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.).

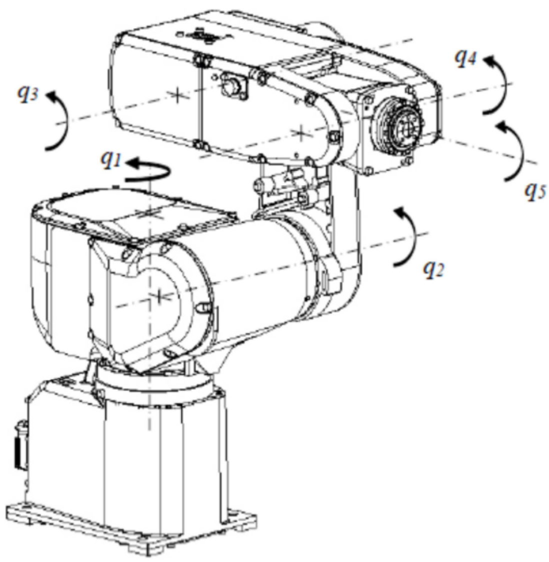


Figure 1.1: SCARA robot

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.

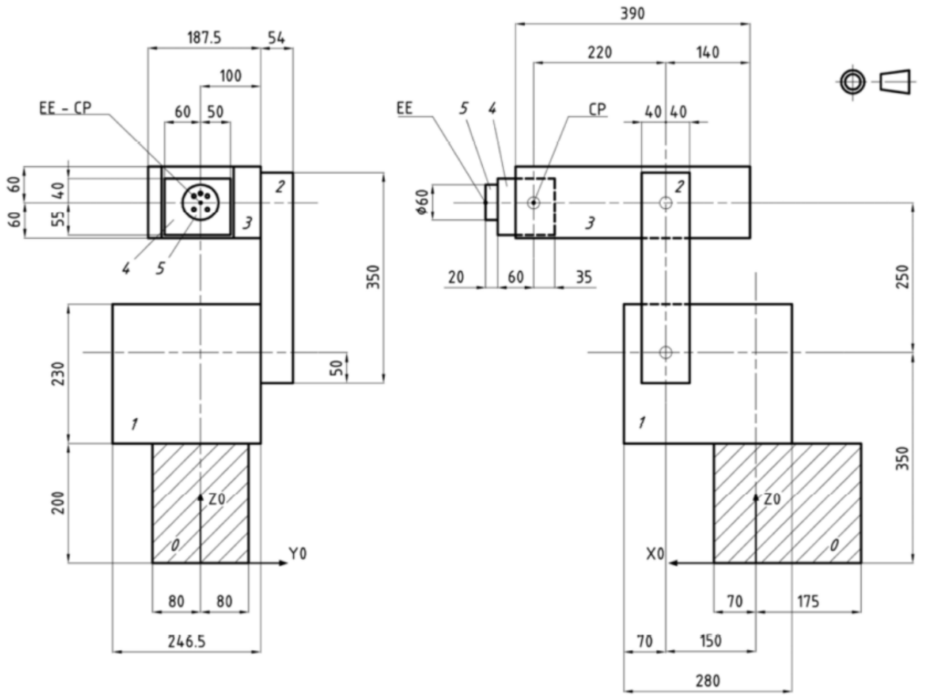


Figure 1.2: Scara robot drafting

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.

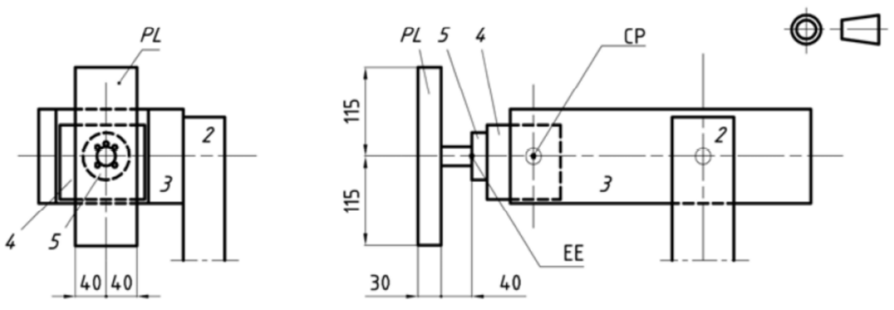


Figure 1.3: Payload drafting

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

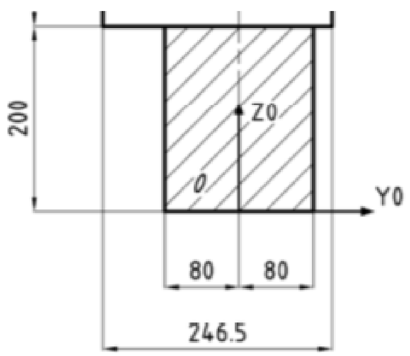
In order to make the explanation as clear as possible, for the first body all the steps will be seen in the details. For the next, which are practically identical, only the dimensions of bodies and the reference frames will be reported.

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".

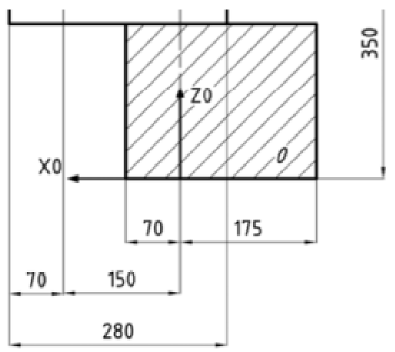
Next, proceeding to the window for bodies, it is chosen the "Box" option in solids section. After that, "New Part" is selected and the necessary dimensions to place the element on the ground are computed. Once done, Adams automatically creates a reference marker to define the position of the new body.

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 **??**.



(a) Body 0 Front view



(b) Body 0 Lateral view

Figure 1.4: Body 0 sketch

	Dimensions			Frame position		
Body	**Length**	**Height**	**Depth**	**X_i**	**Y_i**	**Z_i**
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 **??**.

	Dimensions			Frame position		
Body	**Length**	**Height**	**Depth**	$\mathbf{X_i}$	$\mathbf{Y_i}$	$\mathbf{Z_i}$
0	245	200	160	-70	0	-80
1	280	230	246.5	-220	200	-146.5
2	80	350	54	-190	300	100
3	390	120	187.5	-400	540	-87.5
4	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.

	Dimensions		Frame position		
Body	**Length**	**Radius**	$\mathbf{X_i}$	$\mathbf{Y_i}$	$\mathbf{Z_i}$
5	20	30	-430	600	0
EE	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.

	Dimensions			Frame position		
Body	**Length**	**Height**	**Depth**	$\mathbf{X_i}$	$\mathbf{Y_i}$	$\mathbf{Z_i}$
PL	30	230	80	-520	485	-40

Table 1.4: Payload (and relative frame) values

The resulting wire frame is shown in figure **??**

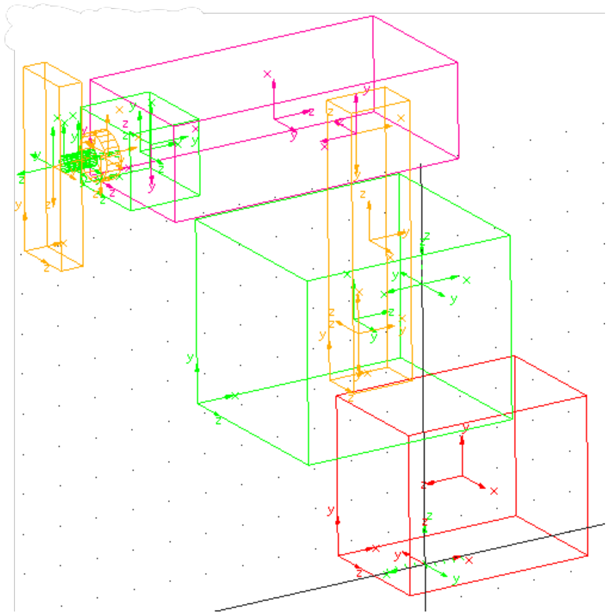


Figure 1.5: wire frame visualization of 5 D.O.F. robot in Adams

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.

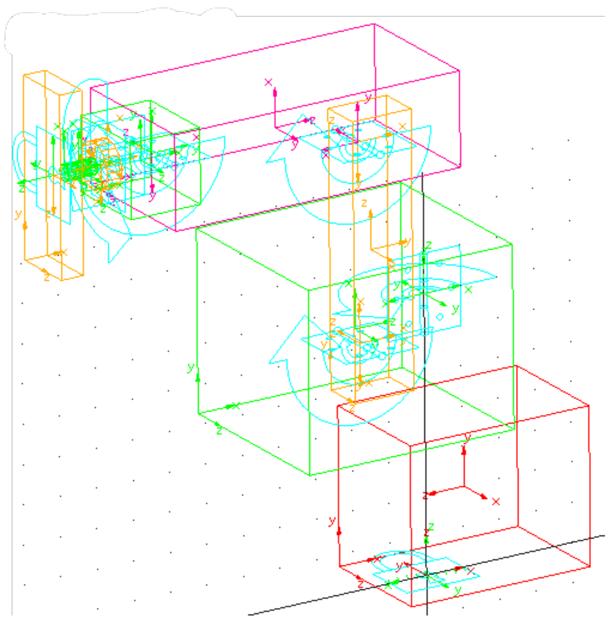


Figure 1.6: wire frame visualization of 5 D.O.F. robot with joints

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.

