



**Politecnico  
di Torino**

**Automotive Engineering**

**A.Y. 2024/2025**

**Numerical Modelling and Simulation - Part B**

**LAIB Reports**

**Students:**

Walter Maggio (S343988)

Giuseppe Maria Marchese (S348145)

# **Contents**

# **List of Tables**

# **List of Figures**

# Report 1

## Adams

### 1.1 Introduction

This report primarily details the process of modeling a S.C.A.R.A. robot (as shown in Figure ??), using the software Adams View. The studied robot has 5-axis, with an articulated arm (3 d.o.f.) and a wrist (2 d.o.f.).

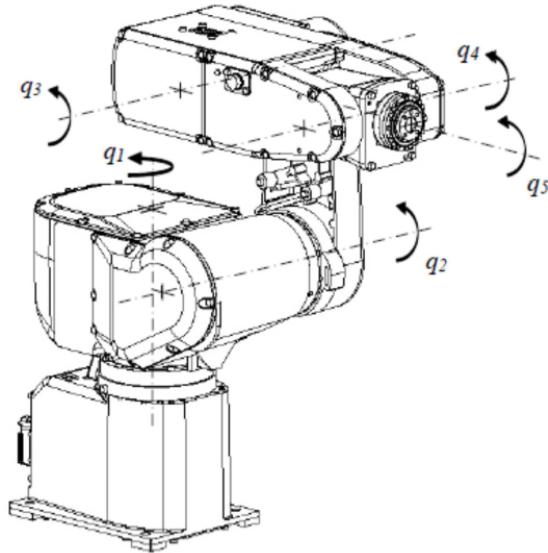


Figure 1.1: SCARA robot

Afterwards, trajectory files will be implemented to verify that the robot operates correctly and follows the intended motion paths.

## 1.2 Problem illustration

In figure ?? a front and a lateral view of the robot are shown. From that, the dimensions of the links and the position of the joints can be extrapolated. The basement frame is fixed and reported in the figure, where the length is in  $x_0$  direction, the height in  $z_0$  and the depth in  $y_0$ .

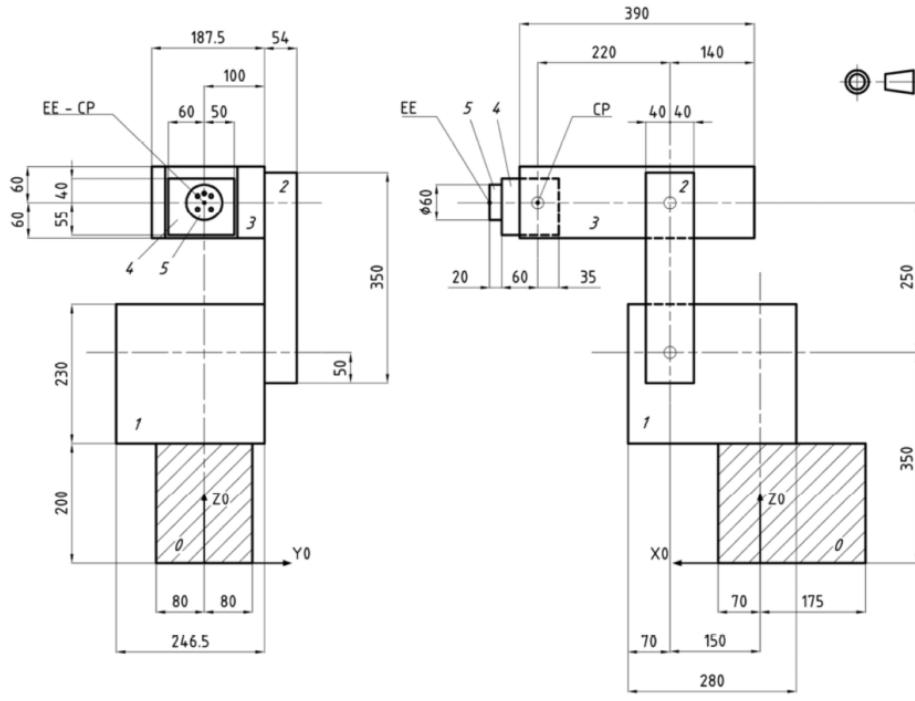


Figure 1.2: Scara robot drafting

All the parts belonging to the robot are parallelepiped blocks, except for the end-effector (element 5) of figure ?? which is cylindrical.

In figure ?? the final part of the robot and the payload are shown.

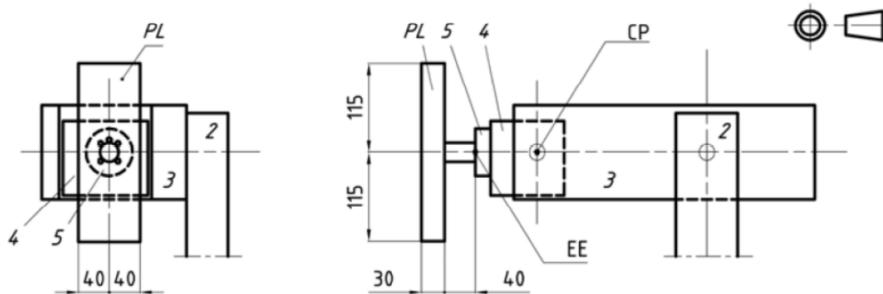


Figure 1.3: Payload drafting

Gravity has been set in the  $y_0$  direction and the selected working directory has been MMKS (mm, kg, N, s, deg).

## 1.3 Implementation

### 1.3.1 Creation of the model

In order to make the explanation as clear as possible, for the first body all the steps will be seen in the details. For the next, which are practically identical, only the dimensions of bodies and the reference frames will be reported.

First of all, the element 0 has to be positioned and fixed on the basement frame, otherwise in the simulations it would fall indefinitely due to gravity. In order to do this, a reference frame is placed, selecting the options "add to ground" and "global XZ plane".

Next, proceeding to the window for bodies, it is chosen the "Box" option in solids section. After that, "New Part" is selected and the necessary dimensions to place the element on the ground are computed. Once done, Adams automatically creates a reference marker to define the position of the new body.

The final step is to adjust the reference marker, ensuring it is in the correct position. Considering that the reference marker is on the lower edge of the box, its location is equal to the difference between it and the global reference frame.

The mentioned values can be evaluated in figure ?? and are reported in table ??.

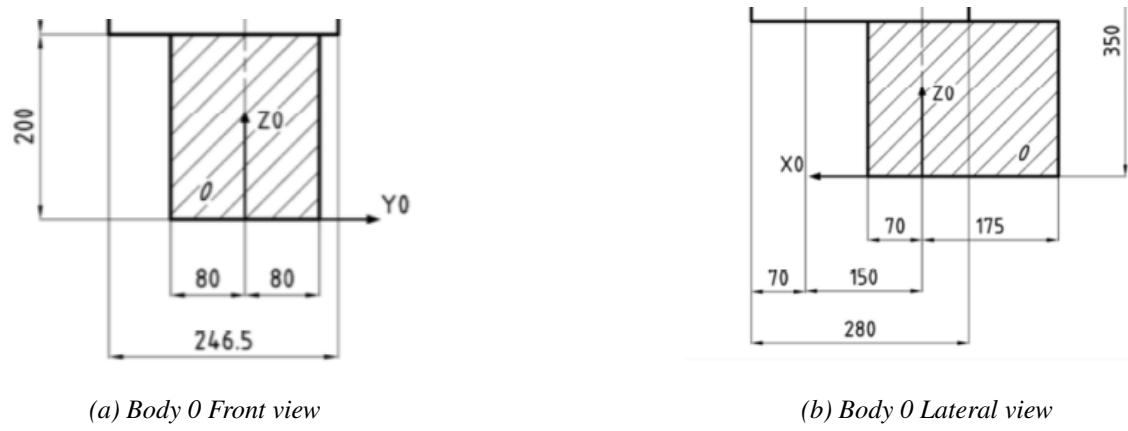


Figure 1.4: Body 0 sketch

| Body | Dimensions |        |       | Frame position |                |                |
|------|------------|--------|-------|----------------|----------------|----------------|
|      | Length     | Height | Depth | X <sub>i</sub> | Y <sub>i</sub> | Z <sub>i</sub> |
| 0    | 245        | 200    | 160   | -80            | 0              | -175           |

Table 1.1: Body and frame 0 values

For the next steps, which follow the same procedure, only the dimensions of the boxes and the location of their reference frame will be shown in table ??.

| Body     | Dimensions |        |       | Frame position |                |                |
|----------|------------|--------|-------|----------------|----------------|----------------|
|          | Length     | Height | Depth | X <sub>i</sub> | Y <sub>i</sub> | Z <sub>i</sub> |
| <b>0</b> | 245        | 200    | 160   | -70            | 0              | -80            |
| <b>1</b> | 280        | 230    | 246.5 | -220           | 200            | -146.5         |
| <b>2</b> | 80         | 350    | 54    | -190           | 300            | 100            |
| <b>3</b> | 390        | 120    | 187.5 | -400           | 540            | -87.5          |
| <b>4</b> | 95         | 95     | 110   | -430           | 545            | -60            |

*Table 1.2: Box (and relative frame) values*

The next step involves creating two cylinders (bodies 5 and End Effector). These are implemented using the same procedure as before, but by selecting the 'Cylinders' option in the Bodies window and specifying the length and radius in the Dimensions section. The same steps as in the box case have been followed to position them properly.

| Body      | Dimensions |        |  | Frame position |                |                |
|-----------|------------|--------|--|----------------|----------------|----------------|
|           | Length     | Radius |  | X <sub>i</sub> | Y <sub>i</sub> | Z <sub>i</sub> |
| <b>5</b>  | 20         | 30     |  | -430           | 600            | 0              |
| <b>EE</b> | 40         | 10     |  | -490           | 600            | 0              |

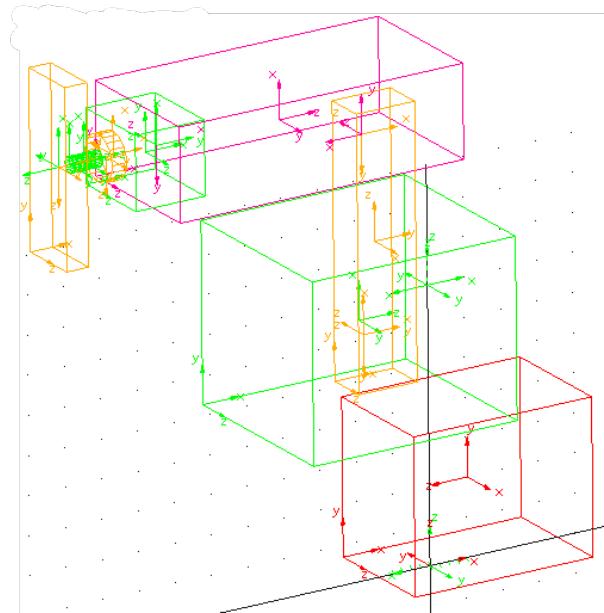
*Table 1.3: Cylinder (and relative frame) values*

The last body to add is the payload (PL). In this case the body type is a box, but the option "Add to part" has to be selected. Then the necessary values for dimensions and locations are given in the following table.

| Body      | Dimensions |        |       | Frame position |                |                |
|-----------|------------|--------|-------|----------------|----------------|----------------|
|           | Length     | Height | Depth | X <sub>i</sub> | Y <sub>i</sub> | Z <sub>i</sub> |
| <b>PL</b> | 30         | 230    | 80    | -520           | 485            | -40            |

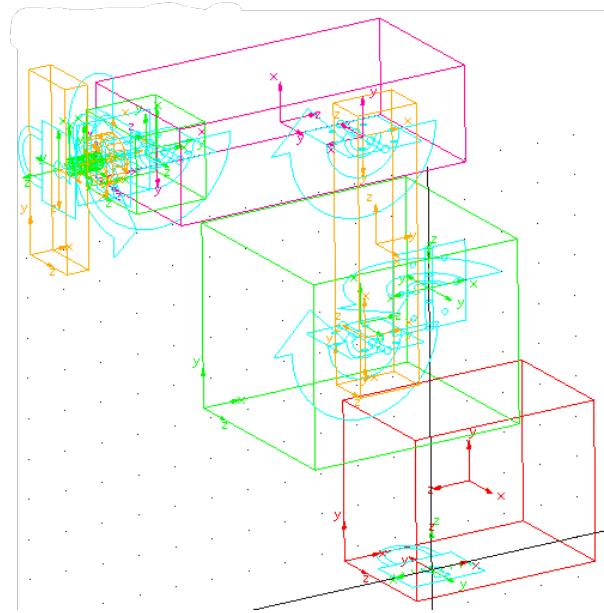
*Table 1.4: Payload (and relative frame) values*

The resulting wire frame is shown in figure ??



*Figure 1.5: wire frame visualization of 5 D.O.F. robot in Adams*

At this point the aim is to fix the connectors. The first one is the fixed joint, to fix the first box to the ground. The option two bodies one location has to be selected. Then the inputs are the body 0 and the ground, with the origin of the reference system as location. With the same procedure, the payload has to be connected to the body EE, selecting the origin of the body EE frame.



*Figure 1.6: wire frame visualization of 5 D.O.F. robot with joints*

Now, it is possible to create joints between the bodies. The joint type to be used is the revolute joint. However, this time, the option "Pick Geometry Feature" must be selected to specify the axis along which the revolute joint will rotate. In order to do this, a new reference

frame for the joints has to be created. Their location is evaluable in figure ??, where it is marked with a red dot. To define their orientation, it is sufficient to put the Z axis coherent with the rotation and follow the Denavit-Hartenberg convention.

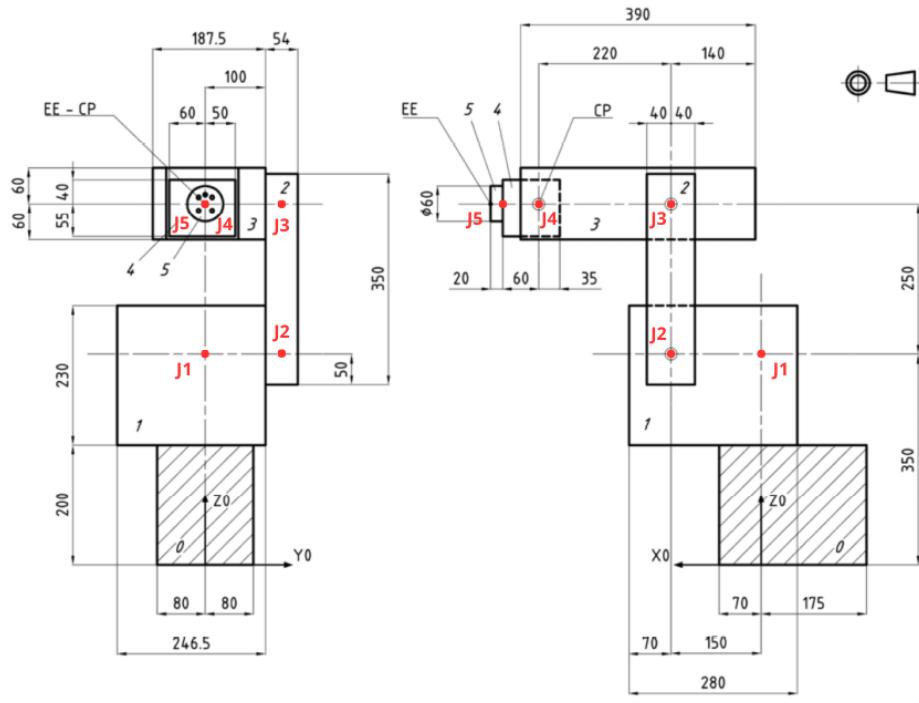


Figure 1.7: Joints location

To place the revolute joints, simply select body  $n = i$ , body  $n = i + 1$ , and then define the axis of rotation. The following table lists the corresponding axes for each joint and the location of the reference axis.

| Bodies | Joint    | RF location [mm] |     |     | RF orientation [degrees] |          |          |
|--------|----------|------------------|-----|-----|--------------------------|----------|----------|
|        |          | X                | Y   | Z   | Rot. Z 1                 | Rot. X 2 | Rot. Z 3 |
| 0 - 1  | <b>1</b> | 0                | 350 | 0   | 180                      | 90       | 0        |
| 1 - 2  | <b>2</b> | -150             | 350 | 100 | 90                       | 180      | 0        |
| 2 - 3  | <b>3</b> | -150             | 600 | 100 | 180                      | 180      | 0        |
| 3 - 4  | <b>4</b> | -370             | 600 | 0   | 90                       | 180      | 0        |
| 4 - 5  | <b>5</b> | -430             | 600 | 430 | 270                      | 90       | 180      |

Table 1.5: Revolute joints data

### 1.3.2 Motion generation

To create the desired motion, the first step is to insert the joints data by selecting "Create a 2D or 3D Data Spline" in the Data Elements window. Then, using the provided data, create the spline. The last step consists of selecting the spline in the simulation tree, then choosing the rotary motion in the Motion tab, and clicking on the appropriate joint where the motion needs to be applied.

At this point, for every motion created, by double clicking on a motion of the simulation tree, the edit tab has to be opened, in order to choose the desired fitting method for the spline and the data type (between displacement, velocity and acceleration). In this case, the Akima fitting method is chosen.

By inserting the initial conditions and confirming, the simulation can be done.

## 1.4 Results

In the following image the final result is reported with the End Effector (EE) trajectory.

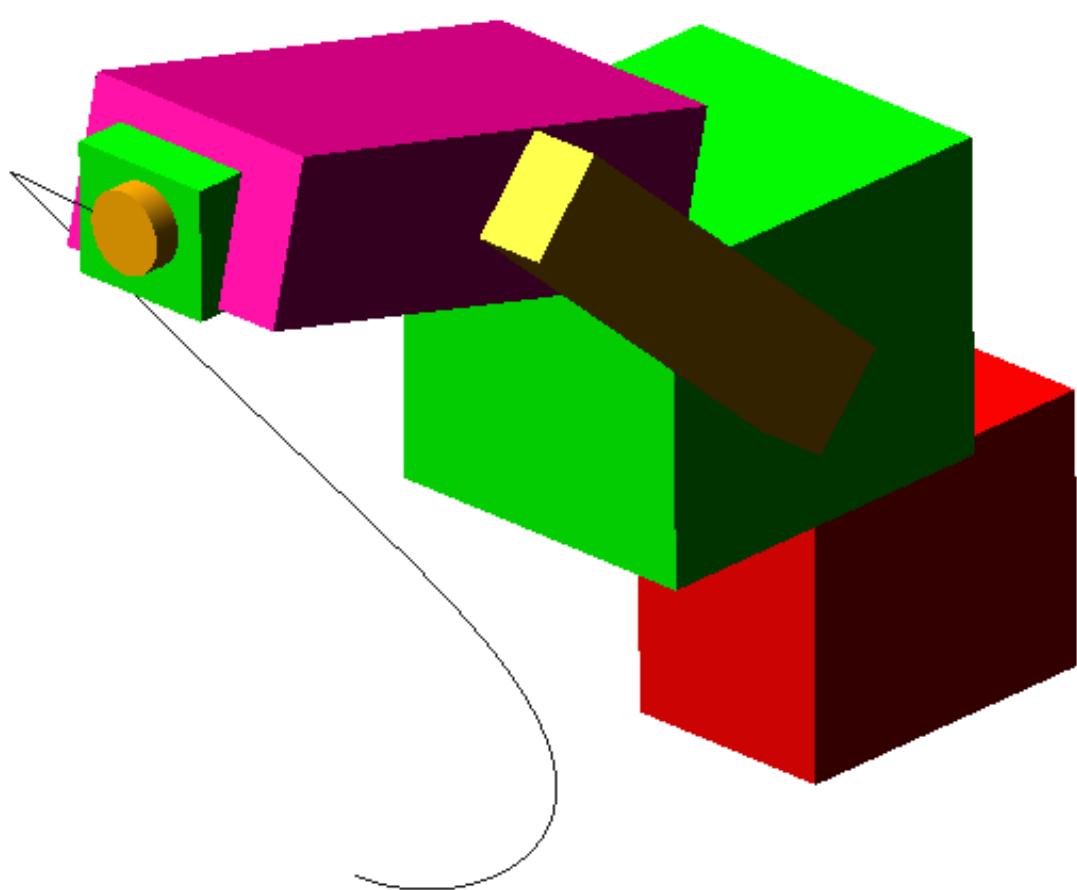
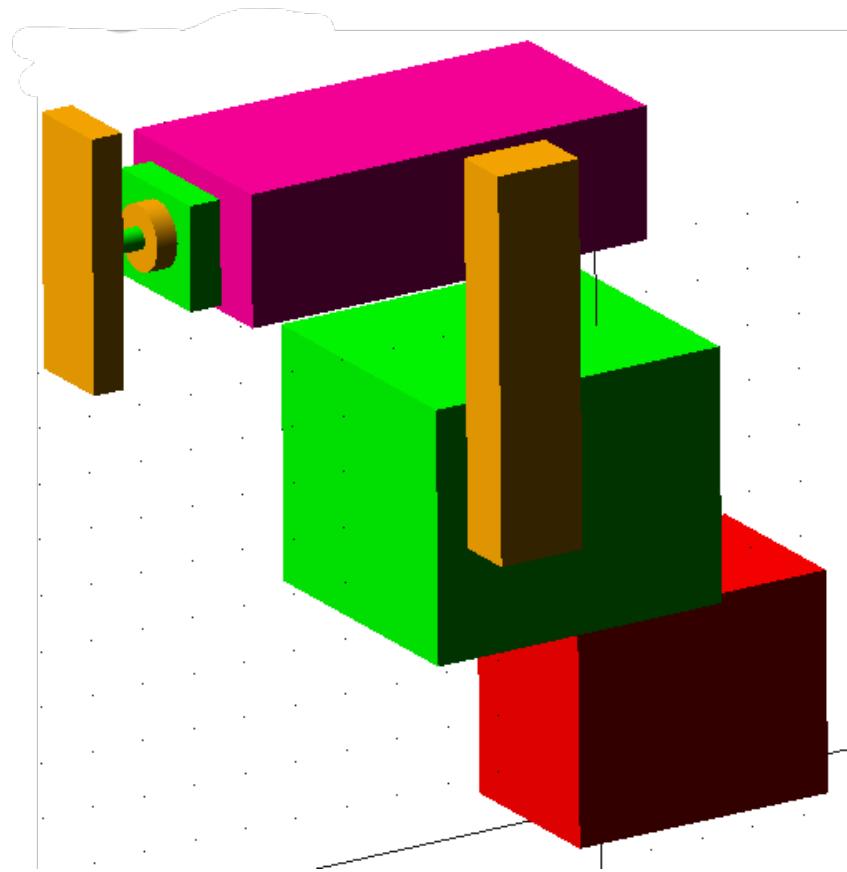


Figure 1.8: 5 D.O.F. robot modeled in Adams with E.E. trajectory

The version with the payload attached to the end effector is reported, too



*Figure 1.9: 5 D.O.F. robot modeled in Adams with payload*

# Report 2

## Transformation matrices

### 2.1 Introduction

The purpose of this report is to evaluate the transformation matrices of a triangle and outline the key steps taken to fulfill the following requirements:

1. Calculate the positioning matrix of the triangle ABC ( ${}^0A_1$ ).
2. Plot the unit vectors of the reference frames  $o_0x_0y_0z_0$  and the triangle  $O_1x_1y_1z_1$ .
3. Rotate and plot the initial triangle of  $90^\circ$  about axis  $y_1$ .
4. Rotate and plot the initial triangle of  $-90^\circ$  about axis  $y_0$ .
5. Rotate and plot the triangle from the last position of  $90^\circ$  about axis  $x_2$ .

### 2.2 Problem illustration

Triangle coordinates A, B and C are expressed in the fixed reference system  $o_0x_0y_0z_0$

$$\mathbf{P}_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{P}_2 = \begin{bmatrix} 0 \\ 7 \\ 1 \end{bmatrix} \quad \mathbf{P}_3 = \begin{bmatrix} 0 \\ 4 \\ 7 \end{bmatrix}$$

An other mobile reference system  $o_1x_1y_1z_1$  is fixed in the middle of the triangle base with the following coordinates expressed respect to fixed reference system.

$$\mathbf{P}_{O_1} = \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix} \quad (2.1)$$

The previous data are imported in the Matlab code as following

```

1 %triangle points
2
3 P1 = [0 1 1] ';
4 P2 = [0 7 1] ';
5 P3 = [0 4 7] ';
6
7 %positioning matrix of triangle
8
9 T0o = [P1,P2,P3,P1;...
10      1, 1, 1, 1];

```

Where the homogeneous positioning matrix of the triangle in the fixed reference frame  $o_0x_0y_0z_0$  is defined.

$${}^0\hat{\mathbf{T}} = \begin{bmatrix} P_{1x} & P_{2x} & P_{3x} & P_{1x} \\ P_{1y} & P_{2y} & P_{3y} & P_{1y} \\ P_{1z} & P_{2z} & P_{3z} & P_{1z} \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 7 & 4 & 1 \\ 1 & 1 & 7 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (2.2)$$

In figure ?? a schematization of the problem is provided.

