



**TÉCNICO**  
LISBOA

# ROBOTICS OF MANIPULATION

## MASTERS IN BIOMEDICAL AND ELECTRICAL ENGINEERING

---

### Assignment #1 - Kinematics [EN]

---

#### Authors:

Afonso Araújo (96138)  
Tiago Freitas (94186)

[afonso.d.araujo@tecnico.ulisboa.pt](mailto:afonso.d.araujo@tecnico.ulisboa.pt)  
[tiagogoncalvesfreitas@tecnico.ulisboa.pt](mailto:tiagogoncalvesfreitas@tecnico.ulisboa.pt)

**Group 5**

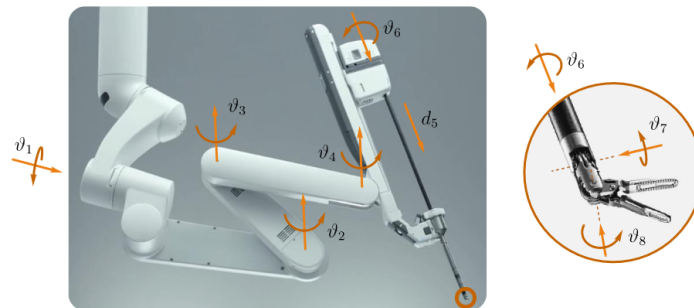
2022/2023 – 2<sup>nd</sup> Semester, P4

# Contents

<b>1</b>	<b>Question 1</b>	<b>2</b>
<b>2</b>	<b>Question 2</b>	<b>3</b>
<b>3</b>	<b>Question 3</b>	<b>9</b>
3.1	Arm Solution . . . . .	11
3.2	Wrist Solution . . . . .	13
3.3	Joint Offsets . . . . .	13
3.4	Model Validation . . . . .	14
<b>4</b>	<b>Question 4</b>	<b>15</b>
4.1	Validation through Numerical Differentiation . . . . .	16
4.2	Kinematic Singularities . . . . .	17
<b>5</b>	<b>Question 5</b>	<b>19</b>
5.1	Position Control . . . . .	19
5.2	Orientation Control . . . . .	20
5.3	CLIK Controller Architecture . . . . .	20
5.4	Orientation Error . . . . .	21
5.5	Validation through Comparison with One-Shot Inverse Kinematics and Error Analysis . . . . .	23
<b>6</b>	<b>References</b>	<b>26</b>

# 1 Question 1

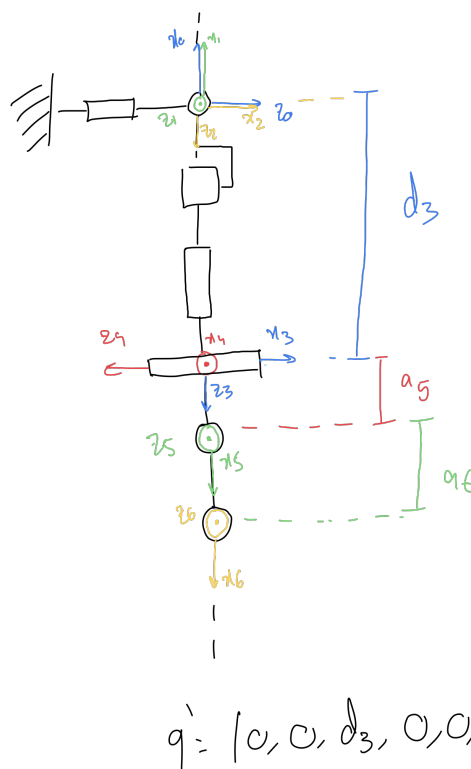
Below is the Da Vinci Xi Surgical Robot model:



**Figure 1:** Da Vinci Xi Technical Drawing, with rotation axis

In order to succeed in creating a Direct and Inverse Kinematics of the model, the Denavit-Hartenberg conditions must first be stabilised.

As such it is important to define the cartesian coordinates for the 6 cartesian referentials:



$$q = (0, 0, d_3, 0, 0, 0)$$

**Figure 2:** Denavit Hartenberg Robot Model Home 2D Referentials

that will then yield the following Denavit-Hartenberg table (the offset for  $Link_3$  is 0 however for representation purposes the drawing above is for a general  $d_3$ ).

$Link_i$	d	$\theta$	a	$\alpha$	reference'
1	0	$\theta_1$	0	$-\frac{\pi}{2}$	0
2	0	$\theta_2$	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$
3	$d_3$	0	0	0	0
4	0	$\theta_4$	0	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
5	0	$\theta_5$	$a_5$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
6	0	$\theta_6$	$a_6$	0	0

which gives the following homogenous matrix,

$$A_i^{i-1}(q_i) = \begin{pmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_ic_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_is_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

## 2 Question 2

Following the Denavit-Hartenberg matrix created before, 1, the Direct Kinematics model is given by:

$$T_n^0(q) = A_1^0 \cdot A_2^1 \cdot \dots \cdot A_n^{n-1} \quad (2)$$

which means,

$$T_6^0 = A_1^0 \cdot A_2^1 \cdot A_3^2 \cdot A_4^3 \cdot A_5^4 \cdot A_6^5 \quad (3)$$

Taking,

$$A_1^0 = \begin{pmatrix} c_{\theta_1} & 0 & -s_{\theta_1} & 0 \\ s_{\theta_1} & 0 & c_{\theta_1} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_2^1 = \begin{pmatrix} c_{\theta_2} & 0 & s_{\theta_2} & 0 \\ s_{\theta_2} & 0 & -c_{\theta_2} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

$$A_3^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_4^3 = \begin{pmatrix} c_{\theta_4} & 0 & -s_{\theta_4} & 0 \\ s_{\theta_4} & 0 & c_{\theta_4} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5)$$

$$A_5^4 = \begin{pmatrix} c_{\theta_5} & 0 & -s_{\theta_5} & a_5c_{\theta_5} \\ s_{\theta_5} & 0 & c_{\theta_5} & a_5s_{\theta_5} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_6^5 = \begin{pmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & a_6c_{\theta_6} \\ s_{\theta_6} & c_{\theta_6} & 0 & a_6s_{\theta_6} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6)$$

it is possible to compute  $T_6^0$ .

From  $T_6^0$  the only thing necessary are the the rotation,  $R_6^0$ , and position,  $p_6^0$ , matrices. Since these are complex to calculate from an analytical standpoint, a graphical approach will be taken, to validate the model.

Three different configurations will be considered, throughout this report, such that:

Configuration#	$\theta_1$	$\theta_2$	$d_3$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	0	0.3	0	0	0
2	0	45	0.5	0	0	90
3	-90	0	1	90	0	0

Configuration 1 can be seen at 2. For this configuration the calculation of both the rotation,  $R_6^0$ , and position,  $p_6^0$ , matrices is rather trivial.

Both matrices are defined such that,

$$R_6^0 = [x_6^0, y_6^0, z_6^0] \quad (7)$$

$$p_6^0 = [\hat{i}_6^0, \hat{j}_6^0, \hat{k}_6^0] \quad (8)$$

Going back to 2, it is possible to see that:

$$z_6 // y_0 \quad (9)$$

$$x_6 // -x_0 \quad (10)$$

$$y_6 // z_0 \quad (11)$$

$$(12)$$

Moreover, the only shift in position from the origin of referential 0,  $O_0$ , and the origin of referential 6,  $O_6$ , occurs along the  $x_0$  axis, collinearly to  $-\vec{x}_0$ .

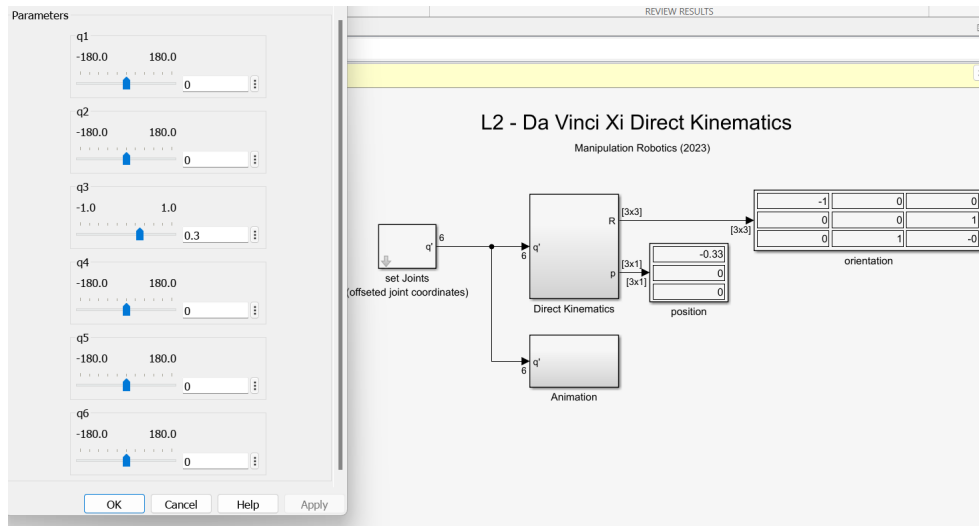
Hence:

$$R_6^0 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (13)$$

$$p_6^0 = [-d_3 - a_5 - a_6 \quad 0 \quad 0] \quad (14)$$

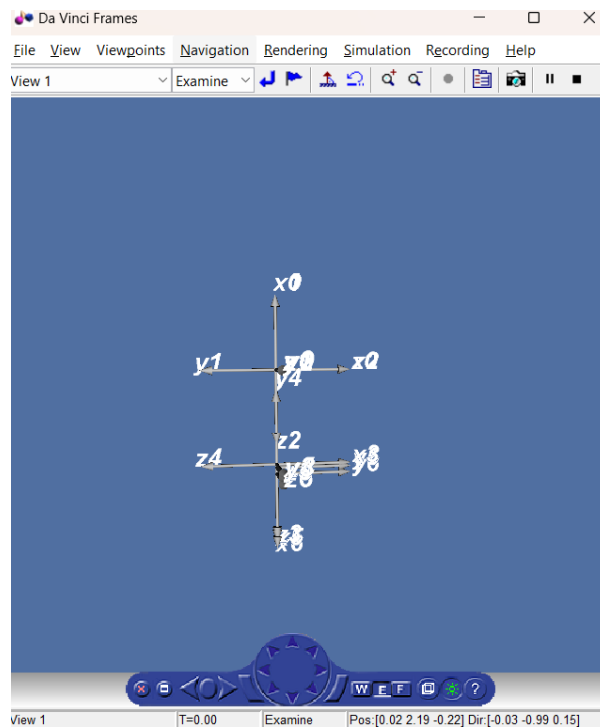
$$p_6^0 = [-0.33 \quad 0 \quad 0] \quad (15)$$

