *Motor support for the joint J_1.* 60
3.21 *Motor support for the joint J_1, real model (left) and CAD model(right).* 60
3.22 *Motor support for the joint J_5.* 61
3.23 *RCRCR CAD model.* 62
3.24 *Generic revolute joint, new design.* 64
3.25 *Pin to deny the rotations of the bar.* 64
3.26 *RCRCR CAD model, top view.* 66
3.27 *RCRCR real prototype, top view.* 66
3.28 *RCRCR CAD model, main view (configuration: $\theta_5 = 148.96$ deg, $\theta_1 = 37,30$ deg).* 67
3.29 *RCRCR real prototype, main view (configuration: $\theta_5 = 148.96$ deg, $\theta_1 = 37,30$ deg).* 68
3.30 *RCRCR CAD model, joint J_1 and motor placed.* 69
3.31 *RCRCR real prototype, joint J_1 and motor placed.* 69
3.32 *RCRCR CAD model, joint J_3.* 70
3.33 *RCRCR real prototype, joint J_3.* 70
3.34 *RCRCR CAD model, joint J_2 and J_3.* 71
3.35 *RCRCR real prototype, joint J_2 and J_3.* 71
3.36 *RCRCR CAD model, joint J_5 and motor placed.* 72
3.37 *RCRCR real prototype, joint J_5 and motor placed.* 72
3.38 *RCRCR real prototype, main view (configuration: $\theta_5 = 303.66$ deg, $\theta_1 = 296,66$ deg).* 73
3.39 *RCRCR real prototype, main view (configuration: $\theta_5 = 263.56$ deg, $\theta_1 = 281.93$ deg).* 74

4.1 *(1)Robot 3R and (2) USB2Dynamixel hardware.* 76
4.2 *Robotis Dynamixel, motors environment [10].* 76
4.3 *Dynamixel AX-12A Robot Actuator [10].* 77
4.4 *Trapezoidal velocity law.* 80
4.5 *Feed Forward Position control.* 81
4.6 *Comparison of real and theoretical position and velocity obtained.* 83
4.7 *Feed-forward position control, influence of K_p.* 84

4.8 *Feed-forward position control, influence of t_f.* 85
4.9 *Dynamixel AX-12A, coordinate control of three motor.* 86
4.10 *Robotis Dynamixel MX-64T actuator.* 87
4.11 *Dynamixel Motor MX-64T, Position Control Mode block diagram [10].* 91
4.12 *Set of the Velocity profile, Dynamixel Motor MX-64T [10].* 93
4.13 *Feed Forward Position control, trapezoidal sahpe of velocity.* 94
4.14 *Feed Forward Position control, influence of the final time.* 95
4.15 *Feed Forward Position control, influence of K_I.* 96
4.16 *Plot relationship θ_5-θ_1* 97
4.17 *Feed Forward Position control J_5 motor (θ_5), from the configuration 1 to 2.* . . 98
4.18 *Feed Forward Position control J_1 motor (θ_1), from the configuration 2 to 3.* . . 99
4.19 *Feed Forward Position control, test on the mechanism, motor in J_5.* 101

5.1 *Plot, $\theta_1 - \theta_3$.* 121
5.2 *Plot, $\theta_1 - \theta_5$.* 121
5.3 *Plot, $\theta_1 - \theta_2$.* 122
5.4 *Plot, $\theta_1 - \theta_4$.* 122
5.5 *Plot, $\theta_1 - s_2$.* 123
5.6 *Plot, $\theta_1 - s_4$.* 123
5.7 *Plot, $\theta_3 - \theta_1$.* 124
5.8 *Plot, $\theta_3 - \theta_5$.* 124
5.9 *Plot, $\theta_3 - \theta_2$.* 125
5.10 *Plot, $\theta_3 - \theta_4$.* 125
5.11 *Plot, $\theta_3 - s_2$.* 126
5.12 *Plot, $\theta_3 - s_4$.* 126
5.13 *Plot, $\theta_5 - \theta_3$.* 127
5.14 *Plot, $\theta_5 - \theta_1$.* 127
5.15 *Plot, $\theta_5 - \theta_2$.* 128
5.16 *Plot, $\theta_5 - \theta_4$.* 128
5.17 *Plot, $\theta_5 - s_2$.* 129
5.18 *Plot, $\theta_5 - s_4$.* 129

List of Tables

1.1 *Fixed parameters for our case of study.* 16
1.2 *Lengths of the cylindrical pairs in assembly mode 1 and 2.* 18
1.3 *Range of motion of rotational variables θ_1, θ_3, and θ_5 for assembly mode 2 [1].* 18

2.1 *Configuration corresponding to the point 1.* 27
2.2 *Configuration corresponding to the point 2.* 27
2.3 *Configuration corresponding to the point 3.* 27
2.4 *Configuration corresponding to the point 4.* 27
2.5 *Determinant changes of sign detected.* 39
2.6 *Collisions detected.* 39

3.1 *Fixed parametres scaled.* 45
3.2 *3D printers used.* 46
3.3 *Initial configuration of the physical model.* 59

4.1 *Dynamixel AX-12A, motor specifications [10].* 77
4.2 *Dynamixel AX-12A, Control Table [10].* 78
4.3 *Dynamixel MX-64T, motor specification [10].* 87
4.4 *Dynamixel MX-64T, Control Table [10].* 89
4.5 *Dynamixel MX-64T, table of conversion [10].* 90

Abstract

The purpose of this work is the study of the kinematics of an RCRCR spatial mechanism and the design of a 3D prototype able to perform the full range of mobility allowed in theory.

To achieve this, the solution of the input-output equations of the mechanism has been implemented in a commercial mathematical software (Maple) that allowed the analysis of all the aspects relevant to our task. It has been realized that driving the mechanism along the full cycle of configurations requires the alternative actuation on two different rotational pairs, which have been chosen to optimize the controllability of the mechanism.

To design the shape of each link, so that the interference between different parts of the mechanism is avoided along the whole range of movement, a process of collision detection between links has been developed.

Given the final link shapes, a complete design of all mechanism parts, including the fixation of the joints bearings and the connections between joints and links, has been carried out in order to build them using a 3D printing process called FDM (fusion deposition modelling). A simulation of the motion of the complete design has been performed, showing that, using the two motors, the mechanism can perform a complete motion cycle with no collisions.

Introduction

The goal of this project is the kinematic analysis of the general RCRCR spatial mechanism and the study of one particular instance of it in order to design and implement a physical prototype able to perform the full range of movement theoretically allowed.

The RCRCR is a particular case of the family of 3R-2C spatial mechanisms, consisting in a set of five links connected by three rotational and two cylindrical pairs forming a kinematic loop. Since each R pair provides one rotational d.o.f. and each C pair provides one translational and one rotational d.o.f., all mechanisms in the family of 3R-2C have a total of seven d.o.f. or variable joint parameters. The fact that the mechanism forms a loop means that the last link can not move freely in space, but must remain attached to the first, and this removes 6 d.o.f. from the global mobility of the mechanism. This means that, in general, 3R-2C mechanisms have mobility 1, so that the motion of the mechanism can be produced by driving a single input pair variable.
The set of values taken by all the pair variables in a feasible combination is called a configuration of the mechanism. A central task in Kinematics is the determination of all possible configurations of a mechanism for each possible value of the variable taken as input. This is usually done by obtaining input-output functions relating the value of the input variable with the value of each other variable considered as the output. Solving a mechanism consists in providing its input-output functions, and the process to do that may be very complex. For some mechanisms, it is possible to decouple the equations involving the rotational variables from those involving the translational ones, so that the rotational part can be solved first independently of the translations, and their solution is relatively simple. Among all the spatial mechanisms having a single global d.o.f. and for which its solution can not be decoupled, the 3R-2C family is the simplest one, and in this family, the RCRCR mechanism is the less complex to solve. This fact is what gives a special interest to this mechanism and motivates its choice as the subject of this work.

The kinematic analysis of a mechanism is commonly performed without taking into consideration the physical material of which each link is made of, nor its exact shape or its volume, i.e., each link is described simply by the fixed parameters defining the spatial transformation imposed on the relative position between the reference frames attached to consecutive pairs. Thus, a configuration of a mechanism can be theoretically feasible but impossible to reach in a real implementation of it, due to physical interferences between different links. Our goal, then, is to find appropriate shapes for all links of our particular mechanism, so that all theoretically feasible configurations can be physically realizable, and which at the same time are capable to keep their rigidity and resist the forces applied to them when driving the input pair to follow a trajectory along a full cycle of configurations.

The final goal of this work will be the physical construction and assembly of the links with the adequate motorization and control of one or more input pairs, so as to achieve the movement of the mechanism through its whole range of mobility by the actuation of one servo at a time.

Thus, the following report is divided into six chapters. In the first (Chapter 1), the theoretical background of the project is developed. Next, Chapter 2 discusses the design of the links shape and Chapter 3 presents the solutions adopted for the construction of the mechanism. At the end, in Chapter 4 is shown the implementation of the motors planned and Chapter 5 presents a brief overview on the cost analysis.

Chapter 1

Study of the RCRCR Mechanism

The RCRCR mechanism and its inversions RCRRC and RRCRC (where the notation indicates which rotational pair is taken as input by placing it in the first position from left), have been thoroughly analyzed in the literature [15, 3, 4, 16]. To perform the complex algebraic manipulations required to obtain the input-output equations, dual number 3×3 *matrices* have been most often used instead of the more common 4×4 homogeneous transformation matrices. Next, we review the solution of the mechanism in general and, using an implementation of the equations in a commercial mathematical software (Maple), we obtain the numerical solutions corresponding to the particular parameters of the mechanism under study.

1.1 State of the art

The Kinematics of the RCRCR mechanism, has been deeply analysed in literature. The most relevant studies [3, 15, 16] are concentrated on the process of obtaining of the input output (I-O) equations and solving them. Starting from the first results [15], we count remarkable improvements in the obtaintion of the I-O relationships among the variables of the mechanism [3, 4, 8, 5].
Since the main interest of the 3R2C mechanisms is of a theoretical nature, their physical construction has received much less attention. One of the few attempts we can find in the literature to build a real model is [7]. Interestingly, the author mentions that some of the theoretically possible configurations of the mechanism could not be reached in the real model due to link interferences. This tells us that if links are not designed with the explicit purpose to avoid collisions between them, they are likely to occur. This problem is further illustrated in figure 1.1, where a CAD model of the mechanism under study has been implemented without taking into consideration the possible collisions of links when performing the full cycle of configurations. It is easy to foresee that the provided shape of links will cause multiple collisions during the relative motion.

Thus, the purpose of this work is to develop a method for link design that takes into account link interferences when the mechanism moves along its whole rage of mobility.

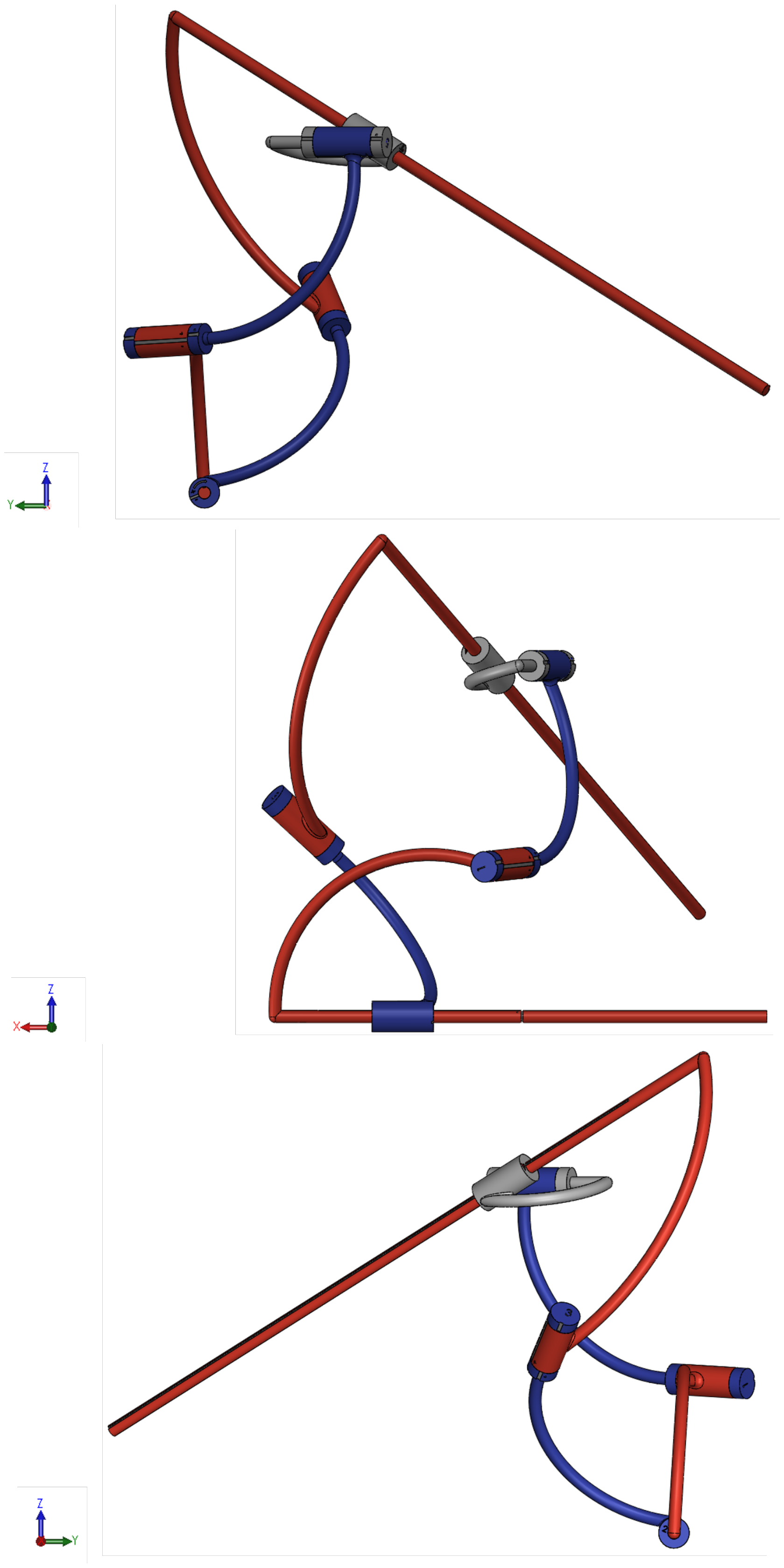


Figure 1.1: *Example of a CAD model of the RCRCR mechanism with the chosen fixed parameters [1].*

1.2 Description of the mechanism

The mechanism of study is an RCRCR mechanism as shown in figure 1.2. It is composed of three **revolute pairs** and two **cylindrical pairs**. Referring to the figure 1.2, joints **1**,**3** and **5** correspond to revolute pairs and **2** and **4** correspond to cylindrical pairs. The pairing axes of 1,2, 3, 4 and 5 are specified, respectively, by unit line vectors x_1, x_2, x_3, x_4, and x_5. The unit line vector z_1 lies along the common perpendicular between x_1 and x_2, directed from x_1 to x_2. Then, z_2, z_3, z_4, and z_5 are similarly defined.
The generic third axis is defined by $y_i = z_i \times x_i$.

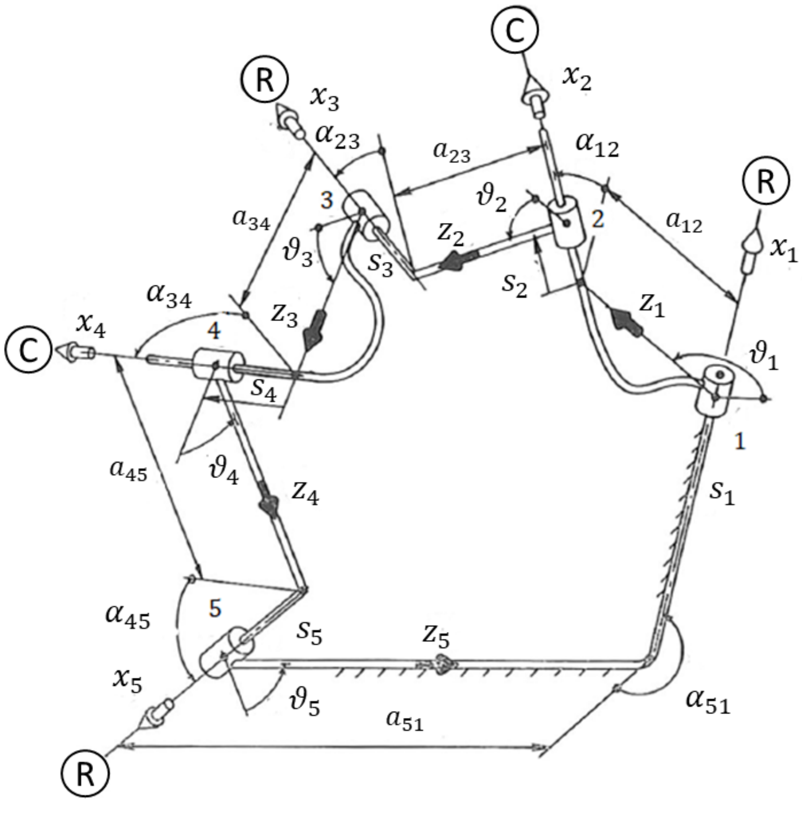


Figure 1.2: *RCRCR mechanism with general proportions (adapted from [15]).*

With reference to the figure 1.2, a general configuration of the RCRCR mechanism is **completely defined** by two sets of five dual angles, as follows:

- Between adjacent pairing axes:

$$\hat{\alpha_{ij}} = \alpha_{ij} + \epsilon a_{ij}$$

  where $\hat{\alpha_{ij}}$ is the dual angle between x_i and x_j, having positive sense with respect to z_i.

- Between adjacent common perpendiculars:

$$\hat{\theta}_i = \theta_i + \epsilon s_i$$

  where $\hat{\theta}_i$ is the dual angle between z_{i-1} and z_i, having positive sense with respect to the revolute axis x_i. By convention when $i = 1$, $i - 1 = 5$.

The angles $\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5$ and the two sliding components along the cylindric axes s_2 and s_4 constitute the **seven linkage variables**. Instead, the five dual angles $\hat{\alpha_{12}},\ \hat{\alpha_{23}},\ \hat{\alpha_{34}},\ \hat{\alpha_{45}},\ \hat{\alpha_{51}}$ and the three constant offsets along the revolute axis $s_1,\ s_3$ and s_5 constitute the **thirteen real parameters** necessary to specify a RCRCR mechanism. Therefore, the mechanism presents only **one dof**, what means that, to move our mechanism, we need to manage only one of the seven linkage variables.
A brief digression on the dual numbers can be found on the Appendix A.

1.3 Computation of the I-O equations

The RCRCR mechanism can be defined by the **loop equation**[1]:

$$I = \hat{R}x(\hat{\theta_1})\hat{R}z(\hat{\alpha_{12}})\hat{R}x(\hat{\theta_2})\hat{R}z(\hat{\alpha_{23}})\hat{R}x(\hat{\theta_3})\hat{R}z(\hat{\alpha_{34}})\hat{R}x(\hat{\theta_4})\hat{R}z(\hat{\alpha_{45}})\hat{R}x(\hat{\theta_5})\hat{R}z(\hat{\alpha_{51}}), \tag{1.1}$$

where $\hat{\theta_1}, \hat{\theta_3}$ and $\hat{\theta_5}$ are the **revolute pairs** and $\hat{\theta_2}$ and $\hat{\theta_4}$ are the **cylindric pairs**.
We rearrange (1.1) so as to have a C pair at both ends of the right hand side:

$$\hat{R}z(-\hat{\alpha_{34}})\hat{R}x(-\hat{\theta_3})\hat{R}z(-\hat{\alpha_{23}}) = \hat{R}x(\hat{\theta_4})\hat{R}z(\hat{\alpha_{45}})\hat{R}x(\hat{\theta_5})\hat{R}z(\hat{\alpha_{51}})\hat{R}x(\hat{\theta_1})\hat{R}z(\hat{\alpha_{12}})\hat{R}x(\hat{\theta_2}). \tag{1.2}$$

Applying the dual Euler's decomposition (see Appendix A) to both sides, excluding the cylindric pairs:

$$\hat{R}z(-\hat{\alpha_{34}})\hat{R}x(-\hat{\theta_3})\hat{R}z(-\hat{\alpha_{23}}) = \hat{R}x(\hat{\varphi_1})\hat{R}z(\hat{\phi_1})\hat{R}x(\hat{\psi_1}) \tag{1.3}$$

$$\hat{R}z(\hat{\alpha_{45}})\hat{R}x(\hat{\theta_5})\hat{R}z(\hat{\alpha_{51}})\hat{R}x(\hat{\theta_1})\hat{R}z(\hat{\alpha_{12}}) = \hat{R}x(\hat{\varphi_2})\hat{R}z(\hat{\phi_2})\hat{R}x(\hat{\psi_2}), \tag{1.4}$$

and substituting in 1.2:

$$\hat{R}x(\hat{\varphi_1})\hat{R}z(\hat{\phi_1})\hat{R}x(\hat{\psi_1}) = \hat{R}x(\hat{\theta_4}+\hat{\varphi_2})\hat{R}z(\hat{\phi_2})\hat{R}x(\hat{\psi_2}+\hat{\theta_2}) \tag{1.5}$$

The necessary condition for (1.5) to be fulfilled is $\hat{\phi_1} = \pm\hat{\phi_2}$, which can be expressed equivalently as $cos\hat{\phi_1} = cos\hat{\phi_2}$. This condition is also sufficient since $\hat{\theta_4}$ and $\hat{\theta_2}$ correspond to the cylindrical pairs, so their real and dual parts can be chosen to satisfy $\hat{\theta_4} + \hat{\varphi_2} = \hat{\theta_1}$ and $\hat{\psi_2} + \hat{\theta_2} = \hat{\psi_1}$.
The value of $cos\hat{\phi_1}$ depends on the rotational variable θ_3, and is given by the matrix element $[1, 1]$ of the left hand side of equation(1.3).
Writing $\hat{\phi_1} = \phi_1 + \epsilon d_1$ the real and the dual parts of $cos\hat{\phi_1}$ as functions of θ_3 are given by:

$$\begin{cases} cos\phi_1 = A + Bcos\theta_3 \\ d_1 sin\phi_1 = C + Dcos\theta_3 + Esin\theta_3 \end{cases} \tag{1.6}$$

where A, B, C, D and E depend on $\hat{\alpha_{23}}, \hat{\alpha_{34}}$ and s_3.

The value of $cos\hat{\phi_2}$ depends on the rotational variables θ_1 and θ_5, and is given by the $[1, 1]$ of the matrix element of the left hand side of equation (1.4).
Writing $\hat{\phi_2} = \phi_2 + \epsilon d_2$ the real and the dual parts of $cos\hat{\phi_2}$ as functions of θ_1 and θ_5 are given by:

$$\begin{cases} cos\phi_2 = F + Gcos\theta_1 + Hsin\theta_1 \\ d_2 sin\phi_2 = J + Kcos\theta_1 + Lsin\theta_1 \end{cases} \tag{1.7}$$

where F, G, H, J, K, L are functions of $\hat{\alpha_{12}}, \hat{\alpha_{45}}, \hat{\alpha_{51}}, \hat{\theta_5}$ and s_1. So we can collect the equations 1.6 and 1.7:

$$\begin{cases} A + Bcos\theta_3 = F + Gcos\theta_1 + Hsin\theta_1 \\ C + Dcos\theta_3 + Esin\theta_3 = J + Kcos\theta_1 + Lsin\theta_1 \end{cases} \tag{1.8}$$

Summarizing, equations (1.8) are the **necessary and sufficient condition** for the RCRCR mechanism to close, and only involve the real variables of the rotational pairs θ_1, θ_3 and θ_5. It should be noted that the choice of the variables of the three revolute pairs is not casual. In fact, in real mechanism, it is simpler to drive a revolute pair than a cylindrical one. Therefore, in what follows, we will obtain the input output functions using each one of these three variables as input.

1.3.1 Input-Output functions

To obtain the input-output function between an input and an output variable, we only need to eliminate the third variable from the equations (1.8) [1, 3].

1.3.1.1 Input(θ_5)-Output(θ_1) function

We can isolate $cos(\theta_3)$ from the first equation and $sin(\theta_3)$ from the equation obtained with the appropriate combination of the two equations to eliminate $cos(\theta_3)$ [1].
Using the identity $sin^2(\theta_3) + cos^2(\theta_3) = 1$, we have an $2nd$ degree polynomial in $sin(\theta_1)$ and $cos(\theta_1)$, whose coefficients are 2nd degree polynomials of $sin(\theta_5)$ and $cos(\theta_5)$.
Applying the tangent half-angle substitution $T_1 = tan(\theta_1/2)$, we obtain a $4th$ degree polynomial for T_1 in which the coefficients depend on θ_5. For each given value of the input angle θ_5, the corresponding value of θ_1 can be obtained computing the roots of the resulting $4th$ degree polynomial [1].

1.3.1.2 Input(θ_5)-Output (θ_3) function

Could be obtained by eliminating θ_1 istead θ_3 from the equation 1.8 and using the same procedure shown in the section 1.3.1.1.

1.3.1.3 Input(θ_3)-Output (θ_5) function

We need to eliminate θ_1 from the equation 1.8, but in this case, the elimination process gives rise to expressions for $sin(\theta_1)$ and $cos\theta_1$ that are quadratic in $sin(\theta_5)$ and $cos(\theta_5)$, then the identity $sin^2(\theta_3) + cos^2(\theta_3) = 1$ involves a $4th$ degree polynomial in $sin(\theta_5)$ and $cos(\theta_5)$, so we need to find the roots of an $8th$ degree polynomial in $T_5 = tan(\theta_5/2)$ for each input value of θ_3, what can be done numerically.

1.4 Fixed parameters to the particular example of study

For this study, we have chosen to use the same parameters that have been used in many previous works [15, 3], shown in the table 1.1 :

$i =$	**1**	**2**	**3**	**4**	**5**
α_{ij} (deg)	60	45	35	30	10
a_{ij}	25	30	40	10	32
s_i	30	/	25	/	0

Table 1.1: *Fixed parameters for our case of study.*

Where $j = i + 1$ and by convention, when $i = 5 \rightarrow j = 1$. Referring to table (1.1), unit of measurement are not been inserted, because the mechanism analysis is independent of the scale. It should be noted that in the table 1.1 there are only the fixed parameters, in fact the parameters s_2 and s_4, that are unknown, constitute the two cylindrical displacements.

1.5 Analysis with Maple®software

In order to compute the I-O functions 1.3.1 for each couple of input-otput variables, the Maple ®software has been used.
The created **procedures**, shown in the Appendix B, compute the I-O equations choosing an input variable and an output one (section 1.3.1). Given a **value of the input variable**, the procedure returns the **four possible solutions** that correspond to the **four possible mechanism configurations** with this input value, when they exist.
In fact, giving a value of revolute input variable, we can obtain the value of the remaining two revolute pair angles from the equations as explained in the Section 1.3.1. The other 4 linkage variables can be obtained in sequence, from appropiate members of the matrix (equation (1.2)) [3, 1].
In order to obtain the configurations for a 360° range of an input variable, an algorithm has been created that, **samples** the input variable, solves 360/*step* times the I-O equation and obtains the plots as shown in figure 1.3.

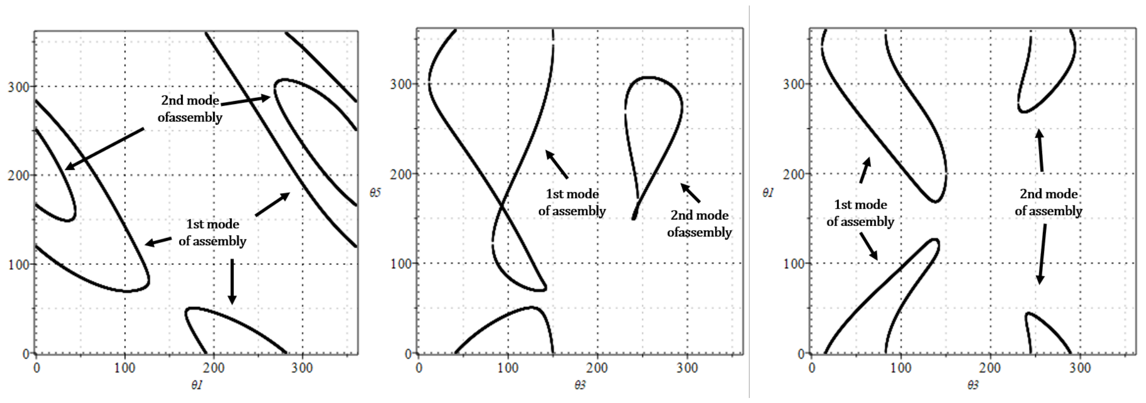


Figure 1.3: *I-O functions involving the three rotational pair variables θ_1, θ_3, θ_5.*

The obtained plots, as shown in figure 1.3, show us that there are **two different modes of assembly** of the mechanism. This means that we can have a mechanism that, with the given fixed parameters, can move continuously along each assembly mode, but can't pass from one assembly mode to the other without *breaking*.
The other plots regarding the relationship between each possible input variable and the other six linkage variables are shown in the Appendix C.

1.6 Implementation choiches: assembly mode and driving pair

Once the plots have been obtained, our goal is the construction of a mechanism performing the provided range of motion using one driving pair.

1.6.1 Choice of the assembly mode

As we have seen, our mechanism has two possible modes of assembly. In a real implementation of the mechanism, the shape of each link has to be designed in such a way that no interference occurs between the different parts of the mechanism during its whole cycle of motion.
Clearly, each assembly mode will impose different restrictions on the shape of the links in order to achieve this. For this reason, in this work, we will **restrict our study to a single assembly mode** and will try to design the link shapes appropriate for it.
Our choice about which assembly mode to implement will take into account the **maximum length of the links** that will be required, since longer links imply increased structural problems, stronger rigidity requirements, and more powerful motors to move the heavier links. A first analysis reveals that the longest parts of the mechanism correspond, by far, to the guides of the two cylindrical pairs.

Thus, we need to compare the lengths of the cylindrical pairs for the two assembly modes. We can obtain the range of displacement of each cylindrical pair in each assembly mode from the figures obtained for the input-output functions involving s_2 and s_4 (figure 1.4). The corresponding lengths are collected in table 1.2.

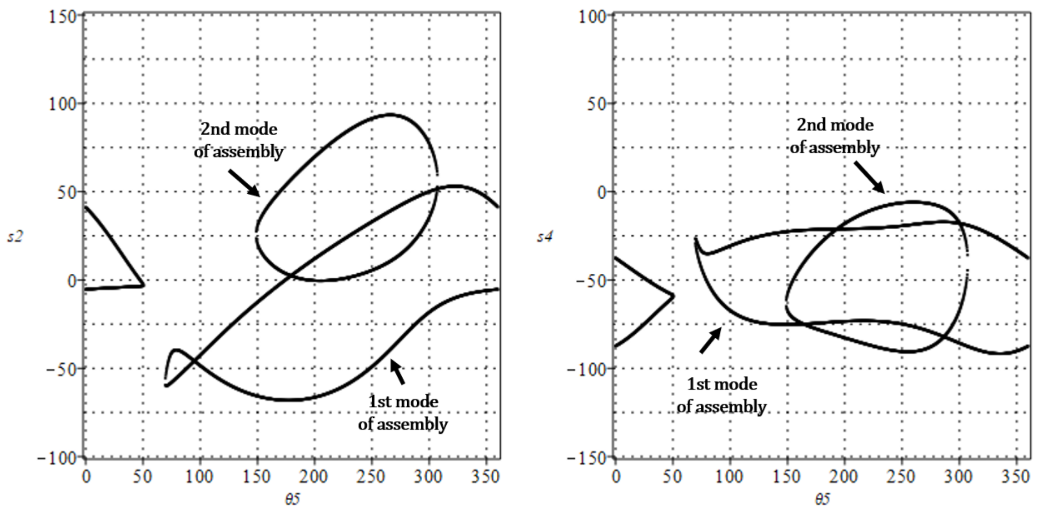


Figure 1.4: *Relationship θ_5 with the displacement s_2 and s_4.*

Displacement		**Min**	**Max**	**Total length**
Assembly mode 1	s_2	-68,12	53,51	121,63
	s_4	-91,75	-18,42	73,33
Assembly mode 2	s_2	-0,48	93,36	93,84
	s_4	-90,57	-5,97	84,60

Table 1.2: *Lengths of the cylindrical pairs in assembly mode 1 and 2.*

We can see that the longest length corresponds to s_2 in assembly mode 1. Therefore, to reduce the maximum length of the mechanism, we decided to use the **assembly mode 2** for the physical implementation.

1.6.2 Choice of the driving pair

We already know that the mechanism has mobility one, what means that the mechanism can be driven by actuating a single input pair. In order to properly design the mechanism, it is necessary to determine which pair has to be actuated, so that the motor housing can be conveniently accommodated.

Since a rotational pair is much simpler to actuate than a cylindrical one, our choice will be made between the three R pairs of the mechanism, corresponding to the variables θ_1, θ_3, and θ_5. From them, we will prefer the pair showing the widest range of motion since it will require a lower torque to drive the joint, what implies a better controllability of the motion and, at the same time, provides a better accuracy to reach specific configurations of the mechanism. The ranges of the rotational variables can be obtained from the appropriate graphs of the I-O functions, but we can also take them from [1], where they are computed accurately as given in table 1.3.

Driving pair	**Range Extremes (°)**		**Range length (°)**
θ_1	268,49	43,97+360	135,48
θ_3	230,73	293,99	63,26
θ_5	148,78	307,29	158,51

Table 1.3: *Range of motion of rotational variables θ_1, θ_3, and θ_5 for assembly mode 2 [1].*

In this case, the widest range of motion corresponds to θ_5, therefore, this will be our choice for the driving pair.

1.7 Extreme configurations: using an alternative driving pair

Even if the mechanism has mobility one, driving it along its full cycle of motion by actuating a single input pair is not possible because of the existence of singular configurations that appear at the extreme values of the input variable.

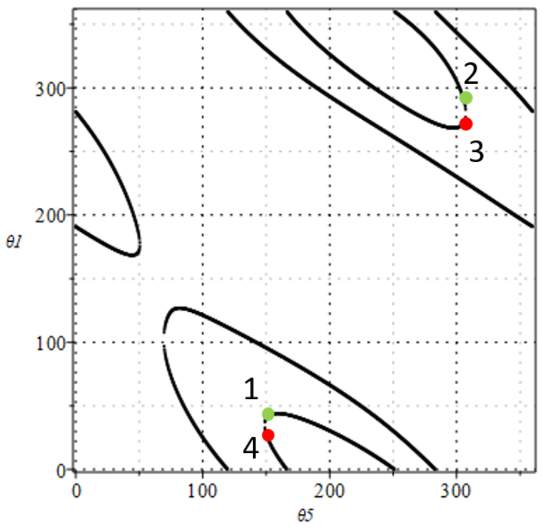


Figure 1.5: *Plot $\theta_5 - \theta_5$.*

Thus, we see that near an extreme position of θ_5 (point 2-3 and 4-1 figure 4.16), the velocity rate from θ_5 to θ_1 (or to any other variable) rises to infinity, what means that a very small variation in θ_5 corresponds to a very large variation of θ_1, which in the limit would imply to use an infinite torque to drive the mechanism in this configuration. A further problem of the extreme configurations is that, when θ_5 reaches an extreme value it can only move in one direction, but there are two possible paths that the mechanism can follow, and there is no way to make it to take the path we want by simply driving θ_5.

According to this, the actuation on a **second pair** is required to govern the motion of the mechanism in those situations in which θ_5 is not able to perform the required control. To select the second driving pair, as discussed above, we have to choose between one of the remaining R pairs: θ_1 and θ_3. In the case of θ_3, it looks like a good candidate to drive the mechanism around the extreme value of $\theta_5 = 307.30$ (point 2-3 in figure 1.6), since the I-O function there is rather flat.

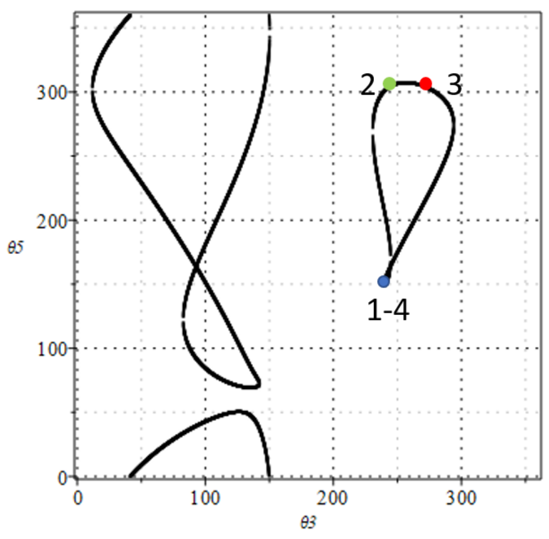


Figure 1.6: *Plot $\theta_3 - \theta_5$.*

However, if we look at the other extreme of $\theta_5 = 148.79°$ (point 1-4 in figure 1.6)the value of θ_3 changes rapidly and the I-O graph becomes vertical very soon. This makes θ_3 inappropriate to be used in this situation.

In the case of θ_1, instead, we can appreciate in fig 1.7 that it has a sufficiently wide range of values near both extrema of θ_5 in which the curve is approximately horizontal, what will allow a correct control of the mechanism in these situations.

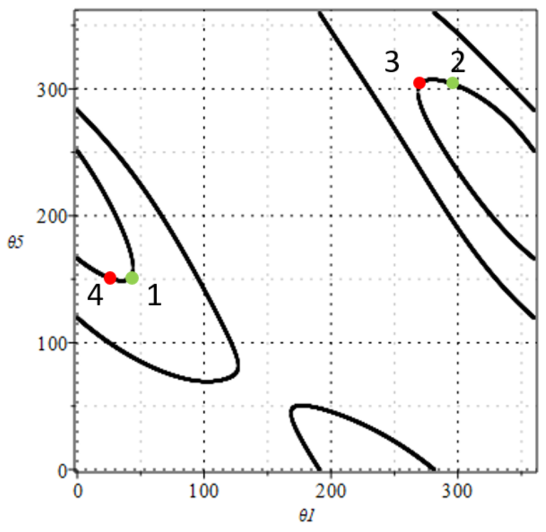


Figure 1.7: *Plot $\theta_1 - \theta_5$.*

Summarizing, for the physical implementation we will use the assembly mode 2 of the mechanism, and it will be alternatively driven by two motors, a first one actuating on θ_5 and a second one actuating on θ_1 only when the value of θ_5 approaches one of its extreme positions.

1.8 Discriminating the two assembly modes

Having two assembly modes means that, having two possible mechanisms which, despite being defined by the same parameters, are actually different.

In practice, when solving the mechanism, it would be useful to be able to distinguish the solutions corresponding to each one of the assembly modes. Even more, noting that each assembly mode has multiple solutions for each input value, we can classify each configuration as belonging to a different branch of solutions, in which the configurations of the mechanism change smoothly. If we want to describe a smooth trajectory along a continuous path, we need to differentiate between solution branches. This will be particularly useful in the next Chapter, where the collision detection between links will require monitored given property along consecutive nearby configurations of the mechanism. Next we perform this analysis taking θ_5 as input variable.

The implemented Maple algorithm returns from 0 to 4 solutions for each input value, depending on how many different configurations are possible for this specific input. Our problem consists in determining to what assembly mode and branch of solutions corresponds each of the 4 possible solutions. The order in which Maple returns the solutions depends on how it manages to solve the fourth degree equation, but this is not directly related with the assembly mode nor the branch of solutions. In order to investigate this we follow an empirical approach.

In order to perform this, a **colour assignment** has been implemented. Each solution root has been plotted with a different colour in order to understand how maple manages the order of the four solution for each step. In particular:

- **first solution** $\rightarrow$ black;
- **second solution** $\rightarrow$ red;
- **third solution** $\rightarrow$ green;
- **fourth solution** $\rightarrow$ blue;

The results obtained are shown in the figure 1.8.

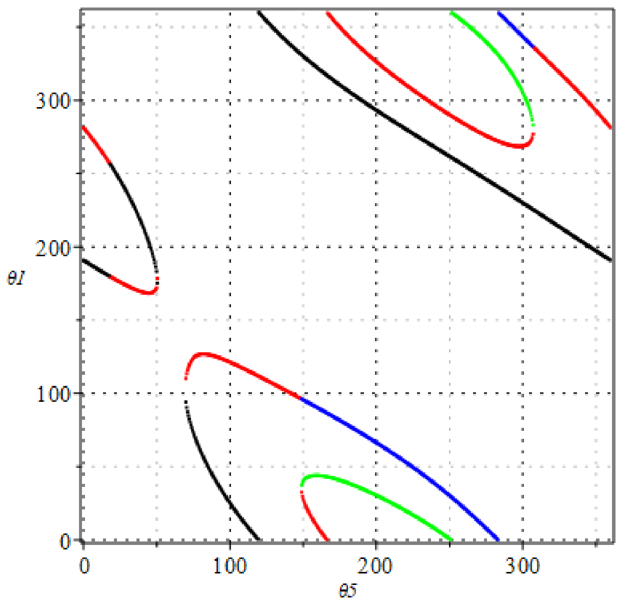


Figure 1.8: *Plotting the solutions with different colours.*

As we can see, for the **assembly mode 2** the results are clean: the upper branch has green colour, instead we have a red colour for the lower one, that means that we can rely on the ordering of the maple solutions to identify the two branches of assembly mode 2. For the **assembly mode 1** instead, we have a total of three colour (red,black,blue), i.e., there is no consistent assignment of the ordering solution to identify the solution branch.

1.8.1 Alternative resolution of the fourth-degree equations

In an attempt to describe each branch with one root, we adopted another approach. Instead of using the maple resolution, an **explicit form** resolution of the fourth-degree equation has been adopted.

Given the fourth-degree equation:

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

We can obtain the four different solutions as:

$$x_{1,2} = -\frac{b}{4a} - Q \pm \frac{1}{2}\sqrt{-Q^2 - 2p + \frac{S}{Q}} \tag{1.9}$$

$$x_{3,4} = -\frac{b}{4a} + Q \pm \frac{1}{2}\sqrt{-Q^2 - 2p - \frac{S}{Q}} \tag{1.10}$$

Where the intermediate variables are defined as:

$$p = \frac{-3b^2 + 8ac}{8a^2}$$
$$S = \frac{8a^2d - 4abcd + b^3}{8a^3}$$

$$s = 27b^2e + 2c^3 + 27ad^2 - 72ace - 9bcd$$
$$q = 12ae - 3bd + c^2$$

$$Q = \frac{1}{2}\sqrt{-\frac{2}{3}p + \frac{1}{3a}(\Delta_0 + \frac{q}{\Delta_0})}$$
$$\Delta_0 = \sqrt[3]{\frac{s + \sqrt{s^2 - 4q^3}}{2}}$$

A done for the Maple solutions, **we can assign one different colour for each of the four roots**. In particular:

- $x_1 \to$ black;
- $x_2 \to$ red;
- $x_3 \to$ green;
- $x_4 \to$ blue;

The results are shown in figure 1.9:

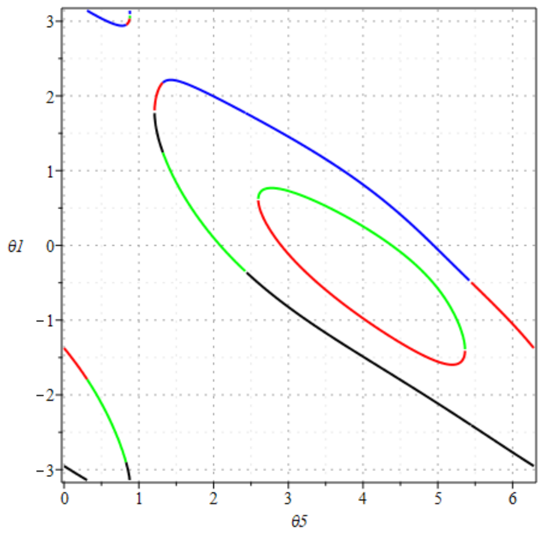


Figure 1.9: *Relationship between θ_5 and θ_1 obtained with the alternative method.*

The plot, shown in figure 1.9, is in a radians scale but is coherent with the previous one (figure 1.8). We can observe that, the **assembly mode 2** has been described by using only two colours (roots), one for the upper branch and one for the lower one, finding the same results given by the Maple resolution. For the **assembly mode 1** we find several colour alternations that do not allow us to use only one root per branch.
This method discriminates the branches even worst than the Maple resolution. For the purpose of this work we do not need to make a deeper analysis of this question and we leave it for further investigation.

Chapter 2

Designing the Links Shape

Given the relationship between the chosen input variable θ_5 and the other linkage variables, the next step, developed in this chapter, is the design of the links.
In fact, the link parameters used in the description of a mechanism are just a convenient way to specify the relative positions of the reference frames attached to the pairs connected by each link. But link parameters do not determine their precise shape, nor even the location of the joints. For example, the definition of an R pair only fixes its axis of rotation, but not the position along this axis. Thus, we are free to place each rotational joint at the position we like along its axis. Similarly, in the case of a C pair, we have also the freedom to translate its full range of displacement along the C axis.Thus, when designing the shape of each link we have to choose first the position of the joints along their axes and, then, the appropriate outline of the link to achieve this relative position.

