Instituto Tecnologico y de Estudios Superiores de Monterrey

Department of Computer Engineering

Robotics Final Project

Kinematic and dynamic analysis for a serial mechanism

Authors
Noé Marcelo Yungaicela N.
David Absalón Duarte D.
Henry Gianfranco Niquín H.

Professor
Dr. Ernesto Rodríguez Leal

May 11, 2017

Abstract

This document presents a serial robot mechanism with three degrees of freedom. The analyses performed were: forward and inverse dynamics, forward and inverse kinematics, mobility analysis through screw theory, both static and dynamic. The robot mechanism has three contained rotational degrees of freedom, with two collinear axes perpendicular to a rotation axis. The mathematical models created were tested through simulations performed in Solid Works.

Contents

List of Figures

1. Introduction

The construction of robot mechanisms requires the analysis of the mathematical models of the kinematics and dynamics of the structure in order to properly size the actuators. Also, a mobility analysis is important in order to establish the scope of the mechanism.

In this paper we present the complete study of a robotic arm, which has three rotational degrees of freedom. The first two degrees of freedom are on the same plane, which is perpendicular to the plane where the other degree of freedom lies. The analyses carried out were: forward and inverse kinematics, instantaneous dynamics, and both static and dynamic mobility.

In low-order kinematics , we study the relations between the parameters of the joints and the positions and orientations of the final effector with respect to an inertial frames of reference. There are two approaches to kinematic analysis: On one hand, direct kinematics calculates the position and orientation of the effector, given a configuration of the parameters of the joints. On the other hand, inverse kinematics calculates the required configuration at the joints, according to a specific position and orientation of the effector with respect to the overall frame of reference.

The instantaneous high-order kinematics relates to the linear and angular velocities of the components of the final effector, with respect to the global frame of reference, with the angular velocities of the joint parameters. Analogously as in low-order kinematics, in this instantaneous analysis there are inverse and forward studies, generating the same concepts described above. Through a detailed manipulation of the equations it is possible to obtain the Jacobian matrix, which represents the relations in instantaneous kinematics, in the forward approach and possibly in the inverse one.

In the study of dynamics the interactions of linear and angular forces in the joints of the mechanism were analyzed. This analysis allows to establish the dimensions of limbs and actuators according to the specifications that the application of the algorithm demand.

The mobility analysis helps to identify the reaches in the mechanism's workspace. Many spaces may be restricted to the mechanism as well as to its speed. It is interesting to notice that screw theory simplifies this analysis to a great extent.

Each model developed is tested through numerical simulations using the SOLIDWORKS software and the PYTHON language is used for numerical comparisons. In addition, the design was printed in 3D for physical testing. Servomotors were programmed in Arduino to serve as actuators.

Objectives

This project aims to analyze a serial mechanism, by implementing the mathematical models and simulation through the SolidWorks software to evaluate the results. Finally, the mechanism was constructed through 3D printing to demonstrate the results obtained from numerical analysis.

2. Robot Schematics

The schematics of the serial robot are shown on figure 2.1. It is a mechanism with three rotational degrees of freedom, where two axes are contained in the same plane, that is perpendicular to the plane that contains the other rotation axis.

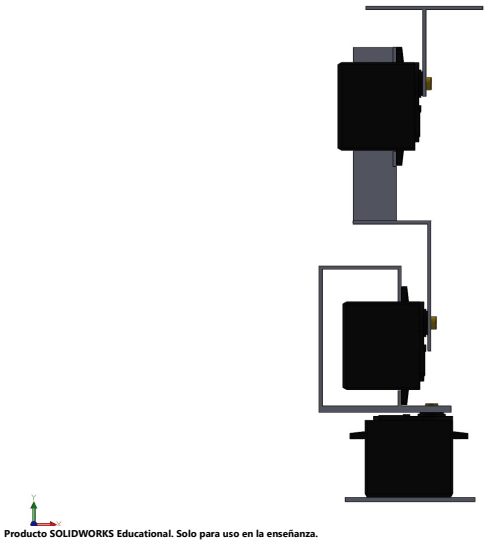

Figure 2.1.: Robot Mechanism

Each link has a different length; the measures of said links are as follows:

- $d_1 = 46.25mm$, first link, counting upwards from base
- $a_1 = 82.25mm$, second link
- $a_2 = 34mm$, third link, containing the final effector

Figure 2.1 shows the diagram of the mechanism with reference frames. The global frame is set in the base of the robot.

Figure 2.3 shows the relationships between the reference frames assigned to each joint.

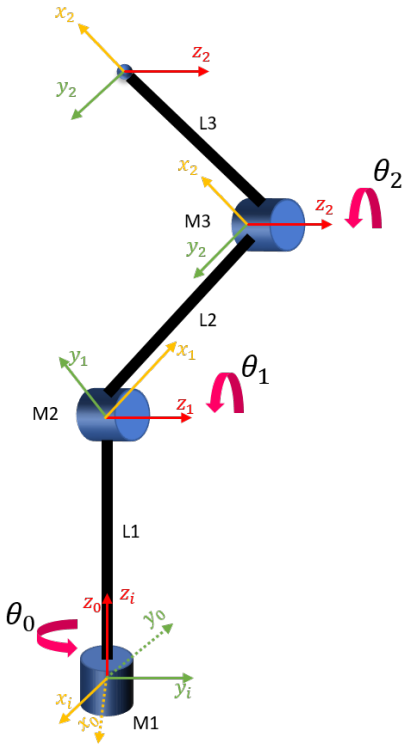

Figure 2.2.: Robot Diagram

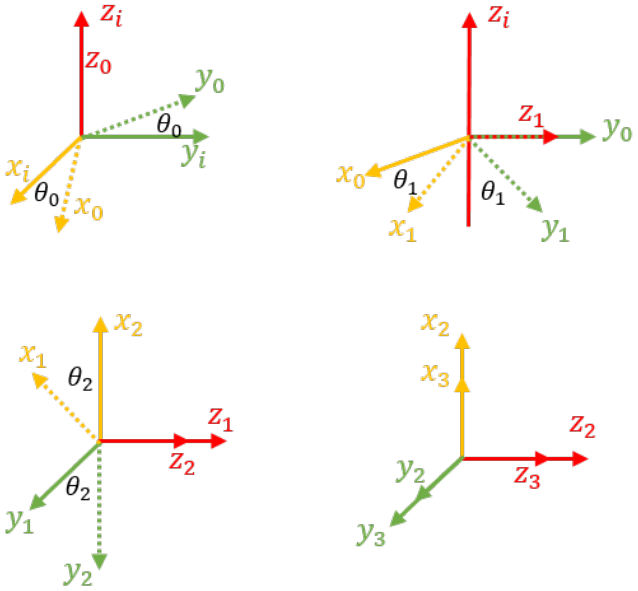

Figure 2.3.: Coordinates Transformation

3. Forward Kinematics

Forward kinematics were analyzed using the Denavit-Hartenberg procedure in addition to the general matrix process, obtaining equivalent results. The applied equations are presented below.

3.1. Denavit-Hartenberg Representation

By substituting the parameters obtained from the model into the Denavit-Hartenberg matrix, the corresponding transformation matrices are obtained.

$$DH = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{(i-1)} \\ \sin(\theta_i)\cos(\alpha_{(i-1)}) & \cos(\theta_i)\cos(\alpha_{(i-1)}) & -\sin(\alpha_{(i-1)}) & -\sin(\alpha_{(i-1)}d_i) \\ \sin(\theta_i)\sin(\alpha_{(i-1)}) & \cos(\theta_i)\sin(\alpha_{(i-1)}) & \cos(\alpha_{(i-1)}) & -\sin(\alpha_{(i-1)}d_i) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$${}^{i}_{0}T = \begin{bmatrix} \cos(\theta_0) & -\sin(\theta_0) & 0 & 0 \\ \sin(\theta_0) & \cos(\theta_0) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$${}^{0}_{1}T = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin(\theta_1) & -\cos(\theta_1) & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$${}^{1}_{2}T = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$
{}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}
$$

From the multiplication of each of the intermediate matrices, the general matrix can be generated.

$$
T = ({}^{i}_{0}T)({}^{0}_{1}T)({}^{1}_{2}T)({}^{2}_{3}T) \tag{3.6}
$$

$$
T = \begin{bmatrix} \cos(\theta_0)\cos(\theta_1+\theta_2) & -\cos(\theta_0)\sin(\theta_1+\theta_2) & -\sin(\theta_0) & \cos(\theta_0)(a_1\cos(\theta_1)+a_2\cos(\theta_1+\theta_2)) \\ \sin(\theta_0)\cos(\theta_1+\theta_2) & -\sin(\theta_0)\sin(\theta_1+\theta_2) & \cos(\theta_0) & \sin(\theta_0)(a_1\cos(\theta_1)+a_2\cos(\theta_1+\theta_2)) \\ -\sin(\theta_1+\theta_2) & -\cos(\theta_1+\theta_2) & 0 & -a_1\sin(\theta_1)-a_2\sin(\theta_1+\theta_2)+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}
$$

3.2. Using the General Matrix

From the general reference frame (x_i, y_i, z_i), local reference frame (x_0, y_0, z_0) is obtained using the following matrix:

$$
{}^{i}_{0}T = \begin{bmatrix} \cos(\theta_0) & -\sin(\theta_0) & 0 & 0 \\ \sin(\theta_0) & \cos(\theta_0) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}
$$

where:

$$
{}^{i}_{0}R = \begin{bmatrix} \cos(\theta_0) & -\sin(\theta_0) & 0 \\ \sin(\theta_0) & \cos(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}
$$

represents the rotation matrix from frame 0 to i.

From reference frame (x_0, y_0, z_0), coordinates (x_1, y_1, z_1) are obtained by performing the following transformations:

$$ {}^{0}_{1}R = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} \tag{3.10} $$

$$ {}^{0}_{1}R = \begin{bmatrix} cos(-\theta_1) & cos(90-(-\theta_1)) & 0 \\ 0 & 0 & 1 \\ sin(-\theta_1) & -sin(90-(-\theta_1)) & 0 \end{bmatrix} = \begin{bmatrix} cos(\theta_1) & -sin(\theta_1)) & 0 \\ 0 & 0 & 1 \\ -sin(\theta_1) & -cos(\theta_1) & 0 \end{bmatrix} \tag{3.11} $$

The resulting homogeneous transformation matrix is:

$$ {}^{0}_{1}T = \left[\begin{array}{ccc|c} cos(\theta_1) & -sin(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin(\theta_1) & -cos(\theta_1) & 0 & d_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.12} $$

From reference frame (x_0, y_0, z_0), coordinates (x_1, y_1, z_1) are calculated using the following expressions:

$$ {}^{1}_{2}R = \begin{bmatrix} x_2 \cdot x_1 & y_2 \cdot x_1 & z_2 \cdot x_1 \\ x_2 \cdot y_1 & y_2 \cdot y_1 & z_2 \cdot y_1 \\ x_2 \cdot z_1 & y_2 \cdot z_1 & z_2 \cdot z_1 \end{bmatrix} \tag{3.13} $$

$$ {}^{1}_{2}R = \begin{bmatrix} cos(-\theta_2) & cos(90-(-\theta_2)) & 0 \\ -sin(-\theta_2) & sin(90-(-\theta_2)) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2)) & 0 \\ sin(\theta_2) & cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14} $$

The resulting homogeneous transformation matrix is:

$$ {}^{1}_{2}T = \left[\begin{array}{ccc|c} cos(\theta_2) & -sin(\theta_2) & 0 & a_1 \\ sin(\theta_2) & cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.15} $$

Finally, for the rotation-less effector with a single translation, the following is acquired:

$$
{}^{2}_{3}T = \left[\begin{array}{cccc} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.16}
$$

Thus, the homogeneous transformation matrix from the origin to the final effector is:

$$
{}^{i}_{3}T = ({}^{0}_{i}T)({}^{0}_{1}T)({}^{1}_{2}T)({}^{2}_{3}T) \tag{3.17}
$$

$$
{}^{i}_{3}T = \begin{bmatrix} \cos(\theta_0)\cos(\theta_1+\theta_2) & -\cos(\theta_0)\sin(\theta_1+\theta_2) & -\sin(\theta_0) & \cos(\theta_0)(a_1\cos(\theta_1)+a_2\cos(\theta_1+\theta_2)) \\ \sin(\theta_0)\cos(\theta_1+\theta_2) & -\sin(\theta_0)\sin(\theta_1+\theta_2) & \cos(\theta_0) & \sin(\theta_0)(a_1\cos(\theta_1)+a_2\cos(\theta_1+\theta_2)) \\ -\sin(\theta_1+\theta_2) & -\cos(\theta_1+\theta_2) & 0 & -a_1\sin(\theta_1)-a_2\sin(\theta_1+\theta_2)+d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}
$$

Parting from this analysis, consistent with the results obtained by the Denavit–Hartenberg methodology, we proceed with the inverse kinematics analysis.