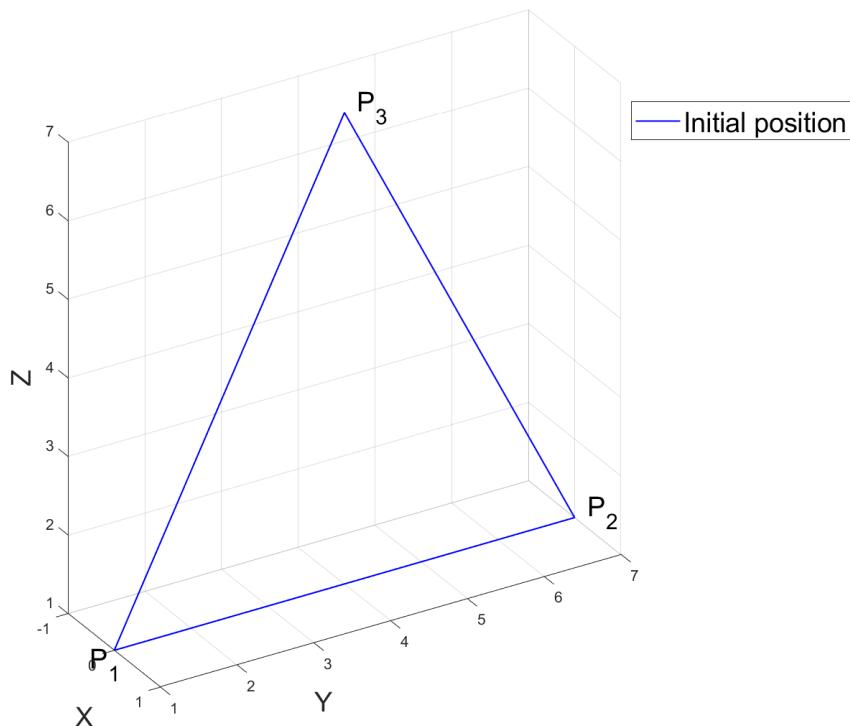


Figure 2.1: Schematization of initial conditions

```
1 % plot settings
2 triang_lw = 1;
3 legend_fs = 18;
4 label_fs = 18;
5 text_fs = 18;
6
7 X_trang = [P1(1,1) P2(1,1) P3(1,1) P1(1,1)];
8 Y_trang = [P1(2,1) P2(2,1) P3(2,1) P1(2,1)];
9 Z_trang = [P1(3,1) P2(3,1) P3(3,1) P1(3,1)];
10
11 %initial condition plot
12 figure(1)
13
14 plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
15
16 xlabel("X",FontSize=label_fs)
17 ylabel("Y",FontSize=label_fs)
18 zlabel("Z",FontSize=label_fs)
19 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
20 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
21 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
22 legend("Initial position",fontsize=legend_fs)
23 axis equal
24 grid on
```

## 2.3 Implementation of resolutions

### 2.3.1 Orientation matrix

To compute the orientation matrix  ${}^0A_1$  the versors of fixed reference system are defined.

$$\mathbf{i}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{j}_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{k}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

After that is possible to compute the versors of mobile reference system with respect to the fixed one.

$$\mathbf{i}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{j}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{k}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

It possible to assemble them in the orientation matrix  ${}^0A_1$ :

$${}^0\mathbf{A}_1 = \begin{bmatrix} i_{1x} & j_{1x} & k_{1x} \\ i_{1y} & j_{1y} & k_{1y} \\ i_{1z} & j_{1z} & k_{1z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.3)$$

The implementation on Matlab is reported.

```

1 %define versors of reference systems
2
3 scale = 1;
4
5 %versor sys 0
6
7 i0 = [1 0 0]' * scale;
8 j0 = [0 1 0]' * scale;
9 k0 = [0 0 1]' * scale;
10
11 %versors sys 1
12
13 i1 = [0 1 0]' * scale;
14 j1 = [0 0 1]' * scale;
15 k1 = [1 0 0]' * scale;
16

```

```

17 %positioning matrix A01
18
19 A01 = [i1 j1 k1];
20

```

### 2.3.2 Versors of reference systems plot

Vectors of both reference systems  $o_0x_0y_0z_0$  and  $o_1x_1y_1z_1$  are plotted.

```

1 %plot versors
2 figure(2)
3
4 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
5 hold on
6 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
7 hold on
8 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
9 hold on
10
11 plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
12 hold on
13
14 quiver3(P01(1),P01(2),P01(3),i1(1),i1(2),i1(3),Color=[1 0.5 0.5],...
15 LineWidth=vector_lw)
16 hold on
17 quiver3(P01(1),P01(2),P01(3),j1(1),j1(2),j1(3),Color = "#77AC30",...
18 LineWidth=vector_lw)
19 hold on
20 quiver3(P01(1),P01(2),P01(3),k1(1),k1(2),k1(3),Color = "#4DBEEE",...
21 LineWidth=vector_lw)
22 xlabel("X",FontSize=label_fs)
23 ylabel("Y",FontSize=label_fs)
24 zlabel("Z",FontSize=label_fs)
25 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
26 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
27 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
28 text(P01(1)-0.2,P01(2)-0.5,P01(3)-0.5,"P_{0_1}",...

```

```

29 "FontSize",text_fs)
30 legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
31 fontsize=legend_fs)
32 axis equal
33 grid on

```

The result is shown in figure ??.

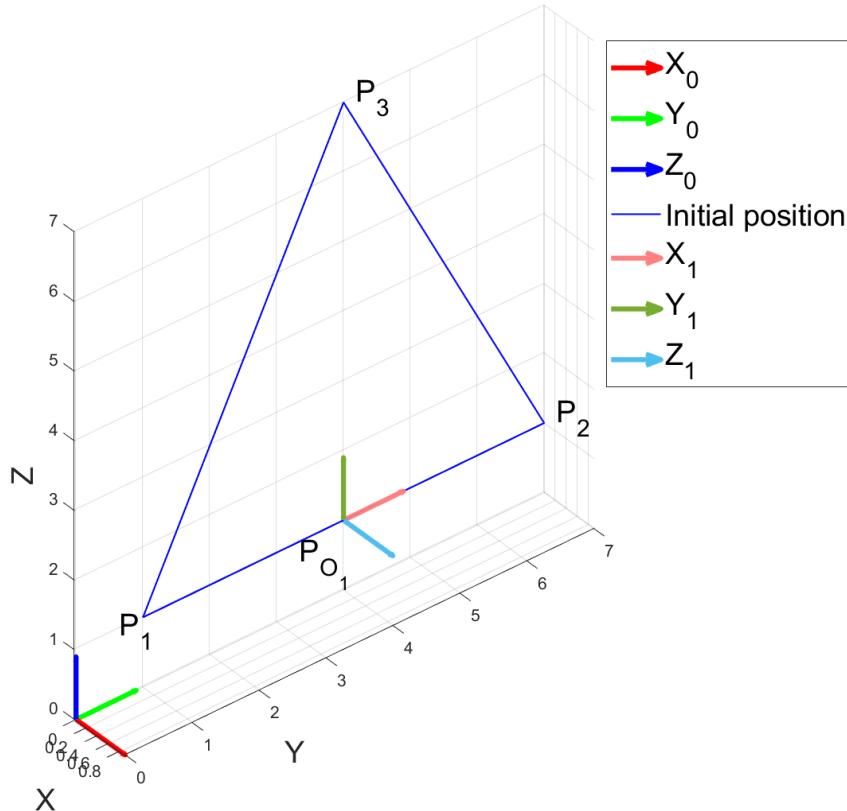


Figure 2.2: Versors of reference systems plot

### 2.3.3 Rotation of 90° about $y_1$

To perform the rotation it is necessary to calculate the homogeneous orientation matrix  ${}^0\hat{A}_1$  using the orientation matrix  ${}^0A_1$  and the positioning vector  $P_{O_1}$  (equations ??, ??).

$${}^0\hat{A}_1 = \begin{bmatrix} {}^0A_1 & P_{O_1x} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

The homogeneous rotation matrix  $R\hat{O}T(y, \theta)$  is computed in the following way.

$$\hat{\mathbf{ROT}}(\mathbf{y}, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

To compute all the homogeneous rotation matrices about  $y$ , a function has been developed .

```

1 function [hom_rotation_matrix] = rotYo(theta)
2
3 hom_rotation_matrix = [cos(theta)      0      sin(theta)      0; ...
4                      0          1          0          0; ...
5                      -1*sin(theta)  0      cos(theta)      0; ...
6                      0          0          0          1];
7 end

```

The resultant homogeneous rotation matrix about  $y$  of local reference system of an angle  $\theta_1 = 90^\circ$  is:

$$\hat{\mathbf{ROT}}(\mathbf{y}_1, \theta_1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation about the  $y$  axis of the mobile reference system  $o_1x_1y_1z_1$  is performed by post-multiplying the homogeneous orientation matrix  ${}^0\hat{A}_1$  with the homogeneous rotation matrix  $\hat{ROT}(y, \theta)$ .

$${}^0\hat{B}_1 = {}^0\hat{A}_1 \cdot \hat{ROT}(y_1, 90) \quad (2.6)$$

Afterwards, it is necessary to move the reference system of  ${}^0\hat{T}$  to the mobile reference system  ${}^1x_1y_1z_1$ , to obtain  ${}^1\hat{T}$ .

$${}^1\hat{A}_0 = {}^0\hat{A}_1^{-1} \implies {}^1\hat{T} = {}^1\hat{A}_0 \cdot {}^0\hat{T} \quad (2.7)$$

At this point, we are able to perform the rotation of the initial triangle using the equation ??

$${}^0\hat{T}' = {}^0\hat{B}_1 \cdot {}^1\hat{T} \quad (2.8)$$

The result is shown in the figure ??:

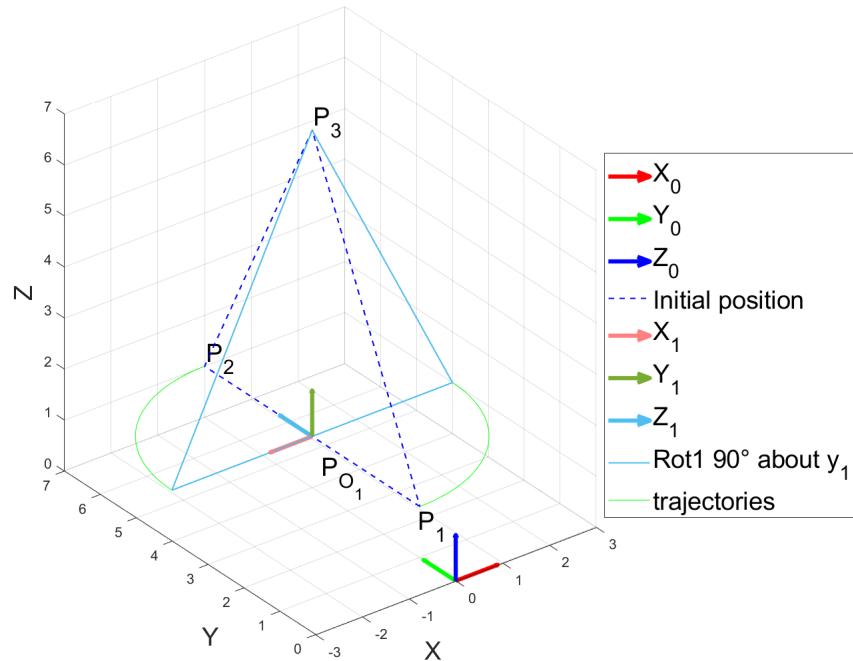


Figure 2.3: Rotation of the triangle about  $y_1$  of 90 degrees

The implementation in Matlab is now reported:

```

1 % homogeneous matrix A01o
2
3 A01o = [A01,P01;...
4     0 0 0 1];
5
6 %rotation of triangle about y1 of 90
7
8 theta1 = pi/2;
9
10 rotYo90 = rotYo(theta1);
11
12 B01o= A01o*rotYo90;
13
14 A10o = inv(A01o);
15
16 T1o = A10o * T0o;
17
18 T0_rotated1 = B01o * T1o;
19

```

```

20 %trajection of rotation
21
22 angles = linspace(0,theta1,1000);
23
24 for i=1:length(angles)
25     trajP1_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P1;1]);
26     trajP2_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P2;1]);
27     trajP3_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P3;1]);
28 end

```

```

1 %plot rotation 1
2 figure(3)
3
4 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
5 hold on
6 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
7 hold on
8 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
9 hold on
10
11 plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
12 hold on
13
14 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
15             Color=[1 0.5 0.5], LineWidth=vector_lw)
16 hold on
17 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
18             Color = "#77AC30", LineWidth=vector_lw)
19 hold on
20 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
21             Color = "#4DBEEE", LineWidth=vector_lw)
22 hold on
23
24 plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
25             Color = "#4DBEEE",LineWidth=triang_lw)
26 hold on
27

```

```

28 plot3(trajP1_rot1(:,1),trajP1_rot1(:,2),trajP1_rot1(:,3),"g")
29 hold on
30 plot3(trajP2_rot1(:,1),trajP2_rot1(:,2),trajP2_rot1(:,3),"g")
31 hold on
32 plot3(trajP3_rot1(:,1),trajP3_rot1(:,2),trajP3_rot1(:,3),"g"))
33
34 xlabel("X",FontSize=label_fs)
35 ylabel("Y",FontSize=label_fs)
36 zlabel("Z",FontSize=label_fs)
37 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
38 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
39 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
40 text(P01(1)-0.2,P01(2)-0.5,P01(3)-0.5,"P_{0_1}","FontSize",text_fs)
41 legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
42 "Rot1 90° about y_1","trajectory",fontsize=legend_fs)
43 axis equal
44 grid on

```

### 2.3.4 Rotation of -90° about $y_0$

Using the same procedure of the previous paragraph (??), the rotation matrix about the fixed reference system of an angle  $\theta_2 = -90^\circ$  is computed .

$$\hat{\text{ROT}}(\mathbf{y}_0, \theta_2) = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To realize the new set of rotations a new local reference system  $o_2x_2y_2z_2$ , equal to  $o_1x_1y_1z_1$ , is defined. What was said implies that:

$${}^0\hat{A}_2 = {}^0\hat{A}_1 \quad \text{and} \quad {}^2\hat{T} = {}^1\hat{T}$$

The rotation about the y axis of the mobile reference system  $o_0x_0y_0z_0$  is performed by pre-multiplying the homogeneous orientation matrix  ${}^0\hat{A}_1$  with the homogeneous rotation matrix  $R\hat{O}T(y_0, \theta_2)$ .

$${}^0\hat{C}_2 = R\hat{O}T(y_0, 90) \cdot {}^0\hat{A}_2 \quad (2.9)$$

Using the positioning matrix of the triangle of the local reference system, computed in paragraph ??, the rotation is performed.

$${}^0\hat{T}'' = {}^0\hat{C}_2 \cdot {}^2\hat{T} \quad (2.10)$$

In the next figure ?? the result is shown:

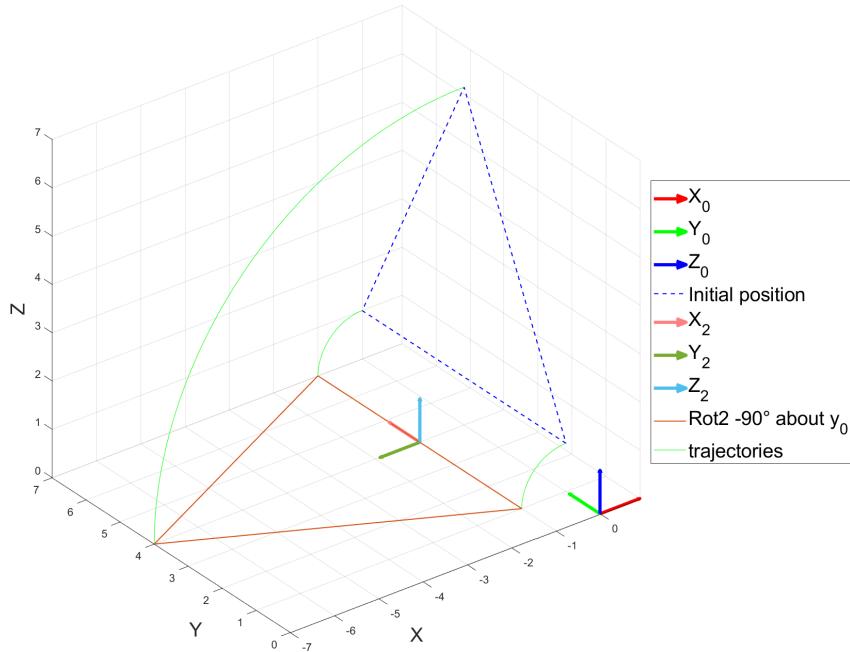


Figure 2.4: Rotation of the triangle about  $y_0$  of -90 degrees

In the following lines the Matlab implementation is reported.

```

1 %rotation of triangle about y0 oh theta2
2 theta2 = -pi/2;
3
4 A20o = A10o;
5
6 T2o = T1o;
7
8 A02o = A01o;
9
10 rotYom90 = rotYo(theta2);
11
12 C02o= rotYom90 * A02o;
13
14 T0_rotated2 = C02o * T2o;
15
16 %trajection of rotation 2
17
18 angles_2 = linspace(0,theta2,1000);
19
20 for i=1:length(angles)
21     trajP1_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P1;1]);
22     trajP2_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P2;1]);
23     trajP3_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P3;1]);
24 end

```

```

1 %plot rotation 2
2 figure(4)
3
4 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
5 hold on
6 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
7 hold on
8 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
9 hold on
10
11 plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
12 hold on

```

```
13
14 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
15     Color=[1 0.5 0.5], LineWidth=vector_lw)
16 hold on
17 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
18     Color = "#77AC30", LineWidth=vector_lw)
19 hold on
20 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
21     Color = "#4DBEEE", LineWidth=vector_lw)
22 hold on
23
24 plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
25 Color= "#D95319",LineWidth=triang_lw)
26 hold on
27
28 plot3(trajP1_rot2(:,1),trajP1_rot2(:,2),trajP1_rot2(:,3),"g")
29 hold on
30 plot3(trajP2_rot2(:,1),trajP2_rot2(:,2),trajP2_rot2(:,3),"g")
31 hold on
32 plot3(trajP3_rot2(:,1),trajP3_rot2(:,2),trajP3_rot2(:,3),"g")
33
34 xlabel("X",FontSize=label_fs)
35 ylabel("Y",FontSize=label_fs)
36 zlabel("Z",FontSize=label_fs)
37
38 legend("X_0","Y_0","Z_0","Initial position","X_2","Y_2","Z_2",...
39 "Rot2 -90° about y_0","trajectories",fontsize=legend_fs)
40 axis equal
41 grid on
```

### 2.3.5 Rotation from last position of 90° about $x_2$

To perform the rotation about the  $x$  axis a new rotation matrix is created.

$$\hat{\mathbf{ROT}}(\mathbf{y}, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is implemented in Matlab, as a function, in the following way.

```

1 function [hom_rotated_matrix_X] = rotXo(theta)
2
3 hom_rotated_matrix_X = [ 1           0           0           0; ...
4                         0           cos(theta) -sin(theta) 0; ...
5                         0           sin(theta)  cos(theta) 0; ...
6                         0           0           0           1];
7 end

```

The resultant homogeneous rotation matrix about  $x_2$  of  $\theta_3 = 90^\circ$  is equal to:

$$\hat{\mathbf{ROT}}(\mathbf{x}_2, \theta_3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

At this point it is possible to calculate the new orientation matrix ( ${}^0\hat{D}_2$ ) by post-multiplying it with the rotation matrix ( $\hat{ROT}(x_2, \theta_3)$ ).

$${}^0\hat{D}_2 = {}^0\hat{C}_2 \cdot \hat{ROT}(x_2, 90) \quad (2.11)$$

Afterwards, it is necessary to move the triangle rotated in paragraph ?? from reference system  $o_0x_0y_0z_0$  to reference system  $o_2x_2y_2z_2$ . After that the rotation can be performed.

$${}^2\hat{C}_0 = {}^0\hat{C}_2^{-1} \quad (2.12)$$

$${}^2\hat{T}'' = {}^2\hat{C}_0 \cdot {}^0\hat{T}'' \quad (2.13)$$

$${}^0\hat{T}''' = {}^0\hat{D}_2 \cdot {}^2\hat{T}'' \quad (2.14)$$

The result is shown in the following figure.

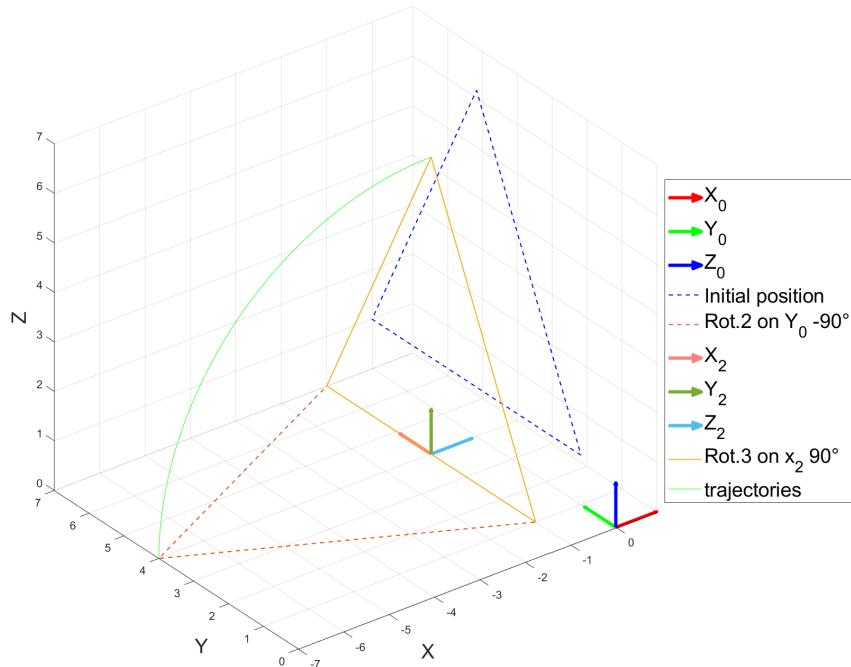


Figure 2.5: Rotation of the triangle about  $x_1$  of 90 degrees

The Matlab implementation is now reported.

```

1 %rotation of triangle about x2 oh theta3
2 theta3 = pi/2;
3
4 rotXo90 = rotXo(theta3);
5
6 D02o = C02o * rotXo90;
7
8 C20o = inv(C02o);
9
10 T2 = C20o * T0_rotated2;
11
12 T0_rotated3 = D02o * T2;
13
14 %trajection of rotation 2
15
16 angles_3 = linspace(0,theta3,1000);
17
18 for i=1:length(angles)
19     trajP1_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...

```

```

20    T0_rotated2(:,1));
21    trajP2_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
22    T0_rotated2(:,2));
23    trajP3_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
24    T0_rotated2(:,3));
25 end

```

```

1 %plot rotation 2
2 figure(5)
3
4 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
5 hold on
6 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
7 hold on
8 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
9 hold on
10
11 plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
12 hold on
13
14 plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),"--",...
15 LineWidth= triang_lw, Color= "#D95319")
16 hold on
17
18 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,1),D02o(2,1),D02o(3,1),...
19     Color=[1 0.5 0.5], LineWidth=vector_lw)
20 hold on
21 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,2),D02o(2,2),D02o(3,2),...
22     Color = "#77AC30", LineWidth=vector_lw)
23 hold on
24 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,3),D02o(2,3),D02o(3,3),...
25     Color = "#4DBEEE", LineWidth=vector_lw)
26 hold on
27
28 plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
29 LineWidth= triang_lw, Color="#EDB120")
30 hold on

```

```
31
32 plot3(trajP1_rot3(:,1),trajP1_rot3(:,2),trajP1_rot3(:,3),"g")
33 hold on
34 plot3(trajP2_rot3(:,1),trajP2_rot3(:,2),trajP2_rot3(:,3),"g")
35 hold on
36 plot3(trajP3_rot3(:,1),trajP3_rot3(:,2),trajP3_rot3(:,3),"g")
37
38 xlabel("X",FontSize=label_fs)
39 ylabel("Y",FontSize=label_fs)
40 zlabel("Z",FontSize=label_fs)
41
42 legend("X_0","Y_0","Z_0","Initial position","Rot.2 on Y_0 -90°",...
43 "X_2","Y_2","Z_2","Rot.3 on x_2 90°","trajectories",fontsize=legend_fs)
44 axis equal
45 grid on
```

## 2.4 Final plot

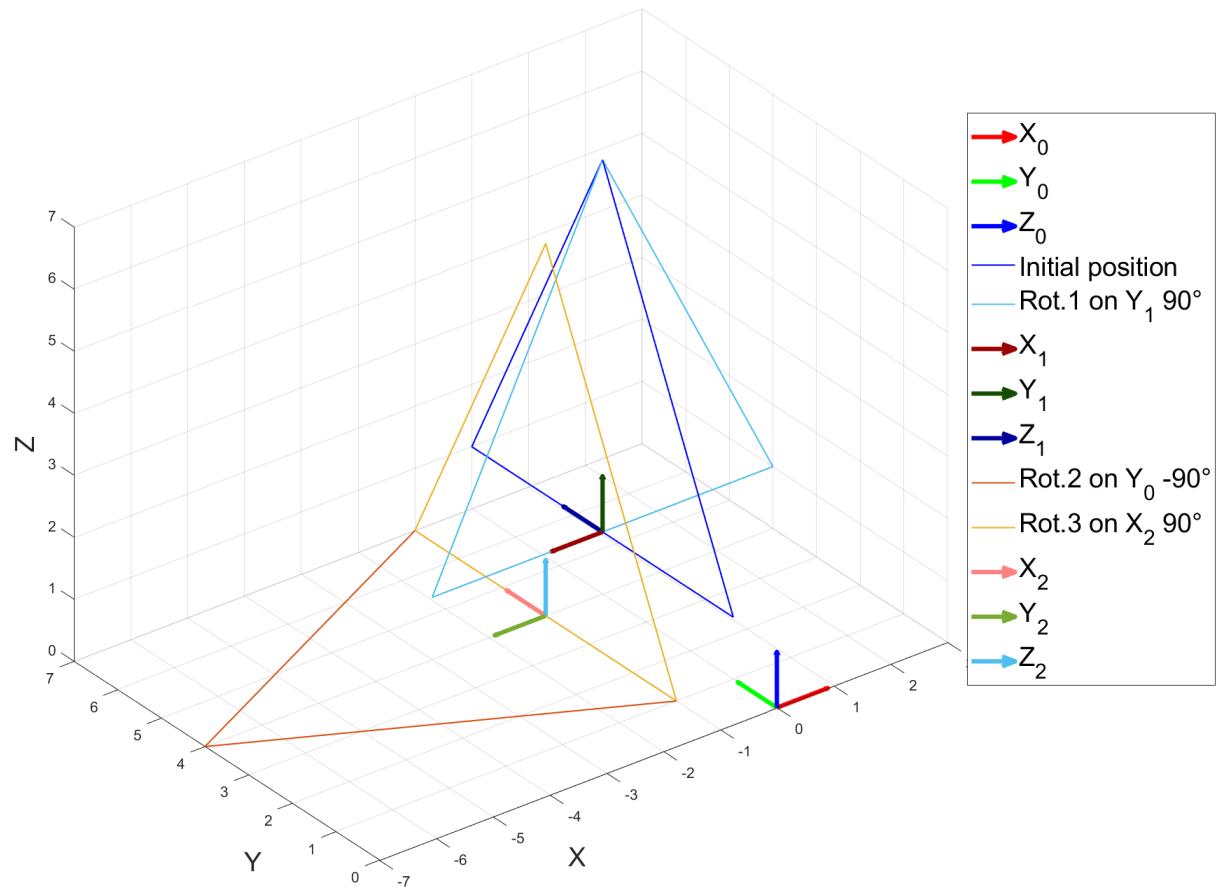


Figure 2.6: Final plot

# Report 3

## SCARA robot

### 3.1 Introduction

The object of this report is the SCARA robot shown in figure ??.