Our **main goal** is to design the shape of each link in order to realize a full cycle of configurations for the mechanism, avoiding the collisions between them. As mentioned above we have focused on the input-output function with θ_5 as input variable. In addition we have chosen the **assembly mode 2** that allows us to describe a full range of mobility using only one solution per branch. For the following analysis, the results given by the Maple resolution, has been used.

As **secondary goal**, we have obtain link design which are simple as possible, keeping structural rigidity of the mechanism.
Since during the motion it could happen that the mechanism has intersecting trajectories, we have designed the links shape, following the next steps:

- Choose a placement of the joints, checking if they keep a minimum distance between them along the chosen assembly mode;
- Make a first design by defining each link as a straight line between the centres of consecutive joints. Then, check if there are collision between them;
- Analyse the problematic configurations and modify the shape of the links involved, in order to avoid collisions.

2.1 Positioning the joints

As a first step, we will try to determine the placement of each joint along its axis. Our first choice will be to **place each joint at the origin of the reference frame** attached to

this pair in the theoretical model.
Therefore, we will call as O_i the origin of reference frame i belonging to the pair i.

We have placed the joint , following the sequence of matrices:

$$O_i \rightarrow \overline{R_x}(\theta_i) \cdot \overline{T_x}(s_i) \cdot \overline{T_z}(a_{i,i+1}) \cdot \overline{R_z}(\alpha_{i,i+1}) \rightarrow O_{i+1}$$

The other joints are similarly placed, obtaining the results shown in figure 2.1:

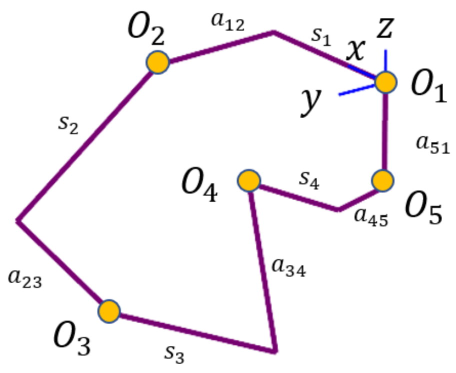


Figure 2.1: *Reference frame origins.*

The **first analysis** implies to check if two reference frame collide among the feasible configurations, if we have this several collision, we need to change the general dimensions (fixed parameters) of the mechanism.

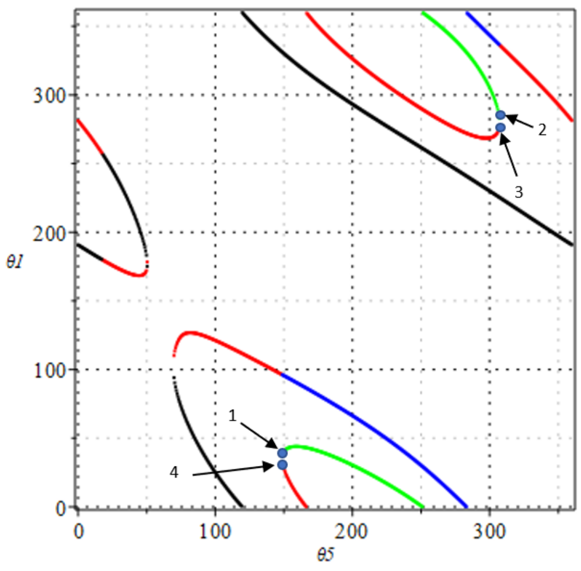


Figure 2.2: *$\theta_5 - \theta_1$ plot.*

As explained in the chapter 1.6, we can discern the four solutions given by the Maple procedures assigning different colours. The internal cycle of solutions correspond to the **green solutions** (upper solutions branch) and to the **red ones** (lower solutions branch), both for the range $148,78° < \theta_5 < 307,29°$.
Referring to the figure 2.2, starting from the **point 1** on the **green branch** (table 2.1) , it's possible to save (in a matrix) all the values of the variables until the **point 2** (table 2.2) choosing always the same root (green solution).

θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	s_1	s_2
35,11	227,74	239,60	77,72	148,78	25,51	-63,15

Table 2.1: *Configuration corresponding to the point 1.*

θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	s_1	s_2
279,96	272,54	256,76	356,58	307,29	55,98	-39,88

Table 2.2: *Configuration corresponding to the point 2.*

Once get the point 2 we can change the solution chosen, passing to the **red solutions branch**, giving the **point 3**, with the same value of θ_5,(table 2.3).

θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	s_1	s_2
279,63	272,62	256,39	357,09	307,29	56,35	-40,34

Table 2.3: *Configuration corresponding to the point 3.*

Now can we save the values coming back on the red branch, until the **point 4** choosing the red solution.It should be noted that the **point 4** have the same θ_5 value of the point 1 (table 2.4).

θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	s_1	s_2
35,08	227,72	239,61	77,81	148,78	25,53	-63,17

Table 2.4: *Configuration corresponding to the point 4.*

In this way we can obtain a matrix that has in each line the linkage variables values defining a **unique configuration** of our mechanism, and through the columns cover the internal cycle of configurations in a clockwise sense.
Obtained this matrix, knowing the linkage variable values, for of each line we can calculate the homogeneous matrices that describe the poses of each reference frame. These homogeneous matrices are expressed, in the following lines, with the notation of ${}^m\overline{O}_n$ that describe the *homogeneous matrix that describe the pose of the reference frame n with reference to the m one.*

$$
\begin{aligned}
{}^1\overline{O}_2 &= \overline{R_x}(\theta_1) \cdot \overline{T_x}(s_1) \cdot \overline{T_z}(a_{12}) \cdot \overline{R_z}(\alpha_{12}) \\
{}^1\overline{O}_3 &={}^1 \overline{O}_2 \cdot \overline{R_x}(\theta_2) \cdot \overline{T_x}(s_2) \cdot \overline{T_z}(a_{23}) \cdot \overline{R_z}(\alpha_{23}) \\
{}^1\overline{O}_4 &={}^1 \overline{O}_3 \cdot \overline{R_x}(\theta_3) \cdot \overline{T_x}(s_3) \cdot \overline{T_z}(a_{34}) \cdot \overline{R_z}(\alpha_{34}) \\
{}^1\overline{O}_5 &={}^1 \overline{O}_4 \cdot \overline{R_x}(\theta_4) \cdot \overline{T_x}(s_4) \cdot \overline{T_z}(a_{45}) \cdot \overline{R_z}(\alpha_{45})
\end{aligned}
$$

Only for the cylindrical pairs, we need to place a joints that, as a sliding components, can perform displacements (s_2 and s_4), in order to have a fixed length value of the link that collect the cylindrical pair to the next one.
As example, for the reference frames O_2 and O_3 we can define a displacement until a point O_{2V}. Therefore, starting from the point O_{2V}, we have a fixed length link between O_{2V} and O_3. The same goes for O_4. Analytically, the pose of the new two points O_{2V} and O_{4V} can be defined with the coordinates that came from the first three element of the fourth column of the matrices:

$$^1\overline{O}_{2V} =^1 \overline{O}_2 \cdot \overline{R_x}(\theta_2) \cdot \overline{T_x}(s_2) \qquad\qquad ^1\overline{O}_{4V} =^1 \overline{O}_4 \cdot \overline{R_x}(\theta_4) \cdot \overline{T_x}(s_4)$$

Taking the first three elements of the fourth column of each matrix, we can define the positions of each reference frame , given the set variable with which the homogeneous matrices have been calculated.
If we compute the homogeneous matrices for all the set values corresponding the internal cycle of configurations, we obtain the **trajectories** of each reference frame. The results are shown in the following images.

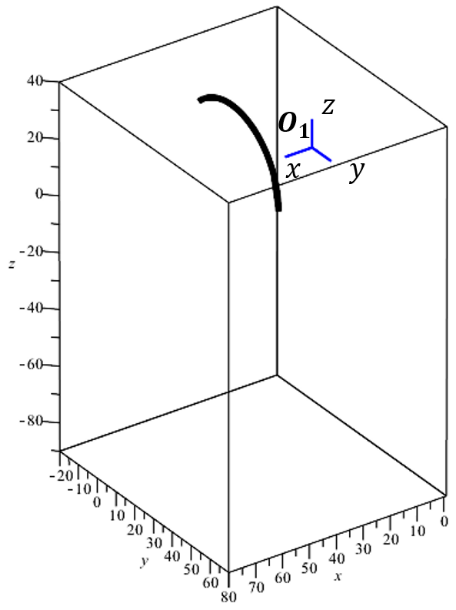


Figure 2.3: *Trajectory of O_2 wrt the O_1.*

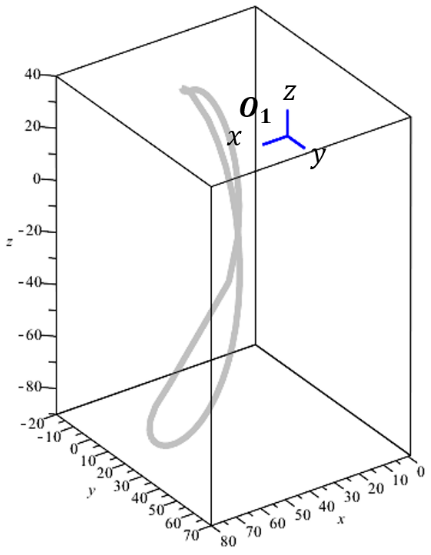


Figure 2.4: *Trajectory of O_{2V} wrt the O_1.*

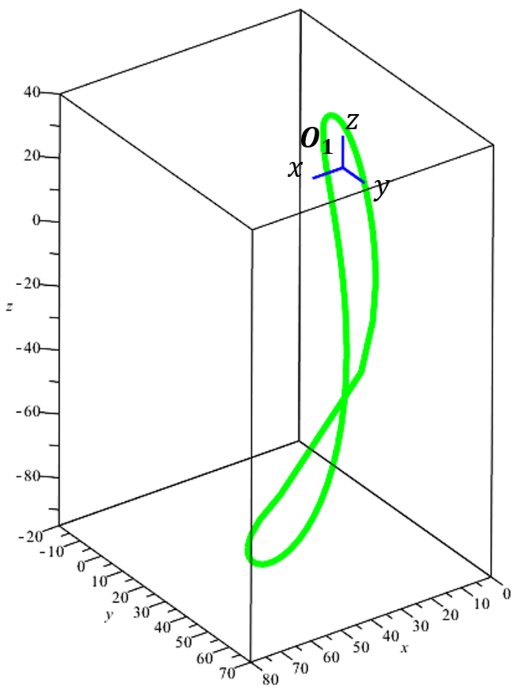


Figure 2.5: Trajectory of O_3 wrt the O_1.

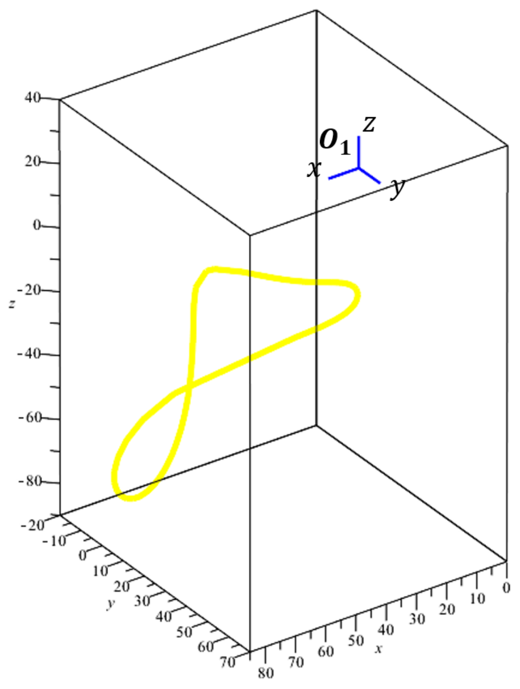


Figure 2.6: Trajectory of O_4 wrt the O_1.

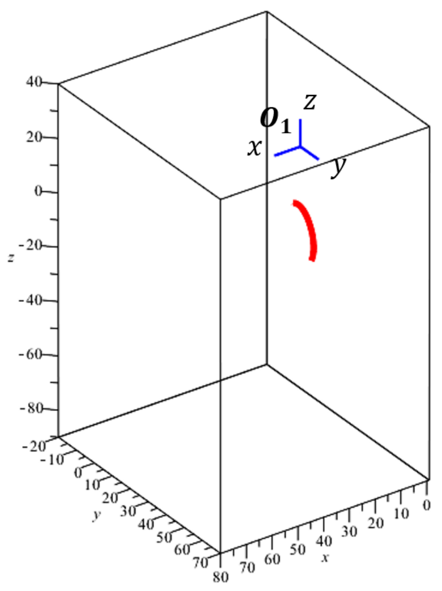


Figure 2.7: Trajectory of O_{4V} wrt the O_1.

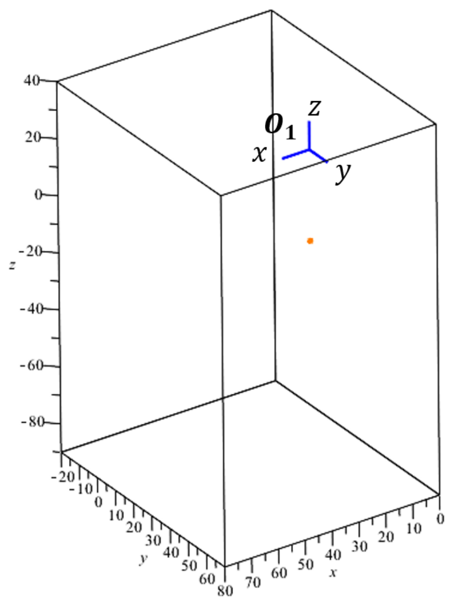


Figure 2.8: Trajectory of O_5 wrt the O_1.

In general, is very difficult to predict these trajectories shape but, we can see that the O_2 (figure 2.4) describes an arc because, it has moved by the rotational pair $(\overline{R_x}(\theta_1))$ instead, the other parameter are fixed. In the same way O_5 (figure 2.8) is described with a single point because, if we fix O_1, the reference frame O_5 can change the orientation in the space, but has always the same position in space wrt O_1. The other reference frame (O_3 and O_4, in figure 2.5 and 2.6) are combination of the other parameters.
With an easy check we can say that there aren't collisions between the reference frame and the first one, as shown in figure 2.9:

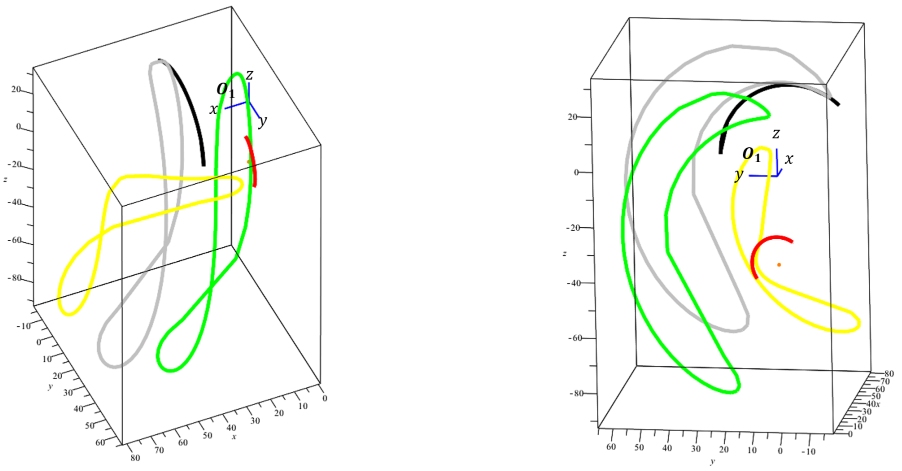


Figure 2.9: All the trajectories of the reference frames wrt O_1.

In the figure 2.9 the trajectories black (O_2) and grey (O_{2V}) collide in one point, moreover, is not a collision, but when the displacement s_2 acquire value of zero.

It should be clear that the analysis of collisions detection between reference frames with each other is not enough because, can not guarantee the absence of collision among the links.

2.1.1 First design: straight links

The first attempt to design the other links is a **straight line between the reference frames center**. As mentioned above, O_1 has been chosen as global reference frame, therefore also the reference frame O_5 is fixed in the space, because its position does not change, therefore, we can draw the link between as shown in figure 2.10:

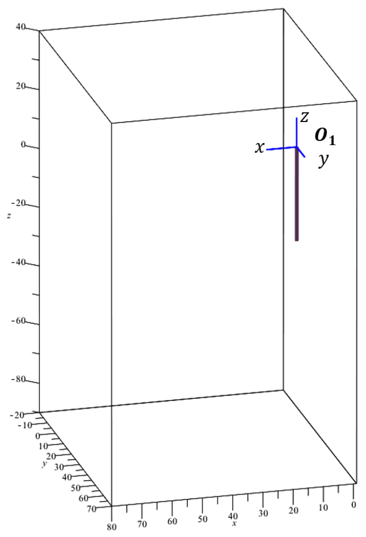


Figure 2.10: *Link O_5 to O_1.*

Following the same process, we can draw the other links, collecting the reference frames with a **straight line**.
For this analysis we need to considerer two additional "segments" because, we need to simulate the cylindrical pairs with a part that can slides respect of the zero of the displacement. Therefore, we need to add two straight lines that simulate the straight guidelines for the cylindrical pairs, each one with the length of the respective maximum displacement.

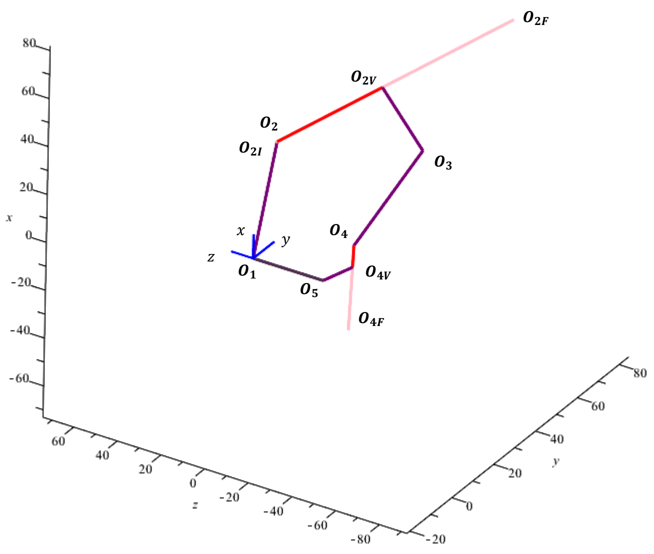


Figure 2.11: *RCRCR mechanism, wire frame representation, in a generic instant.*

As shown in figure 2.11, for the reference frames O_2 and O_3 we can define a red displacement until a point O_{2V} that simulate the active displacement (in that mechanism configuration) and the pink part that simulate a straight guide line that allows the maximum displacement for the cylindrical pairs.

The points that allow us to define the necessary length for straight guidelines for each cylindrical pair, can be obtained analysing the minimum and the maximum values of each cylindrical displacement:

$$\begin{array}{ll} s_{2_{MIN}} = -0.48 & s_{4_{MIN}} = -90.57 \\ s_{2_{MAX}} = 93.36 & s_{4_{MAX}} = -5.97 \end{array}$$

Therefore the other new points, used to define the straight guidelines, are given by the first three component of the fourth column of the following matrices:

$$\begin{array}{ll} {}^1\overline{O}_{2F} = {}^1\overline{O}_2 \cdot \overline{T_x}(Max(s_2)) & {}^1\overline{O}_{4F} = {}^1\overline{O}_4 \cdot \overline{T_x}(Min(s_4)) \\ {}^1\overline{O}_{2I} = {}^1\overline{O}_2 \cdot \overline{T_x}(Min(s_2)) & {}^1\overline{O}_{4I} = {}^1\overline{O}_4 \cdot \overline{T_x}(Max(s_4)) \end{array}$$

It's been decided to start the straight guide line for s_4 from the point O_4, instead from O_{4I} (drop between O_4 and O_{4F}), extending the straight guide line.
It's must be clear that in the figure 2.11 all the points are indicated as O_*, including the reference frame centres and the other points (like O_{2F},O_{4F}) that are not reference frame but are only remarkable points. Referring to the figure 2.11, to summarize:

- O_1, O_2, O_3, O_4, O_5 reference frame centres;
- O_{2V}, O_{4V} position of the sliding component of the cylindrical pair;
- O_{2F}, O_{2I}, O_{4F} defining point for the straight lines, simulating the guidelines for the cylindrical displacements.

2.1.2 Notation

As explained above, we need to check the collision between the five links and the two straight cylindrical guide line.
In the following chapters will be used the notation:

- l_{n-m} to indicate the link between the reference frame centres n and m;
- d_{p-q} to indicate the straight line defined by the extreme points p and q.

2.2 Collisions detection

Verified that there are **no collisions** between the reference frames and **once designed the shape for each link**, the next step consist in the **collisions detection**.
In order to see if the proposed design will allow the mechanism to reach the full cycle of configurations of the **second assembly mode** moving continuously without interferences between links, we will perform a collision detection process.

This process involves sampling the input variable with a sufficiently fine discretization and computing the whole configuration of the mechanism for each sampling value. Knowing the values of all variables in a given configuration, allows us to determine the position of each rectilinear segment associated to each link and check if there is any collision between them. The collision tests must be carried out for each pair of non-consecutive links, since the interference between consecutive links will be dealt with at the phase of design of the mechanical construction of each joint. However, each cylindrical pair includes also a sliding guide to allow the displacement of the joint along it and must be also included for checking collisions with the links (d_{2I-2F} and d_{4-4F}).

Given all the links and the two displacements, the algorithm to detect the collision provide to **treats them as segments** in a 3D space.
For each point of the assembly mode 2, the algorithm computes instantaneous configurations of the mechanism, i.e. the positions of each reference frame through the matrices in the chapter 2.1.
Given two links the algorithm, proceeds through two steps to detect a collision:

- First step: check if they are in the same plane;
- Second step: check if these two link in the same plane, are crossing or not.

2.2.1 First step: test for coplanarity of segments

Given two segments in the space, as shown in figure 2.12:

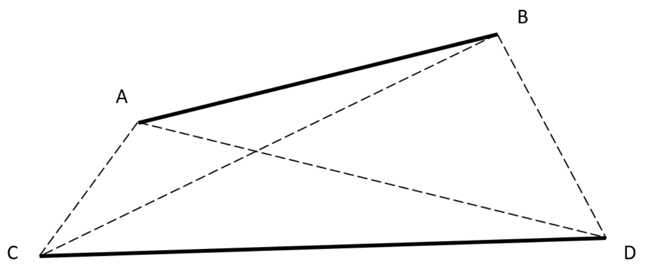


Figure 2.12: *Two generic segments in the space.*

We consider the **tetrahedron** defined by the four points $ABCD$. The volume of this tetrahedron is proportional to the absolute value of the determinant of the *matrix M*.
When the segments are coplanar,$det(M) = 0$. When the segments are not coplanar, the sign of $det(M)$ depends on the sign of the triple product of the 3 vectors going from A to B, C, and D, respectively. This can be also visualized as depending on whether one of the vertexes lies on one side or another of the plane determined by the other three vertexes [14].

$$M = \begin{bmatrix} A_x & A_y & A_z & 1 \\ B_x & B_y & B_z & 1 \\ D_x & D_y & D_z & 1 \\ C_x & C_y & C_z & 1 \end{bmatrix}$$

Monitoring this determinant we can observe that, if this changes in sign means that the segments have passed through the condition in which the determinant is equal to zero and the segments are on the same plane.

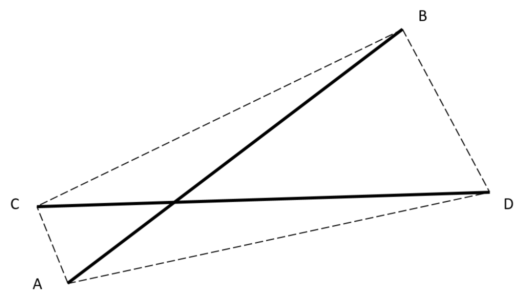


Figure 2.13: *Crossing segments.*

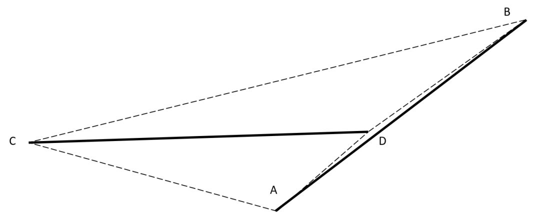


Figure 2.14: *No crossing segments.*

This algorithm in not enough to detect a collision between two segments because we can have the following conditions:

- two segment that are crossing, as shown in figure 2.13;
- two segment that are not crossing, as shown in figure 2.14, even if, they are on the same plane.

Therefore, we need to an additional step.

2.2.2 Second step: test for interference between coplanar segments

The situation in which two segments are coplanar does not imply that the two segments interfere (figures 2.13 and 2.14). We need to differentiate those situations in which the segments are separated even being coplanar.
If the first step detects a changing in sign of the determinant and if we analyse the two segments in the frame in which a change of the determinant sign has been detected, with an approximation, we can analyse them in a 2D space.
In fact, if we were in a plane we could have:

- Crossing segments in a plane;
- No-Crossing segments in a plane;

Crossing segments in a plane

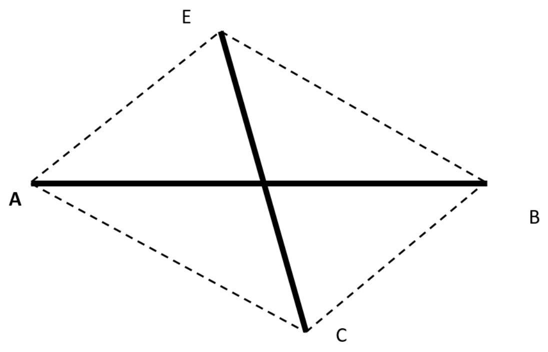


Figure 2.15: *Crossing segments in a plane.*

As shown in figure 2.15 for two crossing segments we can consider four different triangles areas, shown in figure 2.16:

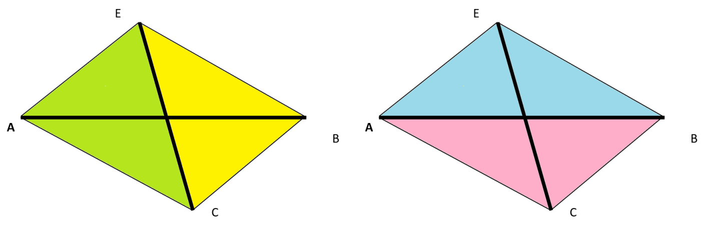


Figure 2.16: *The four different area for crossing segments in a plane.*

We can observe that, the areas, follow the relationship:

$$\overset{\triangle}{AEC} + \overset{\triangle}{BEC} = \overset{\triangle}{AEB} + \overset{\triangle}{ACB} \tag{2.1}$$

No-Crossing segments in a plane

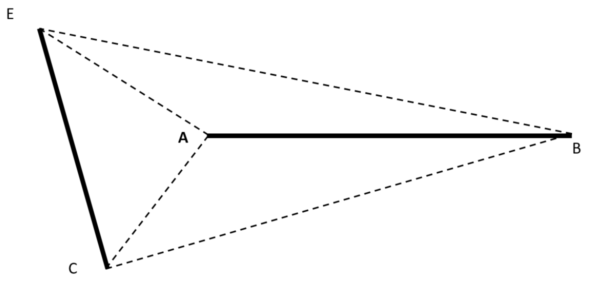


Figure 2.17: *No Crossing segments in a plane.*

As shown in figure 2.17 for two no-crossing segments we can consider four different triangles areas, shown in figure 2.18

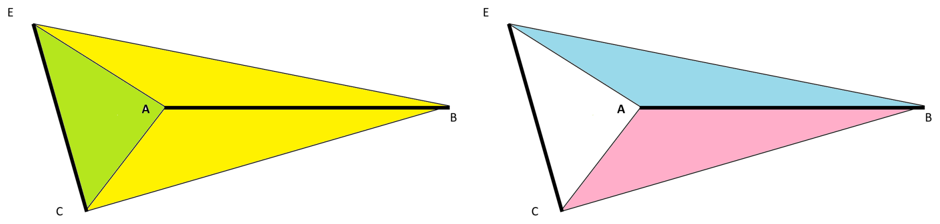


Figure 2.18: *The four different area for no-crossing segments in a plane.*

We can observe that, the areas follow the relationship :

$$\overset{\triangle}{AEC} + \overset{\triangle}{BEC} \neq \overset{\triangle}{AEB} + \overset{\triangle}{ACB}$$

2.3 Algorithm implementation

Using a step of $0,1$ rad,our algorithm to detect a collision between two links is based in **monitoring the sign of the determinant** formed with the corresponding segments along a continuous path along all the cycle of the assembly mode. If a change of sign is detected, it is the indication that a zero crossing has occurred and therefore the two links passed through a coplanar configuration that may give rise to a collision.

Is this case we check with the **second test** if the collision has actually taken place. A certain tolerance in the condition of this test is required, since we can only reach a configuration where the segments are approximately coplanar, so that condition (2.1) will not be exactly fulfilled. So we can set a tolerance ε ($\varepsilon = 0.5$) , small enough, to detect that the difference of the areas produce a collision.:

$$(\overset{\triangle}{AEC} + \overset{\triangle}{BEC}) - (\overset{\triangle}{AEB} + \overset{\triangle}{ACB}) = \varepsilon$$

The Maple files of the algorithm implemented are shown in Appendix D.

2.4 Test results: detected collisions

For the **assembly mode 2**, with the algorithm implemented in Maple (Appendix D), it has been obtained the following interesting results. The **first step** of the algorithm has detected the determinant changes of sign, shown in table 2.5:

Branch	**Couple of segments**	θ_5 (deg)
Green	$l_{5-1} \leftrightarrow d_{4-4F}$	183,34
Green	$l_{2V-3} \leftrightarrow l_{4V-5}$	189,07
Green	$l_{1-2} \leftrightarrow l_{2V-3}$	194,80
Green	$l_{1-2} \leftrightarrow l_{2V-3}$	217,72
Green	$l_{1-2} \leftrightarrow d_{4-4F}$	234,91
Green	$l_{1-2} \leftrightarrow l_{4V-5}$	252,10
Green	$d_{2I-2F} \leftrightarrow l_{5-1}$	303,66
Red	$l_{2V-3} \leftrightarrow l_{4V-5}$	303,66
Red	$d_{2I-2F} \leftrightarrow l_{5-1}$	240,64
Red	$l_{1-2} \leftrightarrow l_{2V-3}$	183,34
Red	$l_{1-2} \leftrightarrow d_{4-4F}$	183,34
Red	$l_{3-4} \leftrightarrow l_{4V-5}$	183,34
Red	$l_{5-1} \leftrightarrow d_{4-4F}$	177,61
Red	$l_{1-2} \leftrightarrow l_{4V-5}$	166,15
Red	$l_{1-2} \leftrightarrow l_{2V-3}$	148,96
Red	$l_{3-4} \leftrightarrow l_{4V-5}$	148,96

Table 2.5: *Determinant changes of sign detected.*

In the table 2.5, the column *"Branch"* indicates, referring to the figure 2.2, which branch the detection belongs to. Therefore, with the *angle* θ_5, on the last column, we can complete define the point in which the collision has been detected.
As we can notice, the algorithm detect a lot change of sign of the determinant, these are due to the change of the relative position of the segments among the motion. As mentioned above, the first step does not guarantee the collision detection, but only that the two segment analysed are (with an approximation) in the same plane.
Therefore, the algorithm provide, for each of the frame detected in table 2.5, to check if this two segment are crossing (chapter 2.2.2). The results obtained are shown in table 2.6

Branch	**Collision number**	**Collision detected**	θ_5 (deg)
Green	1	$l_{5-1} \leftrightarrow d_{4-4F}$	183,34
Green	2	$l_{1-2} \leftrightarrow l_{2V-3}$	194,80
Green	3	$l_{1-2} \leftrightarrow l_{2V-3}$	217,72
Red	4	$l_{5-1} \leftrightarrow d_{4-4F}$	177,61

Table 2.6: *Collisions detected.*

As shown, only four of the determinant changes of sign produce a collision. In particular, in table 2.6, are show the branch belongs to and the angle θ_5, that define the point on the internal cycle of configurations (figure 2.2).

The **first collision**, is shown in the figure 2.19:

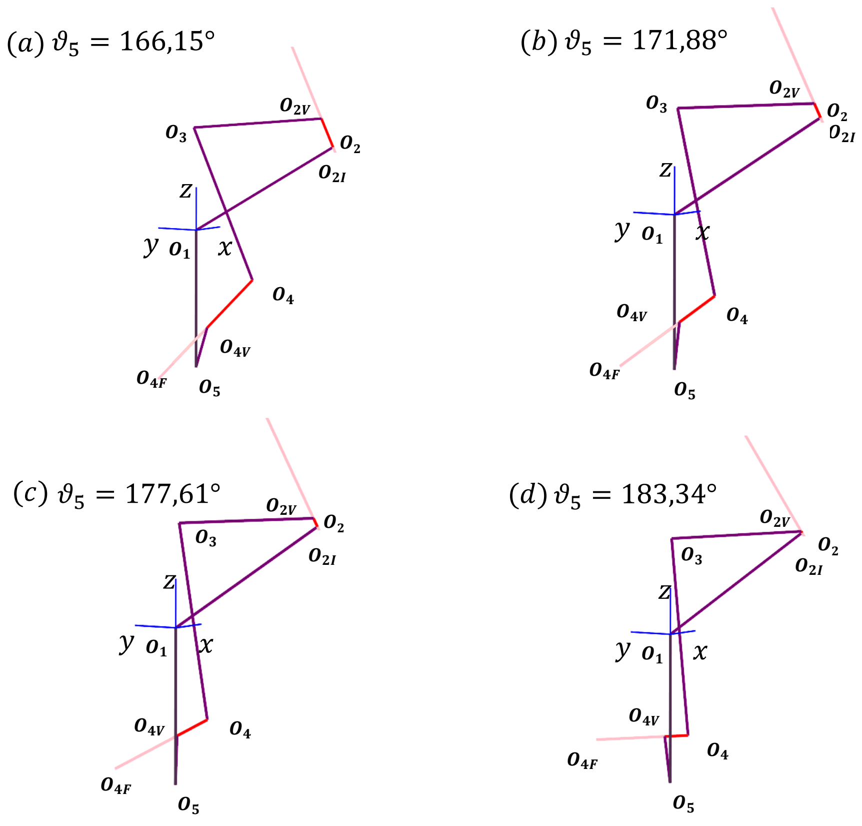
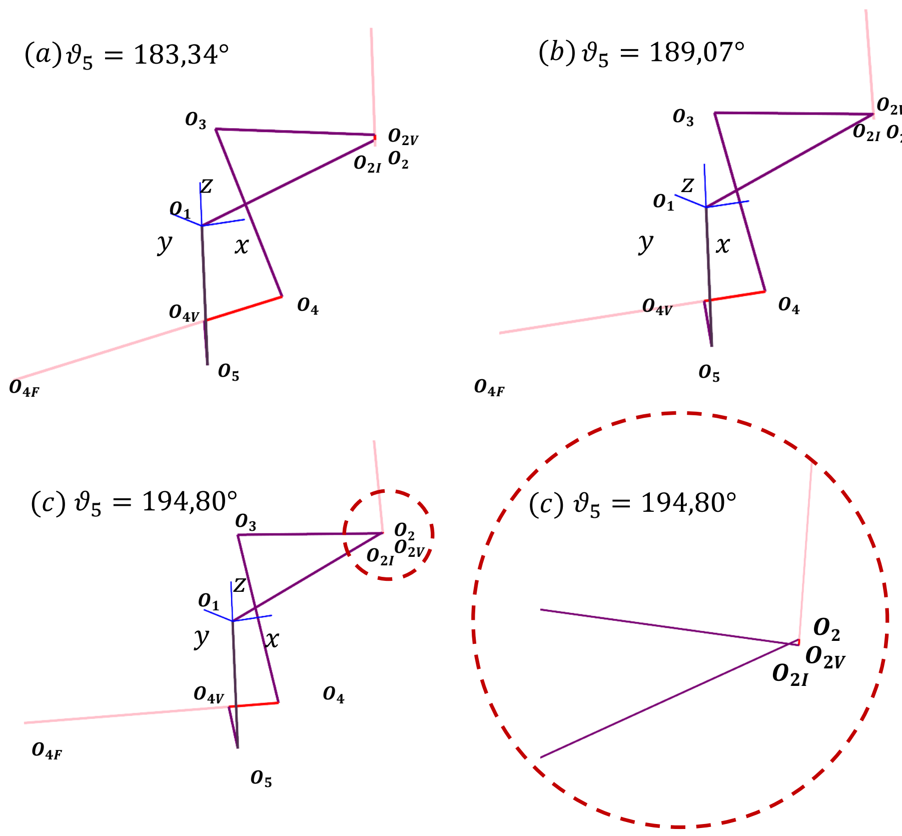


Figure 2.19: *Collisions 1,between the links* $l_{5-1} \leftrightarrow d_{4-4F}$.

It has been noticed that, the collisions 1 and 4, regard the same kind of mechanism configurations, i.e., when the link l_{5-1} , during the motion, it is crosses with the d_{4-4F}. For the sake of brevity, is shown in figure 2.19, only the four frame around the first collision, given that, the collision 4 present the same relative positions between the links involved. In order to avoid the collision, it's been chosen to re-design the link l_{51}.

The collisions 2, is shown in the figure 2.20:



$(a)\vartheta_5 = 183{,}34°$
$(b)\vartheta_5 = 189{,}07°$
$(c)\ \vartheta_5 = 194{,}80°$
$(c)\ \vartheta_5 = 194{,}80°$
O_3
O_{2V}
O_{2I}
O_2
O_1
Z
y
x
O_{4V}
O_4
O_5
O_{4F}

Figure 2.20: *Collisions 2, between the links* $l_{1-2} \leftrightarrow l_{2V-3}$.

The collision 2 regard the link l_{2V-3} and link l_{1-2}, are detected because of the values of the displacement s_2, that pass to a positive value to a negative one. In this condition the two links have a point in common (the origin of the reference frame O_2), therefore coplanar. The collision number 3 regards the same kind of collision, but in this case when s_2 pass from a negative value to a positive one. Also in this case, for the sake of brevity, are shown in figure 2.20, only the four frame around the collision number 2.

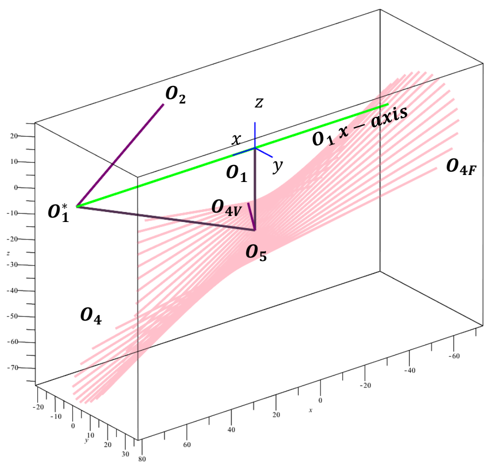
How we will see the easier way to avoid this collision is re-design the link l_{1-2}.

2.4.1 Second design: avoiding collisions

In order to avoid the collisions, changes have been made to the link l_{1-2} and link l_{5-1}. Until now, to make each link we have used straight lines, connecting the origins of the reference frames but we can also connect the link reaching a point on the x-axis of the joint as shown in the following chapters.

2.4.1.1 Avoiding collisions between l_{5-1} and d_{4-4F}

In order to avoid the collisions linked to the link l_{5-1}, shown in the chapter 2.4, we can *"move"* the the origin O_1 along his axis without affecting the mechanism overall functioning. In fact O_2 it's still defined with reference to the oldest O_1 and a rotation about the new origin do not change the effects on the reference frames positions. As shown in figure 2.21 the origin O_1 along the O_1 $x-axis$ has been moved as much as necessary to avoid collisions.



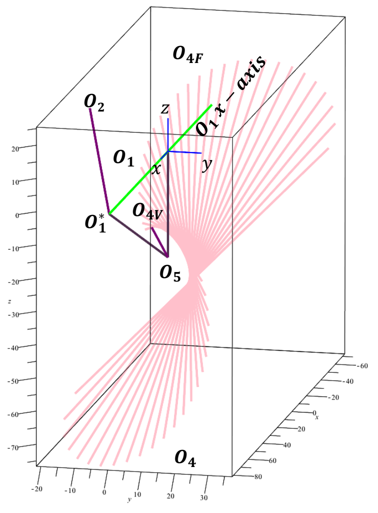


Figure 2.21: *Changes to the link* l_{5-1}.

In particular, to chose the value of displacement on the *x-axis* (green line, figure 2.21) that guarantee the minimum length of the link l_{5-1} and at the same time, avoids the collisions. All the position of the s_4 straight guideline has been plotted, shown with the pink lines, in the figure 2.21, in order to define a point on the green line, so that, connecting that to O_5, does not produce collisions. The joint center (point O_1^*) has been placed 80 units far, positive values on x axis, from the oldest one, so that, the segment between O_5 and O_1^* does not collide in any frame with d_{4-4F}. Change the point O_1 implies to change also the link l_{1-2}, since O_1 has been displayed, we can connect O_1^* with O_2.
For the chosen values the new link l_{5-1} **does not detect collisions**.

2.4.1.2 Avoiding collisions between l_{1-2} and l_{2V-3}

As done for the center O_1 instead to connect the link l_{1-2} from the point O_1 to O_2, we connect the link to a point on the x-axis of the reference frame O_2 displaced to 5 unit on the negative values. To chose this value, we need to analyse the extremes of the s_2 displacement. This displacement s_2 goes from a value of $-0,49$ to a value $93,36$, that means, we can make more longer the straight guidelines to collect the link l_{1-2} to a point outside the displacement (point O_2^*), reaching a point 5 units far from the centre O_2, in the negative values on the *x-axis*. It's should be clear that we need to displace also the point O_{2I} (that define an extreme of a cylindrical guideline), making it coincident with the new point O_2^*.

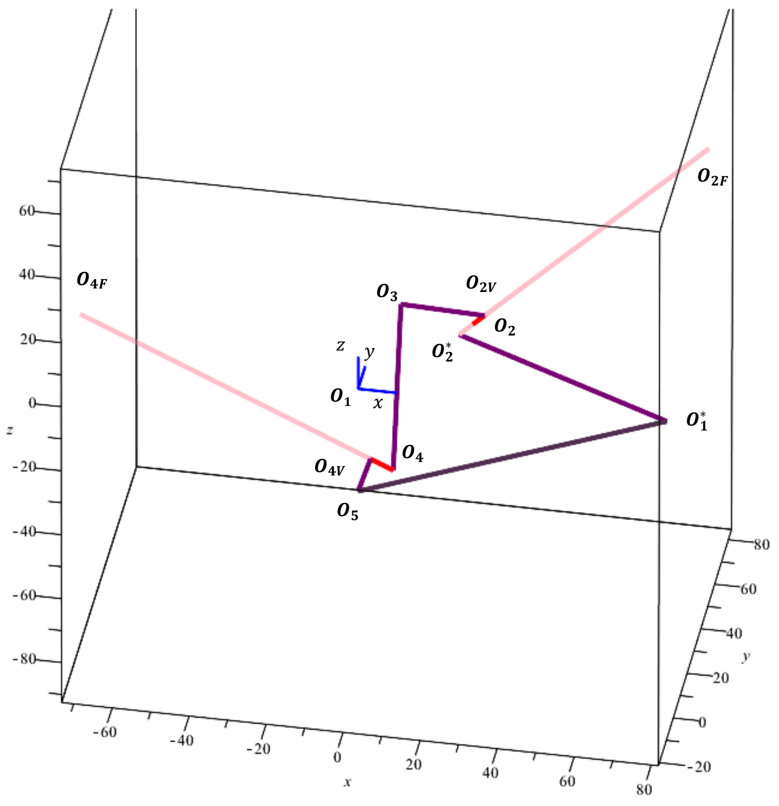


Figure 2.22: *Changes to the Link 12.*

In this way when the link l_{2V-3}, that have a point on the straight s_2 guidelines pass from the positive value to a negative one, does not crosses the link l_{1-2}, as shown in figure 2.22. As mentioned before this displacement of the reference frame O_2 does not changes the effects on the other link and avoid the collisions..
For the chosen values the new link l_{1-2} **does not detect collisions**.

2.5 Implementation of the joint envelope

Until now we have considered the joint as a points in the space but, before to design the CAD model, it's necessary a preliminary analysis on the **joint envelope**. In order to simulate each pair as a cylinder, in the following analysis, each pair will be simulate as a straight line, in the direction of the respective *x-axis*, of 5 unit (coherent with the general proportions of the mechanism).