3.3. Inverse Kinematics

The presented work uses the uncoupling technique proposed by [1] to solve the inverse kinematics equations. From the equations of the transformation matrices a component of the general matrix can be isolated under the following valid relationship:

$$
A_1^{-1}[RHS] = A_2A_3A_4 \tag{3.19}
$$

Where, A_1, A_2, A_3 and A_4, represent the equations ${}^{i}_{0}T$, ${}^{0}_{1}T$, ${}^{1}_{2}T$ and ${}^{2}_{3}T$ respectively, defined by the equations 3.8,3.12, 3.15, 3.16. And RHS is the general transformation matrix, for the final effector, parametrized for the position and orientation values to the final point with respect to the global reference frame.

Expanding the left side of equation 3.19 the following is obtained:

$$A_1^{-1}[RHS] = \begin{bmatrix} \cos(\theta_0) & \sin(\theta_0) & 0 & 0 \\ -\sin(\theta_0) & \cos(\theta_0) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

$$= \begin{bmatrix} n_x\cos(\theta_0) + n_y\sin(\theta_0) & o_x\cos(\theta_0) + o_y\sin(\theta_0) & a_x\cos(\theta_0) + a_y\sin(\theta_0) & p_x\cos(\theta_0) + p_y\sin(\theta_0) \\ n_y\cos(\theta_0) - n_x\sin(\theta_0) & o_y\cos(\theta_0) - o_x\sin(\theta_0) & a_y\cos(\theta_0) - a_x\sin(\theta_0) & p_y\cos(\theta_0) - p_x\sin(\theta_0) \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

Expanding the left side of equation 3.19, the following is obtained:

$$A_2A_3A_4 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos(\theta_1+\theta_2) + a_2\cos(\theta_1) \\ 0 & 0 & 1 & 0 \\ -\sin(\theta_1+\theta_2) & -\cos(\theta_1+\theta_2) & 0 & d_1 - a_1\sin(\theta_1) - a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

To isolate the second component, this relationship is applied:

$$(A_2)^{-1}A_1^{-1}[RHS] \quad = \quad A_3A_4 \tag{3.23}$$

Where, expanding the left side component of equation 3.23 the following results are acquired:

$$(A_2)^{-1}A_1^{-1}[RHS] = \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ -\sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_0) & \sin(\theta_0) & 0 & 0 \\ -\sin(\theta_0) & \cos(\theta_0) & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

Variables are introduced to represent the relationships found:

$$= \begin{bmatrix} & F_1^T & \\ & F_2^T & \\ & F_3^T & \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

$$F_1 = \begin{bmatrix} \cos(\theta_1)(n_x\cos(\theta_0) + n_y\sin(\theta_0)) - n_z\sin(\theta_1) \\ \cos(\theta_1)(o_x\cos(\theta_0) + o_y\sin(\theta_0)) - o_z\sin(\theta_1) \\ \cos(\theta_1)(a_x\cos(\theta_0) + a_y\sin(\theta_0)) - a_z\sin(\theta_1) \\ \cos(\theta_1)(p_x\cos(\theta_0) + p_y\sin(\theta_0)) + \sin(\theta_1)(d_1 - p_z) \end{bmatrix} \tag{3.26}$$

$$F_2 = \begin{bmatrix} -\sin(\theta_1)(n_x\cos(\theta_0) + n_y\sin(\theta_0)) - n_z\cos(\theta_1) \\ -\sin(\theta_1)(o_x\cos(\theta_0) + o_y\sin(\theta_0)) - o_z\cos(\theta_1) \\ -a_z\cos(\theta_1) - \sin(\theta_1)(a_x\cos(\theta_0) + a_y\sin(\theta_0)) \\ \cos(\theta_1)(d_1 - p_z) - \sin(\theta_1)(p_x\cos(\theta_0) + p_y\sin(\theta_0)) \end{bmatrix} \tag{3.27}$$

$$F_3 = \begin{bmatrix} n_y\cos(\theta_0) - n_x\sin(\theta_0) \\ o_y\cos(\theta_0) - o_x\sin(\theta_0) \\ a_y\cos(\theta_0) - a_x\sin(\theta_0) \\ p_y\cos(\theta_0) - p_x\sin(\theta_0) \end{bmatrix} \tag{3.28}$$

For the right side of equation 3.23 it follows:

$$A_3A_4 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_1 + a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

Using these equation, θ_0, θ_1 and θ_2 can be solved.

3.4. Resulting Angles

Solving the equations the angle relationships are found:

$$\theta_0 = \tan^{-1}(\frac{n_y}{n_x}) = \tan^{-1}(\frac{o_y}{o_x}) = \tan^{-1}(\frac{p_y}{p_x}) = -\tan^{-1}(\frac{a_x}{a_y}) \tag{3.30}$$

$$\theta_1 = \sin^{-1}(\frac{d_1 - p_z - a_2 \sin(\theta_1 + \theta_2)}{a_1}) \tag{3.31}$$

$$\theta_2 = -\sin^{-1}(\sin(\theta_1)(n_x \cos(\theta_0) + n_y \sin(\theta_0)) + n_z \cos(\theta_1)) \tag{3.32}$$

$$= \sin^{-1}(\frac{\cos(\theta_1)(d_1 - p_z) - \sin(\theta_1)(p_x \cos(\theta_0) + p_y \sin(\theta_0))}{a_2}) \tag{3.33}$$

$$= -\theta_1 - \sin^{-1}(nz) \tag{3.34}$$

It is important to note the dependence of θ_1 and θ_2 which should be programed as constraints in a the real implementation.

4. Instantaneous Kinematics

Instantaneous kinematics is referred to the analysis of the changes over the time of the end-effector positions and orientations.

4.1. Jacobian Matrix

On the following section, the Jacobian matrix $\vec{J}$ is defined, in order to relate the linear $\vec{v}$ and angular $\vec{\omega}$ velocities, in function of the n parameters $\vec{q}$ of each joint. Said relationships are:

$$\vec{v}_f^0 = (\vec{J}_v)\vec{q'} \tag{4.1}$$

$$\vec{\omega}_f^0 = (\vec{J}_\omega)\vec{q'} \tag{4.2}$$

defining the Jacobian matrix as:

$$\begin{bmatrix} \vec{v_n}_f^0 \\ \vec{\omega_n}_f^0 \end{bmatrix} = \begin{bmatrix} \vec{J}_v \\ \vec{J}_\omega \end{bmatrix} = \vec{J} \tag{4.3}$$

where: $\vec{q'} : nx1$, $\vec{J}_v = 3xn$, $\vec{J}_\omega = 3xn$ y $\vec{J} = 6xn$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \\ j_{41} & j_{42} & j_{43} \\ j_{51} & j_{52} & j_{53} \\ j_{61} & j_{62} & j_{63} \end{bmatrix} \begin{bmatrix} \theta'_0 \\ \theta'_1 \\ \theta'_2 \end{bmatrix} \tag{4.4}$$

According to the previous equations, it's necessary to analyze the linear and angular velocity equations in function of the $\vec{q'}$ parameters, with respect of the fixed reference frame.

4.1.1. Linear Velocity

Linear velocity of the effector can be derived from the position vector $\vec{p}_f^0$, which represents the relationship with the position of the effector, with respect to the fixed reference frame:

$$\vec{v}_f^0 = \frac{d}{dt}(\vec{p}_f^0) \tag{4.5}$$

for each component:

$$\begin{bmatrix} \vec{v_x}_f^0 \\ \vec{v_y}_f^0 \\ \vec{v_z}_f^0 \end{bmatrix} = \begin{bmatrix} \frac{d}{dt}\vec{p_x}_f^0 \\ \frac{d}{dt}\vec{p_y}_f^0 \\ \frac{d}{dt}\vec{p_z}_f^0 \end{bmatrix} \tag{4.6}$$

Showing that:

$$\vec{p}_f^0 = \begin{bmatrix} \cos(\theta_0)(a_1\cos(\theta_1) + a_2\cos(\theta_1+\theta_2)) \\ \sin(\theta_0)(a_1\cos(\theta_1) + a_2\cos(\theta_1+\theta_2)) \\ -a_1\sin(\theta_1) - a_2\sin(\theta_1+\theta_2) + d_1 \end{bmatrix} \tag{4.7}$$

Where $\theta_i = \theta_i(t)$, with i:0,1,2

Deriving and grouping each component the following elements are found:

$$\vec{v'_{x}}_f^0 = j_{11}\theta'_0 + j_{12}\theta'_1 + j_{13}\theta'_2 \tag{4.8}$$

$$\vec{v'_{y}}_f^0 = j_{21}\theta'_0 + j_{22}\theta'_1 + j_{23}\theta'_2 \tag{4.9}$$

$$\vec{v'_{z}}_f^0 = j_{31}\theta'_0 + j_{32}\theta'_1 + j_{33}\theta'_2 \tag{4.10}$$

with:

$$j_{11} = \sin(\theta_0)(-a_2\cos(\theta_1+\theta_2) - a_1\cos(\theta_1)) \tag{4.11}$$

$$j_{12} = -\cos(\theta_0)(a_2 \sin(\theta_1 + \theta_2) + a_1 \sin(\theta_1)) \tag{4.12}$$

$$j_{13} = -a_2 \cos(\theta_0) \sin(\theta_1 + \theta_2) \tag{4.13}$$

$$j_{21} = \cos(\theta_0)(a_2 \cos(\theta_1 + \theta_2) + a_1 \cos(\theta_1)) \tag{4.14}$$

$$j_{22} = -\sin(\theta_0)(a_2 \sin(\theta_1 + \theta_2) + a_1 \sin(\theta_1)) \tag{4.15}$$

$$j_{23} = -a_2 \sin(\theta_0) \sin(\theta_1 + \theta_2) \tag{4.16}$$

$$j_{31} = 0 \tag{4.17}$$

$$j_{32} = -a_1 \cos(\theta_1) - a_2 \cos(\theta_1 + \theta_2) \tag{4.18}$$

$$j_{33} = -a_2 \cos(\theta_1 + \theta_2) \tag{4.19}$$

Being j_{ij}, with $i = 1, 2, 3$ and $j = 1, 2, 3$ the Jacobian matrix components.