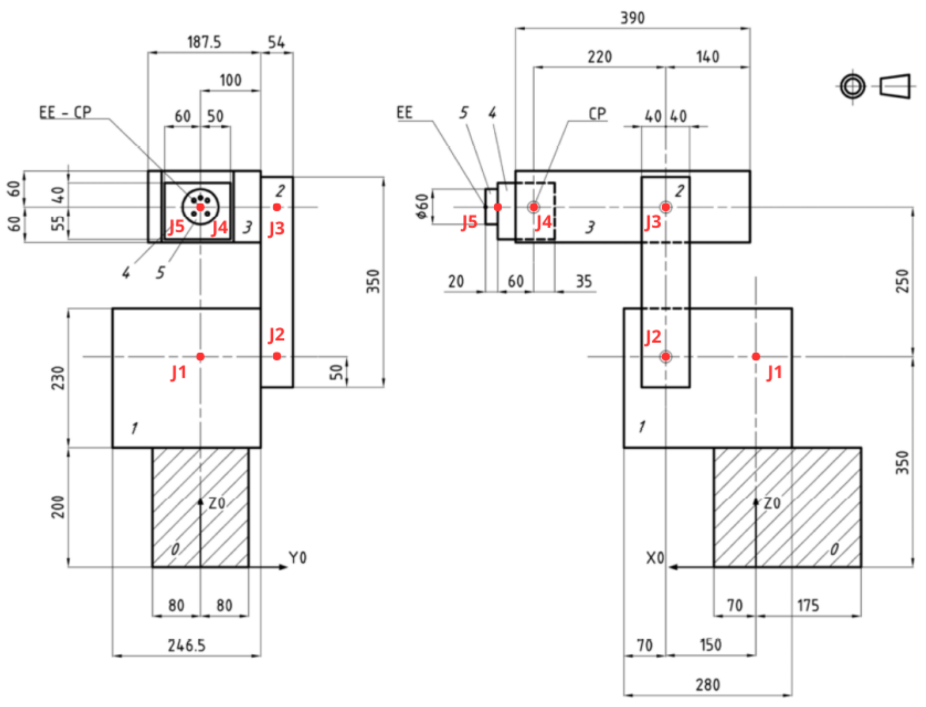


Figure 1.7: Joints location

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	**1**	0	350	0	180	90	0
1 - 2	**2**	-150	350	100	90	180	0
2 - 3	**3**	-150	600	100	180	180	0
3 - 4	**4**	-370	600	0	90	180	0
4 - 5	**5**	-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.

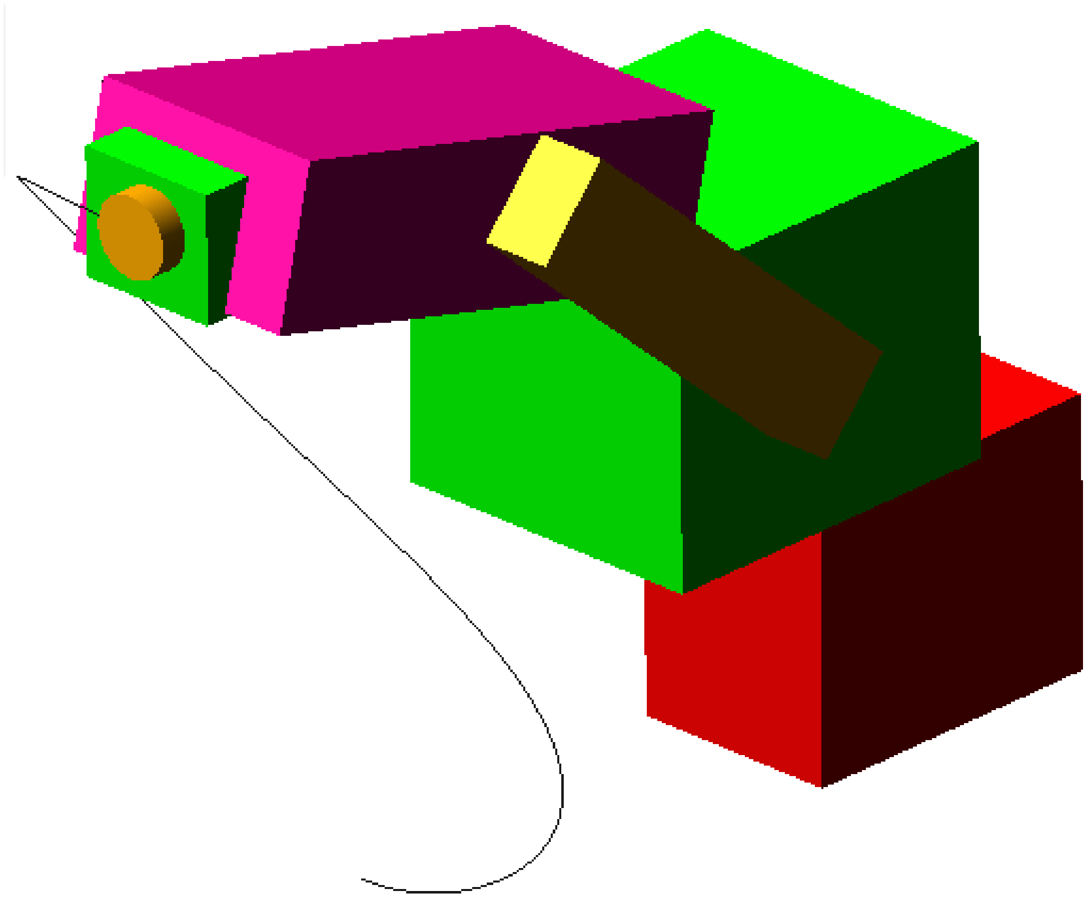


Figure 1.8: 5 D.O.F. robot modeled in Adams with E.E. trajectory

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

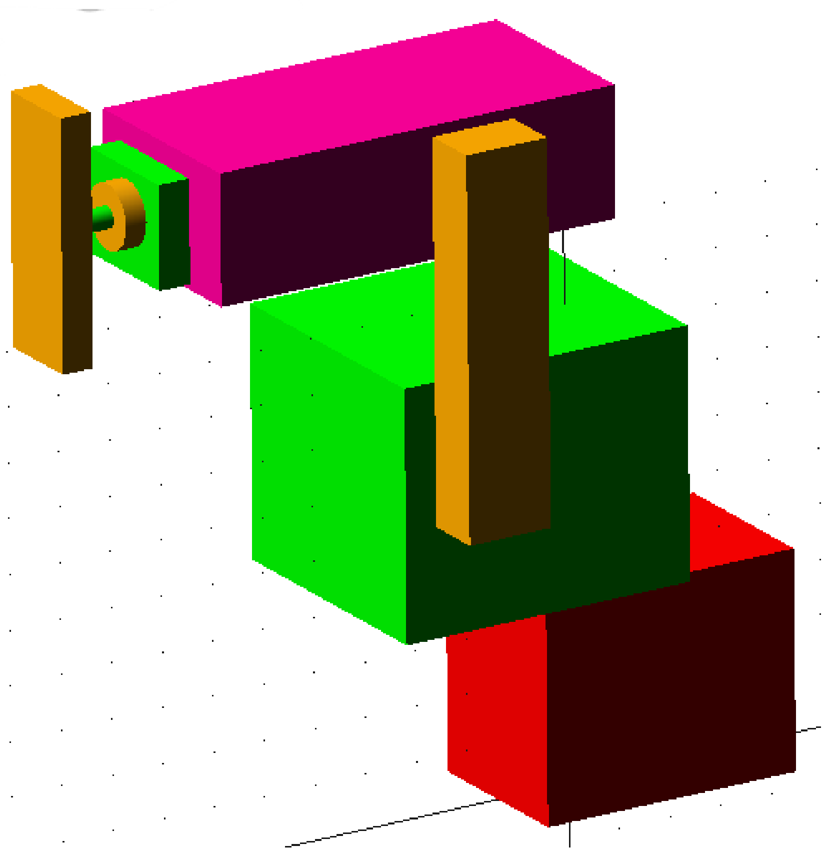


Figure 1.9: 5 D.O.F. robot modeled in Adams with payload

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° about axis y_1.
4. Rotate and plot the initial triangle of -90° about axis y_0.
5. Rotate and plot the triangle from the last position of 90° 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.

$$\mathbf{P_{O_1}} = \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix} \tag{2.1}$$

The previous data are imported in the Matlab code as following

```
%triangle points

P1 = [0 1 1]';
P2 = [0 7 1]';
P3 = [0 4 7]';

%positioning matrix of triangle

T0o = [P1,P2,P3,P1;...
        1, 1, 1, 1];
```