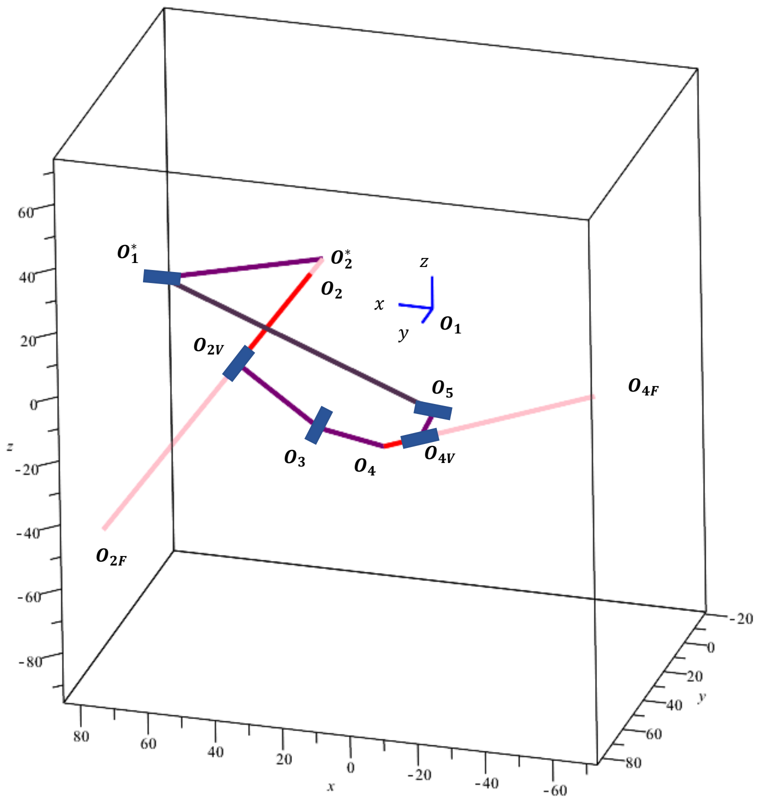


Figure 2.23: *Simulation with the joint envelopes.*

As shown in figure 2.23, the straight blue lines simulating the pairs will be treated as segments in the algorithm shown in the section 2.2.For the collision detection involving the pair we need to consider a tolerance ε bigger (setted to a value of 20), because of the bigger envelope of the pair with reference to a link.

At the end with the algorithm 2.2, has been checked, the possible collision between each pair with each other and with the other links, with the **final result of no collisions detected**.

Chapter 3

Construction of the Mechanism

Once we have determined the basic shape of links that allow the collision-free motion of the mechanism, the last step involves the precise definition of the different parts needed to assemble it.
Except for the commercial components (bearings, motors, aluminium bars, screws, bolts, ...), all the necessary parts will be made of *ABS-P430 plastic* using the 3D printers available in the lab of the research center.

Starting from the Maple model of the RCRCR mechanism it has been created a **CAD model** of the mechanism ,through the use of Solidworks$^{®}$, in order to realize each part with a 3D printer.
First, it is necessary to determine the **scaling factor** of the real mechanism **the scale** of the mechanism, which must be chosen taking into account the size of the commercially available joint components, as bearings and motors, as well as the precision and maximum dimension limitations of the 3D printers available at the lab. Thus, each link and joint has to be precisely reshaped to include all the elements necessary to fix the different parts of the joints with the links.

3.1 Selection of the scaling factor

The theoretical analysis of our mechanism has been done for arbitrary length dimensions, describing it with undefined unit lengths. For our physical implementation, we have chosen to **take the unit length to have the value of** 6 **mm**, resulting in the dimensions for the link parameters shown in table 3.1.

$i =$	**1**	**2**	**3**	**4**	**5**
α_{ij} (deg)	60	45	35	30	10
a_{ij} (mm)	150	180	240	60	192
s_i (mm)	180	/	150	/	0

Table 3.1: *Fixed parametres scaled.*

Where $j = i + 1$ and by convention, when $i = 5 \rightarrow j = 1$. This choice has been motivated by the following conditioning factors:

- **Joint size**: Each joint must include a couple of commercially available bearings providing the necessary robustness to the rotational coupling;

- **Material strength/rigidity**: the characteristics of the plastic material used imposes restrictions about the length and thickness of the links.
- **3D printer precision**: the accuracy in the construction of each part must be compatible with the maximum precision available to the printers;
- **3D printer maximum printable length**: There is a limitation in the size of the parts that can be printer in each printer;
- **Maximum displacement allowed by the cylindrical pairs**: The cylindrical pairs involve the longest parts of the mechanism. They will be implemented with commercially available linear bearings whose possible dimensions are restricted by the supplier [7] ;
- **Motor power**: the mechanism masses must be low enough to allow it to be driven by reasonably powerful motors.

Taking into account all these factors, a length unit of 6 mm has been chosen, that complies with all the requirements, as will be explained in detail in the next sections.

3.2 Used equipment

In order to print the CAD parts the following 3D printers have been used [13]:

Company	3D printer	Build size	Model material	Layer Thickness
Proto3000	1200es Series	254 x 254 x 305 mm	ABSplus-P430	0,330 mm
Stratasys	uPrint SE	203 x 152 x 152 mm	ABSplus-P430	0,254 mm

Table 3.2: *3D printers used.*

Therefore all the parts printed will be respect the maximum printable dimensions.
For the mechanical features of the ABS-P430 reference is made to the company website [13].

3.2.1 Printing guidelines

Both printers use the FDM^{TM} Technology (fused deposition modelling), therefore, they build the 3D model and its support material, layer by layer, from the bottom to the top on a removable modelling base.
The accuracy on the **horizontal plane** is 0.1 mm and the mechanical strength in this plane is bigger than in the other planes. Since we need to print components that require accurate features, we need to take into account the printing capabilities in the design of each model, making compromises among the optimization of the design, the limit and capability of 3D printer and the functioning of the mechanism.
The quality of a part can change by only adapting the design. Considering the printing of the part shown in figure 3.1, we can see a **non optimized printing orientation**.

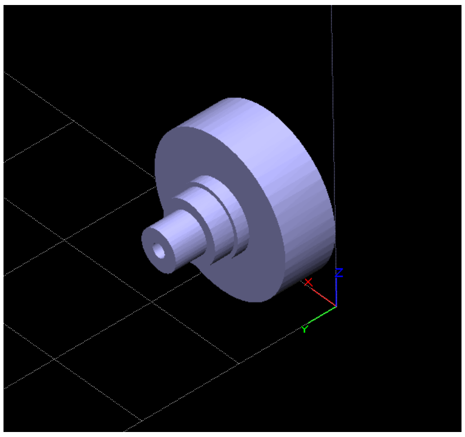


Figure 3.1: *CAD part, in a non optimized printing orientation.*

A software (CatalystEx®) provides to transform the CAD model (*part.sdlprt*), in a model built layer by layer as shown in figure 3.2:

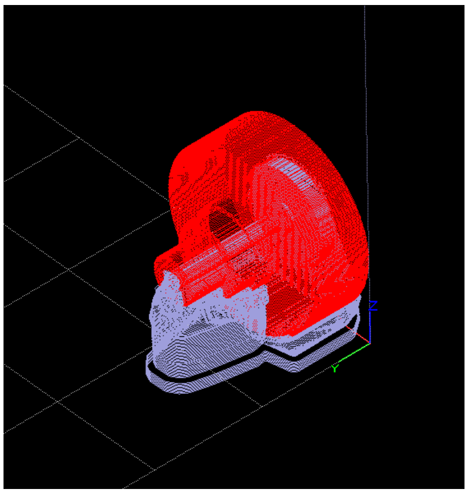


Figure 3.2: *Layer by layer model, in a non optimize printing orientation.*

If we print with the orientation shown in figure 3.2, we will obtain a bad quality of the surface of each cylinder and of the hole, in fact, we have more accuracy on the *plane xy* than on the *planes zx* and *zy* and the circle of the cylinder will be approximated with the lower precision of the layer thickness. Moreover, this design produces a support material wast.
An **optimized printing orientation** is showed in figure 3.3.

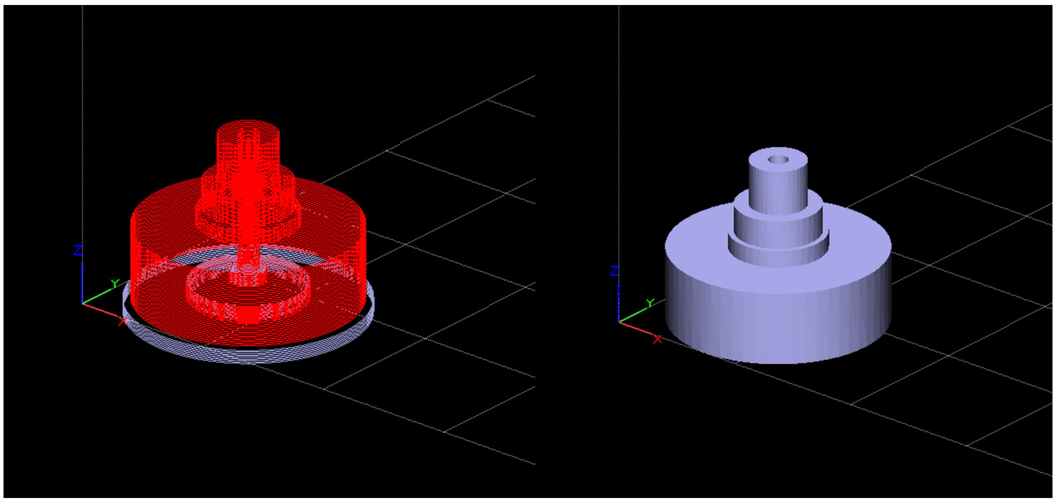


Figure 3.3: *Model in a optimized printing orientation.*

With this orientation, we can obtain a good shape of the cylinder because we are printing it this perpendicularly to the horizontal *plane xy*. As shown in figure 3.3 the layers guarantee a good accuracy of the cylindrical shape. Thus, the previous guideline should be taken into account if we are printing functional surfaces, such as the bearing housing.
Another guideline is related to the material mechanical strength. In particular it has been impossible to used printed links with cylindrical shape, because, using ABS the link will flex losing the position accuracy required by the mechanism.

3.3 Generic rotational joint model

The general idea for a rotational pair is to design a joint that allows a relative rotation in this reference frame. This feature can be obtained with the bearings that perform a relative rotation between two parts. The design made is shown in figure 3.4.

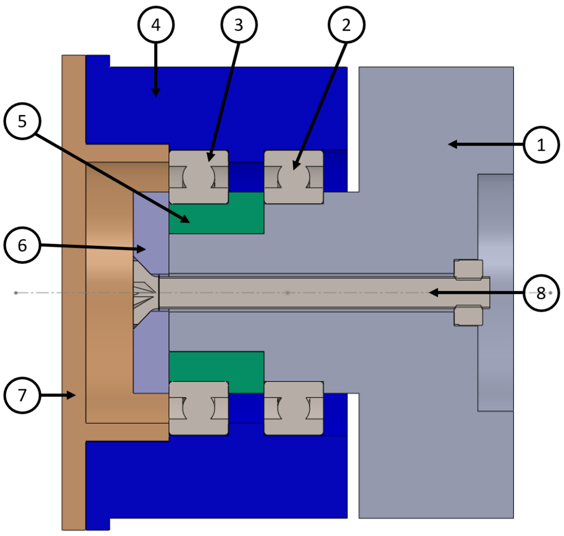


Figure 3.4: *Rotational pair, section view.*

Referring to the figure 3.4, we have a local **Fixed Part** (1) that includes the bearings shaft. As we noticed, two bearings have been used in order to make the system more rigid and to avoid the relative misalignement between the parts. Two bearing (2-3), **SKF-radial ball bearing-61802** [12], connect (1) to the **Mobile Part** (4) that includes the housing seats for the bearings, allowing the relative rotation with (1). We have chosen, considering our purpose, to block the bearing (2) on 3 points and the bearing (3) on 4 points. The contact points are obtained thanks to a shaft abutment shoulder, housing shoulders, the **Spacer** (5), the **End Plate** (6) and the **Top** (7). The end plate (6) forces the contact between the shoulders and the bearings using an **M3-countersunk flat head cross recess screw** (8).

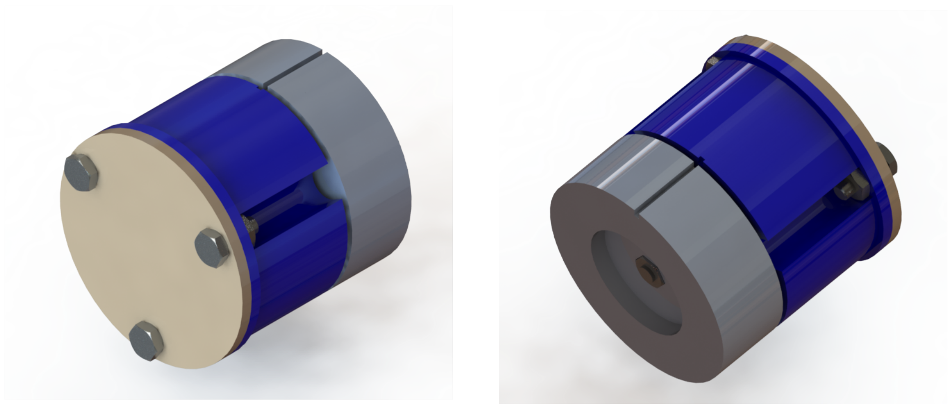


Figure 3.5: *Rotational pair, isometric views.*

It should be specified that the driver for the choice of each component size, was the minimum printable shaft diameter. In fact, given that, it should be possible to choose the bearings and design the other parts.

Assembly Procedure To assemble each rotational pair the following steps had been followed:

- assemble the bearing (2) on its the respective housing, inside the mobile part (4) and place them on the shaft on the fixed part (1) until the shaft shoulder;
- insert the spacer (5) until the shoulder of the shaft and insert the bearing (3);
- insert the screw (8) with the end plate (6) and force the contact with the use of a nut;
- insert the top (7) that makes the four contact points and fix them with the use of nuts and screws, as shown in figure 3.5.

3.4 Generic cylindrical joint model

The cylindrical joints should perform a rotation and a displacement with the minimum possible friction. In order to perform the features required we have used a **Igus-DryLin ®Linear Plain Bearings-RJUM-01-12** and **Igus-DryLin ®shaft-AWMP-12** (shown in figure 3.6) [7].

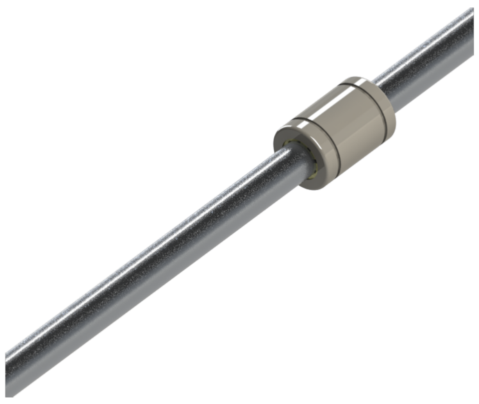


Figure 3.6: *Cylindrical pair, Igus-DryLin ®Linear Plain Bearings-RJUM-01-12* and Igus-DryLin ®shaft-AWMP-12 [11].

The length of the Igus-DryLin ®shaft (600 mm) allows to perform the displacements provided by the theoretical analysis. In particular the longer displacement is given by the joint J_2, that need to display $93,85$ units, that with the scale chosen is $563,07$ mm.
The elements chosen guarantee low frictions and low weight. In order to use these linear bearings in the mechanism, we need also to connect this to the neighbouring joint. Therefore a cover jointed with the cylindrical linear bearing has been designed.

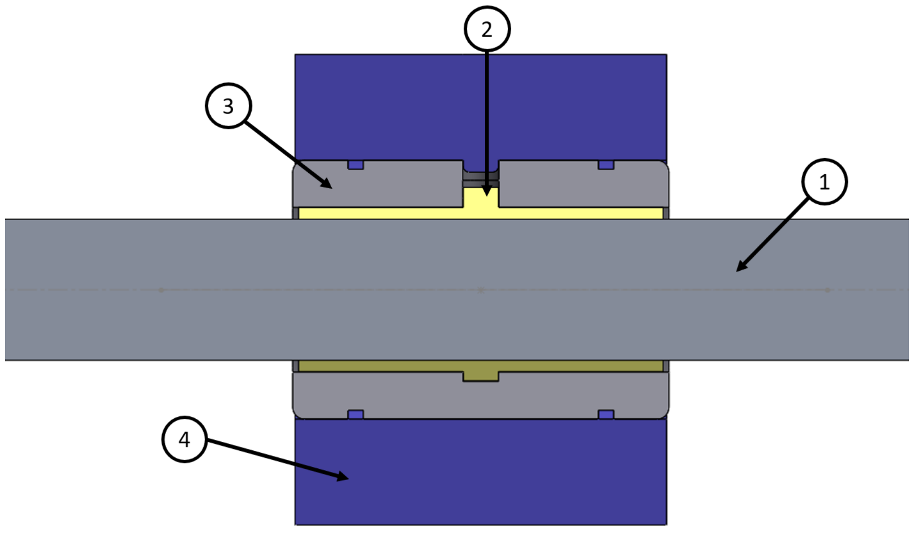


Figure 3.7: *Cylindrical pair, section view.*

As shown in figure 3.7, the cylindrical joint consisted in a precision aluminium **shaft** (1), linear plain bearings (2)(made from solid polymer) mechanically fit into an anodized **aluminium adapter** (3) and the **cover** (4).

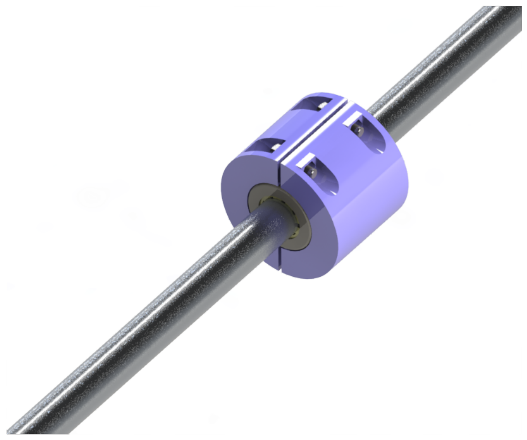


Figure 3.8: *Cylindrical pair, isometric view.*

The cover (4) is composed by two parts that force the mate with the aluminium adapter (3) through the use of $M3$ nuts and $M3 \times 6$ screws, as shown in figure 3.8

Assembly Procedure The Igus-DryLin ®Linear Plain Bearings-RJUM-01-12 is sold as shown in figure 3.6, therefore, to build the joint, it is sufficient to place a half part of the cover with the cylindrical stud present on internal cylindrical surface (shown in figure 3.9), coincident with the hole on the aluminium adapter (3). We can joint the second part of the cover, with the rectangular stud (shown in figure 3.9) forcing the mate through nuts and screws .

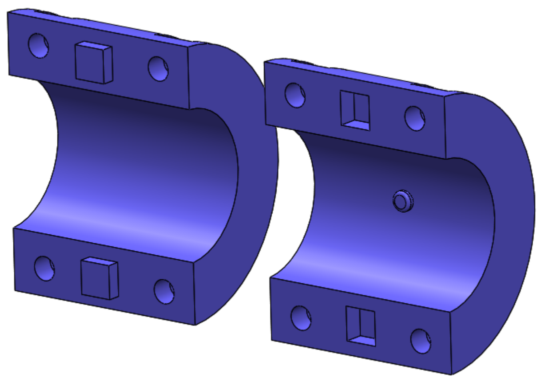


Figure 3.9: *Cylindrical pair,the two half part of the cover.*

3.5 Generic link model

In order to connect one joint with another we can not use the ABS plastic because the distance between joints does not allow us to realize links in that material with sufficient strength. Therefore we have chosen to use bars made of **carbon fiber** of 10 mm of diameter

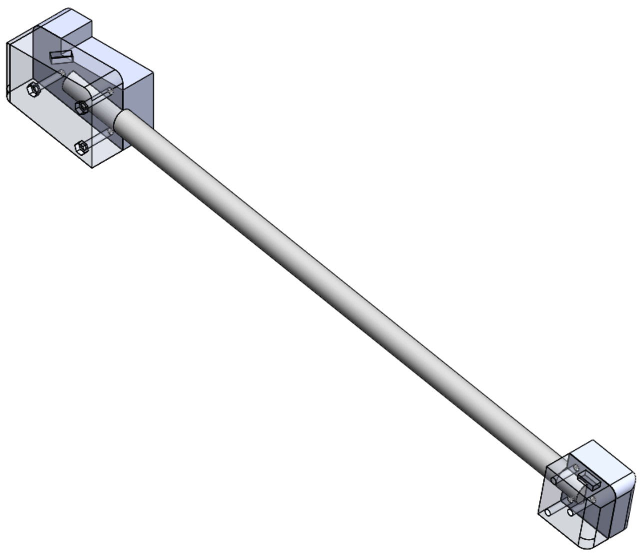


Figure 3.10: *Generic link between two joints.*

and using **chunks** in ABS (as shown in figure 3.10) to orientate this bars in the space to connect two joints. As shown in figure 3.10 each chunk denies the relative rotation with the bar through a force mate. The bar is blocked by the two parts of the chunk through the use of screws and nuts. Next, each chunk will be connect to the joint or to another chunk and the orientation between them is given by **rectangular stud** on the chunks surfaces, as shown in figure 3.11. The fixation between chunks is made through the use of **acetone** that dissolves the firsts surface layers of each part and makes a bond between the plastic parts that is stronger than the bond between layers.

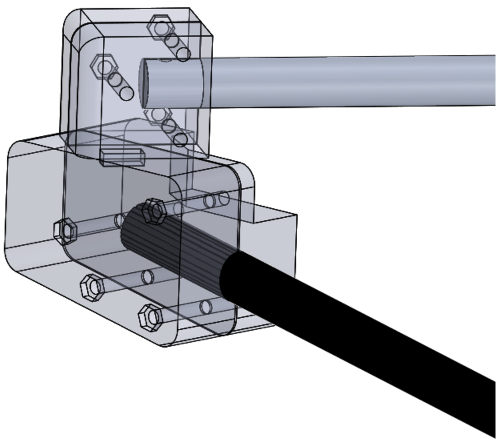


Figure 3.11: *Generic link between two joints, chunks mate.*

3.6 RCRCR mechanism CAD model

All the mechanism is built using the joint models shown above. For the connections between joints, the generic link model has been used.
It was necessary to define a CAD reference frame for each joint to place the CAD model in the space in order to respect the parameters given by the analytical results.

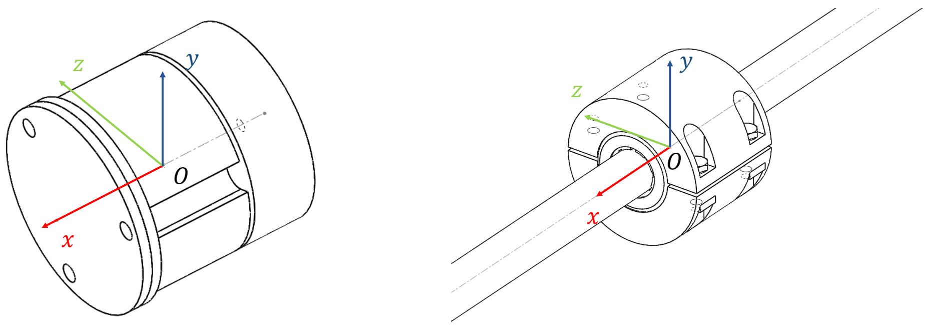


Figure 3.12: *CAD reference frame.*

As shown in figure 3.12,the reference frame has been placed in the geometric center of each joint, with the *x-axis* coincident with the rotational axis. This choice avoids the use of additional transformation matrices. In this way the reference frames shown in figure follow the trajectories found in the analytical study.
A complete assembly of the **RCRCR CAD model** is shown in figure 3.13.

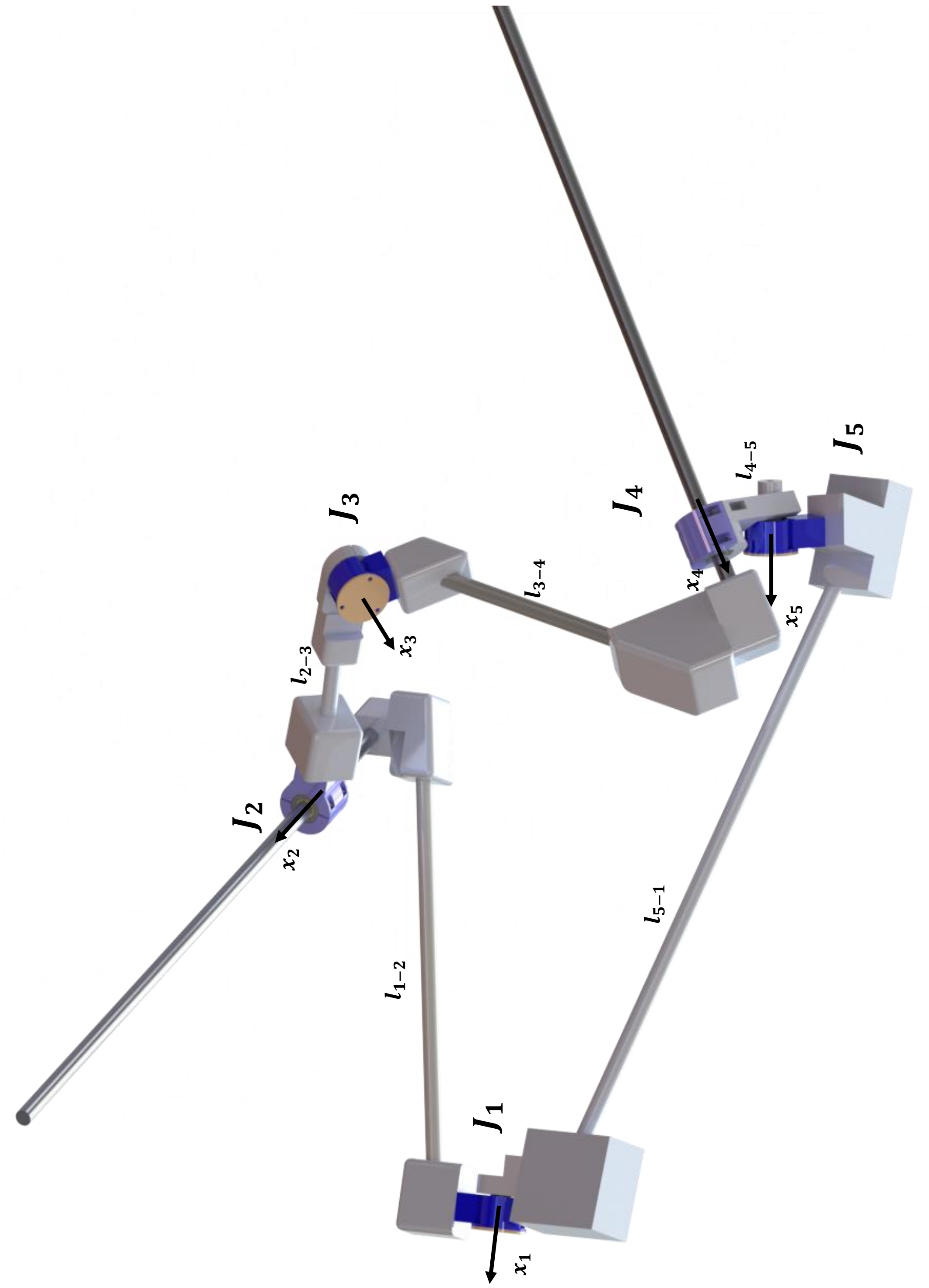


Figure 3.13: *RCRCR CAD assembly.*

The configuration shown corresponds to the point (1) of the figure 2.2, i.e the initial point of the green branch. Referring to figure 3.13, are indicated as:

- J_1, J_3, J_5 rotational joints;
- J_2, J_4 cylindrical joints;
- x_i the rotational axes of each joint;
- l_{i-j} the link that connect the joints i and j.

The link l_{4-5} has been designed without any bar. This is because the distance between the center of joints (10 units $\rightarrow$ scaled $\rightarrow$ 60 mm) allows us to connect the two joints with only one chunk. It should be noted that, for each joint, the generic models shown in figures 3.4

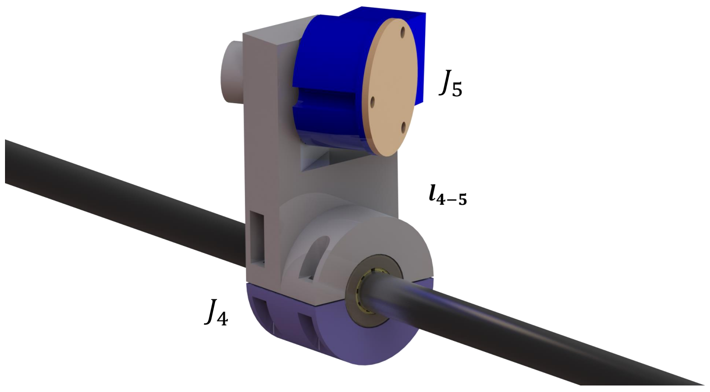


Figure 3.14: *RCRCR CAD assembly,particular* l_{4-5}.

and 3.8 have been modified in order to create a surface for the mate with the chunk.

As shown in figure 3.14, the link l_{4-5} is composed of a chunk that is included in a half part of the cylindrical cover belonging to the cylindrical joint J_4 and by another orientated part that includes the fixed part of the rotational joint J_5.
Given the necessity to connect the two joints, the short distance does not allows us to place an aluminium bar with the two chunks.

Note that the **final design** differs slightly from the schematic model used to check collisions in the maple implementation. A new test with the final shapes of all joints and links would be very complex to do in Maple. Instead, provided that the final model has been implemented in Solidworks, we can take advantage of the *collision detection* tools provided by this software.
Through the use of Solidworks tools (*"Solidworks Motion"* and *"Interferences detections"*), it has been possible to drive the mechanism through each feasible configuration along the assembly mode checking the collision (between each solid body) during the motion. The simulations take into account what provided by the Section 1.7, i.e. moving the mechanism driving only one joint (at time), switching the driving joint in the extreme configurations.

3.7 Physical implementation

Taking advantage of the laboratory tools, it has been possible to realize each provided part of the mechanism. Follows the procedure for the mechanism assembling. Once obtained all the parts it is possible to assembly the mechanism following the steps:

- assembling, separately, each **joint**;
- assembling of the **carbon fiber bars** with the **extreme chunks**;
- bonding of the **reference surfaces**;
- installation of the mechanism on the **base**;
- installation of the **motors**.

3.7.1 Assembling of the joints

In order to install each joint it has been followed the assembly procedures shown in the Sections 3.3 and 3.4 for both of the joint types.

3.7.2 Assembling of the links

As shown above in the Section 3.5, to connect the joints it has been chosen to use a carbon fiber bars with ABS chunks.

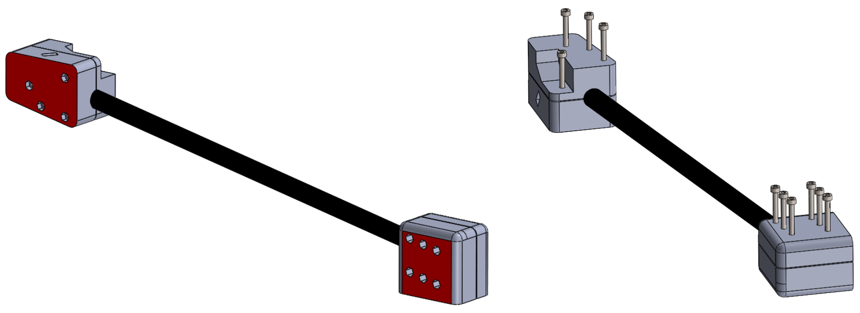


Figure 3.15: *Links assembly, reference surfaces.*

It is necessary to obtain the correct orientation between chunks because they keep the correct orientation between each joint, therefore the fixed parameters. For each couple of chunks, two **reference surfaces** have been created (for instance, the red surfaces shown in figure 3.15). Taking advantage of a plain surface it is possible to place the bar with the chunks, in a parallel condition. Then, through the use of screws, is possible to force the mate and deny the translation and the rotation of the bars.

3.7.3 Bonding of the reference surfaces

The next step in the assembly procedure consist in bounding the chunks through the use of **acetone**, as shown previously in figure 3.11. It is specified that in order to match the chunks with the joint, it has been necessary to create reference surface on the joints with additional extrusions.

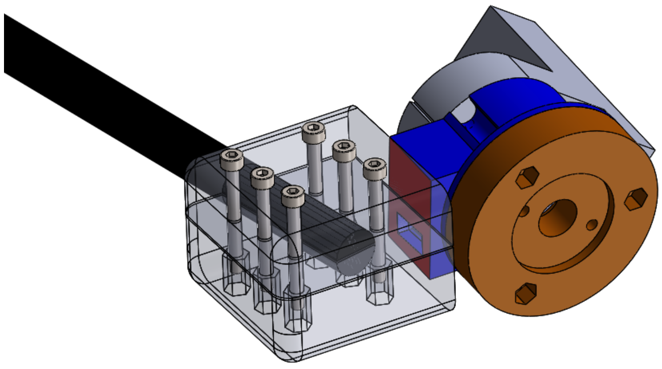


Figure 3.16: *Bounding reference surfaces.*

As shown in figure 3.16, it is possible to obtain the correct orientation between the chunks and the joint through the use of the **rectangular stud** on the chunk and the corresponding mated part on the joint. Next, it is possible to fix the relative positions putting acetone between the chunk surface and the (red) surface on the joint.

3.7.4 Installation on the base

Following the previous steps is possible to obtain the mechanism chain, as shown in the figure 3.13. Through the use of the tool *"Solidworks Motion"*, it is possible to design a **base** that allows the whole range of the mechanism motion (allowing the whole *working space* for each joint and physical component) and keep the necessary rigidity and stability.

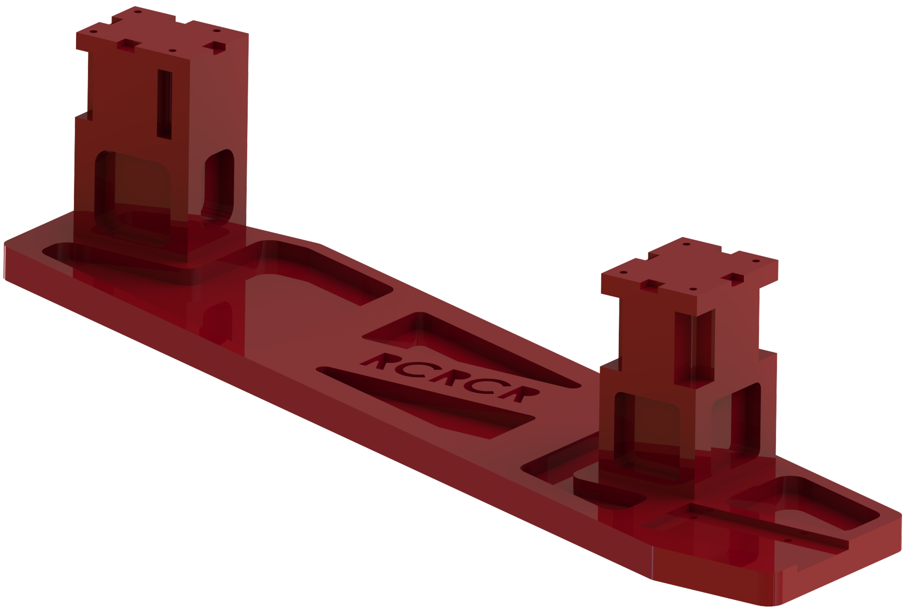


Figure 3.17: *Base of the mechanism.*

Two pillars have been designed, allowing the motion without collision with the ground, as shown in figure 3.17. The base, realized in ABS, gives stability and provides two planar surface where base the mechanism.
The choice to place the mechanism on J_1 and J_5 is not casual, in fact, the base provides dedicated space where place the two motors.

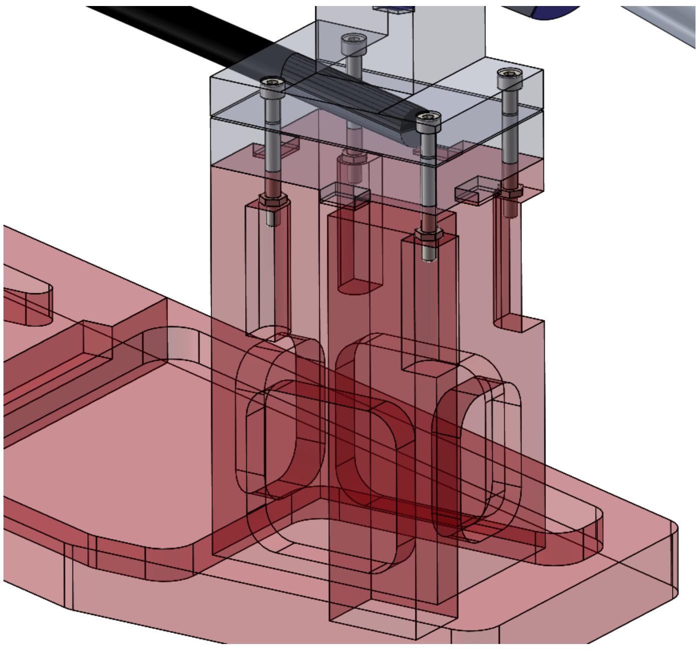


Figure 3.18: *Installation on the base, joint J_5.*

As shown in figure 3.18, it has been possible through the use of passing screws and bolts, to fix the mechanism to the base. The chunks, that contain the housing for the carbon fiber bars that acts the link l_{5-1}, are firstly refer to the base through four perimeter rectangular studs and then, screwed on the base pillars.

3.7.5 Installation of the motors

Fixed the mechanism on the base, the next step was the installation of the motors.
Firstly, it has been necessary to chose **the initial position of the mechanism** and of the motors. In fact, as explained in the Section 4.2.4, the used motors are not used in a multi-turn mode, that means that, is important to set the initial position of the motors in order to perform the required rotations without cross the zero of the motor. Cross the zero of the motor would leads to a sudden inversion of the rotation direction risking to brake the mechanism, or worse, the motors.

Moreover, is necessary to set the *initial configuration* of the mechanism from which start the motion, since the mechanism is designed to perform the cycle of configuration provided by the assembly mode two, but theoretically, it can acquire the assembly mode one configurations too. In fact, in according with the kinematics study, given only one d.o.f., for one value of θ_5 the mechanism presents four possible configurations, but setting another of the remaining six variable it is possible to define an unique configurations for the mechanism.
In order to set the mechanism variables on the correct values corresponding to the assembly

mode two, it has been decided to *"fix"* the variable θ_5 and θ_1 denying the rotations of the joints J_5 and J_1.

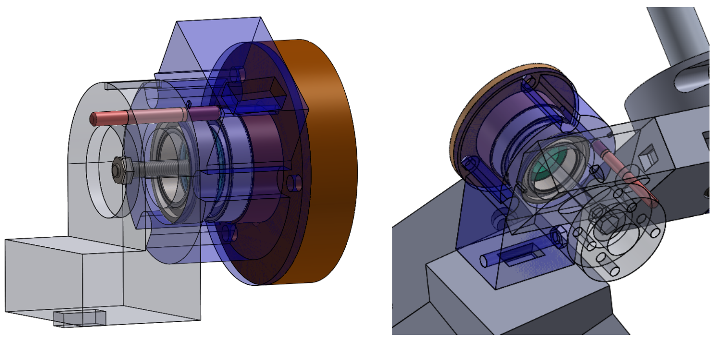


Figure 3.19: *Deny rotation pin, joint J_1 (right) and J_5 (left).*

It has been decided to set as initial configuration the configuration corresponding to the point one in figure 4.16, sufficiently far away from the extremes configurations.Therefore the initial configuration presents the set of variables shown in table 3.3:

θ_1 (deg)	θ_2 (deg)	θ_3 (deg)	θ_4 (deg)	θ_5 (deg)	s_1 (mm)	s_2 (mm)
37,30	229,23	239,35	74,57	148,96	142,35	-367,54

Table 3.3: *Initial configuration of the physical model.*

As shown in figure 3.19, a **pins made of steel** have been inserted through the fixed part and mobile part, denying the relative rotations and defining an unique instantaneous configurations.

Once get the mechanism in the initial position it is possible to install the motors. As explained in the following Section 4.2.4 for the first installation is possible to set the motor in a correct position and then install the motors on the joints.
The unusual position and orientations of the joints lead to necessity of supports where place the motors. Two different solutions have been adopted.

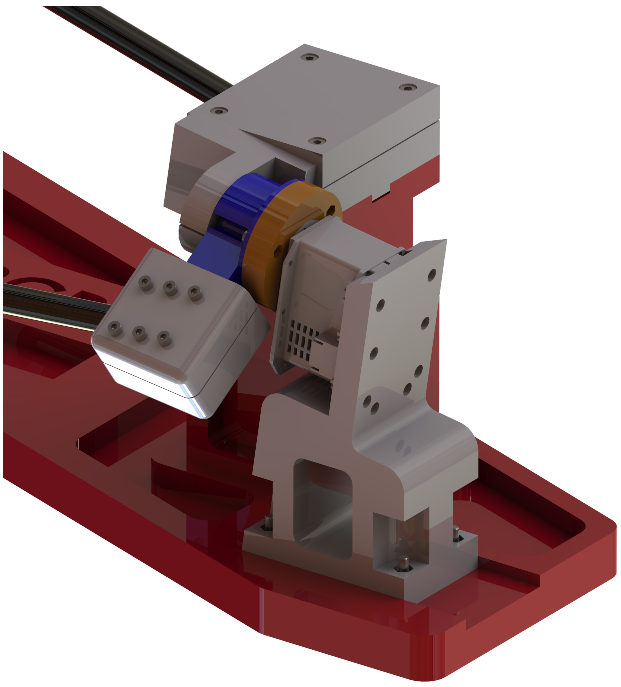


Figure 3.20: *Motor support for the joint J_1.*

As shown in figure 3.20, a support made of ABS has been designed for the joint J_1. In particular, the support presents a reference surface where match and fix through the use of bolts and screws the motor. Then, the sub-assembly motor and support can slide on a guide specifically designed embedded on the base. Once obtained the correct position of the motor on the sliding guide, it is possible to fix the support to the base through the use of bolts and screws.

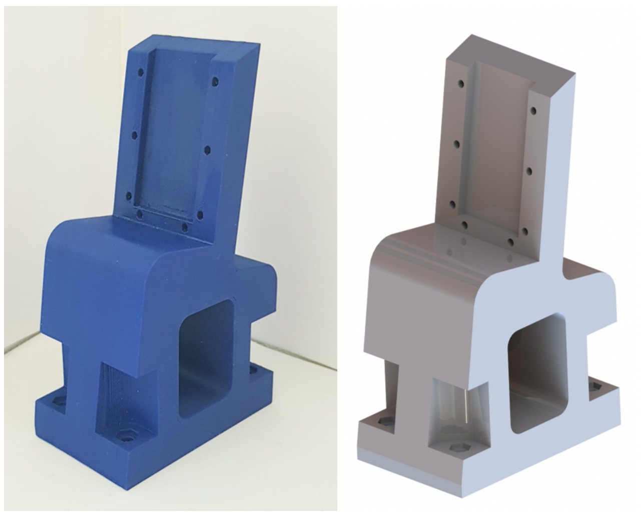


Figure 3.21: *Motor support for the joint J_1, real model (left) and CAD model(right).*

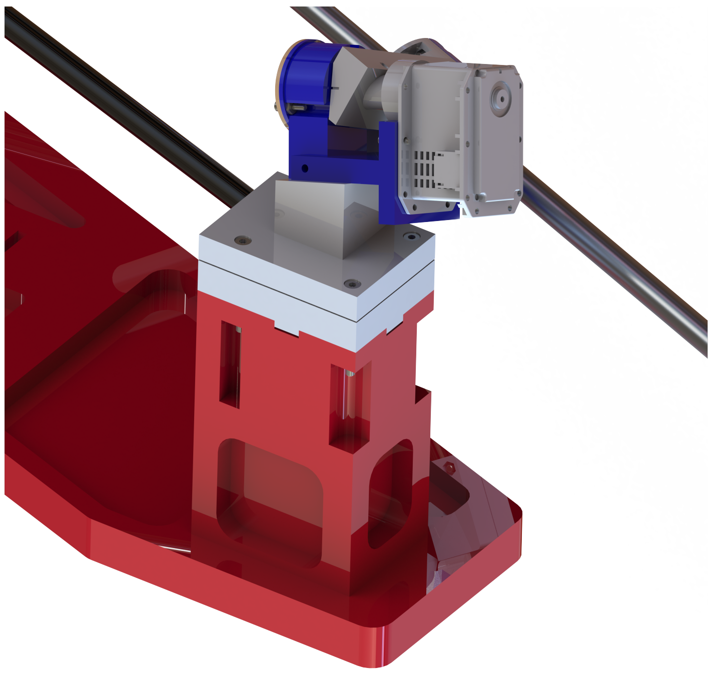


Figure 3.22: *Motor support for the joint J_5.*

For what concern the joint J_5, the orientation combined with the height of the pillar not allows to realize a support linked to the base. As shown in figure 3.22, it has been necessary to realise a support directly linked to the fixed part of the joint. The support can be bound through the use of acetone to the blue part in figure 3.22, because can not be realised embedded with that for question of problem of printing orientation (Section 3.2.1).
Match the support with the rest of the mechanism is possible to install the motor on a reference surface.

In this way is possible to complete the **physical construction of the mechanism**, shown in figure 3.23.