4.1.2. Angular Velocity

To determine the angular velocity ratios, the definition of the asymmetric rotation matrix and the symmetric screw matrix $\vec{S}$ are used:

$$\vec{S}(\omega_3^0) = (\vec{R_3^0})' + (\vec{R_3^0})^T$$

$$(\vec{R_3^0})' = \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix} \tag{4.20}$$

$$W_1 = \begin{bmatrix} \theta_0'(-\sin(\theta_0)\cos(\theta_1+\theta_2)) + \theta_1'(-\cos(\theta_0)\sin(\theta_1+\theta_2)) + \theta_2'(-\cos(\theta_0)\sin(\theta_1+\theta_2)) \\ \theta_0'(\cos(\theta_0)\cos(\theta_1+\theta_2)) + \theta_1'(-\sin(\theta_0)\sin(\theta_1+\theta_2)) + \theta_2'(-\sin(\theta_0)\sin(\theta_1+\theta_2)) \\ \theta_1'(-\cos(\theta_1+\theta_2)) + \theta_2'(-\cos(\theta_1+\theta_2)) \end{bmatrix}$$

(4.21)

$$W_2 = \begin{bmatrix} \theta_0'(\sin(\theta_0)\sin(\theta_1+\theta_2)) + \theta_1'(-\cos(\theta_0)\cos(\theta_1+\theta_2)) + \theta_2'(-\cos(\theta_0)\cos(\theta_1+\theta_2)) \\ \theta_0'(-\cos(\theta_0)\sin(\theta_1+\theta_2)) + \theta_1'(-\sin(\theta_0)\cos(\theta_1+\theta_2)) + \theta_2'(-\sin(\theta_0)\cos(\theta_1+\theta_2)) \\ \theta_1'(\sin(\theta_1+\theta_2)) + \theta_2'(\sin((\theta_1+\theta_2)) \end{bmatrix}$$

(4.22)

$$W_3 = \begin{bmatrix} \theta_0'(-\cos(\theta_0)) \\ \theta_0'(-\sin(\theta_0)) \\ 0 \end{bmatrix} \tag{4.23}$$

$$(\vec{R_3^0})' + (\vec{R_3^0})^T = \begin{bmatrix} 0 & -\theta_0' & (\theta_1'+\theta_2')\cos(\theta_0) \\ \theta_0' & 0 & (\theta_1'+\theta_2')\sin(\theta_0) \\ -(\theta_1'+\theta_2')\cos(\theta_0) & -(\theta_1'+\theta_2')\sin(\theta_0) & 0 \end{bmatrix}$$

(4.24)

$$\vec{S}(\omega_3^0) = \begin{bmatrix} 0 & -(\omega_3^0)_z & (\omega_3^0)_y \\ (\omega_3^0)_z & 0 & -(\omega_3^0)_x \\ -(\omega_3^0)_y & (\omega_3^0)_x & 0 \end{bmatrix} \tag{4.25}$$

Solving as:

$$\vec{S}(\omega_3^0) = \begin{bmatrix} (\omega_3^0)_x \\ (\omega_3^0)_y \\ (\omega_3^0)_z \end{bmatrix} = \begin{bmatrix} 0 & -\sin(\theta_0) & -\sin(\theta_0) \\ 0 & \cos(\theta_0) & \cos(\theta_0) \\ 1 & 0 & 0 \end{bmatrix} \tag{4.26}$$

$$j_{41} = 0 \tag{4.27}$$

$$j_{42} = -\sin(\theta_0) \tag{4.28}$$

$$j_{43} = -\sin(\theta_0) \tag{4.29}$$

$$j_{51} = 0 \tag{4.30}$$

$$j_{52} = \cos(\theta_0) \tag{4.31}$$

$$j_{53} = \cos(\theta_0) \tag{4.32}$$

$$j_{61} = 1 \tag{4.33}$$

$$j_{62} = 0 \tag{4.34}$$

$$j_{63} = 0 \tag{4.35}$$

Being j_{ij}, with $i = 1, 2, 3$ and $j = 1, 2, 3$ the Jacobian matrix components.

The resulting Jacobian matrix J is:

$$\begin{bmatrix} (-a_1 Cos[\theta_1] - a_2 Cos[\theta_1 + \theta_2]) Sin[\theta_0] & -Cos[\theta_0](a_1 Sin[\theta_1] + a_2 Sin[\theta_1 + \theta_2]) & -a_2 Cos[\theta_0] Sin[\theta_1 + \theta_2] \\ Cos[\theta_0](a1 Cos[\theta_1] + a2 Cos[\theta_1 + \theta_2]) & -Sin[\theta_0](a1 Sin[\theta_1] + a2 Sin[\theta_1 + \theta_2]) & -a2 Sin[\theta_0] Sin[\theta_1 + \theta_2] \\ 0 & -a1 Cos[\theta_1] - a2 Cos[\theta_1 + \theta_2] & -a2 Cos[\theta_1 + \theta_2] \\ 0 & -Sin[\theta_0] & -Sin[\theta_0] \\ 0 & Cos[\theta_0] & Cos[\theta_0] \\ 1 & 0 & 0 \end{bmatrix} \tag{4.36}$$

This Jacobian matrix will be used for singularities analysis.

4.2. Inverse Jacobian Matrix

This matrix is obtained from the pseudo-inverse of the Jacobian matrix, as the resulting Jacobian is not a square matrix. This was calculated with the use of the Mathematica software, and its definition is too long to include in this report. The Jacobian inverse matrix relates to the final effector (linear and angular) velocities, with the velocities of the joints.

$$\theta' = \vec{J^{-1}} \begin{bmatrix} \vec{v_{nf}}^0 \\ \vec{\omega_{nf}}^0 \end{bmatrix} \tag{4.37}$$

$$\begin{bmatrix} \theta'_0 \\ \theta'_1 \\ \theta'_2 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} & q_{15} & q_{16} \\ q_{21} & q_{22} & q_{23} & q_{24} & q_{25} & q_{26} \\ q_{31} & q_{32} & q_{33} & q_{34} & q_{35} & q_{36} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \tag{4.38}$$

By using the pseudo-inverse matrix definition, the inverse Jacobian was founded. Mathematic software allow this calculation, but the result can not be reported given its great dimension of equations.

5. Simulations

The validation of the models was done using the Solid-Works movement study tool. For this purpose a model was created in Solid-Works software. The final effector path is analyzed with respect to the global coordinate in the rotation axis of the base motor. Figure 5.1 shows the trajectory of the final effector that will be used for the validation of the models.

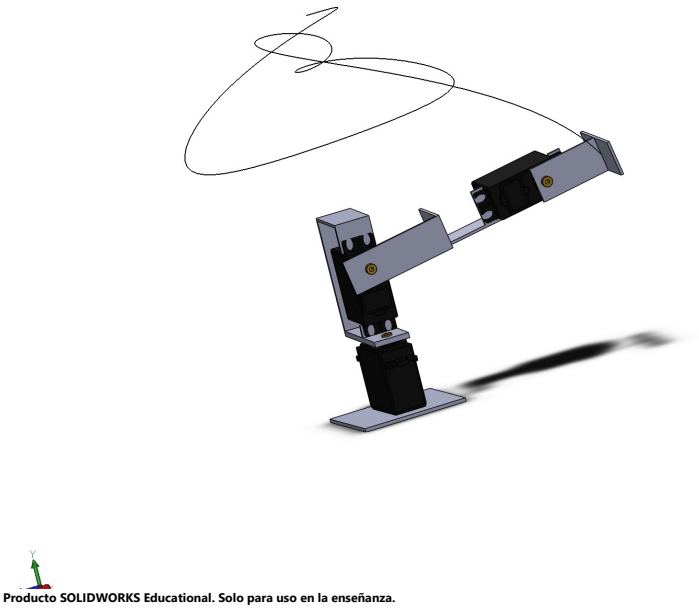

Figure 5.1.: Kinematics Equations Validation

5.1. Forward Kinematics Model Validation

Figure 5.2 shows the local angular movements of the angles of the rotation axes of each of the motors.

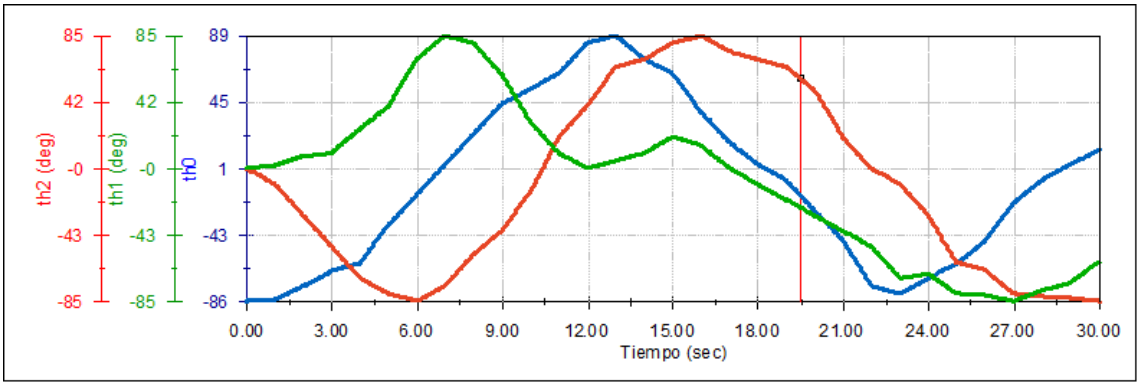

Figure 5.2.: Local coordinates rotation angles

Figure 5.3 shows the comparison between the positions obtained by the model and those captured by the Solid Works simulation. The models have been correctly adjusted.

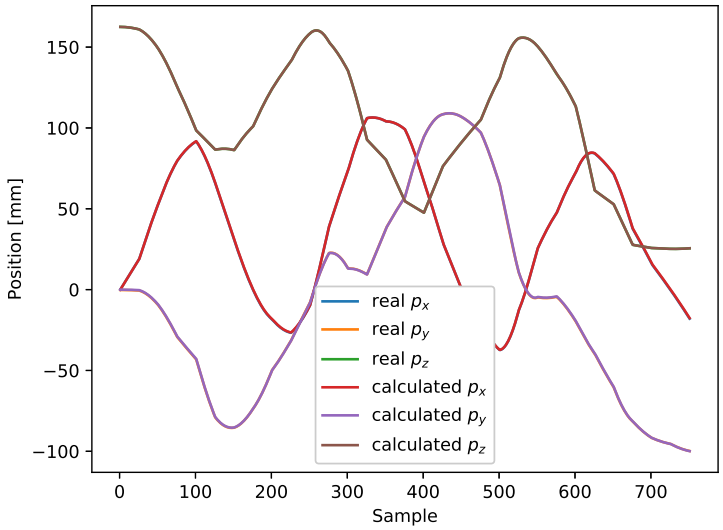

Figure 5.3.: Final effector position: real vs. calculated

Figure 5.4 shows the validation error. The observed maximum error is $0.30mm$.

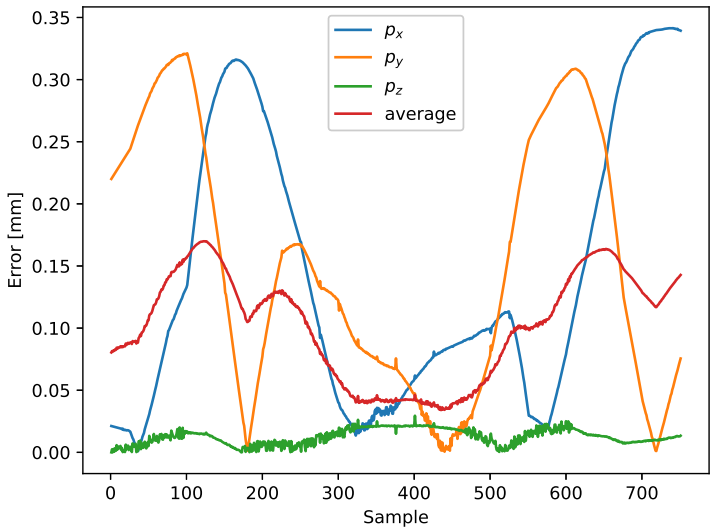

Figure 5.4.: Final effector calculation error

5.2. Inverse Kinematics Model Validation

Inverse kinematics has been probed to validate the developed relations between the angles as a function of the end-effector position and orientation. Figure 5.5 shows the comparison of the measured data in simulation against those found in the equations.

Figure 5.6 shows the error found for the equation and simulation comparison. Note that the the maximum error was less than 0.020 radians, or 1.14 degrees.