This is confirmed via simulation:



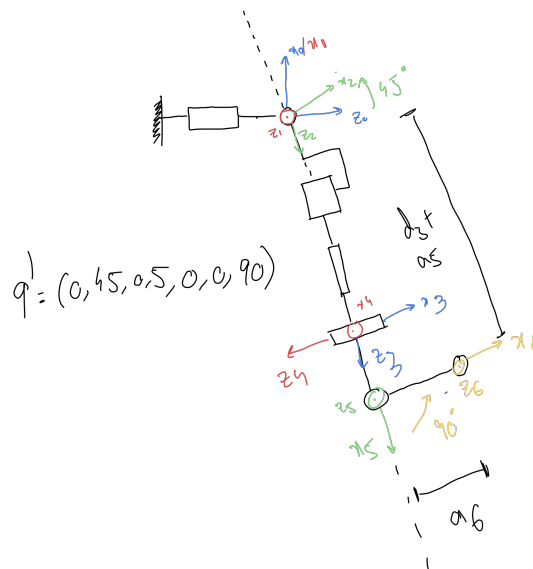
**Figure 3:** Rotation and Position Simulated Matrices for Configuration#1

Below is the referential's positioning for Configuration#1, validating 2,



**Figure 4:** Simulated DaVinci's Referentials for Configuration#1

Configuration#2 can be seen below,:



**Figure 5:** Configuration #2 for the DaVinci Robot Model

With this configuration the calculation of the rotation and position matrices is more difficult. Beginning with the  $R_6^0$  matrix, it is possible to see that:

$$x_6 // x_2 \quad (16)$$

$$z_6 // y_0 \quad (17)$$

Hence there is the need to calculate  $x_2$  and  $y_6$ . Through trigonometric equations it is obtained:

$$x_2^0 = \begin{bmatrix} \sin(45) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \cos(45) \end{bmatrix} \cdot \begin{bmatrix} x_2^2 \\ y_2^2 \\ z_2^2 \end{bmatrix} \quad (18)$$

as well as,

$$y_6^0 = \begin{bmatrix} \sin(45) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\cos(45) \end{bmatrix} \cdot \begin{bmatrix} x_6^6 \\ y_6^6 \\ z_6^6 \end{bmatrix} \quad (19)$$

This results in:

$$R_6^0 = \begin{bmatrix} \sin(45) & \sin(45) & 0 \\ 0 & 0 & 1 \\ \cos(45) & -\cos(45) & 0 \end{bmatrix} \quad (20)$$

or,

$$R_6^0 = \begin{bmatrix} 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \\ 0.7071 & -0.7071 & 0 \end{bmatrix} \quad (21)$$

Following up with the position matrix, the calculation can be divided into two stages. The first one is the calculation from  $O_0$  to  $O_5$ ,

$$p_5^0 = \begin{bmatrix} -(d_3 + a_5)\sin(45) & 0 & (d_3 + a_5)\cos(45) \end{bmatrix} \quad (22)$$

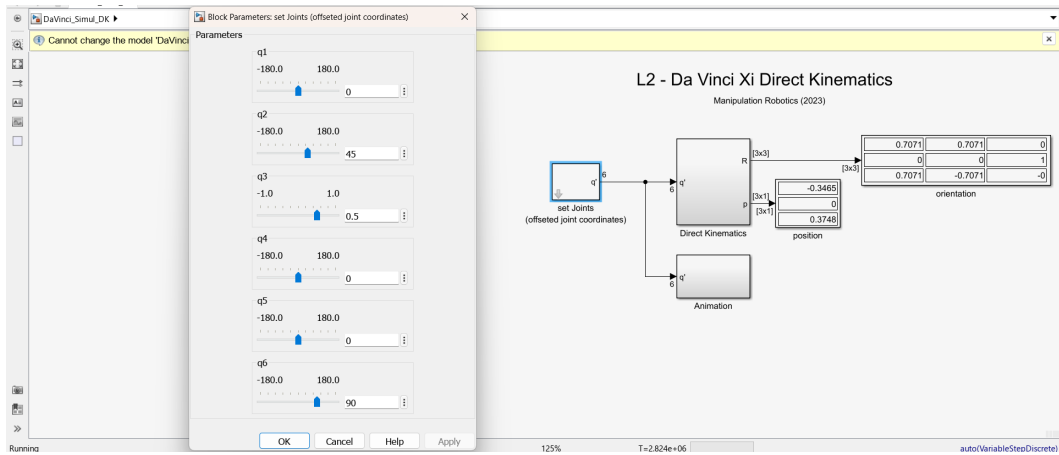
and the second one finishes with the shift from  $O_5$  to  $O_6$  yielding,

$$p_6^0 = \begin{bmatrix} -(d_3 + a_5)\sin(45) + a_6\sin(45) & 0 & (d_3 + a_5)\cos(45) + a_6\cos(45) \end{bmatrix} \quad (23)$$

or,

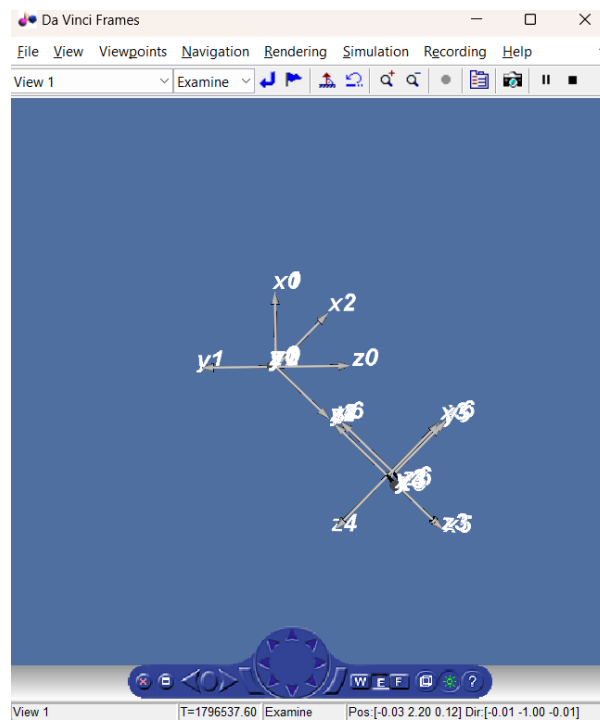
$$p_6^0 = \begin{bmatrix} -0.346482 & 0 & 0.374767 \end{bmatrix} \quad (24)$$

This is confirmed via simulation:



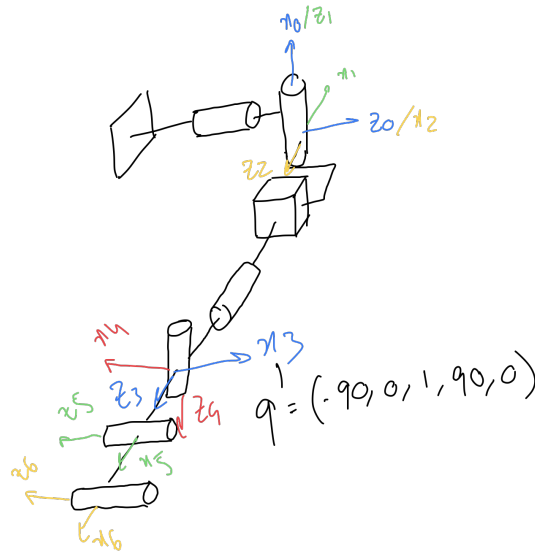
**Figure 6:** Rotation and Position Simulated Matrices for Configuration#2

Below is the referential's positioning for Configuration#2, validating 5.



**Figure 7:** Simulated DaVinci's Referentials for Configuration#2

Configuration#3 can be seen below:



**Figure 8:** Configuration #3 for the DaVinci Robot Model

The purpose behind this configuration was to analyze the DaVinci robot beyond the confines of a 2D space.