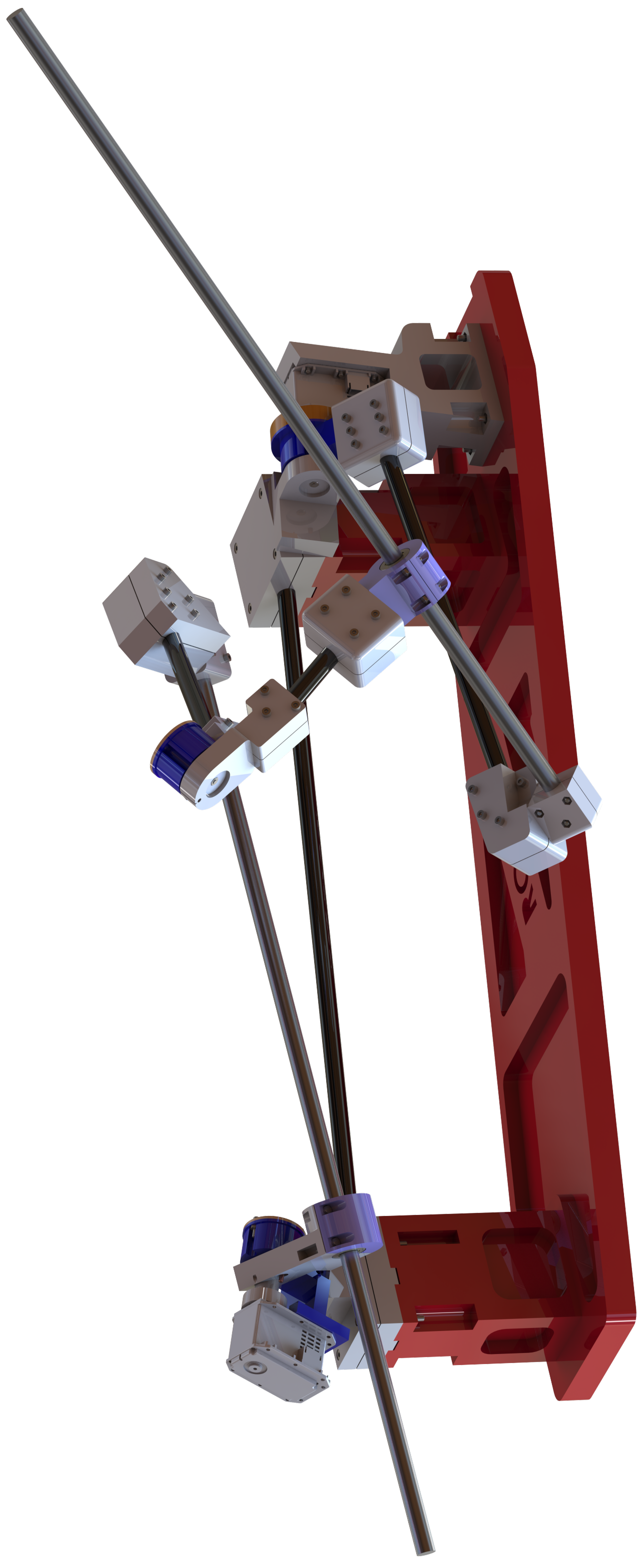


Figure 3.23: *RCRCR CAD model.*

3.8 Mechanism problems diagnosis

The built mechanism is **able to satisfy** each instantaneous position provided by the theoretical study but is **not able** to perform continuously the range of motion actuated by motors.
In fact the mechanism motion require an high accuracy of the mates and high rigidity that can not be reach with the used technology and process. Follows a list of the diagnosed problems.

- **inner tensions:** the mechanism, by nature, is affected by high inner tensions (reactions) being a loop mechanism. In fact, since the last link is not freely to move in the space but is linked to first joint leads to have inner tensions and frictions due to small errors of assembly mate that can not eliminate with the tools available;
- **component flexibility:** realise each component through the F.D.M. process (in ABS plastic) simplify the the design of the prototype allowing to realize shapes not possible with other manufacturing process. However, this implies problems linked to the use of plastic component like the high flexibility. In particular, it was noted that the weight of the mechanism leads to have a unacceptable bending of the bearings shafts of the revolute joints. This lead, unavoidably, to increase the inner tension and friction of the mechanism affecting the smooth motion planned.
- **elevate weight:** the prototype is affected by floating loads hard to foresee. In particular it was noted that the calibrated aluminium bars (Igus-DryLin ®shaft-AWMP-12) are not optimized for our propose. In fact the elevate weight cause bending of the plastics component increasing inner tensions and friction. As future improvement is suggest to use bar made of carbon fiber, losing accuracy in the cylindrical mate but reducing the weight and the overall mechanism accuracy.
- **creep phenomena:** the force mate used to deny the rotations and translations of the carbon fiber bars implies to clamp the chunks with screws and bolts. The small contact surface between screw heads and the chunks leads to a local deformation of the plastic. As time goes by, the plastics lead to have a permanent deformations affecting the force mate. This phenomena leads to lose the contact force and unwanted rotations of the bars mated with the chunks.

3.9 Design improvements

The used technology and the tools available do not allow us to solve all the problems diagnosed. However, is possible make changes in the design to improve the mechanism overall functioning.

Increase the bearing shafts rigidity

In order to avoid the bending of the bearing shafts it has been decided to improve the design of each revolute pair.

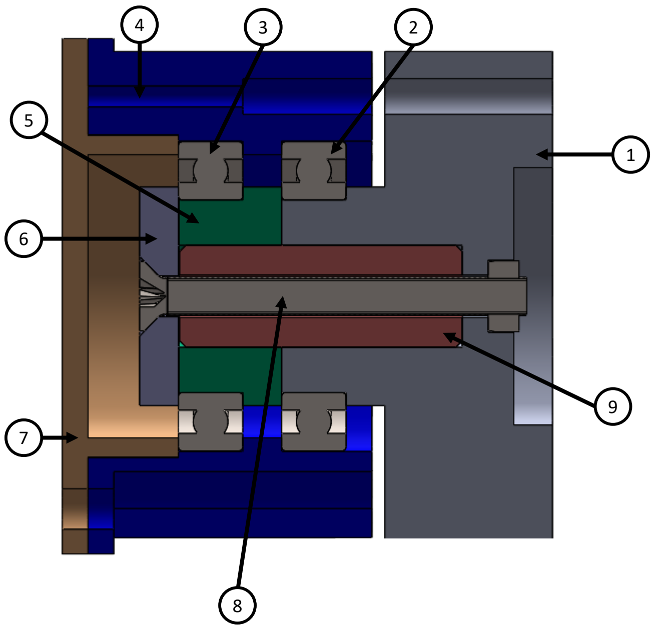


Figure 3.24: *Generic revolute joint, new design.*

It has been decided to insert a **steel core shaft** in the fixed part (1), therefore inside the bearing shaft. Referring to the figure 3.24, the components numbering is the same of the fist design (shown in figure 3.4) by adding the **shaft-core** (9).

Avoid the rotation of the carbon fiber bars

Since is not possible to avoid the creep because is linked to the used technology, in order to avoid the rotation of the bars inside the chunks, we can deny the rotation with a passing pin in case of lose mate force.

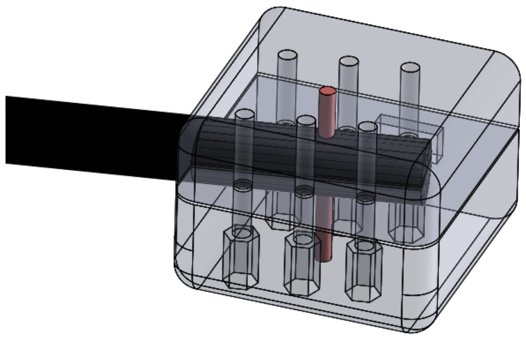


Figure 3.25: *Pin to deny the rotations of the bar.*

As shown in figure 3.25, the pin, made of steel, pass through the chunks and the carbon fiber bars but, it should to clarify that is a safety tool, because the force mate is mainly given by the clamping with screws.

3.10 Final mechanism problems diagnosis

The improvements in the design shown in the previous sections, lead to increase the overall rigidity of the system. In fact, it has been noticed that, with the new design of the rotational joint the bending of the bearing shaft explained in the Section 3.8 is almost in existent. In addition the pins used to avoid the bar rotations have obtained the expected effects.

Even if the simulations in Solidworks environment shown that the kinematics motion of the mechanism respect what provided by the theoretical study, as **final diagnosis**, the mechanism is unable to perform what provided by the theoretical study due to the following problems.

The inner tensions and reactions in the passive joints, lead to the deformations of the plastic components. In particular, has been observed deformations in the chunks for the bar clutches. This phenomena leads to increase frictions and inner tensions, making impossible the whole movement of the mechanism by producing mechanical jamming during the motion.

Therefore, as future work, in necessary a study of the joints reactions to provide a correct design of that mechanism.

Finally, driving only one motor, the mechanism is able to perform only an initial part of the motion as shown in figure 4.19 (Section 4.2.5).

3.11 Prototype real component

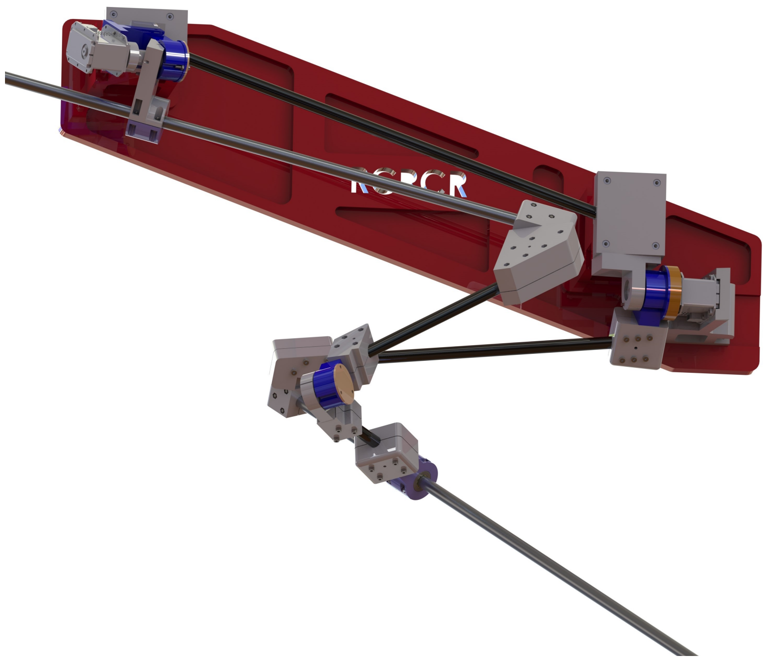


Figure 3.26: *RCRCR CAD model, top view.*

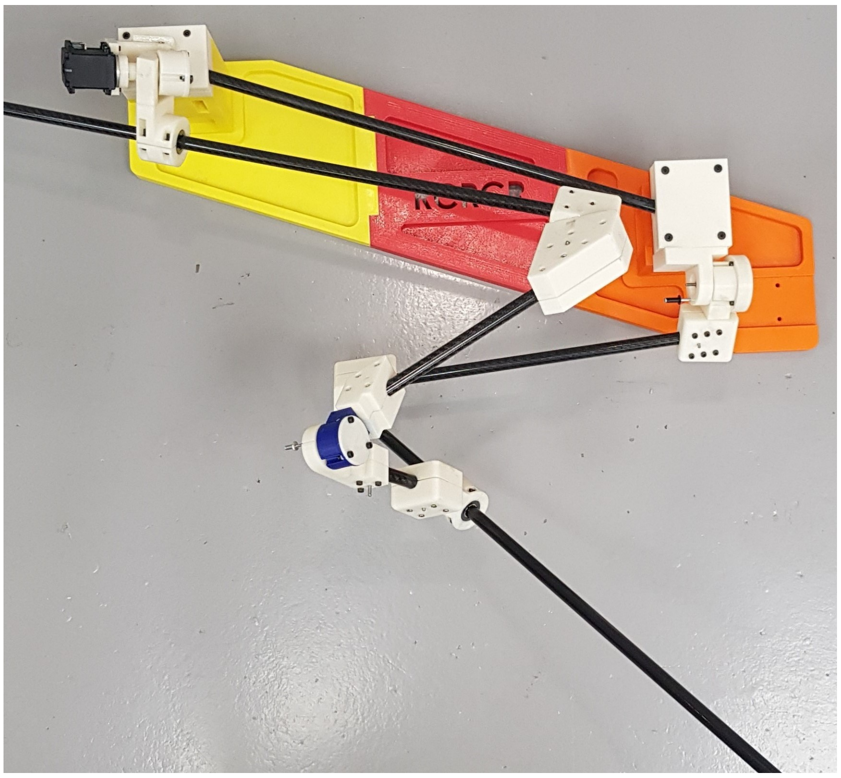


Figure 3.27: *RCRCR real prototype, top view.*

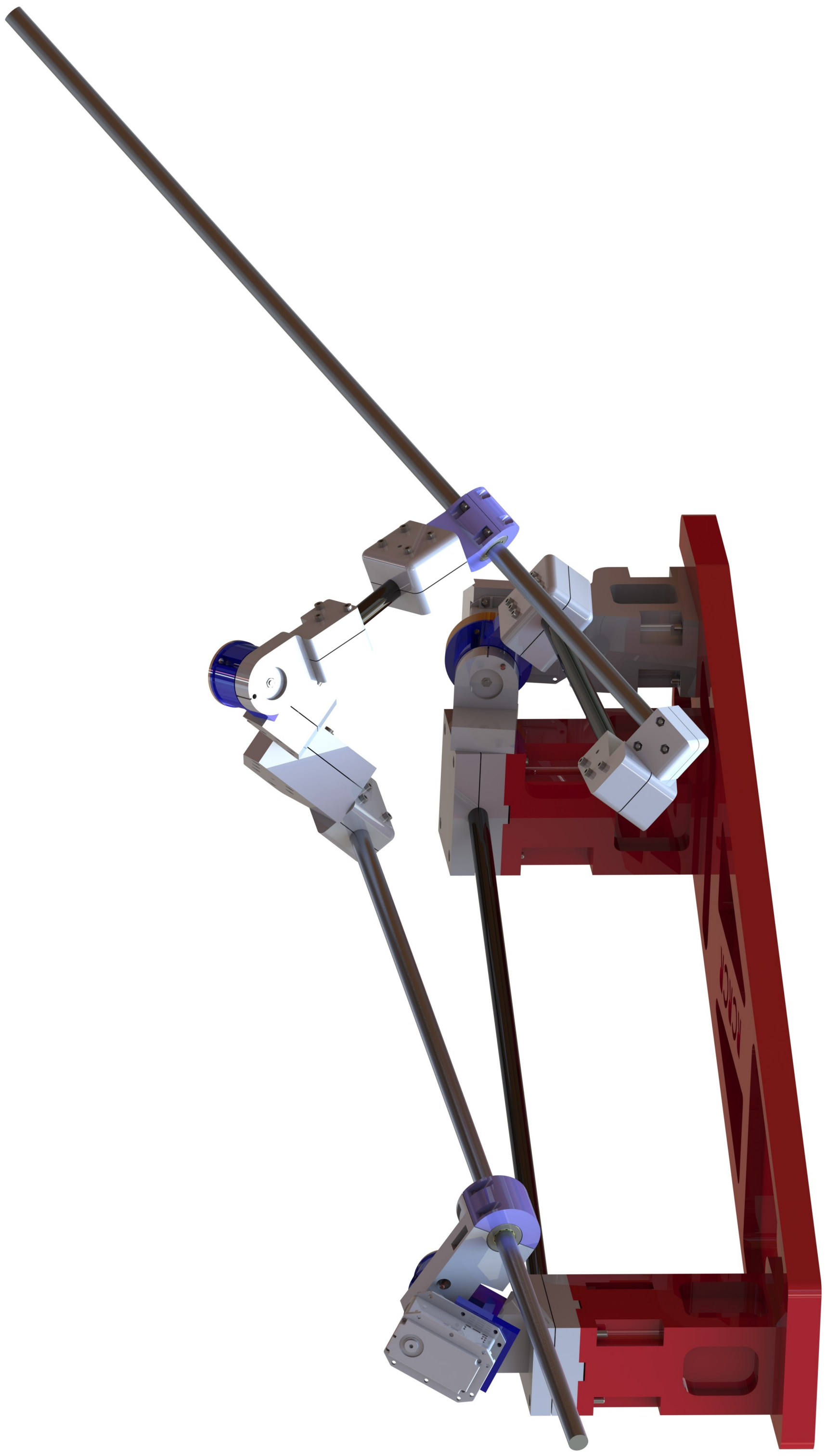


Figure 3.28: *RCRCR CAD model, main view (configuration: $\theta_5 = 148.96$ deg, $\theta_1 = 37,30$ deg).*

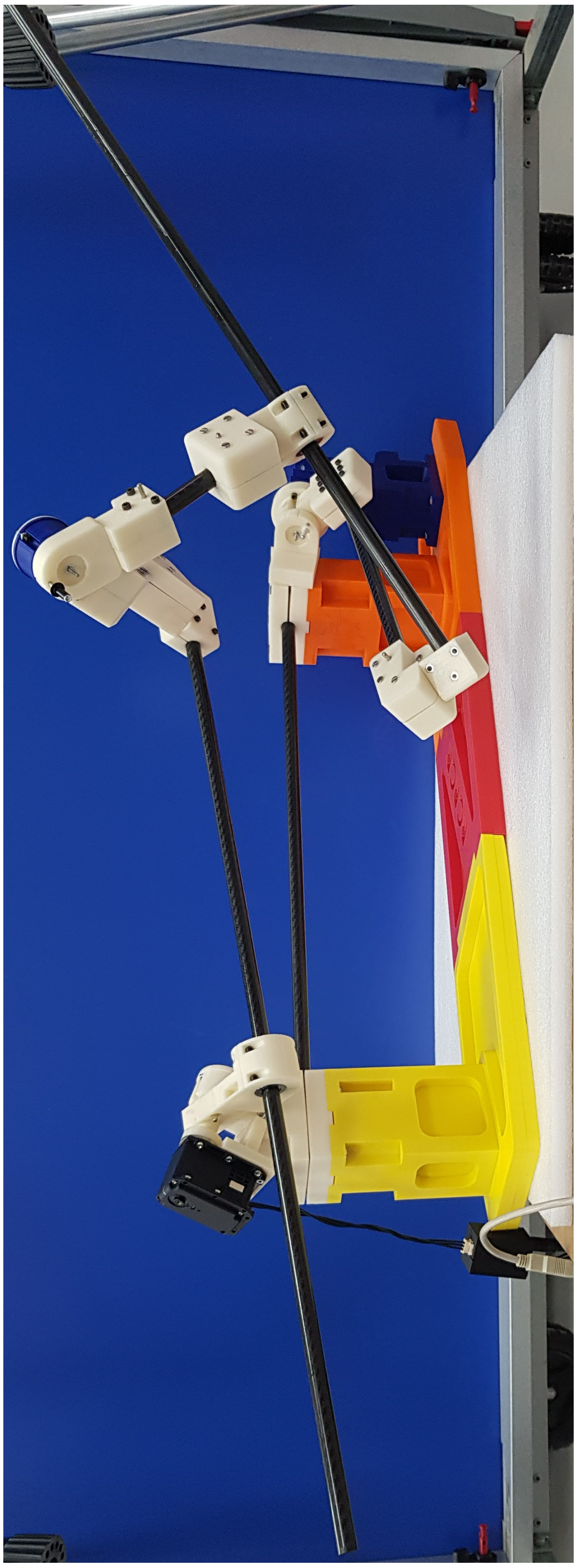


Figure 3.29: *RCRCR real prototype, main view (configuration: $\theta_5 = 148.96$ deg, $\theta_1 = 37, 30$ deg).*

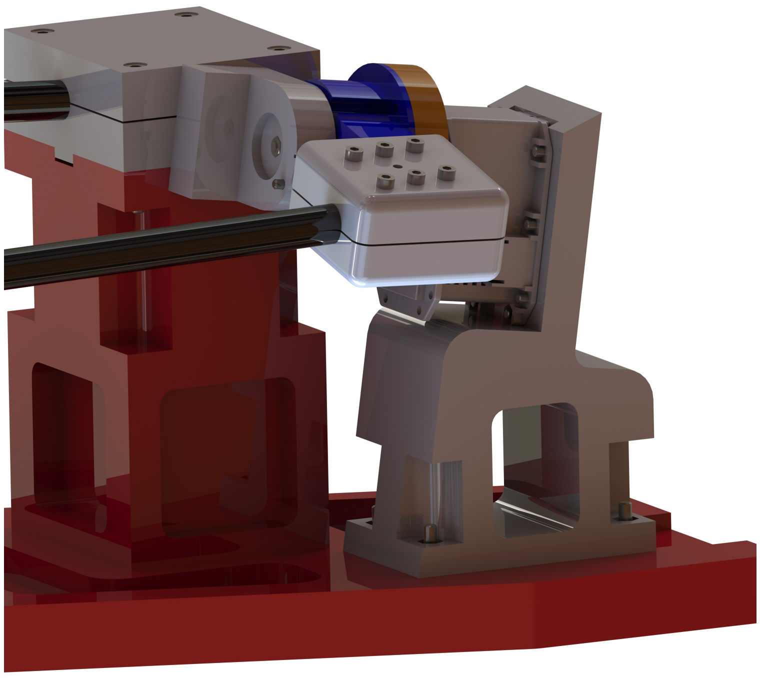


Figure 3.30: *RCRCR CAD model, joint J_1 and motor placed.*

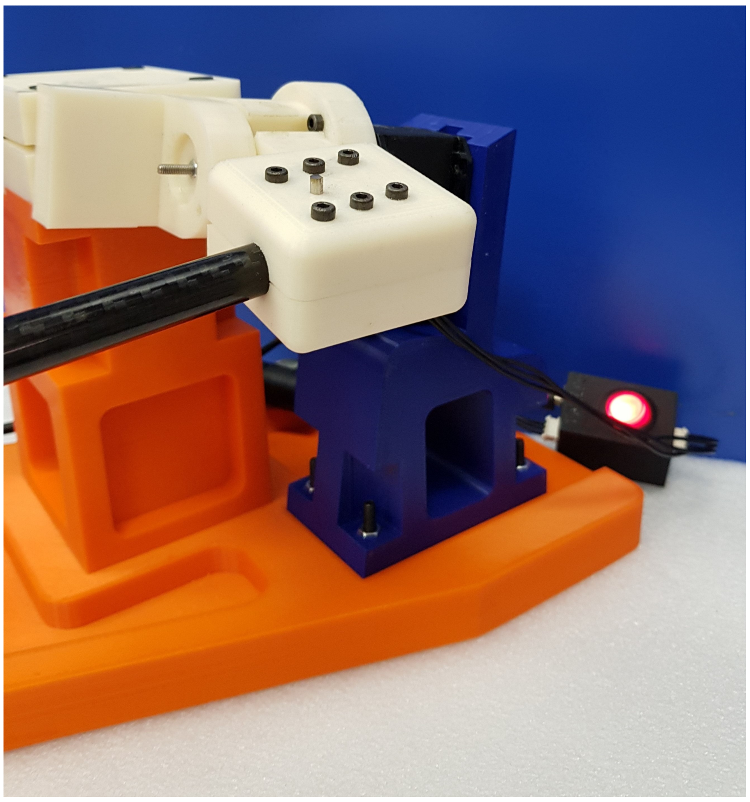


Figure 3.31: *RCRCR real prototype, joint J_1 and motor placed.*

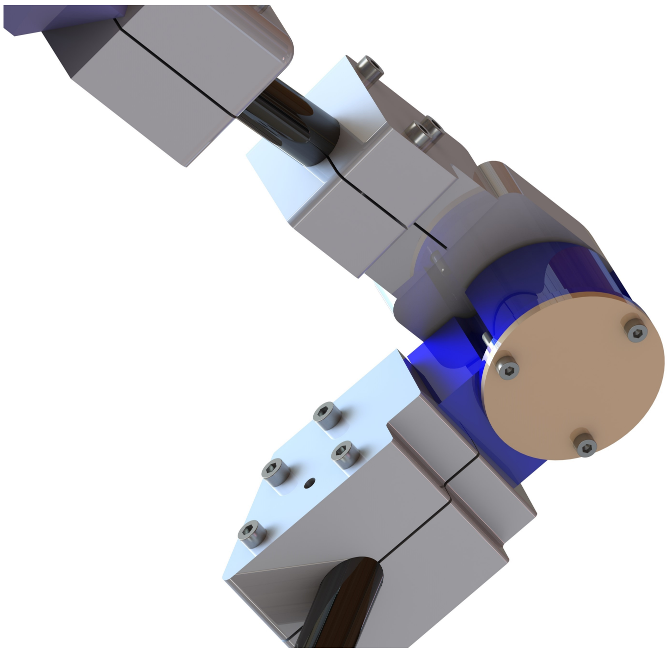


Figure 3.32: *RCRCR CAD model, joint J_3.*

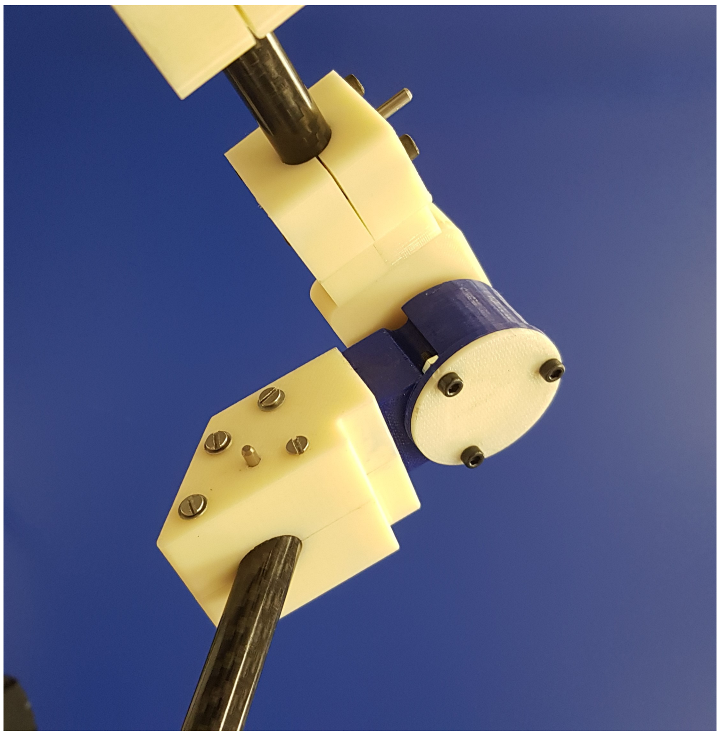


Figure 3.33: *RCRCR real prototype, joint J_3.*

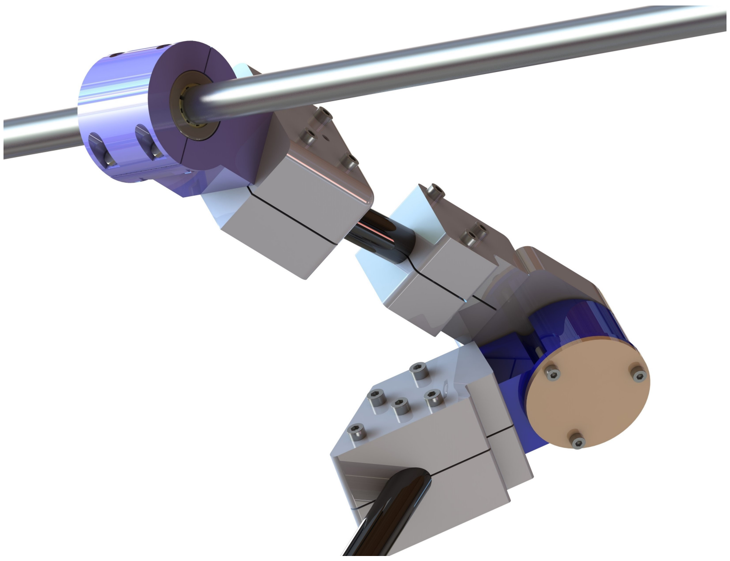


Figure 3.34: *RCRCR CAD model, joint J_2 and J_3.*

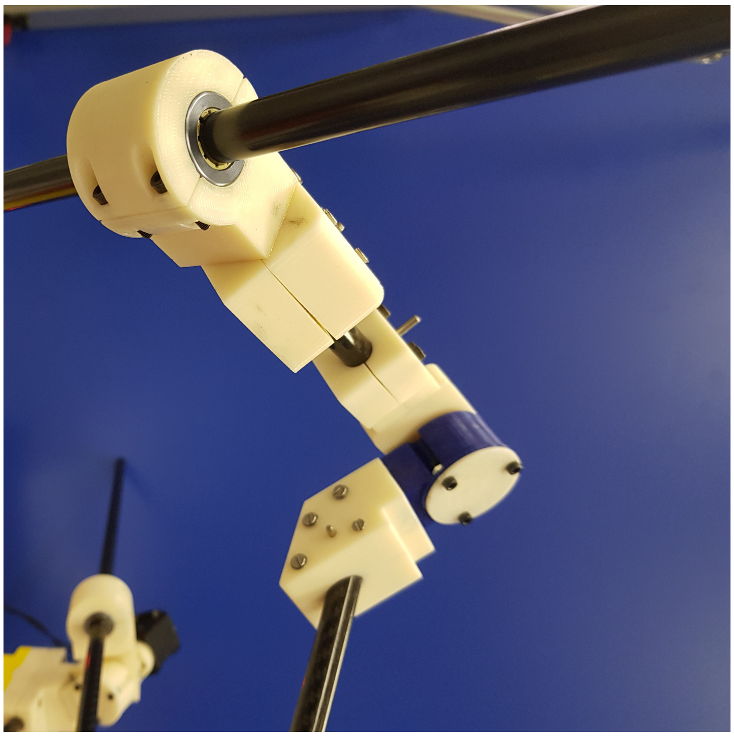


Figure 3.35: *RCRCR real prototype, joint J_2 and J_3.*

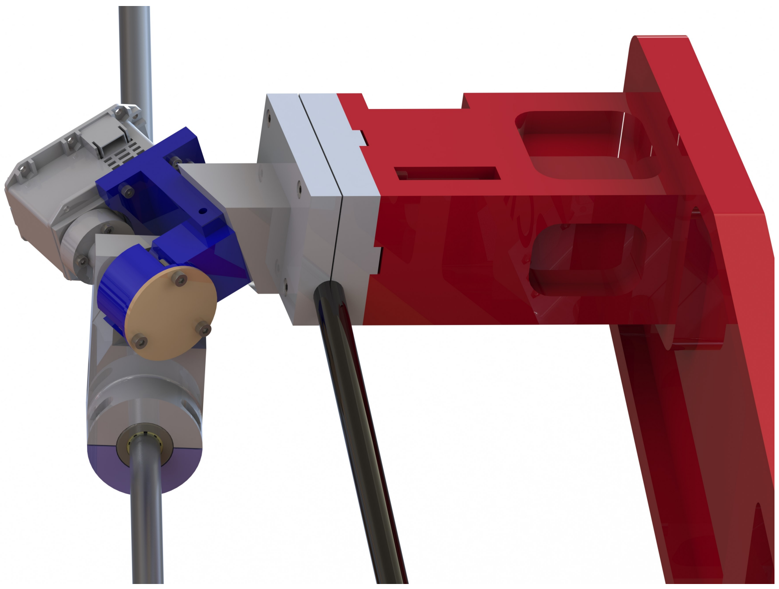


Figure 3.36: *RCRCR CAD model, joint J_5 and motor placed.*

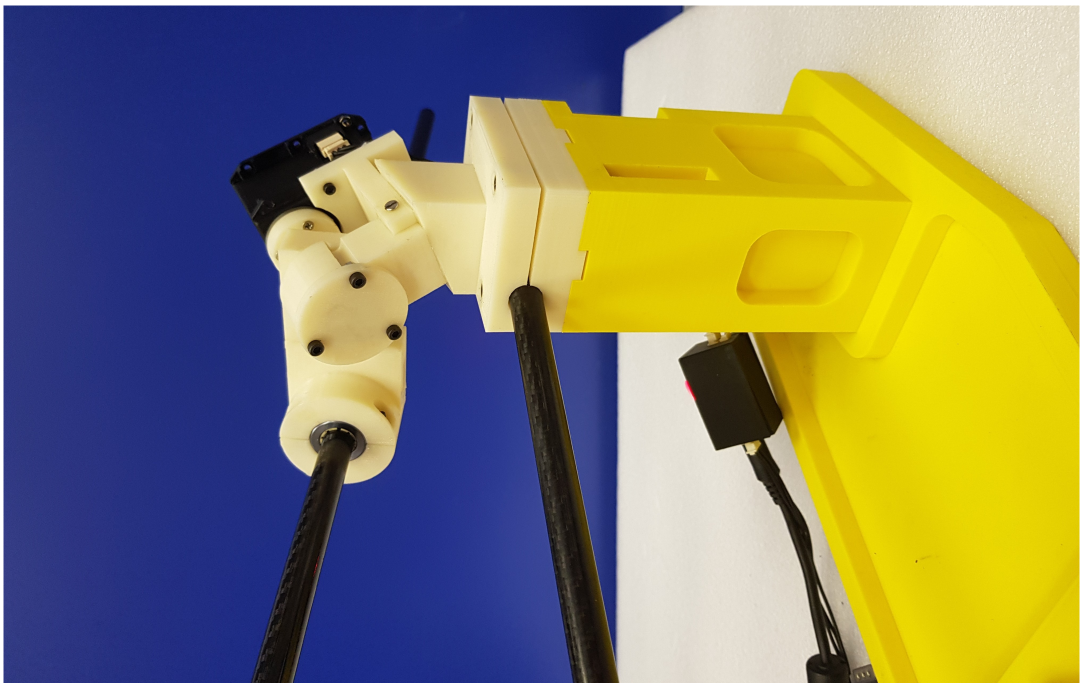


Figure 3.37: *RCRCR real prototype, joint J_5 and motor placed.*

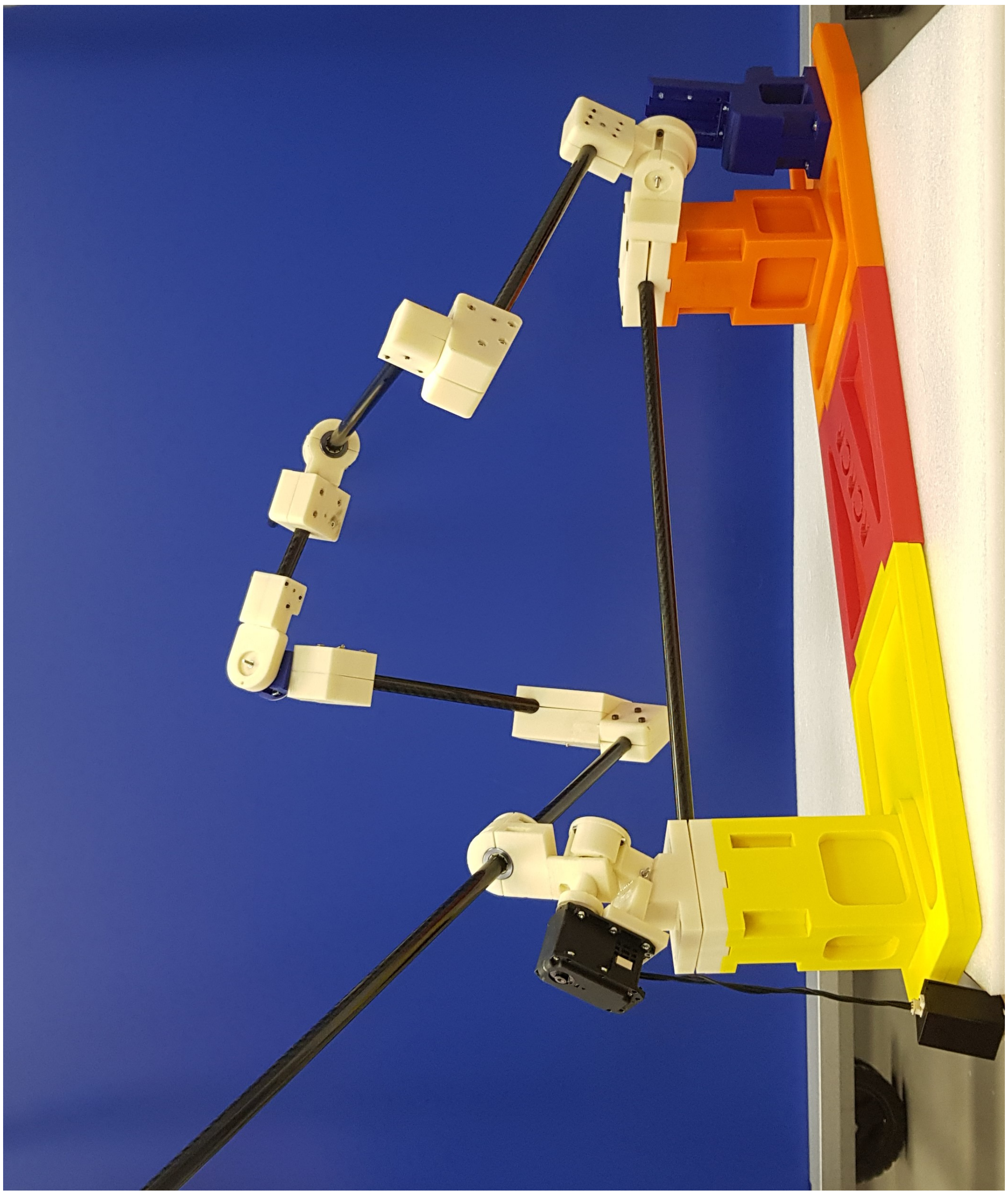


Figure 3.38: *RCRCR real prototype, main view (configuration: $\theta_5 = 303.66$ deg, $\theta_1 = 296,66$ deg).*

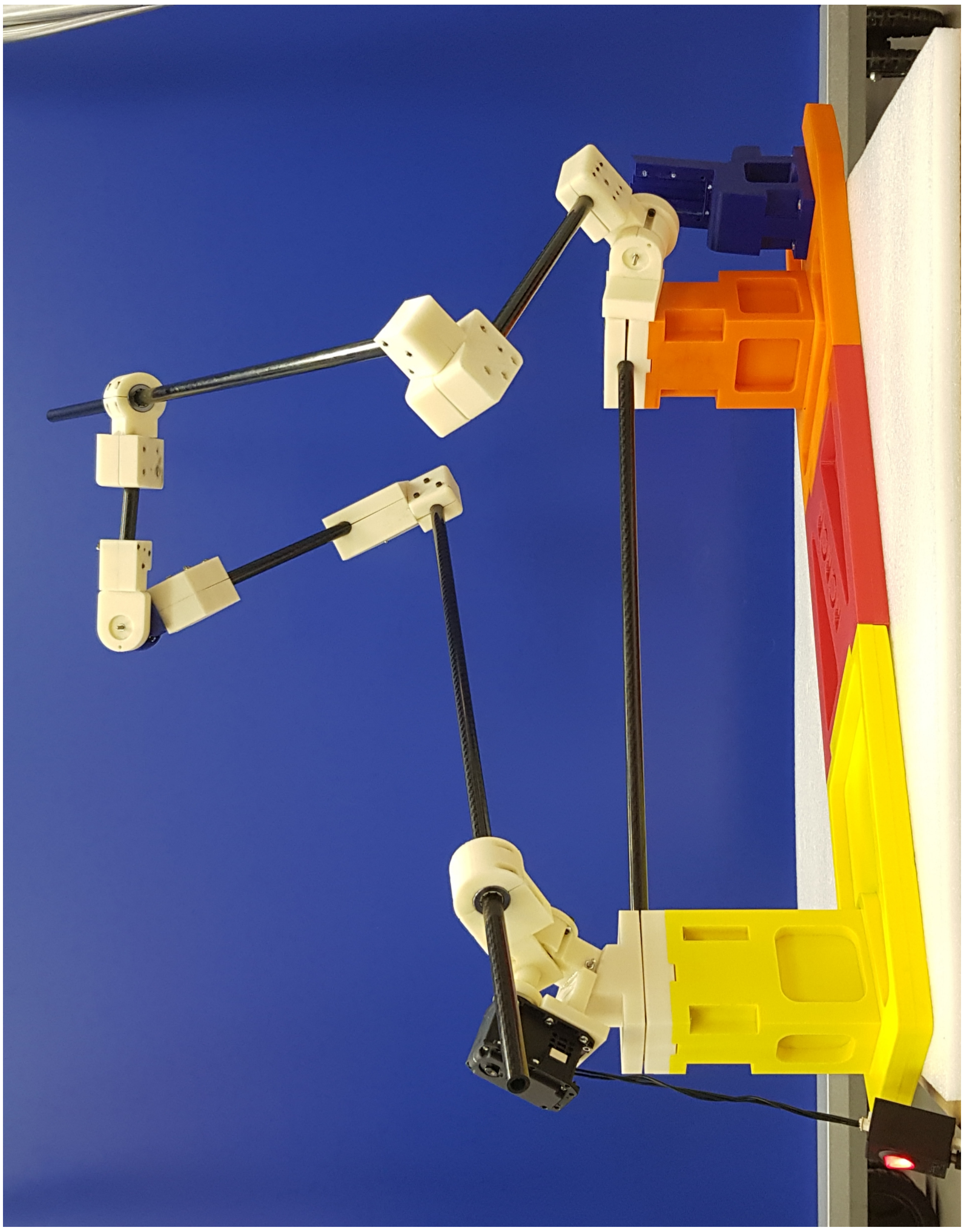


Figure 3.39: *RCRCR real prototype, main view (configuration: $\theta_5 = 263.56$ deg, $\theta_1 = 281.93$ deg).*

Chapter 4

Implementation of the Motors

The implementation of the motors has been initially developed in parallel with the theoretical study and the design of the mechanism. Therefore, in the following Chapter, follows firstly a report on a learning activity. The goal of this intermediate activity was to familiarize with the motor control taking into advantage of the available models in the research center. Than, it has been chosen the motor to install once built the mechanism.

In particular for the learning activity, the model *"Dynamixel AX-12A"* motor has been used. It has been possible to implement a feed-forward position control. Moreover, this motor has been used for the motorization of a 3R serial robot, that allows us to implement an algorithm to drive three coordinate motors.

Once defined the scale and build the mechanism, it has been chosen to use two *"Dynamixel MX-64T"* motors as driving system for the RCRCR mechanism. This new model of motor presents higher performances coherently with the size and weight of the mechanism.

Then, an algorithm in Matlab environment has been implemented, in order to drive the motors according to the theoretical study. Formally, the sequence of analysis developed is not correct. In fact, it should be necessary a dynamic analysis to choose the optimized size of the motors. Moreover, it has been chosen the mentioned above sequence of analysis for the brief time available and the availability of the research center, in order to achieve a correct mechanism motion.

4.1 Learning the use of the Dynamixel Motors

In order to execute the motion of the mechanism as foreseen Section 1.7, the *"Dynamixel Motors"* have been used, coherently with the laboratory availability. The use of these motors have required a first learning activity related to the **velocity control** of these motors, **checking**, in real time, its **position**. It is noted that, in this activity, the general goal is not the realization of a specific shape of velocity with the required dynamic features, but the general comprehension and the familiarization with the control of the motors.
Only for this task, a **3R-robot** (figure 4.1) has been used, i.e. a robot with 3 links connected by 3 revolute joints with three *"Dynamixel Motors AX-12"* connected.

A kinematic study of this robot was not necessary, because we focused on the general control of a single motion of one motor at time and attention has been paid only on the range

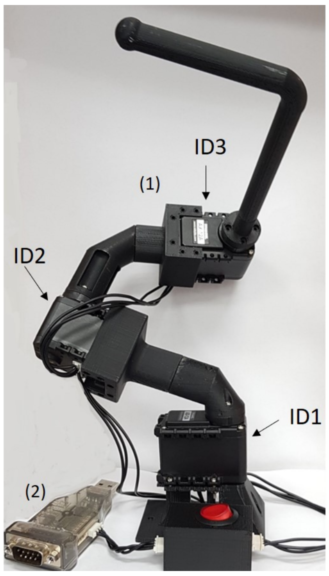


Figure 4.1: *(1)Robot 3R and (2) USB2Dynamixel hardware.*

limits linked to the cables that connected the motors to electric supply.

4.1.1 Brief description of the motors environment

In the following section we review the motor's working environment.
As shown in figure 4.2, the motor receives an input in Volt and presents a **Micro-processor** (μP) that manages the informations received from the users. These servomotors are used to work receiving and sending information through the appropriate addersses of the RAM inside the motor. In fact, the motor receives the information by an **USB2Dynamixel** hardware (shown in 4.2).