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

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

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

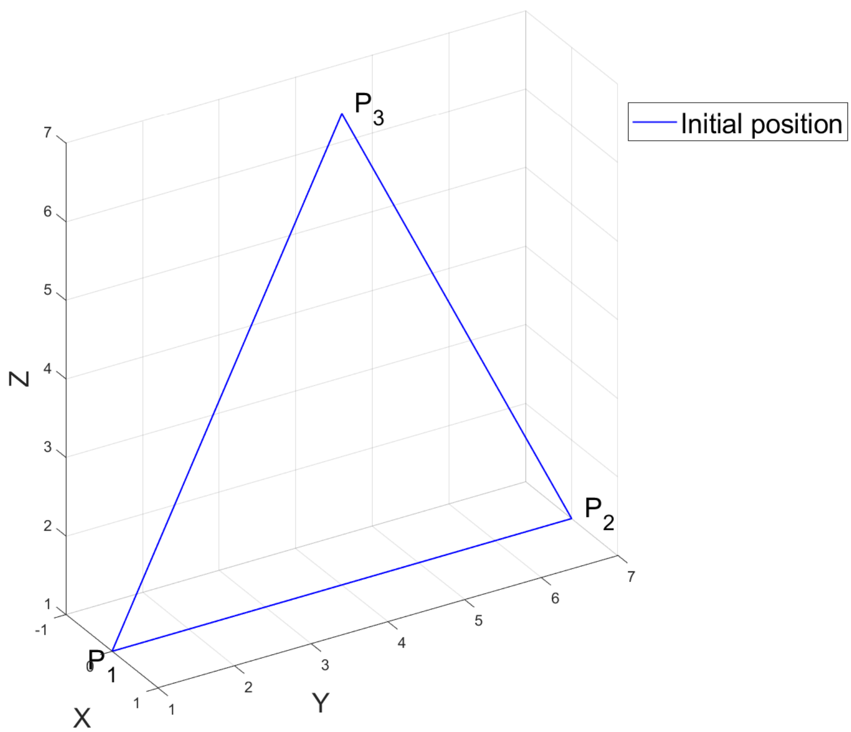


Figure 2.1: Schematization of initial conditions

```
% plot settings
triang_lw = 1;
legend_fs = 18;
label_fs = 18;
text_fs = 18;

X_trang = [P1(1,1) P2(1,1) P3(1,1) P1(1,1)];
Y_trang = [P1(2,1) P2(2,1) P3(2,1) P1(2,1)];
Z_trang = [P1(3,1) P2(3,1) P3(3,1) P1(3,1)];

%initial condition plot
figure(1)

plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
legend("Initial position",fontsize=legend_fs)
axis equal
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 ${}^0A_1$:

$$\mathbf{{}^0A_1} = \begin{bmatrix} i_{1x} & j_{1x} & k_{1x} \\ i_{1y} & j_{1y} & k_{1y} \\ i_{1z} & j_{1z} & k_{1z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

The implementation on Matlab is reported.

```
%define versors of reference systems

scale = 1;

%versor sys 0

i0 = [1 0 0]' * scale;
j0 = [0 1 0]' * scale;
k0 = [0 0 1]' * scale;

%versors sys 1

i1 = [0 1 0]' * scale;
j1 = [0 0 1]' * scale;
k1 = [1 0 0]' * scale;

```

```
%positioning matrix A01

A01 = [i1 j1 k1];

```

2.3.2 Versors of reference systems plot

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

```
%plot versors
figure(2)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
hold on

quiver3(PO1(1),PO1(2),PO1(3),i1(1),i1(2),i1(3),Color=[1 0.5 0.5],...
LineWidth=vector_lw)
hold on
quiver3(PO1(1),PO1(2),PO1(3),j1(1),j1(2),j1(3),Color = "#77AC30",...
LineWidth=vector_lw)
hold on
quiver3(PO1(1),PO1(2),PO1(3),k1(1),k1(2),k1(3),Color = "#4DBEEE",...
LineWidth=vector_lw)
xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
text(PO1(1)-0.2,PO1(2)-0.5,PO1(3)-0.5,"P_{O_1}",...
```

```
"FontSize",text_fs)
legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
fontsize=legend_fs)
axis equal
grid on
```

The result is shown in figure **??**.

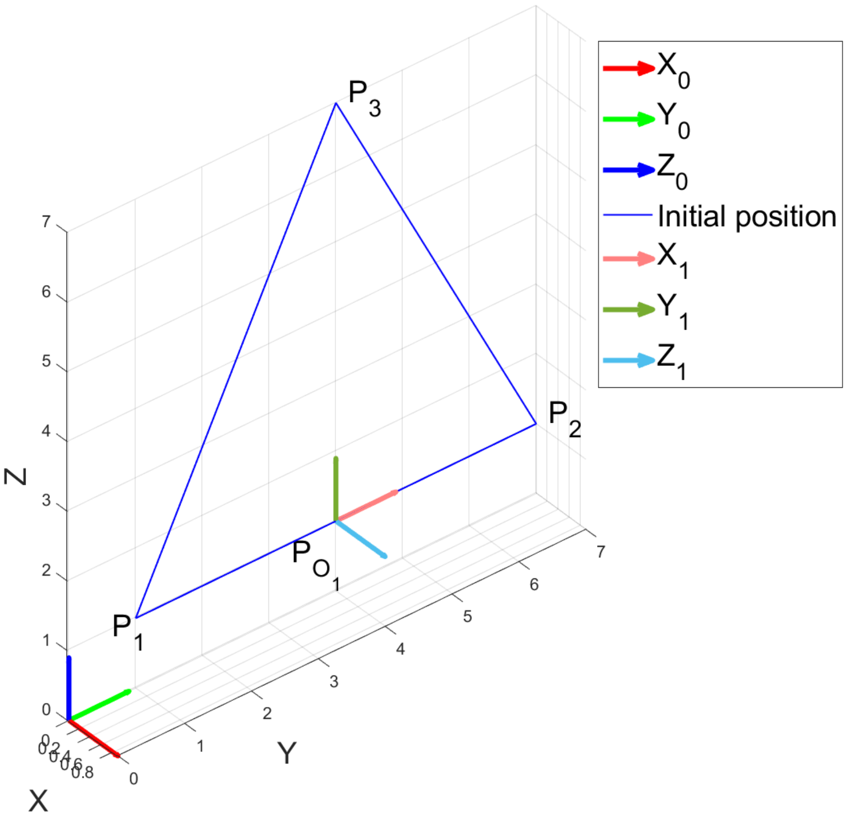


Figure 2.2: Versors of reference systems plot

2.3.3 Rotation of 90° about *y₁*

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 **??**, **??**).

$$
{}^{\mathbf{0}}\hat{\mathbf{A}}_{\mathbf{1}} = \begin{bmatrix} & {}^0A_1 & & \begin{matrix} P_{O_1x} \\ P_{O_1y} \\ P_{O_1z} \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}
$$

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

$$\mathbf{R\hat{O}T}(\mathbf{y},\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

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

```
function [hom_rotation_matrix] = rotYo(theta)

hom_rotation_matrix = [cos(theta)       0    sin(theta)     0;...
                           0            1        0          0;...
                      -1*sin(theta)     0    cos(theta)     0;...
                           0            0        0          1];
end
```

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

$$\mathbf{R\hat{O}T}(\mathbf{y_1},\theta_1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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 $R\hat{O}T(y,\theta)$.

$${}^0\hat{B}_1 =^0 \hat{A}_1 \cdot R\hat{O}T(y_1, 90) \tag{2.6}$$

Afterwards, it is necessary to move the reference system of ${}^0\hat{T}$ to the mobile reference system $0_1x_1y_1z_1$, to obtain ${}^1\hat{T}$.

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

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

$${}^0\hat{T'} =^0 \hat{B}_1 \cdot^1 \hat{T} \tag{2.8}$$

The result is shown in the figure **??**:

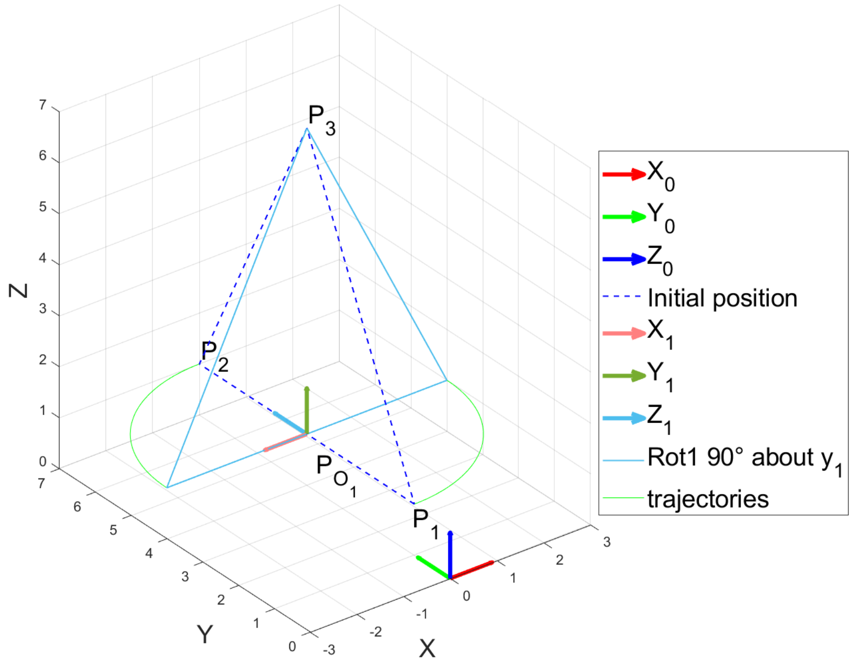


Figure 2.3: Rotation of the triangle about y_1 of 90 degrees

The implementation in Matlab is now reported:

```
% homogeneus matrix A01o

A01o = [A01,P01;...
        0 0 0 1];

%rotation of triangle about y1 of 90

theta1 = pi/2;

rotYo90 = rotYo(theta1);

B01o=  A01o*rotYo90;

A10o = inv(A01o);

T1o = A10o * T0o;

T0_rotated1 = B01o * T1o;

```

```
%trajection of rotation

angles = linspace(0,theta1,1000);

for i=1:length(angles)
    trajP1_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P1;1]);
    trajP2_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P2;1]);
    trajP3_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P3;1]);
end
```

```
%plot rotation 1
figure(3)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
hold on

quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
Color = "#4DBEEE",LineWidth=triang_lw)
hold on

```

```
plot3(trajP1_rot1(:,1),trajP1_rot1(:,2),trajP1_rot1(:,3),"g")
hold on
plot3(trajP2_rot1(:,1),trajP2_rot1(:,2),trajP2_rot1(:,3),"g")
hold on
plot3(trajP3_rot1(:,1),trajP3_rot1(:,2),trajP3_rot1(:,3),"g"))

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
text(PO1(1)-0.2,PO1(2)-0.5,PO1(3)-0.5,"P_{O_1}","FontSize",text_fs)
legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
"Rot1 90° about y_1","trajectory",fontsize=legend_fs)
axis equal
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 $\theta_2 = -90°$ is computed .

$$\mathbf{R\hat{O}T}(\mathbf{y_0}, \boldsymbol{\theta_2}) = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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:

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

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 $R\hat{O}T(y_0, \theta_2)$.

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

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

$$^0\hat{T}'' =^0 \hat{C}_2 \cdot^2 \hat{T} \tag{2.10}$$

In the next figure **??** the result is shown:

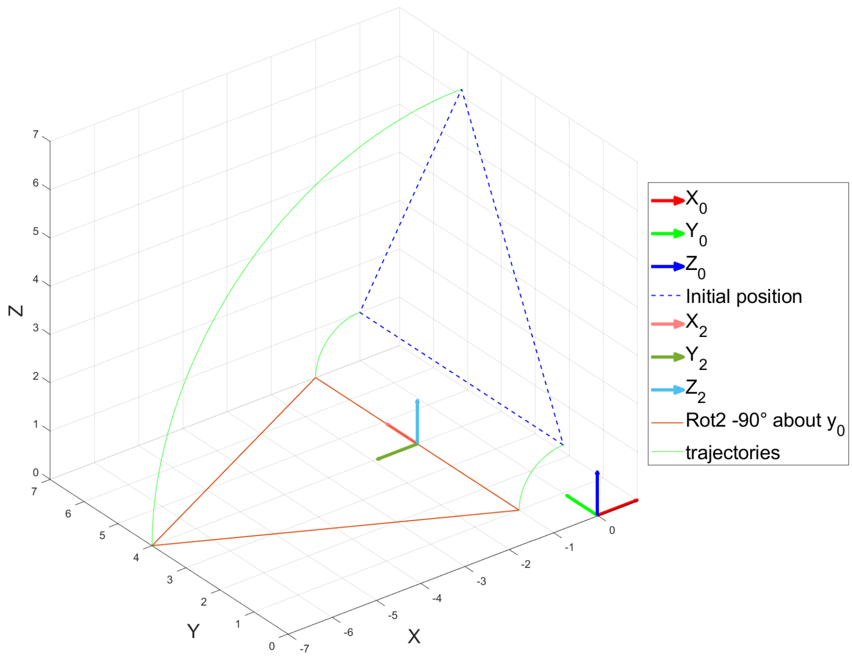


Figure 2.4: Rotation of the triangle about y_0 of -90 degrees

In the following lines the Matlab implementation is reported.

```
%rotation of triangle about y0 oh theta2
theta2 = -pi/2;

A20o = A10o;

T2o = T1o;

A02o = A01o;

rotYom90 = rotYo(theta2);

C02o=  rotYom90 * A02o;

T0_rotated2 = C02o * T2o;

%trajection of rotation 2

angles_2 = linspace(0,theta2,1000);

for i=1:length(angles)
    trajP1_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P1;1]);
    trajP2_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P2;1]);
    trajP3_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P3;1]);
end
```

```
%plot rotation 2
figure(4)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
hold on
```

```

quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
Color= "#D95319",LineWidth=triang_lw)
hold on

plot3(trajP1_rot2(:,1),trajP1_rot2(:,2),trajP1_rot2(:,3),"g")
hold on
plot3(trajP2_rot2(:,1),trajP2_rot2(:,2),trajP2_rot2(:,3),"g")
hold on
plot3(trajP3_rot2(:,1),trajP3_rot2(:,2),trajP3_rot2(:,3),"g")

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)

legend("X_0","Y_0","Z_0","Initial position","X_2","Y_2","Z_2",...
"Rot2 -90° about y_0","trajectories",fontsize=legend_fs)
axis equal
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.

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

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

```
function [hom_rotated_matrix_X] = rotXo(theta)

hom_rotated_matrix_X = [     1           0            0            0;...
                             0      cos(theta)  -sin(theta)        0;...
                             0      sin(theta)  cos(theta)         0;...
                             0           0            0            1];
end
```

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

$$\mathbf{R\hat{O}T}(\mathbf{x_2},\boldsymbol{\theta_3}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$^0\hat{D}_2 =^0 \hat{C}_2 \cdot R\hat{O}T(x_2, 90) \tag{2.11}$$

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.

$$^2\hat{C}_0 =^0 \hat{C}_2^{-1} \tag{2.12}$$

$$^2\hat{T''} =^2 \hat{C}_0 \cdot^0 \hat{T''} \tag{2.13}$$

$$^0\hat{T'''} =^0 \hat{D}_2 \cdot^2 \hat{T''} \tag{2.14}$$

The result is shown in the following figure.

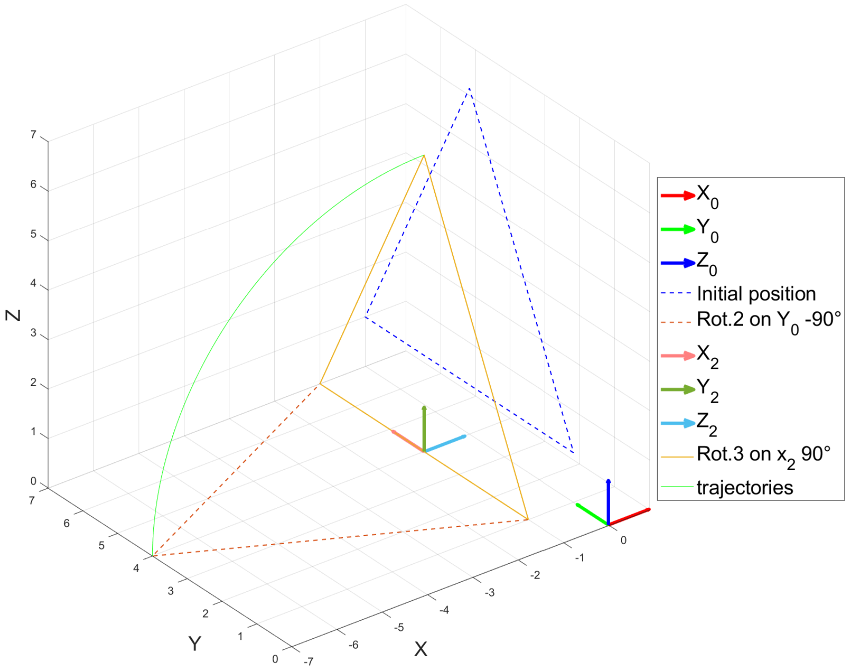


Figure 2.5: Rotation of the triangle about x_1 of 90 degrees

The Matlab implementation is now reported.