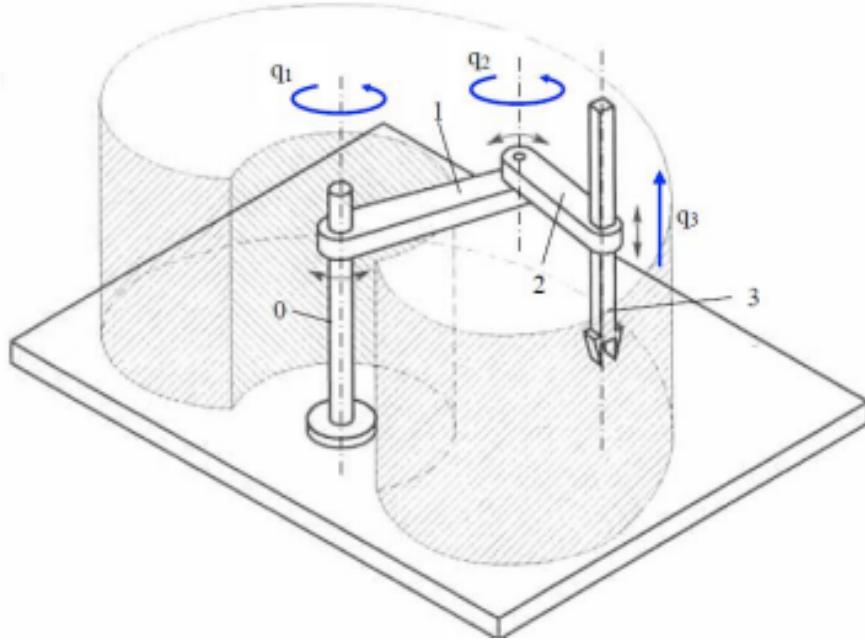


Figure 3.1: SCARA robot

The requirements of the problem are:

1. plot the wire frame  $\overline{O_0O_1AO_2BO_3}$  of the robot in the initial and final positions;
2. plot the trajectory of the end effector  $\overline{O_3}$ .

## 3.2 Problem illustration

A schematization of the problem is reported in figure ?? . The robot is made of three arms, with two rotary joints (1 and 2), one translatory joint (3) and an end effector.

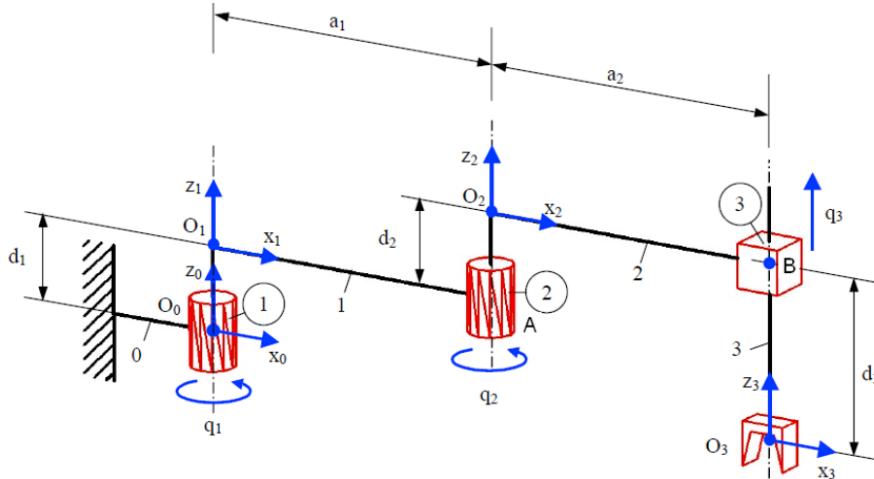


Figure 3.2: SCARA schematization

The dimensions of the robot are reported in table ??:

| Dimension | $d_1$ | $d_2$ | $d_3$ | $a_1$ | $a_2$ |
|-----------|-------|-------|-------|-------|-------|
| [mm]      | 25    | 15    | 10    | 50    | 10    |

Table 3.1: SCARA robot dimensions

The other provided data include the trajectory of each joint, organized into a matrix  $M$ , stored in the file joints.mat. In this matrix, the column vectors  $q_1$ ,  $q_2$  and  $q_3$  represent the displacements of each joint.

In addition to that, two Matlab functions called "denhar\_en01.m" and "joint\_rev\_01.m" are given. They respectively help in the generation of the orientation matrices and the creation of the joint representation in Matlab plots.

The Matlab script with the initial data is reported.

```

1 %% DATA
2 load("joints.mat")
3
4 P000 = [0 0 0] ';
5
6 q1 = M(:,1);    %[rad]
7 q2 = M(:,2);    %[rad]

```

```

8 q3 = M(:,3);    %[mm]
9
10 d1 = 25;      %[mm]
11 d2 = 15;      %[mm]
12 d3 = 10;      %[mm]
13
14 a1 = 50;      %[mm]
15 a2 = 10;      %[mm]

```

## 3.3 Implementation

### 3.3.1 Wire-Frame

The first step to solve the positioning problem is to evaluate the joints type, frames and displacements. The Denavit-Hartenberg convention has been used to define the position and orientation of the reference frames.

Considering the figure ??, the joints 1 and 2 are rotary type, so the displacement  $q_1$  and  $q_2$  are rotational and they are expressed in  $rad$ . The joint 3 is traslatory and its displacement  $q_3$  is expressed in  $mm$ .

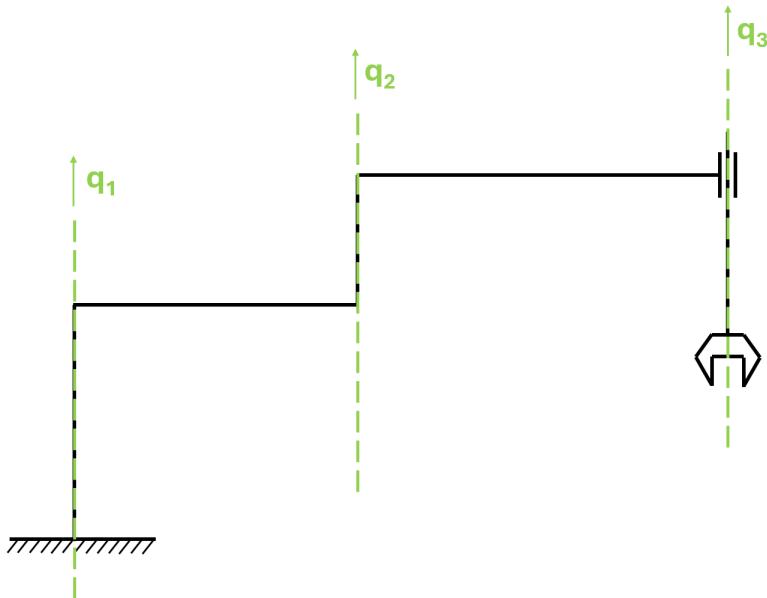


Figure 3.3: Joints schematization of the SCARA robot

The reference frames have been already defined as in figure ?? with the following considerations:

- For RF 1 and RF 2 the z axis are placed on the joint axis and orientated according to the displacement.
- For RF 1, considering that joint axis 1 and 2 are parallel and don't intersect, the x axis is placed along the common normal. The origin is located at the beginning of the body 1.
- For RF 2 the same considerations for RF 1 are done. The origin is placed at the beginning of body 2.
- In order to optimize the Matlab implementation, the origin of RF 0 is placed in the intersection of body 0 and joint 1 axis, the z axis according to  $q_1$  and x according to  $x_1$ .
- For RF 3 the origin is located in the end effector, with the z axis orientated as  $q_3$  and x axis coherent to  $x_2$ .

The schematization of the reference frames is reported in figure ??.

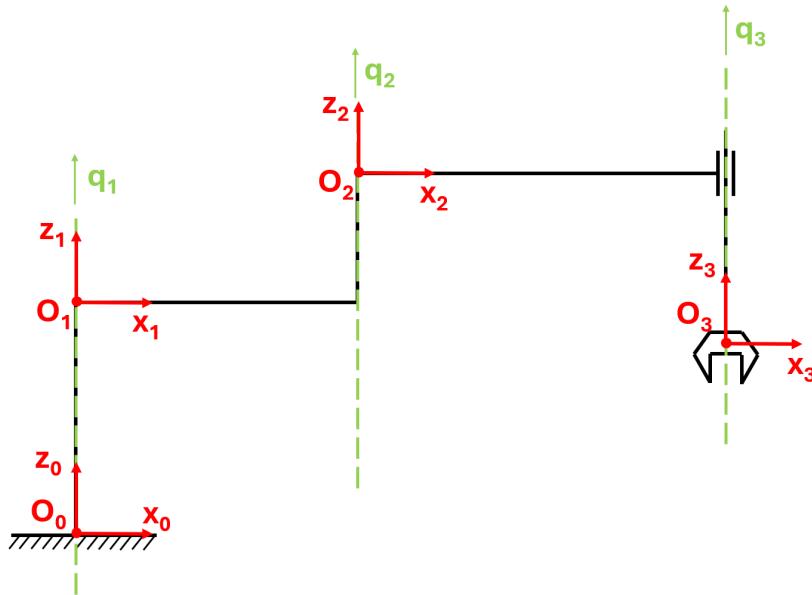


Figure 3.4: Reference systems adopted to schematize the SCARA robot

At this point a table with the angular and longitudinal dimensions is created, in order to compute the orientation matrices.

| body | $\alpha$ | $a$   | $d$          | $\theta$ | orientation matrix |
|------|----------|-------|--------------|----------|--------------------|
| 1    | 0        | 0     | $d_1$        | $q_1$    | ${}^0\hat{A}_1$    |
| 2    | 0        | $a_1$ | $d_2$        | $q_2$    | ${}^1\hat{A}_2$    |
| 3    | 0        | $a_2$ | $-d_3 + q_3$ | 0        | ${}^2\hat{A}_3$    |

Table 3.2: Denavit-Hartenberg coefficients

Using the function denhar\_en01 with input values the ones computed in table ??, the orientation matrices are obtained. The Matlab implementation is reported below.

```

1 %% POSITIONING PROBLEM
2
3 i=1; %initial position
4 %i=length(q1); %final position
5
6 A01o = denhar_en01(0,0,d1,q1(i));
7 A12o = denhar_en01(0,a1,d2,q2(i));
8 A23o = denhar_en01(0,a2,-d3+q3(i),0);

```

Using the orientation matrices previously calculated, the orientation matrices of every local reference frame are computed as follows:

$${}^0\hat{A}_2 = {}^0\hat{A}_1 \cdot {}^1\hat{A}_2 \quad {}^0\hat{A}_3 = {}^0\hat{A}_2 \cdot {}^2\hat{A}_3$$

Moreover, the position of the local reference frames  $P_{0i}$  with respect to the fixed reference frame are extrapolated from orientation matrices above computed.

$${}^{i-1}\hat{A}_i = \begin{bmatrix} {}^{i-1}A_i & P_{0i} \\ 0 & 1 \end{bmatrix}$$

This procedure is useful to define the coordinates of the wire frame segments. In addition, a fictitious orientation matrix is created to plot the joint symbol of body 3 in the correct position.

The Matlab code is implemented as following:

```

1 A02o = A01o * A12o;
2 A03o = A02o * A23o;
3
4 P001 = A01o(1:3,4);
5 P002 = A02o(1:3,4);
6 P003 = A03o(1:3,4);
7
8 wire_frame = [P000(1) P001(1) P002(1) P003(1) P003(1); ...
9 P000(2) P001(2) P002(2) P003(2) P003(2); ...
10 P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];
11
12 %fictitious matrix for body 3 joint
13 A03o_joint = A02o;

```

```
14 A03o_joint(1:2,4) = A03o(1:2,4);
```

The results are plotted by a Matlab code which use a parameter  $i$  to define the position of the robot. In particular, when  $i = 1$  the robot is in the initial position and when  $i = \text{length}(q1)$  it is in final position.

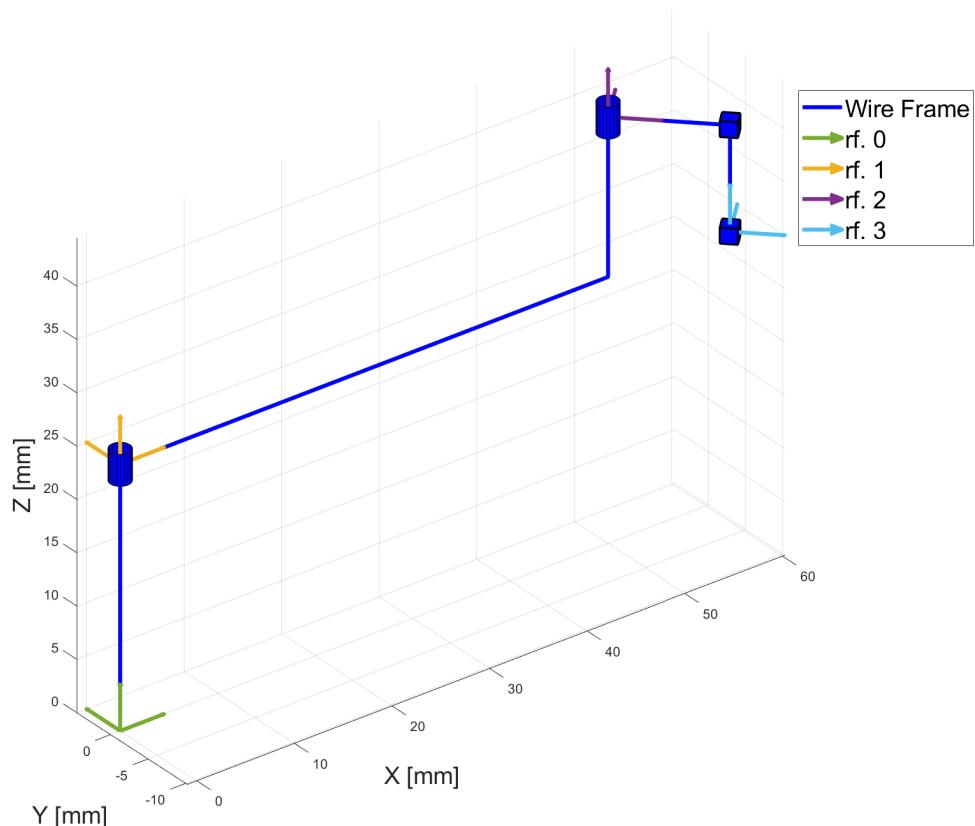


Figure 3.5: Plot of the wire frame in the initial position

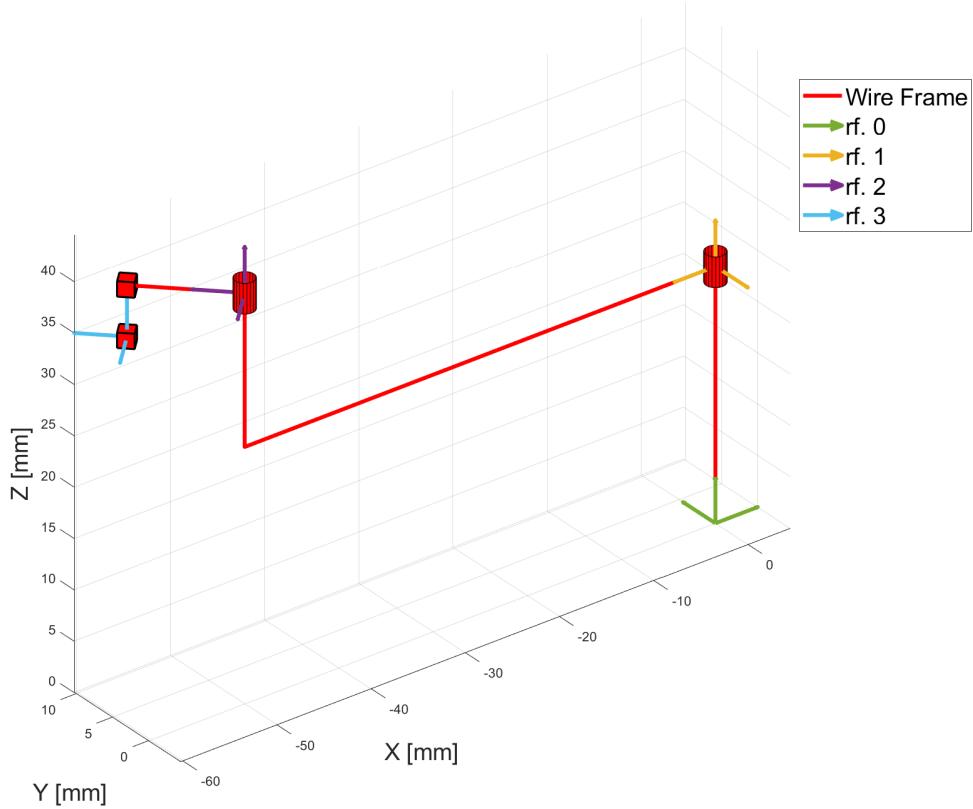


Figure 3.6: Plot of the wire frame in the final position

```

1 %Plot setup
2 radius = 1;
3 height_c = 3;
4 frame_lw = 3;
5 s = 5;
6 ref_lw = 3;
7 text_size = 18;
8 rf0_color = "#77AC30";
9 rf1_color = "#EDB120";
10 rf2_color = "#7E2F8E";
11 rf3_color = "#4DBEEE";
12 wf_color = "b";
13 %wf_color = "r";
14
15 figure(1)
16 %wire frame
17 wf = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),wf_color, ...
18 LineWidth=frame_lw);
19 hold on

```

```
20
21 %plot ref. 0
22 rf0_1 = quiver3(P000(1),P000(2),P000(3),1*s,0*s,0*s, ...
23             LineWidth=ref_lw,Color=rf0_color); %x0
24 hold on
25 rf0_2 = quiver3(P000(1),P000(2),P000(3),0*s,1*s,0*s, ...
26             LineWidth=ref_lw,Color=rf0_color); %y0
27 hold on
28 rf0_3 = quiver3(P000(1),P000(2),P000(3),0*s,0*s,1*s, ...
29             LineWidth=ref_lw,Color=rf0_color); %z0
30 hold on
31
32 %plot ref. 1
33 rf1_1 = quiver3(P001(1),P001(2),P001(3),...
34             A01o(1,1)*s,A01o(2,1)*s,A01o(3,1)*s, ...
35             LineWidth=ref_lw,Color=rf1_color); %x1
36 hold on
37 rf1_2 = quiver3(P001(1),P001(2),P001(3),...
38             A01o(1,2)*s,A01o(2,2)*s,A01o(3,2)*s, ...
39             LineWidth=ref_lw,Color=rf1_color); %y1
40 hold on
41 rf1_3 = quiver3(P001(1),P001(2),P001(3),...
42             A01o(1,3)*s,A01o(2,3)*s,A01o(3,3)*s, ...
43             LineWidth=ref_lw,Color=rf1_color); %z1
44 hold on
45
46 %plot ref. 2
47 rf2_1 = quiver3(P002(1),P002(2),P002(3),...
48             A02o(1,1)*s,A02o(2,1)*s,A02o(3,1)*s, ...
49             LineWidth=ref_lw,Color=rf2_color); %x2
50 hold on
51 rf2_2 = quiver3(P002(1),P002(2),P002(3),...
52             A02o(1,2)*s,A02o(2,2)*s,A02o(3,2)*s, ...
53             LineWidth=ref_lw,Color=rf2_color); %y2
54 hold on
55 rf2_3 = quiver3(P002(1),P002(2),P002(3),...
56             A02o(1,3)*s,A02o(2,3)*s,A02o(3,3)*s, ...
```

```

57     LineWidth=ref_lw,Color=rf2_color); %z2
58 hold on
59
60 %plot ref. 3
61 rf3_1 = quiver3(P003(1),P003(2),P003(3),...
62     A03o(1,1)*s,A03o(2,1)*s,A03o(3,1)*s,...
63     LineWidth=ref_lw,Color=rf3_color); %x3
64 hold on
65 rf3_2 = quiver3(P003(1),P003(2),P003(3),...
66     A03o(1,2)*s,A03o(2,2)*s,A03o(3,2)*s,...
67     LineWidth=ref_lw,Color=rf3_color); %y3
68 hold on
69 rf3_3 = quiver3(P003(1),P003(2),P003(3),...
70     A03o(1,3)*s,A03o(2,3)*s,A03o(3,3)*s,...
71     LineWidth=ref_lw,Color=rf3_color); %z3
72 hold on
73
74 %joints
75 joint_rev_01(radius,height_c,20,A01o,wf_color);
76 hold on
77 joint_rev_01(radius,height_c,20,A02o,wf_color);
78 hold on
79 cube(A03o_joint(1:3,4),radius+0.5,wf_color,A03o_joint);
80 hold on
81 cube(A03o(1:3,4),radius+0.5,wf_color,A03o);
82
83 %plot settings
84 grid on
85 axis equal
86 xlabel("X [mm]", "FontSize", text_size)
87 ylabel("Y [mm]", "FontSize", text_size)
88 zlabel("Z [mm]", "FontSize", text_size)
89 legend([wf rf0_1 rf1_1 rf2_1 rf3_1], ...
90 {"Wire Frame","rf. 0","rf. 1","rf. 2","rf. 3"}, font_size=text_size)

```

### 3.3.2 Trajectory of End-Effector

To compute the trajectory of the end effector, the previous procedure is iterated for all the element of the vectors  $q_i$  in order to compute the vector  $P_{O_3}$  instant by instant.

```

1 % TRAJECTORY
2 for j = 1:length(q1)
3     A01oj = denhar_en01(0,0,d1,q1(j));
4     A12oj = denhar_en01(0,a1,d2,q2(j));
5     A23oj = denhar_en01(0,a2,-d3+q3(j),0);
6     A03oj = A01oj * A12oj * A23oj;
7     if j==1
8         P303o_init = inv(A03oj)*A03oj(:,4);
9     end
10    P003j = A03oj * P303o_init;
11    traj_EF(:,j) = P003j;
12 end

```

So, the plot with final and initial position and end effector trajectory is now reported.