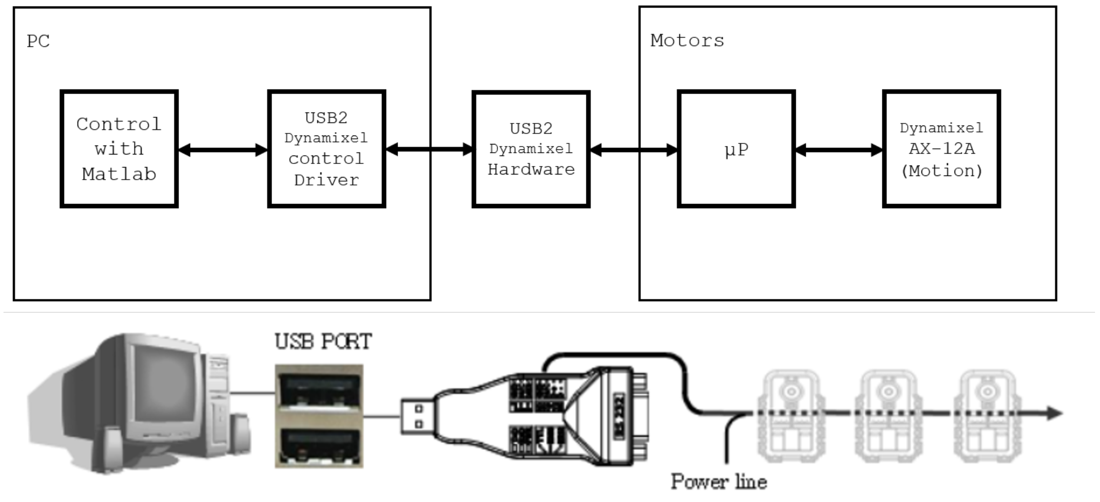


Figure 4.2: *Robotis Dynamixel, motors environment [10].*

This hardware makes a bridge between the language outgoing from the **PC** and that incoming the motors. The protocol that comes from the PC is managed by an *USB2Dynamixel* software inside the PC. Thanks to this, we can use the software **Matlab** to control the motors. Obviously we can also read information from the motors, such as position,velocity, range limits and other important parameters, which are useful to understand, in real time, if the motors are doing what we expected.

4.1.2 Motor model: Dynamixel AX-12A

The model of motor that has been used is the **Dynamixel AX-12A** Robot Actuator [10], shown in figure 4.3:

Figure 4.3: *Dynamixel AX-12A Robot Actuator [10].*

This model of motor allows to track its speed, temperature, shaft position, voltage, and load. All of the sensor management and position control is handled by the servo's built-in microcontroller. The software used is Roboplus ®, that allows to see some motor parameters in real time, to know their allocation in the library and to do small simulations of a robot motion. The motor specifications follow to the table 4.1.

Dynamixel AX-12A Specification	
Operating Voltage	12V
Stall Torque	1.5 Nm
No-load Speed	59 RPM
Weight	55 g
Size	32 x 50 x 40 mm
Resolution	0.29 deg
Reduction Ratio	1/254
Operating Angle	300 deg or Continuous Turn
Max Current	900 mA
Standby Current	50 mA
Operating Temp	-5 ° C - 70°C
Material	Plastic Gears and Body
Motor	Cored Motor

Table 4.1: *Dynamixel AX-12A, motor specifications [10].*

4.1.3 Control Table and functioning type of the motor Dyamixel AX-12A

The **Control Table** consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet. Users can read a specific data to get status of the Dynamixel with read instruction packets, and modify data as well to control Dynamixel with write instruction packets. In particular in the Control Table is possible to found:

- **EEPROM and RAM**: data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.
- **Address**: It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.
- **Access**: Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.
- **Initial Value**: In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.
- **Highest/Lowest Byte**: In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16 bit, but it is divided into 8 bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

As it is explained above, the control and the managing of the motor through the PC, pass from the reading and the writing of the addresses of the RAM, in either the volatile part and the non-volatile one.
In order to understand which address it was used, it is represented in a table, the most important part of the whole **Control Table** of the Motor.

Control Table						
EEPRON Area						
Address	**Size [byte]**	**Data Name**	**Access**	**Initial Value**	**Min**	**Max**
3	1	ID	RW	1	0	252
4	1	Baud Rate	RW	3	0	3
RAM Area						
Address	**Size [byte]**	**Data Name**	**Access**	**Initial Value**	**Min**	**Max**
24	1	Torque Enable	W	0	0	1
30	2	Goal Position	W	-	0	1023
32	2	Moving Speed	W	-	0	2047
37	2	Present Position	R	-	0	1023
39	2	Present Speed	R	-	0	1023

Table 4.2: *Dynamixel AX-12A, Control Table [10].*

The **functioning** of the motor depends on the type of control, but it used to work with a **Modulation of the Power-PWM**, writing a percentage of Voltage in bits, in the address

of the *Moving Speed*. Modulating means to write an amount of the maximum power, given that the current is fixed.
The motors *"Dynamixel Motors AX-12"* can work in two different control mode:

- **Wheel Mode:** in this mode, it is possible to control the velocity by modulating the power from zero to the maximum value. The value in this address move from 0 to 2048, where:
  - $0 - 1023$ (CCW), modulates the power (and the velocity) from 0 to the maximum; writing 512 means to use the 50% of the power, in counter-clockwise direction;
  - $1024 - 2048$ (CW), modulates the power in the clockwise direction;
- **Joint Mode**: in this type of control, it is possible to write and read the *angular position* of the motor in order to control it. It is not possible to control the velocity, but it is only possible to set the maximum value that the motor can use to arrive at the required position.

We can switch from one mode to another by software. The following control has been realized in **"Wheel Mode"**.

Several important considerations have to be done about the value that can be read in both control modes of the motor:

- **Write and read position:** it is allowed to write and read from 0° to 300°. Even if the motor does not have any kind of physical constraint, the range between $300° - 360°$ is an invalid angle, so the value read has no sense;
- **Read velocity**: working in joint mode is possible to read a value directly convertible to a *rpm* value trough a conversion factor. In wheel mode, it is not possible to read the velocity, because of the address read *Present Speed* reports a value that corresponds to the volt percentage. It must be calculated through the derivation of the position. In our case, the law between the percentage of voltage written and the respective desired velocity, in no-load condition, is directly proportional.

4.1.4 Feed-Forward position control:point-to-point motion in the joint space

It has been chosen to perform a point to point motion in joint space, controlled with a **trapezoidal velocity shape**, shown in figure 4.4.
In fact, our interest in this activity is not the dynamic requirements, therefore in the following discussion we will not treat the parameters of the motion laws (they can be easily read from the Matlab file, Appendix E), but there is a focus on the behaviour of our motors in reference to the theoretical motion law.

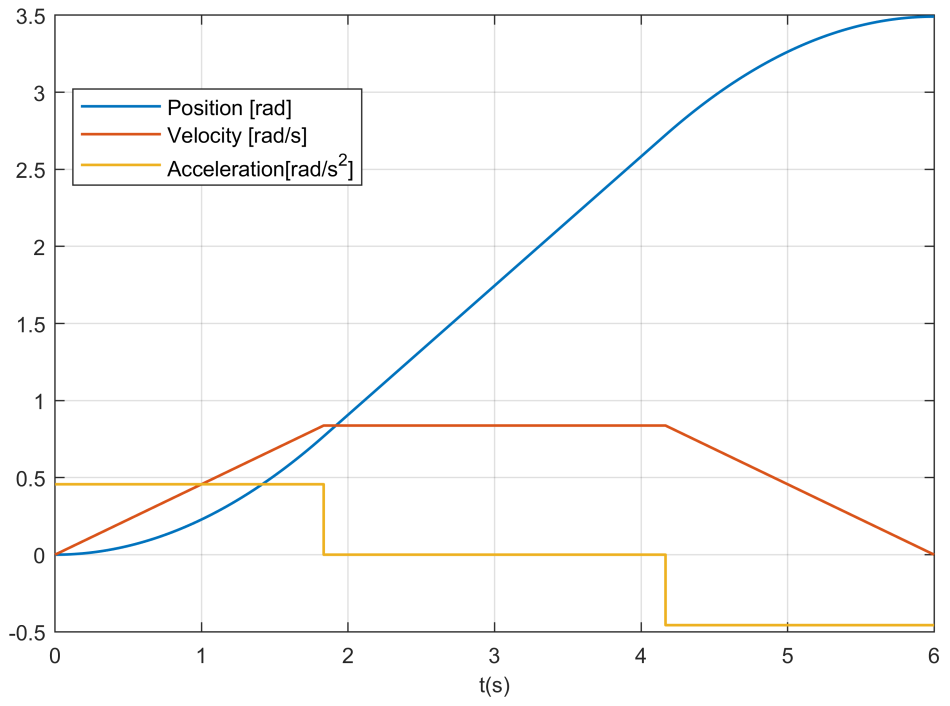


Figure 4.4: *Trapezoidal velocity law.*

Referring to the figure 4.4 the initial joint position and velocity, final joint position, time of execution and maximum velocity have been chosen, with assumption of symmetrical profile. That means that the other parameters can be automatically calculated in Matlab code starting from the known data.
The motor presents an inner PID, that is not sufficient to follow the timing position law. Moreover, the tuning of the PID parameters is not possible. During the several tests, it has been possible to notice that the motor presents other practical problems as response delay, dead band.
Therefore, in the following analysis the motor has been treated as a black-box. In order to perform the desired shape of position, an **Feed-Forward Position Control** it has been realized, as shown in figure 4.5:

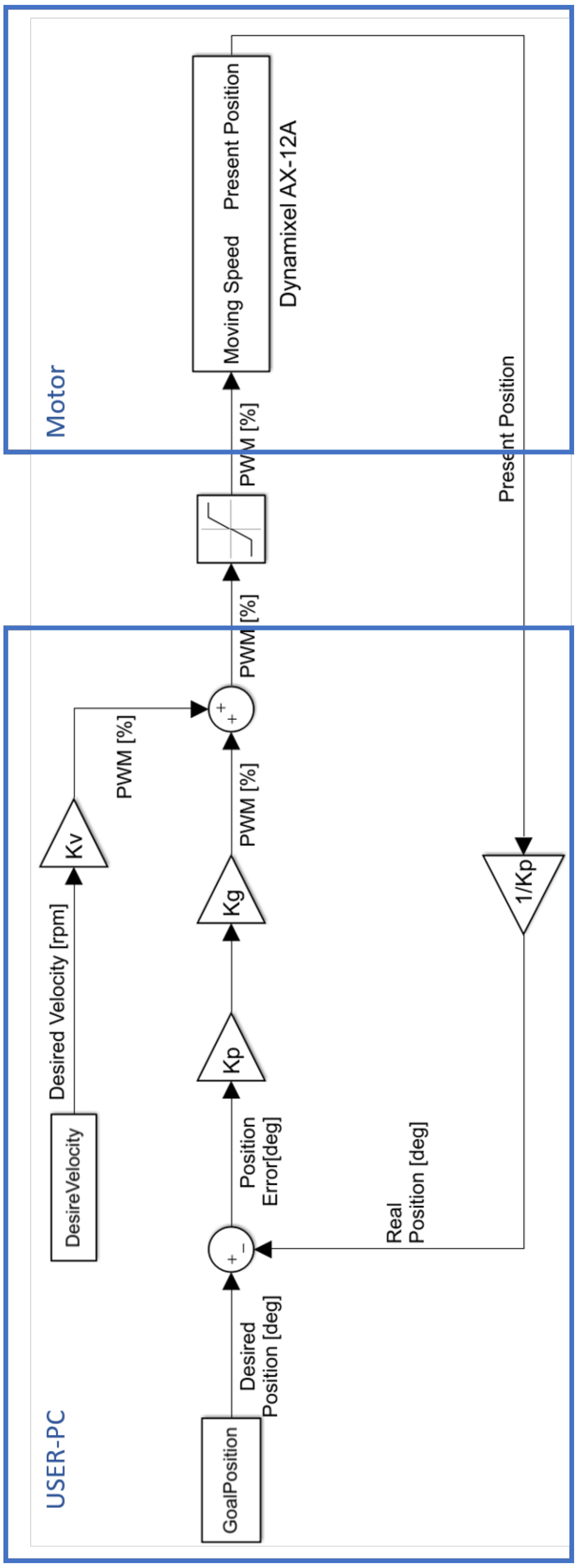


Figure 4.5: *Feed Forward Position control.*

The goal of this control is to follow the theoretical curve of the position for a given timing trajectory.

Therefore, the algorithm evaluate in real time:

- reads the **real position** of the joint (θ_{FB});
- compares θ_{Read} with the **theoretical position** (θ_{SET})that came from the motion law ($\theta_{Err} = \theta_{SET} - \theta_{Read}$);
- proportional control, in order to obtain a compensated velocity;
- compensate the desired velocity ($\dot{\theta} = \dot{\theta_{SET}} + \dot{\theta_{Err}}$), the **gain** Kv from a value in *rpm* to a *volt signal*;
- before to communicate the volt value to the motor, it is necessary a **saturation** provide to send the correct value, since the value given to the motor can be:
  - $0 - 1023$ counter-clockwise rotation;
  - $1024 - 2048$ clockwise rotation;
- Write the value of moving speed in the address value;
- the “Dynamixel AX-12A” block, carries out the instruction of Input percentage of voltage (input velocity) and moves the motor. This last block is not inside the Matlab code, but is located inside the motor.

Basically, with this control, we try to follow the theoretical position driving the velocity, as shown in figure (4.6). That means that the shape of obtained position is smooth and close to the theoretical one, instead, the velocity fluctuates around the theoretical one.

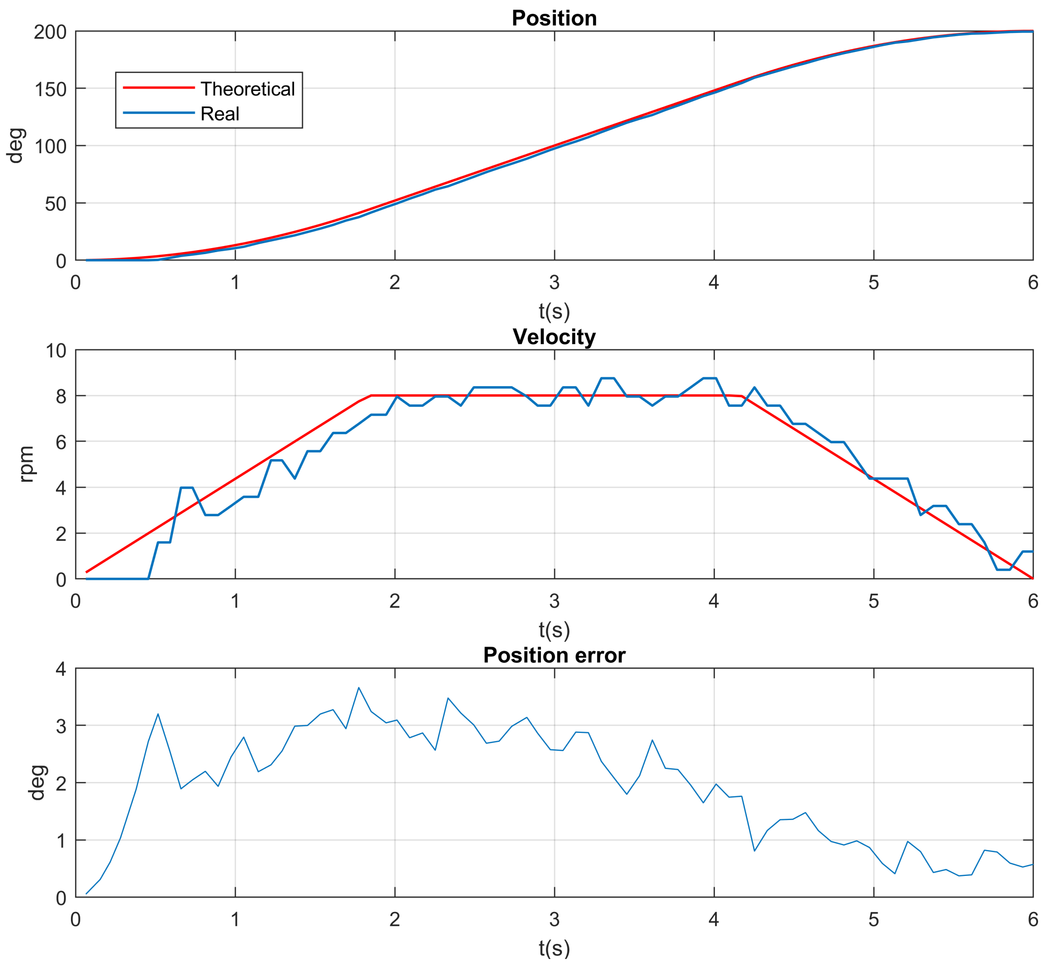


Figure 4.6: *Comparison of real and theoretical position and velocity obtained.*

It should be noted that, the implemented proportional control is enough to follow the desired position trajectory, even if by changing the requested application, it is necessary in the external closed loop to implement a PID controller, coherently with the motor one, in order to obtain the desire behaviour.

All the Matlab files produced are included in the Appendix E.

As learning activity on the control, several experiments have been done, using a trapezoidal velocity shape of timing law:

- finding the best proportional gain K_g in linear condition, without any kind of load;
- evaluating the motor behaviour for different values of final time t_f;
- evaluating in a coordinate control of three motors, the different behaviour under variable load.

4.1.5 Further considerations

By analysing the behaviour of the motor in this first learning activity, it is possible to underline some considerations linked to the control used.

- by analysing the plots, it should be noticed a starting delay due to the dead band of the motor. For low speeds (that correspond to a low percentage of voltage to modulate the power) the motor is not ready to follow the initial values of timing law.
- by analysing the linear response without any load in the figure 4.7, it is possible to notice, the reduction of the position error for growing values of the proportional gain K_p. It has been noticed that, for values higher than $K_p = 8$, a quite small variation of the position error. The behaviour of the motor does not change, due to the linear condition.
- by analysing the dependence to the parameter of the final time t_f (in linear condition, without any load), in figure 4.8, it is possible to notice that, the behaviour of the motor doesn't change, but the position error increase for lower final time.

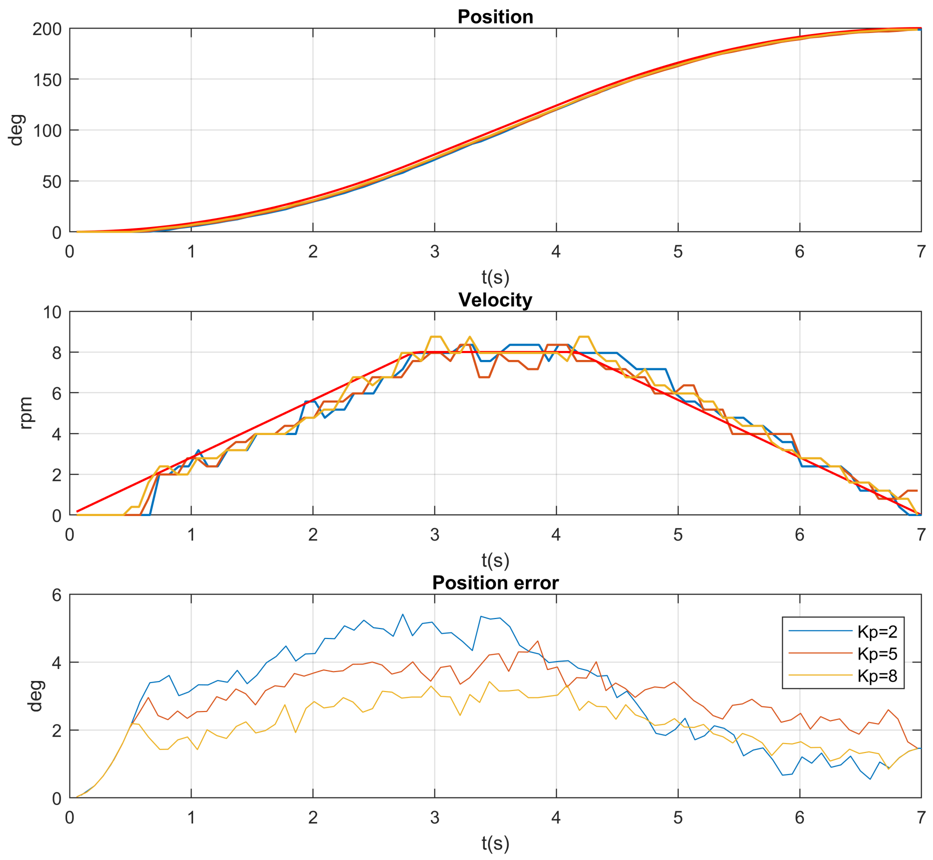


Figure 4.7: *Feed-forward position control, influence of K_p.*

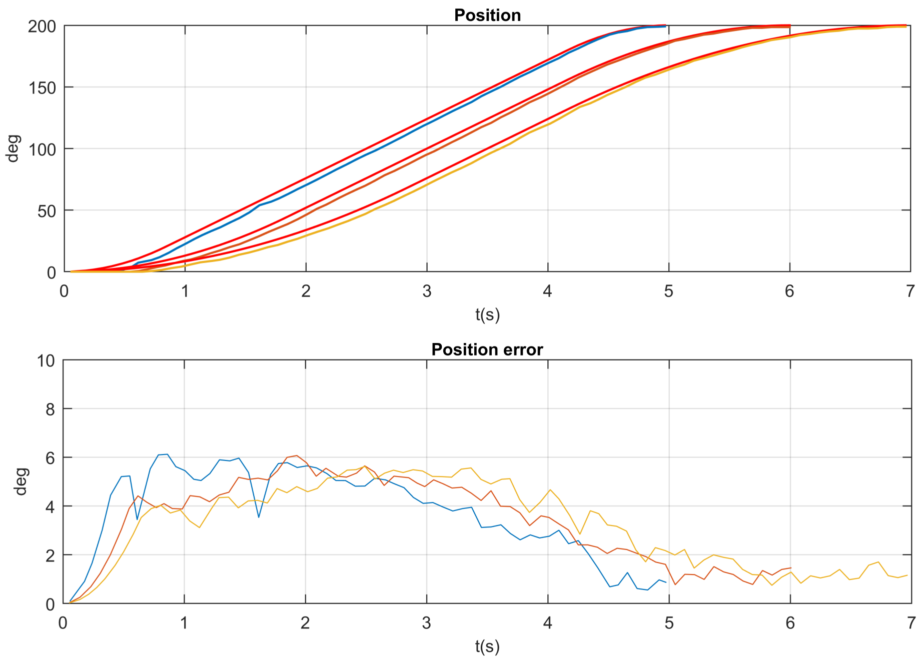


Figure 4.8: *Feed-forward position control, influence of t_f.*

The serial robot 3R allows us to make other important considerations. In fact it has been possible to plan a coordinate control motion with the three motors and make some considerations.
In figure 4.9, is shown the **coordinate control** of the three motor, in which it has been chosen the trapezoidal shape of velocity for the three motor, using the same time law parameters and gains. In figure 4.9, is possible to observe the behaviour of the three motors, called ID1, ID2 and ID3, enumerating the motor form the bottom to the top (referring to the figure 4.1). It has been planned a motion that, for each motor, provides to start from the zero position (that corresponds to the *zero configurations* of the robot) and displays a rotation of 100 deg following the sametime law.
This experiment has been useful to learn the coordinate control of the three motor but is useful to make other considerations. In fact, the same model of motor has been used in each joint of the 3R robot, this fact allows us to analyse the behaviour of the motor under a **no linear load**.

- it's possible to notice that, the motor ID3, presents the lowest position error, being affected only by the weight of the last link.
- by analysing the other two motor, it is possible to notice that the position error increase;
- The motor ID1 and ID2 have approximately the same value of the position error due to the structure of the mechanism.

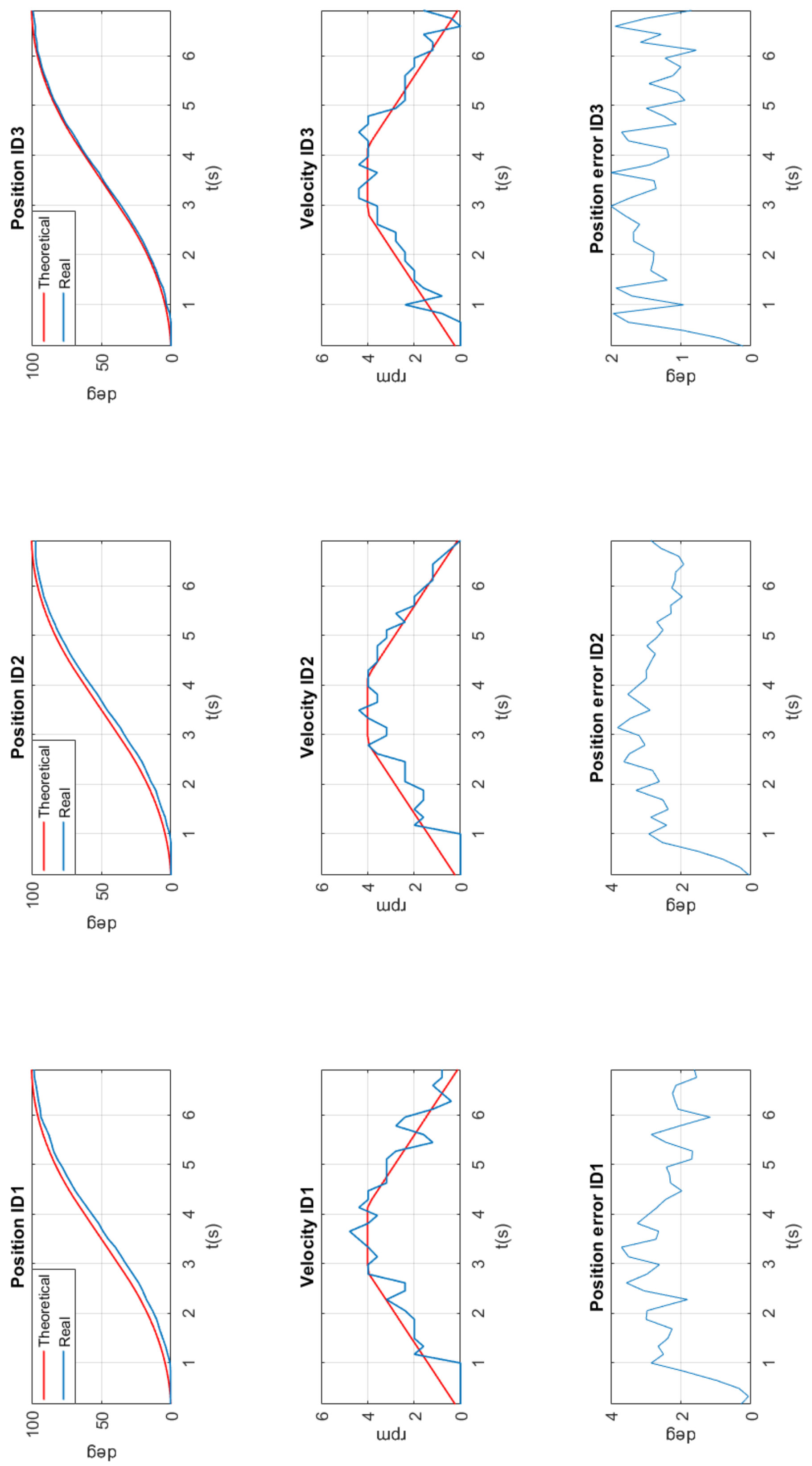


Figure 4.9: *Dynamixel AX-12A, coordinate control of three motor.*

4.2 Installed Motor: Robotis Dynamixel MX-64T

In the learning activity (Section 4.1), it has been possible to familiarize with the control of the *"Robotis Dynamixel"* servomotors. The implementation and installation of the motors has been developed after the assembly of the RCRCR mechanism.
Therefore, given the physical dimensions of the mechanism, with a qualitative evaluation of the necessary torque, the model *"Robotis Dynamixel MX-64T"* has been chosen. The specification are shown in table 4.3:

Figure 4.10: *Robotis Dynamixel MX-64T actuator.*

Dynamixel MX-64T Specification	
Operating Voltage	12V
Stall Torque	6.0 Nm
No-load Speed	63 rpm
Max Current	4.1 A at 12 V
Weight	126 g
Size	40.2 × 61.1 × 41.0 mm
Resolution	0.088 deg
Reduction Ratio	1/200
Operating Angle	360 deg or Continuous Turn
Standby Current	100 mA
Operating Temperature	-5°C - 80°C
Material	Metal Gears and Engineering Plastic Body
Motor	Maxon RE-MAX

Table 4.3: *Dynamixel MX-64T, motor specification [10].*

The motor working environment was the same of the motor used for the learning activity (being part of the same *"Robotis Dynamixel X"* servomotors family), shown in the Section 4.1.1. The advantage respect to the previous model was the possibility to control the motor with an inner implemented PID control algorithm that allows to control the motor response with different operating mode.

4.2.1 Control Table and functioning type of the motor Dyamixel MX-64T

As explained above, the **Control Table** is a structure of data implemented in the Dynamixel. Users can read a specific Data to get status of the Dynamixel with *Read* Instruction Packets, and modify Data as well to control Dynamixels with *Write* Instruction Packets.
In particular, for the motor Dynamixel MX-64T:

- **Control Table, Data, Address**: the Control Table is a structure that consists of multiple Data fields to store status of the Dynamixel or to control the Dynamixel. Users can check current status of the Dynamixel by reading a specific Data from the Control Table with Read Instruction Packets. *Write* Instruction Packets enable users to control the Dynamixel by changing specific Data in the Control Table. The Address is a unique value when accessing a specific Data in the Control Table with Instruction Packets. In order to read or write data, users must designate a specific Address in the Instruction Packet. Please refer to the Protocol section of e-Manual for more details about Packets.

- **Area (EEPROM, RAM)**: the Control Table is divided into 2 Areas. Data in the RAM Area is reset to initial values when the Dynamixel is turned on (Volatile). On the other hand, modified data in the EEPROM Area keeps their values even when the Dynamixel is turned off (Non-Volatile). Data in the EEPROM Area can only be changed when the value of Torque Enable(64) is cleared to 0.

- **Access**: the Control Table has two different access properties. RW property stands for read and write access permission while R stands for read only access permission. Data with the read only property cannot be changed by the write Instruction. Read only property(R) is generally used for measuring and monitoring purpose, and read write property(RW) is used for controlling Dynamixels.

- **Initial Value**: each data in the Control Table is restored to initial values when the Dynamixel is turned on. Default values in the EEPROM area are initial values of the Dynamixel (factory default settings). If any values in the EEPROM area are modified by a user, modified values will be restored as initial values when the Dynamixels is turned on. Initial Values in the RAM area are restored when the Dynamixels is turned on.

- **Size**: the Size of data varies from 1 to 4 bytes depend on their usage. Please check the size of data when updating the data with an Instruction Packet.

The most important items of the MX-64T Control Table are shown in table: 4.4 :

Control Table						
EEPRON Area						
Address	**Size [byte]**	**Data Name**	**Access**	**Initial Value**	**Min**	**Max**
7	1	ID	RW	1	0	252
8	1	Baud Rate	RW	1	0	7
11	1	Operating Mode	RW	3	0	16
13	1	Protocol Version	RW	2	1	2
24	4	Moving Threshold	RW	10	0	1023
RAM Area						
Address	**Size [byte]**	**Data Name**	**Access**	**Initial Value**	**Min**	**Max**
64	1	Torque Enable	RW	0	0	1
76	2	Velocity I Gain	RW	1920	0	16383
78	2	Velocity P Gain	RW	100	0	16383
80	2	Position D Gain	RW	0	0	16383
82	2	Position I Gain	RW	0	0	16383
84	2	Position P Gain	RW	850	0	16383
88	2	FF 2nd Gain	RW	0	0	16383
90	2	FF 1st Gain	RW	0	0	16383
104	4	Goal Velocity	RW	-	0	1023
108	4	Profile Acceleration	RW	0	0	32767
112	4	Profile Velocity	RW	0	0	1023
116	4	Goal Position	RW	850	0	4095
124	2	Present PWM	R	850	0	850
126	2	Present Current	R	-	0	1941
128	4	Present Velocity	R	-	0	1023
132	4	Present Position	R	-	0	4095
136	4	Velocity Trajectory	R	-	0	1023
128	4	Position Trajectory	R	0	0	1023

Table 4.4: *Dynamixel MX-64T, Control Table [10].*

The functioning of the motor can be selected by changing the value, in address of the EEPRON area, of the *Operating Mode*. The motor can work in different control mode:

- **Current control mode:** this mode controls current(torque) regardless of speed and position. This mode is ideal for a gripper or a system that only uses current(torque) control or a system that has additional velocity/position controllers;
- **Velocity control mode:**this mode controls velocity;
- **Position control mode:** this mode controls position. Operating position range is limited by maximum position limit and minimum position limit. This mode is ideal for articulated robots that each joint rotates less than 360 degrees;
- **Extended Position Control Mode(Multi-turn):** this mode controls position. This mode is ideal for multi-turn wrists or conveyer systems or a system that requires an additional reduction gear;
- **Current-based Position Control Mode:** this mode controls both position and current(torque). This mode is ideal for a system that requires both position and current control such as articulated robots or grippers;

UNITS TABLE CONVERSION			
Physical Quantities	**Physical Range**	**Address Range**	**Units**
Voltage	9.5-16.0 V	95-160	0.1 V
PWM	0-100%	0-850	0.118 %
Current	0-6.5A	0-1941	3.36 mmA
Acceleration	-	0-32736	214.577 rpm2
Velocity	0-234 rpm	0-1023	0.229 rpm
Position	0-360 deg	0-4095	0.088 deg
Velocity I Gain	0-0.25	0-16383	65536
Velocity P Gain	0-128	0-16383	128
Position D Gain	0-1023	0-16383	16
Position I Gain	0-0.25	0-16383	65536
Position P Gain	0-128	0-16383	128
FF 2nd Gain	0-4095	0-16383	4
FF 1st Gain	0-4095	0-16383	4

Table 4.5: *Dynamixel MX-64T, table of conversion [10].*

- **PWM control mode:** this mode directly controls PWM output. (Voltage Control Mode)

4.2.2 Operating mode chosen: position control mode

In order to realize the desired motion, the **Position control mode** has been used. In fact, the requirement of the motion in the prototype was to display a certain rotation of the joints in which is placed a motor with an high accuracy on the final position.

With the chosen control mode we can realize the feed-forward position control, taking advantage of the implemented controller inside the motor, shown in figure 4.11:

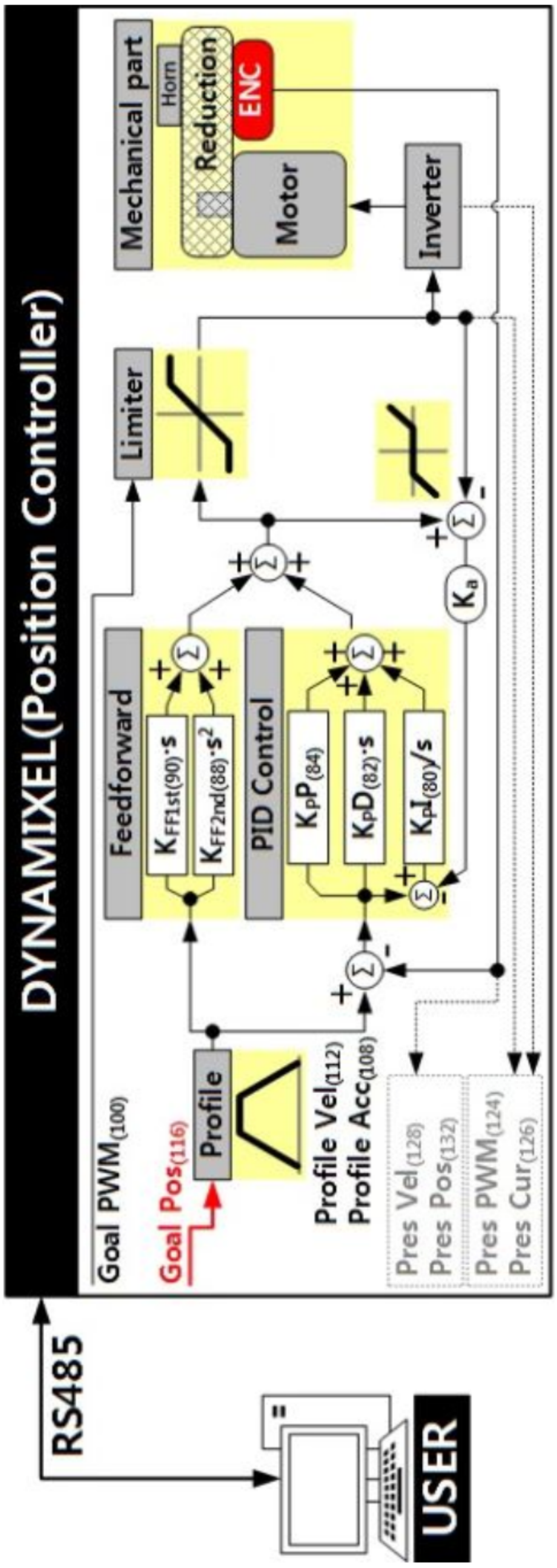


Figure 4.11: *Dynamixel Motor MX-64T, Position Control Mode block diagram [10].*

Where the gains used are:

- k_P proportional gain;
- k_D derivative gain;
- k_I integrative gain;
- k_{FF1} feedforward velocity gain;
- k_{FF2} feedforward acceleration gain.

The operating mode *"Position Control Mode"* corresponds to a properly feed-forward position control as shown in figure 4.11. When the instruction from the user is received, the algorithm takes the following steps until driving the horn:

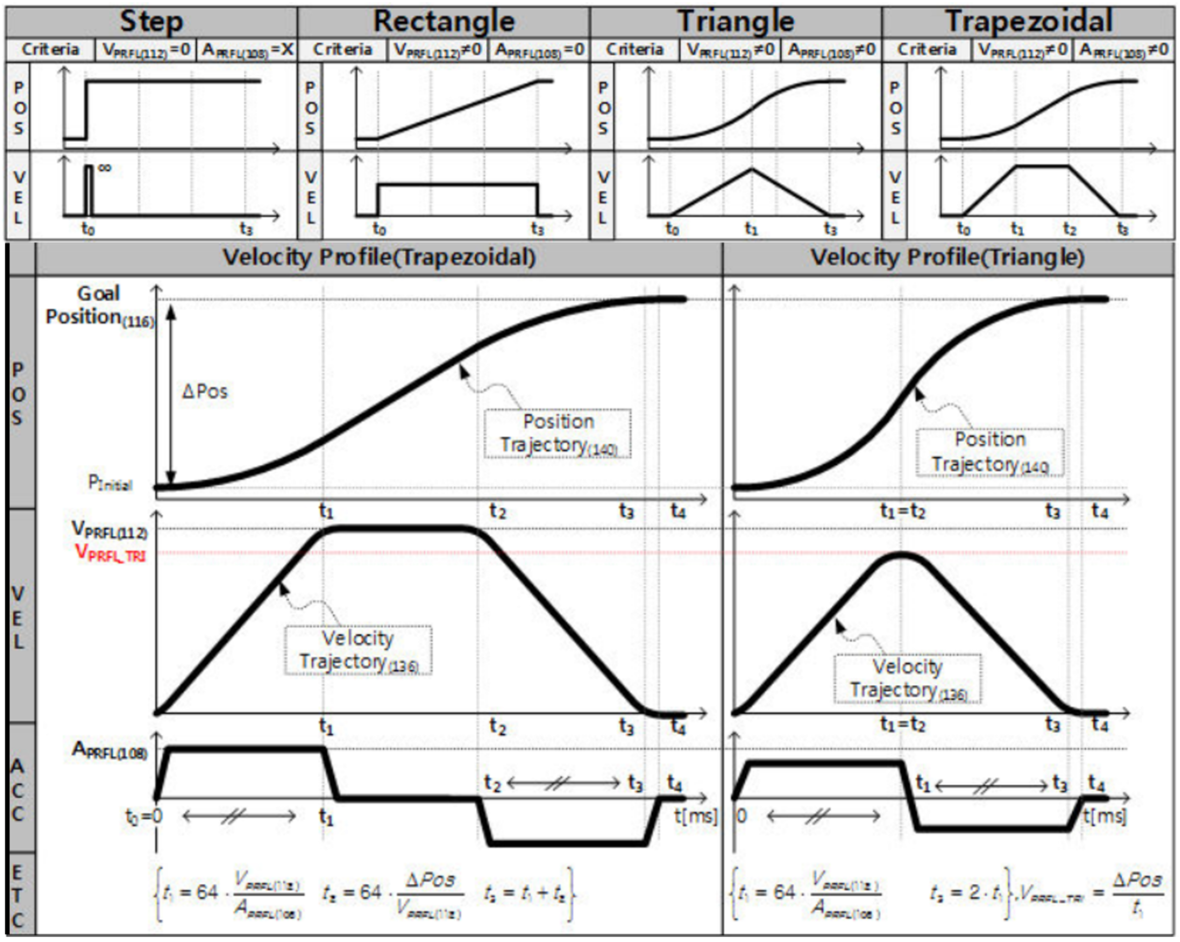
- An Instruction from the user is transmitted via Dynamixel bus, then registered to *Goal Position*;
- *Goal Position* is converted to target position trajectory and target velocity trajectory by *Profile Velocity* and *Profile Acceleration*;
- The target position trajectory and target velocity trajectory is stored at *Position Trajectory* and *Velocity Trajectory* respectively;
- Feedforward and PID controller calculate PWM output for the motor based on target trajectories;
- *Goal PWM* sets a limit on the calculated PWM output and decides the final PWM value;
- The final PWM value is applied to the motor through an Inverter, and the horn of Dynamixel is driven;
- Results are stored at *Present Position*, *Present Velocity*, *Present PWM* and *Present Current.*

The Profile is an acceleration/deceleration control method to reduce vibration, noise and load of the motor by controlling dramatically changing velocity and acceleration. It is also called Velocity Profile as it controls acceleration and deceleration based on velocity. The motor provides 4 different types of Profile. Profiles are usually selected by a combination of *Profile Velocity* and *Profile Acceleration.* Triangular and Trapezoidal Profiles exceptionally consider total travel distance (ΔPos, the distance difference between target position and current position) as an additional factor. For convenience, *Profile Velocity* is abbreviated to V_{PRFL} and *Profile Acceleration* is abbreviated to A_{PRFL}.

When given *Goal Position*, Dynamixel's profile creates target velocity trajectory based on current velocity(initial velocity of the Profile). When Dynamixel receives updated target position from a new Goal Position while it is moving toward the previous Goal Position, velocity smoothly varies for the new target velocity trajectory. The following explains how Profile processes *Goal Position* instruction in Position Control mode adopted.

- An Instruction from the user is transmitted via Dynamixel bus, then registered to *Goal Position.*

- Acceleration time(t_1) is calculated from *Profile Velocity* and *Profile Acceleration.*
- Types of Profile is decided based on *Profile Velocity*, *Profile Acceleration* and total travel distance(ΔPos, the distance difference between target position and current position)
- Selected Profile type is stored at *Moving Status.*
- Dynamixel is driven by the calculated target trajectory from *Profile.*
- Target velocity trajectory and target position trajectory from Profile are stored at *Velocity Trajectory* and *Position Trajectory* respectively.
- $V_{PRFL-TRI}$ and *Travel time* (t_3) to reach *Goal Position* is calculated as in figure 4.12.



$$\left\{ t_1 = 64 \cdot \frac{V_{PRFL(112)}}{A_{PRFL(108)}} \quad t_2 = 64 \cdot \frac{\Delta Pos}{V_{PRFL(112)}} \quad t_3 = t_1 + t_2 \right\}$$

$$\left\{ t_1 = 64 \cdot \frac{V_{PRFL(112)}}{A_{PRFL(108)}} \quad t_3 = 2 \cdot t_1 \right\}, V_{PRFL\_TRI} = \frac{\Delta Pos}{t_1}$$

Figure 4.12: *Set of the Velocity profile, Dynamixel Motor MX-64T [10].*

4.2.3 Considerations on the motor behaviour

Using the *Position Control* implemented it has been possible to investigate on the behaviour of the motor. By setting the shape parameters in Matlab environment is possible to make several consideration on the motor behaviour.In the following graphs, it has been setted a trapezoidal shape of velocity as time law

In figure 4.13 is shown the response in a linear condition, without any load, trapezoidal shape of velocity has been setted as time law. The goal of the control is to **achieve the final position** with the higher possible accuracy. In fact, the mechanism does not require a low error during the motion but, in order to follow the cycle of configuration provided by the theoretical study, is important to reduce the final position error, i.e. obtain a certain configuration of the mechanism.