```
%rotation of triangle about x2 oh theta3
theta3 = pi/2;

rotXo90 = rotXo(theta3);

D02o = C02o * rotXo90;

C20o = inv(C02o);

T2 = C20o * T0_rotated2;

T0_rotated3 = D02o * T2;

%trajection of rotation 2

angles_3 = linspace(0,theta3,1000);

for i=1:length(angles)
    trajP1_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
```

```
    T0_rotated2(:,1));
    trajP2_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
    T0_rotated2(:,2));
    trajP3_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
    T0_rotated2(:,3));
end
```

```
%plot rotation 2
figure(5)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
hold on

plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),"--",...
LineWidth= triang_lw, Color= "#D95319")
hold on

quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,1),D02o(2,1),D02o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,2),D02o(2,2),D02o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,3),D02o(2,3),D02o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
LineWidth= triang_lw, Color="#EDB120")
hold on
```

```

plot3(trajP1_rot3(:,1),trajP1_rot3(:,2),trajP1_rot3(:,3),"g")
hold on
plot3(trajP2_rot3(:,1),trajP2_rot3(:,2),trajP2_rot3(:,3),"g")
hold on
plot3(trajP3_rot3(:,1),trajP3_rot3(:,2),trajP3_rot3(:,3),"g")

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)

legend("X_0","Y_0","Z_0","Initial position","Rot.2 on Y_0 -90°",...
"X_2","Y_2","Z_2","Rot.3 on x_2 90°","trajectories",fontsize=legend_fs)
axis equal
grid on
```

2.4 Final plot

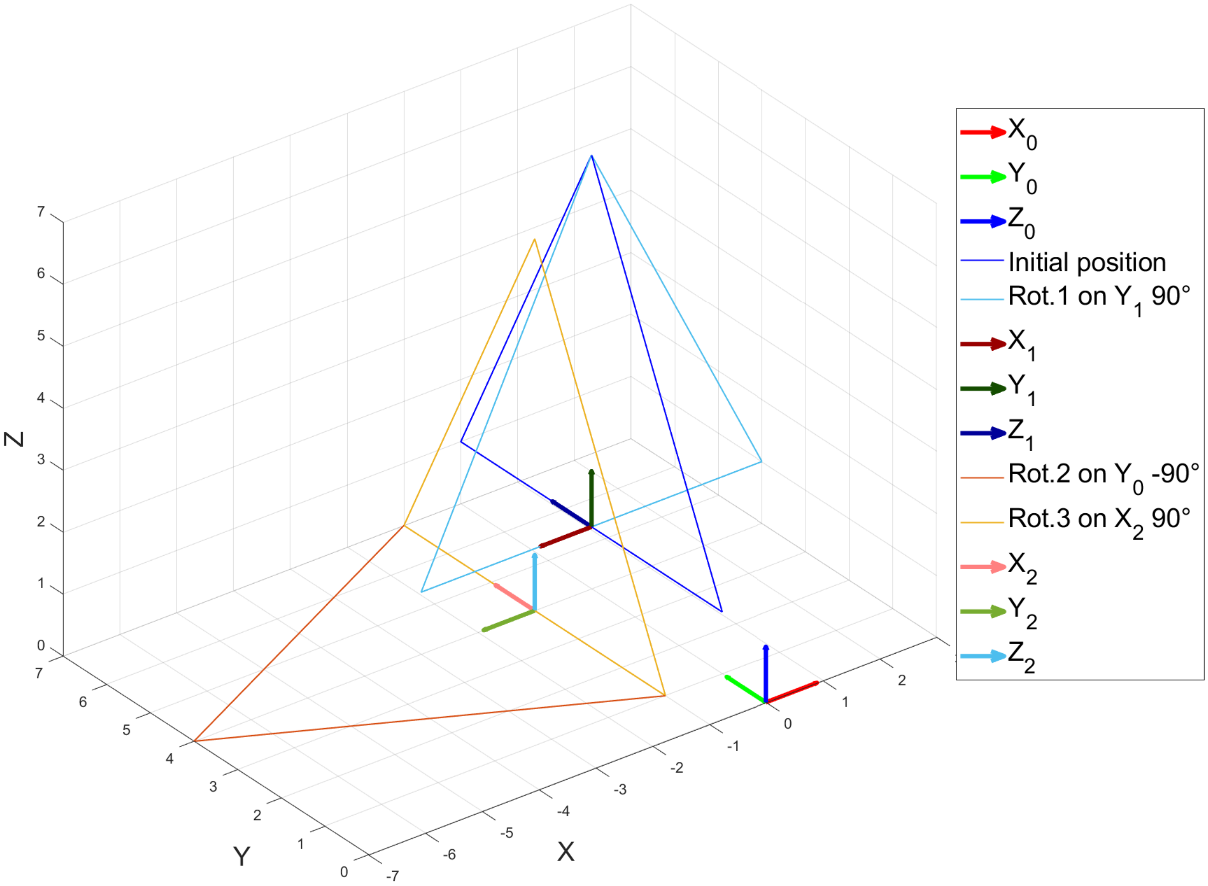


Figure 2.6: Final plot

Report 3

SCARA robot

3.1 Introduction

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

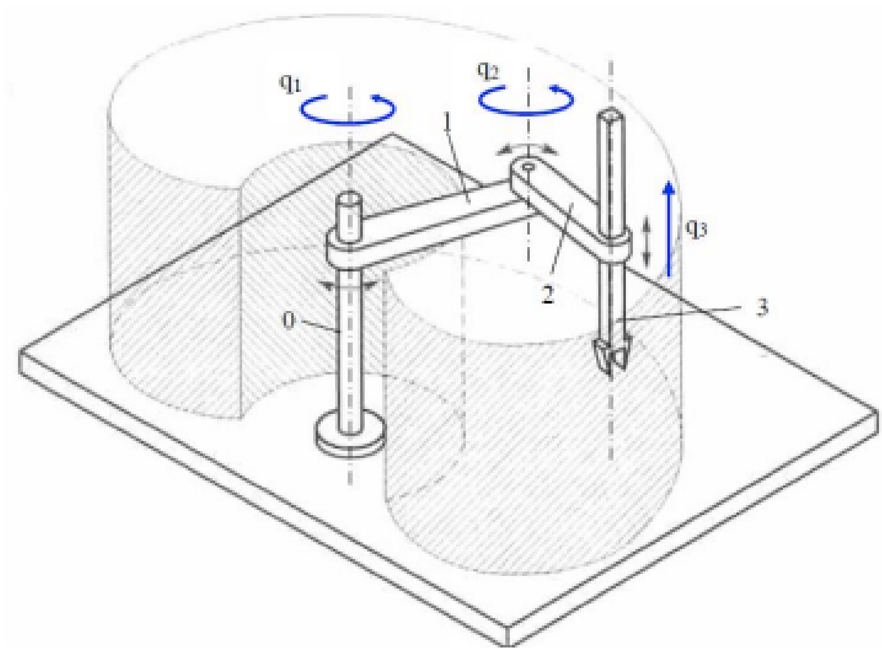


Figure 3.1: SCARA robot

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 traslatory joint (3) and an end effector.

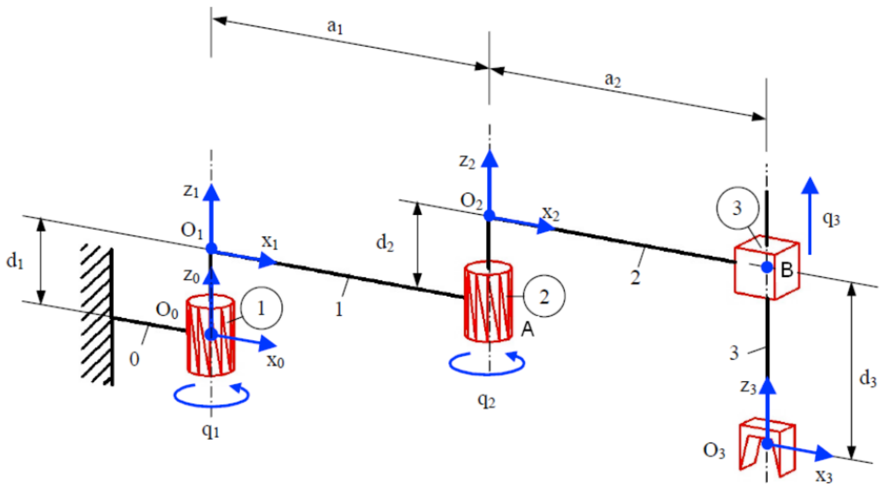


Figure 3.2: SCARA schematization

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.

```
%% DATA
load("joints.mat")

P0O0 = [0 0 0]';

q1 = M(:,1);    %[rad]
q2 = M(:,2);    %[rad]
```

```
q3 = M(:,3);    %[mm]

d1 = 25;    %[mm]
d2 = 15;    %[mm]
d3 = 10;    %[mm]

a1 = 50;    %[mm]
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.

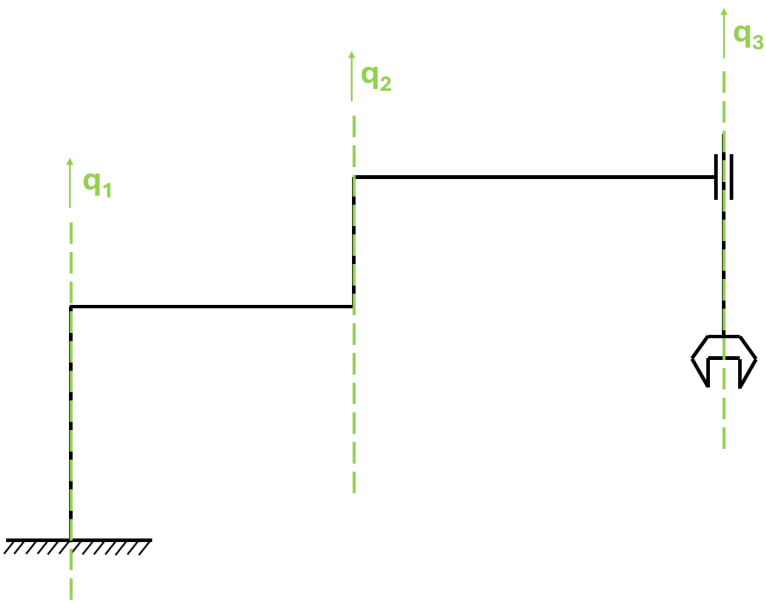


Figure 3.3: Joints schematization of the SCARA robot

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 **??**.