Starting off with the rotation matrix, the following conditions are satisfied,

$$x_6 // y_0 \quad (25)$$

$$y_6 // x_0 \quad (26)$$

$$z_6 // -z_0 \quad (27)$$

$$(28)$$

yielding,

$$R_6^0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (29)$$

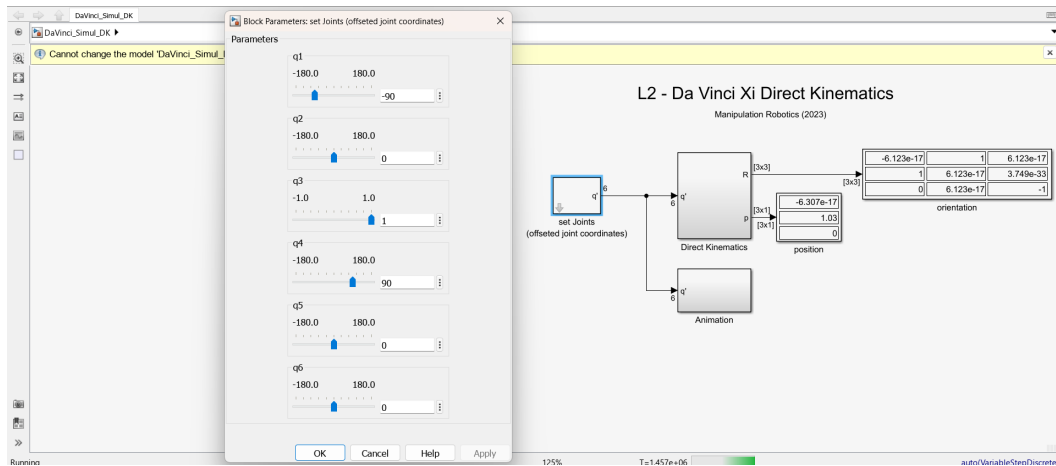
As for the position matrix, the shift from  $O_0$  to  $O_6$  occurs along the  $y_0$  axis such that

$$p_6^0 = \begin{bmatrix} 0 & d_3 + a_5 + a_6 & 0 \end{bmatrix} \quad (30)$$

or,

$$p_6^0 = \begin{bmatrix} 0 & 1.03 & 0 \end{bmatrix} \quad (31)$$

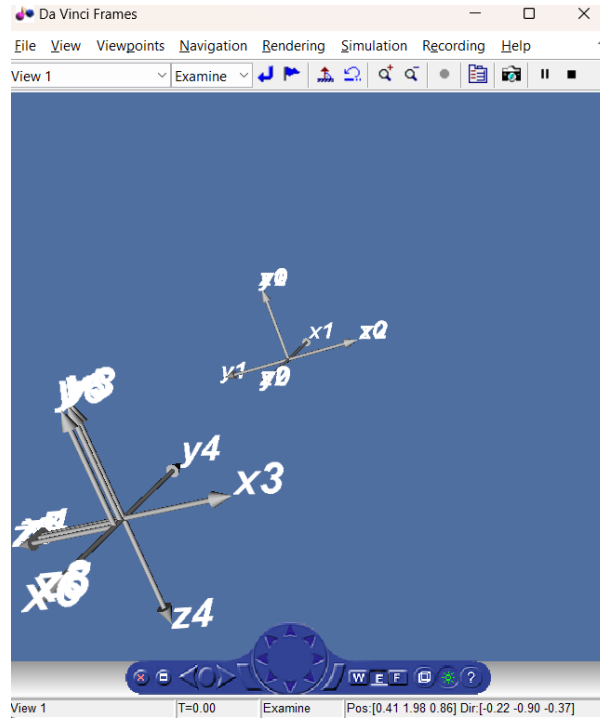
This is confirmed via simulation:



**Figure 9:** Rotation and Position Simulated Matrices for Configuration#3

Note that due to the numerical methodology of *Simulink*, some values of the  $R$  matrix that should be zero do not appear as so, but as very small values. This is expected and does not invalidate the model.

Below is the referential's positioning for Configuration#3, validating 8.



**Figure 10:** Simulated DaVinci's Referentials for Configuration#3

### 3 Question 3

Solving the Inverse Kinematics model is a more complex task than solving the Direct Kinematics model. This is because for a given set of rotation,  $R_6^0$ , and position,  $p_6^0$ , matrices there can be more than one joint coordinates,  $q'$ , solution, whereas for the Direct Kinematics one set of joint coordinates  $q'$  has only one solution in terms of  $R_6^0$  and  $p_6^0$ .

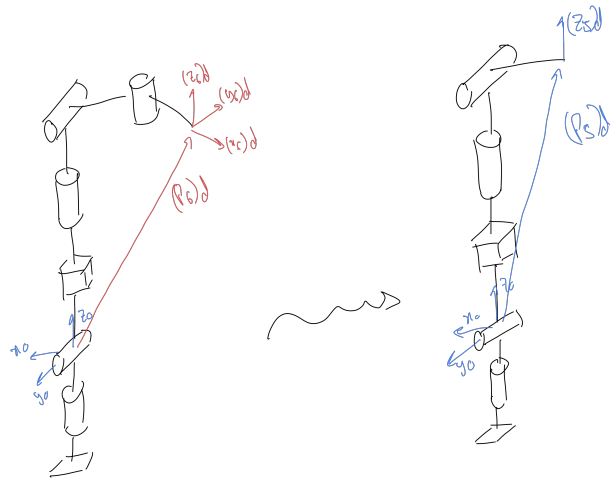
To solve the Inverse Kinematics of the model, a kinematic decoupling approach was taken where the arm  $(\theta_1, \theta_2, d_3)$  and the wrist  $(\theta_4, \theta_5, \theta_6)$  are treated separately.

Before approaching both the arm and the wrist of the DaVinci, there are some simplifications that can be made:

$$p_5 = p_6 - a_6 \cdot x_6 \quad (32)$$

$$z_5 = z_6 \quad (33)$$

This can be seen through the model below:



**Figure 11:** Inverse Kinematics Model for  $p_5$

Furthermore,

$$z_4 = \frac{p_5 \times z_5}{\|p_5 \times z_5\|} \quad (34)$$

$$x_5 = z_5 \times z_4 \quad (35)$$

The first simplification, 34, can be made since,

$$(p_0, p_1, p_2, p_3, p_4, p_5) \in \Sigma \quad (36)$$

$$z_5 \in \Sigma, \text{ coplanar} \quad (37)$$

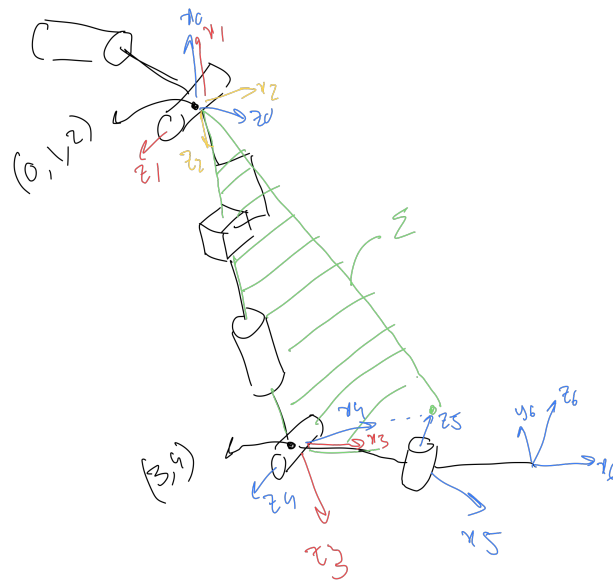
$$\angle(z_4, \Sigma) = 90 \quad (38)$$

Whereas the second simplification, 35, can be made since,

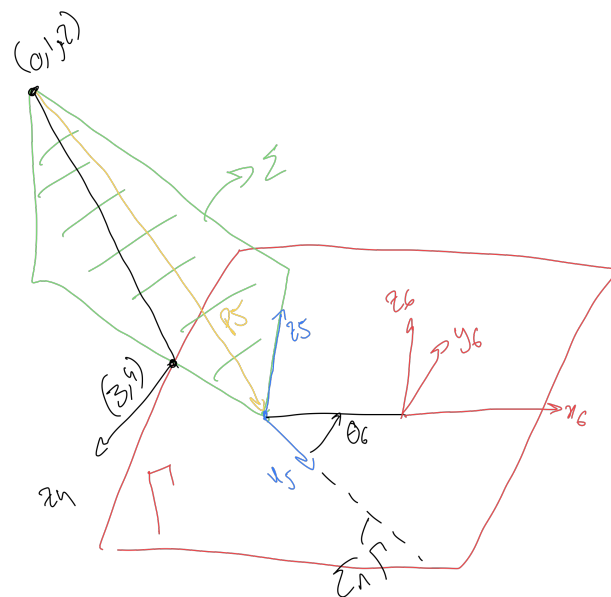
$$\angle(z_5, \Gamma) = 90 \quad (39)$$

$$\begin{aligned} \Gamma // x_5, \text{ coplanar} \\ \Gamma // y_5, \text{ coplanar} \end{aligned} \quad (40)$$

which allows  $x_5$  to lie along the intersection between  $\Sigma$  and  $\Gamma$ . Both of these simplifications are confirmed through the models below:



**Figure 12:**  $\Sigma$  Plane representation for Inverse Kinematics model



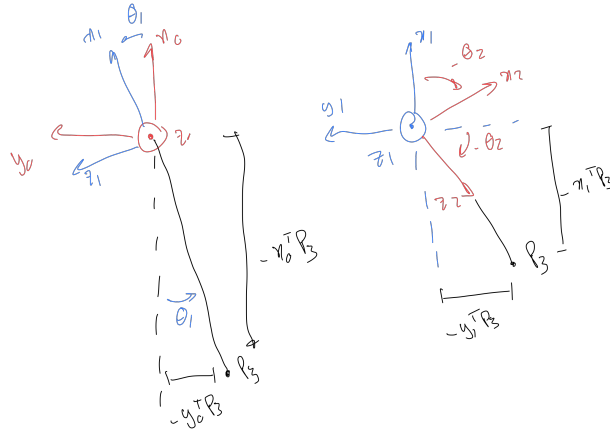
**Figure 13:**  $\Gamma$  Plane representation for Inverse Kinematics model