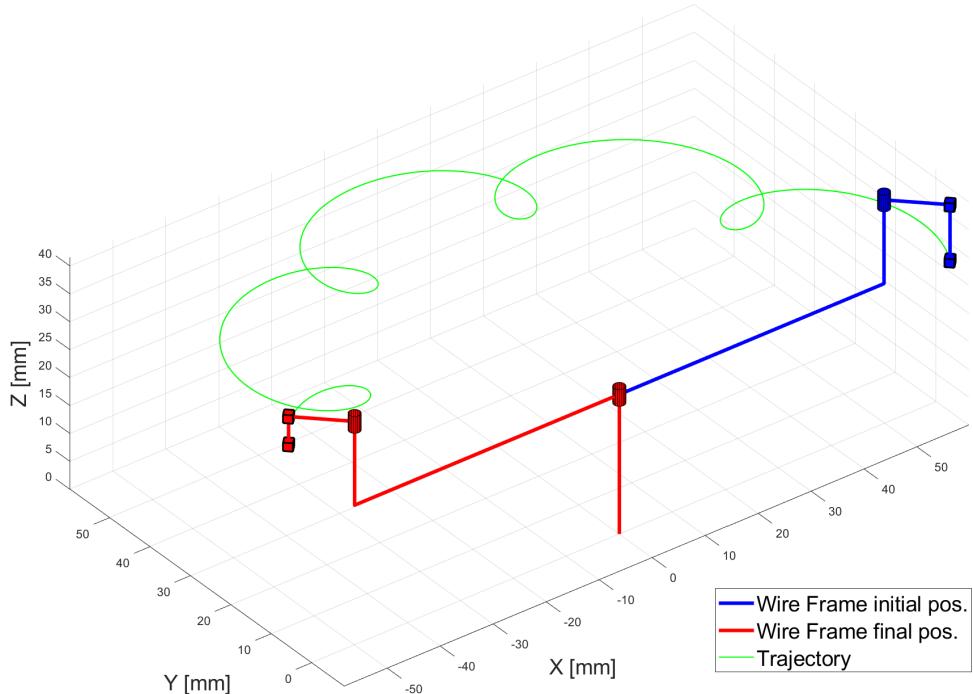


Figure 3.7: Trajectory of the end effector with the initial and final position of the SCARA robot

Below the Matlab implementation is reported.

```

1 %%TOTAL PLOT
2 i=1;
3
4 A01o = denhar_en01(0,0,d1,q1(i));
5 A12o = denhar_en01(0,a1,d2,q2(i));
6 A23o = denhar_en01(0,a2,-d3+q3(i),0);
7
8 A02o = A01o * A12o;
9 A03o = A02o * A23o;
10
11 P001 = A01o(1:3,4);
12 P002 = A02o(1:3,4);
13 P003 = A03o(1:3,4);
14
15 wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
16
17
18
19 A03o_joint = A03o;
20
21 A03o_joint(3,4) = A02o(3,4);
22
23 %Plot setup
24 radius = 1;
25 height_c = 3;
26 frame_lw = 3;
27 traj_lw = 1;
28
29 ref_lw = 3;
30 text_size = 18;
31 wf1_color = "b";
32 wf2_color = "r";
33 traj_color = "g";
34
35
36 figure(2)

```

```

37 %wire frame
38 wf1 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
39             wf1_color,LineWidth=frame_lw);
40 hold on
41
42 %joints
43 joint_rev_01(radius,height_c,20,A01o,wf1_color);
44 hold on
45 joint_rev_01(radius,height_c,20,A02o,wf1_color);
46 hold on
47 cube(A03o_joint(1:3,4),radius+0.5,wf1_color,A03o_joint);
48 hold on
49 cube(A03o(1:3,4),radius+0.5,wf1_color,A03o);
50 hold on
51
52 i=length(q1);
53
54 A01o = denhar_en01(0,0,d1,q1(i));
55 A12o = denhar_en01(0,a1,d2,q2(i));
56 A23o = denhar_en01(0,a2,-d3+q3(i),0);
57
58 A02o = A01o * A12o;
59 A03o = A02o * A23o;
60
61 P001 = A01o(1:3,4);
62 P002 = A02o(1:3,4);
63 P003 = A03o(1:3,4);
64
65 wire_frame = [P000(1) P001(1) P002(1) P003(1) P003(1);...
66                 P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
67                 P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];
68
69 A03o_joint = A03o;
70
71 A03o_joint(3,4) = A02o(3,4);
72
73 figure(2)

```

```
74 %wire frame
75 wf2 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
76     wf2_color,LineWidth=frame_lw);
77 hold on
78
79 %joints
80 joint_rev_01(radius,height_c,20,A01o,wf2_color);
81 hold on
82 joint_rev_01(radius,height_c,20,A02o,wf2_color);
83 hold on
84 cube(A03o_joint(1:3,4),radius+0.5,wf2_color,A03o_joint);
85 hold on
86 cube(A03o(1:3,4),radius+0.5,wf2_color,A03o);
87 hold on
88 traj = plot3(traj_EF(1,:),traj_EF(2,:),traj_EF(3,:),...
89     LineWidth=traj_lw,Color=traj_color);
90
91 %plot settings
92 grid on
93 axis equal
94 xlabel("X [mm]", "FontSize",text_size)
95 ylabel("Y [mm]", "FontSize",text_size)
96 zlabel("Z [mm]", "FontSize",text_size)
97 legend([wf1 wf2 traj], {"Wire Frame initial pos.",...
98     "Wire Frame final pos.", "Trajectory"}, ...
99     fontsize=text_size)
```

# Report 4

## 5 DoF robot

### 4.1 Introduction

A 5-axis robot (figure ??) with an articulated arm (3 d.o.f.) and a wrist (2 d.o.f.) is considered.

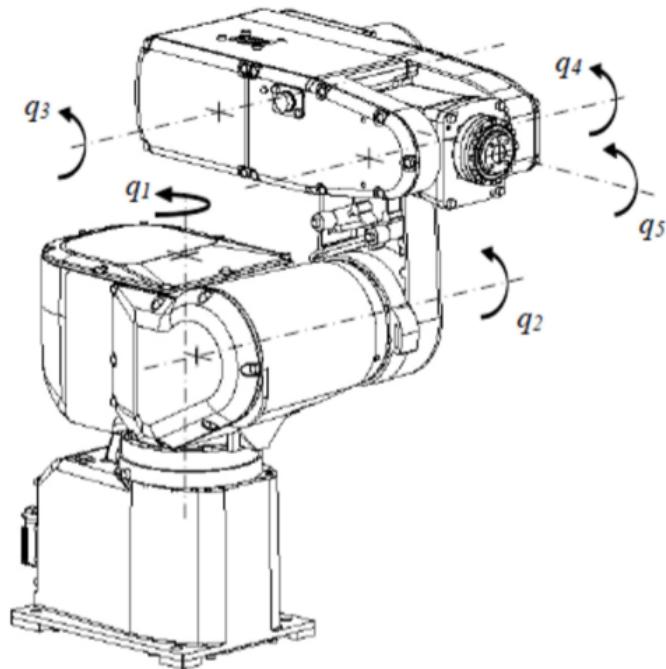


Figure 4.1: 5 DoF robot

This report is divided in 2 exercises.

1. Kinematics study of the robot in figure ??.
2. Dynamics study of the robot in figure ?? and its payload shown in figure ??.

Some tools are given:

- The Matlab functions denhar\_en01.m, dynam\_en02.m, kinem\_en02.m which respectively calculate the orientation matrices, the dynamic and the kinematic equations.
- The file trajectory1.mat which is table with the angular displacements, velocities and accelerations ( $q_i$ ,  $\dot{q}_i$  and  $\ddot{q}_i$ ) for every joint.

The requirements of the first problem are:

1. Define the reference frames for each body according to DH convention and represent them on both views of figure ??.
2. Load the joints parameters from file trajectory1.mat and plot with Matlab in figure 1 the angles in joints, in figure 2 the speeds and in figure 3 the accelerations.
3. Plot with Matlab in figure 4 the wire frame of the robot in the initial and final positions.
4. Represent in the same figure the trajectory of the wrist center point CP and of the end effector EE.
5. Verify the correctness of kinematic equations in file kinem\_en02.m.
6. Calculate the speeds and accelerations of the centers of mass for frame 5 using function kinem\_en02.m and plot them in figures 5 (speeds) and 6 (accelerations).

The requirements of the second problem are:

1. Compute the inertial matrixes of the bodies and payload.
2. Solve the inverse dynamics using function dynam.m to calculate the actuator torques  $\tau_1$ ,  $\tau_2$ ,  $\tau_3$ ,  $\tau_4$ ,  $\tau_5$  in the joints during the movement.
3. Represent the forces of constraint exerted by base 0 to floor during the movement.

In figure ?? a schematization of the 5 DoF robot and in figure ?? a schematization of the payload attached to the wrist of the robot are reported.

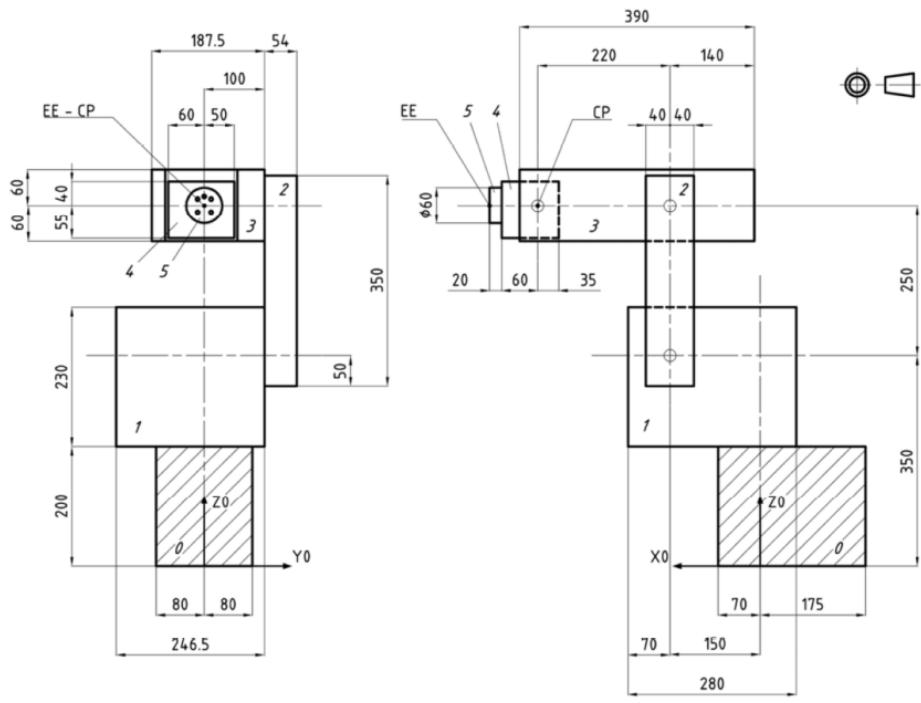


Figure 4.2: 5 DoF robot schematization

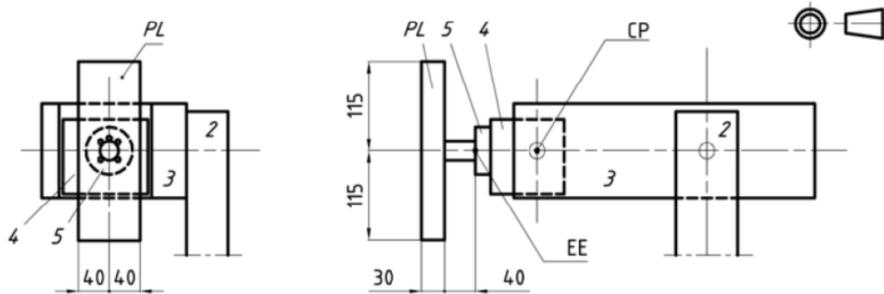


Figure 4.3: Payload schematization

## 4.2 Problem illustration

The following hypothesis are assumed:

1 / 4 - The bodies from 1 to 4 are taken equivalent to a parallelepiped with homogeneous distributed mass.

5 - The body 5 is taken equivalent to a cylinder with homogeneous distributed mass.

CP - The wrist center point is the intersection of axes 4 and 5.

EE - The End effector center point is at the end of body 5, in the center of the wrist interface plate.

PL - A payload is attached to the wrist interface plate as figure 3 shows. To define its inertial characteristics the cylindrical portion is neglected and the payload is taken equivalent to a parallelepiped with homogeneous distributed mass equal to  $mL = 1 \text{ kg}$  and dimensions as in figure 3.

The mass of the bodies is reported in table ??:

| <b>Body</b>      | 0  | 1    | 2 | 3 | 4 | 5   | PL |
|------------------|----|------|---|---|---|-----|----|
| <b>Mass [kg]</b> | 18 | 10.5 | 2 | 6 | 2 | 0.5 | 1  |

*Table 4.1: Masses of bodies*

The Matlab code where the dimensions, the trajectory data and the position vectors of the centers of mass are reported is the following one.

```

1 %% DATA
2
3 data = load("trajectory.mat");
4
5 a = 200;
6 b = 150;
7 c = 150;
8 d = 100;
9 e = 250;
10 f = 100;
11 g = 220;
12 h = 60;
13 i = 20;
14 l = 40;
15
16 q1 = data.q1vec;
17 q2 = data.q2vec;
18 q3 = data.q3vec;
19 q4 = data.q4vec;
20 q5 = data.q5vec;
21
22 q1d = data.q1dvec.* (pi/180);
23 q2d = data.q2dvec.* (pi/180);
24 q3d = data.q3dvec.* (pi/180);
25 q4d = data.q4dvec.* (pi/180);
26 q5d = data.q5dvec.* (pi/180);
27
28 q1dd = data.q1ddvec.* (pi/180);
```

```

29 q2dd = data.q2ddvec.*((pi/180);
30 q3dd = data.q3ddvec.*((pi/180);
31 q4dd = data.q4ddvec.*((pi/180);
32 q5dd = data.q5ddvec.*((pi/180);

33
34 time = data.timevec;
35
36 qPd = zeros(length(time),1);
37 qPdd = zeros(length(time),1);
38
39 CM0_0 = [-140 0 100]';
40 CM1_1 = [80 0 35]';
41 CM2_2 = [125 0 -27]';
42 CM3_3 = [55 0 100]';
43 CM4_4 = [-7.5 -12.5 5]';
44 CM5_5 = [0 0 10]';
45 CMP_P = [0 0 15]';

46
47 m0 = 18;
48 m1 = 10.5;
49 m2 = 2;
50 m3 = 6;
51 m4 = 2;
52 m5 = 0.5;
53 mP = 1;
54
55 delta = zeros(6,1);

```

## 4.3 Implementation - Exercise 1

### 4.3.1 Application of DH convention

The Denavit-Hartenberg convention is applied, as requested, to define the reference system of each body. Following the procedure the joints, in this case rotational, are defined and z-axis of each body are located along them with concordant direction as shown in figure ??.

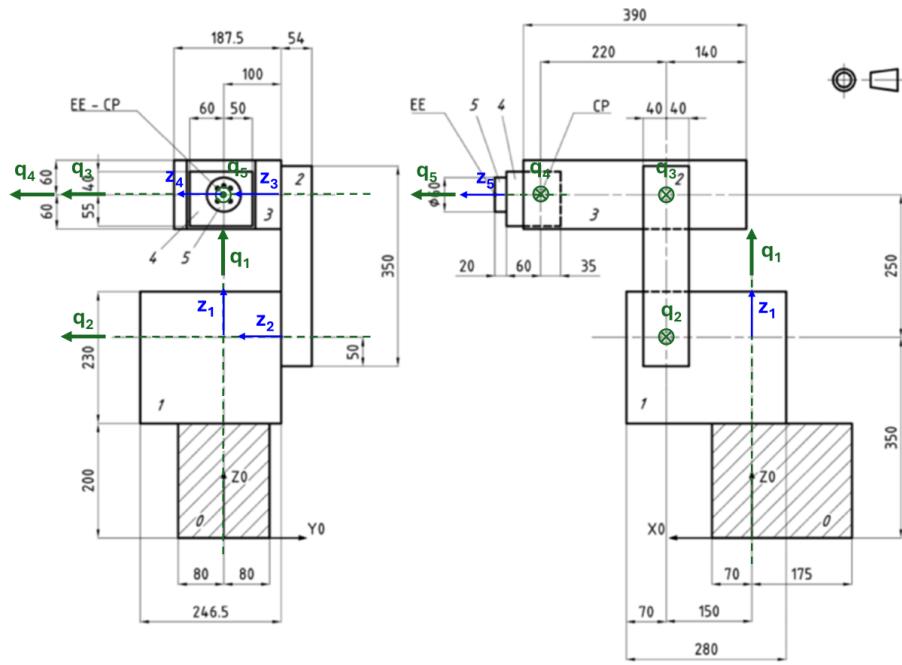


Figure 4.4: Joints and z axis according to Denavit-Hartenberg convention

At this point the origin and the x axis for each joint reference system is defined, the procedure is reported below:

**Body 1** Axis of joint 1 and 2 are taken into account. Considering that they are skew, the x axis is placed along the common normal in direction of joint 2. The origin is positioned in the intersection of the common normal and joint axis.

**Body 2** Axis of joint 2 and 3 are parallel, therefore the origin is located at the beginning of body 2 and the x axis along the common normal that pass through the origin, in direction of body 3.

**Body 3** As described for body 2, the axis of joint 3 and 4 are parallel. Therefore the origin is located in the intersection between axis of joint 5 and 3. The x axis is placed along the common normal that pass through the origin, in direction of body 4.

**Body 4** Joints 4 and 5 intersect in a point where the origin is placed. For ease the x axis is defined parallel and concordant with  $z_0$

For bodies 0 and 5 the use of the Denavit-Hartenberg convention is recommended but not mandatory, therefore:

**Body 0** The reference system is defined by the text

**Body 5** The Denavit-Hartenberg convention is used. z axis is placed along joint 5 axis and concordant to it and x axis parallel and concordant to  $x_4$ . The origin is located in the begin of body 5.

Moreover, an additional reference system is defined for end effector point (EE) and payload (PL) (defined in paragraph ??) parallel to  $o_5x_5y_5z_5$ . At this point, a schematization of bodies reference system can be made and it is report in the following figure.

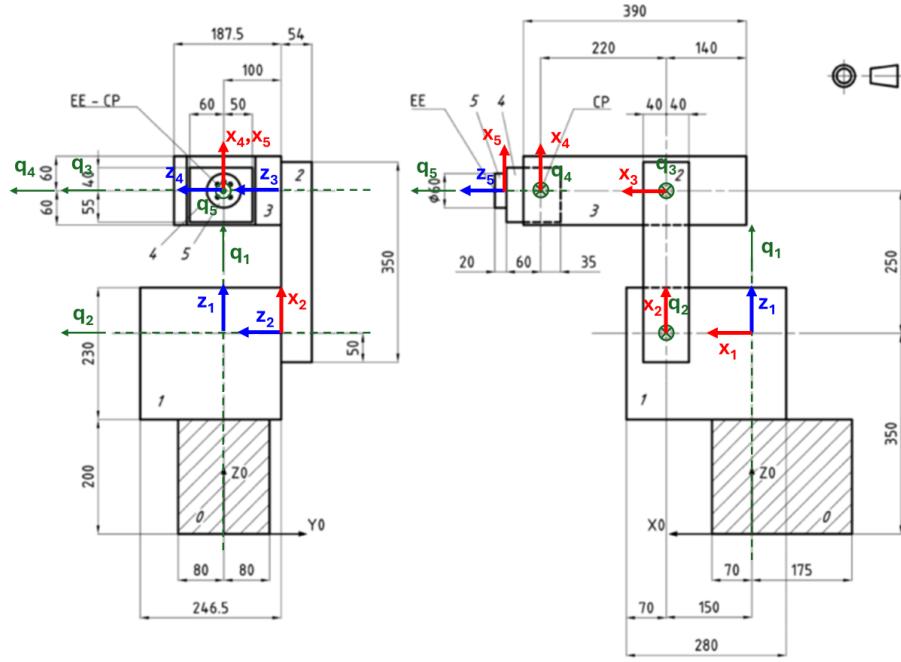


Figure 4.5: Denavit-Hartenberg convention applied in the 5 D.O.F. robot

For more clarity, a 3D sketch of wire frame with bodies reference system is provided.

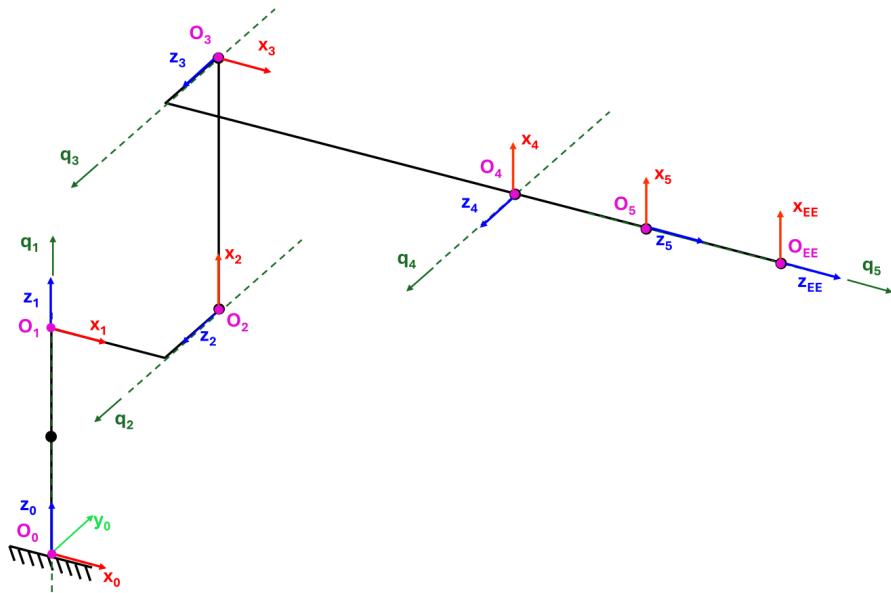


Figure 4.6: 3D sketch of wire frame with bodies reference system

At this point is possible to define the table (??) reporting the Denavit-Hartenberg coefficients useful to define the orientation matrices  $i^{-1}A_i$ .

| body | <i>alpha</i> | <i>a</i> | <i>d</i> | <i>theta</i>      | orientation matrix |
|------|--------------|----------|----------|-------------------|--------------------|
| 1    | 0            | 0        | $a + b$  | $q_1$             | ${}^0\hat{A}_1$    |
| 2    | $90^\circ$   | $c$      | $-d$     | $90^\circ + q_2$  | ${}^1\hat{A}_2$    |
| 3    | 0            | $e$      | 0        | $-90^\circ + q_3$ | ${}^2\hat{A}_3$    |
| 4    | 0            | $g$      | $f$      | $90^\circ + q_4$  | ${}^3\hat{A}_4$    |
| 5    | $90^\circ$   | 0        | $h$      | $q_5$             | ${}^4\hat{A}_5$    |
| EE   | 0            | 0        | $i$      | 0                 | ${}^5\hat{A}_{EE}$ |
| PL   | 0            | 0        | $l + i$  | 0                 | ${}^2\hat{A}_{PL}$ |

Table 4.2: Denavit-Hartenberg coefficients

The distances reported in table ?? are referred as shown in the following figure:

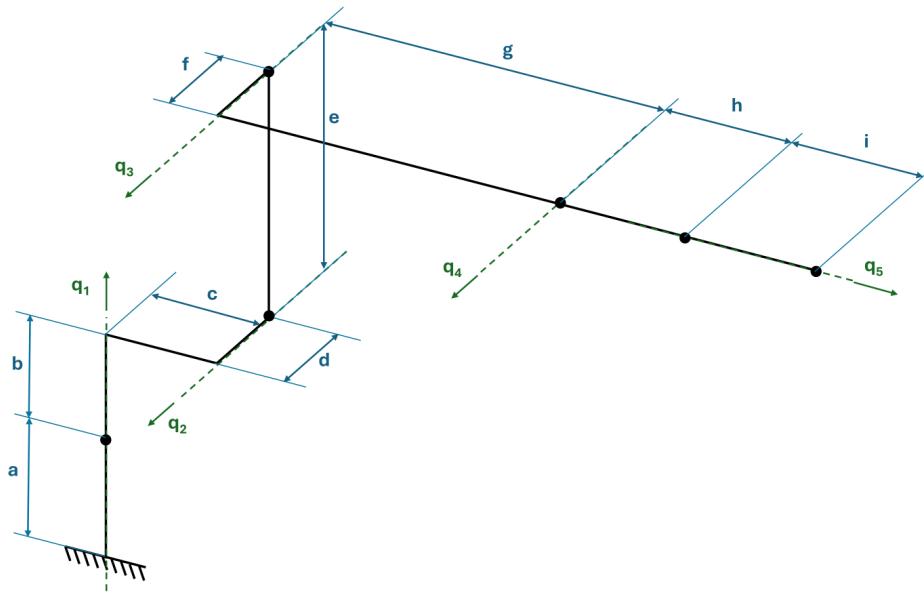


Figure 4.7: Distances used in the Denavit-Hartenberg convention

|                      | <i>a</i> | <i>b</i> | <i>c</i> | <i>d</i> | <i>e</i> | <i>f</i> | <i>g</i> | <i>h</i> | <i>i</i> | <i>l</i> |
|----------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>Distance [mm]</b> | 200      | 150      | 150      | 100      | 250      | 100      | 220      | 60       | 20       | 40       |

Table 4.3: Value of the Denavit-Hartenberg convention

Furthermore, the orientation matrices are been referred to the reference system  $O_0x_0y_0z_0$

$${}^0\hat{A}_i = {}^0\hat{A}_{i-1} \cdot {}^{i-1}\hat{A}_i \quad (4.1)$$

The positioning problem is solved on Matlab using the function denhar\_en01.m and the result is shown in the following script.

```

1 for j = 1:length(time)
2
3
4 % POSITIONING PROBLEM
5
6 %Denavit-Huntberg convention
7 A01o = denhar_en01( 0,      0,      a+b,      deg2rad(q1(j)));
8 A12o = denhar_en01(pi/2,    c,      -d,      pi/2+deg2rad(q2(j)));
9 A23o = denhar_en01( 0,      e,      0,      -pi/2+deg2rad(q3(j)));
10 A34o = denhar_en01( 0,      g,      f,      pi/2+deg2rad(q4(j)));
11 A45o = denhar_en01(pi/2,    0,      h,      deg2rad(q5(j)));
12 A5EEo = denhar_en01( 0,      0,      i,      0);
13 A5Po = denhar_en01( 0,      0,      l+i,     0);
14
15 A02o = A01o * A12o;
16 A03o = A02o * A23o;
17 A04o = A03o * A34o;
18 A05o = A04o * A45o;
19 A0EEo = A05o * A5EEo;
20 AOPo = A05o * A5Po;
21
22
23 %wire frame in the initial position
24 if j == 1
25     wf_init = [0 A01o(1,4) A01o(1,4) A01o(1,4)+c A02o(1,4)...
26                     A03o(1,4) A03o(1,4) A04o(1,4) A05o(1,4) A0EEo(1,4);
27                     0 A01o(2,4) A01o(2,4) A01o(2,4) A02o(2,4)...
28                     A03o(2,4) A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
29                     0 A01o(3,4)-b A01o(3,4) A01o(3,4) A02o(3,4)...
30                     A03o(3,4) A03o(3,4) A04o(3,4) A05o(3,4) A0EEo(3,4)];
31
32     A_joint1 = A01o;
33     A_joint1(3,4) = A01o(3,4)-b;
34
35     initial_cond = {wf_init,A_joint1,A02o,A03o,A04o,A05o,A0EEo}';
36 end
37
38 %wire frame in the final position
39 if j == length(time)
40     wf_fin = [0 A01o(1,4) A01o(1,4) A01o(1,4)+c A02o(1,4)...
41                     A03o(1,4) A03o(1,4) A04o(1,4) A05o(1,4) A0EEo(1,4);
42                     0 A01o(2,4) A01o(2,4) A01o(2,4) A02o(2,4)...
43                     A03o(2,4) A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
44                     0 A01o(3,4)-b A01o(3,4) A01o(3,4) A02o(3,4)...