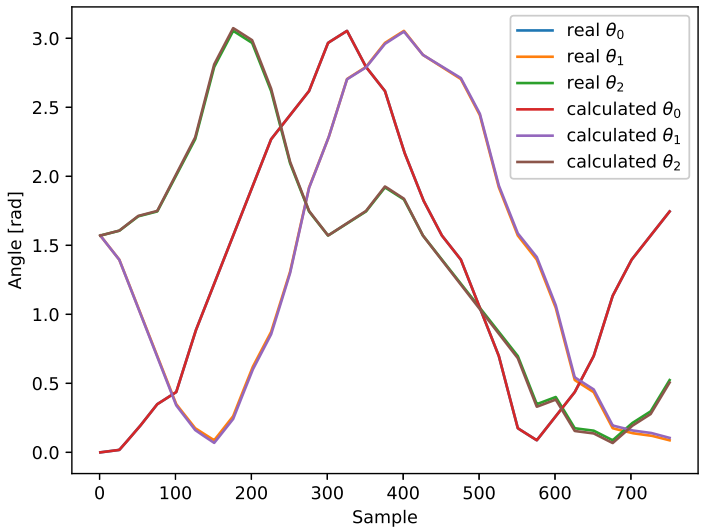

Figure 5.5.: Servomotor Angles: real vs. calculated

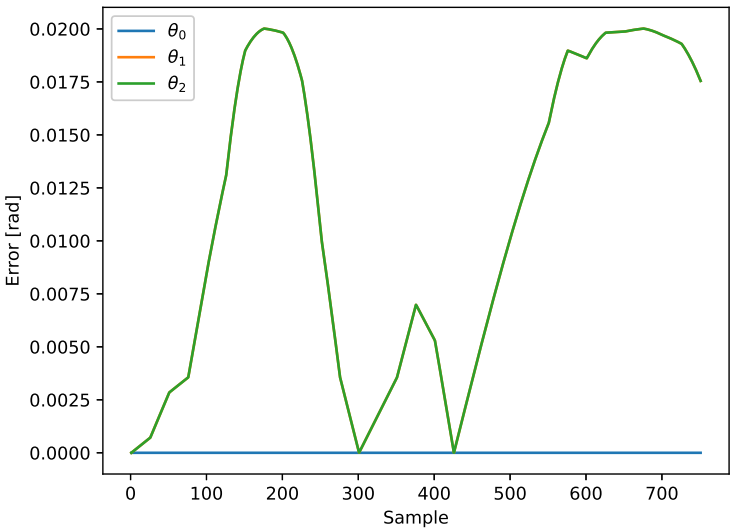

Figure 5.6.: Inverse kinematics angle calculation error

5.3. Instantaneous Forward Kinematics Model Validation

Figure 5.7 shows the comparisons between the simulated and calculated linear velocities values .

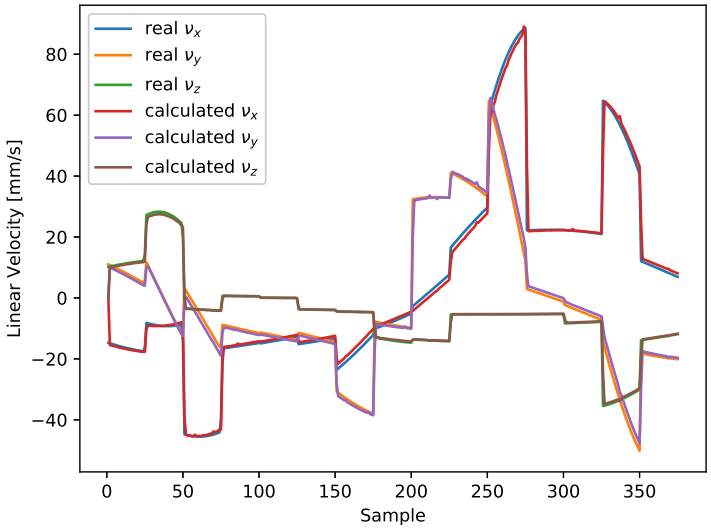

Figure 5.7.: Final effector linear velocity: real vs. calculated

Figure 5.8 shows the error values of simulated and calculated linear velocities comparisons.

Figure 5.9 shows the comparisons between the simulated and calculated angular velocities values.

Figure 5.10 shows the error values of simulated and calculated angular velocities comparisons.

According to the validation results, the errors are very low, which means that the mathematical models are correctly defined.

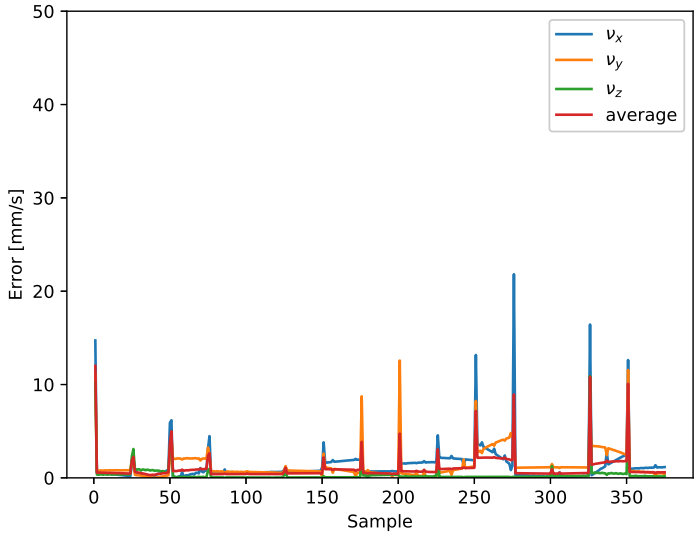

Figure 5.8.: Final effector linear velocities calculation error

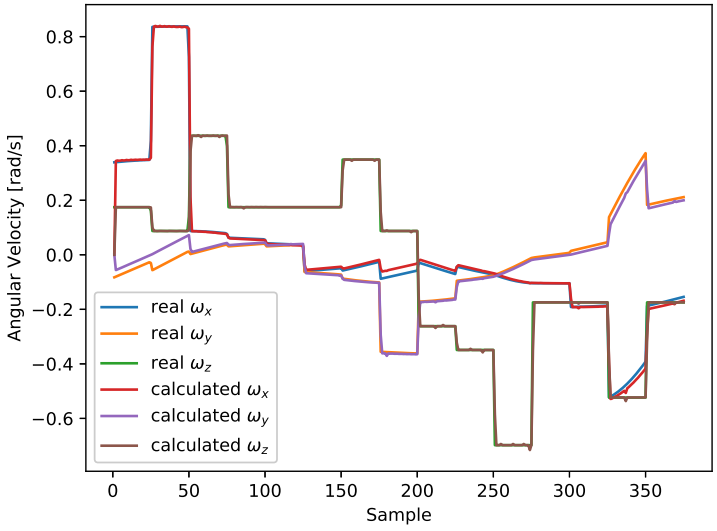

Figure 5.9.: Final effector angular velocity: real vs. calculated

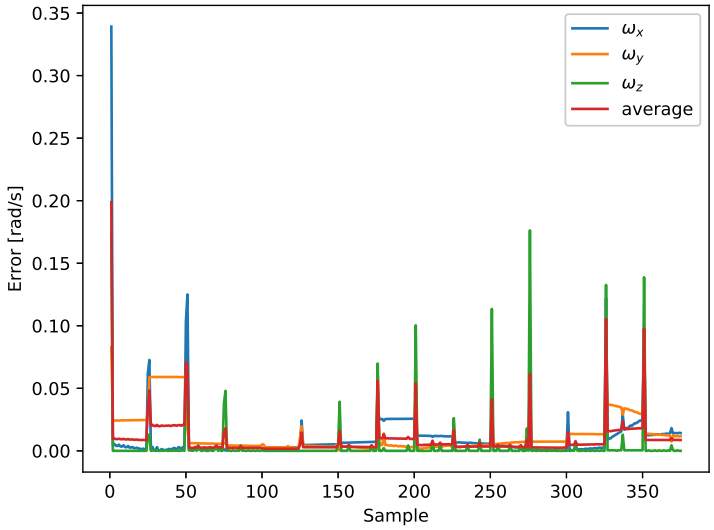

Figure 5.10.: Final effector angular velocity calculation error

6. Screw Theory Mobility Analysis

In this section, the mobility of the robot arm is analyzed through: 1) Grübler-Kutzbach Degrees of Freedom (DOF) and 2) Screw Theory.

6.1. Grübler-Kutzbach DOF

According to the rule proposed by Grübler-Kutzbach, the DOF, or mobility, of a mechanism is calculated using this equation:

$$DOF = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i = 6 * (4 - 3 - 1) + 1 + 1 + 1 = 3 \tag{6.1}$$

Being λ the work-space dimension, n the number of bodies (or links), g the number of joints and f_i the degree freedom of each joint. It is important to note that n includes the fixed link. For our mechanism, equation 6.1 shows the mobility estimated under the Grübler-Kutzbach criteria.

6.2. Screw Theory

Methodologies presented in [2], [3] [4] and [5] have been taken into consideration for the analysis of the mechanism through Screw Theory.

Figure 6.1 presents the schematics of the arm, and the definition of the screws.

The equations for the defined screws are:

$$\$_1 = \begin{bmatrix} \vec{s_1} \\ 0 \end{bmatrix} \quad \$_2 = \begin{bmatrix} \vec{s_2} \\ \vec{d_1} x \vec{s_2} \end{bmatrix} \quad \$_3 = \begin{bmatrix} \vec{s_3} \\ (d_1 \vec{+} a_1) x \vec{s_3} \end{bmatrix} \tag{6.2}$$

where:

$$\vec{s_1} = [0 \quad 0 \quad 1] \quad \vec{s_2} = [-\sin(\theta_0) \;\; \cos(\theta_0) \quad 0] \quad \vec{s_3} = [-\sin(\theta_0) \;\; \cos(\theta_0) \quad 0] \tag{6.3}$$

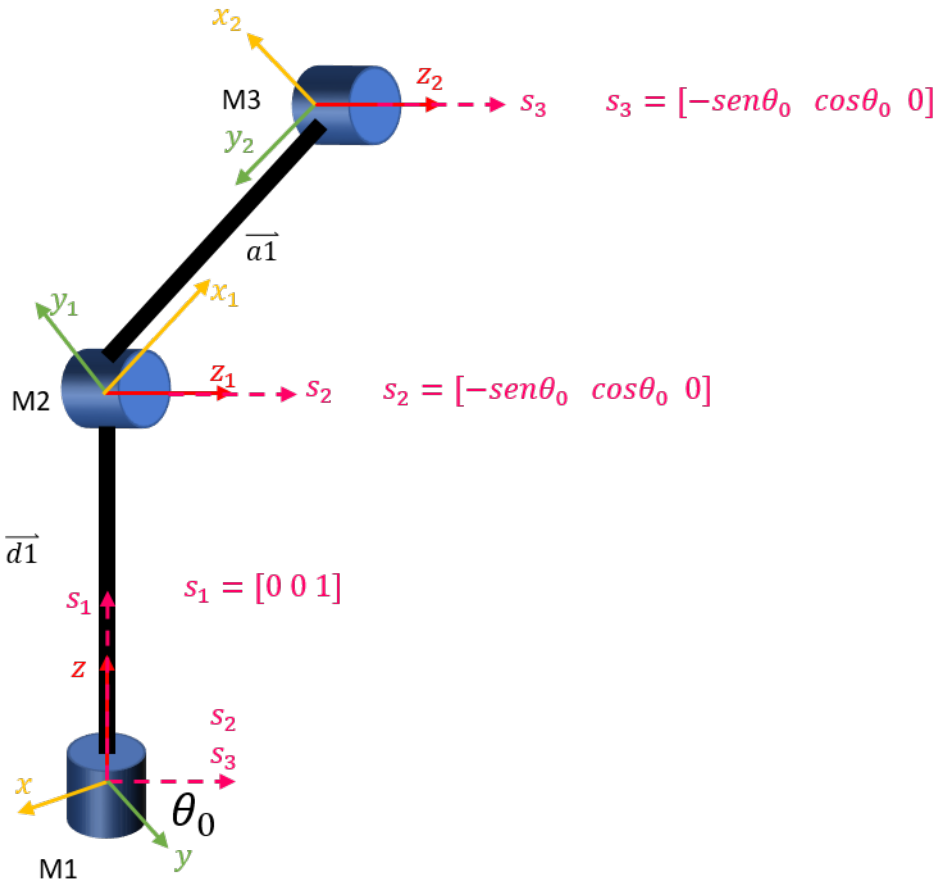

Figure 6.1.: Mobility analysis diagram

are the unit vectors of the screws.