Lastly this allows for the calculation of  $p_3$ ,

$$p_3 = p_5 - a_5 \times x_5 \quad (41)$$

### 3.1 Arm Solution

The arm solution encompasses the DaVinci from the referential  $O_0$ , up until point  $p_3$  as detailed below:



**Figure 14:** DaVinci's Arm representation for Inverse Kinematics model

This gives,

$$\theta_1 = \arctan(-y_0^T p_3, -x_0^T p_3) \quad (42)$$

which knowing that,

$$x_1 = \underbrace{\begin{bmatrix} c_1 & 0 & -s_1 \\ s_1 & 0 & c_1 \\ 0 & -1 & 0 \end{bmatrix}}_{R_1^0} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad (43)$$

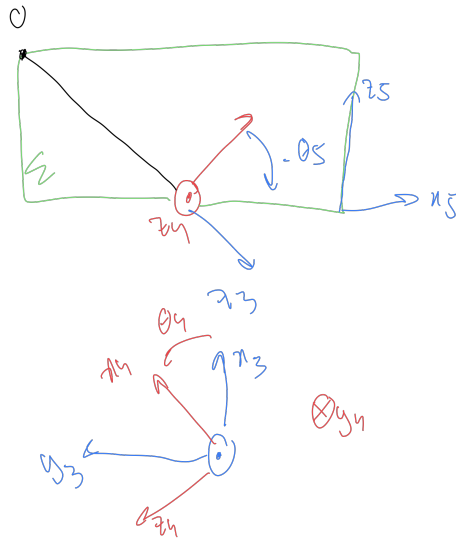
yields,

$$\theta_2 = -\arctan(-x_1^T p_3, -y_1^T p_3) \quad (44)$$

$$d_3 = ||p_3|| \quad (45)$$

### 3.2 Wrist Solution

The wrist solution encompasses the DaVinci from the point  $p_3$ , up until referential  $O_6$ , following along the planes  $\Sigma$  and  $\Gamma$ , as detailed below:



**Figure 15:** DaVinci's Wrist representation for Inverse Kinematics model

$$z_3 = \frac{p_3}{\|p_3\|} \quad (46)$$

$$x_4 = z_4 \times z_3 \quad (47)$$

$$\theta_5 = -\arctan(z_5^T x_4, x_5^T x_4) \quad (48)$$

$$y_5 = z_5 \times x_5 \quad (49)$$

$$\theta_6 = \arctan(y_5^T x_6, x_5^T x_6) \quad (50)$$

which knowing that,

$$x_3 = \underbrace{\begin{bmatrix} c_1 c_2 & -s_1 & c_1 s_2 \\ s_1 c_2 & c_1 & s_1 s_2 \\ -s_2 & 0 & c_2 \end{bmatrix}}_{R_3^0} \cdot \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad (51)$$

yields,

$$\theta_4 = \arctan(y_3^T x_4, x_3^T x_4) \quad (52)$$

### 3.3 Joint Offsets

After having,

$$q = (\theta_1, \theta_2, d_3, \theta_4, \theta_5, \theta_6) \quad (53)$$

the offsets are inputted creating a new joint coordinate vector  $q'$ ,

$$q' = q - \text{offset} \quad (54)$$

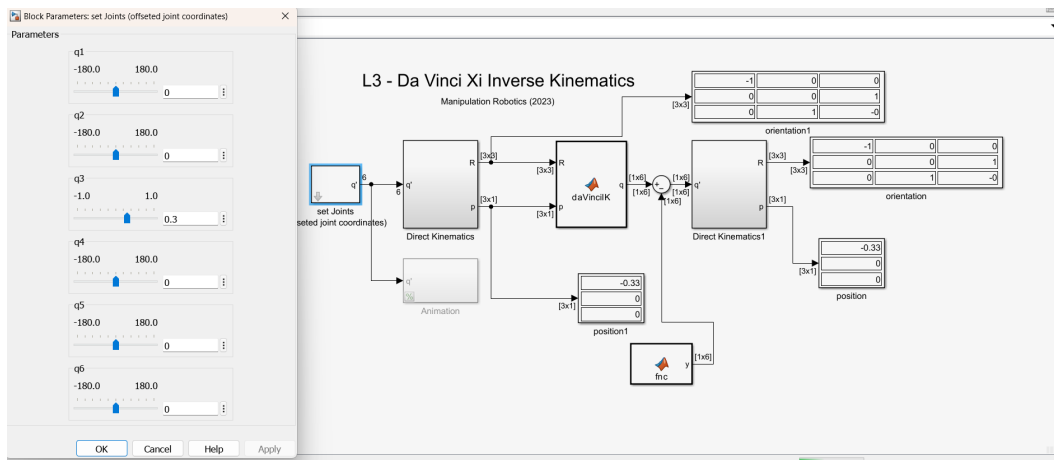
### 3.4 Model Validation

After creating one closed-form solution for the Inverse Kinematics of the DaVinci, it is necessary to validate it against the previously confirmed Direct Kinematics model.

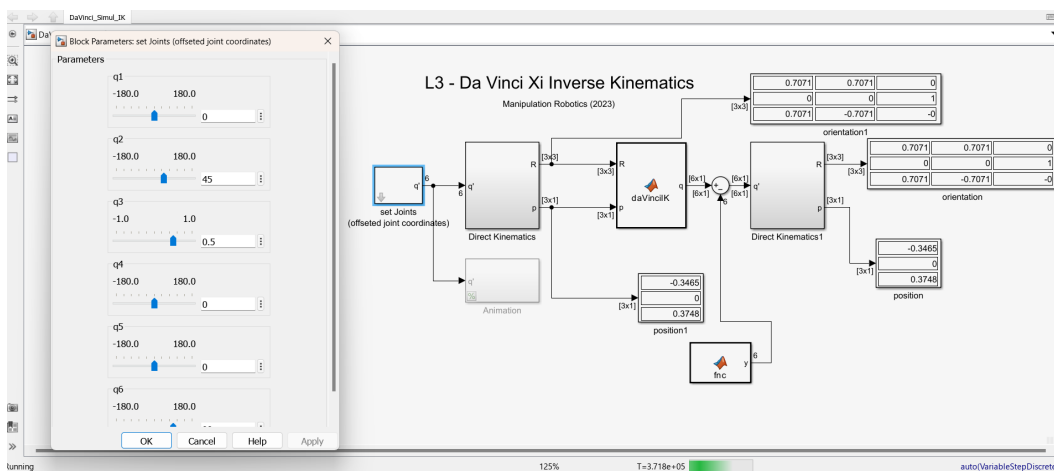
To verify that the Inverse Kinematics model is giving joint coordinates,  $q'_\alpha$ , such model must be fed a given rotation,  $(R_6^0)_\alpha$ , and position,  $(p_6^0)_\alpha$ , matrices.

However, to validate the Inverse Kinematics model, it is necessary to verify that the joint coordinates produced by it,  $q'_\alpha$ , lead to the same rotation,  $(R_6^0)_\alpha$ , and position,  $(p_6^0)_\alpha$ , matrices through the same Direct Kinematics model.

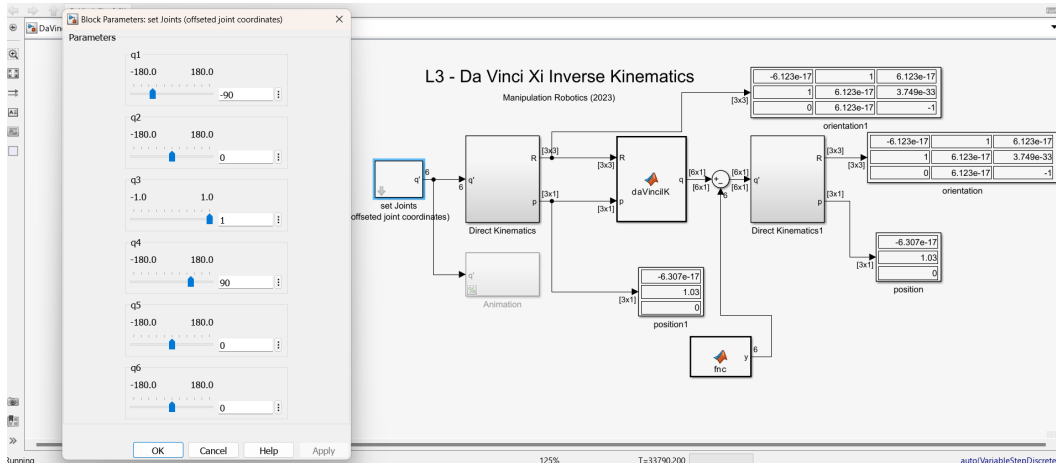
Below is the Inverse Kinematics model validation with the 3 configurations as proposed in 2:



**Figure 16:** Inverse Kinematics Validation for Home Configuration



**Figure 17:** Inverse Kinematics Validation for Configuration #2



**Figure 18:** Inverse Kinematics Validation for Configuration #3

Note that the numerical methodology approximations come into play again, whilst not invalidating the model.