```

```

45      A03o(3,4) A03o(3,4)    A04o(3,4) A05o(3,4) A0EEo(3,4)] ;
46
47      A_joint1 = A01o;
48      A_joint1(3,4) = A01o(3,4)-b;
49
50      final_cond = {wf_fin,A_joint1,A02o,A03o,A04o,A05o,A0EEo}' ;
51  end

```

### 4.3.2 Plot of angles, speeds and accelerations

In figure ??, ?? and ?? respectively the angles in joints, the angular speeds and the accelerations are plotted in function of time.

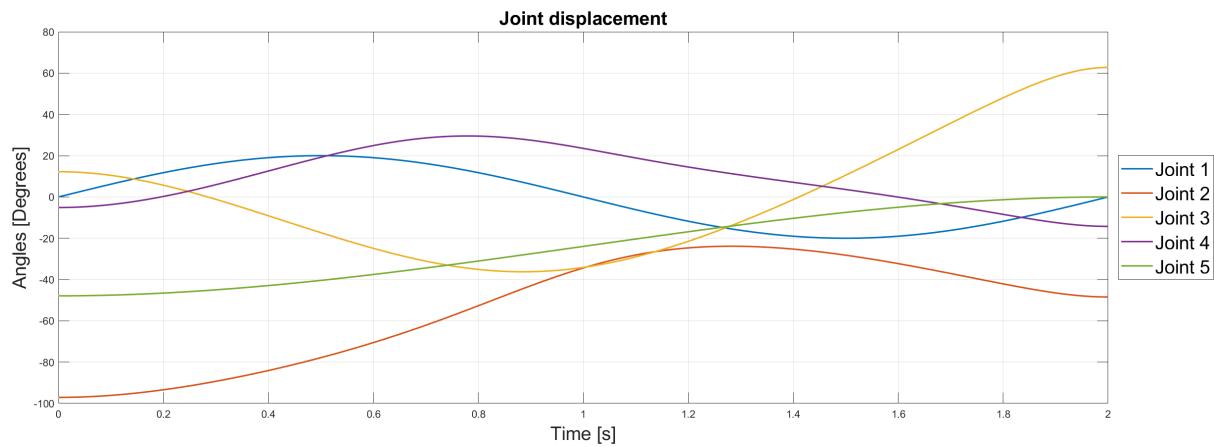


Figure 4.8: Joint angles plot

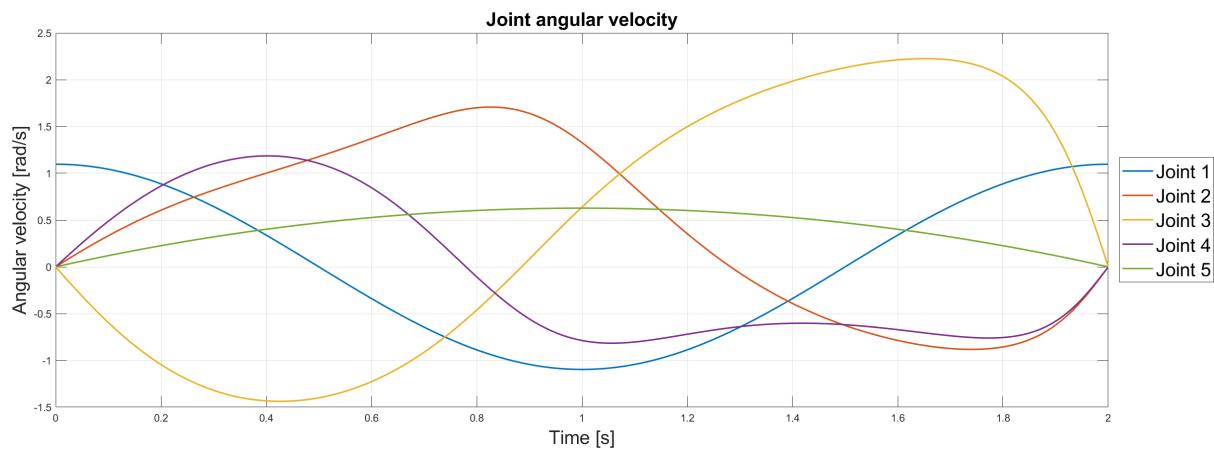


Figure 4.9: Joint angular velocities plot

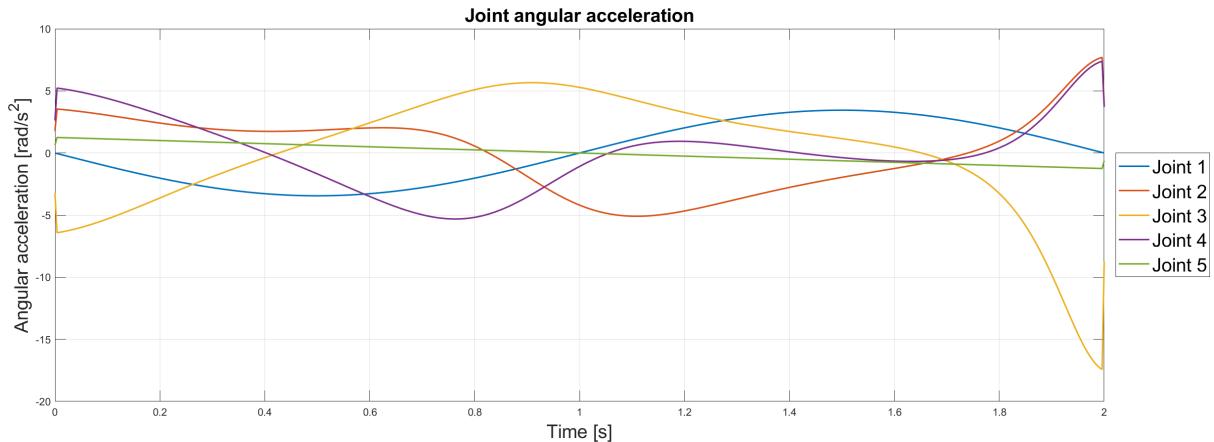


Figure 4.10: Joint angular accelerations plot

The Matlab script to obtain them is reported below.

```

1 %requirement 2 - ang,vel,acc plot
2 figure(3)
3 plot(time,q1,time,q2,time,q3,time,q4,time,q5,LineWidth=kin_lw)
4 hold on
5 grid on
6 title("Joint displacement",FontSize=fs)
7 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
8 "eastoutside",fontsize=fs)
9 xlabel("Time [s]",FontSize=fs)
10 ylabel("Angles [Degrees]",FontSize=fs)
11
12 figure(4)
13 plot(time,q1d,time,q2d,time,q3d,time,q4d,time,q5d,LineWidth=kin_lw)
14 grid on
15 title("Joint angular velocity",FontSize=fs)
16 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
17 "eastoutside",fontsize=fs)
18 xlabel("Time [s]",FontSize=fs)
19 ylabel("Angular velocity [rad/s]",FontSize=fs)
20
21 figure(5)
22 plot(time,q1dd,time,q2dd,time,q3dd,time,q4dd,time,q5dd,LineWidth=kin_lw)
23 grid on
24 title("Joint angular acceleration",FontSize=fs)
25 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
26 "eastoutside",fontsize=fs)
27 xlabel("Time [s]",FontSize=fs)
28 ylabel("Angular acceleration [rad/s^2]",FontSize=fs)

```

### 4.3.3 Wire frame in initial and final position

In figure ?? the plot of the wire frame in its initial and final position is reported.

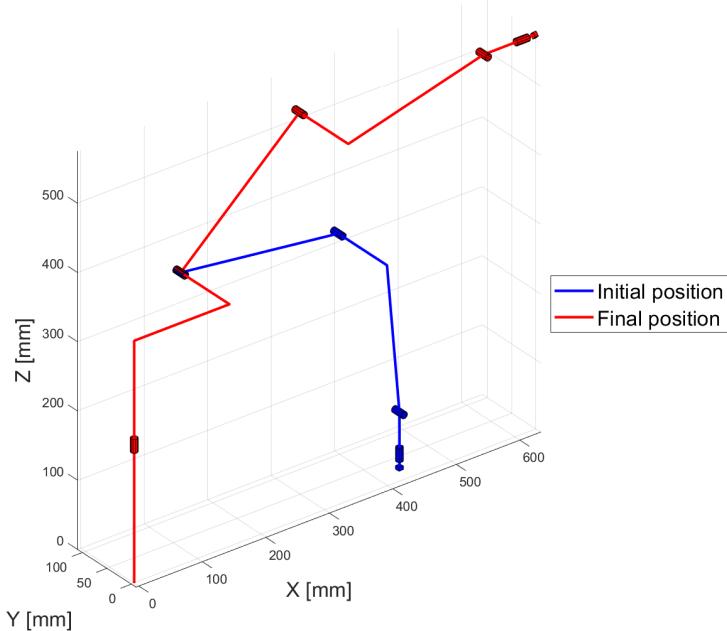


Figure 4.11: Wire frame (initial and final position)

The Matlab script to implement it is reported below.

```

1 %% Requirement 3 - PLOT POSITIONS AND TRAJECTORY
2
3 figure(2)
4 wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),
5 Color="b",LineWidth=wf_lw); %Initial position
6 hold on
7 wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),
8 Color="r",LineWidth=wf_lw); %Final position
9 hold on
10
11 t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
12 hold on
13 t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory
14
15 joint_rev_01(r_j,l_j,10,initial_cond{2}, "b");
16 joint_rev_01(r_j,l_j,10,initial_cond{3}, "b");
17 joint_rev_01(r_j,l_j,10,initial_cond{4}, "b");
18 joint_rev_01(r_j,l_j,10,initial_cond{5}, "b");
19 joint_rev_01(r_j,l_j,10,initial_cond{6}, "b");
20 joint_rev_01(r_j,r_j,4,initial_cond{7}, "b");
21
```

```

22 joint_rev_01(r_j,l_j,10,final_cond{2}, "r");
23 joint_rev_01(r_j,l_j,10,final_cond{3}, "r");
24 joint_rev_01(r_j,l_j,10,final_cond{4}, "r");
25 joint_rev_01(r_j,l_j,10,final_cond{5}, "r");
26 joint_rev_01(r_j,l_j,10,final_cond{6}, "r");
27 joint_rev_01(r_j,r_j,4,final_cond{7}, "r");

28
29 legend([wf_plot(1) wf_plot(2) t1 t2], {"Initial position", "Final position", ...
30 "EE Trajectory", "CP Trajectory"}, fontsize=fs)
31 xlabel("X [mm]", "FontSize", fs)
32 ylabel("Y [mm]", "FontSize", fs)
33 zlabel("Z [mm]", "FontSize", fs)
34 axis equal
35 grid on

```

#### 4.3.4 Trajectory of point CP and EE

The point CP is the wrist center point and EE is the end-effector, both visible in figure ???. In figure ?? the trajectory of this two points is plotted.

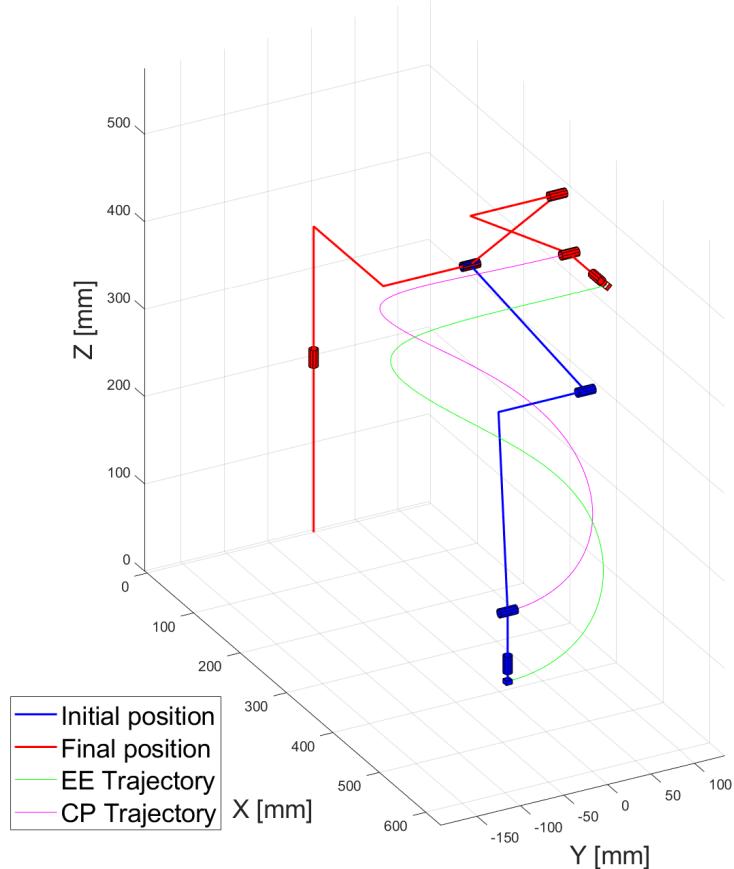


Figure 4.12: CP and EE trajectory plot

The Matlab implementation is reported.

```

1 %% Requirement 4 - Trajectory of CP and EE
2 for j = 1:length(time)
3
4     %trajectory EE
5     traj_EE(:,j) = AOEEo(1:3,4);
6
7     %trajectory EE
8     traj_CP(:,j) = A04o(1:3,4);
9
10 end
11
12 figure(2)
13 wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),
14 Color="b",LineWidth=wf_lw); %Initial position
15 hold on
16 wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),
17 Color="r",LineWidth=wf_lw); %Final position
18 hold on
19
20 t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
21 hold on
22 t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory
23
24 joint_rev_01(r_j,l_j,10,initial_cond{2}, "b");
25 joint_rev_01(r_j,l_j,10,initial_cond{3}, "b");
26 joint_rev_01(r_j,l_j,10,initial_cond{4}, "b");
27 joint_rev_01(r_j,l_j,10,initial_cond{5}, "b");
28 joint_rev_01(r_j,l_j,10,initial_cond{6}, "b");
29 joint_rev_01(r_j,r_j,4,initial_cond{7}, "b");
30
31 joint_rev_01(r_j,l_j,10,final_cond{2}, "r");
32 joint_rev_01(r_j,l_j,10,final_cond{3}, "r");
33 joint_rev_01(r_j,l_j,10,final_cond{4}, "r");
34 joint_rev_01(r_j,l_j,10,final_cond{5}, "r");
35 joint_rev_01(r_j,l_j,10,final_cond{6}, "r");
36 joint_rev_01(r_j,r_j,4,final_cond{7}, "r");
37
38 legend([wf_plot(1) wf_plot(2) t1 t2], {"Initial position", "Final position", ...
39 "EE Trajectory", "CP Trajectory"}, fontsize=fs)
40 xlabel("X [mm]", "FontSize", fs)
41 ylabel("Y [mm]", "FontSize", fs)
42 zlabel("Z [mm]", "FontSize", fs)
43 axis equal

```

44 grid on)

### 4.3.5 Verify the kinematic function

The code of the Matlab function kinem\_en02.m is reported below.

```

1 % kinem_en02.m
2 % Function for recursive forward computation of kinematic variables of link i
3 % in terms of kinematic variables of link (i-1) and of d.o.f. in joint i.
4 %
5 % input variables:
6 % wim1 = angular velocity of link (i-1) expressed in frame (i-1)
7 % wim1p = angular acceleration of link (i-1) expressed in frame (i-1)
8 % vim1 = linear velocity of the origin of link (i-1) expressed in frame (i-1)
9 % vim1p = linear acceleration of the origin of link (i-1) expressed in frame (i-1)
10 % ki = unit vector k of joint i axis expressed in frame i
11 % lim1 = position vector of the origin i with respect to frame (i-1) expressed
12 %        in frame (i-1)
13 % bi = position vector of center of mass of link i with respect to frame i
14 %        expressed in frame i
15 % deltai = joint i parameter (1/0)
16 % qip = joint i d.o.f. velocity
17 % qipp = joint i d.o.f. acceleration
18 % iAim1 = rotation matrix (i)A(i-1) of frame (i-1) with respect to frame i
19 %
20 % Output variables:
21 % wi = angular velocity of link i expressed in frame i
22 % wip = angular acceleration of link i expressed in frame i
23 % vi = linear velocity of the origin i expressed in frame i
24 % vip = linear acceleration of the origin i expressed in frame i
25 % vGip = linear acceleration of the center of mass of link i expressed in frame i
26
27
28 function [wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,%
29 ki,lim1,bi,deltai,qip,qipp,iAim1)
30
31 %computation of angular velocity omega(i)
32 wi = iAim1*wim1+qip*(1-deltai)*ki;
33
34 %computation of linear velocity v(i) of origin Oi in frame i
35 vi=iAim1*vim1+iAim1*cross(wim1,lim1)+qip*deltai*ki;
36
37 %computation of linear velocity vG(i) of centre of mass in frame i
38 vGi=vi+cross(wi,bi);

```

```

39 %computation of angular acceleration omegadot(i)
40 wip = iAim1*wim1p+qipp*(1-deltai)*ki+qip*(1-deltai)*...
41     cross(iAim1*wim1,ki);
42
43 %computation of linear acceleration vdot(i)
44 vip = iAim1*(vim1p+cross(wim1p,lim1)+cross(wim1,cross(wim1,lim1)))+...
45     qipp.*deltai*ki+2*qip*deltai*cross(iAim1*wim1,ki);
46
47 %computation of linear acceleration vGdot(i)
48 vGip = vip+cross(wip,bi)+cross(wi,cross(wi,bi));
49
50

```

The given formulas in theory handouts are considered, in order to make a comparison with the formulas written in the function script.

$${}^i\vec{\omega}_i = {}^i\vec{\omega}_{i-1} + \dot{q}_i(1 - \delta_i) \cdot {}^i\vec{k}_i \quad (4.2)$$

$${}^i\vec{v}_i = {}^i\vec{v}_{i-1} + {}^i\vec{\omega}_{i-1} \times {}^i\vec{l}_{i-1} + \dot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i \quad (4.3)$$

$${}^i\vec{v}_{G,i} = {}^i\vec{v}_i + {}^i\vec{\omega}_i \times {}^i\vec{b}_i \quad (4.4)$$

$$\dot{{}^i\vec{\omega}}_i = {}^i\dot{\vec{\omega}}_{i-1} + \ddot{q}_i \cdot (1 - \delta_i) \cdot {}^i\vec{k}_i + \dot{q}_i \cdot (1 - \delta_i) \cdot {}^i\vec{\omega}_{i-1} \times {}^i\vec{k}_i \quad (4.5)$$

$$\dot{\vec{v}}_i = {}^i\dot{\vec{v}}_{i-1} + {}^i\dot{\vec{\omega}}_{i-1} \times {}^i\vec{l}_{i-1} + {}^i\vec{\omega}_{i-1} \times [{}^i\vec{\omega}_{i-1} \times {}^i\vec{l}_{i-1}] + \ddot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i + 2 \cdot \dot{q}_i \cdot \delta_i \cdot {}^i\vec{\omega}_{i-1} \times {}^i\vec{k}_i \quad (4.6)$$

$$\dot{\vec{v}}_{G,i} = \dot{\vec{v}}_i + {}^i\vec{\omega}_i \times b_i + {}^i\vec{\omega}_i \times ({}^i\vec{\omega}_i \times b_i) \quad (4.7)$$

Function inputs as  $\vec{\omega}_{i-1}$ ,  $\dot{\vec{\omega}}_{i-1}$ ,  $\vec{v}_{i-1}$ ,  $\dot{\vec{v}}_{i-1}$ ,  $\vec{l}_{i-1}$  are linked to reference system  $i - 1$  but, to perform vectorial product is necessary that all vectors are referred to the same reference system. Using the rotation matrix  ${}^iA_{i-1}$ , provided as input, all vectors are linked to reference system  $i$ .

$${}^i\vec{\omega}_i = {}^iA_{i-1} \cdot {}^{i-1}\vec{\omega}_{i-1} + \dot{q}_i(1 - \delta_i) \cdot {}^i\vec{k}_i \quad (4.8)$$

$${}^i\vec{v}_i = {}^iA_{i-1} \cdot {}^{i-1}\vec{v}_{i-1} + {}^iA_{i-1} \cdot ({}^{i-1}\vec{\omega}_{i-1} \times {}^{i-1}\vec{l}_{i-1}) + \dot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i \quad (4.9)$$

$${}^i\dot{\vec{\omega}}_i = {}^iA_{i-1} \cdot {}^{i-1}\dot{\vec{\omega}}_{i-1} + \ddot{q}_i \cdot (1 - \delta_i) \cdot {}^i\vec{k}_i + \dot{q}_i \cdot (1 - \delta_i) \cdot ({}^iA_{i-1} \cdot {}^{i-1}\vec{\omega}_{i-1}) \times {}^i\vec{k}_i \quad (4.10)$$

$$\begin{aligned} \dot{\vec{v}}_i = & {}^iA_{i-1} \cdot ({}^{i-1}\dot{\vec{v}}_{i-1} + {}^{i-1}\dot{\vec{\omega}}_{i-1} \times {}^{i-1}l_{i-1} + {}^{i-1}\vec{\omega}_{i-1} \times ({}^{i-1}\vec{\omega}_{i-1} \times {}^{i-1}l_{i-1})) + \\ & + \ddot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i + 2 \cdot \dot{q}_i \cdot \delta_i \cdot ({}^iA_{i-1} \cdot {}^{i-1}\vec{\omega}_{i-1}) \times {}^i\vec{k}_i \end{aligned} \quad (4.11)$$

The computed equations are equal and verify the ones reported in the Matlab functions.

### 4.3.6 Speed and accelerations of centers of mass of body 5

In order to compute the kinematic equations, the input data for the Matlab function shown in the previous chapter have to be all available. For this purpose, the following steps are done:

- Calculation of the homogeneous orientation matrices:  ${}^{i-1}A_o^i$ .
- Calculation of the position vector of the origin i with respect to frame (i-1) expressed in frame (i-1):  $O_{i-1}O_i$ .
- Definition of the versor  $k_i$ .
- Using the Matlab function kinem\_en.02.m, solve the kinematic equations.
- Plot the results.

The Speeds and the accelerations of the centers of mass of body 5 are shown in figure ??.

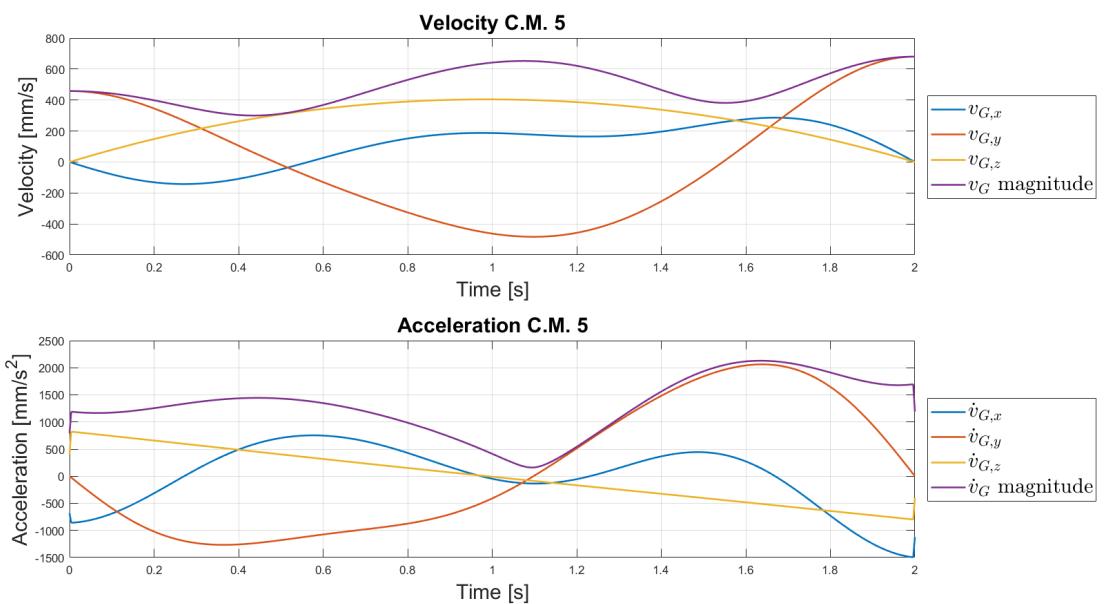


Figure 4.13: Kinematics of centers of mass 5

To verify the results showed previously, a Adams simulation (developed in chapter ??) is run. The extrapolated plots are shown below.