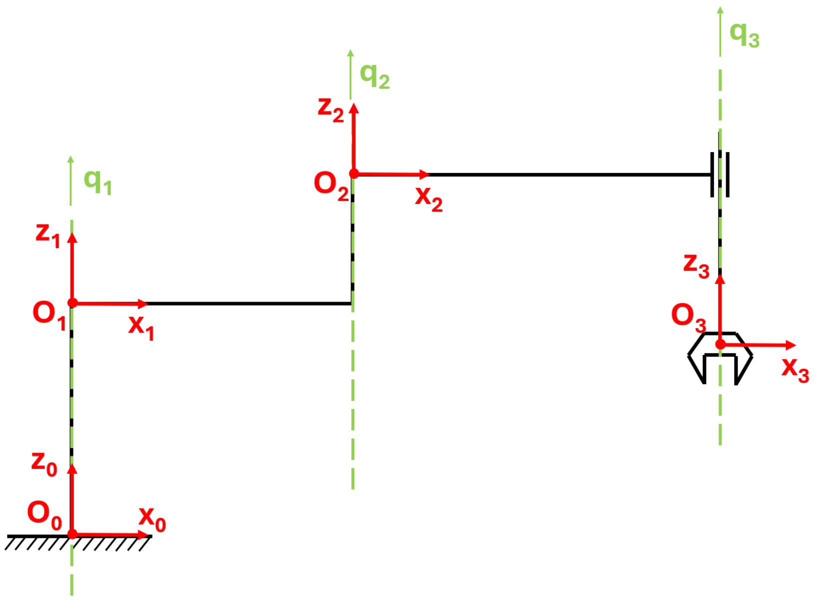


Figure 3.4: Reference systems adopted to schematize the SCARA robot

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.

```
%% POSITIONING PROBLEM

i=1;                    %initial position
%i=length(q1);          %final position

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
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:

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

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:

```
A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

%ficticius matrix for body 3 joint
A03o_joint = A02o;
```

```
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 $i = lenght(q1)$ it is in final position.

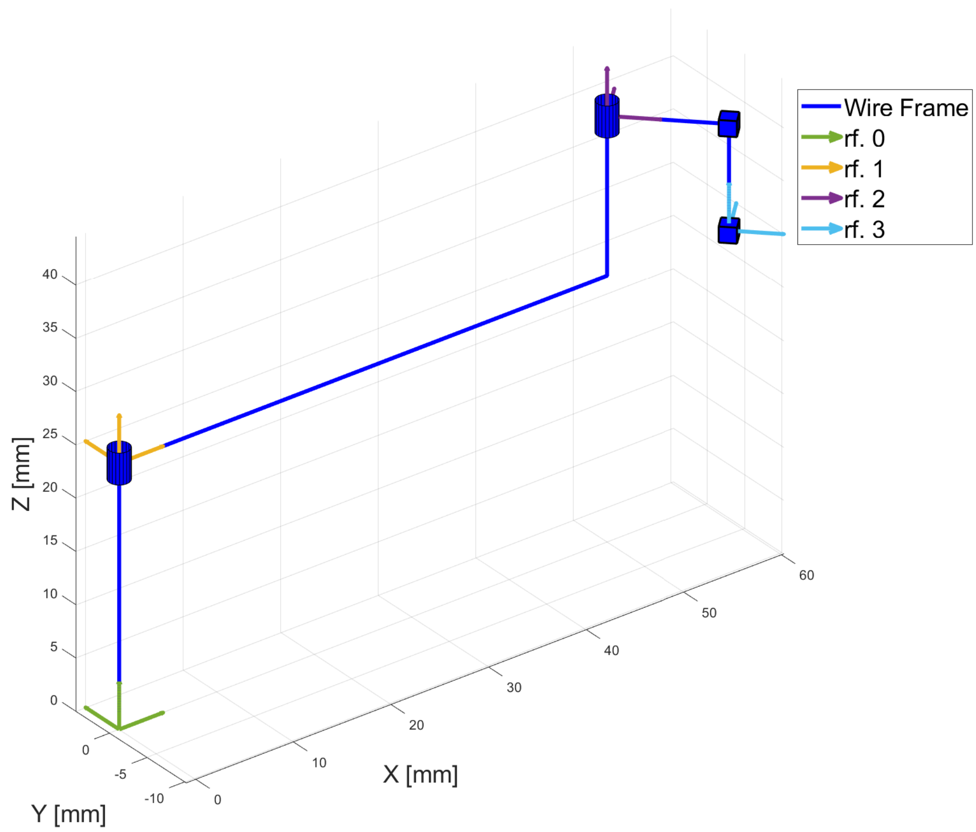


Figure 3.5: Plot of the wire frame in the initial position

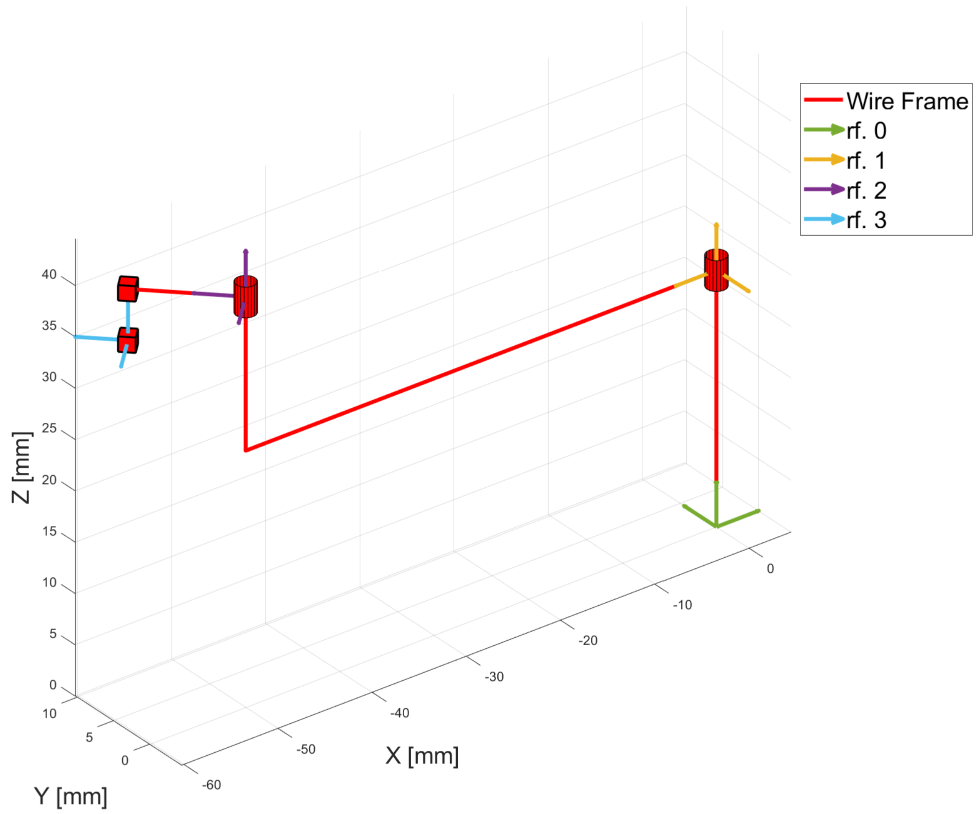


Figure 3.6: Plot of the wire frame in the final position

```
%Plot setup
radius = 1;
height_c = 3;
frame_lw = 3;
s = 5;
ref_lw = 3;
text_size = 18;
rf0_color = "#77AC30";
rf1_color = "#EDB120";
rf2_color = "#7E2F8E";
rf3_color = "#4DBEEE";
wf_color = "b";
%wf_color = "r";

figure(1)
%wire frame
wf = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),wf_color,...
     LineWidth=frame_lw);
hold on
```

```

%plot ref. 0
rf0_1 = quiver3(P000(1),P000(2),P000(3),1*s,0*s,0*s,...
        LineWidth=ref_lw,Color=rf0_color); %x0
hold on
rf0_2 = quiver3(P000(1),P000(2),P000(3),0*s,1*s,0*s,...
        LineWidth=ref_lw,Color=rf0_color); %y0
hold on
rf0_3 = quiver3(P000(1),P000(2),P000(3),0*s,0*s,1*s,...
        LineWidth=ref_lw,Color=rf0_color); %z0
hold on

%plot ref. 1
rf1_1 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,1)*s,A01o(2,1)*s,A01o(3,1)*s,...
        LineWidth=ref_lw,Color=rf1_color); %x1
hold on
rf1_2 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,2)*s,A01o(2,2)*s,A01o(3,2)*s,...
        LineWidth=ref_lw,Color=rf1_color); %y1
hold on
rf1_3 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,3)*s,A01o(2,3)*s,A01o(3,3)*s,...
        LineWidth=ref_lw,Color=rf1_color); %z1
hold on

%plot ref. 2
rf2_1 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,1)*s,A02o(2,1)*s,A02o(3,1)*s,...
        LineWidth=ref_lw,Color=rf2_color); %x2
hold on
rf2_2 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,2)*s,A02o(2,2)*s,A02o(3,2)*s,...
        LineWidth=ref_lw,Color=rf2_color); %y2
hold on
rf2_3 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,3)*s,A02o(2,3)*s,A02o(3,3)*s,...
```

```
        LineWidth=ref_lw,Color=rf2_color); %z2
hold on

%plot ref. 3
rf3_1 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,1)*s,A03o(2,1)*s,A03o(3,1)*s,...
        LineWidth=ref_lw,Color=rf3_color); %x3
hold on
rf3_2 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,2)*s,A03o(2,2)*s,A03o(3,2)*s,...
        LineWidth=ref_lw,Color=rf3_color); %y3
hold on
rf3_3 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,3)*s,A03o(2,3)*s,A03o(3,3)*s,...
        LineWidth=ref_lw,Color=rf3_color); %z3
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf_color);
hold on
joint_rev_01(radius,height_c,20,A02o,wf_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf_color,A03o);

%plot settings
grid on
axis equal
xlabel("X [mm]","FontSize",text_size)
ylabel("Y [mm]","FontSize",text_size)
zlabel("Z [mm]","FontSize",text_size)
legend([wf rf0_1 rf1_1 rf2_1 rf3_1],...
{"Wire Frame","rf. 0","rf. 1","rf. 2","rf. 3"},fontsize=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.