The *func* block is responsible for the joint offsets.

## 4 Question 4

The Jacobian matrix is defined as follows:

$$J = \begin{pmatrix} j_{P1} & \dots & j_{Pn} \\ \vdots & \ddots & \vdots \\ j_{O1} & \dots & j_{On} \end{pmatrix} \quad (55)$$

where,

$$\begin{pmatrix} j_{Pi} \\ j_{Oi} \end{pmatrix} = \begin{cases} \begin{pmatrix} z_{i-1} \\ 0 \end{pmatrix}, & \text{for Prismatic Joints} \\ \begin{pmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{pmatrix}, & \text{for Revolute Joints} \end{cases} \quad (56)$$

For this particular case of the DaVinci Robot, since there is only 1 prismatic joint at  $z_3$ , the Jacobian matrix is as follows:

$$J(q) = \begin{pmatrix} z_0 \times (p_e - p_0) & z_1 \times (p_e - p_1) & z_2 & z_3 \times (p_e - p_3) & z_4 \times (p_e - p_4) & z_5 \times (p_e - p_5) \\ z_0 & z_1 & 0 & z_3 & z_4 & z_5 \end{pmatrix}$$

where the calculation for  $z_i$  and  $p_e/p_i$  is obtained through the following:

$$z_{i-1} = R_1^0(q_1) \dots R_{i-1}^{i-2}(q_{i-1}) z_0 \quad (57)$$

$$p_e = A_1^0(q_1) \dots A_n^{n-1}(q_n) p_0 \quad (58)$$

$$p_{i-1} = A_1^0(q_1) \dots A_{i-1}^{i-2}(q_{i-1}) p_0 \quad (59)$$

For the problem at hand, it is assumed:

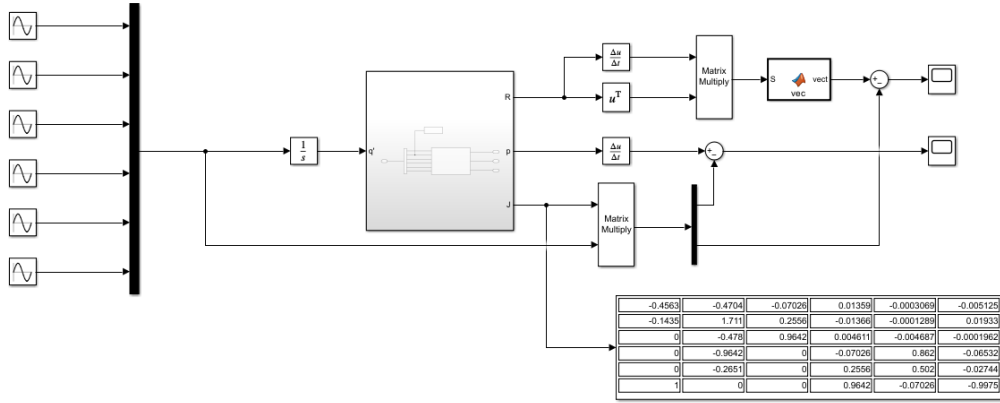
$$z_0^T = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \quad (60)$$

$$p_0^T = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \quad (61)$$



## 4.1 Validation through Numerical Differentiation

The Jacobian model of the DaVinci Robot was implemented as:



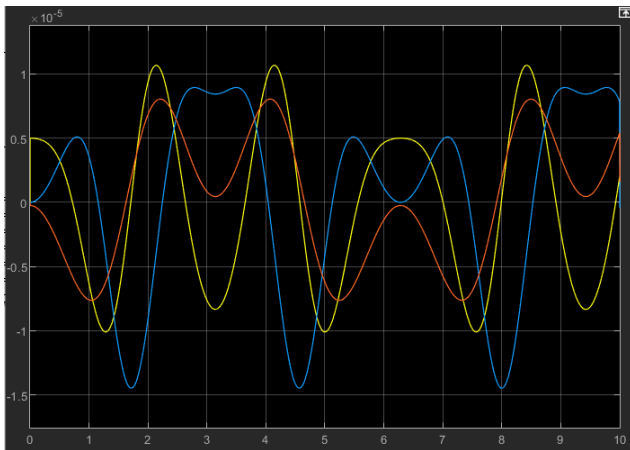
**Figure 19:** Jacobian Simulink Model

In order to validate the Jacobian model through numerical differentiation, it is necessary to verify,

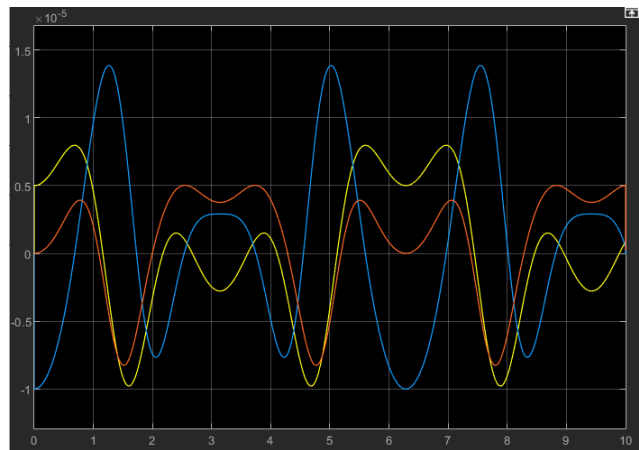
$$\dot{P}_n - J_p \dot{q} = 0 \quad (62)$$

$$\omega_n - J_0 \dot{q} = 0 \quad (63)$$

which are the differences between the linear/angular velocities obtained numerically and obtained through the Jacobian.



**(a)** Linear Velocity throughout Time



**(b)** Angular Velocity throughout Time

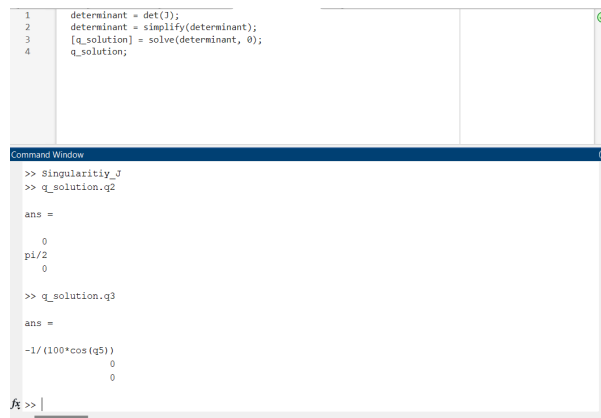
The results obtained in 20a/20b were as expected, since the difference between the linear velocity ( $\dot{P}$ ) calculated through the Jacobian and the numerical one are very close to zero (order of magnitude of  $10^{-5}$ ), and the same happens for the angular velocity ( $\omega_n$ ).

It is concluded that the results calculated through the Jacobian are close to the numerical results, hence the model is validated.

## 4.2 Kinematic Singularities

The Jacobian is, in general, a function of the configuration  $q$ . Should that configuration generate a rank-deficient  $J$ , such configuration is deemed a kinematic singularity. Singularities represent configurations at which mobility of the structure is reduced and it could result in infinite solutions for the Inverse Kinematics.

Four different singularities were found,



```

1 determinant = det(J);
2 determinant = simplify(determinant);
3 [q_solution] = solve(determinant, 0);
4 q_solution;

Command Window
>> Singularity_J
>> q_solution.q2

ans =

0
pi/2
0

>> q_solution.q3

ans =

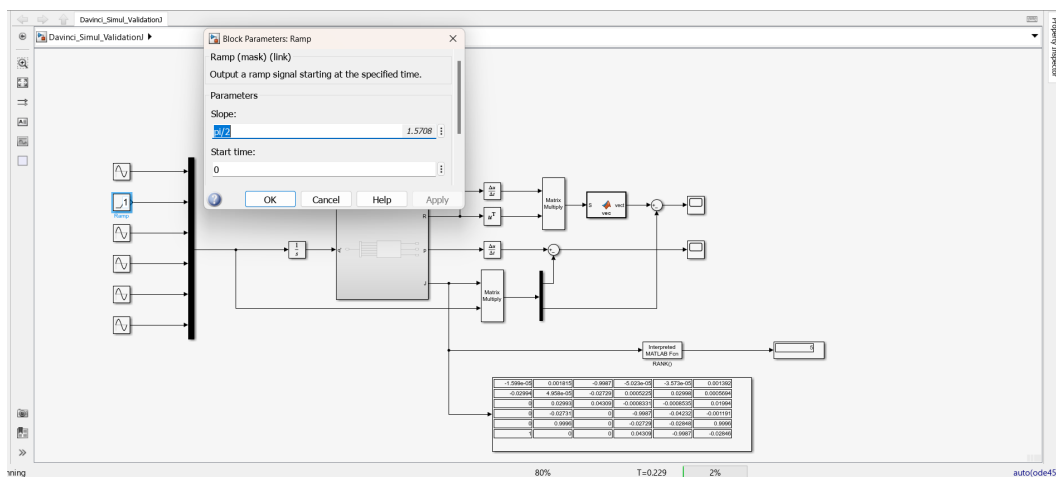
-1/(100*cos(q5))
0
0