Also:

$$\vec{d_1} = [0 \quad 0 \quad d_1] \quad \vec{a_1} = {}^{i}_{0}R *^{0}_{1} R * [a_1 \quad 0 \quad 0] \tag{6.4}$$

are the vectors representing the joints shown on figure 6.1.

$${}^{0}_{1}R = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1)) & 0 \\ 0 & 0 & 1 \\ -\sin(\theta_1) & -\cos(\theta_1) & 0 \end{bmatrix} \tag{6.5}$$

$${}^{0}_{i}R = \begin{bmatrix} \cos(\theta_0) & -\sin(\theta_0) & 0 \\ \sin(\theta_0) & \cos(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

Expanding the equations in 6.2, and using equations 6.3, 6.4, 3.9 and 3.11 the screws that represent the moving system are:

$$\$ = \begin{bmatrix} \$_1 = & [0,0,1;0,0,0] \\ \$_2 = & [-\sin(\theta_0), \cos(\theta_0), 0; -d_1\cos(\theta_0), -d_1\sin(\theta_0), 0] \\ \$_3 = & [-\sin[\theta_0], \cos(\theta_0), 0; \cos(\theta_0)(-d_1 + a_1\sin(\theta_1)), \sin(\theta_0)(-d_1 + a_1\sin(\theta_1)), a_1\cos(\theta_1)] \end{bmatrix} \tag{6.7}$$

The reciprocal screws (restriction forces) calculated are:

$$\$^r = \begin{bmatrix} \$_1^r = & [-\cot(\theta_1)\sec(\theta_0), 0, 1; d_1\cot(\theta_1)\ \csc(\theta_0), 0, 0] \\ \$_2^r = & [-\tan(\theta_0), 1, 0; 0, 0, 0] \\ \$_3^r = & [0,0,0; \cot(\theta_0), 1, 0] \end{bmatrix} \tag{6.8}$$

The freedom screws are:

$$\$^f = \begin{bmatrix} \$_1^f = & [0,0,1;0,0,0] \\ \$_2^f = & [-\tan(\theta_0), 1, 0; -d_1, -d_1\tan(\theta_0), 0] \\ \$_3^f = & [0,0,0; \cos(\theta_0)\tan(\theta_1), \sin(\theta_0)\tan(\theta_1), 1] \end{bmatrix} \tag{6.9}$$

The freedom screw equations show the DOF for the arm mechanism. The screw $\$_1^f$ shows that the mechanism has a rotation DOF on the z axis of the global reference frame. The $\$_2^f$ screw indicates that the mechanism has angular motion freedom around an axis defined in the (x, y) plane and perpendicular to the plane containing the first axis. The $\$_3^f$ and $\$_3^f$ screws indicate that the mechanism can move in x, y or z directions under the constraints represented by the equations of those screws.

6.3. Singularity Analysis

The singularity analysis here presented is of type-1, as defined by [6], which is usually performed by defining a Jacobian matrix and equating the determinant to zero:

$$\det[J] = 0 \tag{6.10}$$

Here the mobility constraints generated by the singularities, were also analyzed, using the screws.

Using equation 6.10 the following type-1 singularities are determined:

$$\theta_0 = 0, \pi/2 \tag{6.11}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0 & -a_1 \sin[\theta_1] - a_2 \sin[\theta_1 + \theta_2] & -a_2 \sin[\theta_1 + \theta_2] \\ a_1 \cos[\theta_1] + a_2 \cos[\theta_1 + \theta_2] & 0 & 0 \\ 0 & -a_1 \cos[\theta_1] - a_2 \cos[\theta_1 + \theta_2] & -a_2 \cos[\theta_1 + \theta_2] \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_0' \\ \theta_1' \\ \theta_2' \end{bmatrix} \tag{6.12}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} -a_1 \cos[\theta_1] - a_2 \cos[\theta_1 + \theta_2] & 0 & 0 \\ 0 & -a_1 \sin[\theta_1] - a_2 \sin[\theta_1 + \theta_2] & -a_2 \sin[\theta_1 + \theta_2] \\ 0 & -a_1 \cos[\theta_1] - a_2 \cos[\theta_1 + \theta_2] & -a_2 \cos[\theta_1 + \theta_2] \\ 0 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_0' \\ \theta_1' \\ \theta_2' \end{bmatrix} \tag{6.13}$$

$$\theta_1 = [-\pi/2] \quad \theta_2 = [0] \tag{6.14}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0 & -(-a_1 - a_2) \cos[\theta_0] & a_2 \cos[\theta_0] \\ 0 & -(-a_1 - a_2) \sin[\theta_0] & a_2 \sin[\theta_0] \\ 0 & 0 & 0 \\ 0 & -\sin[\theta_0] & -\sin[\theta_0] \\ 0 & \cos[\theta_0] & \cos[\theta_0] \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_0' \\ \theta_1' \\ \theta_2' \end{bmatrix} \tag{6.15}$$

Singularity mobility analysis

By replacing the values of the angles obtained in the singularity analysis in the freedom screws, these values are found:

$$\theta_0 = 0, \pi/2 \tag{6.16}$$

$$\$^f = \begin{bmatrix} \$_1^f = & [0,0,1;0,0,0] \\ \$_2^f = & [0,1,0;-d_1,0,0] \\ \$_3^f = & [0,0,0;\tan(\theta_1),0,1] \end{bmatrix} \tag{6.17}$$

$$\$^f = \begin{bmatrix} \$_1^f = & [0,0,1;0,0,0] \\ \$_2^f = & [1,0,0;0,d_1,0] \\ \$_3^f = & [0,0,0;0,\tan(\theta_1),1] \end{bmatrix} \tag{6.18}$$

$$\theta_1 = [\pi/2] \quad \theta_2 = [0] \tag{6.19}$$

$$\$^f = \begin{bmatrix} \$_1^f = & [0,0,1;0,0,0] \\ \$_2^f = & [-\tan(\theta_0),1,0;0,0,0] \\ \$_3^f = & [0,0,0;\cot(\theta_0),1,0] \end{bmatrix} \tag{6.20}$$

When the arm mechanism encounters singular positions it loses DOF, as seen in the resulting screws.

7. Statics

Figure 7.1 shows the free body diagram for the analysis of the statics relationships.

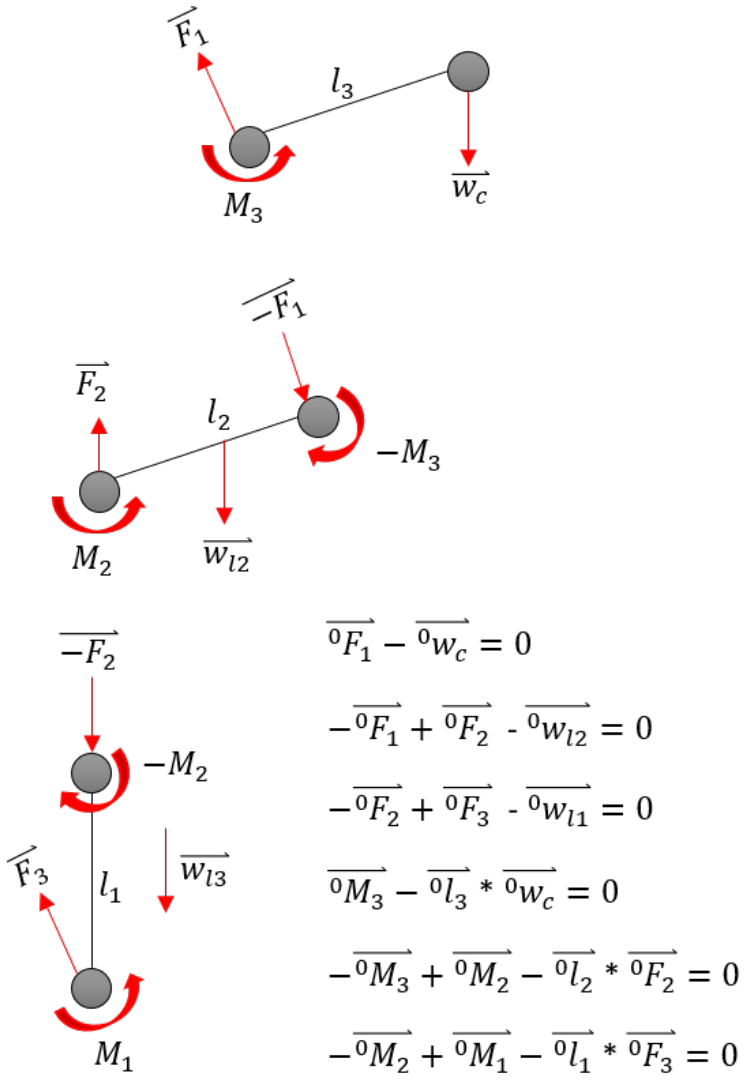

Figure 7.1.: Diagrama para el análisis de estática

The equations defining statics are given by:

Linear forces:

$$^{0}\vec{F}_{1} -^{0} \vec{w}_{c} = 0 \tag{7.1}$$

$$-^{0}\vec{F}_{1} +^{0} \vec{F}_{2} -^{0} \vec{w}_{l_2} = 0 \tag{7.2}$$

$$-^{0}\vec{F}_{2} +^{0} \vec{F}_{3} -^{0} \vec{w}_{l_1} = 0 \tag{7.3}$$

Angular forces around the origin of each local system:

$$^{0}\vec{M}_{3} -^{0} \vec{l}_{3} \times^{0} \vec{w}_{c} = 0 \tag{7.4}$$

$$-^{0}\vec{M}_{3} +^{0} \vec{M}_{2} -^{0} \vec{l}_{2} \times^{0} \vec{F}_{2} = 0 \tag{7.5}$$

$$-^{0}\vec{M}_{2} +^{0} \vec{M}_{1} -^{0} \vec{l}_{1} \times^{0} \vec{F}_{3} = 0 \tag{7.6}$$

Given the following values:

$$^{0}\vec{l_1} = [0 \quad 0 \quad l_1] \tag{7.7}$$

$$^{0}\vec{l_2} =^{i}_{0} R^{0}_{1}R \ [l_2 \quad 0 \quad 0] \tag{7.8}$$

$$^{0}\vec{l_3} =^{i}_{0} R^{0}_{1}R^{1}_{2}R \ [l_3 \quad 0 \quad 0] \tag{7.9}$$

$$^{0}\vec{w}_{c} = [0 \quad 0 \quad w_c] \tag{7.10}$$

$$^{0}\vec{w}_{2} = [0 \quad 0 \quad w_2] \tag{7.11}$$

$$^0\vec{w}_1 = [0 \quad 0 \quad w_1] \tag{7.12}$$

The static system is defined by the linear and angular force equations:

$$^0\vec{F}_1 = \begin{bmatrix} 0 \\ 0 \\ w_c \end{bmatrix}^0 \vec{F}_2 = \begin{bmatrix} 0 \\ 0 \\ w_2 + w_c \end{bmatrix}^0 \vec{F}_3 = \begin{bmatrix} 0 \\ 0 \\ w_1 + w_2 + w_c \end{bmatrix} \tag{7.13}$$

$$^0\vec{M}_3 = \begin{bmatrix} -l_3 w_c \cos[\theta_1]\cos[\theta_2]\sin[\theta_0] + l_3 w_c \sin[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ l_3 w_c \cos[\theta_0]\cos[\theta_1]\cos[\theta_2] - l_3 w_c \cos[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ 0 \end{bmatrix} \tag{7.14}$$

$$^0\vec{M}_2 = \begin{bmatrix} l_2 w_2 \cos[\theta_1]\sin[\theta_0] + l_2 w_c \cos[\theta_1]\sin[\theta_0] - l_3 w_c \cos[\theta_1]\cos[\theta_2]\sin[\theta_0] + l_3 w_c \sin[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ -l_2 w_2 \cos[\theta_0]\cos[\theta_1] - l_2 w_c \cos[\theta_0]\cos[\theta_1] + l_3 w_c \cos[\theta_0]\cos[\theta_1]\cos[\theta_2] - l_3 w_c \cos[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ 0 \end{bmatrix}$$

(7.15)

$$^0\vec{M}_1 = \begin{bmatrix} l_2 w_2 \cos[\theta_1]\sin[\theta_0] + l_2 w_c \cos[\theta_1]\sin[\theta_0] - l_3 w_c \cos[\theta_1]\cos[\theta_2]\sin[\theta_0] + l_3 w_c \sin[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ -l_2 w_2 \cos[\theta_0]\cos[\theta_1] - l_2 w_c \cos[\theta_0]\cos[\theta_1] + l_3 w_c \cos[\theta_0]\cos[\theta_1]\cos[\theta_2] - l_3 w_c \cos[\theta_0]\sin[\theta_1]\sin[\theta_2] \\ 0 \end{bmatrix}$$

(7.16)

The static linear forces, are generated, as expected with a compensation in the vertical direction (z axis) because of the effect of mass and the gravitational force.