```
% TRAJECTORY
for j = 1:length(q1)
    A01oj = denhar_en01(0,0,d1,q1(j));
    A12oj = denhar_en01(0,a1,d2,q2(j));
    A23oj = denhar_en01(0,a2,-d3+q3(j),0);
    A03oj = A01oj * A12oj * A23oj;
    if j==1
        P303o_init = inv(A03oj)*A03oj(:,4);
    end
    P003j = A03oj * P303o_init;
    traj_EF(:,j) = P003j;
end
```

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

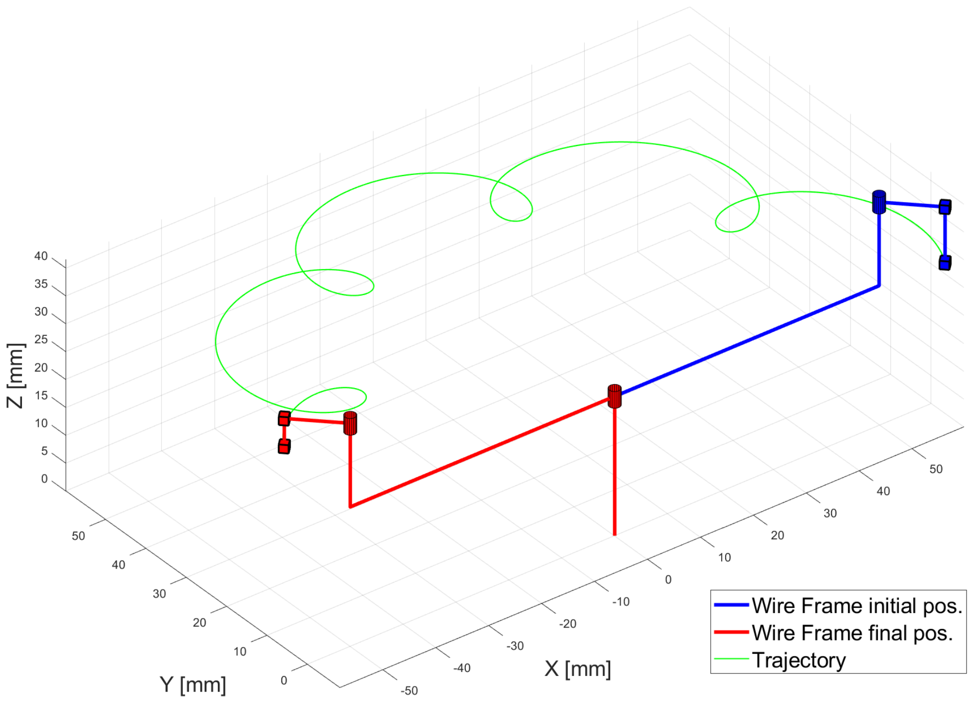


Figure 3.7: Trajectory of the end effector with the initial and final position of the SCARA robot

Below the Matlab implementation is reported.

```
%%TOTAL PLOT
i=1;

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
A23o = denhar_en01(0,a2,-d3+q3(i),0);

A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

A03o_joint = A03o;

A03o_joint(3,4) = A02o(3,4);

%Plot setup
radius = 1;
height_c = 3;
frame_lw = 3;
traj_lw = 1;

ref_lw = 3;
text_size = 18;
wf1_color = "b";
wf2_color = "r";
traj_color = "g";



figure(2)
```

```
%wire frame
wf1 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
      wf1_color,LineWidth=frame_lw);
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf1_color);
hold on
joint_rev_01(radius,height_c,20,A02o,wf1_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf1_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf1_color,A03o);
hold on

i=length(q1);

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
A23o = denhar_en01(0,a2,-d3+q3(i),0);

A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

A03o_joint = A03o;

A03o_joint(3,4) = A02o(3,4);

figure(2)
```

```
%wire frame
wf2 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
      wf2_color,LineWidth=frame_lw);
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf2_color);
hold on
joint_rev_01(radius,height_c,20,A02o,wf2_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf2_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf2_color,A03o);
hold on
traj = plot3(traj_EF(1,:),traj_EF(2,:),traj_EF(3,:),...
       LineWidth=traj_lw,Color=traj_color);

%plot settings
grid on
axis equal
xlabel("X [mm]","FontSize",text_size)
ylabel("Y [mm]","FontSize",text_size)
zlabel("Z [mm]","FontSize",text_size)
legend([wf1 wf2 traj],{"Wire Frame initial pos.",...
       "Wire Frame final pos.","Trajectory"},...
       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.

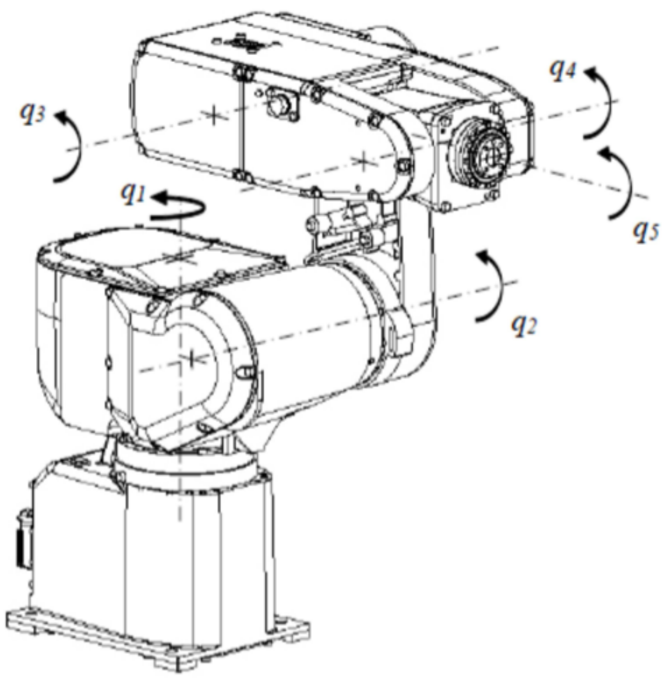


Figure 4.1: 5 DoF robot

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_en.02.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.

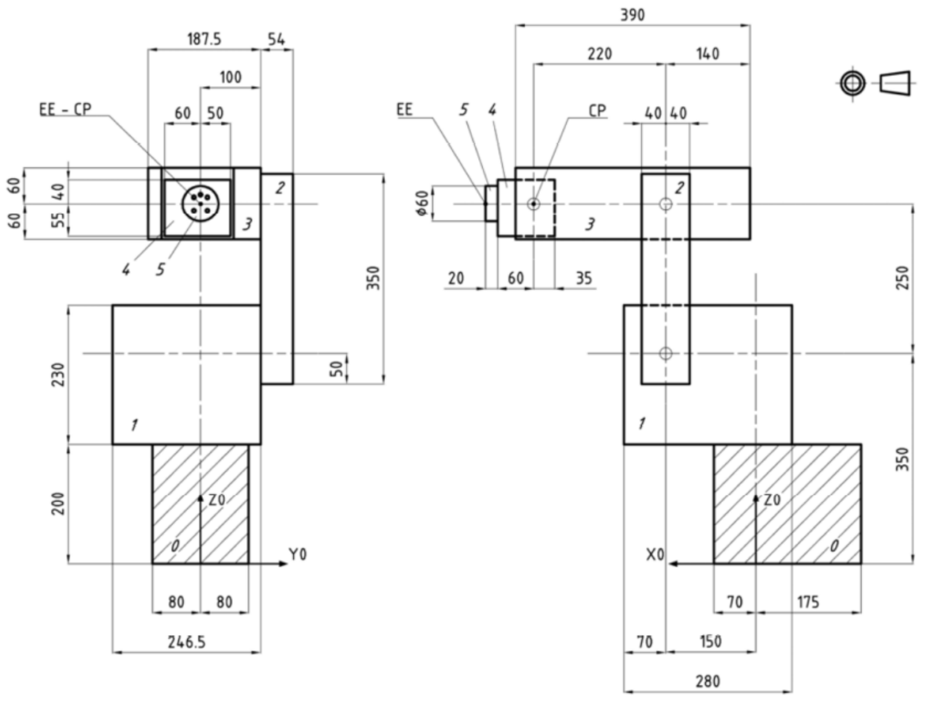


Figure 4.2: 5 DoF robot schematization

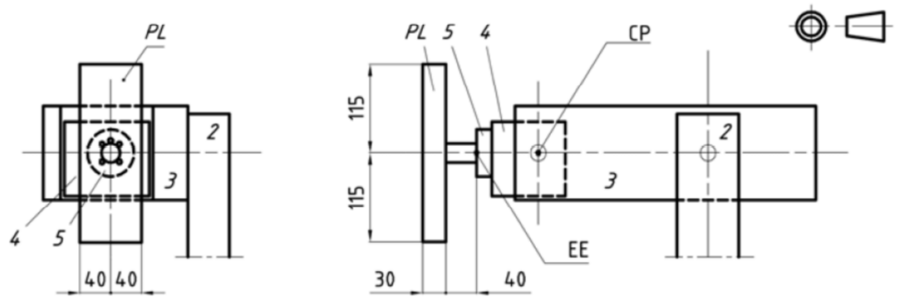


Figure 4.3: Payload schematization

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 kg and dimensions as in figure 3.