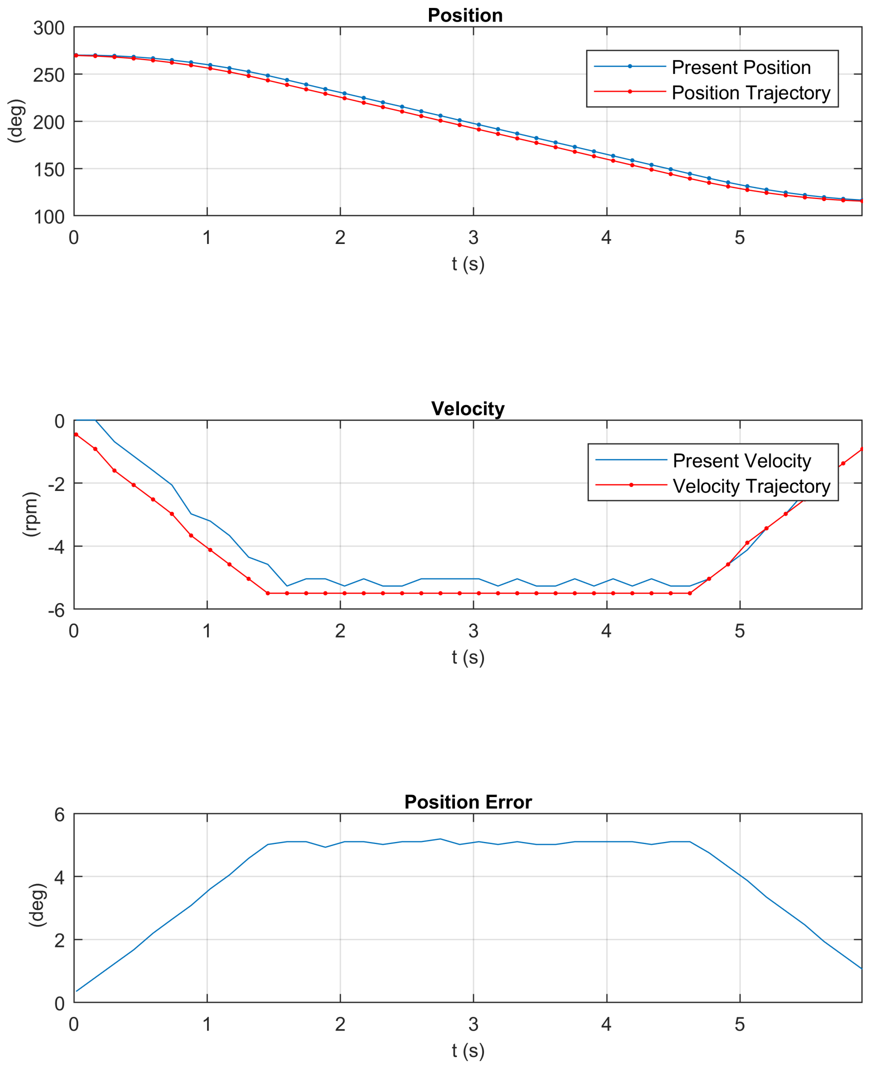


Figure 4.13: *Feed Forward Position control, trapezoidal sahpe of velocity.*

By using an external script in Matlab environment, it is possible to set the time law parameters managing the general parameters as t_3 (referring to the figure 4.12) or ΔPos and calculating the parameters of V_{PFRL} and A_{PFRL}. In particular in figure 4.14 is possible to notice that, for decreasing value of the t_3 it is possible to observe the growth of the position error. Even if the test shown in figure 4.14 has been carry out in linear condition, it should be noticed that, in order to have an acceptable final error in this condition, is necessary to plan a motion at least of 5 s.

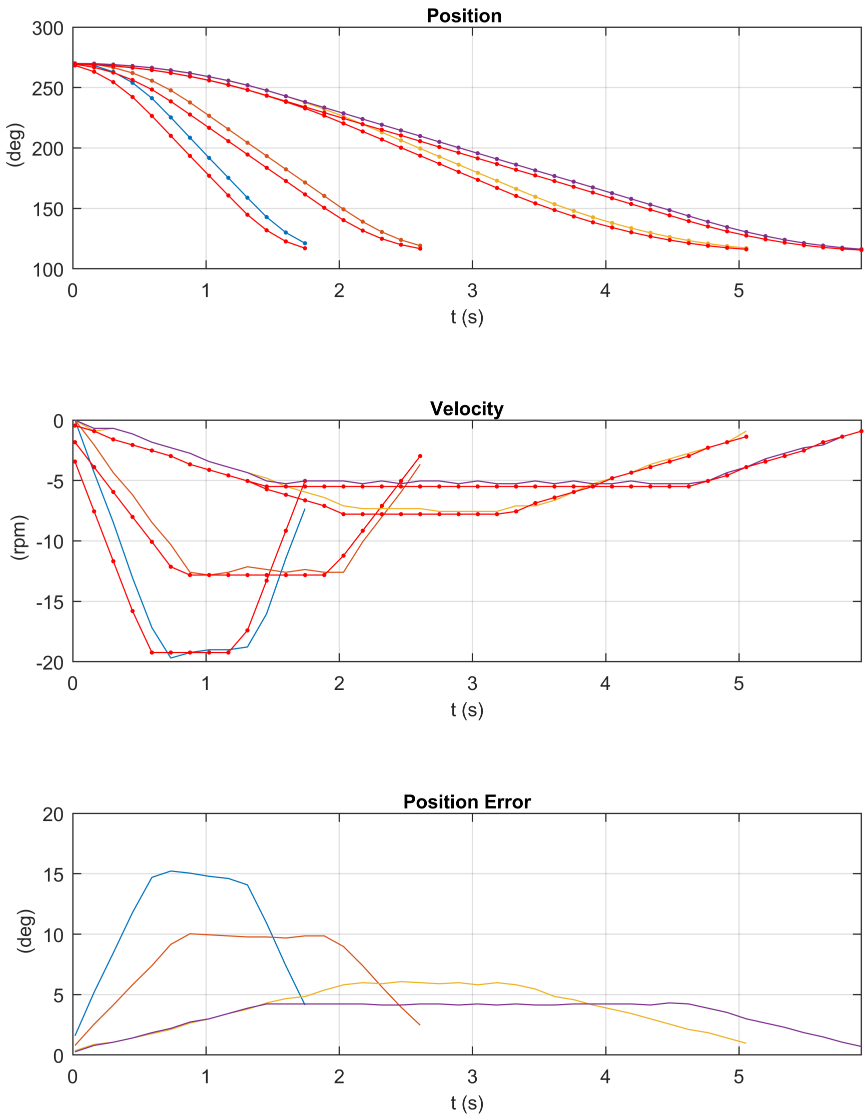


Figure 4.14: *Feed Forward Position control, influence of the final time.*

In figure 4.15 is possible to notice the behaviour of the motor with increasing value of the K_I. Even if the test has been carry out in linear condition without any load is possible to see the reduction of the position error for growing value of the integrative gain, starting from a no-integrative control ($K_I = 0$).

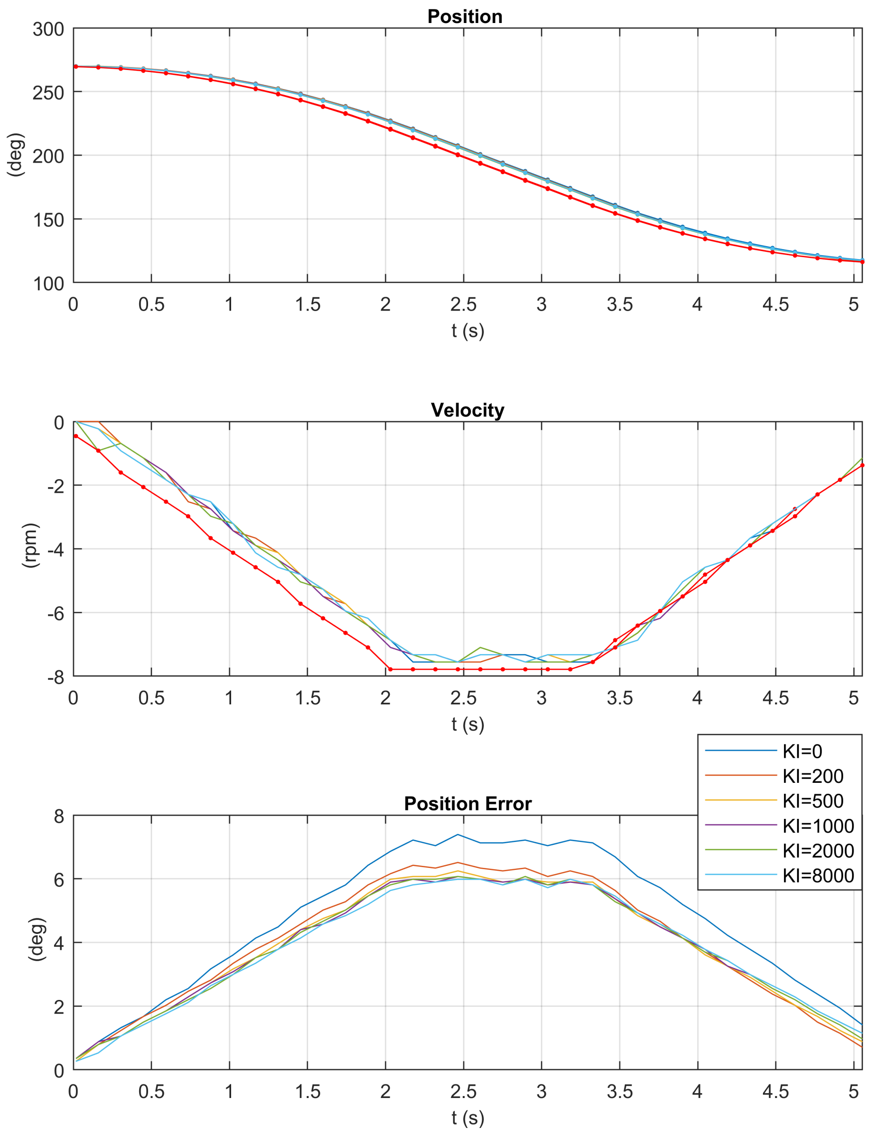


Figure 4.15: *Feed Forward Position control, influence of K_I.*

4.2.4 Feed-forward position control implemented: linear condition

Using the feed-forward position control described above is possible to implement, in Matlab environment, a coordinated control that allows to perform the complete cycle of configuration. In the following section will be reported the results obtained with the motor tested on the bench in a no-load condition. In fact, for the reasons shown in the Section 3.10 it has been impossible to test the whole motion of the mechanism with the motors installed. Therefore, as follows, are shown approximated results without any consideration on the PID and feed-forward parameters that, require an appropriate tuning once get a correct functioning of the mechanism motion.
As explained in the Section 1.7, the mechanism can be driven using only one motor but, to perform a complete (closed) cycle of configurations needs an additional motor, that allows to overtake the singularity configurations given by the principal driving pair.

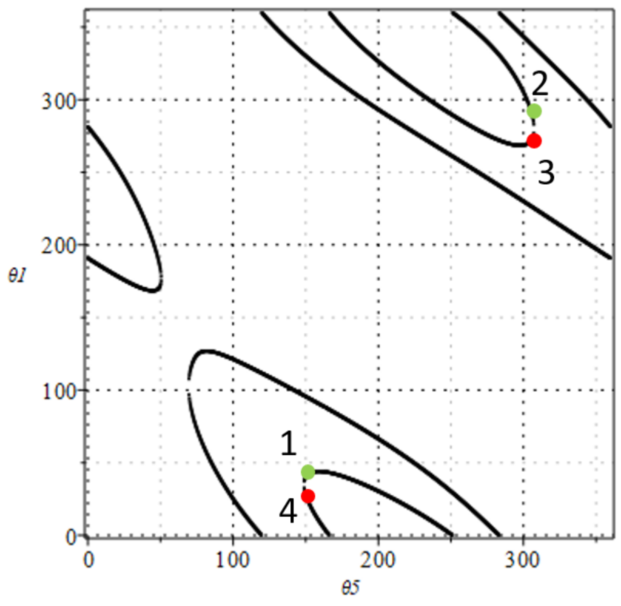


Figure 4.16: *Plot relationship θ_5-θ_1*

Referring to the figure 4.16, the initial position of the mechanism corresponds to the **point 1** (θ_5=148.96 deg,θ_1=37.30 deg). Next, it has been implemented a motion to compute the complete cycle of configuration, as follows:

- drive the motor placed in J_5, displaying a rotation of +154,7 deg, until the **point 2** (θ_5=303.66 deg,θ_1=296.66 deg);
- drive the motor placed in J_1, displaying a rotation of -26,4 deg, until the **point 3** (θ_5=303.66 deg,θ_1=270.24 deg);
- drive the motor placed in J_5, displaying a rotation of -154,7 deg, until the **point 4** (θ_5=148.96 deg,θ_1=32.52 deg);
- drive the motor placed in J_1, displaying a rotation of +4.72 deg, until the **point 1** (θ_5=148.96 deg,θ_1=37.30 deg);

This control allows to control the motor checking the position in real time, that is appropriate for our proposes. However, the use of *Position Control Mode* that, it not implies the multi-turn possibility adds an additional constrain for the algorithm. In fact, for both joints, is necessary to provide an initial position of the motor and then install the motor to the mechanism, making sure to not overtake the value of 360 deg or 0 deg during the motion.A crossing of the zero of the motor leads to a sudden inversion of the rotation direction. The same kind of control has been used for the motor placed in the joint J_5 and J_1.

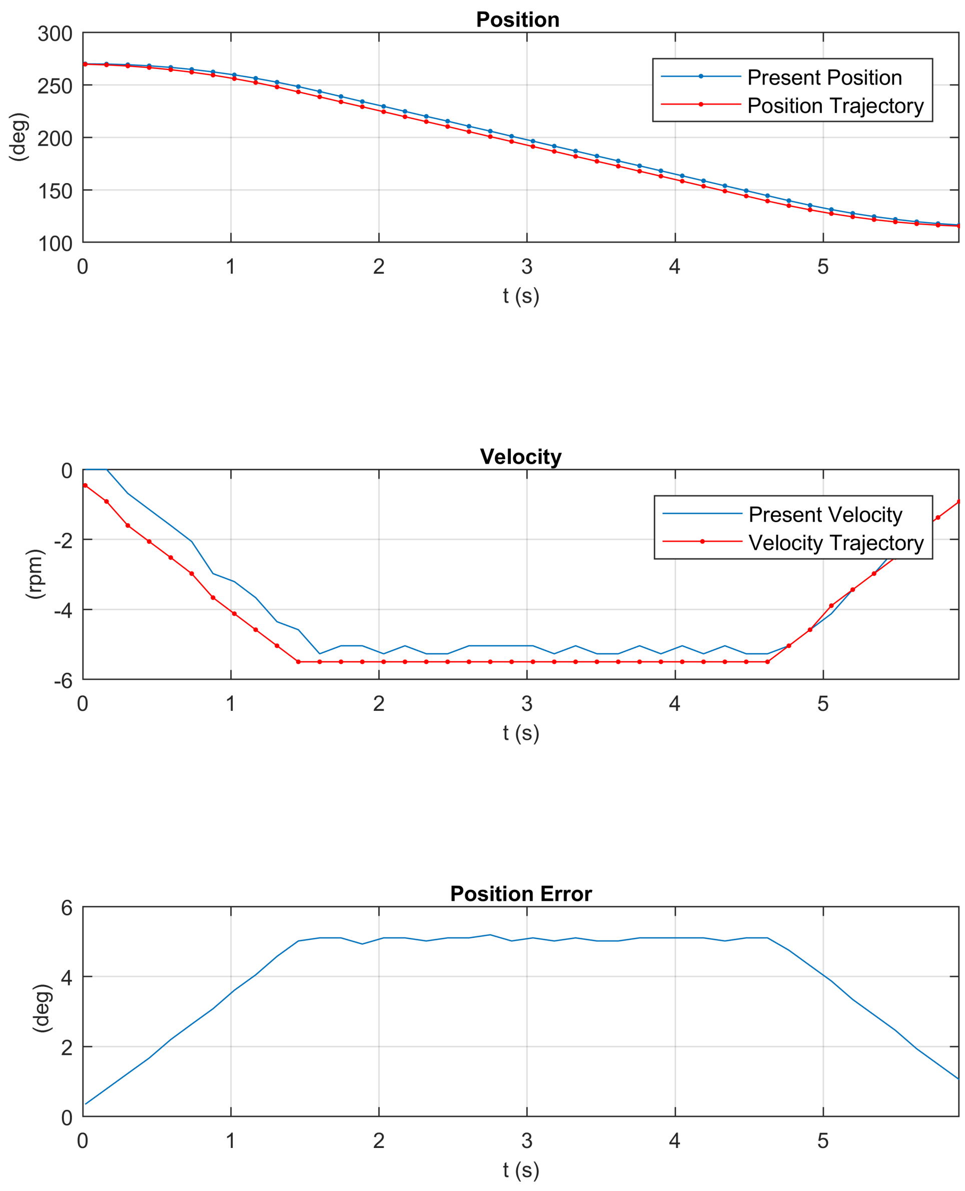


Figure 4.17: *Feed Forward Position control J_5 motor (θ_5), from the configuration 1 to 2.*

Referring to the figure 4.17, the initial position of the motor has been setted to 270 deg. The implemented algorithm in Matlab environment allows to install the motor once get the position of 270 deg. In fact, J_5 must display a rotation of about 154.7 deg in counter-clockwise direction (coherently with the rotation sense of the motor). A trapezoidal shape of velocity has been chosen as motion law. In our case, are not required particular dynamic feature but an accuracy on the final position of the motion. Therefore, any kind of consideration on the tuning of the parameters are leaved as future works, once get the correct functioning of the mechanism. At the end of the motion shown in figure 4.17, the mechanism achieve the point 2.

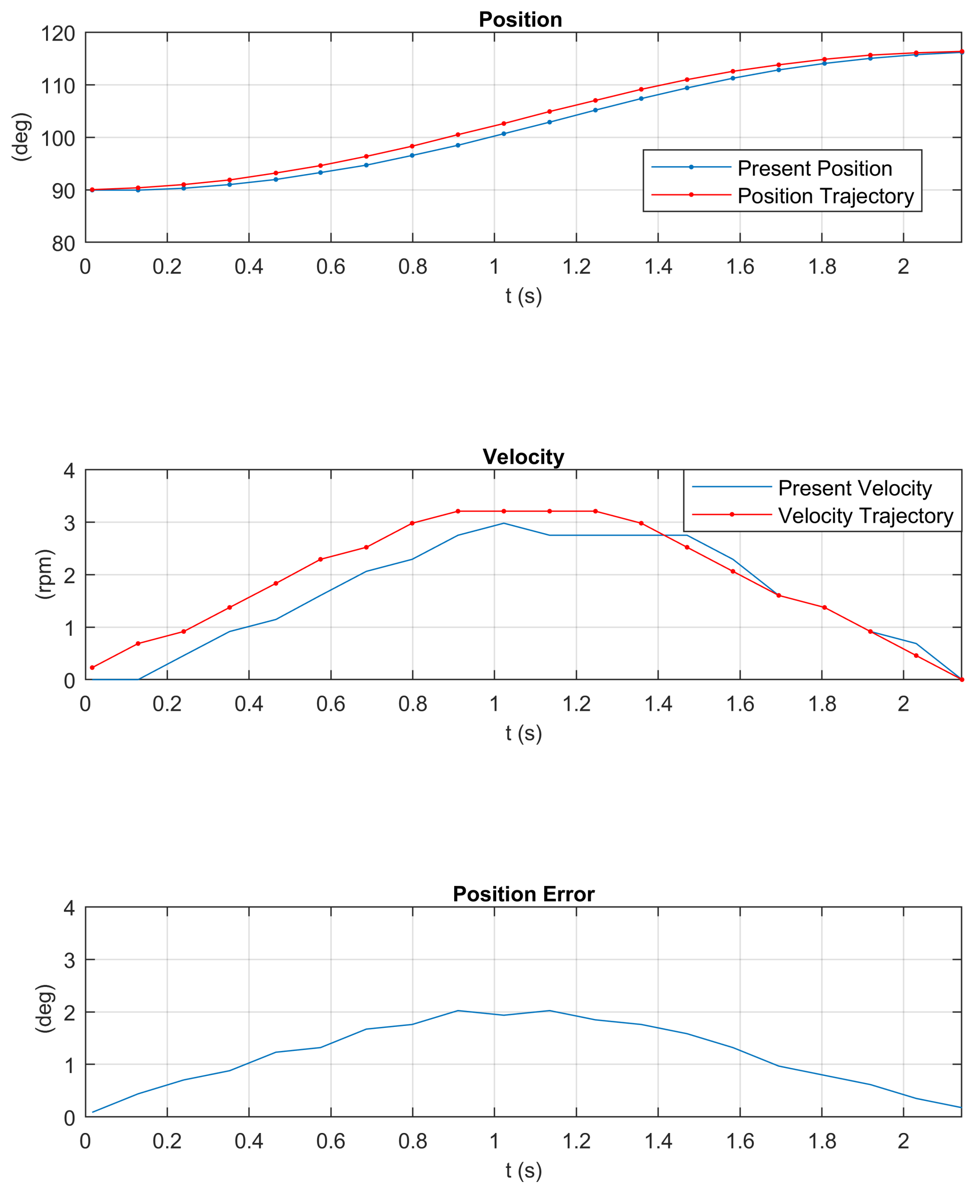


Figure 4.18: *Feed Forward Position control J_1 motor (θ_1), from the configuration 2 to 3.*

Referring to the figure 4.18, changing driving motor, is possible to overtake the singularity condition (for θ_5). Therefore, using the motor placed in J_1 is possible to display a rotation of 26.4 deg achieving the point 3.
It must be noted that,it has been chosen a **trapezoidal shape of velocity** as time law, and also in this case, the requirement consists in the accuracy of the final position. In particular, in both cases, the final error is comparable with the resolution of the encoder.

The other two path of the motion are not report for sake of brevity but the path of trajectories will be the same for both joints,with the only of the reverse direction of rotation of the motors.

4.2.5 Feed-forward position control implemented: test on the mechanism

As explained above, the mechanism is not able to perform the complete range of motion driven by motors. In this section is shown the experimental data collected that confirm what provided by the final diagnosis (Section 3.10). In fact, aside from the experimental observations, it has been possible to test the mechanism with the motors installed, observing the behaviour as following explained.

In figure 4.19 is shown the behaviour of the motor installed on the joint J_5, in the first part of motion (graphs in no load condition in figure 4.17). It was possible to notice that, the correct motion of the mechanism has been obtained until the instant $t = 2.1$ s.
In that instant the position of the motor placed in J_5 presents a value of 246.1 deg that, equivalent to the value of the linkage variable $\theta_5 = 172,86$ deg.

Approximately, after that configuration the phenomena of jamming, linked to the friction and inner tensions, deny the correct motion. In fact, after that instant the value of *Present Position* read, does not change, increasing the error. In that instant, the torque given by the motor, achieve the maximum value and try to move the mechanism providing the maximum value for the following seconds.

Between the instant 2.1 s and 4.4 s, it can be observed that, the position error increase until unacceptable values, and the present toque remains approximately constant. Due to the high value of requested torque (current), starting from the instant of 4.6 s, the motor enter in a *safe mode* to preserve the components. This leads to a turn-off of the driving system making not valid the data collected after that instant and confirming what provided by the experimental observation.

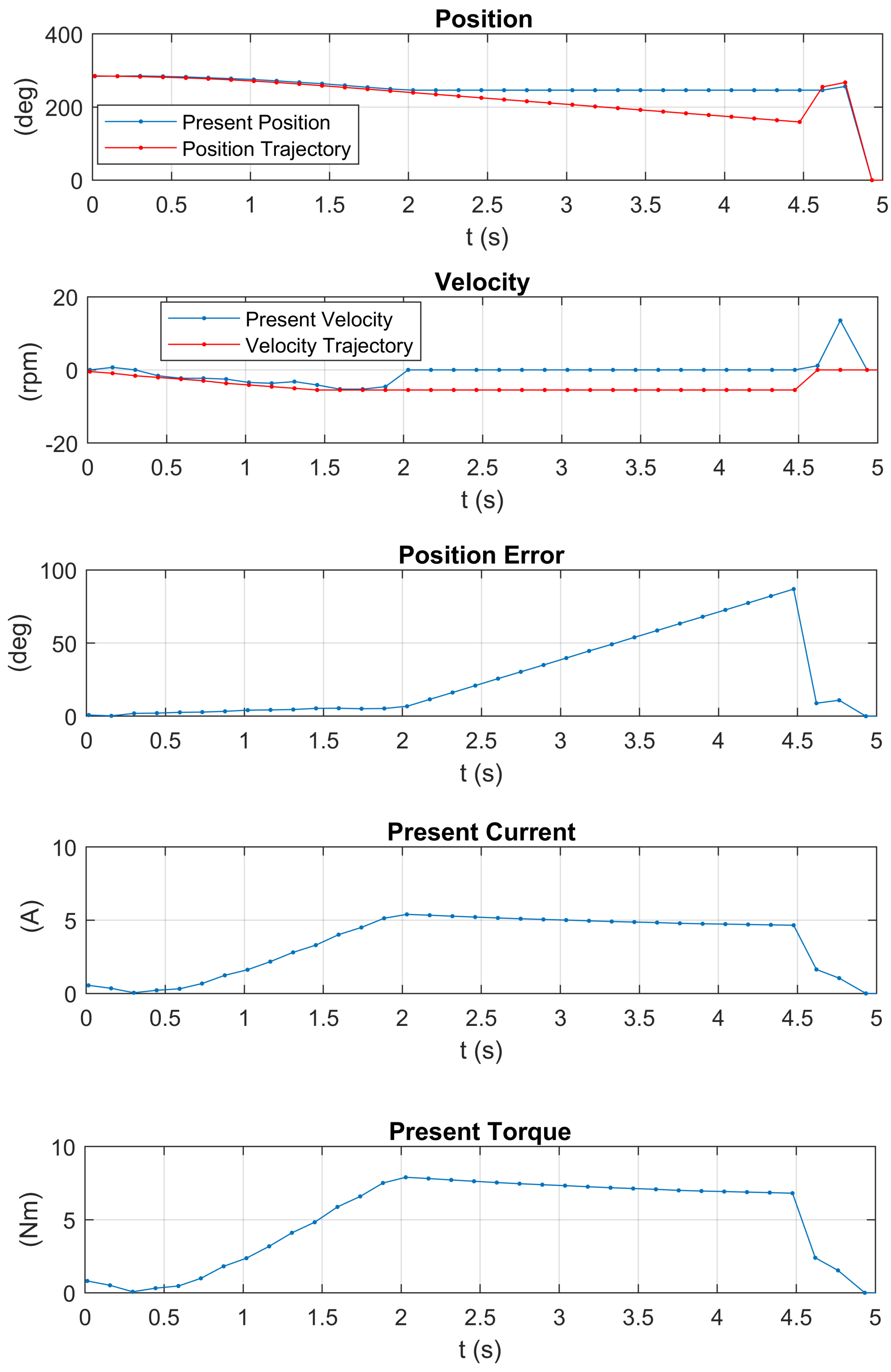


Figure 4.19: *Feed Forward Position control, test on the mechanism, motor in J_5.*

Chapter 5

Cost Analysis

The thesis has been carried out in *"Institut de Robotica i Informatica Industrial (I.R.I.)"* taking advantage of the laboratory tools.
As explained in the previous Chapters, all the mechanism components have been printed with 3D printers using the FDM^{TM} Technology (fused deposition modelling) in ABS-P430. Therefore, for the printed components a general assessment has been estimate, involving cost of material, energy supply and labour cost. For the commercial component, reference was made to the [11], except for the Motors, bought from the seller website.

Cost of Materials			
Component	Unit Cost	Quantity	Cost (€)
SKF-radial ball bearing-61802	10,07 €	6	60,42
Igus-DryLin ®Linear Plain Bearings-RJUM-01-12	14,22 €	2	28,44
Igus-DryLin ®shaft-AWMP-12	30,29 €	2	60,58
Dynamixel MX-64T	299,9 €	2	599,8
Carbon fiber bar (1000 mm)	29,9 €	2	59,8
ABS-P430 Components	82 €/kg	Estimate	200
Total Costs			1010

Moreover, we need to take into account the additional cost, i.e., the manpower required to carry out the project and the amortization of the used tools.

Additional Costs			
Cost items	€/h	Hours (h)	Cost €
Engineering Manpower	12	750	9000
3D printer amortization	5	70	350
Laboratory tools amortization	0,6	750	450
Total Costs			9800

Therefore, the final cost that the project has required is shown in the following table:

Budget of the project	
Material Cost	1010 €
Additional Cost	9800€
Total cost	10810 €

Conclusions

In this final chapter we summarize the goals achieved and propose some directions of future works.

Thesis Contributions

This work has had as principal goal the design of the appropriate shape of links in order to satisfy the motion along the full cycle of configurations of a RCRCR mechanism. Thus, the final goal has been the physical construction of the mechanism assembly able to perform the whole mobility range by actuating one servo-motor at a time.

In order to design the shape of each link, firstly, through the use of Maple, the numerical solutions have been obtained, defining all the possible configurations of the RCRCR mechanism, as known in literature [3, 1, 15].

The results obtained have showed that, in the real construction of the mechanism, it is possible to move across the extreme positions only through the use of two motors, due the singularities found in these configurations.

Then, without taking into account the physical envelope, the links shapes have been designed, using an implemented algorithm to checks the possible collisions between links, allowing the mechanism to move along all cycle of configurations of the selected assembly mode.

Since the previous study is valid in general, independently of a value assigned of the length unit, a convenient scale has been chosen that allows its construction with a 3D printer (in ABS-plastics) keeping the costs low of the commercial functional elements.

Subsequently, the rotational and the cylindrical pairs have been designed following the 3D printing guidelines. Thus, links have been designed according to the theoretical study providing rigidity and robustness to support the forces applied to them by the driving input pairs.

Finally, a motor control has been planned. Even if, the control implemented provide the position control for the whole range of mobility, it was not possible to implement the real motion o the mechanism due to the practical problems explained in the report.

Future Works

Even though the results achieved have been satisfactory, further work can be done. Some proposals for future works are:

- *improving the design of the mechanism*: the design implemented allows to achieve each possible configuration of the assembly mode 2, by positioning manually the linkage variable but not to actuate the mechanism with the motors. A re-design of the components (in particular the chunks) can improve the overall rigidity of the mechanism and allows the correct motion;
- *performing the dynamic analysis of the mechanism*: once we get the RCRCR model, a dynamic analysis would be necessary in order to optimize torques and velocities required;
- *implementing the control of two motors to drive the mechanism along the whole cycle of configurations*: perform the whole cycle of configurations driving the mechanism with two motor. Driving two linkage variables with a coordinate control, will increase the overall accuracy;
- *perform an equivalent study for the mode of assembly 1 and try to find a design for the links that allows their use in both modes of assembly.*

Attachments

Appendix A

Dual Numbers

A dual number $\hat{x}$ is defined as the sum of a real and a dual component:

$$\hat{x} = x + \epsilon x_0$$

The dual component is a multiple of the dual unit ϵ, which by definition has the property of $\epsilon^2 = 0$. The sum and product of two dual numbers are given by:

$$\begin{aligned} \hat{x} + \hat{y} &= (x + \epsilon x_0) + (y + \epsilon y_0) = (x + y) + \epsilon(x_0 + y_0) \\ \hat{x}\hat{y} &= xy + \epsilon(xy_0 + yx_0) \end{aligned}$$

The trigonometric function *sin* and *cos* of a dual variable are:

$$\begin{aligned} sin\hat{x} &= sin\ x + \epsilon x_0 cos\ x \\ cos\hat{x} &= cos\ x - \epsilon x_0 sin\ x \end{aligned}$$

A spatial transformation involving a translation of a vector $v = [v_x, v_y, v_z]^T$ and a rotation R can be represented by a dual-number rotation matrix $\hat{R} = R{+}\epsilon D$, where the real component R is an orthogonal matrix corresponding to the rotation part, and the dual component is $D = P_v R$, where P_v is a skew-symmetric matrix obtained form the coordinates of v as:

$$P_v = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

In particular, dual rotation about the x and z axes describing a pair of dual angle $\hat{\theta_i}$ and a link of dual angle $\hat{\alpha_{ij}}$ respectively are:

$$\hat{R}x(\hat{\theta_i}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\hat{\theta_i} & -sin\hat{\theta_i} \\ 0 & sin\hat{\theta_i} & cos\hat{\theta_i} \end{bmatrix}$$

$$\hat{R}z(\hat{\alpha_{ij}}) = \begin{bmatrix} cos\hat{\alpha_{ij}} & -sin\hat{\alpha_{ij}} & 0 \\ sin\hat{\alpha_{ij}} & cos\hat{\alpha_{ij}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix allow us to describe the pose of a reference frame wrt another using a 3×3 matrix insted the classics 4×4 ones.

Dual Euler's decomposition of a spatial transformation

The Euler decomposition allow espressing any 3D rotation as the product of the three rotations about $x - z - x$ axes[1] :

$$R = Rx(\varphi)Rz(\phi)Rx(\psi)$$

Similarly, we can also express any spatial transformation, as a product of three dual angle rotations along the $x - z - x$ axes. Thus, a spatial transformation involving a translation $v = [v_x, v_y, v_z]^T$ and a rotation R can be represented by:

$$\hat{R} = \hat{R}x(\hat{\varphi})\hat{R}z(\hat{\phi})\hat{R}x(\hat{\psi}) = R + \epsilon D$$

Where:

$$\hat{\varphi} = \varphi + \epsilon p \quad \hat{\phi} = \phi + \epsilon q \quad \hat{\psi} = \psi + \epsilon r$$

With p, q and r may be obtained from the relation:

$$P_v = DR^{-1}$$

Appendix B

```
# Dual_Number.txt

# rutines per numeros duals.

dtx := proc(a) #### (a) es un numero dual de la forma: dual(r,d).
    array(1..3,1..3,[[1, 0, 0],[0, dcos(a),-dsin(a)],[0, dsin(a), dcos(a)]])
end:

dty := proc(a)
    array(1..3,1..3,[[dcos(a), 0, dsin(a)],[0, 1, 0],[-dsin(a), 0, dcos(a)]])
end:

dtz := proc(a)
    array(1..3,1..3,[[dcos(a),-dsin(a), 0],[dsin(a), dcos(a), 0],[0, 0, 1]])
end:

# Suposo que un numero dual ve en forma de funcio: dual(r,d)
partreal:=proc(ex)
    if type(ex,function) and op(0,ex)='dual' then
        op(1,ex);
    elif type(ex,function) and op(0,ex)='dsin' then
        sin(partreal(op(1,ex)))
    elif type(ex,function) and op(0,ex)='dcos' then
        cos(partreal(op(1,ex)))
    elif type(ex,equation) then
        partreal(op(1,ex))=partreal(op(2,ex))
 ###    elif (type(ex,list) and nops(ex)=2)  or type(ex,Vector) then
 # (no) Elimino els vectors.
###     ex[1];
    elif type(ex, array) or type(ex,list) then
        map(partreal,ex)
    elif type(ex,`+`)  then
        partreal(op(1,ex))+partreal(ex-op(1,ex))
    elif denom(ex) <> 1 then
        partreal(numer(ex))/partreal(denom(ex))
    elif type(ex,`*`)  then
        partreal(op(1,ex))*partreal(ex/op(1,ex))
    elif type(ex,`**`) then
        partreal(op(1,ex))**op(2,ex)
    elif type(ex,numeric) or type(ex,constant) then
        ex
    else 'partreal(ex)'
    fi;
end:

partdual:=proc(ex)
    if type(ex,function) and op(0,ex)='dual' then
        op(2,ex);
    elif type(ex,function) and op(0,ex)='dsin' then
         partdual(op(ex))*cos(partreal(op(ex)))
    elif type(ex,function) and op(0,ex)='dcos' then
        -partdual(op(ex))*sin(partreal(op(ex)))
    elif type(ex,equation) then
        partdual(op(1,ex))=partdual(op(2,ex))
#   elif type(ex,list) and nops(ex)=2 then  # or type(ex,vector)
# Elimino els vectors.
#         ex[2]
    elif type(ex, array) or type(ex,list) then
        map(partdual,ex)
    elif denom(ex) <> 1 then
         (partreal(numer(ex))*partdual(denom(ex)) -
          partdual(numer(ex))*partreal(denom(ex)))
                    /partreal(denom(ex))**2
    elif type(ex,`*`) or type(ex,`**`) then
        partreal(op(1,ex))*partdual(ex/op(1,ex))+
        partdual(op(1,ex))*partreal(ex/op(1,ex))
    elif type(ex,`+`) then
        partdual(op(1,ex))+partdual(ex-op(1,ex))
    elif type(ex,numeric) or type(ex,constant) then
        0
    else 'partdual(ex)'
    fi;
end:

raddeg :=proc(x)
    if x < 0 then evalf(360+x*180/Pi); else evalf(x*180/Pi); fi;
end:



# Definicio de rotacions 3D (reals):
tx:= proc(a) array(1..3,1..3,[[1, 0, 0],[0, cos(a), -sin(a)],[0, sin(a), ...
                            cos(a)]]) end:
ty:= proc(a) array(1..3,1..3,[[cos(a), 0, sin(a)],[0, 1, 0],[-sin(a), 0,...
                             cos(a)]]) end:
tz:= proc(a) array(1..3,1..3,[[cos(a), -sin(a), 0],[sin(a), ...
                            cos(a), 0],[0, 0, 1]]) end:


# Definicio de rotacions+traslacio (cilindric) 3D (homogeneas):
Tx:= proc(a,s) array(1..4,1..4,[[1, 0, 0, s],[0, cos(a), -sin(a), 0]...
            ,[0, sin(a), cos(a), 0],[0, 0, 0, 1]]) end:
Ty:= proc(a,s) array(1..4,1..4,[[cos(a), 0, sin(a), 0],[0, 1, 0, s],...
                [-sin(a), 0, cos(a), 0],[0, 0, 0, 1]]) end:
Tz:= proc(a,s) array(1..4,1..4,[[cos(a), -sin(a), 0, 0],[sin(a),...
             cos(a), 0, 0],[0, 0, 1, s],[0, 0, 0, 1]]) end:

# Definicio de la matriu P corresponent a la traslacio d'una rot
##dual (Pennock-Yang 1984):
# matriu dual A: si la part real es R, la part dual = PR, de manera que A=R+ePR
P:=proc(x,y,z) array(1..3,1..3,[[0, -z, y],[z, 0, -x],[-y, x, 0]]) end:

# funcio per transposar (invertir) una rotacio 3x3.
Array3x3Transpose:= proc(A)
    array(1..3,1..3,[[A[1,1], A[2,1], A[3,1]],
             [A[1,2], A[2,2], A[3,2]],
             [A[1,3], A[2,3], A[3,3]]])
end:
```

```
### Essential_func.txt#########################

# R3C2_IO :: Compuation of valid configurations for a given input value of the
 RCRCR mechanism.
```

```

# WARNING! the names of the coefficients appearing in the equations may differ
 from those in the
# paper: "Solution Intervals for Variables in Spatial RCRCR linkages"
#by Enric Celaya.

# You can use this line to call the program from an interactive Maple session.

# Please provide the path that is appropriate for you system (windows paths
are separated by \)
# path:=`C:\???`: read cat(path, `R3C2_IO.map`); # your system
# path:=`/home/celaya/Dropbox/PatatoidesDual/`:  read cat(path,
`R3C2_IO.map`); # Linux Celaya
# path:=`/Users/celaya/Dropbox/PatatoidesDual/`: read cat(path,
`R3C2_IO.map`); # MAC Celaya

# Read the utilities for dual numbers, defined as dual(a,b). Variable path
must be set before.
read cat(path,`DualDefs.map`);

Digits:=8; #16;
with(plots);


# NOTE:
# Due to numerical approximations, sometimes one or more solutions are lost.
Often, lost solutions are found by simply re-issuing the call.

# It is frequent to get different results in successive identical calls.


###############################
# Computes the value of all variables for a given value of the input variable
vi. 've' is the variable to be eliminated from the system of equations.
# Writes the result in a CSV file solutions.txt
####################################
RCRCR_IO:=proc(alfas,as,ss,ve,vi, valorvi) # valorvi is the input vi value
# in radians
local vr, T, teta, equs, e4,eqf, eqt2, eqs2, eqt4, eqs4,s3,c3,x,y,j,sol,
solT,solr,sole, solt2, sols2, solt4, sols4,puntsir, punts, i,Alfa,Teta, fd,
config, confdeg;

    vr:=op({1,3,5} minus {ve,vi});# vr is the variable appearing in the
quartic e4(T)

    # Write the main equations
    equs:=eqT(alfas,as,ss,teta,T,x,y,ve,vi);
    e4:=equs[1];    # eq of 4 or 8 degree to solve to get the output
 variable
    s3:=equs[2];  c3:=equs[3]; # sin and cos of teta[3]

        # Definition of dual variables.
    for i from 1 to 5 do
        Alfa[i]:=dual(alfas[i]*Pi/180, as[i]):
        Teta[i]:=dual(teta[i], ss[i]):
    od;

    # Loop equation (used to successively get the values of remaining

variables)
     eqf:=evalm(&*(dtz(Alfa[5]),dtx(Teta[5]),dtz(Alfa[1]),dtx(Teta[1]),dtz

(Alfa[2]),dtx(Teta[2]),dtz(Alfa[3]),dtx(Teta[3]),dtz(Alfa[4]),dtx(Teta[4]))):

    puntsir:=NULL:
    punts:=NULL:

    # Write the caption in the CSV file
    #fd := fopen(cat(path, `solutions_b.txt`), WRITE);
    #fprintf(fd, "theta1,theta2,theta3,theta4,theta5,s2,s4 \n");
    #fclose(fd);

    sol:= fsolve(evalf(subs(teta[vi]=valorvi,e4)),T); # returns solution

values for vr
    for j from 1 to nops([sol]) do # if no solutions found, returns [].
        solT:= op(j,[sol]): # solution for T=tan(vr/2)
        solr:= evalf(arctan((2*solT),(1-solT**2)));
sole:= evalf(subs([teta[vr]=solr,teta[vi]=valorvi], eval(arctan(s3,c3))));

        eqt2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=valorvi], evalf(partreal(eqf[1,1])=1)));
        solt2:=map(Re,op(1,[solve(eqt2,teta[2])]));

        eqs2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=valorvi,teta[2]=solt2], evalf(partdual(eqf[1,1])=0)));
        sols2:=map(Re,op(1,[solve(eqs2,ss[2])]));

        eqt4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=valorvi,teta[2]=solt2], evalf(partreal(eqf[3,3])=1)));
        solt4:=map(Re,op(1,[solve(eqt4,teta[4])]));

        eqs4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=valorvi,teta[2]=solt2,teta[4]=solt4, ss[2]=sols2], evalf(partdual(eqf
[2,3])=0)));
        sols4:=map(Re,op(1,[solve(eqs4,ss[4])]));

        config:=[0,solt2, 0, solt4, 0, sols2, sols4];
        config[vi]:=valorvi;
        config[ve]:=sole;
        config[vr]:=solr;

        confdeg:=[raddeg(config[1]), raddeg(config[2]), raddeg(config
[3]), raddeg(config[4]), raddeg(config[5]), sols2, sols4];

        # Make a list with variable names to print on screen
        puntsir:=[theta(1)=confdeg[1], theta(2)=confdeg[2],theta
(3)=confdeg[3],theta(4)=confdeg[4], theta(5)=confdeg[5],
                    s(2)=confdeg[6],s(4)=confdeg[7]], " \n",
puntsir;

        # Check if solution is correct and append the configuration to
 the list of solution points for the given input.
        checksol(eqf,teta,ss,config); # Nothing special is done if the
 solution is incorrect. Only warning messages will be displayed.
        punts:=punts,config;

    # Write configuration in the CSV file
```

```
        #fd := fopen(cat(path, `solutions_b.txt`), APPEND);
        #fprintf(fd,cat("",confdeg[1],",",confdeg[2],",",confdeg
[3],",",confdeg[4],",",confdeg[5],",",confdeg[6],",",confdeg[7],"\n"));
        #fclose(fd);
    od;

    print(cat("solutions for theta",vi, "=", raddeg(valorvi)));
    print(puntsir);
    #return(punts): # uncomment this if required.
end:


###########################################
# Equation of degree 4 or 8 for variable T=tan(vr/2) with coeficients as
funtions of vi.
#############################################
eqT:=proc(alfas,as,ss,teta,T,x,y,ve,vi)
local vr,vx,fe, n,m, Alfa,Teta, Cf1,Cf2,dS1,dS2, Cf12,dS12, s3,c3,s3c3,eq,ex,
 x1,y1,x3,y3, i,j, A,B,C,E,F,G, e4;
    vr:=op({1,3,5} minus {ve,vi});# vr es la variable de l'eq quartica

    if ve=3 then
        vx:=vr; # vx is either 1 or 5
        fe:=2;  # sin(ve), cos(ve) will be eliminated with the
coefficients of the second ellipse.
    else
        vx:=ve;
        fe:=1;  # sin(ve) i cos(ve) will be eliminated with the
coefficients of the first ellipse.
    fi;

    # Preliminary step: elimination of ve form equations Cf1-Cf2=0 and
#dS1-dS2=0 to get an equation on vr, vi.
    Cf12:=A[1]+ B[1]*cos(teta[vx])+ C[1]*sin(teta[vx]) - (A[2]+ B[2]*cos(teta
[3]));
    dS12:=E[1]+ F[1]*cos(teta[vx])+ G[1]*sin(teta[vx]) - (E[2]+ F[2]*cos(teta
[3])+ G[2]*sin(teta[3]));
    C[2]:=0; # the coefficient C of the fixed ellipse is 0.

    s3:=solve(Cf12*F[fe]-dS12*B[fe], sin(teta[ve])); # eliminate cos(teta[ve])
and isolate sin(teta[ve])
    c3:=solve(Cf12*G[fe]-dS12*C[fe], cos(teta[ve])); # eliminate sin(teta[ve])
and isolate cos(teta[ve])
    s3c3:=(s3**2+c3**2-1)*(B[fe]*G[fe]-C[fe]*F[fe])**2; #(sin^2+cos^2=1)=>eq.
 in teta[vr] with coefs in teta[vi]

    # Definition of dual variables.
    for i from 1 to 5 do
        Alfa[i]:=dual(alfas[i]*Pi/180, as[i]):
        Teta[i]:=dual(teta[i], ss[i]):
    od;

    # Definition of the two subchains:
    n:=evalm(&*(dtz(Alfa[5]), dtx(Teta[5]), dtz(Alfa[1]), dtx(Teta[1]),
 dtz(Alfa[2]))):
    m:=evalm(&*(dtz(-Alfa[4]),dtx(-Teta[3]),dtz(-Alfa[3]))):

#print(n[1,1]);
#print(m[1,1]);
```

```

    Cf1:= evalf(partreal(n[1,1]));
    dS1:=-evalf(partdual(n[1,1]));

    Cf2:= evalf(partreal(m[1,1]));
    dS2:=-evalf(partdual(m[1,1]));

    ###### Instanciating coeficients of vx, dependent on vi
    x1:=collect(Cf1,[cos(teta[vx]),sin(teta[vx])]): # expression of the
form A + B cos1 + C sin1:
    A[1]:=evalf(coeff(coeff(x1,cos(teta[vx]),0),sin(teta[vx]),0)):
    B[1]:=evalf(coeff(x1,cos(teta[vx]),1)):
    C[1]:=evalf(coeff(x1,sin(teta[vx]),1)):

    y1:=collect(dS1,[cos(teta[vx]),sin(teta[vx])]): # expression of the
 form E + F cos1 + G sin1:
    E[1]:=evalf(coeff(coeff(y1,cos(teta[vx]),0),sin(teta[vx]),0)):
    F[1]:=evalf(coeff(y1,cos(teta[vx]),1)):
    G[1]:=evalf(coeff(y1,sin(teta[vx]),1)):

    ###### coefficients of teta[3]
    x3:=collect(Cf2,cos(teta[3])): # expression of the form A + B cos3:
    A[2]:=evalf(coeff(x3,cos(teta[3]),0)):
    B[2]:=evalf(coeff(x3,cos(teta[3]),1)):

    y3:=collect(dS2,[cos(teta[3]),sin(teta[3])]):# expression of the form
L + M cos3 + N sin3:
    E[2]:=evalf(coeff(coeff(y3,cos(teta[3]),0),sin(teta[3]),0));
    F[2]:=evalf(coeff(y3,cos(teta[3]),1)):
    G[2]:=evalf(coeff(y3,sin(teta[3]),1)):

    s3c3:=eval(s3c3); # Put the values A B C D E F G in s3c3.

    eq:=subs([cos(teta[vr])=(1-T**2)/(1+T**2),sin(teta[vr])=(2*T)/
(1+T**2)],s3c3); #Change of variable T=tg(vr/2)

    if vi=3 then ex:=4; else ex:=2; fi;
    e4:=normal(eval(eq*(1+T**2)**ex));  # equation of 4° or 8° degree in
T(vr) (coefficients are functions of vi)

    return([e4,s3,c3]);
end:

####################################
# Check if a configuration (list of 7 variable values) satisfies the loop
equation.
##################################
checksol:=proc(eqf,teta,ss,conf)
    local preal, pdual, ok, i,j, v;
    ok:=1;
    # Chek rotation part
    preal:=evalf(subs([teta[1]=conf[1],teta[2]=conf[2],teta[3]=conf[3],teta
[4]=conf[4],teta[5]=conf[5],ss[2]=conf[6],ss[4]=conf[7]], partreal(eqf)));

    for i from 1 to 3 do
    for j from 1 to 3 do
        if i=j then v:=1; else v:=0; fi;
        if abs(preal[i,j]-v) > 0.01 then
        print(cat("wrong value (rotation)", i,j,":", preal[i,j]));
```

```
          ok:= 0;
          fi;
      od;
      od;

      # check translation
      pdual:=evalf(subs([teta[1]=conf[1],teta[2]=conf[2],teta[3]=conf[3],teta
[4]=conf[4],teta[5]=conf[5],ss[2]=conf[6],ss[4]=conf[7]], partdual(eqf)));
      for i from 1 to 3 do
      for j from 1 to 3 do
          if abs(pdual[i,j]) > 0.05 then
          print(cat("wrong value (translation)", i,j,":", pdual[i,j]));
          ok:= 0;
          fi;
      od;
      od;

      return(ok);
end:


########## Parameters Duffy pag 232.
alfas:=[10,60,45,35,30];
as:= [32,25,30,40,10];
ss:=[30,s2,25,s4,0];


####################################################
# Calls for the extreme values of the input variables:

### input teta3:
RCRCR_IO(alfas,as,ss,5,3,11.7635*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,82.74850*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,142.32367*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,150.31600*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,230.73737*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,239.25965*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,244.75766*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,293.99364*evalf(Pi)/180);
#
RCRCR_IO(alfas,as,ss,5,3,92.79834*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,5,3,244.49660*evalf(Pi)/180);

### input teta1:
RCRCR_IO(alfas,as,ss,3,1,126.86436*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,1,168.41779*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,1,43.97516*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,1,268.49319*evalf(Pi)/180);

### input teta5:
RCRCR_IO(alfas,as,ss,3,5,50.47197*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,5,69.35086*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,5,148.78673*evalf(Pi)/180);
RCRCR_IO(alfas,as,ss,3,5,307.29955*evalf(Pi)/180);
```