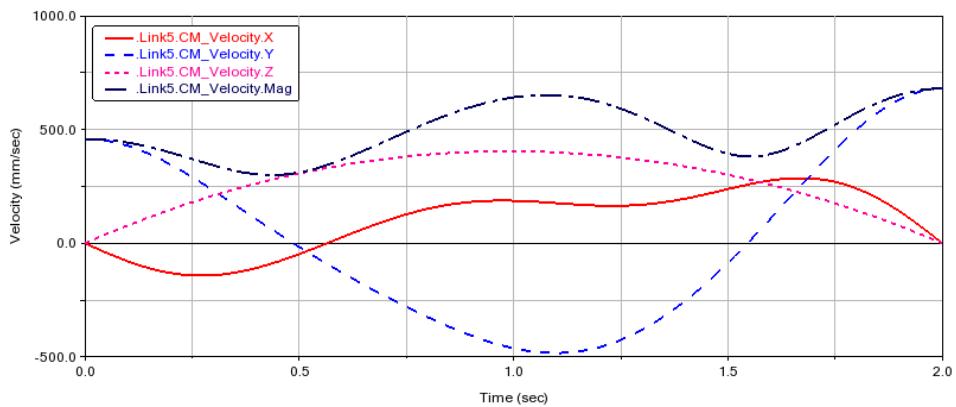


Figure 4.14: Adams simulation of body 5 CM velocity

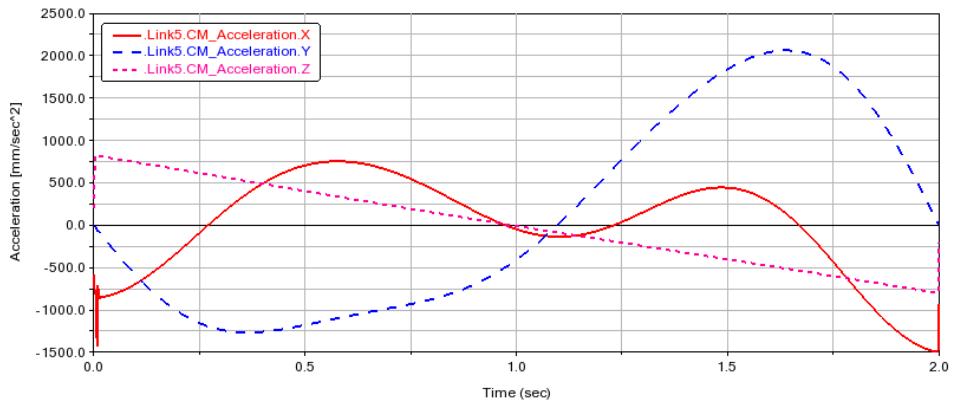


Figure 4.15: Adams simulation of body 5 CM acceleration

The Adams plot confirm the correctness of obtained results. The Matlab implementation is now reported.

```

1 %% Requirement 6 - Speeds and accelerations of CM 5
2
3 for j = 1:length(time)
4
5     % body 1
6     0001 = A01o(1:3,4);
7     A10 = inv(A01o);
8     A10 = A10(1:3,1:3);
9     k1 = [0 0 1]';
10    [w1(:,j),w1p(:,j),v1(:,j),vG1(:,j),v1p(:,j),vG1p(:,j)]=...
11        kinem_en02([0 0 0]',[0 0 0]',[0 0 0]',[0 0 0]',k1,0001, ...
12        CM1_1(1:3),delta(1),q1d(j),q1dd(j),A10);

```

```

13
14 % body 2
15 0102 = A12o(1:3,4);
16 A21 = inv(A12o);
17 A21 = A21(1:3,1:3);
18 k2 = [0 0 1]';
19 [w2(:,j),w2p(:,j),v2(:,j),vG2(:,j),v2p(:,j),vG2p(:,j)]=...
20 kinem_en02(w1(1:3,j),w1p(1:3,j),v1(1:3,j),v1p(1:3,j),k2,0102, ...
21 CM2_2(1:3),delta(2),q2d(j),q2dd(j),A21);

22
23 % body 3
24 k3 = [0 0 1]';
25 0203 = A23o(1:3,4);
26 A32 = inv(A23o);
27 A32 = A32(1:3,1:3);
28 [w3(:,j),w3p(:,j),v3(:,j),vG3(:,j),v3p(:,j),vG3p(:,j)]=...
29 kinem_en02(w2(1:3,j),w2p(1:3,j),v2(1:3,j),v2p(1:3,j),k3,0203, ...
30 CM3_3(1:3),delta(3),q3d(j),q3dd(j),A32);

31
32 % body 4
33 0304 = A34o(1:3,4);
34 A43 = inv(A34o);
35 A43 = A43(1:3,1:3);
36 k4 = [0 0 1]';
37 [w4(:,j),w4p(:,j),v4(:,j),vG4(:,j),v4p(:,j),vG4p(:,j)]=...
38 kinem_en02(w3(1:3,j),w3p(1:3,j),v3(1:3,j),v3p(1:3,j),k4,0304, ...
39 CM4_4(1:3),delta(4),q4d(j),q4dd(j),A43);

40
41 % body 5
42 0405 = A45o(1:3,4);
43 A54 = inv(A45o);
44 A54 = A54(1:3,1:3);
45 k5 = [0 0 1]';
46 [w5(:,j),w5p(:,j),v5(:,j),vG5(:,j),v5p(:,j),vG5p(:,j)]=...
47 kinem_en02(w4(1:3,j),w4p(1:3,j),v4(1:3,j),v4p(1:3,j),k5,0405, ...
48 CM5_5(1:3),delta(5),q5d(j),q5dd(j),A54);

49
50
51 %Body Payload
52 050P = A5Po(1:3,4);
53 AP5 = inv(A5Po);
54 AP5 = AP5(1:3,1:3);
55 kP = [0 0 1]';
56 [wP(:,j),wPp(:,j),vP(:,j),vGP(:,j),vPp(:,j),vGPP(:,j)]=...
57 kinem_en02(w5(1:3,j),w5p(1:3,j),v5(1:3,j),v5p(1:3,j),kP,050P, ...

```

```

58     CMP_P(1:3),delta(6),qPd(j),qPdd(j),AP5);
59
60 %velocity and acceleration of body 5 CM
61 vG5_0(:,j) = A05o(1:3,1:3)*(vG5(:,j));
62 vG5p_0(:,j) = A05o(1:3,1:3)*(vG5p(:,j));
63
64 end
65
66 %plot setup
67 vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
68 vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
69 legend_f = 18;
70 plot_s = 1.5;
71
72 %figure plot
73 %kinematic of body 5 CM plot
74 vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
75 vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
76 figure(6)
77 subplot(2,1,1)
78 plot(time,vG5_0(1,:),time,vG5_0(2,:),time,vG5_0(3,:),time,vG5_mag,LineWidth=kinCM_lw)
79 grid on
80 title("Velocity C.M. 5",FontSize=fs)
81 legend("$v_{[G,x]}$","$v_{[G,y]}$","$v_{[G,z]}$","$v_{[G]}$ magnitude", 'Interpreter',...
82 ' latex','Location',"eastoutside",fontsize=fs)
83 xlabel("Time [s]",FontSize=fs)
84 ylabel("Velocity [mm/s]",FontSize=fs)
85
86 subplot(2,1,2)
87 plot(time,vG5p_0(1,:),time,vG5p_0(2,:),time,vG5p_0(3,:),time,vG5p_mag,....
88 LineWidth=kinCM_lw)
89 grid on
90 title("Acceleration C.M. 5",FontSize=fs)
91 legend("$\dot{v}_{[G,x]}$","$\dot{v}_{[G,y]}$","$\dot{v}_{[G,z]}$","$\dot{v}_{[G]}$ magnitude", 'Interpreter',...
92 ' latex','Location',"eastoutside",fontsize=fs)
93 xlabel("Time [s]",FontSize=fs)
94 ylabel("Acceleration [mm/s^2]",FontSize=fs)
95
96

```

## 4.4 Implementation Exercise 2

### 4.4.1 Inertia matrices computations

The inertia matrices of the center of mass reference system are computed with the following formulas. In particular, the reference system of center od mass ( $(o_{G,i}x_{G,i}y_{G,i}z_{G,i})$ ) is set parallel to reference system of the joint ( $(o_i x_i y_i z_i)$ ) because, in this way, the inertia tensor computed in the center of mass is equal to the inertia tensor referred to the joint reference system. The equation ?? is for the parallelepipeds and the equation ?? is for the cylinders.

$$I_{G,i} = m_i \cdot \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (4.12)$$

$$I_{G,i} = \frac{m_i}{12} \cdot \begin{bmatrix} y^2 + z^2 & 0 & 0 \\ 0 & x^2 + z^2 & 0 \\ 0 & 0 & x^2 + y^2 \end{bmatrix} \quad (4.13)$$

$$I_{G,i} = m_i \cdot \begin{bmatrix} \frac{d^2}{16} + \frac{L^2}{12} & 0 & 0 \\ 0 & \frac{d^2}{16} + \frac{L^2}{12} & 0 \\ 0 & 0 & \frac{d^2}{8} \end{bmatrix} \quad (4.14)$$

The computed inertia tensors are reported below.

$$\begin{aligned} I_{G,0} &= \frac{m_0}{12} \cdot \begin{bmatrix} 200^2 + 160^2 & 0 & 0 \\ 0 & 200^2 + 245^2 & 0 \\ 0 & 0 & 245^2 + 160^2 \end{bmatrix} \cdot 10^{-6} = \\ &= \begin{bmatrix} 9.84 \cdot 10^{-2} & 0 & 0 \\ 0 & 1.5 \cdot 10^{-1} & 0 \\ 0 & 0 & 1.284 \cdot 10^{-1} \end{bmatrix} [kg \cdot m^2] \end{aligned} \quad (4.15)$$

$$\begin{aligned} I_{G,1} &= \frac{m_1}{12} \cdot \begin{bmatrix} 230^2 + 246.5^2 & 0 & 0 \\ 0 & 280^2 + 230^2 & 0 \\ 0 & 0 & 246.5^2 + 280^2 \end{bmatrix} \cdot 10^{-6} = \\ &= \begin{bmatrix} 9.95 \cdot 10^{-2} & 0 & 0 \\ 0 & 1.149 \cdot 10^{-1} & 0 \\ 0 & 0 & 1.1218 \cdot 10^{-1} \end{bmatrix} [kg \cdot m^2] \end{aligned} \quad (4.16)$$

$$\begin{aligned}
 I_{G,2} &= \frac{m_2}{12} \cdot \begin{bmatrix} 54^2 + 80^2 & 0 & 0 \\ 0 & 350^2 + 54^2 & 0 \\ 0 & 0 & 350^2 + 80^2 \end{bmatrix} \cdot 10^{-6} = \\
 &= \begin{bmatrix} 1.6 \cdot 10^{-2} & 0 & 0 \\ 0 & 2.09 \cdot 10^{-2} & 0 \\ 0 & 0 & 2.15 \cdot 10^{-2} \end{bmatrix} [kg \cdot m^2]
 \end{aligned} \tag{4.17}$$

$$\begin{aligned}
 I_{G,3} &= \frac{m_3}{12} \cdot \begin{bmatrix} 120^2 + 187.5^2 & 0 & 0 \\ 0 & 120^2 + 390^2 & 0 \\ 0 & 0 & 390^2 + 187.5^2 \end{bmatrix} \cdot 10^{-6} = \\
 &= \begin{bmatrix} 2.48 \cdot 10^{-2} & 0 & 0 \\ 0 & 8.32 \cdot 10^{-2} & 0 \\ 0 & 0 & 9.36 \cdot 10^{-2} \end{bmatrix} [kg \cdot m^2]
 \end{aligned} \tag{4.18}$$

$$\begin{aligned}
 I_{G,4} &= \frac{m_4}{12} \cdot \begin{bmatrix} 110^2 + 95^2 & 0 & 0 \\ 0 & 95^2 + 110^2 & 0 \\ 0 & 0 & 95^2 + 95^2 \end{bmatrix} \cdot 10^{-6} = \\
 &= \begin{bmatrix} 3.5 \cdot 10^{-2} & 0 & 0 \\ 0 & 3.5 \cdot 10^{-2} & 0 \\ 0 & 0 & 3 \cdot 10^{-2} \end{bmatrix} [kg \cdot m^2]
 \end{aligned} \tag{4.19}$$

$$\begin{aligned}
 I_{G,5} &= m_5 \cdot \begin{bmatrix} \frac{60^2}{16} + \frac{20^2}{12} & 0 & 0 \\ 0 & \frac{60^2}{16} + \frac{20^2}{12} & 0 \\ 0 & 0 & \frac{60^2}{8} \end{bmatrix} \cdot 10^{-6} = \\
 &= \begin{bmatrix} 1.2917 \cdot 10^{-4} & 0 & 0 \\ 0 & 1.2917 \cdot 10^{-4} & 0 \\ 0 & 0 & 2.25 \cdot 10^{-4} \end{bmatrix} [kg \cdot m^2]
 \end{aligned} \tag{4.20}$$

$$\begin{aligned}
 I_{G,P} &= \frac{m_P}{12} \cdot \begin{bmatrix} 80^2 + 30^2 & 0 & 0 \\ 0 & 230^2 + 30^2 & 0 \\ 0 & 0 & 80^2 + 230^2 \end{bmatrix} \cdot 10^{-6} = \\
 &= \begin{bmatrix} 6.08 \cdot 10^{-4} & 0 & 0 \\ 0 & 4.5 \cdot 10^{-3} & 0 \\ 0 & 0 & 4.9 \cdot 10^{-3} \end{bmatrix} [kg \cdot m^2]
 \end{aligned} \tag{4.21}$$

The Matlab implementation is now reported, where the final multiplication ( $10^{-6}$ ) is done to move from  $mm^2$  to  $m^2$ .

```

1 %% Exercise 2 - Requirement 1 - INERTIA MATRICES
2
3 IG0 = (m0/12).*[200^2+160^2      0      0;
4                  0      200^2+245^2      0;
5                  0      0      245^2+160^2].*(10^(-6));
6
7 IG1 = (m1/12).*[230^2+246.5^2      0      0;
8                  0      280^2+230^2      0;
9                  0      0      246.5^2+280^2].*(10^(-6));
10
11 IG2 = (m2/12).*[54^2+80^2      0      0;
12                  0      350^2+54^2      0;
13                  0      0      350^2+80^2].*(10^(-6));
14
15 IG3 = (m3/12).*[120^2+187.5^2      0      0;
16                  0      120^2+390^2      0;
17                  0      0      390^2+187.5^2].*(10^(-6));
18
19 IG4 = (m4/12).*[110^2+95^2      0      0;
20                  0      95^2+110^2      0;
21                  0      0      95^2+95^2].*(10^(-6));
22
23 IG5 = (m5).*[(60^2)/16+(20^2)/12      0      0;
24                  0      (60^2)/16+(20^2)/12      0;
25                  0      0      (60^2)/8].*(10^(-6));
26
27
28 IGP = (mP/12).*[80^2+30^2      0      0;
29                  0      230^2+30^2      0;
30                  0      0      230^2+80^2].*(10^(-6));

```

#### 4.4.2 Inverse dynamics solution - actuator torques

To solve the inverse dynamic problem, the Matlab function `dynam_en02.m` (equation ??) is used. It is fully reported in appendix ??.

$$[F_i, M_i, t_i] = \text{dynam\_en02}(F_{ip1}, M_{ip1}, mi, vGip, wi, wip, li, bi, Ii, iAip1, iA0) \quad (4.22)$$

The input data are many more for the payload in this case, so it is started from the computations of the last body in order to go back up to the ground body. In fact, reaction forces and momentum ( $F_{ip1}$  and  $M_{ip1}$ ) are null for the payload and the orientation matrix ( ${}^iA_{i+1}$ ) is equal to the identity matrix. Then, all the computations are done using as input data the ones obtained from the previous calculation. After that, is possible to compute the reacting forces and torques produced by the floor:

$$F_{floor} = -F_{0,basement} \quad (4.23)$$

$$M_{floor} = -M_{0,basement} \quad (4.24)$$

Results are shown in the following graph:

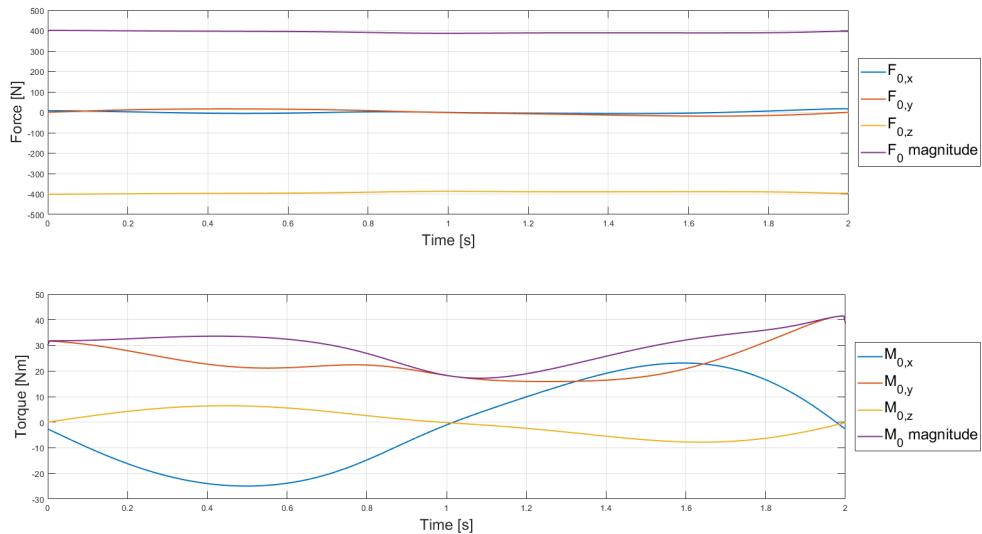


Figure 4.16: Reacting forces and torques produced by the floor

In order to do a comparison, the Adams simulation with the results of the forces and torques of body 0 are plotted.

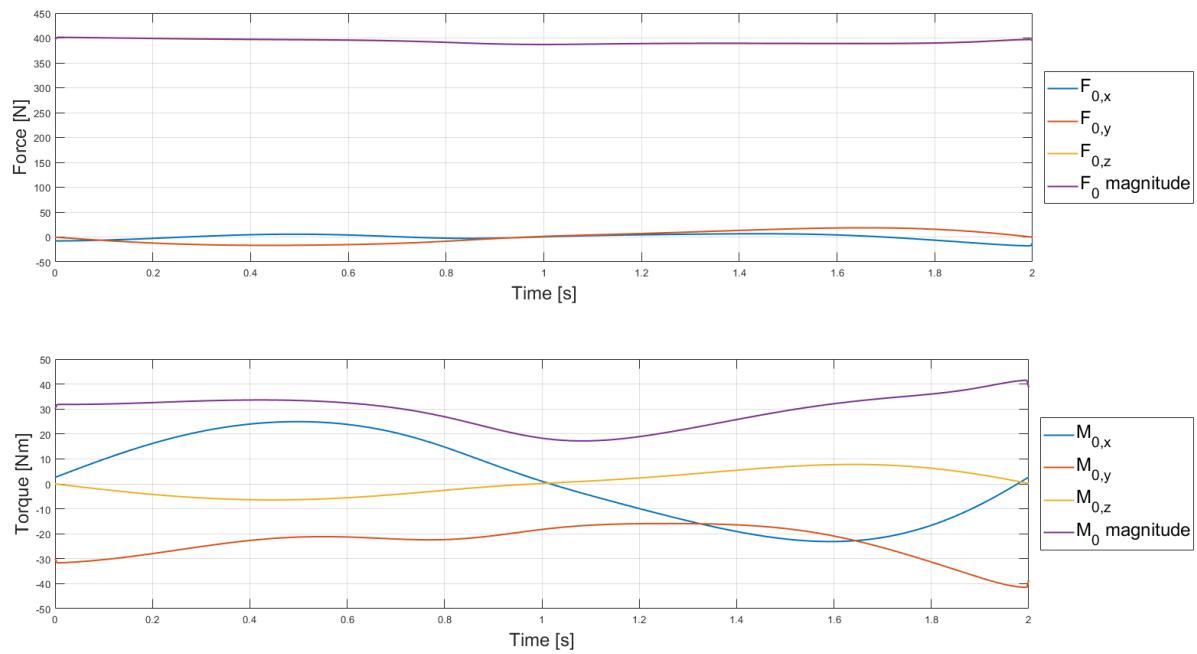


Figure 4.17: Forces and torques of by the body 0

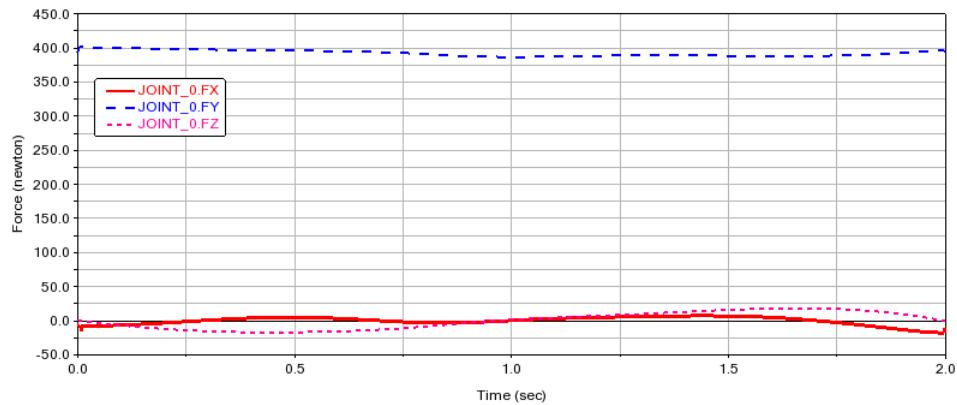


Figure 4.18: Adams simulation of body 0 forces

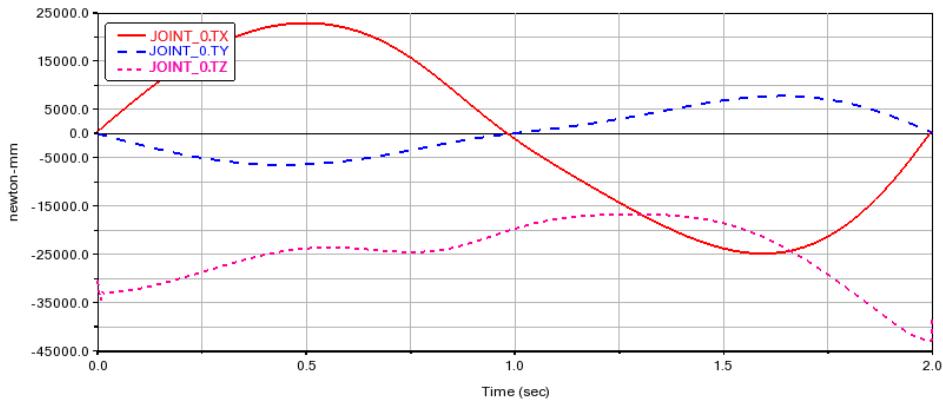


Figure 4.19: Adams simulation of body 0 torques

Afterwards, the actuator torques are calculated using the following equation:

$$\tau_i = k'_i \cdot F_i \cdot \delta_i + k'_i \cdot M_i \cdot (1 - \delta_i)$$

The plot showing  $\tau_i$  in function of time is reported and compared with the Adams simulation.

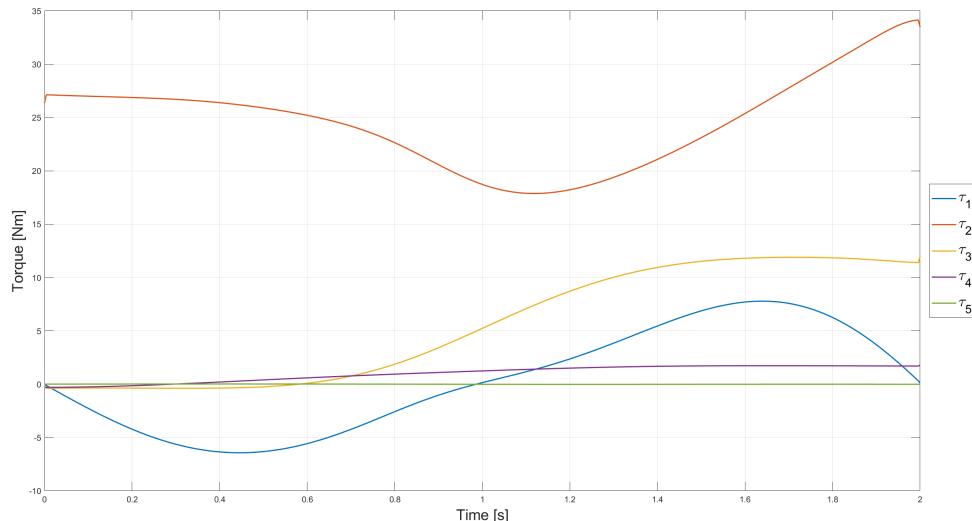


Figure 4.20: Actuator torque in function of time

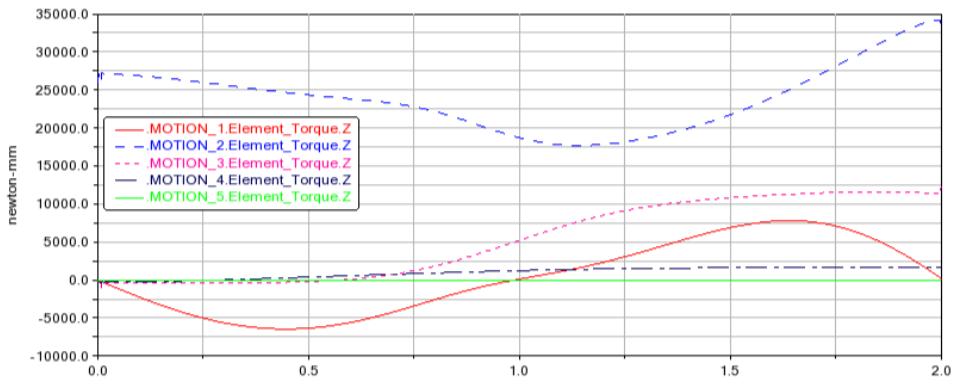


Figure 4.21: Adams simulation of actuator torque

The Matlab implementation is reported below.

```

1  %% Exercise 2 - Requirement 2 - INVERSE DYNAMIC PROBLEM
2  for j = 1:length(time)
3      %Payload
4      lP = [1 1 1]';
5      APO=inv(AOPo(1:3,1:3));
6      [FP(:,j),MP(:,j)] = dynam_en02([0 0 0]',[0 0 0]',...
7          mP,vGPP(:,j).*10^(-3),wP(:,j),wPP(:,j),lP, ...
8          CMP_P.*10^(-3),IGP,eye(3,3),APO);
9
10     %Body 5
11     l5 = A5Po(1:3,4);
12     A5O = inv(A05o(1:3,1:3));
13     [F5(:,j),M5(:,j)] = dynam_en02(FP(:,j),MP(:,j),...
14         m5,vG5p(:,j).*10^(-3),w5(:,j),w5p(:,j),15.*10^(-3),...
15         CM5_5.*10^(-3),IG5,A5Po(1:3,1:3),A5O);
16
17     %Body 4
18     l4 = A45o(1:3,4);
19     A4O = inv(A05o(1:3,1:3));
20     [F4(:,j),M4(:,j)] = dynam_en02(F5(:,j),M5(:,j),...
21         m4,vG4p(:,j).*10^(-3),w4(:,j),w4p(:,j),...
22         14.*10^(-3),CM4_4.*10^(-3),IG4,A45o(1:3,1:3),A4O);
23
24     %Body 3
25     l3 = A34o(1:3,4);
26     A3O = inv(A03o(1:3,1:3));
27     [F3(:,j),M3(:,j)] = dynam_en02(F4(:,j),M4(:,j),...
28         m3,vG3p(:,j).*10^(-3),w3(:,j),w3p(:,j),...
29         13.*10^(-3),CM3_3.*10^(-3),IG3,A34o(1:3,1:3),A3O);
30
31     %Body 2