```

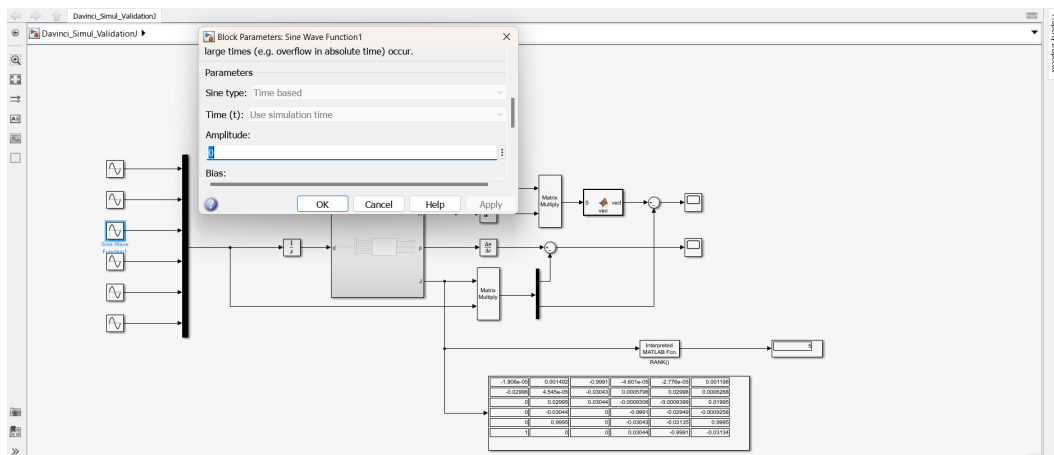
**Figure 21:** MATLAB Script for Singularity Calculation

such that  $\det(J) = 0$ . These are -  $q_2 = 0$ ,  $q_2 = \frac{\pi}{2}$ ,  $q_3 = 0$ ,  $q_3 = -\frac{1}{100\cos(q_5)}$ . However from these four, only two appear to affect the rank of the Jacobian matrix, decreasing it to 5, indicating a linear dependency between 2 columns.

These are  $q_2 = \frac{\pi}{2}$  and  $q_3 = 0$ . Below is their effect on the rank (bottom right corner of both images):



**Figure 22:** Effect of  $q_2 = \frac{\pi}{2}$  on the rank

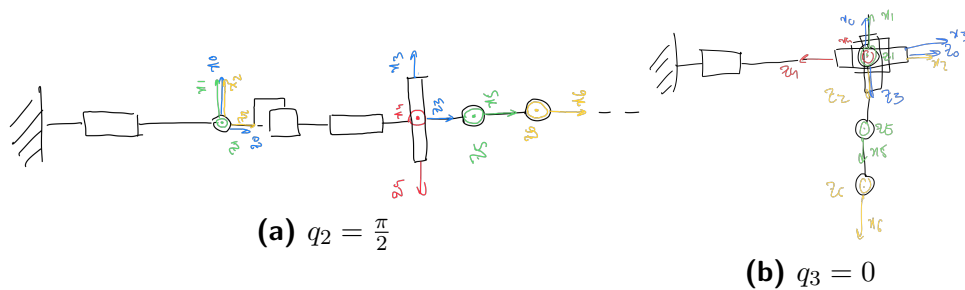


**Figure 23:** Effect of  $q_3 = 0$  on the rank

For the first represented singularity ( $q_2 = \frac{\pi}{2}$ ), it can be said that the arm of the DaVinci lies all along the  $z_0$  axis. On the other hand, the second singularity ( $q_3 = 0$ ) translates into a junction of all reference frames (except 5 and 6, due to  $a_5$  and  $a_6$  being greater than 0) in a single point. In this way,  $q_2$  is a boundary singularity and  $q_3$  is an internal singularity.

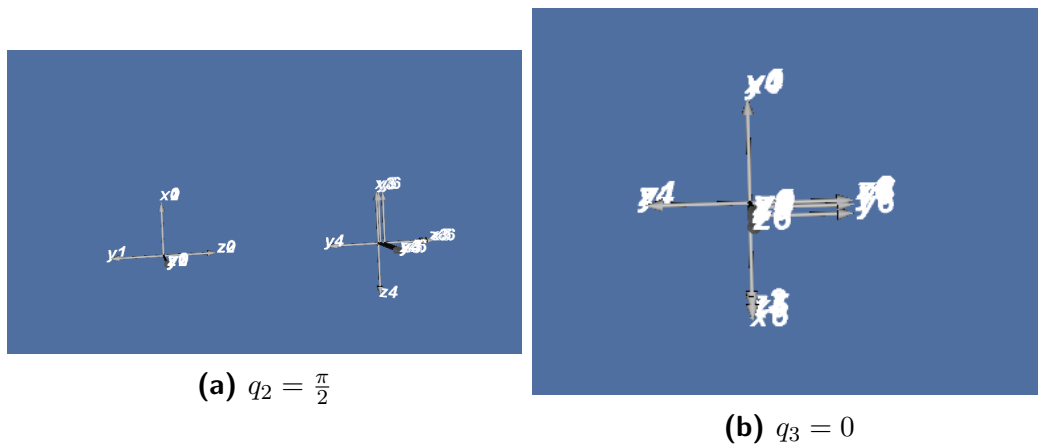
The reason why the rank was verified through *Simulink* and not a **MATLAB** script was the fact that due to the symbolic nature of matrix  $J$ , the rank sometimes did not update to 5 when plugging in  $q_3 = 0$ . Hence, the group decided to check the rank through *Simulink*.

Concluding here are the effects of the singularities in the DaVinci,



**Figure 24:** Singularities Design

Such designs are confirmed through simulation:



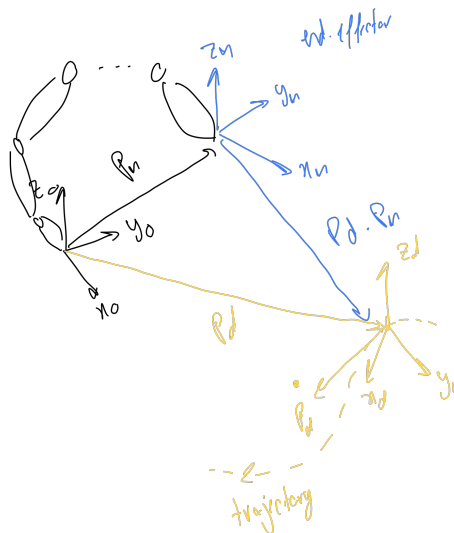
**Figure 25:** Singularities Design through Simulation

## 5 Question 5

As for the last analysis of the DaVinci Robot, the CLIK, Closed Loop Inverse, Kinematics, model was implemented, giving a more control oriented perspective on the DaVinci's Inverse Kinematics. The CLIK's controller is made up of the position control and orientation control's sub-architectures whilst its error sub-architecture is the orientation error.

### 5.1 Position Control

As outlined below,



**Figure 26:** Position Control Design

The position control is a sub-architecture of the CLIK whose purpose is to calculate the end-effector's linear velocity,  $\dot{P}_n$ . It achieves this with the aid of a position error, [65](#) that hyper-tunes the real linear velocity, [64](#), of the end-effector, for a given time frame.

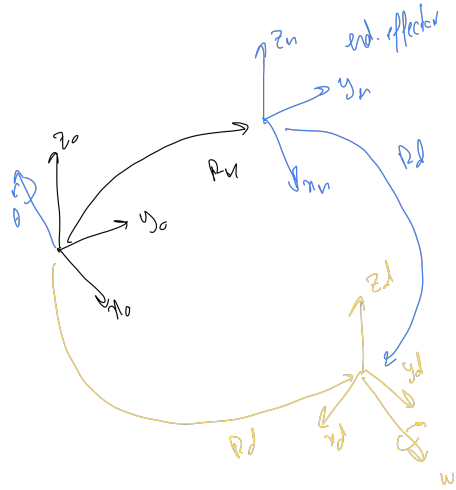
$$\dot{P}_n = \dot{P}_d + K_p(P_d - P_n) \quad (64)$$

$$e_P = P_d - P_n \quad (65)$$

where  $K_p$  is a positive definite matrix.

## 5.2 Orientation Control

As outlined below,



**Figure 27:** Orientation Control Design

The orientation control is a sub-architecture of the CLIK whose purpose is to calculate the angular velocity, 67. To achieve this, it first calculates the rotation matrix 66, through the dot-product of the desired rotation,  $R_d$ , and the Direct Kinematics rotation,  $R_n$ . From here the rotation matrix provides the parameters  $\theta$  and  $r$ , as it will be described later, to form the vector  $e_O = r\theta$ . With this the angular velocity,  $w_n$  is also hyper-tuned for a given time frame, while the controller is running.

$$R_e = R_d R_n^T \quad (66)$$

$$\omega_n = \omega_d + K_o e_o(\theta, r) \quad (67)$$

## 5.3 CLIK Controller Architecture

With the sub-architectures for position control and orientation error, the CLIK controller can be assembled,

$$(\dot{P}_n^T \omega_n^T)^T = J \dot{q} \quad (68)$$

such that,

$$\dot{q}(t) = J^{-1} \begin{pmatrix} \dot{P}_d + K_p(P_d - P_n) \\ \omega_d + K_o e_o(\theta, r) \end{pmatrix}, q(t) = \int_0^t \dot{q}(u) du + q(0) \quad (69)$$

where  $q(0)$  denotes the initial configuration of the manipulator.