Referring to the `RCRCR_IO procedure`, the following notation has been adopted:

- `alfas`, are the five linkage angles α_{ij} between pairing axes;
- `as`, are the five displacement a_{ij} along the x axes;

- ss, are the five displacement s_i along the z axes;
- ve, is the number of the *eliminating* variable;
- vi, is the number of the *input* variable;
- valorvi, is the value of *input* variable in radiant.

```
### Plot_embedded.txt #####

################################ Duoble plot output variable file with an
embedded procecess

####################################

RCRRC:=proc(alfas,as,ss,ve,vi,step)

local vr, T, teta, equs, e4, s3,c3,Cf1,Cf2,dS1,dS2, r1,r3,x,y, i,j,
 lcolor,sol, solT,solr,sole, puntsir,puntsie,lints, puntsi,Alfa,Teta,eqf,
puntsiT2, puntsiT4,puntsiS2,puntsiS4,
eqt2,solt2, eqs2,sols2, eqt4,solt4, eqs4,sols4;



vr:=op({1,3,5} minus{ve,vi}); # vr es la variable de l'eq de grau 8 e4(T).



    equs:=eqT(alfas,as,ss,teta,T,x,y,ve,vi); # vi es 3.



    e4:= equs[1]; # eq de vi=3 i vr=1|5 (<> ve)

#    print('e4'= e4);

    s3:= equs[2]; c3:= equs[3];

    #Cf1:=equs[4]; dS1:=equs[5];

    #Cf2:=equs[6]; dS2:=equs[7];



    lints:=[]: puntsi:=[]:

########################## Part to add the process to evaluate theta2
theta4 s2 e s4  #################################################

# Definition of dual variables.
    for i from 1 to 5 do
        Alfa[i]:=dual(alfas[i]*Pi/180, as[i]):
        Teta[i]:=dual(teta[i], ss[i]):
    od;

    # Loop equation (used to successively get the values of remaining
variables)
        eqf:=evalm(&*(dtz(Alfa[5]),dtx(Teta[5]),dtz(Alfa[1]),dtx(Teta
[1]),dtz(Alfa[2]),dtx(Teta[2]),dtz(Alfa[3]),dtx(Teta[3]),dtz(Alfa[4]),dtx
(Teta[4]))):






##########Inizialization lists of points to plot#####
    puntsir:=NULL: puntsie:=NULL: lcolor:=NULL: puntsiT2:=NULL:
 puntsiT4:=NULL: puntsiS2:=NULL: puntsiS4:=NULL:

    ### Grafics de les solucions del mecanisme per diferents valors de vi.

    for i from 0 by step to evalf(2*Pi) do

if i=100 then print(ok100) fi;
if i=200 then print(ok200) fi;

        sol:= fsolve(evalf(subs(teta[vi]=i,e4)),T); #torna valors per vr



    for j from 1 to nops([sol]) do

 # si no troba solucions torna una llista buida.

            solT:= op(j,[sol]): # solution per T=tan(vr/2)

            solr:= evalf(arctan((2*solT),(1-solT**2)));

            sole:= evalf(subs([teta[vr]=solr,teta[vi]=i], eval(arctan(s3,c3))));

            puntsir:=[raddeg(i),raddeg(solr)],puntsir;

            puntsie:=[raddeg(i),raddeg(sole)],puntsie;

            lcolor:= black,lcolor;



                eqt2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i],
 evalf(partreal(eqf[1,1])=1)));
        solt2:=map(Re,op(1,[solve(eqt2,teta[2])]));

        eqs2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=i,teta[2]=solt2], evalf(partdual(eqf[1,1])=0)));
        sols2:=map(Re,op(1,[solve(eqs2,ss[2])]));

        eqt4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=i,teta[2]=solt2], evalf(partreal(eqf[3,3])=1)));
        solt4:=map(Re,op(1,[solve(eqt4,teta[4])]));

        eqs4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta
[vi]=i,teta[2]=solt2,teta[4]=solt4, ss[2]=sols2], evalf(partdual(eqf
[2,3])=0)));
        sols4:=map(Re,op(1,[solve(eqs4,ss[4])]));


```

```
                puntsiT2:=[raddeg(i),raddeg(solt2)],puntsiT2;
        puntsiS2:=[raddeg(i),sols2],puntsiS2;
        puntsiT4:=[raddeg(i),raddeg(solt4)],puntsiT4;
        puntsiS4:=[raddeg(i),sols4],puntsiS4;



        od;

    od;

        print(plotRCRCR(vi,vr,ve, [puntsir], [puntsie], [lcolor], lints,
puntsi, [puntsiT2], [puntsiT4], [puntsiS2], [puntsiS4])); # dibuix dels
resultats.

    #plotRCRCR(3,vr,ve, [puntsir], [puntsie] , [lcolor], lints, puntsi);

return(lints);

end:



####################################

########## Grafics de les relacions entre angles (1,3,5) del RCRCR.

####################################

plotRCRCR:=proc(vi,vr,ve, puntsi1, puntsi2, col, lint, puntsi,
puntsi3, puntsi4, puntsi5,
 puntsi6)

local tit, rg, rgs, plotint, plot1, plot2, plot3, plot4, plot5, plot6,
plotl, i, ylow, xp;

  ylow:=0;

  plotint:=NULL;

for i from 1 to nops(lint) do

      plotint:=plotint,plot(ylow,x=lint[i][1]..lint[i][2], thickness=3,
color=black);

  od;



  plotl:=NULL;

for i from 1 to nops(puntsi) do

      xp:=raddeg(puntsi[i]);

      plotl:=plotl,plot([[xp,0],[xp,360]], color=gray, thickness=0);

  od;



  rg:=[0..360,ylow..360];

  rgs:=[0..360,-360..400];





  if nops(puntsi1)>0 then

    tit:=cat(`theta`, vr,`(theta`, vi,`) in degrees`):

    plot1:=pointplot(puntsi1,  color=col, symbol=point);

     #print(display(pointplot([puntsi1])));

    # print(display({plotint,plot1,plotl}, view=rg, title=tit,
scaling=constrained, labels=[vi,vr]));

    print(display({plot1},view=rg,scaling=constrained,labels=[cat
(`theta`,vi),cat(`theta`,vr)],axes=boxed));

    #display(plot1,view=rg,scaling=constrained,labels=[cat
(`theta`,vi),cat(`theta`,vr)],axes=boxed);



  fi;


if nops(puntsi2)>0 then

    tit:=cat(`theta`, ve,`(theta`, vi,`) in degrees`):

    plot2:=pointplot(puntsi2,  color=col, symbol=point);

## print(display({plotl, plotint, plot2}, view=rg, title=tit,
scaling=constrained, labels=[vi,ve]));

print(display({plotl,plotint,plot2},view=rg,scaling=constrained,labels=
[cat(`theta`,vi),cat(`theta`,ve)],axes=boxed)):

  fi;


if nops(puntsi3)>0 then

    tit:=cat(`theta`, 2,`(theta`, vi,`) in degrees`):

    plot3:=pointplot(puntsi3,  color=col, symbol=point);


print(display({plotl,plotint,plot3},view=rg,scaling=constrained,labels=
[cat(`theta`,vi),cat(`theta`,2)],axes=boxed)):

  fi;

```

```

if nops(puntsi4)>0 then

    tit:=cat(`theta`, 4,`(theta`, vi,`) in degrees`):

    plot4:=pointplot(puntsi4,  color=col, symbol=point);


print(display({plotl,plotint,plot4},view=rg,scaling=constrained,labels=
[cat(`theta`,vi),cat(`theta`,4)],axes=boxed)):

  fi;




if nops(puntsi5)>0 then

    tit:=cat(`s`, 2,`(theta`, vi,`) in degrees`):

    plot5:=pointplot(puntsi5,  color=col, symbol=point);



print(display({plot5},view=rgs,scaling=constrained,labels=[cat
(`theta`,vi),cat(`s`,2)],axes=boxed)):

  fi;


if nops(puntsi6)>0 then

    tit:=cat(`s`, 4,`(theta`, vi,`) in degrees`):

    plot6:=pointplot(puntsi6,  color=col, symbol=point);


print(display({plot6},view=rgs,scaling=constrained,labels=[cat
(`theta`,vi),cat(`s`,4)],axes=boxed)):

  fi;

end:

RCRRC(alfas,as,ss,5,3,0.001);
```

Appendix C

Using the function shown in the appendix B, have been obtained the plots for each d input variables.

θ_1 as input variable

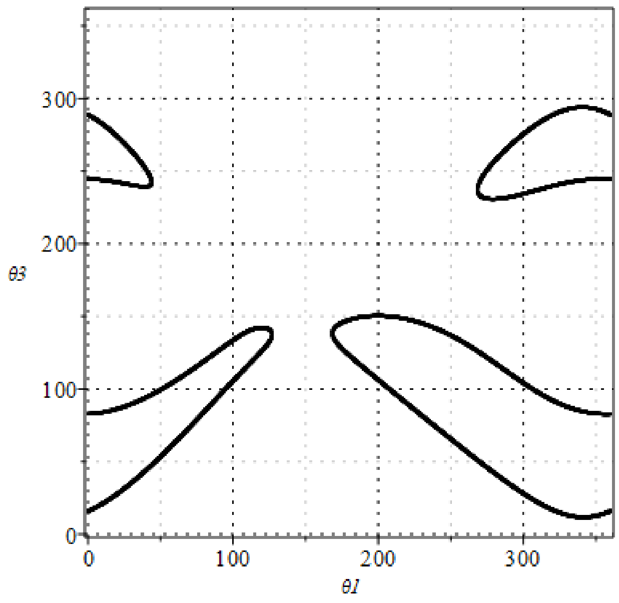


Figure 5.1: *Plot, $\theta_1 - \theta_3$.*

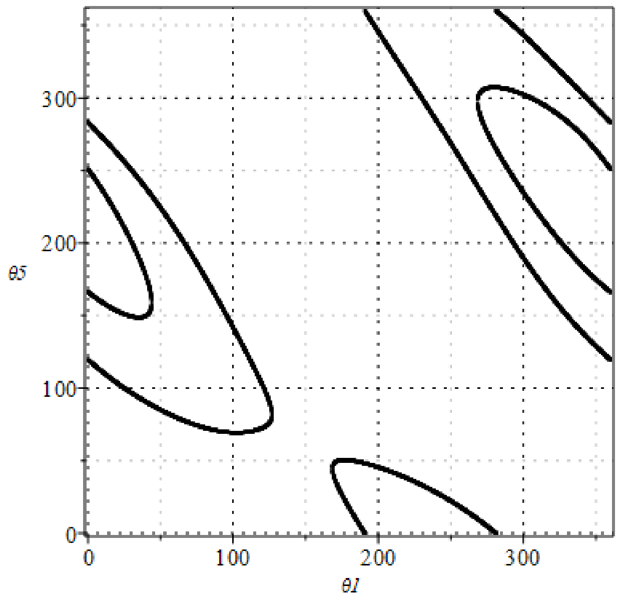


Figure 5.2: *Plot, $\theta_1 - \theta_5$.*

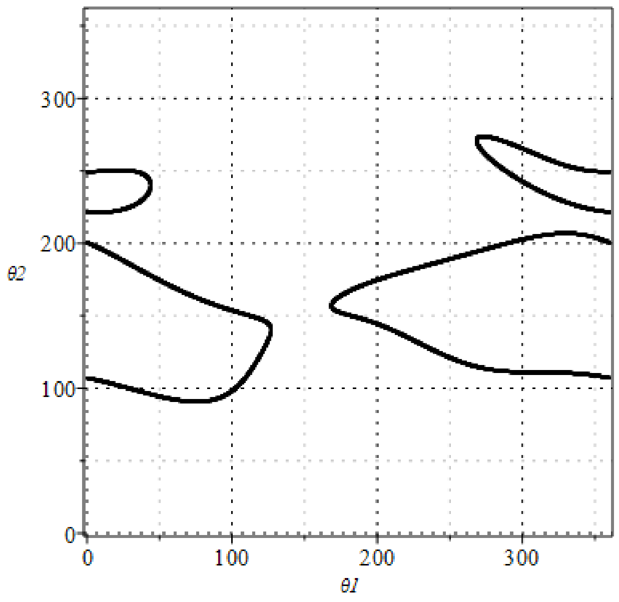


Figure 5.3: Plot, $\theta_1 - \theta_2$.

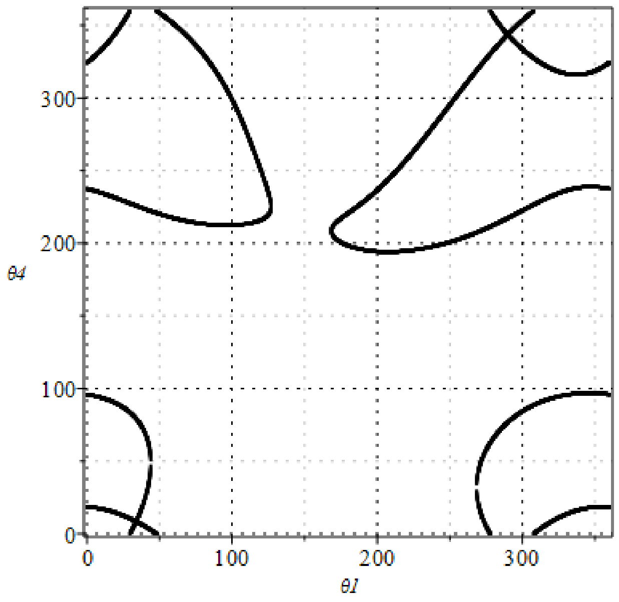


Figure 5.4: Plot, $\theta_1 - \theta_4$.

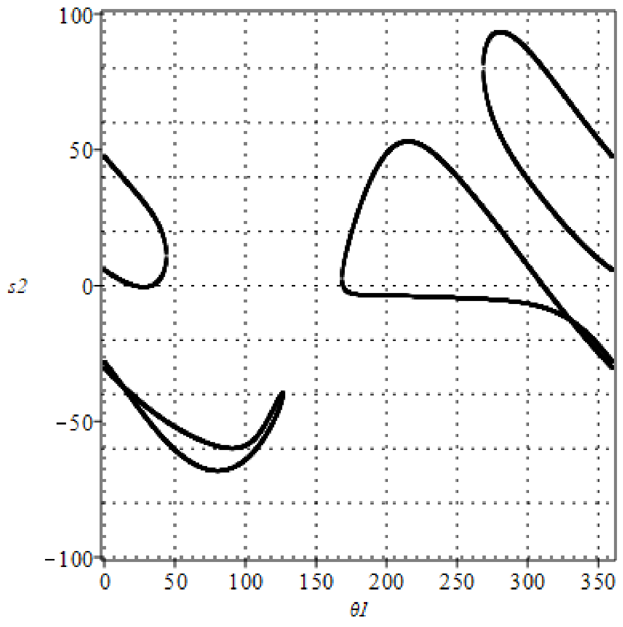


Figure 5.5: Plot, $\theta_1 - s_2$.

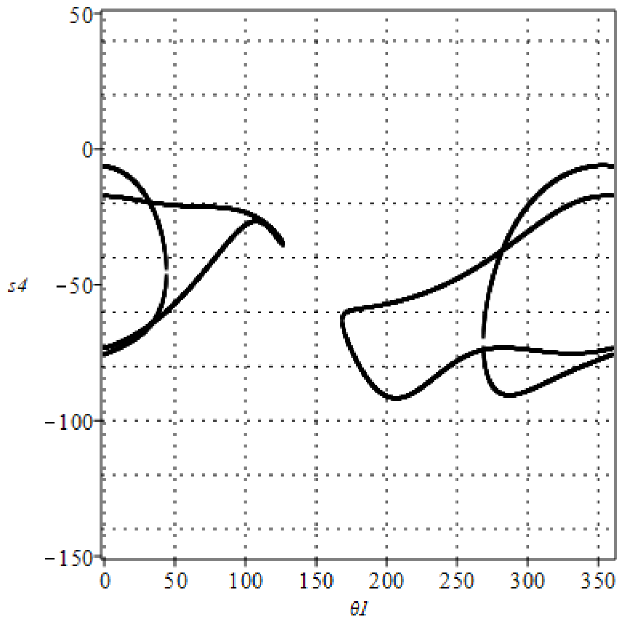


Figure 5.6: Plot, $\theta_1 - s_4$.

θ_3 as input variable

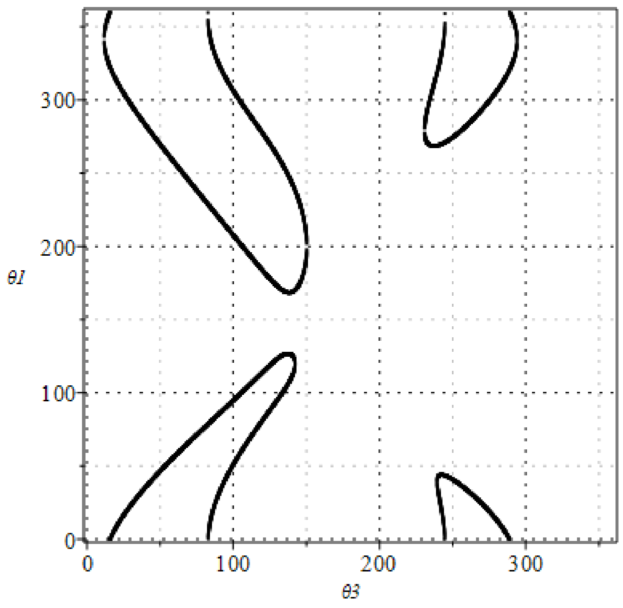


Figure 5.7: *Plot,* $\theta_3 - \theta_1$.

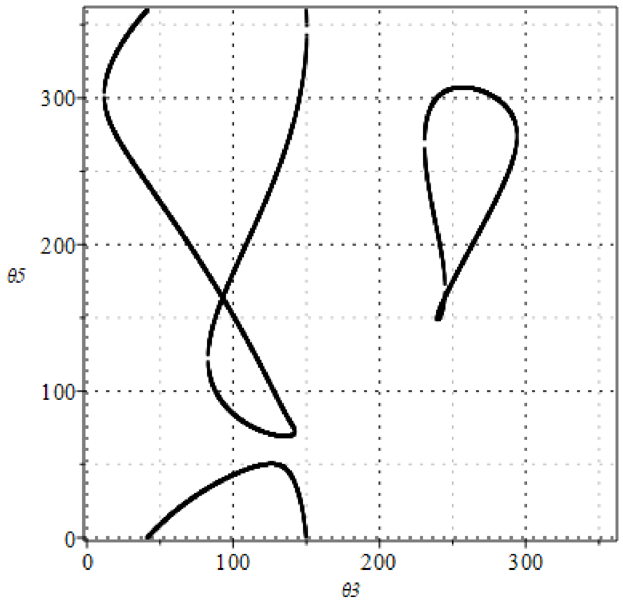


Figure 5.8: *Plot,* $\theta_3 - \theta_5$.

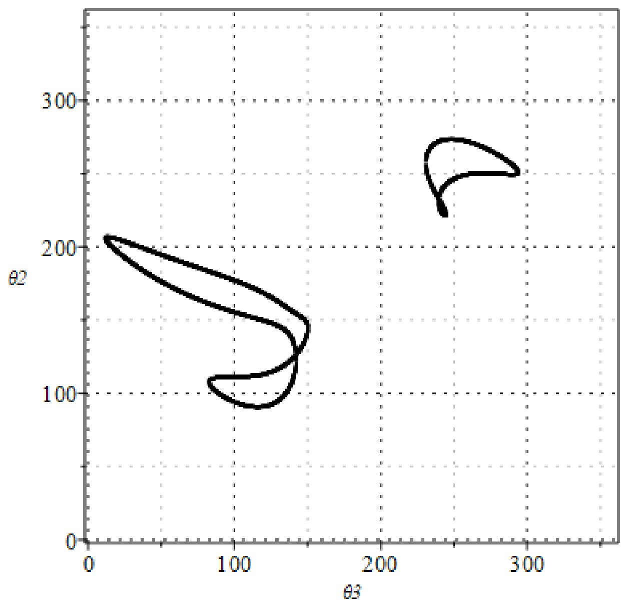


Figure 5.9: Plot, $\theta_3 - \theta_2$.

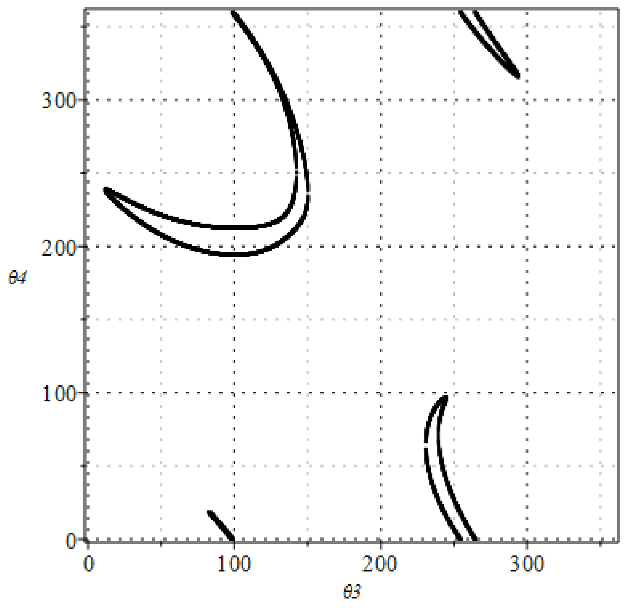


Figure 5.10: Plot, $\theta_3 - \theta_4$.

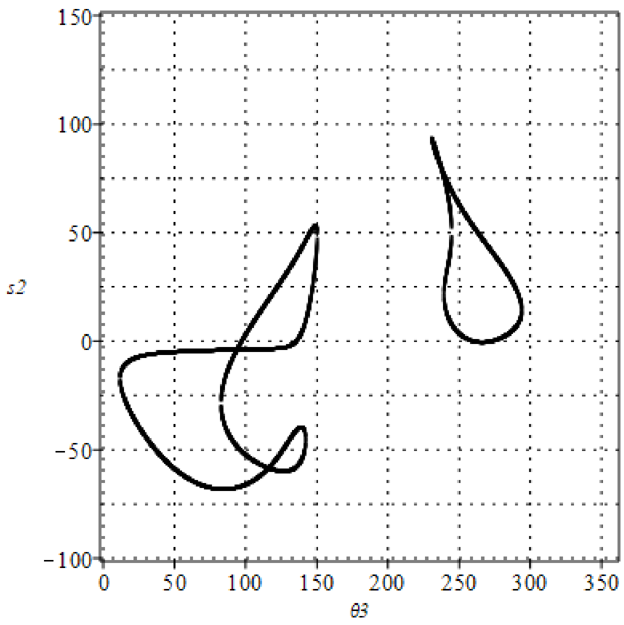


Figure 5.11: Plot, $\theta_3 - s_2$.

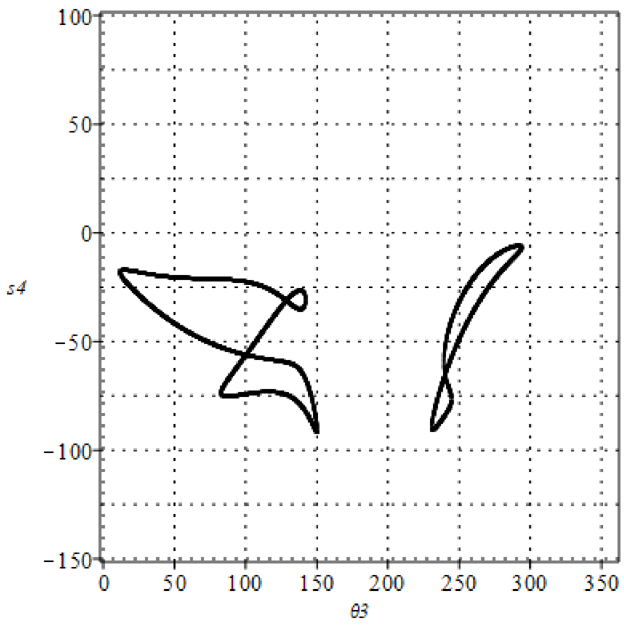


Figure 5.12: Plot, $\theta_3 - s_4$.

θ_5 as input variable

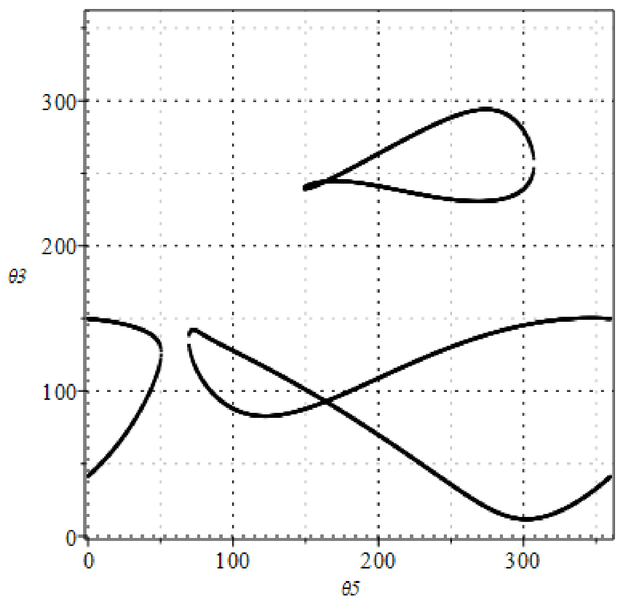


Figure 5.13: *Plot, $\theta_5 - \theta_3$.*

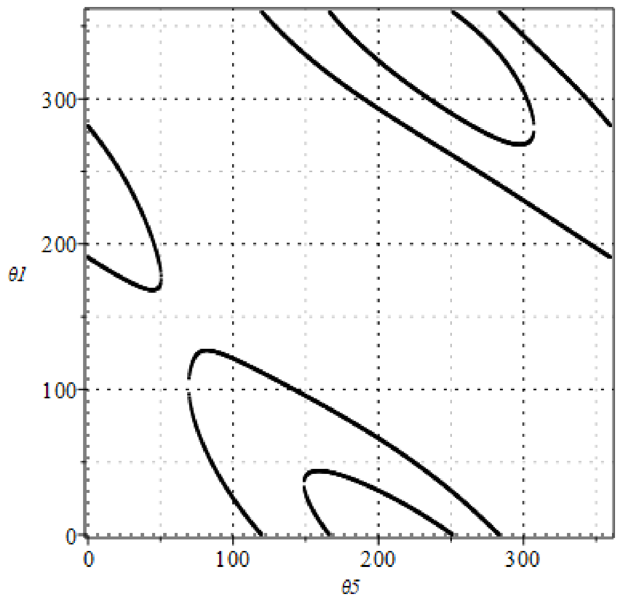


Figure 5.14: *Plot, $\theta_5 - \theta_1$.*

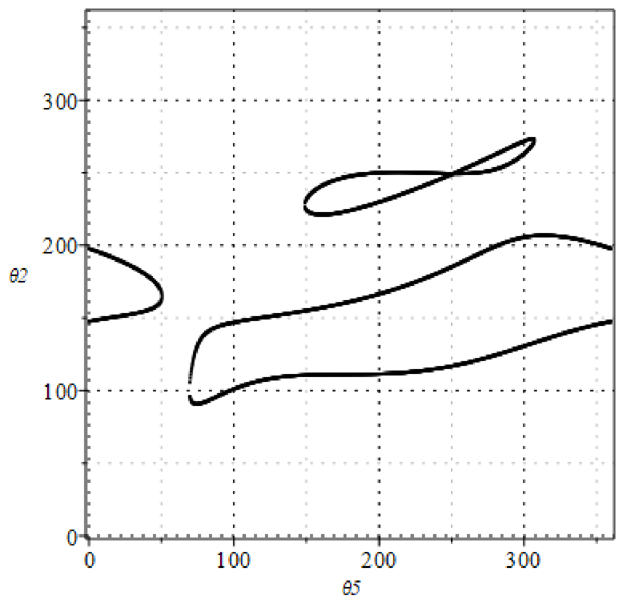


Figure 5.15: Plot, $\theta_5 - \theta_2$.

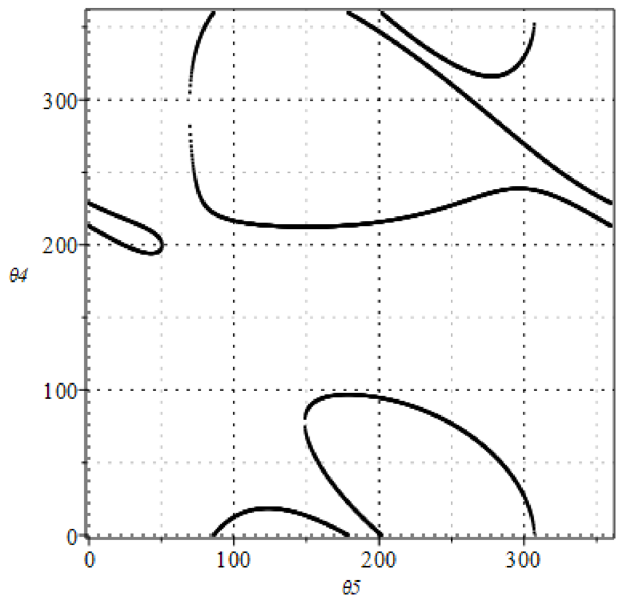


Figure 5.16: Plot, $\theta_5 - \theta_4$.

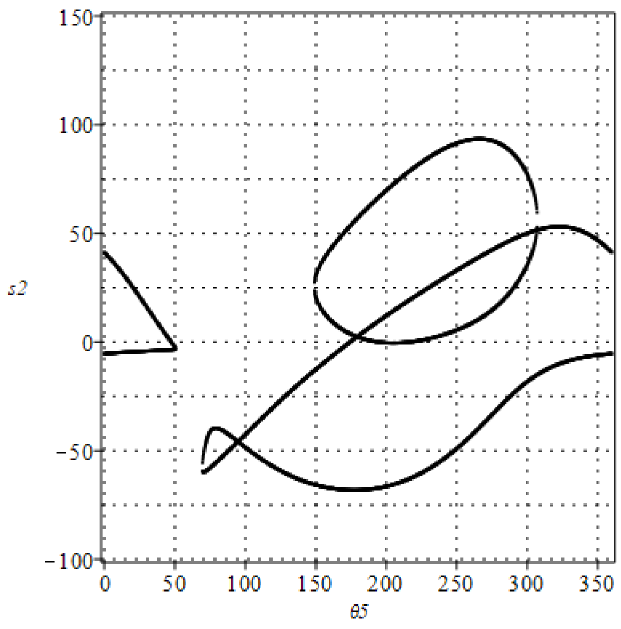


Figure 5.17: *Plot*, $\theta_5 - s_2$.

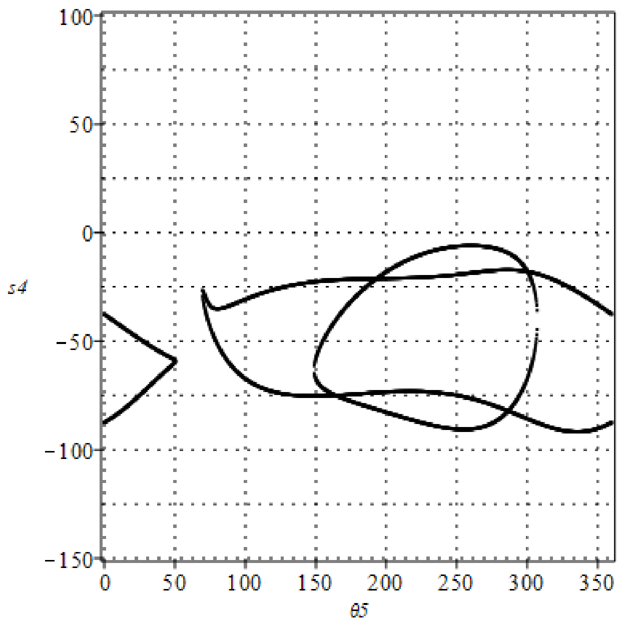


Figure 5.18: *Plot*, $\theta_5 - s_4$.

Appendix D

```
####### Tarea.txt
#########Create a procedure to calculate the area of a trangle
###############gives three point
AreaT:=proc(A,B,C)
local AB,AC,mAB,mAC,cphi,sphi,phi,Area;


AB:=[B[1]-A[1],B[2]-A[2],B[3]-A[3]];  mAB:=sqrt(AB[1]^2+AB[2]^2+AB[3]^2);
AC:=[C[1]-A[1],C[2]-A[2],C[3]-A[3]];  mAC:=sqrt(AC[1]^2+AC[2]^2+AC[3]^2);

cphi:=evalf((evalm(&*(AB,AC)))/(mAB*mAC));                      #cosin of phi
phi:=acos(cphi);
sphi:= sqrt(1-cphi^2);                                    #sin of phi
Area:=(mAB*mAC*sphi)/2;                                   #Area of the triangle



return(Area);

end:
```

```
####### Angle_min.txt
#########Create a procedure to calculate the angle between two vectors  #####


AngleMin:=proc(A,B,C,E)
######### Given 4 points that constitute the two vector AB e CD  #######
######### In case of consecutive links A and C are the same point  #####

local phi,Cphi,AB,CE,mAB,mCE;


AB:=[B[1]-A[1],B[2]-A[2],B[3]-A[3]]; mAB:=sqrt(AB[1]^2+AB[2]^2+AB[3]^2);
CE:=[E[1]-C[1],E[2]-C[2],E[3]-C[3]]; mCE:=sqrt(CE[1]^2+CE[2]^2+CE[3]^2);
#print(AB,CE);
#print(cat("The cross product has a value of:",evalm(&*(AB,CE))));

Cphi:=evalf((evalm(&*(AB,CE)))/(mAB*mCE));
phi:=arccos(Cphi);
#print(raddeg(phi));


return(raddeg(phi)): #The angle is return in degree
end:
```

```
############################################################################

EXTRACT:=proc(alfas,as,ss,ve,vi,step)

global puntT5,confdeg;


```

```
local vr, T, teta, equs, e4, s3,c3,Cf1,Cf2,dS1,dS2, r1,r3,x,y, i,j, lcolor,sol,
 solT,solr,sole, puntsir,puntsie,lints, puntsi,Alfa,Teta,eqf,puntsiT2,
 puntsiT4,puntsiS2,puntsiS4,eqt2,solt2, eqs2,sols2, eqt4,solt4, eqs4,sols4,
 config,k,confdeg1,conf,conf1;

k:=1:

vr:=op({1,3,5} minus{ve,vi}); # vr es la variable de l'eq de grau 8 e4(T).



    equs:=eqT(alfas,as,ss,teta,T,x,y,ve,vi); # vi es 3.



    e4:= equs[1]; # eq de vi=3 i vr=1|5 (<> ve)


    s3:= equs[2]; c3:= equs[3];





    lints:=[]: puntsi:=[]:

########################### Part to add the process to evaluate theta2
########################### theta4 s2 e s4.
# Definition of dual variables.
    for i from 1 to 5 do
        Alfa[i]:=dual(alfas[i]*Pi/180, as[i]):
        Teta[i]:=dual(teta[i], ss[i]):
    od;

    # Loop equation (used to successively get the values of ...
    remaining variables)
        eqf:=evalm(&*(dtz(Alfa[5]),dtx(Teta[5]),dtz(Alfa[1]),dtx(Teta[1])...
        ,dtz(Alfa[2]),dtx(Teta[2]),dtz(Alfa[3]),dtx(Teta[3]),dtz(Alfa[4])...
        ,dtx(Teta[4]))):


##########Inizialization lists of points to plot############################
    puntsir:=NULL: puntsie:=NULL: lcolor:=NULL: puntsiT2:=NULL:
 puntsiT4:=NULL: puntsiS2:=NULL: puntsiS4:=NULL: puntT5:=NULL:

confdeg:=Matrix(300,7):
conf:=Matrix(300,7):
############################### First Branch #################################

    for i from 2.6 by step to 5.4 do


        sol:= fsolve(evalf(subs(teta[vi]=i,e4)),T); #torna valors per vr



    for j from 1 to nops([sol]) do

           solT:= op(j,[sol]): # solucio per T=tan(vr/2)
```

```

            solr:= evalf(arctan((2*solT),(1-solT**2)));

            sole:= evalf(subs([teta[vr]=solr,teta[vi]=i], eval(arctan(s3,c3))));

            puntsir:=[raddeg(i),raddeg(solr)],puntsir;

            puntsie:=[raddeg(i),raddeg(sole)],puntsie;
if j=1 then
            lcolor:= black,lcolor
elif j=2 then
            lcolor:= red,lcolor
elif j=3 then
            lcolor:= green,lcolor
else
    lcolor:= blue,lcolor
fi;



        eqt2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i],...
        evalf(partreal(eqf[1,1])=1)));
        solt2:=map(Re,op(1,[solve(eqt2,teta[2])]));

        eqs2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,...
        teta[2]=solt2], evalf(partdual(eqf[1,1])=0)));
        sols2:=map(Re,op(1,[solve(eqs2,ss[2])]));

        eqt4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,...
        teta[2]=solt2], evalf(partreal(eqf[3,3])=1)));
        solt4:=map(Re,op(1,[solve(eqt4,teta[4])]));

        eqs4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,...
        teta[2]=solt2,teta[4]=solt4, ss[2]=sols2],..
        evalf(partdual(eqf[2,3])=0)));
        sols4:=map(Re,op(1,[solve(eqs4,ss[4])]));


                puntsiT2:=[raddeg(i),raddeg(solt2)],puntsiT2;
                puntsiS2:=[raddeg(i),sols2],puntsiS2;
                puntsiT4:=[raddeg(i),raddeg(solt4)],puntsiT4;
                puntsiS4:=[raddeg(i),sols4],puntsiS4;


if j=3 then
        config:=[0,solt2, 0, solt4, 0, sols2, sols4];
        config[vi]:=i;
        config[ve]:=sole;
        config[vr]:=solr;
        confdeg[k,1..7]:=<raddeg(config[1]), raddeg(config[2]), ...
        raddeg(config[3]), raddeg(config[4]), raddeg(config[5]),...
        sols2, sols4>;
        conf[k,1..7]:=<config[1], config[2], config[3],...
        config[4], config[5], sols2, sols4>;
        k:=k+1;
fi


        od;


    od;

################################ Second Branch   ######################
 for i from 5.3 by -step to 2.6 do


             sol:= fsolve(evalf(subs(teta[vi]=i,e4)),T); #torna valors per vr



    for j from 1 to nops([sol]) do


         # si no troba solucions torna una llista buida.

                solT:= op(j,[sol]): # solucio per T=tan(vr/2)

                solr:= evalf(arctan((2*solT),(1-solT**2)));

          sole:= evalf(subs([teta[vr]=solr,teta[vi]=i], eval(arctan(s3,c3))));

                puntsir:=[raddeg(i),raddeg(solr)],puntsir;

                puntsie:=[raddeg(i),raddeg(sole)],puntsie;

            if j=1 then
                        lcolor:= black,lcolor
            elif j=2 then
                        lcolor:= red,lcolor
            elif j=3 then
                        lcolor:= green,lcolor
            else
                lcolor:= blue,lcolor
            fi;



                        eqt2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,...
                        teta[vi]=i], evalf(partreal(eqf[1,1])=1)));
                solt2:=map(Re,op(1,[solve(eqt2,teta[2])]));

                eqs2:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,...
                teta[2]=solt2], evalf(partdual(eqf[1,1])=0)));
                sols2:=map(Re,op(1,[solve(eqs2,ss[2])]));

                eqt4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,...
                teta[2]=solt2], evalf(partreal(eqf[3,3])=1)));
                solt4:=map(Re,op(1,[solve(eqt4,teta[4])]));

                eqs4:=evalf(subs([teta[vr]=solr,teta[ve]=sole,teta[vi]=i,....
                teta[2]=solt2,teta[4]=solt4, ss[2]=sols2], ...
                evalf(partdual(eqf[2,3])=0)));
                sols4:=map(Re,op(1,[solve(eqs4,ss[4])]));


                        puntsiT2:=[raddeg(i),raddeg(solt2)],puntsiT2;
                puntsiS2:=[raddeg(i),sols2],puntsiS2;
                puntsiT4:=[raddeg(i),raddeg(solt4)],puntsiT4;
                puntsiS4:=[raddeg(i),sols4],puntsiS4;


            if j=2 then
                    config:=[0,solt2, 0, solt4, 0, sols2, sols4];
                    config[vi]:=i;
                    config[ve]:=sole;
                    config[vr]:=solr;
                    confdeg[k,1..7]:=<raddeg(config[1]), raddeg(config[2]),...
                    raddeg(config[3]), raddeg(config[4]), raddeg(config[5])...
                    , sols2, sols4>; #Degrees
                    conf[k,1..7]:=<config[1], config[2], config[3], ...
                    config[4], config[5], sols2, sols4>;    #Radiants
                    k:=k+1;
                fi


    od;


od;

#######################################################################

conf1:= Matrix(k-1,7):
conf1[1..k-1,1..7]:= conf[1..k-1,1..7]:

###################################
confdeg1:= Matrix(k-1,7):
confdeg1[1..k-1,1..7]:= confdeg[1..k-1,1..7]:

return(conf1,k,confdeg1):

end:


#EXTRACT(alfas,as,ss,3,5,0.1);
```

```

###### Shape_study.txt
###### Simulation Links with maple tools ###################

############# Read useful procdures #########################################
read cat(path, `Tarea.txt`);
read cat(path, `Detect_collision.txt`);
read cat(path, `Check_Instant_v02.txt`);
read cat(path, `Angle_min.txt`);
read cat(path, `Distance.txt`);
read cat(path, `Extract_point.txt`);


############# Adding Library ###################################
with(plots):    with(Typesetting):  with(plottools): with(LinearAlgebra):


############################ Start Script ########################
config:=EXTRACT(alfas,as,ss,3,5,0.1);
```

```


# Conversion of the alphas in radiants
alfasd:=[degrad(alfas[1]),degrad(alfas[2]),degrad(alfas[3]),degrad(alfas[4])...
,degrad(alfas[5])]:

Mconfig:=config[1]:          #Matrix with angles in radiant
k:=config[2]:                #Dimension of the matrix
Mconfigdeg:=config[3]:       #Matrix with angles in degree

punto1:=NULL: punto1i:=NULL:    punto1f:=NULL: punto1a:=NULL: punto1b:=NULL:
punto2:=NULL: punto2V:=NULL: punto2F:=NULL: punto2I:=NULL: punto2N:=NULL:
punto3:=NULL:
punto4:=NULL: punto4V:=NULL: punto4F:=NULL:
punto5:=NULL: punto1A:=NULL: punto5A:=NULL:
punto2a:=NULL: punto2b:=NULL:
punto2Va:=NULL: punto2Vb:=NULL:
punto3a:=NULL: punto3b:=NULL:
punto4a:=NULL: punto4b:=NULL:
punto4Va:=NULL: punto4Vb:=NULL:
punto5a:=NULL: punto5b:=NULL:



psdM51i44F:=0:
psdM1i2N2V3:=0:
psdM2I2F44F:=0:
psdM1i2N34:=0:
psdM2I2F34:=0:
psdM1i2N44F:=0:
psdM1i2N4V5:=0:
psdM2V344F:=0:
psdM2I2F4V5:=0:
psdM2I2F51i:=0:
psdM2V34V5:=0:
psdM2V351i:=0:
psdM344V5:=0:
psdM3451i:=0:

# mf is the multiplication factor

psdMJ1J2V:=0: psdMJ1J3:=0:   psdMJ1J4V:=0: psdMJ1J5:=0:
psdMJ2VJ3:=0: psdMJ2VJ4V:=0: psdMJ2VJ5:=0:
psdMJ3J5:=1:
psdMJ4J5:=0:


MaxS2:=max(Mconfig[..,6]):
MinS2:=min(Mconfig[..,6]):

MaxS4:=min(Mconfig[..,7]):

er:=5: #Is the error in degrees between two consecutive couple
hl:=5*mf:        :  #joint lenght
think:=20: #thickness of the lines in the plot
radiu:=22.5: #radius of the spheres that simulates the joint envelope
epsi:=0.5: # Error for the link
epsj:=20    : #Error for the joint