```

```

32    l2 = A23o(1:3,4);
33    A20 = inv(A02o(1:3,1:3));
34    [F2(:,j),M2(:,j)] = dynam_en02(F3(:,j),M3(:,j),...
35        m2,vG2p(:,j).*10^(-3),w2(:,j),w2p(:,j),...
36        12.*10^(-3),CM2_2.*10^(-3),IG2,A23o(1:3,1:3),A20);

37
38    %Body 1
39    l1 = A12o(1:3,4);
40    A10 = inv(A01o(1:3,1:3));
41    [F1(:,j),M1(:,j)] = dynam_en02(F2(:,j),M2(:,j),...
42        m1,vG1p(:,j).*10^(-3),w1(:,j),w1p(:,j),...
43        11.*10^(-3),CM1_1.*10^(-3),IG1,A12o(1:3,1:3),A10);

44
45    %Body 0
46    l0 = A01o(1:3,4);
47    [F0(:,j),M0(:,j)] = dynam_en02(F1(:,j),M1(:,j),...
48        m0,[0 0 0]',[0 0 0]',[0 0 0]',...
49        10.*10^(-3),CM0_0.*10^(-3),IG0,A01o(1:3,1:3),eye(3,3));

50
51    t1 = [0 0 1]*M1(:,j);
52    t2 = [0 0 1]*M2(:,j);
53    t3 = [0 0 1]*M3(:,j);
54    t4 = [0 0 1]*M4(:,j);
55    t5 = [0 0 1]*M5(:,j);

56
57    t(:,j)=[t1 t2 t3 t4 t5]';
58 end

59
60 % Reacting force on the basement plot
61 figure(7)
62 plot(time,t(1,:),time,t(2,:),time,t(3,:),time,t(4,:),time,t(5,:),
63 LineWidth=dyn_lw)
63 legend("\tau_1","\tau_2","\tau_3","\tau_4","\tau_5","Location","eastoutside",...
64 fontsize=fs)
65 xlabel("Time [s]",fontsize=fs)
66 ylabel("Torque [Nm]",fontsize=fs)
67 grid on

68
69 % Reacting force on the basement plot
70 F0_mag = sqrt(F0(1,:).^2+F0(2,:).^2+F0(3,:).^2);
71 M0_mag = sqrt(M0(1,:).^2+M0(2,:).^2+M0(3,:).^2);

72
73 figure(8)
74 subplot(2,1,1)
75 plot(time,-1.*F0(1,:),time,-1.*F0(2,:),time,-1.*F0(3,:),
76 F0_mag,LineWidth=dyn_lw)
76 xlabel("Time [s]",fontsize=fs)

```

```
77 ylabel("Force [N]",fontsize=fs)
78 legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
79         fontsize=fs)
80 grid on
81
82 subplot(2,1,2)
83 plot(time,-1.*M0(1,:),time,-1.*M0(2,:),time,-1.*M0(3,:),time,M0_mag,LineWidth=dyn_lw)
84 xlabel("Time [s]",fontsize=fs)
85 ylabel("Torque [Nm]",fontsize=fs)
86 legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
87         fontsize=fs)
88 grid on
89
90 % Force on the body 0
91 figure(9)
92 subplot(2,1,1)
93 plot(time,F0(1,:),time,F0(2,:),time,F0(3,:),time,F0_mag,LineWidth=dyn_lw)
94 xlabel("Time [s]",fontsize=fs)
95 ylabel("Force [N]",fontsize=fs)
96 legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
97         fontsize=fs)
98 grid on
99
100 subplot(2,1,2)
101 plot(time,M0(1,:),time,M0(2,:),time,M0(3,:),time,M0_mag,LineWidth=dyn_lw)
102 xlabel("Time [s]",fontsize=fs)
103 ylabel("Torque [Nm]",fontsize=fs)
104 legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
105         fontsize=fs)
106 grid on
```

# Appendix A

## Matlab full codes

### A.1 Report 2 - Transformation Matrix

```
1 clear all
2 clc
3
4
5 %triangle points
6
7 P1 = [0 1 1]';
8 P2 = [0 7 1]';
9 P3 = [0 4 7]';
10
11
12
13 %positioning matrix of triangle
14
15 T0o = [P1,P2,P3,P1;...
16     1, 1, 1, 1];
17
18 %Coordinates of mobile reference system origin
19
20 P01 = [0 4 1]';
21
22 %define versors of reference systems
23
24 scale = 1;
25
26 %versor sys 0
27
28 i0 = [1 0 0]' * scale;
29 j0 = [0 1 0]' * scale;
```

```

30 k0 = [0 0 1]' * scale;
31
32 %versors sys 1
33
34 i1 = [0 1 0]' * scale;
35 j1 = [0 0 1]' * scale;
36 k1 = [1 0 0]' * scale;
37
38 %positioning matrix A01
39
40 A01 = [i1 j1 k1];
41
42
43
44
45 % homogeneus matrix A01o
46
47 A01o = [A01,P01;...
48     0 0 0 1];
49
50 %rotation of triangle about y1 of 90
51
52 theta1 = pi/2;
53
54 rotYo90 = rotYo(theta1);
55
56 B01o= A01o*rotYo90;
57
58 A10o = inv(A01o);
59
60 T1o = A10o * T0o;
61
62 T0_rotated1 = B01o * T1o;
63
64 %trajection of rotation 1
65
66 angles = linspace(0,theta1,1000);
67
68 for i=1:length(angles)
69     trajP1_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P1;1]);
70     trajP2_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P2;1]);
71     trajP3_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P3;1]);
72 end
73
74

```

```

75
76 %rotation of triangle about y0 oh theta2
77 theta2 = -pi/2;
78
79 A20o = A10o;
80
81 T2o = T1o;
82
83 A02o = A01o;
84
85 rotYom90 = rotYo(theta2);
86
87 C02o= rotYom90 * A02o;
88
89 T0_rotated2 = C02o * T2o;
90
91 %trajection of rotation 2
92
93 angles_2 = linspace(0,theta2,1000);
94
95 for i=1:length(angles)
96     trajP1_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P1;1]);
97     trajP2_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P2;1]);
98     trajP3_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P3;1]);
99 end
100
101 %rotation of triangle about x2 oh theta3
102 theta3 = pi/2;
103
104 rotXo90 = rotXo(theta3);
105
106 D02o = C02o * rotXo90;
107
108 C20o = inv(C02o);
109
110 T2 = C20o * T0_rotated2;
111
112 T0_rotated3 = D02o * T2;
113
114 %trajection of rotation 2
115
116 angles_3 = linspace(0,theta3,1000);
117
118 for i=1:length(angles)
119     trajP1_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...

```

```

120     T0_rotated2(:,1));
121     trajP2_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
122     T0_rotated2(:,2));
123     trajP3_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
124     T0_rotated2(:,3));
125 end
126
127
128
129 %% triangular plot
130
131 % plot settings
132 vector_lw = 3;
133 triang_lw = 1;
134 legend_fs = 18;
135 label_fs = 18;
136 text_fs = 18;
137
138 X_trang = [P1(1,1) P2(1,1) P3(1,1) P1(1,1)];
139 Y_trang = [P1(2,1) P2(2,1) P3(2,1) P1(2,1)];
140 Z_trang = [P1(3,1) P2(3,1) P3(3,1) P1(3,1)];
141
142 %initial plot
143
144 figure(1)
145
146 plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
147
148 xlabel("X",FontSize=label_fs)
149 ylabel("Y",FontSize=label_fs)
150 zlabel("Z",FontSize=label_fs)
151 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
152 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
153 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
154 legend("Initial position",fontsize=legend_fs)
155 axis equal
156 grid on
157
158
159 %plot versors
160
161 figure(2)
162
163 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
164 hold on

```

```

165 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
166 hold on
167 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
168 hold on
169
170 plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
171 hold on
172
173 quiver3(P01(1),P01(2),P01(3),i1(1),i1(2),i1(3),Color=[1 0.5 0.5],...
174 LineWidth=vector_lw)
175 hold on
176 quiver3(P01(1),P01(2),P01(3),j1(1),j1(2),j1(3),Color = "#77AC30",...
177 LineWidth=vector_lw)
178 hold on
179 quiver3(P01(1),P01(2),P01(3),k1(1),k1(2),k1(3),Color = "#4DBEEE",...
180 LineWidth=vector_lw)
181 xlabel("X",FontSize=label_fs)
182 ylabel("Y",FontSize=label_fs)
183 zlabel("Z",FontSize=label_fs)
184 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
185 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
186 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
187 text(P01(1)-0.2,P01(2)-0.5,P01(3)-0.5,"P_{0_1}","FontSize",text_fs)
188 legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
189 fontsize=legend_fs)
190 axis equal
191 grid on
192
193
194 %plot rotation 1
195 figure(3)
196
197 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
198 hold on
199 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
200 hold on
201 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
202 hold on
203
204 plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
205 hold on
206
207 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
208 Color=[1 0.5 0.5], LineWidth=vector_lw)
209 hold on

```

```

210 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
211     Color = "#77AC30", LineWidth=vector_lw)
212 hold on
213 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
214     Color = "#4DBEEE", LineWidth=vector_lw)
215 hold on
216
217 plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
218 Color = "#4DBEEE",LineWidth=triang_lw)
219 hold on
220
221 plot3(trajP1_rot1(:,1),trajP1_rot1(:,2),trajP1_rot1(:,3),"g")
222 hold on
223 plot3(trajP2_rot1(:,1),trajP2_rot1(:,2),trajP2_rot1(:,3),"g")
224 hold on
225 plot3(trajP3_rot1(:,1),trajP3_rot1(:,2),trajP3_rot1(:,3),"g")
226
227 xlabel("X",FontSize=label_fs)
228 ylabel("Y",FontSize=label_fs)
229 zlabel("Z",FontSize=label_fs)
230 text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
231 text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
232 text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
233 text(P01(1)-0.2,P01(2)-0.5,P01(3)-0.5,"P_{0_1}","FontSize",text_fs)
234 legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
235 "Rot1 90° about y_1","trajectories",fontsize=legend_fs)
236 axis equal
237 grid on
238
239 %plot rotation 2
240 figure(4)
241
242 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
243 hold on
244 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
245 hold on
246 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
247 hold on
248
249 plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
250 hold on
251
252 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
253     Color=[1 0.5 0.5], LineWidth=vector_lw)
254 hold on

```

```

255 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
256     Color = "#77AC30", LineWidth=vector_lw)
257 hold on
258 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
259     Color = "#4DBEEE", LineWidth=vector_lw)
260 hold on
261
262 plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
263 Color= "#D95319",LineWidth=triang_lw)
264 hold on
265
266 plot3(trajP1_rot2(:,1),trajP1_rot2(:,2),trajP1_rot2(:,3),"g")
267 hold on
268 plot3(trajP2_rot2(:,1),trajP2_rot2(:,2),trajP2_rot2(:,3),"g")
269 hold on
270 plot3(trajP3_rot2(:,1),trajP3_rot2(:,2),trajP3_rot2(:,3),"g")
271
272 xlabel("X",FontSize=label_fs)
273 ylabel("Y",FontSize=label_fs)
274 zlabel("Z",FontSize=label_fs)
275
276 legend("X_0","Y_0","Z_0","Initial position","X_2","Y_2","Z_2",...
277 "Rot2 -90° about y_0","trajectories",fontsize=legend_fs)
278 axis equal
279 grid on
280
281 %plot rotation 2
282 figure(5)
283
284 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
285 hold on
286 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
287 hold on
288 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
289 hold on
290
291 plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
292 hold on
293
294 plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),"--",...
295 LineWidth= triang_lw, Color= "#D95319")
296 hold on
297
298 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,1),D02o(2,1),D02o(3,1),...
299     Color=[1 0.5 0.5], LineWidth=vector_lw)

```

```

300 hold on
301 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,2),D02o(2,2),D02o(3,2),...
302     Color = "#77AC30", LineWidth=vector_lw)
303 hold on
304 quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,3),D02o(2,3),D02o(3,3),...
305     Color = "#4DBEEE", LineWidth=vector_lw)
306 hold on
307
308 plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
309     LineWidth= triang_lw, Color="#EDB120")
310 hold on
311
312 plot3(trajP1_rot3(:,1),trajP1_rot3(:,2),trajP1_rot3(:,3),"g")
313 hold on
314 plot3(trajP2_rot3(:,1),trajP2_rot3(:,2),trajP2_rot3(:,3),"g")
315 hold on
316 plot3(trajP3_rot3(:,1),trajP3_rot3(:,2),trajP3_rot3(:,3),"g")
317
318 xlabel("X",FontSize=label_fs)
319 ylabel("Y",FontSize=label_fs)
320 zlabel("Z",FontSize=label_fs)
321
322 legend("X_0","Y_0","Z_0","Initial position","Rot.2 on Y_0 -90°",...
323 "X_2","Y_2","Z_2","Rot.3 on x_2 90°","trajectories",fontsize=legend_fs)
324 axis equal
325 grid on
326
327 %Total Plot
328
329 figure(5)
330
331 quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
332 hold on
333 quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
334 hold on
335 quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
336 hold on
337
338 plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
339 hold on
340
341
342 plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
343 Color = "#4DBEEE",LineWidth=triang_lw)
344 hold on

```

```

345
346 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
347     Color=[0.6 0 0], LineWidth=vector_lw)
348 hold on
349 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
350     Color = [0.08 0.3 0], LineWidth=vector_lw)
351 hold on
352 quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
353     Color = [0 0 0.6], LineWidth=vector_lw)
354 hold on
355
356 plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
357 LineWidth=triang_lw,Color=" #D95319")
358 hold on
359
360 plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
361 LineWidth=triang_lw,Color="#EDB120")
362 hold on
363
364 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
365     Color=[1 0.5 0.5], LineWidth=vector_lw)
366 hold on
367 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
368     Color = "#77AC30", LineWidth=vector_lw)
369 hold on
370 quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
371     Color = "#4DBEEE", LineWidth=vector_lw)
372
373 xlabel("X",FontSize=label_fs)
374 ylabel("Y",FontSize=label_fs)
375 zlabel("Z",FontSize=label_fs)
376 legend("X_0","Y_0","Z_0","Initial position","Rot.1 on Y_1 90°",...
377 "X_1","Y_1","Z_1","Rot.2 on Y_0 -90°","Rot.3 on X_2 90°","X_2",...
378 "Y_2","Z_2",fontsize=legend_fs)
379 axis equal
380 grid on

```

## A.2 Report 3 - SCARA robot

```

1 clear all
2 clc
3 close all
4

```

```

5  %% DATA
6
7  load("joints.mat")
8
9  P000 = [0 0 0]';
10
11 q1 = M(:,1);    %[rad]
12 q2 = M(:,2);    %[rad]
13 q3 = M(:,3);    %[mm]
14
15 d1 = 25;        %[mm]
16 d2 = 15;        %[mm]
17 d3 = 10;        %[mm]
18
19 a1 = 50;        %[mm]
20 a2 = 10;        %[mm]
21
22 %% POSITIONING PROBLEM
23
24 i=1;            %initial position
25 %i=length(q1);      %final position
26
27 A01o = denhar_en01(0,0,d1,q1(i));
28 A12o = denhar_en01(0,a1,d2,q2(i));
29 A23o = denhar_en01(0,a2,-d3+q3(i),0);
30
31 A02o = A01o * A12o;
32 A03o = A02o * A23o;
33
34 P001 = A01o(1:3,4);
35 P002 = A02o(1:3,4);
36 P003 = A03o(1:3,4);
37
38 wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
39                      P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
40                      P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];
41
42 %ficticius matrix for body 3 joint
43 A03o_joint = A02o;
44 A03o_joint(1:2,4) = A03o(1:2,4);
45
46 % TRAJECTORY
47 for j = 1:length(q1)
48     A01oj = denhar_en01(0,0,d1,q1(j));
49     A12oj = denhar_en01(0,a1,d2,q2(j));

```

```

50     A23oj = denhar_en01(0,a2,-d3+q3(j),0);
51     A03oj = A01oj * A12oj * A23oj;
52     if j==1
53         P303o_init = inv(A03oj)*A03oj(:,4);
54     end
55     P003j = A03oj * P303o_init;
56     traj_EF(:,j) = P003j;
57 end
58
59 %Plot setup
60 radius = 1;
61 height_c = 3;
62 frame_lw = 3;
63 s = 5;
64 ref_lw = 3;
65 text_size = 18;
66 rf0_color = "#77AC30";
67 rf1_color = "#EDB120";
68 rf2_color = "#7E2F8E";
69 rf3_color = "#4DBEEE";
70 wf_color = "b";
71 %wf_color = "r";
72
73 figure(1)
74 %wire frame
75 wf = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),wf_color, ...
76             LineWidth=frame_lw);
77 hold on
78
79 %plot ref. 0
80 rrf0_1 = quiver3(P000(1),P000(2),P000(3),1*s,0*s,0*s, ...
81                     LineWidth=ref_lw,Color=rf0_color); %x0
82 hold on
83 rrf0_2 = quiver3(P000(1),P000(2),P000(3),0*s,1*s,0*s, ...
84                     LineWidth=ref_lw,Color=rf0_color); %y0
85 hold on
86 rrf0_3 = quiver3(P000(1),P000(2),P000(3),0*s,0*s,1*s, ...
87                     LineWidth=ref_lw,Color=rf0_color); %z0
88 hold on
89
90 %plot ref. 1
91 rrf1_1 = quiver3(P001(1),P001(2),P001(3), ...
92                     A01o(1,1)*s,A01o(2,1)*s,A01o(3,1)*s, ...
93                     LineWidth=ref_lw,Color=rf1_color); %x1
94 hold on

```

```

95 rf1_2 = quiver3(P001(1),P001(2),P001(3),...
96     A01o(1,2)*s,A01o(2,2)*s,A01o(3,2)*s,...
97     LineWidth=ref_lw,Color=rf1_color); %y1
98 hold on
99 rf1_3 = quiver3(P001(1),P001(2),P001(3),...
100    A01o(1,3)*s,A01o(2,3)*s,A01o(3,3)*s,...
101    LineWidth=ref_lw,Color=rf1_color); %z1
102 hold on
103
104 %plot ref. 2
105 rf2_1 = quiver3(P002(1),P002(2),P002(3),...
106    A02o(1,1)*s,A02o(2,1)*s,A02o(3,1)*s,...
107    LineWidth=ref_lw,Color=rf2_color); %x2
108 hold on
109 rf2_2 = quiver3(P002(1),P002(2),P002(3),...
110    A02o(1,2)*s,A02o(2,2)*s,A02o(3,2)*s,...
111    LineWidth=ref_lw,Color=rf2_color); %y2
112 hold on
113 rf2_3 = quiver3(P002(1),P002(2),P002(3),...
114    A02o(1,3)*s,A02o(2,3)*s,A02o(3,3)*s,...
115    LineWidth=ref_lw,Color=rf2_color); %z2
116 hold on
117
118 %plot ref. 3
119 rf3_1 = quiver3(P003(1),P003(2),P003(3),...
120    A03o(1,1)*s,A03o(2,1)*s,A03o(3,1)*s,...
121    LineWidth=ref_lw,Color=rf3_color); %x3
122 hold on
123 rf3_2 = quiver3(P003(1),P003(2),P003(3),...
124    A03o(1,2)*s,A03o(2,2)*s,A03o(3,2)*s,...
125    LineWidth=ref_lw,Color=rf3_color); %y3
126 hold on
127 rf3_3 = quiver3(P003(1),P003(2),P003(3),...
128    A03o(1,3)*s,A03o(2,3)*s,A03o(3,3)*s,...
129    LineWidth=ref_lw,Color=rf3_color); %z3
130 hold on
131
132 %joints
133 joint_rev_01(radius,height_c,20,A01o,wf_color);
134 hold on
135 joint_rev_01(radius,height_c,20,A02o,wf_color);
136 hold on
137 cube(A03o_joint(1:3,4),radius+0.5,wf_color,A03o_joint);
138 hold on
139 cube(A03o(1:3,4),radius+0.5,wf_color,A03o);

```

```

140
141 %plot settings
142 grid on
143 axis equal
144 xlabel("X [mm]", "FontSize", text_size)
145 ylabel("Y [mm]", "FontSize", text_size)
146 zlabel("Z [mm]", "FontSize", text_size)
147 legend([wf rf0_1 rf1_1 rf2_1 rf3_1], ...
148 {"Wire Frame", "rf. 0", "rf. 1", "rf. 2", "rf. 3"}, fontsize=text_size)
149
150 %%TOTAL PLOT
151 i=1;
152
153 A01o = denhar_en01(0,0,d1,q1(i));
154 A12o = denhar_en01(0,a1,d2,q2(i));
155 A23o = denhar_en01(0,a2,-d3+q3(i),0);
156
157 A02o = A01o * A12o;
158 A03o = A02o * A23o;
159
160 P001 = A01o(1:3,4);
161 P002 = A02o(1:3,4);
162 P003 = A03o(1:3,4);
163
164 wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1); ...
165 P000(2) P001(2) P002(2) P002(2) P003(2) P003(2); ...
166 P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];
167
168 A03o_joint = A03o;
169
170 A03o_joint(3,4) = A02o(3,4);
171
172 %Plot setup
173 radius = 1;
174 height_c = 3;
175 frame_lw = 3;
176 traj_lw = 1;
177
178 ref_lw = 3;
179 text_size = 18;
180 wf1_color = "b";
181 wf2_color = "r";
182 traj_color = "g";
183
184

```

```

185 figure(2)
186 %wire frame
187 wf1 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
188     wf1_color,LineWidth=frame_lw);
189 hold on
190
191 %joints
192 joint_rev_01(radius,height_c,20,A01o,wf1_color);
193 hold on
194 joint_rev_01(radius,height_c,20,A02o,wf1_color);
195 hold on
196 cube(A03o_joint(1:3,4),radius+0.5,wf1_color,A03o_joint);
197 hold on
198 cube(A03o(1:3,4),radius+0.5,wf1_color,A03o);
199 hold on
200
201 i=length(q1);
202
203 A01o = denhar_en01(0,0,d1,q1(i));
204 A12o = denhar_en01(0,a1,d2,q2(i));
205 A23o = denhar_en01(0,a2,-d3+q3(i),0);
206
207 A02o = A01o * A12o;
208 A03o = A02o * A23o;
209
210 P001 = A01o(1:3,4);
211 P002 = A02o(1:3,4);
212 P003 = A03o(1:3,4);
213
214 wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
215             P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
216             P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];
217
218 A03o_joint = A03o;
219
220 A03o_joint(3,4) = A02o(3,4);
221
222 figure(2)
223 %wire frame
224 wf2 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
225     wf2_color,LineWidth=frame_lw);
226 hold on
227
228 %joints
229 joint_rev_01(radius,height_c,20,A01o,wf2_color);

```

```
230 hold on
231 joint_rev_01(radius,height_c,20,A02o,wf2_color);
232 hold on
233 cube(A03o_joint(1:3,4),radius+0.5,wf2_color,A03o_joint);
234 hold on
235 cube(A03o(1:3,4),radius+0.5,wf2_color,A03o);
236 hold on
237 traj = plot3(traj_EF(1,:),traj_EF(2,:),traj_EF(3,:),...
238     LineWidth=traj_lw,Color=traj_color);
239
240 %plot settings
241 grid on
242 axis equal
243 xlabel("X [mm]", "FontSize",text_size)
244 ylabel("Y [mm]", "FontSize",text_size)
245 zlabel("Z [mm]", "FontSize",text_size)
246 legend([wf1 wf2 traj], {"Wire Frame initial pos.", ...
247     "Wire Frame final pos.", "Trajectory"}, ...
248     fontsize=text_size)
```

### A.3 Report 4 - 5 D.O.F. Robot

```
1 %% SETUP
2 clear all
3 clc
4 close all
5 format long e
6
7 %% DATA
8
9 data = load("trajectory.mat");
10
11 a = 200;
12 b = 150;
13 c = 150;
14 d = 100;
15 e = 250;
16 f = 100;
17 g = 220;
18 h = 60;
19 i = 20;
20 l = 40;
21
22 q1 = data.q1vec;
23 q2 = data.q2vec;
24 q3 = data.q3vec;
25 q4 = data.q4vec;
26 q5 = data.q5vec;
27
28 q1d = data.q1dvec.*((pi/180));
29 q2d = data.q2dvec.*((pi/180));
30 q3d = data.q3dvec.*((pi/180));
31 q4d = data.q4dvec.*((pi/180));
32 q5d = data.q5dvec.*((pi/180));
33
34 q1dd = data.q1ddvec.*((pi/180));
35 q2dd = data.q2ddvec.*((pi/180));
36 q3dd = data.q3ddvec.*((pi/180));
37 q4dd = data.q4ddvec.*((pi/180));
38 q5dd = data.q5ddvec.*((pi/180));
39
40 time = data.timevec;
41
42 qPd = zeros(length(time),1);
43 qPdd = zeros(length(time),1);
```

```

44
45 CM0_0 = [-140 0 100]';
46 CM1_1 = [80 0 35]';
47 CM2_2 = [125 0 -27]';
48 CM3_3 = [55 0 100]';
49 CM4_4 = [-7.5 -12.5 5]';
50 CM5_5 = [0 0 10]';
51 CMP_P = [0 0 15]';

52
53 m0 = 18;
54 m1 = 10.5;
55 m2 = 2;
56 m3 = 6;
57 m4 = 2;
58 m5 = 0.5;
59 mP = 1;

60
61 delta = zeros(6,1);

62
63 %% INERTIA MATRIX

64
65 IG0 = (m0/12).*[200^2+160^2 0 0;
66 0 200^2+245^2 0;
67 0 0 245^2+160^2].*(10^(-6));
68
69 IG1 = (m1/12).*[230^2+246.5^2 0 0;
70 0 280^2+230^2 0;
71 0 0 246.5^2+280^2].*(10^(-6));
72
73 IG2 = (m2/12).*[ 54^2+80^2 0 0;
74 0 350^2+54^2 0;
75 0 0 350^2+80^2].*(10^(-6));
76
77 IG3 = (m3/12).*[120^2+187.5^2 0 0;
78 0 120^2+390^2 0;
79 0 0 390^2+187.5^2].*(10^(-6));
80
81 IG4 = (m4/12).*[110^2+95^2 0 0;
82 0 95^2+110^2 0;
83 0 0 95^2+95^2].*(10^(-6));
84
85 IG5 = (m5).*[(60^2)/16+(20^2)/12 0 0;
86 0 (60^2)/16+(20^2)/12 0;
87 0 0 (60^2)/8].*(10^(-6));
88