## 5.4 Orientation Error

The orientation error is a sub-architecture of the CLIK whose purpose is to create the error vector, as seen below

$$e_o = \theta r \quad (70)$$

The parameters for the error vector are obtained through Rodrigues' rotation formula. This formulation can be derived geometrically through the observation of how a point  $p$ , rotates about the unitary axis,  $r$  ( $\|r\| = 1$ ), by an angle  $\theta$ ,

$$p' = rr^T p + s_\theta S(r)p - c_\theta S(r)S(r)p \quad (71)$$

where  $S(a)$  denotes the skew-symmetric form of vector  $\vec{a}$ ,

$$a \times b = S(a)b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}, \text{vect}(S(a)) = a \quad (72)$$

With this the rotation formula is given by,

$$R(\theta, r) = rr^T + s_\theta S(r) - c_\theta S(r)S(r), \text{ with } \|r\| = 1, \theta \in \mathbb{R} \quad (73)$$

whose expansion yields,

$$R(\theta, r) = \begin{pmatrix} r_x^2 & r_x r_y & r_x r_z \\ r_x r_y & r_y^2 & r_y r_z \\ r_x r_z & r_y r_z & r_z^2 \end{pmatrix} + \sin \theta \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} - \cos \theta \begin{pmatrix} -r_y^2 - r_z^2 & r_x r_y & r_x r_z \\ r_x r_y & -r_x^2 - r_z^2 & r_y r_z \\ r_x r_z & r_y r_z & -r_x^2 - r_y^2 \end{pmatrix}$$

With the rotation matrix built, the angle and axis of the referential must be extracted from it.

The most general case of axis extraction is,

$$R - R^T = 2s_\theta S(r) \quad (74)$$

$$r = \frac{1}{2s_\theta} \text{vect}(R - R^T), \forall s_\theta \neq 0 \quad (75)$$

while for the angle it comes from the  $\text{trace}(R)$ ,

$$\text{trace}(R) = r_x^2 + r_y^2 + r_z^2 + 2c_\theta(r_x^2 + r_y^2 + r_z^2) = 1 + 2c_\theta \in [-1, 3] \quad (76)$$

giving,

$$\theta = \arccos\left(\frac{\text{trace}(R) - 1}{2}\right), \forall -1 \leq \text{trace}(R) \leq 3 \quad (77)$$

However looking back at 75, there can be a conflict should  $s_\theta = 0$ .

To solve this it is assumed,

$$R(0, r) = I_3, \text{ and arbitrary direction} \quad (78)$$

or,

$$R(\pm\pi, \pm r) = rr^T - S(r)S(r) \text{ reflection along the } r \text{ axis as seen below} \quad (79)$$

Nevertheless, any direction  $x, y, z$  may be used to determine  $r$ ,

$$r = \frac{x + x^i}{\|x + x^i\|} = \frac{\hat{e} + R\hat{e}}{\sqrt{(\hat{e} + R\hat{e})^T(\hat{e} + R\hat{e})}} = \frac{\hat{e} + R\hat{e}}{\sqrt{2 + 2\hat{e}^T R\hat{e}}} \quad (80)$$

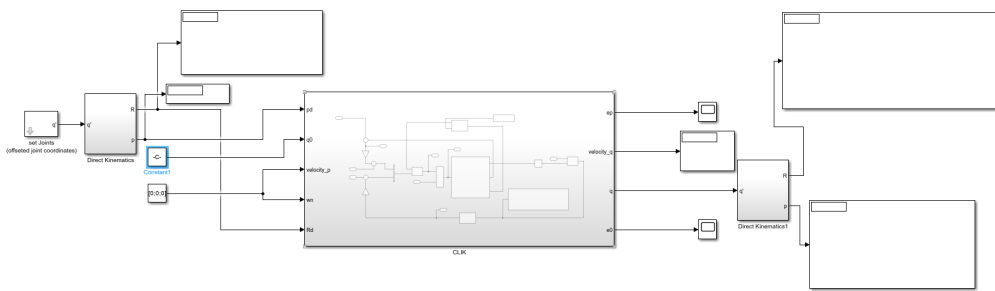
Similar to before, there is a case where the denominator may be 0, for example,  $x' = -x$ . Hence a voting scheme may be employed to determine the largest possible denominator, resulting in,

$$r = \pm \frac{\hat{e}_i + R\hat{e}_i}{\sqrt{2 + 2\hat{e}_i^T R\hat{e}_i}}, \text{ with } i : \hat{e}_i^T R\hat{e}_i \in \max(\text{diag}(R)) \quad (81)$$

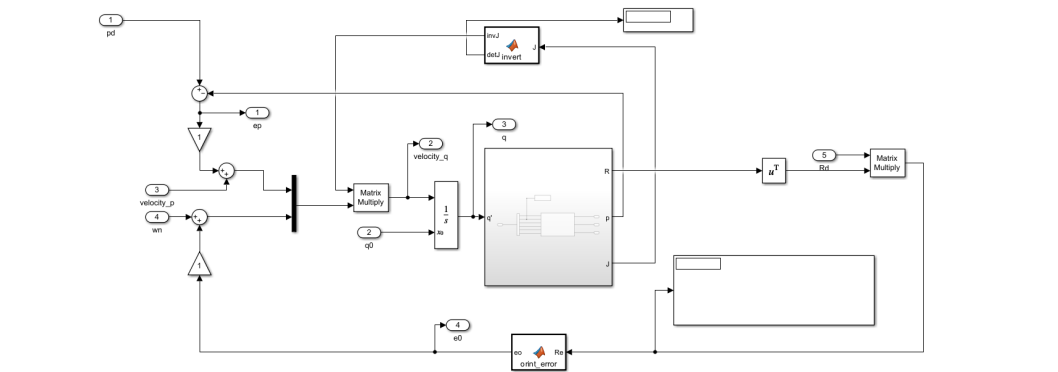
In conclusion, below is the generalized form as to how to obtain both  $\theta$  and  $r$  for the calculation of the error vector,

$$\{\theta, r\} = \begin{cases} \begin{cases} \theta = \arccos\left(\frac{\text{trace}(R)-1}{2}\right) \\ r = \frac{1}{2s_\theta} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix} \end{cases} & , \text{ if } -1 < \text{trace}(r) < 3 \\ \begin{cases} \theta = \pm\pi \\ \begin{cases} r = \pm \frac{\hat{i} + R\hat{i}}{\sqrt{2+2R_{11}}} & , \text{ if } R_{11} \in \max(\text{diag}(R)) \\ r = \pm \frac{\hat{j} + R\hat{j}}{\sqrt{2+2R_{22}}} & , \text{ if } R_{22} \in \max(\text{diag}(R)) \\ r = \pm \frac{\hat{k} + R\hat{k}}{\sqrt{2+2R_{33}}} & , \text{ if } R_{33} \in \max(\text{diag}(R)) \end{cases} \end{cases} & , \text{ if } \text{trace}(r) = -1 \\ \begin{cases} \theta = 0 \\ r = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \end{cases} & , \text{ otherwise} \end{cases} \quad (82)$$

Below is the CLIK macro system and subsystem,



**Figure 28:** CLIK Macro System

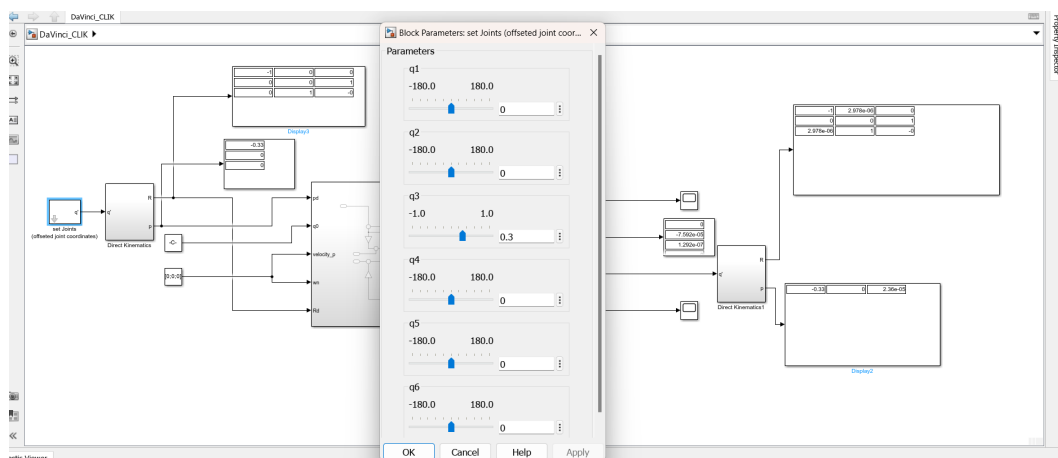


**Figure 29:** CLIK SubSystem

Initial conditions were taken as  $q(0) = [0; \pi/4; 0.5; 0; 0; 0]$ , since they are not a singularity, and initial linear and angular velocities as 0.