```

```
LINK:=NULL:
NLINK:=NULL:

##########################################################################

origin:=[[1,1,1],[-1,1,1],[-1,-1,1],[1,-1,1],[1,-1,-1],[-1,-1,-1],[-1,1,-1]...
,[1,1,-1],[1,1,1],[1,-1,1]]:
ori:=spacecurve(origin,title=" center",color=blue,labels=[x, y, z],...
titlefont= ["ROMAN", 15],scaling=constrained):


#############################################################################


for i from 1 by 1 to k-1 do

    O1P:=[0,0,0];

    O1i:=evalf(evalm(&*(Tx(80*mf)))):
    O1iP:=[O1i[1,4],O1i[2,4],O1i[3,4]]:
    puntO1i:=puntO1i,O1iP:

    O1f:=evalf(evalm(&*(Tx(-60*mf)))):
    O1fP:=[O1f[1,4],O1f[2,4],O1f[3,4]]:
    puntO1f:=puntO1f,O1fP:


    ##########################################################################

    O2:=evalf(evalm(&*(Rx(Mconfig[i,1]),Tx(ss[1]),Tz(as[2]),Rz(alfasd[2])))):
    O2P:=[O2[1,4],O2[2,4],O2[3,4]]:
    puntO2:=puntO2,O2P:

    #        #        #        #        #        #        #        #        #

    O2V:=evalf(evalm(&*(O2,Rx(Mconfig[i,2]),Tx(Mconfig[i,6])))):
    O2PV:=[O2V[1,4],O2V[2,4],O2V[3,4]]:
    puntO2V:=puntO2V,O2PV:



    #                 #                 #                 #                 #

    O2F:=evalf(evalm(&*(O2,Tx(MaxS2)))):
    O2PF:=[O2F[1,4],O2F[2,4],O2F[3,4]]:
    puntO2F:=puntO2F,O2PF:

    #                 #                 #                 #                 #
    (* ##Is the old O2I
    O2I:=evalf(evalm(&*(O2,Tx(MinS2)))):
    O2PI:=[O2I[1,4],O2I[2,4],O2I[3,4]]:
    puntO2I:=puntO2I,O2PI:
    *)

    #                 #                 #            New Part that we link at O1
    O2N:=evalf(evalm(&*(O2,Tx(evalf(MinS2-5*mf))))):
    O2PN:=[O2N[1,4],O2N[2,4],O2N[3,4]]:
    puntO2N:=puntO2N,O2PN:

        #               #               #               #               #
    O2I:= O2N:
    O2PI:= O2PN:
    puntO2I:= puntO2N:


    ##########################################################################

    O3:=evalf(evalm(&*(O2,Rx(Mconfig[i,2]),Tx(Mconfig[i,6]),Tz(as[3]),...
    Rz(alfasd[3])))):
    O3P:=[O3[1,4],O3[2,4],O3[3,4]]:
    puntO3:=puntO3,O3P:

#####

    O4:=evalf(evalm(&*(O3,Rx(Mconfig[i,3]),Tx(ss[3]),Tz(as[4]),...
    Rz(alfasd[4])))):
    O4P:=[O4[1,4],O4[2,4],O4[3,4]]:
    puntO4:=puntO4,O4P:

        #       #       #       #       #       #       #       #

    O4V:=evalf(evalm(&*(O4,Rx(Mconfig[i,4]),Tx(Mconfig[i,7])))):
    O4PV:=[O4V[1,4],O4V[2,4],O4V[3,4]]:
    puntO4V:=puntO4V,O4PV:


    #               #               #               #

    O4F:=evalf(evalm(&*(O4,Tx(MaxS4)))):
    O4PF:=[O4F[1,4],O4F[2,4],O4F[3,4]]:
    puntO4F:=puntO4F,O4PF:


    ##################################################################

    O5:=evalf(evalm(&*(O4,Rx(Mconfig[i,4]),Tx(Mconfig[i,7]),Tz(as[5]),...
    Rz(alfasd[5])))):
    O5P:=[O5[1,4],O5[2,4],O5[3,4]]:
    puntO5:=puntO5,O5P:




    ######################## Build the joints with the lines ##########
    O1a:=evalf(evalm(&*(O1i,Tx(hl)))):
    O1aP:=[O1a[1,4],O1a[2,4],O1a[3,4]]:
    puntO1a:=puntO1a,O1aP:

    O1b:=evalf(evalm(&*(O1i,Tx(-hl)))):
    O1bP:=[O1b[1,4],O1b[2,4],O1b[3,4]]:
    puntO1b:=puntO1b,O1bP:

    #               #               #               #               #
    O2a:=evalf(evalm(&*(O2,Tx(hl)))):
    O2aP:=[O2a[1,4],O2a[2,4],O2a[3,4]]:
    puntO2a:=puntO2a,O2aP:

    O2b:=evalf(evalm(&*(O2,Tx(-hl)))):
```

```
    O2bP:=[O2b[1,4],O2b[2,4],O2b[3,4]]:
    puntO2b:=puntO2b,O2bP:


    #               #               #               #               #
    O2Va:=evalf(evalm(&*(O2V,Tx(hl)))):
    O2VaP:=[O2Va[1,4],O2Va[2,4],O2Va[3,4]]:
    puntO2Va:=puntO2Va,O2VaP:

    O2Vb:=evalf(evalm(&*(O2V,Tx(-hl)))):
    O2VbP:=[O2Vb[1,4],O2Vb[2,4],O2Vb[3,4]]:
    puntO2Vb:=puntO2Vb,O2VbP:


        #               #               #               #
    O3a:=evalf(evalm(&*(O3,Tx(hl)))):
    O3aP:=[O3a[1,4],O3a[2,4],O3a[3,4]]:
    puntO3a:=puntO3a,O3aP:

    O3b:=evalf(evalm(&*(O3,Tx(-hl)))):
    O3bP:=[O3b[1,4],O3b[2,4],O3b[3,4]]:
    puntO3b:=puntO3b,O3bP:


    #               #               #               #               #
    O4a:=evalf(evalm(&*(O4,Tx(hl)))):
    O4aP:=[O4a[1,4],O4a[2,4],O4a[3,4]]:
    puntO4a:=puntO4a,O4aP:

    O4b:=evalf(evalm(&*(O4,Tx(-hl)))):
    O4bP:=[O4b[1,4],O4b[2,4],O4b[3,4]]:
    puntO4b:=puntO4b,O4bP:


    #               #               #               #               #
    O4Va:=evalf(evalm(&*(O4V,Tx(hl)))):
    O4VaP:=[O4Va[1,4],O4Va[2,4],O4Va[3,4]]:
    puntO4Va:=puntO4Va,O4VaP:

    O4Vb:=evalf(evalm(&*(O4V,Tx(-hl)))):
    O4VbP:=[O4Vb[1,4],O4Vb[2,4],O4Vb[3,4]]:
    puntO4Vb:=puntO4Vb,O4VbP:

    #               #               #               #               #
    O5a:=evalf(evalm(&*(O5,Tx(hl)))):
    O5aP:=[O5a[1,4],O5a[2,4],O5a[3,4]]:
    puntO5a:=puntO5a,O5aP:

    O5b:=evalf(evalm(&*(O5,Tx(-hl)))):
    O5bP:=[O5b[1,4],O5b[2,4],O5b[3,4]]:
    puntO5b:=puntO5b,O5bP:


    if i=1 then
    ##### Link O1 to O2N #####################################################

        LINK1i2N:=[[O1iP[1],O1iP[2],O1iP[3]],[O2PN[1],O2PN[2],O2PN[3]]]:
        link1i2N:=spacecurve(LINK1i2N,color=purple):
        S1:=sqrt((O2PN[1]-O1iP[1])^2+(O2PN[2]-O1iP[2])^2+(O2PN[3]-O1iP[3])^2):
```

```

        #######################################################################
        LINK22I:=[[O2P[1],O2P[2],O2P[3]],[O2PI[1],O2PI[2],O2PI[3]]]:
        link22I:=spacecurve(LINK22I,color=pink):


        #######################################################################
        LINK22F:=[[O2P[1],O2P[2],O2P[3]],[O2PF[1],O2PF[2],O2PF[3]]]:
        link22F:=spacecurve(LINK22F,color=pink):

        #######################################################################
        LINK44F:=[[O4P[1],O4P[2],O4P[3]],[O4PF[1],O4PF[2],O4PF[3]]]:
        link44F:=spacecurve(LINK44F,color=pink):


        #######################################################################

        ##### Link O2V to O3 ##################################################

        LINK2V3:=[[O2PV[1],O2PV[2],O2PV[3]],[O3P[1],O3P[2],O3P[3]]]:
        link2V3:=spacecurve(LINK2V3,color=purple):
        S2:=sqrt((O3P[1]-O2PV[1])^2+(O3P[2]-O2PV[2])^2+(O3P[3]-O2PV[3])^2):

        ##### Link O3 to O4 ###################################################

        LINK34:=[[O3P[1],O3P[2],O3P[3]],[O4P[1],O4P[2],O4P[3]]]:
        link34:=spacecurve(LINK34,color=purple):
        S3:=sqrt((O4P[1]-O3P[1])^2+(O4P[2]-O3P[2])^2+(O4P[3]-O3P[3])^2):

        ##### Link O4V to O5 ##################################################

        LINK4V5:=[[O4PV[1],O4PV[2],O4PV[3]],[O5P[1],O5P[2],O5P[3]]]:
        link4V5:=spacecurve(LINK4V5,color=purple):
        S4:=sqrt((O5P[1]-O4PV[1])^2+(O5P[2]-O4PV[2])^2+(O5P[3]-O4PV[3])^2):

        ##### Link O1i changhed to O5 ##### BUT US FIXED ####################

        LINK1i5:=[[O1iP[1],O1iP[2],O1iP[3]],[O5P[1],O5P[2],O5P[3]]]:
        link1i5:=spacecurve(LINK1i5,color=purple):
        S5:=sqrt((O5P[1]-O1iP[1])^2+(O5P[2]-O1iP[2])^2+(O5P[3]-O1iP[3])^2):

        ##### Displacement s2 #################################################

        DISP2:=[[O2P[1],O2P[2],O2P[3]],[O2PV[1],O2PV[2],O2PV[3]]]:
        disp2:=spacecurve(DISP2,color=orange):
        D2:=sqrt((O2P[1]-O2PV[1])^2+(O2P[2]-O2PV[2])^2+(O2P[3]-O2PV[3])^2):


        ##### Displacement s4 ################################################

        DISP4:=[[O4P[1],O4P[2],O4P[3]],[O4PV[1],O4PV[2],O4PV[3]]]:
        disp4:=spacecurve(DISP4,color=orange):
        D4:=sqrt((O4P[1]-O4PV[1])^2+(O4P[2]-O4PV[2])^2+(O4P[3]-O4PV[3])^2):

    fi;




    ##################### Crossing analysis Link 51i with 44F ##############
    nam:="NCC------For 51i44F":
    psdM51i44F:=col_dec_Is(O1iP,O5P,O4P,O4PF,nam,psdM51i44F,i,epsi):


    ##################### Crossing analysis Link 1i2N with 2V3 ############
    nam:="NCC------For 1i2N2V3":
    psdM1i2N2V3:=col_dec_Is(O1iP,O2PN,O2PV,O3P,nam,psdM1i2N2V3,i,epsi):


    ##################### Crossing analysis Link 2I2F with 44F ###########
    nam:="NCC------For 2I2F44F":
    psdM2I2F44F:=col_dec_Is(O2PI,O2PF,O4P,O4PF,nam,psdM2I2F44F,i,epsi):


    ##################### Crossing analysis Link 1i2N with 34 ############
    nam:="NCC------For 1i2N34":
    psdM1i2N34:=col_dec_Is(O1iP,O2PN,O3P,O4P,nam,psdM1i2N34,i,epsi):


        #NB The 2I is automatically changed and pose equal to the 2N
    ##################### Crossing analysis Link 2I2F with 34 ###########
    nam:="NCC------For 2I2F34":
    psdM2I2F34:=col_dec_Is(O2PI,O2PF,O3P,O4P,nam,psdM2I2F34,i,epsi):



    ##################### Crossing analysis Link 1i2N with 44F #############
    nam:="NCC------For 1i2N44F":
    psdM1i2N44F:=col_dec_Is(O1iP,O2PN,O4P,O4PF,nam,psdM1i2N44F,i,epsi):



    ##################### Crossing analysis Link 1i2N with 4V5 #############
    nam:="NCC------For 1i2N4V5":
    psdM1i2N4V5:=col_dec_Is(O1iP,O2PN,O4PV,O5P,nam,psdM1i2N4V5,i,epsi):



    ##################### Crossing analysis Link 2V3 with 44F ###############
    nam:="NCC------For 2V344F":
    psdM2V344F:=col_dec_Is(O2PV,O3P,O4P,O4PF,nam,psdM2V344F,i,epsi):



    ##################### Crossing analysis Link 2I2F with 4V5 #############
    nam:="NCC------For 2I2F4V5":
    psdM2I2F4V5:=col_dec_Is(O2PI,O2PF,O4PV,O5P,nam,psdM2I2F4V5,i,epsi):



    ##################### Crossing analysis Link 2I2F with 51i #############
    nam:="NCC------For 2I2F51i":
    psdM2I2F51i:=col_dec_Is(O2PI,O2PF,O1iP,O5P,nam,psdM2I2F51i,i,epsi):



    ##################### Crossing analysis Link 2V3 with 4V5 #############
    nam:="NCC------For 2V34V5":
```

```
    psdM2V34V5:=col_dec_Is(O2PV,O3P,O4PV,O5P,nam,psdM2V34V5,i,epsi):



    ##################### Crossing analysis Link 2V3 with 51i #############
    nam:="NCC------For 2V351i":
    psdM2V351i:=col_dec_Is(O2PV,O3P,O5P,O1iP,nam,psdM2V351i,i,epsi):


    ##################### Crossing analysis Link 34 with 4V5 #################
    nam:="NCC------For 344V5":
    psdM344V5:=col_dec_Is(O3P,O4P,O4PV,O5P,nam,psdM344V5,i,epsi):


#################### Crossing analysis Link 34 with 51i ###################
    nam:="NCC------For 3451i":
    psdM3451i:=col_dec_Is(O3P,O4P,O5P,O1iP,nam,psdM3451i,i,epsi):



############################ Checking the joint envelope################
################ Joint collision detetection ############################
    nam:="JOINTS------For J1 and J2V":
    psdMJ1J2V:=col_dec_Is(O1aP,O1bP,O2VaP,O2VbP,nam,psdMJ1J2V,i,epsj):

    nam:="JOINTS------For J1 and J3":
    psdMJ1J3:=col_dec_Is(O1aP,O1bP,O3aP,O3bP,nam,psdMJ1J3,i,epsj):

    nam:="JOINTS------For J1 and J4V":
    psdMJ1J4V:=col_dec_Is(O1aP,O1bP,O4VaP,O4VbP,nam,psdMJ1J4V,i,epsj):

    nam:="JOINTS------For J1 and J5":
    psdMJ1J5:=col_dec_Is(O1aP,O1bP,O5aP,O5bP,nam,psdMJ1J5,i,epsj):

    nam:="JOINTS------For J2V and J4V":
    psdMJ2VJ4V:=col_dec_Is(O2VaP,O2VbP,O4VaP,O4VbP,nam,psdMJ2VJ4V,i,epsj):

    nam:="JOINTS------For J2V and J5":
    psdMJ2VJ5:=col_dec_Is(O2VaP,O2VbP,O5aP,O5bP,nam,psdMJ2VJ5,i,epsj):

    nam:="JOINTS------For J3 and J4V":
    psdMJ3J4V:=col_dec_Is(O3aP,O3bP,O4VaP,O4VbP,nam,psdMJ3J4V,i,epsj):
    nam:="JOINTS------For J3 and J5":
    psdMJ3J5:=col_dec_Is(O3aP,O3bP,O5aP,O5bP,nam,psdMJ3J5,i,epsj):




################## Checking the joint envelope with sphere ##############
    #               #               #               #               #
D1i2V:=Dist(O1iP,O2PV);
if D1i2V<2*radiu then
    print(cat("The joint 1i and 2V are to close, the distance is:"...
    ,D1i2V,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D1i3:=Dist(O1iP,O3P);
if D1i3<2*radiu then
    print(cat("The joint 1i and 3 are to close, the distance is:"...
    ,D1i3,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D1i4:=Dist(O1iP,O4P);
if D1i4<2*radiu then
    print(cat("The joint 1i and 4 are to close, the distance is:"...
    ,D1i4,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D1i4V:=Dist(O1iP,O4PV);
if D1i4V<2*radiu then
    print(cat("The joint 1i and 4V are to close, the distance is:"...
    ,D1i4V,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #



            #               #               #               #
D1i5:=Dist(O1iP,O5P);
if D1i5<2*radiu then
    print(cat("The joint 1i and 5 are to close, the distance is:"...
    ,D1i5,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #



D23:=Dist(O2PV,O3P);
if D23<2*radiu then
    print(cat("The joint 2V and 3 are to close, the distance is:"...
    ,D23,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #

            #               #               #               #
D24:=Dist(O2PV,O4P);
if D24<2*radiu then
    print(cat("The joint 2V and 4 are to close, the distance is:"...
    ,D24,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D24V:=Dist(O2PV,O4PV);
if D24V<2*radiu then
    print(cat("The joint 2V and 4V are to close, the distance is:"...
    ,D24V,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #



            #               #               #               #
D25:=Dist(O2PV,O5P);
if D25<2*radiu then
    print(cat("The joint 2V and 5 are to close, the distance is:"...
    ,D25,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D2V4:=Dist(O2PV,O4P);
if D2V4<2*radiu then
    print(cat("The joint 2V and 4 are to close, the distance is:"...
    ,D2V4,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


D2V4V:=Dist(O2PV,O4PV);
if D2V4V<2*radiu then
    print(cat("The joint 2V and 4V are to close, the distance is:"...
    ,D2V4V,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #



            #               #               #               #
D2V5:=Dist(O2PV,O5P);
if D2V5<2*radiu then
    print(cat("The joint 2V and 5 are to close, the distance is:"...
    ,D2V5,"At the frame",i));
        flag_det:=1;
fi;
            #               #               #               #


            #               #               #               #
D34V:=Dist(O3P,O4PV);
if D34V<2*radiu then
    print(cat("The joint 3 and 4V are to close, the distance is:"...
    ,D34V,"At the frame",i));
        flag_det:=1;
fi;
```

```
                #                   #                   #                   #



                #                   #                   #                   #
D35:=Dist(O3P,O5P);
if D25<2*radiu then
    print(cat("The joint 3 and 5 are to close, the distance is:"...
    ,D35,"At the frame",i));
        flag_det:=1;
fi;
                #                   #                   #                   #

D45:=Dist(O4PV,O5P);
if D45<2*radiu then
    print(cat("The joint 4V and 5 are to close, the distance is:"...
    ,D45,"At the frame",i));
        flag_det:=1;
fi;
                #                   #                   #                   #



###################### Finishing the cicle for ####################
od:




############### Is the link 5 to 1 ##################################
LINK:=[[O1iP[1],O1iP[2],O1iP[2]],[O5P[1],O5P[2],O5P[3]]]:
link:=spacecurve(LINK,title=" LINK between O5 and O1",color=violet,...
labels=[x, y, z],titlefont = ["ROMAN", 15],scaling=constrained):




ori2:=spacecurve([puntO2],title="O2 joint center",color=black,...
labels=[x, y, z],titlefont = ["ROMAN", 15],scaling=constrained):
display({ori2},{ori},thickness=8);

ori3:=spacecurve([puntO3],title="O3 joint center",color=green,...
labels=[x, y, z],
titlefont = ["ROMAN", 15],scaling=constrained):
display({ori3},{ori},thickness=8);

ori4:=spacecurve([puntO4],title="O4 joint center",color=yellow,...
labels=[x, y, z],
titlefont = ["ROMAN", 15],scaling=constrained):
display({ori4},{ori},thickness=8);

ori5:=spacecurve([puntO5],title="O5 joint center",color=coral,...
labels=[x, y, z],
titlefont = ["ROMAN", 15],scaling=constrained):
display({ori5},{ori},thickness=8);

display({ori5},{ori},{link},thickness=8,title="O5 and O1 ...
with link 5 to 1");

```

```
display({ori},{ori5},{ori2},{ori3},{ori4},thickness=8,...
title="All shapes of the reference frames");
Allshapes:=display({ori},{ori5},{ori2},{ori3},{ori4},...
thickness=8,title="All shapes of the reference frames"):

lg := mrow(          mn("\n      ",mathbackground=red),
                     mn("  "), Typesetting:-Typeset(O1),

                     mn("\ \      ",mathbackground=black),
                     mn("  "), Typesetting:-Typeset(O2),

                     mn("\n      ",mathbackground=green),
                     mn("  "), Typesetting:-Typeset(O3),

                     mn("      ",mathbackground=yellow),
                     mn("  "), Typesetting:-Typeset(O4),

                     mn("\n      ",mathbackground=coral),
                     mn("  "), Typesetting:-Typeset(O5),

                     mn("\n      ",mathbackground=purple),
                     mn("\ \  "), Typesetting:-Typeset(Link51)):

display({ori},{ori5},{ori2},{ori3},{ori4},{link},thickness=8,...
title="All shapes with link 51",caption=lg);

print("Identity matrix");
i:=33:
evalf(evalm(&*(Rx(Mconfig[i,1]),Tx(ss[1]),Tz(as[2]),Rz(alfasd[2]),...
Rx(Mconfig[i,2]),Tx(Mconfig[i,6]),Tz(as[3]),Rz(alfasd[3]),...
Rx(Mconfig[i,3]),Tx(ss[3]),Tz(as[4]),Rz(alfasd[4]),...
Rx(Mconfig[i,4]),Tx(Mconfig[i,7]),Tz(as[5]),Rz(alfasd[5]),...
Rx(Mconfig[i,5]),Tx(ss[5]),Tz(as[1]),Rz(alfasd[1]))));

display({ori},{ori5},{ori2},{ori3},{ori4},{link},{link1i2N},...
{link2V3},{link34},{link4V5},{link22F},{link22I},{link44F},{disp2},...
{disp4},thickness=8,title="All shapes with ALL THE LINKS...
 in a generic instant");



display({ori},{ori5},{link},{link1i2N},{link2V3},{link34},...
{link4V5},{link22F},{link22I},{link44F},{disp2},{disp4},...
thickness=8,title="All shapes with,Only, ALL THE LINKS ...
in a generic instant");

########### Costruction Video with animate #####################

### Build the single animations of the
##links (starting from the link 1 to 2) ###########

#   #   #   #   #   #   #   #   #   #   #   Link O1 to O2   #   #   #
l1i1f := [seq([puntO1f[t],puntO1i[t]], t = 1 .. k-1)]:
L1i1f:=animate(spacecurve, [l1i1f [trunc(j)]],j=1..k-1,frames=k-1,...
thickness=2,color=green):


#   #   #   #   #   #   #   #   #   #   #   Link O1 to O2N  #   #   #
l1i2N := [seq([puntO1i[t],puntO2N[t]], t = 1 .. k-1)]:
```

```
L1i2N:=animate(spacecurve, [l1i2N[trunc(j)]],j=1..k-1,frames=k-1,...
thickness=5,color=purple):


#   #   #   #   #   #   #   #   #   #   #   Link O2 to O2V  #   #   #
l22V := [seq([puntO2[t],puntO2V[t]], t = 1 .. k-1)]:
L22V:=animate(spacecurve, [l22V[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=red):

#   #   #   #   #   #   #   #   #   #   #   Link O2I to O2F #   #   #
l2I2F := [seq([puntO2I[t],puntO2F[t]], t = 1 .. k-1)]:
L2I2F:=animate(spacecurve, [l2I2F[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=pink):

#   #   #   #   #   #   #   #   #   #   #   Link O2V to O3  #   #   #
l2V3 := [seq([puntO2V[t],puntO3[t]], t = 1 .. k-1)]:
L2V3:=animate(spacecurve, [l2V3[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=purple):

#   #   #   #   #   #   #   #   #   #   #   Link O3 to O4   #   #   #
l34 := [seq([puntO3[t],puntO4[t]], t = 1 .. k-1)]:
L34:=animate(spacecurve, [l34[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=purple):

#   #   #   #   #   #   #   #   #   #   #   Link O4 to O4F  #   #   #
l44F := [seq([puntO4[t],puntO4F[t]], t = 1 .. k-1)]:
L44F:=animate(spacecurve, [l44F[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=pink):

#   #   #   #   #   #   #   #   #   #   #   Link O4 to O4V  #   #   #
l44V := [seq([puntO4[t],puntO4V[t]], t = 1 .. k-1)]:
L44V:=animate(spacecurve, [l44V[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=red):

#   #   #   #   #   #   #   #   #   #   #   Link O4V to O5  #   #   #
l4V5 := [seq([puntO4V[t],puntO5[t]], t = 1 .. k-1)]:
L4V5:=animate(spacecurve, [l4V5[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=purple):


#   #   #   #   #   #   #   #   #   #   #   Link O5 to O1   #   #   #
l51i := [seq([puntO5[t],puntO1i[t]], t = 1 .. k-1)]:
L51i:=animate(spacecurve, [l51i[trunc(j)]],j=1..k-1,frames=k-1,thickness=5,...
color=violet):

######################################### JOINT ########################
#   #   #   #   #   #   #   #   #   #   #   Joint O1    #   #   #   #
j1 := [seq([puntO1a[t],puntO1b[t]], t = 1 .. k-1)]:
J1:=animate(spacecurve, [j1[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O2    #   #   #   #
j2 := [seq([puntO2a[t],puntO2b[t]], t = 1 .. k-1)]:
J2:=animate(spacecurve, [j2[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O2V   #   #   #   #
j2V := [seq([puntO2Va[t],puntO2Vb[t]], t = 1 .. k-1)]:
J2V:=animate(spacecurve, [j2V[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O3    #   #   #   #
j3:= [seq([puntO3a[t],puntO3b[t]], t = 1 .. k-1)]:
J3:=animate(spacecurve, [j3[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O4    #   #   #   #
j4:= [seq([puntO4a[t],puntO4b[t]], t = 1 .. k-1)]:
J4:=animate(spacecurve, [j4[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O4V   #   #   #   #
j4V:= [seq([puntO4Va[t],puntO4Vb[t]], t = 1 .. k-1)]:
J4V:=animate(spacecurve, [j4V[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

#   #   #   #   #   #   #   #   #   #   #   Joint O5    #   #   #   #
j5:= [seq([puntO5a[t],puntO5b[t]], t = 1 .. k-1)]:
J5:=animate(spacecurve, [j5[trunc(j)]],j=1..k-1,frames=k-1,thickness=think,...
color=blue):

display(L1i2N,L2I2F,L22V,L2V3,L34,L44F,L44V,L4V5,L51i,ori,...
Allshapes,title="Animation of the links with all motion ...
shapes of the reference frames",orientation=[75,25,0]);


display(L1i2N,L2I2F,L22V,L2V3,L34,L44F,L44V,L4V5,L51i,...
ori,title="Animation of the links",orientation=[75,25,0]);

AllLinks:=display(L1i2N,L2I2F,L22V,L2V3,L34,L44F,L44V,...
L4V5,L51i,ori,title="Animation of the links",...
orientation=[75,25,0]):

display(J1,AllLinks,J2V,J3,J4V,J5,title="With joint"...
,orientation=[75,25,0]);



if flag_det<>1 then
    print("NO COLLISION has been detected");
    else
    print("COLLISION has been detected, look up");
fi;



vecto:=seq(i, i = 1..k-1):





#   #   #   #   #   #   #   #   #       #   Link O4 to O4F  #   #   #
l44F := [seq([puntO4[t],puntO4F[t]], t = 1 .. k-1)]:
L44F:=animate(spacecurve, [l44F[trunc(j)]],j=1..k-1,frames=k-1,...
trace=[vecto],thickness=20,color=pink):

display(L44F,   L1i2N,L44V,L4V5,L51i,ori,L1i1f,title="x-axis of ..
the first reference frame",orientation=[75,25,0]);

print(cat("With a multiplication factor of:",mf));
print(cat("the length of the link 1 to 2:",S1));
print(cat("the length of the link 2 to 3:",S2));
print(cat("the length of the link 3 to 4:",S3));
print(cat("the length of the link 4 to 5:",S4));
print(cat("the length of the link 5 to 1:",S5));
print(cat("the displacement s4 is:",MaxS4));
print(cat("the displacement s2 is:",MaxS2));
```

Appendix E

```
%% tPosition_Velocity_v02.m Control Position and velocity
%% Reset Section
close all
clear all
clc
%% Chose the motor that you want to move
ID=3;
%% Gain and scale factor
Kp=1/0.2933;        %degrees-->volt signal
Kv=8.9737;          %rpm------>volt signal
Kg=10;              %gain
K=Kg*Kp;            %global gain

%% Reset position of robot
run treset_position_ID.m %reset to 0 position
pause(2)
%% Create handles structure
fDynamixelGolbalVariables();
% Registers
handles=fDynamixelInstructionsMemoryStruct();

%%  set USB2Dynamixel comunication parameters
handles.DEFAULT_PORTNUM = 5;          % Com Port of
% handles.DEFAULT_BAUDNUM = 1;        %   1Mbps Baud rate
handles.DEFAULT_BAUDNUM = 34;         %   57142bps Baud rate (0x22)

%% set win64 or win32  to be used
% handles=fDynamixel_OS_USED(handles,'win32'); % Uncomment to chose 32-bit
% system
handles=fDynamixel_OS_USED(handles,'win64');

%% start usb comunication
loadlibrary(handles.AliasLib,handles.AliasLib_h)    % open library

libfunctions(handles.AliasLib)                 % Display Library Functions

response=fDynamixel_OpenDevice(handles);  % if response=1 device opened and ok


%% Settimg Mode
run tSetting_Wheel_Mode_v01
%% Closed Loop
vel=0;          %Initializing velocity value
```

```
i=1;
TIME=0;         %Initializing time value
run  Accelerazione_costante_a_tratti_control_qf_v02
format shortg
c = clock;

while TIME<tf-1e-3
format shortg
d = clock;
%pause(0.0001)

TIME=d(6)-c(6)

run tCheck_position_v01 %Check the real position and velocity of the mechanism
%in rad and rad/s

 %Read the real position of the robot
 WORD_POS_PRESENT_POSITION=36;
 [PRESENT_POSITION_VALUE, StatusError, RXError]=fDynamixelReadWord(handles,ID...
     ,WORD_POS_PRESENT_POSITION);
 real_position=PRESENT_POSITION_VALUE*1/Kp;  %real position form volt-->degrees
 real_position_vec(i)=real_position;    %save in a row vector the real position

 err=qq*(360/2/pi)-real_position;               %Error is in degrees
 vel_volt=qqp*30/pi*Kv;           %from rad/s -->rpm--> volt signal (0-->2047)

%Speed Control
disp('Move to value motor _ID  ')
    vel=vel_volt+K*err;% in rpm


%% Saturation section
    if vel>1023
        vel=1023;
    end


    if vel<0
       if vel<-1023
           vel=2047;
       else
           vel=1023-vel;
       end
    end

%% Write the velovity

    val_vel=vel;
    WORD_VALUE_VEL=ceil(val_vel);
    WORD_POS_VEL=handles.Table.Moving_Speed_L; % Position Number in RAM
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_VEL),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_VEL,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_VEL,...
        WORD_VALUE_VEL);
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_VEL,...
        WORD_VALUE_VEL, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end





% Sampling Data
%Read the real speed of the robot
WORD_POS_PRESENT_SPEED=38;
[PRESENT_SPEED_VALUE, StatusError, RXError]=fDynamixelReadWord(handles,ID,...
    WORD_POS_PRESENT_SPEED);
real_speed_vec(i)=PRESENT_SPEED_VALUE/Kv;        %convert volt ----> rpm

err_vec(i)=PRESENT_SPEED_VALUE/Kv-qqp*30/pi;
%Check the value of the velocity error
i=i+1;

end



 %set ZERO Velocita
disp('Move to value motor _ID  ')
    vel=0;                    %0 value as final value of the velocity
    val_vel=vel*Kv;           %rpm--> volt
    WORD_VALUE_VEL=ceil(val_vel);
    WORD_POS_VEL=handles.Table.Moving_Speed_L; % Position Number in RAM
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_VEL),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_VEL,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_VEL,...
        WORD_VALUE_VEL);
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_VEL,...
        WORD_VALUE_VEL, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end

%% Close all Tool
calllib(handles.AliasLib,'dxl_terminate');
unloadlibrary(handles.AliasLib);
disp('Closed USB2Dynamixel!');

%% Plot Section
figure(1)
subplot(211)
plot(TTT,qqq*180/pi,TTT,real_position_vec,'LineWidth',1);grid on
xlabel('t(s)');ylabel('° degrees');legend('Theoretical ','Real')
subplot(212)
plot(TTT,qqqp*30/pi,TTT,real_speed_vec,'LineWidth',1);grid on
xlabel('t(s)');ylabel('rpm')

```

```
figure(2)
plot(TTT,abs(err_vec));grid on
```

```
%
clc
fDynamixelGolbalVariables();

%% create handles structure

% registers
 handles=fDynamixelInstructionsMemoryStruct();

% ini values of motors
% handles.IniMotors=fDynamixelIniMotorsValues();
%handles.MemStruc.IniMotors=fDynamixelIniMotorsValuesViejos();

%  set USB2Dynamixel comunication parameters
%(Check the port number from the advanced settig of computer)
handles.DEFAULT_PORTNUM = 5;   % Com Port
% handles.DEFAULT_BAUDNUM = 1;   %   1Mbps Baud rate
% handles.DEFAULT_BAUDNUM = 34;   %   57142bps Baud rate  (0x22)
handles.DEFAULT_BAUDNUM = 34;   %


% set win64 or win32  to be used
% handles=fDynamixel_OS_USED(handles,'win32');
handles=fDynamixel_OS_USED(handles,'win64');

%% start usb comunication
loadlibrary(handles.AliasLib,handles.AliasLib_h)   % open library

libfunctions(handles.AliasLib)                % Display Library Functions

response=fDynamixel_OpenDevice(handles);% if response = 1  device opened and ok




run tSetting_Joint_Mode_v01
%% ping ID motor  for n iterations

disp('Move to value motor _ID  ')

%      ID=1;
    %ID=5;
    b=0; % insert the position in degrees
    aa=b/0.2933; %scale with the factor
    WORD_VALUE=ceil(aa);
    WORD_POS=handles.Table.Goal_Position_L;% position number in the RAM
 disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',num2str(WORD_VALUE)...
     ,' HEXvalue:',num2str(dec2hex(WORD_VALUE,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS,...
        WORD_VALUE);
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS,...
        WORD_VALUE, StatusError);
```

```
else
    disp('Failed to open USB2Dynamixel!');
end


calllib(handles.AliasLib,'dxl_terminate');
unloadlibrary(handles.AliasLib);
disp('Closed USB2Dynamixel!');
```

```
%% Settimg Joint Mode
WORD_VALUE_CW=0;
    WORD_POS_CW=handles.Table.CW_Angle_Limit_L; %RAM position number
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_CW),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_CW,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_CW,...
        WORD_VALUE_CW); %%Motion Execution
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_CW,...
        WORD_VALUE_CW, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end



WORD_VALUE_CCW=1023;
    WORD_POS_CCW=handles.Table.CCW_Angle_Limit_L;%RAM position number
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_CCW),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_CCW,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_CCW,...
        WORD_VALUE_CCW); %%Motion Execution
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_CCW,...
        WORD_VALUE_CCW, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end
```

```
%% Settimg Wheel Mode
WORD_VALUE_CW=0;
    WORD_POS_CW=handles.Table.CW_Angle_Limit_L;%RAM position number
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_CW),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_CW,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_CW,...
        WORD_VALUE_CW); %%Qui viene eseguito il movimento
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_CW,...
        WORD_VALUE_CW, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end



WORD_VALUE_CCW=0;
    WORD_POS_CCW=handles.Table.CCW_Angle_Limit_L;%RAM position number
    disp(strcat('Move motor ID:',num2str(ID),' to DECvalue:',...
        num2str(WORD_VALUE_CCW),' HEXvalue:',...
        num2str(dec2hex(WORD_VALUE_CCW,4)),' position' ));

if response == 1 % if port opens okay
    disp('USB2Dynamixel Opened');
    [StatusError, RXError]=fDynamixelWriteWord(handles,ID,WORD_POS_CCW,...
        WORD_VALUE_CCW); %%Qui viene eseguito il movimento
    fDynamixelDispInstructionWordResult('Write',ID,handles,WORD_POS_CCW,...
        WORD_VALUE_CCW, StatusError);
else
    disp('Failed to open USB2Dynamixel!');
end
```

```
%% Trapezoidal_theoretical_shape.m
%% Set input Parameters
%qi=0;                            %initial position Set in treset_position_ID.m
qf=deg2rad(200) ;                 %final position %Max 300 degrees
tf=6;                             %sec
Vel_Max=8;                        %in rpm
qpmax=Vel_Max*pi/30;              %max 11.93 rad/s
% Maximum velocity is 114 rpm ---->11.93 rad/s
tauv=tf-(qf-qi)/qpmax;
tauw=tf-tauv;

% Definition of the times vectors
tau1=linspace(0,tauv,100);
tau2=linspace(tauv,tauw,10);
tau3=linspace(tauw,tf,100);
tau=[tau1 tau2 tau3];




  qppmax=qpmax/tauv; %acceleration in rad/sec^2
% qppmax=30;
% qpmax=qppmax*tauv;

%% Leggi a tratti

% I Branch
qI=qi+1/2*qppmax*tau1.^2;
qpI=qppmax*tau1;
qppI=qppmax*ones(1,length(tau1));

% II Branch
qII=qI(end)+qpmax*(tau2-tauv);
qpII=qpmax*ones(1,length(tau2));
qppII=0*ones(1,length(tau2));
```

```
% % III Branch
qIII=qII(end)-1/2*qppmax*(tau3-tauw).^2+qpmax*(tau3-tauw);
qpIII= qpmax-qppmax*(tau3-tauw);
qppIII=-qppmax*ones(1,length(tau1));

%% Assemblaggi
%Posizione
q=[qI qII qIII];                %rad
qp=[qpI qpII qpIII];            %rad/s
qpp=[qppI qppII qppIII];        %rad/sec^2

%% Grafici di controllo
figure(44)

plot(tau,q,tau,qp,tau,qpp,'LineWidth',1); grid on
legend('Position [rad]','Velocity [rad/s]','Acceleration[rad/s^2]');
xlabel('t(s)')
```

Acknowledgement

This thesis was carried out in the research center *"Institut de Robotica i Informatica Industrial (I.R.I.)"* in cooperation with the *"Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa (U.P.C.-E.S.E.I.A.A.T.)"*, during the mobility period of Erasmus between U.P.C. and Politecnico di Torino.

I would like to express my deep gratitude to Prof. E. Celaya and Prof. F. Thomas, my thesis host supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this work. I would like to offer my special thanks to Prof. P. Grosch, for his advice and assistance in keeping my progress on experimental activity. I would also like to acknowledge Prof. R.M. Planas of the ESEIAAT University as second host supervisor of this thesis. I would also like to extend my thanks to the technicians of the laboratory of the IRI research center for their help in offering me the resources in running the schedule. I would like to express my appreciation to Prof. S. Pastorelli for his availability and kindness as home supervisor of this work.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

Bibliography

[1] E. Celaya. *Solution intervals for variables in spatial RCRCR linkages.* Mechanism and Machine Theory, vol. 133, pages 481-492, 2019.

[2] H.C. Cheng and S. Thompson. *Computer-aided displacement analysis of spatial mechanisms using the CH programming language*, volume 23. Advanced in Engeneering Software, pages 163-172, 1995.

[3] J. Duffy. *Analysis of mechanisms and robot manipulators.* John Wiley & Sons, Incorporated, 1980.

[4] J. Duffy and H. Habibolahi. *A displacement analysis of spatial five-link 3R-2C mechanisms part 2: Analysis of the RRCRC Mechanism*, volume 6. Journal of Mechanisms,pages 463-473, 1971.

[5] I. S. Fischer. *A geometric method for determining joint rotations in the inverse kinematics of robotic manipulators.* 2000.

[6] grabcad.com. *https://grabcad.com/.*

[7] igus.com. *https://www.igus.com/.*

[8] I.J. Lee. *The RCRRC Five-Link Space Mechanism-Displacement Analysis, Force and Torque Analysis and Its Transmission Criteria.* Journal of Mechanisms, pages 581-594, 1975.

[9] maplesoft.com. *https://www.maplesoft.com/products/Maple/.*

[10] robotis.com. *http://www.robotis.us/ax-12a/.*

[11] rs online.com. *https://es.rs-online.com/web.*

[12] skf.com. *https://www.skf.com/group/splash/index.html.*

[13] stratasys.com. *https://www.stratasys.com/.*

[14] F. Thomas and C. Torras. *A projectively invariant intersection test for polyhedra.* The Visual Computer, pages 1-9, 2002.

[15] T. A. Yang. *Displacement Analysis of Spatial Five-Link Mechanisms Using (3x3) Matrices With Dual-Number Elements*, volume 91. Journal of Engineering for Industry, pages 152-156, 1969.

[16] M. S.C. Yuan. *Displacement Analysis of the RRCCR Five-Link Spatial Mechanism*, volume 6. Journal of Mechanisms, pages 119-133, 1971.