The angular forces do not have a component on the z axis, as it is the action and compensation axis of the torques generated by the interactions between limbs.

8. Dynamics

The dynamic model allows to analyze the interactions of the forces and torques of the limbs, in order to determine the characteristics of the actuators.

Figures 8.1 and 8.2 show the free body diagrams for the arm mechanism.

The parameters used to represent the model are:

- α_i : angular acceleration of frame i with respect to frame 0
- ω_i : angular velocity of the frame i with respect to frame 0
- $a_{c,i}$: acceleration of the center of mass i
- r_{i,c_j} : distance of the center of mass j from the origin frame i.
- I_i : Inertia on the center of mass of the link i
- τ_i : Torque on frame i

Force equations derived from the free body diagrams given by Newton's 2nd Law are:

$$
\begin{aligned}
{}^3\vec{F}_3 &= m_3 a_{c,3} - m_3 g_3 && (8.1)\\
{}^2\vec{F}_2 &= {}^2_3R({}^3\vec{F}_3) + m_2 a_{c,2} - m_2 g_2 && (8.2)\\
{}^1\vec{F}_1 &= {}^1_2R({}^2\vec{F}_2) + m_1 a_{c,1} - m_1 g_1 && (8.3)
\end{aligned}
$$

Also, the torque equations are:

$$
\begin{aligned}
{}^3\tau_3 &= -{}^3\vec{F}_3 \times r_{3,c_3} + \alpha_3 + \omega_3 \times I_3\omega_3 && (8.4)\\
{}^2\tau_2 &= {}^2_3R({}^3\vec{\tau}_3) - {}^2\vec{F}_2 \times r_{2,c_2} + {}^2_3R({}^3\vec{F}_3) \times r_{3,c_2} + \alpha_2 + \omega_2 \times I_2\omega_2 && (8.5)\\
{}^1\tau_1 &= {}^2_1R({}^2\vec{\tau}_1) - {}^1\vec{F}_1 \times r_{1,c_1} + {}^2_1R({}^2\vec{F}_2) \times r_{2,c_1} + \alpha_1 + \omega_1 \times I_1\omega_1 && (8.6)
\end{aligned}
$$

Solutions to these equations were found using Mathematics software.

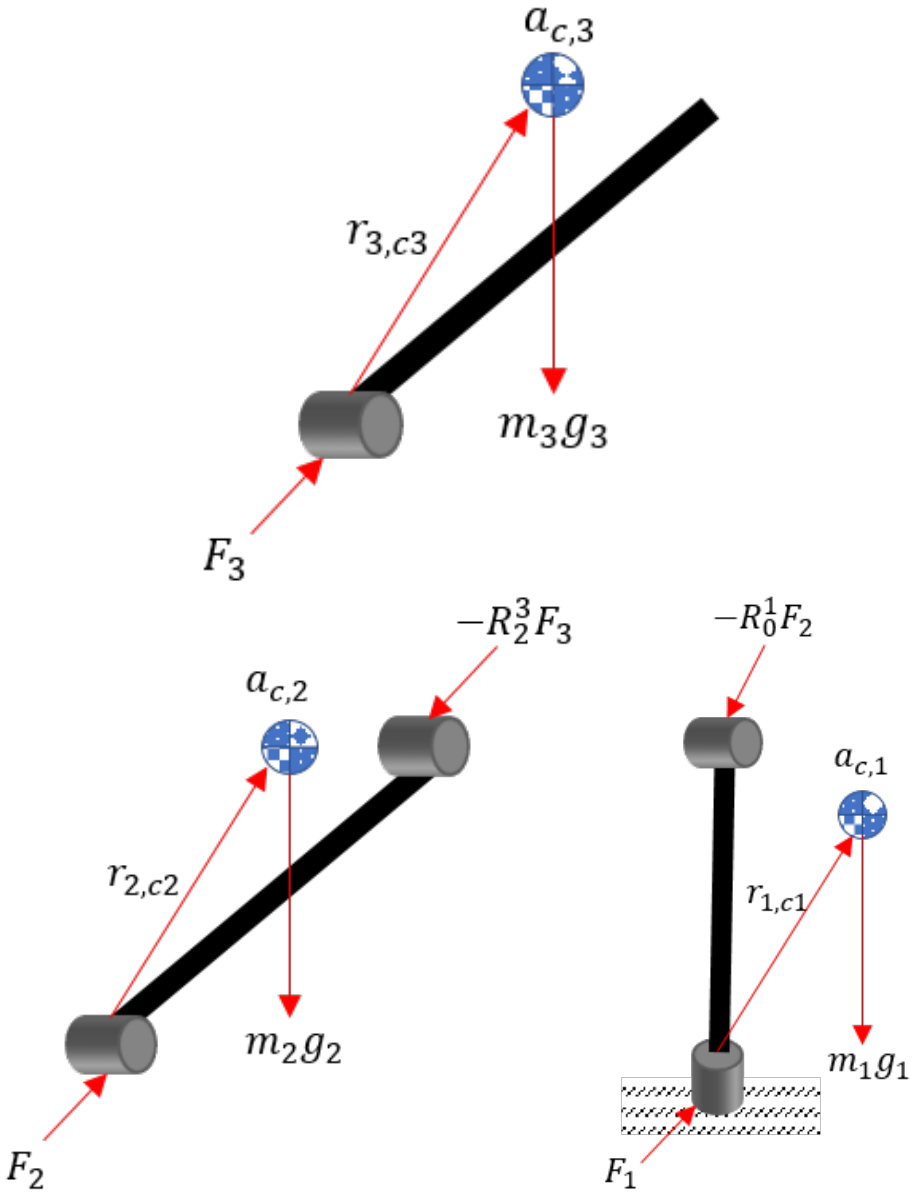

Figure 8.1.: Free body diagrams for the analysis of the forces

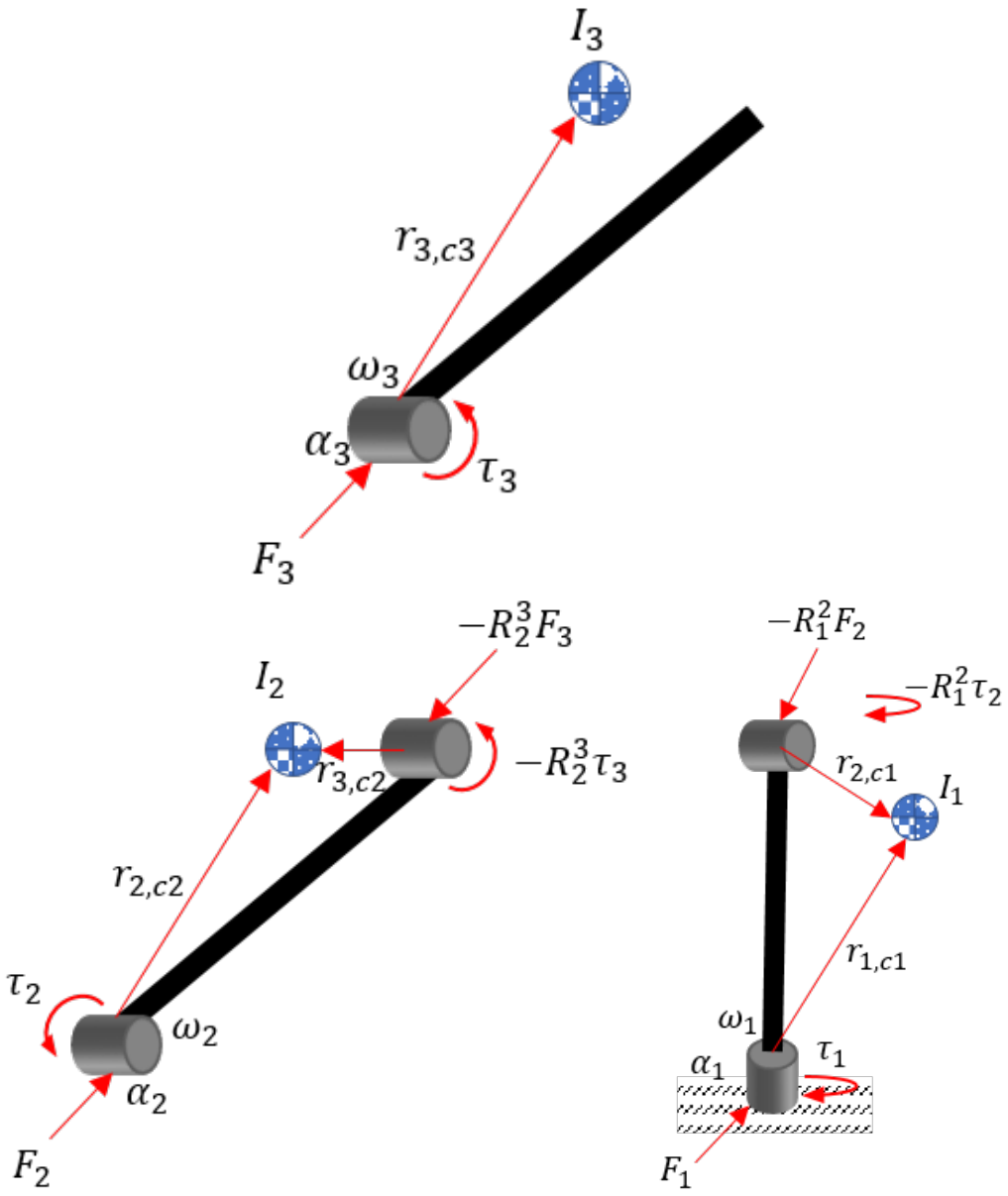

Figure 8.2.: Free body diagrams for the analysis of the torques

9. Prototype

Figure 9.1 shows the prototype used for experiments. 3 Servo-motors, 1 Arduino Uno and were used.

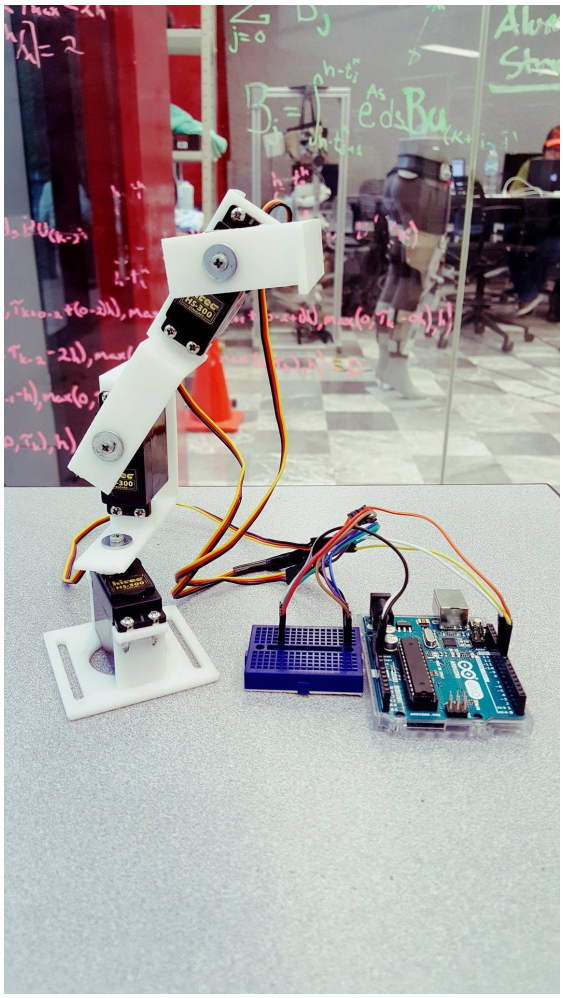

Figure 9.1.: Prototype for experiments

We annex the components of the design, including measures and the codes used for gave the functionality of the mechanism.