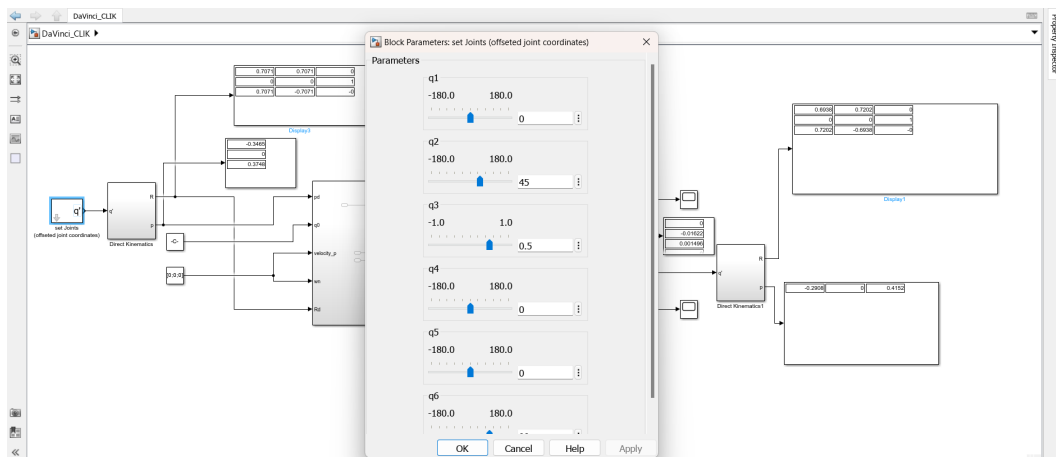
## 5.5 Validation through Comparison with One-Shot Inverse Kinematics and Error Analysis

Below are the three configurations as implemented previously

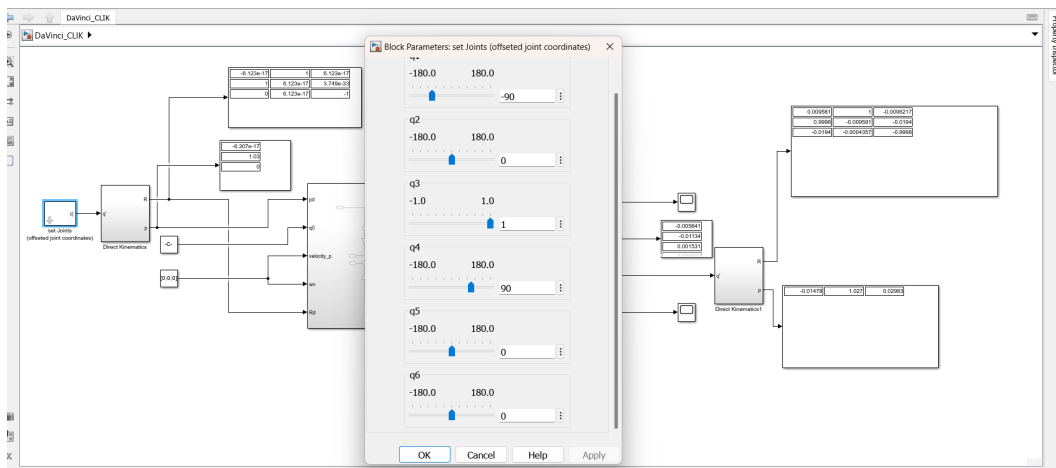


**Figure 30:** Configuration#1 for CLIK Model





**Figure 31:** Configuration#2 for CLIK Model

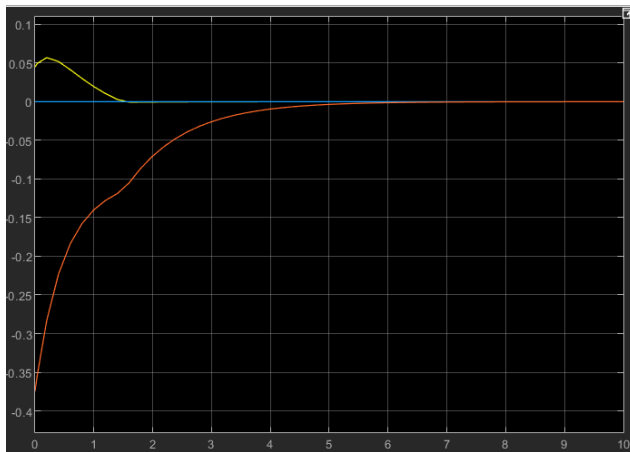


**Figure 32:** Configuration#3 for CLIK Model

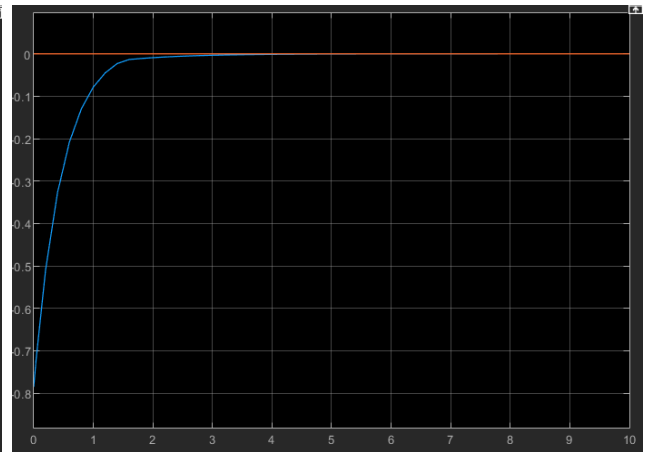
With this model the numerical methodology factor of *Simulink* comes more into play since the model is a closed loop model. However, this does not invalidate the model, and it is possible to see that the output for the rotation and position matrices coincides with the input and is very similar to the matrices computed in 3.4.

Another way to validate this model is to check the position and orientation errors. If these tend to 0 then it follows the theoretical model.

Below are the errors for the three proposed configurations,

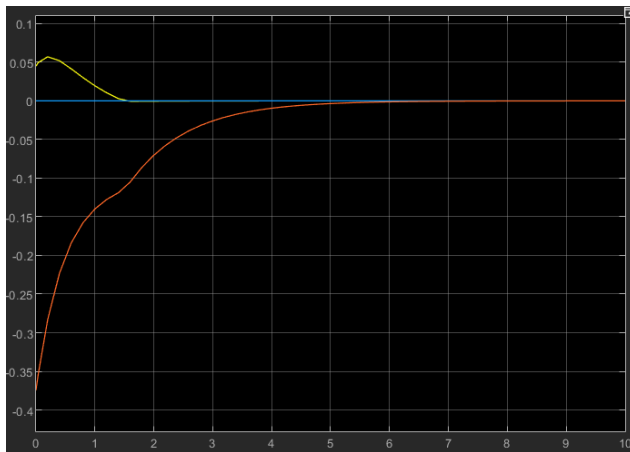


(a) Position Error for Configuration#1

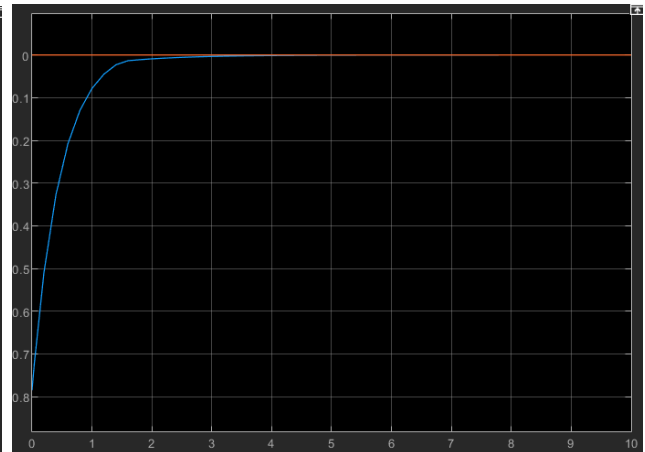


(b) Orientation Error for Configuration#1

**Figure 33:** Configuration#1

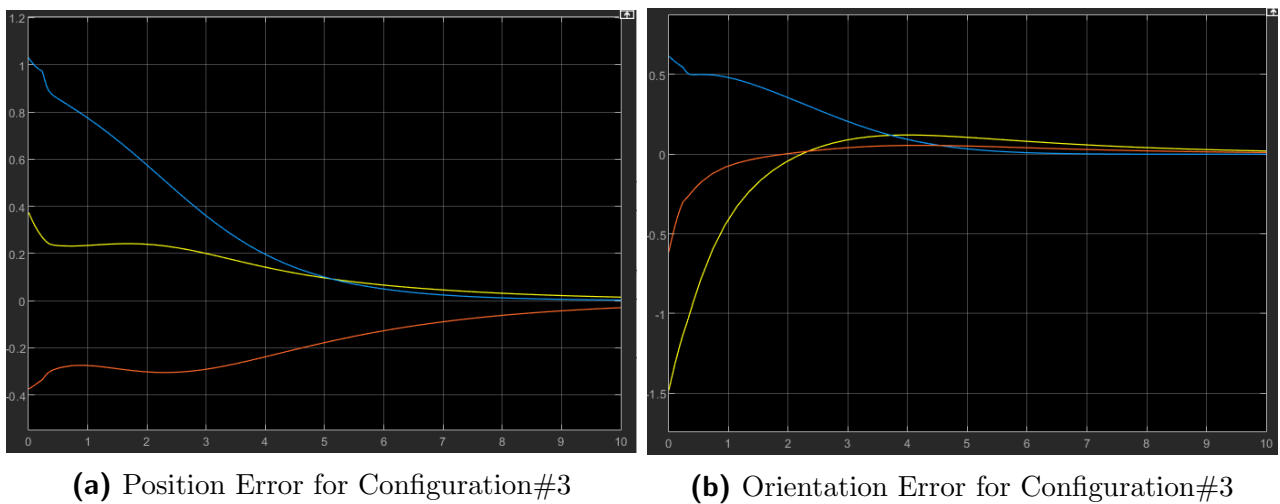


(a) Position Error for Configuration#2



(b) Orientation Error for Configuration#2

**Figure 34:** Configuration#2



**Figure 35:** Configuration#3

Hence the model is validated as both position and orientation errors tend to 0.

## 6 References

[1] Robotics of Manipulation - [RMan - Kinematics] Slides of Theoretical Classes 2022/2023, 2nd Semester (MEMec), by Professor Jorge M. M. Martins