```

```

89
90 IGP = (mP/12).*[80^2+30^2          0          0;
91           0          230^2+30^2          0;
92           0          0          230^2+80^2].*(10^(-6));
93
94
95 %% Solution
96
97 for j = 1:length(time)
98
99
100    % POSITIONING PROBLEM
101
102    %Denavit-Huntberg covention
103    A01o = denhar_en01( 0,      0,      a+b,      deg2rad(q1(j)));
104    A12o = denhar_en01(pi/2,    c,      -d,      pi/2+deg2rad(q2(j)));
105    A23o = denhar_en01( 0,      e,      0,      -pi/2+deg2rad(q3(j)));
106    A34o = denhar_en01( 0,      g,      f,      pi/2+deg2rad(q4(j)));
107    A45o = denhar_en01(pi/2,    0,      h,      deg2rad(q5(j)));
108    A5EEo = denhar_en01( 0,      0,      i,      0);
109    A5Po = denhar_en01( 0,      0,      l+i,     0);
110
111    A02o = A01o * A12o;
112    A03o = A02o * A23o;
113    A04o = A03o * A34o;
114    A05o = A04o * A45o;
115    A0EEo = A05o * A5EEo;
116    AOPo = A05o * A5Po;
117
118
119    %wire frame in the initial position
120    if j ==1
121        wf_init = [0 A01o(1,4)   A01o(1,4) A01o(1,4)+c   A02o(1,4)   A03o(1,4)...
122                           A03o(1,4)   A04o(1,4) A05o(1,4) A0EEo(1,4);
123                           0 A01o(2,4)   A01o(2,4) A01o(2,4)   A02o(2,4)   A03o(2,4)...
124                           A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
125                           0 A01o(3,4)-b A01o(3,4) A01o(3,4)   A02o(3,4)   A03o(3,4)...
126                           A03o(3,4)   A04o(3,4) A05o(3,4) A0EEo(3,4)];
127
128        A_joint1 = A01o;
129        A_joint1(3,4) = A01o(3,4)-b;
130
131        initial_cond = {wf_init,A_joint1,A02o,A03o,A04o,A05o,A0EEo}' ;
132    end
133

```

```

134 %wire frame in the final position
135 if j == length(time)
136     wf_fin = [0 A01o(1,4) A01o(1,4) A01o(1,4)+c A02o(1,4) A03o(1,4)...
137                 A03o(1,4) A04o(1,4) A05o(1,4) A0EEo(1,4);
138                 0 A01o(2,4) A01o(2,4) A01o(2,4) A02o(2,4) A03o(2,4)...
139                 A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
140                 0 A01o(3,4)-b A01o(3,4) A01o(3,4) A02o(3,4) A03o(3,4)...
141                 A03o(3,4) A04o(3,4) A05o(3,4) A0EEo(3,4)];
142
143     A_joint1 = A01o;
144     A_joint1(3,4) = A01o(3,4)-b;
145
146     final_cond = {wf_fin,A_joint1,A02o,A03o,A04o,A05o,A0EEo}' ;
147 end
148
149 %trajectory EE
150 traj_EE(:,j) = A0EEo(1:3,4);
151
152 %trajectory EE
153 traj_GP(:,j) = A04o(1:3,4);
154
155
156
157 % KINEMATIC PROBLEM
158
159 % body 1
160 0001 = A01o(1:3,4);
161 A10 = inv(A01o);
162 A10 = A10(1:3,1:3);
163 k1 = [0 0 1]';
164 [w1(:,j),w1p(:,j),v1(:,j),vG1(:,j),v1p(:,j),vG1p(:,j)]=...
165 kinem_en02([0 0 0]',[0 0 0]',[0 0 0]',[0 0 0]',...
166 k1,0001,CM1_1(1:3),delta(1),q1d(j),q1dd(j),A10);
167
168
169 % body 2
170 0102 = A12o(1:3,4);
171 A21 = inv(A12o);
172 A21 = A21(1:3,1:3);
173 k2 = [0 0 1]';
174 [w2(:,j),w2p(:,j),v2(:,j),vG2(:,j),v2p(:,j),vG2p(:,j)]=...
175 kinem_en02(w1(1:3,j),w1p(1:3,j),v1(1:3,j),v1p(1:3,j),...
176 k2,0102,CM2_2(1:3),delta(2),q2d(j),q2dd(j),A21);
177
178 % body 3

```

```

179 k3 = [0 0 1]';
180 O203 = A23o(1:3,4);
181 A32 = inv(A23o);
182 A32 = A32(1:3,1:3);
183 [w3(:,j),w3p(:,j),v3(:,j),vG3(:,j),v3p(:,j),vG3p(:,j)]=...
184 kinem_en02(w2(1:3,j),w2p(1:3,j),v2(1:3,j),v2p(1:3,j),...
185 k3,O203,CM3_3(1:3),delta(3),q3d(j),q3dd(j),A32);

186
187 % body 4
188 O304 = A34o(1:3,4);
189 A43 = inv(A34o);
190 A43 = A43(1:3,1:3);
191 k4 = [0 0 1]';
192 [w4(:,j),w4p(:,j),v4(:,j),vG4(:,j),v4p(:,j),vG4p(:,j)]=...
193 kinem_en02(w3(1:3,j),w3p(1:3,j),v3(1:3,j),v3p(1:3,j),...
194 k4,O304,CM4_4(1:3),delta(4),q4d(j),q4dd(j),A43);

195
196 % body 5
197 O405 = A45o(1:3,4);
198 A54 = inv(A45o);
199 A54 = A54(1:3,1:3);
200 k5 = [0 0 1]';
201 [w5(:,j),w5p(:,j),v5(:,j),vG5(:,j),v5p(:,j),vG5p(:,j)]=...
202 kinem_en02(w4(1:3,j),w4p(1:3,j),v4(1:3,j),v4p(1:3,j),...
203 k5,O405,CM5_5(1:3),delta(5),q5d(j),q5dd(j),A54);

204
205
206 %Body Payload
207 O50P = A5Po(1:3,4);
208 AP5 = inv(A5Po);
209 AP5 = AP5(1:3,1:3);
210 kP = [0 0 1]';
211 [wP(:,j),wPp(:,j),vP(:,j),vGP(:,j),vPp(:,j),vGPP(:,j)]=...
212 kinem_en02(w5(1:3,j),w5p(1:3,j),v5(1:3,j),v5p(1:3,j),...
213 kP,O50P,CMP_P(1:3),delta(6),qPd(j),qPdd(j),AP5);

214
215 %velocity and acceleration of body 5 CM
216 vG5_0(:,j) = A05o(1:3,1:3)*(vG5(:,j));
217 vG5p_0(:,j) = A05o(1:3,1:3)*(vG5p(:,j));
218
219
220 %DYNAMIC PROBLEM
221
222 %Payload
223 lP = [1 1 1]';

```

```

224 AP0=inv(AOPo(1:3,1:3));
225 [FP(:,j),MP(:,j)] = dynam_en02([0 0 0]',[0 0 0]',mP,vGPP(:,j).*10^(-3),...
226 wP(:,j),wPP(:,j),1P,CMP_P.*10^(-3),IGP,eye(3,3),AP0);
227
228 %Body 5
229 l5 = A5Po(1:3,4);
230 A50 = inv(A05o(1:3,1:3));
231 [F5(:,j),M5(:,j)] = dynam_en02(FP(:,j),MP(:,j),m5,vG5p(:,j).*10^(-3),...
232 w5(:,j),w5p(:,j),l5.*10^(-3),CM5_5.*10^(-3),IG5,A5Po(1:3,1:3),A50);
233
234 %Body 4
235 l4 = A45o(1:3,4);
236 A40 = inv(A04o(1:3,1:3));
237 [F4(:,j),M4(:,j)] = dynam_en02(F5(:,j),M5(:,j),m4,vG4p(:,j).*10^(-3),...
238 w4(:,j),w4p(:,j),l4.*10^(-3),CM4_4.*10^(-3),IG4,A45o(1:3,1:3),A40);
239
240 %Body 3
241 l3 = A34o(1:3,4);
242 A30 = inv(A03o(1:3,1:3));
243 [F3(:,j),M3(:,j)] = dynam_en02(F4(:,j),M4(:,j),m3,vG3p(:,j).*10^(-3),...
244 w3(:,j),w3p(:,j),l3.*10^(-3),CM3_3.*10^(-3),IG3,A34o(1:3,1:3),A30);
245
246 %Body 2
247 l2 = A23o(1:3,4);
248 A20 = inv(A02o(1:3,1:3));
249 [F2(:,j),M2(:,j)] = dynam_en02(F3(:,j),M3(:,j),m2,vG2p(:,j).*10^(-3),...
250 w2(:,j),w2p(:,j),l2.*10^(-3),CM2_2.*10^(-3),IG2,A23o(1:3,1:3),A20);
251
252 %Body 1
253 l1 = A12o(1:3,4);
254 A10 = inv(A01o(1:3,1:3));
255 [F1(:,j),M1(:,j)] = dynam_en02(F2(:,j),M2(:,j),m1,vG1p(:,j).*10^(-3),...
256 w1(:,j),w1p(:,j),l1.*10^(-3),CM1_1.*10^(-3),IG1,A12o(1:3,1:3),A10);
257
258 %Body 0
259 l0 = A01o(1:3,4);
260 [F0(:,j),M0(:,j)] = dynam_en02(F1(:,j),M1(:,j),m0,[0 0 0]',[0 0 0]',[0 0 0]',...
261 10.*10^(-3),CM0_0.*10^(-3),IGO,A01o(1:3,1:3),eye(3,3));
262
263 t1 = [0 0 1]*M1(:,j);
264 t2 = [0 0 1]*M2(:,j);
265 t3 = [0 0 1]*M3(:,j);
266 t4 = [0 0 1]*M4(:,j);
267 t5 = [0 0 1]*M5(:,j);

```

```

269 t(:,j)=[t1 t2 t3 t4 t5]';  

270  

271 end  

272  

273  

274 %% PLOTS  

275  

276 %Plots setup  

277 wf_lw =1.5;  

278 r_j = 5;  

279 l_j = 20;  

280 fs = 18;  

281 kin_lw = 1.5;  

282 kinCM_lw = 1.5;  

283 dyn_lw = 1.5;  

284  

285  

286 %Wire frame plot  

287 figure(1)  

288 wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...  

289 Color="b",LineWidth=wf_lw); %Initial position  

290 hold on  

291 wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...  

292 Color="r",LineWidth=wf_lw); %Final position  

293  

294 joint_rev_01(r_j,l_j,10,initial_cond{2}, "b");  

295 joint_rev_01(r_j,l_j,10,initial_cond{3}, "b");  

296 joint_rev_01(r_j,l_j,10,initial_cond{4}, "b");  

297 joint_rev_01(r_j,l_j,10,initial_cond{5}, "b");  

298 joint_rev_01(r_j,l_j,10,initial_cond{6}, "b");  

299 joint_rev_01(r_j,r_j,4,initial_cond{7}, "b");  

300  

301 joint_rev_01(r_j,l_j,10,final_cond{2}, "r");  

302 joint_rev_01(r_j,l_j,10,final_cond{3}, "r");  

303 joint_rev_01(r_j,l_j,10,final_cond{4}, "r");  

304 joint_rev_01(r_j,l_j,10,final_cond{5}, "r");  

305 joint_rev_01(r_j,l_j,10,final_cond{6}, "r");  

306 joint_rev_01(r_j,r_j,4,final_cond{7}, "r");  

307  

308 legend([wf_plot(1) wf_plot(2)], {"Initial position", "Final position"}, fontsize=fs)  

309 xlabel("X [mm]", "FontSize", fs)  

310 ylabel("Y [mm]", "FontSize", fs)  

311 zlabel("Z [mm]", "FontSize", fs)  

312 axis equal  

313 grid on

```

```

314
315
316 %wire frame+trajectories plot
317 figure(2)
318 wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...
319             Color="b",LineWidth=wf_lw); %Initial position
320 hold on
321 wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...
322             Color="r",LineWidth=wf_lw); %Final position
323 hold on
324
325 t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
326 hold on
327 t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory
328
329 joint_rev_01(r_j,l_j,10,initial_cond{2},"b");
330 joint_rev_01(r_j,l_j,10,initial_cond{3},"b");
331 joint_rev_01(r_j,l_j,10,initial_cond{4},"b");
332 joint_rev_01(r_j,l_j,10,initial_cond{5},"b");
333 joint_rev_01(r_j,l_j,10,initial_cond{6},"b");
334 joint_rev_01(r_j,r_j,4,initial_cond{7},"b");
335
336 joint_rev_01(r_j,l_j,10,final_cond{2},"r");
337 joint_rev_01(r_j,l_j,10,final_cond{3},"r");
338 joint_rev_01(r_j,l_j,10,final_cond{4},"r");
339 joint_rev_01(r_j,l_j,10,final_cond{5},"r");
340 joint_rev_01(r_j,l_j,10,final_cond{6},"r");
341 joint_rev_01(r_j,r_j,4,final_cond{7},"r");
342
343 legend([wf_plot(1) wf_plot(2) t1 t2],{"Initial position", "Final position",...
344         "EE Trajectory", "CP Trajectory"},fontSize=fs)
345 xlabel("X [mm]", "FontSize", fs)
346 ylabel("Y [mm]", "FontSize", fs)
347 zlabel("Z [mm]", "FontSize", fs)
348 axis equal
349 grid on
350
351
352 %requirement 2 - ang,vel,acc plot
353 figure(3)
354 plot(time,q1,time,q2,time,q3,time,q4,time,q5,LineWidth=kin_lw)
355 hold on
356 grid on
357 title("Joint displacement",FontSize=fs)
358 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...

```

```

359     fontsize=fs)
360 xlabel("Time [s]",FontSize=fs)
361 ylabel("Angles [Degrees]",FontSize=fs)
362
363 figure(4)
364 plot(time,q1d,time,q2d,time,q3d,time,q4d,time,q5d,LineWidth=kin_lw)
365 grid on
366 title("Joint angular velocity",FontSize=fs)
367 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...
368         fontsize=fs)
369 xlabel("Time [s]",FontSize=fs)
370 ylabel("Angular velocity [rad/s]",FontSize=fs)
371
372 figure(5)
373 plot(time,q1dd,time,q2dd,time,q3dd,time,q4dd,time,q5dd,LineWidth=kin_lw)
374 grid on
375 title("Joint angular acceleration",FontSize=fs)
376 legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...
377         fontsize=fs)
378 xlabel("Time [s]",FontSize=fs)
379 ylabel("Angular acceleration [rad/s^2]",FontSize=fs)
380
381
382 %kinematic of body 5 CM plot
383 vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
384 vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
385 figure(6)
386 subplot(2,1,1)
387 plot(time,vG5_0(1,:),time,vG5_0(2,:),time,vG5_0(3,:),time,vG5_mag,LineWidth=kinCM_lw)
388 grid on
389 title("Velocity C.M. 5",FontSize=fs)
390 legend("$v_{G,x}$","$v_{G,y}$","$v_{G,z}$",...
391         "$v_G$ magnitude", 'Interpreter','latex','Location',...
392         "eastoutside",fontsize=fs)
393 xlabel("Time [s]",FontSize=fs)
394 ylabel("Velocity [mm/s]",FontSize=fs)
395
396 subplot(2,1,2)
397 plot(time,vG5p_0(1,:),time,vG5p_0(2,:),time,vG5p_0(3,:),time,vG5p_mag, ...
398       LineWidth=kinCM_lw)
399 grid on
400 title("Acceleration C.M. 5",FontSize=fs)
401 legend("$\dot{v}_{G,x}$","$\dot{v}_{G,y}$","$\dot{v}_{G,z}$",...
402         "$\dot{v}_G$ magnitude", 'Interpreter',...
403         'latex','Location',"eastoutside",fontsize=fs)

```

```

404 xlabel("Time [s]",FontSize=fs)
405 ylabel("Acceleration [mm/s^2]",FontSize=fs)
406
407
408 % Reacting force on the basement plot
409 figure(7)
410 plot(time,t(1,:),time,t(2,:),time,t(3,:),time,t(4,:),time,t(5,:),LineWidth=dyn_lw)
411 legend("\tau_1","\tau_2","\tau_3","\tau_4","\tau_5","Location","eastoutside",...
412         fontsize=fs)
413 xlabel("Time [s]",fontsize=fs)
414 ylabel("Torque [Nm]",fontsize=fs)
415 grid on
416
417 % Reacting force on the basement plot
418 F0_mag = sqrt(F0(1,:).^2+F0(2,:).^2+F0(3,:).^2);
419 M0_mag = sqrt(M0(1,:).^2+M0(2,:).^2+M0(3,:).^2);
420
421 figure(8)
422 subplot(2,1,1)
423 plot(time,-1.*F0(1,:),time,-1.*F0(2,:),time,-1.*F0(3,:),time,F0_mag,LineWidth=dyn_lw)
424 xlabel("Time [s]",fontsize=fs)
425 ylabel("Force [N]",fontsize=fs)
426 legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
427         fontsize=fs)
428 grid on
429
430 subplot(2,1,2)
431 plot(time,-1.*M0(1,:),time,-1.*M0(2,:),time,-1.*M0(3,:),time,M0_mag,LineWidth=dyn_lw)
432 xlabel("Time [s]",fontsize=fs)
433 ylabel("Torque [Nm]",fontsize=fs)
434 legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
435         fontsize=fs)
436 grid on
437
438 % Force on the body 0
439 figure(9)
440 subplot(2,1,1)
441 plot(time,F0(1,:),time,F0(2,:),time,F0(3,:),time,F0_mag,LineWidth=dyn_lw)
442 xlabel("Time [s]",fontsize=fs)
443 ylabel("Force [N]",fontsize=fs)
444 legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
445         fontsize=fs)
446 grid on
447
448 subplot(2,1,2)

```

```
449 plot(time,M0(1,:),time,M0(2,:),time,M0(3,:),time,M0_mag,LineWidth=dyn_lw)
450 xlabel("Time [s]",fontsize=fs)
451 ylabel("Torque [Nm]",fontsize=fs)
452 legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
453         fontsize=fs)
454 grid on
```

# Appendix B

## Matlab functions

### B.1 Rotation about y-axis

```
1 function [hom_rotation_matrix] = rotYo(theta)
2
3 hom_rotation_matrix = [cos(theta)      0      sin(theta)      0;...
4                      0          1          0          0;...
5                      -1*sin(theta)  0      cos(theta)      0;...
6                      0          0          0          1];
7 end
```

### B.2 Rotation about x-axis

```
1 function [hom_rotated_matrix_X] = rotXo(theta)
2
3 hom_rotated_matrix_X = [      1          0          0          0;...
4                      0      cos(theta)  -sin(theta)  0;...
5                      0      sin(theta)  cos(theta)  0;...
6                      0          0          0          1];
7 end
```

### B.3 Denavit-Hartenberg convention

```
1 % denhar_en01.m
2 % Function to compute the transformation matrix of the reference frame i with
3 % respect to the reference frame (i-1)
4 % in accordance with the Denavit-Hartenberg's convention (Craig's version)
5 %
6 % Input variables (Denavit-Hartenberg's parameters):
```

```

7 % alfa = twist angle [rad]
8 % a = offset distance [m]
9 % d = length [m]
10 % teta = angle [rad]
11 %
12 % Output variables:
13 % A = transformation matrix of the reference frame i with respect to the reference
14 % frame (i-1)
15
16 function A=denhar_en01(alfa,a,d,teta)
17
18 A=[cos(teta)           -sin(teta)           0           a
19     sin(teta)*cos(alfa)   cos(teta)*cos(alfa)   -sin(alfa)   -d*sin(alfa)
20     sin(teta)*sin(alfa)   cos(teta)*sin(alfa)   cos(alfa)    d*cos(alfa)
21     0                     0                     0           1];

```

## B.4 Revolution joint representation

```

1 % joint_rev_01.m
2 % Function to plot a revolute joint like a cylinder
3 %
4 % Input variables:
5 % R = cylinder radius [m]
6 % H = cylinder height [m]
7 % N = points numero punti sulla circonferenza
8 % A = 4x4 homogenous matrix of the reference frame attached to the cylinder,
9 %      with cylinder axis aligned with the 3rd reference axis and
10 %      cylinder centre located at the reference origin
11 % col = color ('blu','red','green',yellow',...
12
13 function [h]=joint_rev_01(R,H,N,A,col)
14
15 [xc,yc,zc1] = cylinder(R, N);
16 zc=zc1*H-H/2;
17 base1=A*[xc(1,:); yc(1,:);zc(1,:);ones(1,N+1)];
18 base2=A*[xc(2,:); yc(2,:);zc(2,:);ones(1,N+1)];
19 xc0=[base1(1,:);base2(1,:)];
20 yc0=[base1(2,:);base2(2,:)];
21 zc0=[base1(3,:);base2(3,:)];
22 surface(xc0,yc0,zc0,'FaceColor', col)
23
24 end

```

## B.5 Kinematic equations

```

1 % kinem_en02.m
2 % Function for recursive forward computation of kinematic variables of link i
3 % in terms of kinematic variables of link (i-1) and of d.o.f. in joint i.
4 %
5 % input variables:
6 % wim1 = angular velocity of link (i-1) expressed in frame (i-1)
7 % wim1p = angular acceleration of link (i-1) expressed in frame (i-1)
8 % vim1 = linear velocity of the origin of link (i-1) expressed in frame (i-1)
9 % vim1p = linear acceleration of the origin of link (i-1) expressed in frame (i-1)
10 % ki = unit vector k of joint i axis expressed in frame i
11 % lim1 = position vector of the origin i with respect to frame (i-1) expressed
12 %       in frame (i-1)
13 % bi = position vector of center of mass of link i with respect to frame i expressed
14 %       in frame i
15 % deltai = joint i parameter (1/0)
16 % qip = joint i d.o.f. velocity
17 % qipp = joint i d.o.f. acceleration
18 % iAim1 = rotation matrix (i)A(i-1) of frame (i-1) with respect to frame i
19 %
20 % Output variables:
21 % wi = angular velocity of link i expressed in frame i
22 % wip = angular acceleration of link i expressed in frame i
23 % vi = linear velocity of the origin i expressed in frame i
24 % vip = linear acceleration of the origin i expressed in frame i
25 % vGip = linear acceleration of the center of mass of link i expressed in frame i
26
27
28 function[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,...
29                               deltai,qip,qipp,iAim1)
30
31 %computation of angular velocity omega(i)
32 wi = iAim1*wim1+qip*(1-deltai)*ki;
33
34 %computation of linear velocity v(i) of origin Oi in frame i
35 vi=iAim1*vim1+iAim1*cross(wim1,lim1)+qip*deltai*ki;
36
37 %computation of linear velocity vG(i) of centre of mass in frame i
38 vGi=vi+cross(wi,bi);
39
40 %computation of angular acceleration omegadot(i)
41 wip = iAim1*wim1p+qipp*(1-deltai)*ki+qip*(1-deltai)*cross(iAim1*wim1,ki);
42
43 %computation of linear acceleration vdot(i)

```

```

44    vip = iAim1*(vim1p+cross(wim1,lim1)+cross(wim1,cross(wim1,lim1)))+...
45        qipp.*deltai*ki+2*qip*deltai*cross(iAim1*wim1,ki);
46
47 %computation of linear acceleration vGdot(i)
48 vGip = vip+cross(wip,bi)+cross(wi,cross(wi,bi));

```

## B.6 Dynamic equations

```

1 % dynam_en02.m
2 % Function for recursive backward computation of dynamic balance of link i
3 % in terms of wrench in joint (i+1).
4 %
5 % Input variables:
6 % Fip1 = resulting force exerted in joint (i+1) by link i expressed in frame (i+1)
7 % Mip1 = resulting moment exerted in joint (i+1) by link i expressed in frame (i+1)
8 % mi = mass of link i
9 % vGip = linear acceleration of the center of mass of link i expressed in frame i
10 % wi = angular velocity of link i expressed in frame i
11 % wip = angular acceleration of link i expressed in frame i
12 % li = position vector of origin i+1 with respect to frame i expressed in frame i
13 % bi = position vector of center of mass of link i with respect to frame i expressed
14 %       in frame i
15 % Ii = inertia matrix of link i about its center of mass coordinate frame expressed
16 %       in frame i
17 % iAip1 = rotation matrix (i)A(i+1) of frame (i+1) with respect to frame i
18 %
19 % Output variables:
20 % Fi = resulting force exerted in joint i by link i-1 expressed in frame i
21 % Mi = resulting moment exerted in joint i by link i-1 expressed in frame i
22
23 function [Fi,Mi,ti]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii, iAip1, iAO)
24
25 %computation of force Fi
26 Fi = iAip1*Fip1+mi*vGip-mi*iAO*[0 0 -9.81]';
27
28 %computation of moment Mi
29 Mi = iAip1*Mip1-cross(Fi,bi)-cross(iAip1*Fip1,(li-bi))+Ii*wip+cross(wi,Ii*wi);

```