10. Conclusions

The forward kinematics model resulted easy to identify, from both of the equivalent approaches taken. The inverse kinematics entails performing more complex calculations, however with the help of the Mathematica software, the equations were equally easy to find.

The instantaneous kinematics analysis enabled to define the Jacobian matrix, which was then used for the singularity study of the mechanism, by the application of the defined screws. Screw theory, originally developed for the study of parallel robots, proved also useful for simplifying the mobility analysis of the robot arm.

Validation of the mathematical models through the comparison of real and simulated data, showed that the models correctly apply to the built mechanism. The relative simplicity of said mechanism proved useful in the general analysis of the concepts introduced during the study of this robotics course.

Bibliography

[1] R. N. Jazar, *Theory of applied robotics: Kinematics, dynamics, and control*. Springer Science & Business Media, 2010.

[2] H. de la Torre and E. Rodriguez-Leal, "Instantaneous kinematics analysis via screw-theory of a novel 3-c rc parallel mechanism," *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 128, 2016.

[3] J. S. Dai, Z. Huang, and H. Lipkin, "Mobility of overconstrained parallel mechanisms," *Journal of Mechanical Design*, vol. 128, no. 1, pp. 220–229, 2006.

[4] E. Cuan-Urquizo and E. Rodriguez-Leal, "Kinematic analysis of the 3-cup parallel mechanism," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, pp. 382–395, 2013.

[5] E. Rodriguez-Leal and J. S. Dai, *Evolutionary design of parallel mechanisms*, 2010.

[6] C. Gosselin and J. Angeles, "Singularity analysis of closed-loop kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 281–290, 1990.

[7] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. wiley New York, 2006, vol. 3.

A. Components

A.1. Servomotor

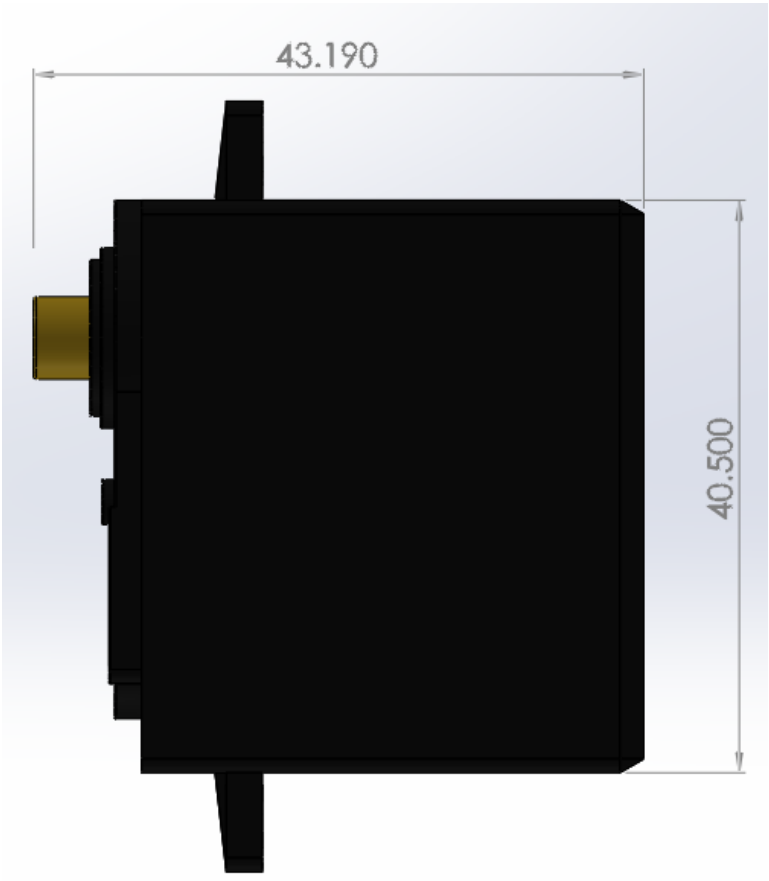

Figure A.1.: Side View

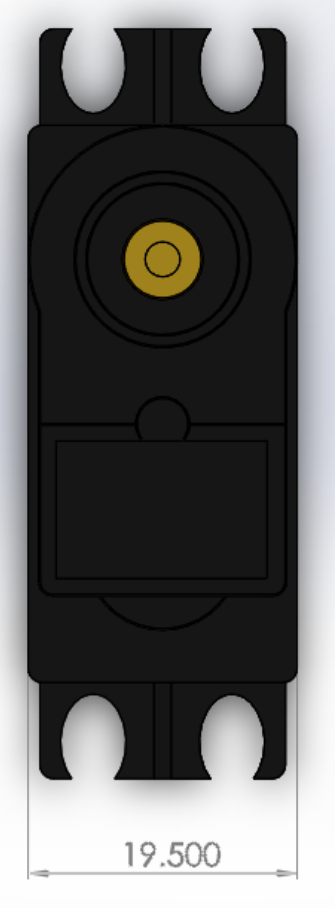

Figure A.2.: Frontal View

A.2. Support 01

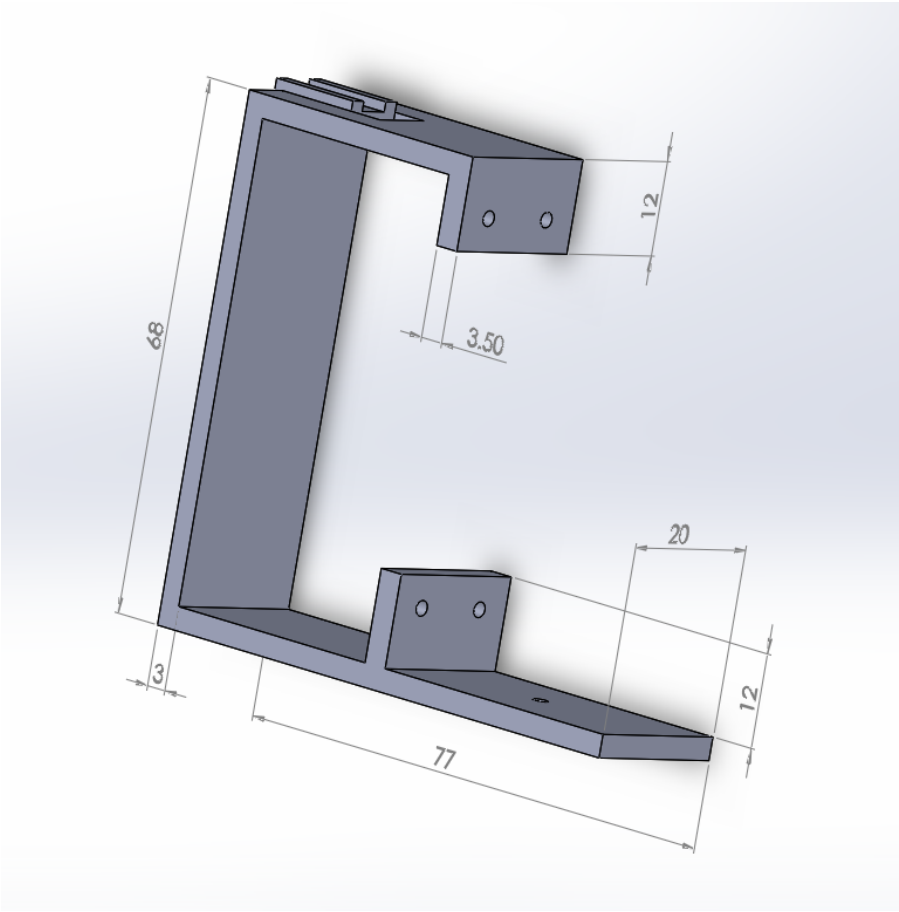

Figure A.3.: Support 01

A.3. Support 02

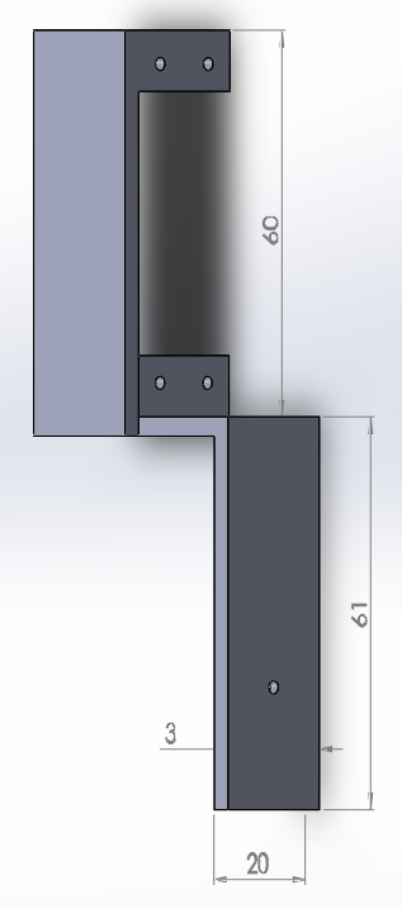

Figure A.4.: Support 02

A.4. Support 03

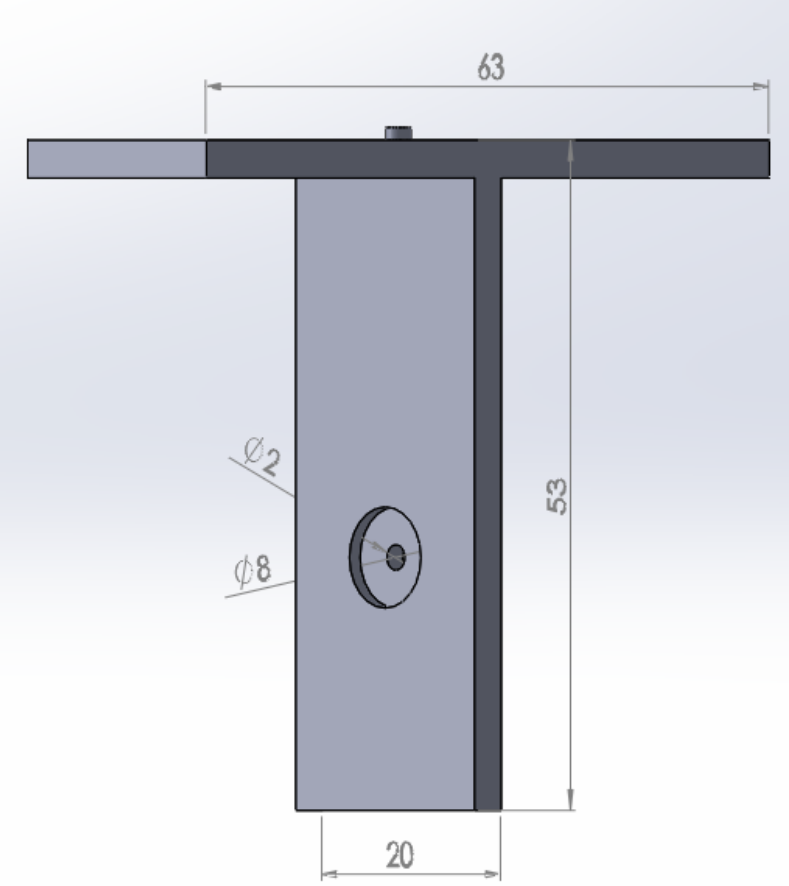

Figure A.5.: Support 03

A.5. General Assemble

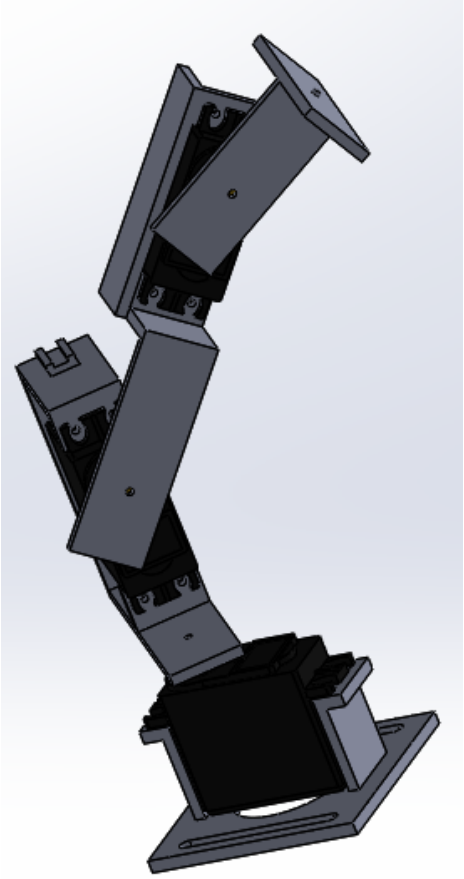

Figure A.6.: Support 03

B. Code

B.1. kine.py

Includes functions related to both inverse and direct kinematics.

```
# Find position Matrix from parameters by forward kinematics
def MatPos(th0, th1, th2, d1, a1, a2, unit='d'):
    if unit == 'd':
        th0, th1, th2 = mpmath.radians(th0), mpmath.radians(th1), mpmath.radians(th2)

    th0 *= -1
    th1 *= -1
    th2 -= pi/2
    th2 *= -1
    T = Matrix([[cos(th0)*cos(th1 + th2), -sin(th1 + th2)*cos(th0), -sin(th0), (a1*co
                [sin(th0)*cos(th1 + th2), -sin(th0)*sin(th1 + th2), cos(th0), (a1*cos
                [-sin(th1 + th2), -cos(th1 + th2), 0, -a1*sin(th1) - a2*sin(th1 + th2
                [0, 0, 0, 1]])
    return T


# Find th0 angle from desired position Matrix by inverse kinematics
def Th0(pos):
    nx, ox, ax, px = pos[0,:]
    ny, oy, ay, py = pos[1,:]

    if nx != 0:
        th = -atan(ny/nx)
    elif ox != 0:
        th = -atan(oy/ox)
    elif ax != 0:
        th = acot(ay/ax)
    elif px != 0:
        th = -atan(py/px)
    else:
        th = atan(ax/ay)

    if th < 0:
        th += pi

    return th
```

```
# Find th1 angle from desired position Matrix and th0 value by inverse
# kinematics
def Th1(pos, th0, d1, a2):
    th0 *= -1
    nx, ny, nz = pos[:3,0]
    px, py, pz = pos[:3,-1]
    th = -2.0*atan((-713.0*nx*tan(0.5*th0)**2 + 713.0*nx + 1426.0*ny*tan(0.5*th0) + 2
    return th

# Find th2 angle from desired position Matrix, th0 and th1 values by inverse
# kinematics
def Th2(pos, th0, th1, d1, a2):
    th0 *= -1
    th1 *= -1
    nx, ny, nz = pos[0:3,0]
    th = asin(sin(th1)*(nx*cos(th0)+ny*sin(th0))+nz*cos(th1))+pi/2
    return th

# Validates position Matrix derived by forward kinematics, plotting res in
# each predicted position
def validate(angles, pos, d1, a1, a2, unit='d', show=False):
    rows = len(angles[:,0])
    res = zeros(rows, 3)
    error = zeros(rows, 3)
    mats = Matrix()
    for i in range(0, rows):
        th0, th1, th2 = angles[i,:]
        mat = MatPos(th0, th1, th2, d1, a1, a2, unit)
        mats  = Matrix([mats, [mat]])
        resy = mat[:3, -1]
        err1, err2, err3 = pos[i,:] - resy.T
        error[i,:] = [[abs(err1), abs(err2), abs(err3)]]
        res[i,:] = resy.T

    xAxis = range(1, rows+1)
    avg = []
    for i in xAxis:
        ex, ey, ez = error[i-1,:]
        avg.append((ex + ey + ez)/3)

    px0, = plt.plot(xAxis, pos[:,0], label=r'real $p_x$')
    py0, = plt.plot(xAxis, pos[:,1], label=r'real $p_y$')
    pz0, = plt.plot(xAxis, pos[:,2], label=r'real $p_z$')
    px1, = plt.plot(xAxis, res[:,0], label=r'calculated $p_x$')
    py1, = plt.plot(xAxis, res[:,1], label=r'calculated $p_y$')
    pz1, = plt.plot(xAxis, res[:,2], label=r'calculated $p_z$')
    plt.legend(handles=[px0, py0, pz0, px1, py1, pz1])
    plt.xlabel("Sample")
```

```
    plt.ylabel("Position [mm]")
    plt.figure()

    [ex], [ey] = plt.plot(xAxis, error[:,0], label=r'$p_x$'), plt.plot(xAxis, error[:
    [ez], [avg] = plt.plot(xAxis, error[:,2], label=r'$p_z$'), plt.plot(xAxis, avg, l
    plt.legend(handles=[ex, ey, ez, avg])
    plt.xlabel("Sample")
    plt.ylabel("Error [mm]")

    if show: plt.show()
    return mats

# Validates rotation angles derived by inverse kinematics, plotting the error in
# each predicted angle
def valinv(mats, angles, d1, a2, unit='d'):
    rows = len(mats[:,0])
    error = zeros(rows, 3)
    res = zeros(rows, 3)
    for i in range(0, rows):
        mat, = mats[i,:]
        th0 = Th0(mat)
        th1 = Th1(mat, th0, d1, a2)
        th2 = Th2(mat, th0, th1, d1, a2)
        res[i,:] = [[th0, th1, th2]]
        a0, a1, a2 = angles[i, :]

        if unit == "d":
            a0, a1, a2 = mpmath.radians(a0), mpmath.radians(a1), mpmath.radians(a2)
            angles[i, :] = [[a0, a1, a2]]

        errow = [abs(a0 - th0), abs(a1 - th1), abs(a2 - th2)]
        error[i,:] = [errow]

    xAxis = range(1, rows + 1)

    plt.figure()
    rth0, = plt.plot(xAxis, angles[:,0], label=r'real $\theta_0$')
    rth1, = plt.plot(xAxis, angles[:,1], label=r'real $\theta_1$')
    rth2, = plt.plot(xAxis, angles[:,2], label=r'real $\theta_2$')
    cth0, = plt.plot(xAxis, res[:,0], label=r'calculated $\theta_0$')
    cth1, = plt.plot(xAxis, res[:,1], label=r'calculated $\theta_1$')
    cth2, = plt.plot(xAxis, res[:,2], label=r'calculated $\theta_2$')
    plt.legend(handles=[rth0, rth1, rth2, cth0, cth1, cth2])
    plt.xlabel("Sample")
    plt.ylabel("Angle [rad]")

    plt.figure()
    et0, = plt.plot(xAxis, error[:,0], label=r'$\theta_0$')
    et1, = plt.plot(xAxis, error[:,1], label=r'$\theta_1$')
```

```
    et2, = plt.plot(xAxis, error[:,2], label=r'$\theta_2$')
    plt.legend(handles=[et0, et1, et2])
    plt.xlabel("Sample")
    plt.ylabel("Error [rad]")
    plt.show()
```

B.2. velo.py

Includes functions related to linear and angular velocity calculations.

```
def MatJac(delta0, delta1, d1, a1, a2, unit='d'):
    if unit == 'd':
        delta0[1], delta0[2] = mpmath.radians(delta0[1]), mpmath.radians(delta0[2])
        delta0[3], delta1[1] = mpmath.radians(delta0[3]), mpmath.radians(delta1[1])
        delta1[2], delta1[3] = mpmath.radians(delta1[2]), mpmath.radians(delta1[3])

    delta0[2]-= pi/2
    delta1[2]-= pi/2
    th0, th1, th2 = delta1[1:]
    dtime, dth0, dth1, dth2 = delta1 - delta0
    ths = Matrix([dth0/dtime, dth1/dtime, dth2/dtime])
    jacob = Matrix([[sin(th0)*(-a2*cos(th1 + th2) - a1*cos(th1)), -cos(th0)*(a2*sin(t
                     -a2*cos(th0)*sin(th1 + th2)],
                    [cos(th0)*(a2*cos(th1 + th2) + a1*cos(th1)), -sin(th0)*(a2*sin(th
                     -a2*sin(th0)*sin(th1 + th2)],
                    [0, -a1*cos(th1) - a2*cos(th1 + th2), -a2*cos(th1 + th2)],
                    [0, -sin(th0), -sin(th0)],
                    [0, cos(th0), cos(th0)],
                    [1, 0, 0]])
    return jacob*ths

def validate(angles, veloc, d1, a1, a2, unit='d'):
    rows = len(angles[:,0])
    res = zeros(rows-1, 6)
    error = zeros(rows-1, 6)
    for i in range(1, rows):
        delta1 = angles[i,:]
        delta0 = angles[i-1,:]
        resy = MatJac(delta0, delta1, d1, a1, a2, unit)
        err1, err2, err3, err4, err5, err6 = veloc[i-1,:] - resy.T
        error[i-1,:] = [[abs(err1), abs(err2), abs(err3), abs(err4), abs(err5), abs(e
        res[i-1,:] = [resy.T]

    xAxis = range(1, rows)
    avgV = []
    avgW = []
    for i in xAxis:
```

```
    evx, evy, evz, ewx, ewy, ewz = error[i-1,:]
    avgV.append((evx + evy + evz)/3)
    avgW.append((ewx + ewy + ewz)/3)

vx0, = plt.plot(xAxis, veloc[:,0], label=r'real $\nu_x$')
vy0, = plt.plot(xAxis, veloc[:,1], label=r'real $\nu_y$')
vz0, = plt.plot(xAxis, veloc[:,2], label=r'real $\nu_z$')
vx1, = plt.plot(xAxis, res[:,0], label=r'calculated $\nu_x$')
vy1, = plt.plot(xAxis, res[:,1], label=r'calculated $\nu_y$')
vz1, = plt.plot(xAxis, res[:,2], label=r'calculated $\nu_z$')
plt.legend(handles=[vx0, vy0, vz0, vx1, vy1, vz1])
plt.xlabel("Sample")
plt.ylabel("Linear Velocity [mm/s]")
plt.figure()

evx, = plt.plot(xAxis, error[:,0], label=r'$\nu_x$')
evy, = plt.plot(xAxis, error[:,1], label=r'$\nu_y$')
[evz], [avgV] = plt.plot(xAxis, error[:,2], label=r'$\nu_z$'), plt.plot(xAxis, av
plt.legend(handles=[evx, evy, evz, avgV])
plt.xlabel("Sample")
plt.ylabel("Error [mm/s]")
plt.ylim(0, 50)
plt.figure()

wx0, = plt.plot(xAxis, veloc[:,3], label=r'real $\omega_x$')
wy0, = plt.plot(xAxis, veloc[:,4], label=r'real $\omega_y$')
wz0, = plt.plot(xAxis, veloc[:,5], label=r'real $\omega_z$')
wx1, = plt.plot(xAxis, res[:,3], label=r'calculated $\omega_x$')
wy1, = plt.plot(xAxis, res[:,4], label=r'calculated $\omega_y$')
wz1, = plt.plot(xAxis, res[:,5], label=r'calculated $\omega_z$')
plt.legend(handles=[wx0, wy0, wz0, wx1, wy1, wz1])
plt.xlabel("Sample")
plt.ylabel("Angular Velocity [rad/s]")
plt.figure()

ewx, = plt.plot(xAxis, error[:,3], label=r'$\omega_x$')
ewy, = plt.plot(xAxis, error[:,4], label=r'$\omega_y$')
ewz, = plt.plot(xAxis, error[:,5], label=r'$\omega_z$')
avgW, = plt.plot(xAxis, avgW, label='average')
plt.legend(handles=[ewx, ewy, ewz, avgW])
plt.xlabel("Sample")
plt.ylabel("Error [rad/s]")
plt.show()

return res
```

B.3. duino.py

Gives a simple control interface for the robot via arduino.

```
# Initialize arduino
def start():
    if sys.platform == 'darwin':
        usbport = '/dev/cu.usbmodem1421'
    else:
        usbport = 'com3'
    return serial.Serial(usbport, 9600, timeout=1)

# When finalized moving, set to default position
def stop(robot):
    move(robot, '90', '90', '90')

# Move robot via arduino
def move(robot, th0, th1, th2):
    time.sleep(2)
    angles = " ".join([th0, th1, th2])
    angles = bytearray(bytes(angles, "ascii"))
    robot.write(angles)
```