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

Body	0	1	2	3	4	5	PL
Mass [kg]	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.

```
%% DATA

data = load("trajectory.mat");

a = 200;
b = 150;
c = 150;
d = 100;
e = 250;
f = 100;
g = 220;
h = 60;
i = 20;
l = 40;

q1 = data.q1vec;
q2 = data.q2vec;
q3 = data.q3vec;
q4 = data.q4vec;
q5 = data.q5vec;

q1d = data.q1dvec.*(pi/180);
q2d = data.q2dvec.*(pi/180);
q3d = data.q3dvec.*(pi/180);
q4d = data.q4dvec.*(pi/180);
q5d = data.q5dvec.*(pi/180);

q1dd = data.q1ddvec.*(pi/180);
```

```
q2dd = data.q2ddvec.*(pi/180);
q3dd = data.q3ddvec.*(pi/180);
q4dd = data.q4ddvec.*(pi/180);
q5dd = data.q5ddvec.*(pi/180);

time = data.timevec;

qPd = zeros(length(time),1);
qPdd = zeros(length(time),1);

CM0_0 = [-140 0 100]';
CM1_1 = [80 0 35]';
CM2_2 = [125 0 -27]';
CM3_3 = [55 0 100]';
CM4_4 = [-7.5 -12.5 5]';
CM5_5 = [0 0 10]';
CMP_P = [0 0 15]';

m0 = 18;
m1 = 10.5;
m2 = 2;
m3 = 6;
m4 = 2;
m5 = 0.5;
mP = 1;

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 **??**.

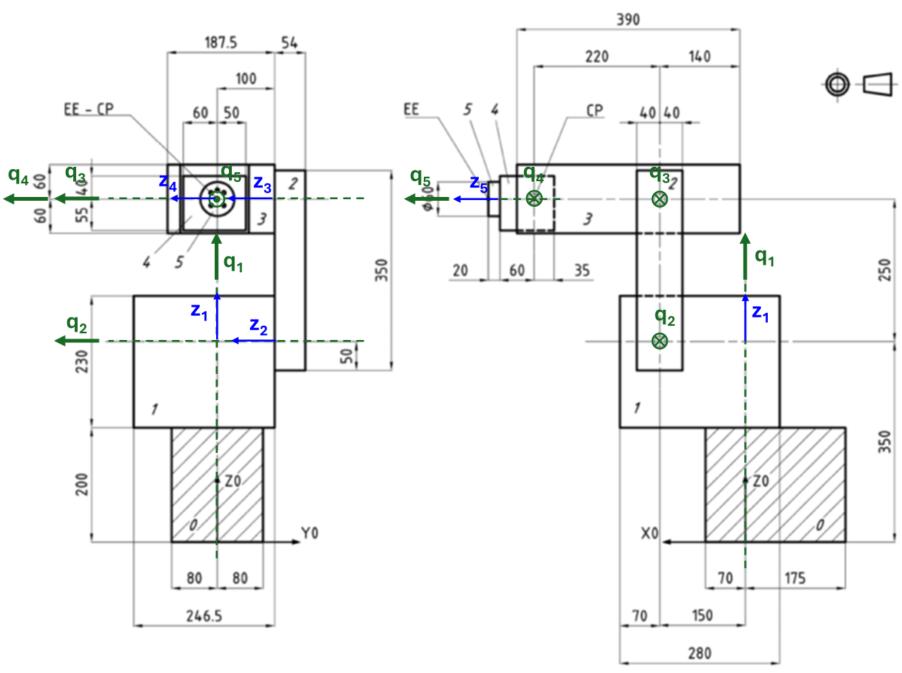


Figure 4.4: Joints and z axis according to Denavit-Hartenberg convention

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 in 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.

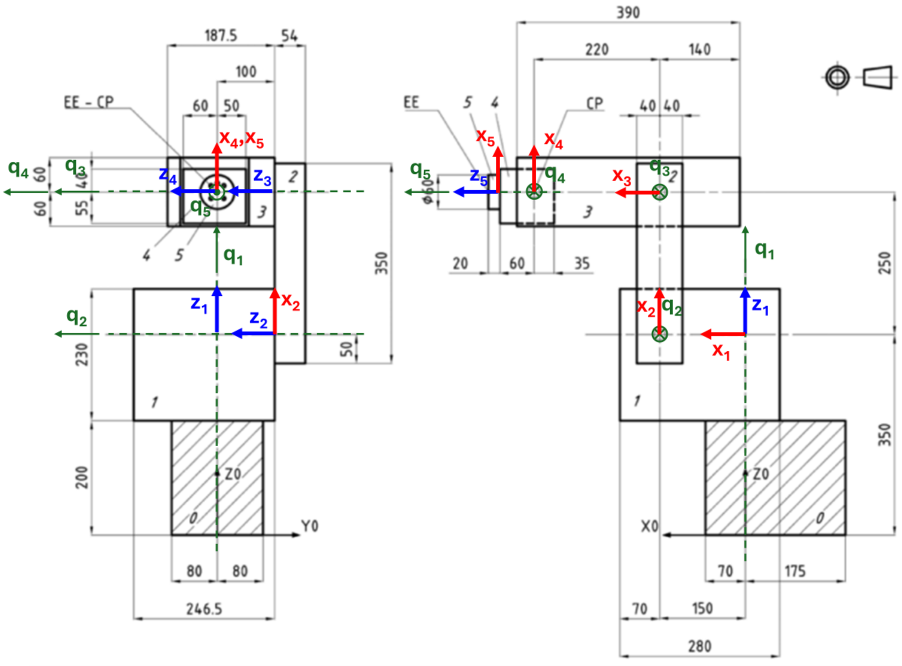


Figure 4.5: Denavit-Hartenberg convention applied in the 5 D.O.F. robot

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

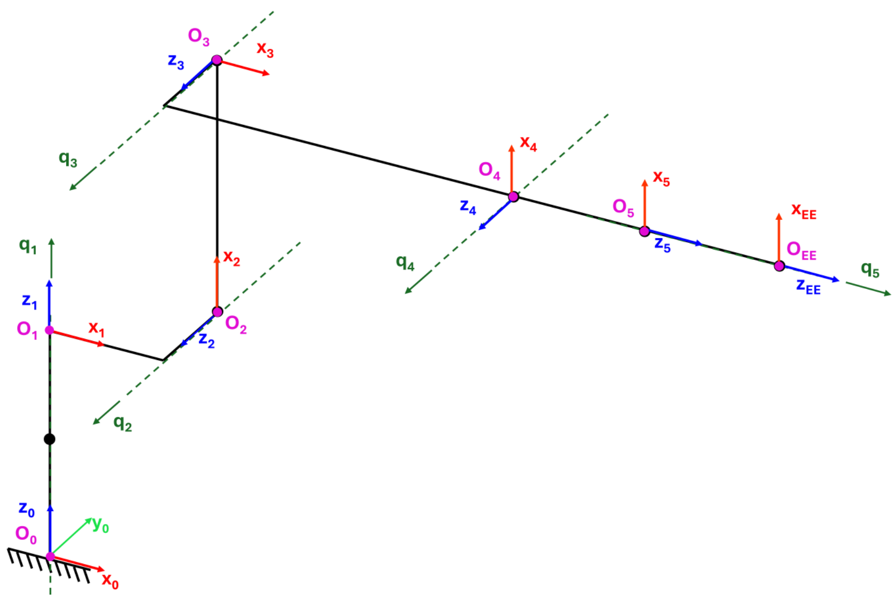


Figure 4.6: 3D sketch of wire frame with bodies reference system

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	$alpha$	a	d	$theta$	orientation matrix
1	0	0	$a+b$	q_1	${}^0\hat{A}_1$
2	90°	c	$-d$	$90° + q_2$	${}^1\hat{A}_2$
3	0	e	0	$-90° + q_3$	${}^2\hat{A}_3$
4	0	g	f	$90° + q_4$	${}^3\hat{A}_4$
5	90°	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:

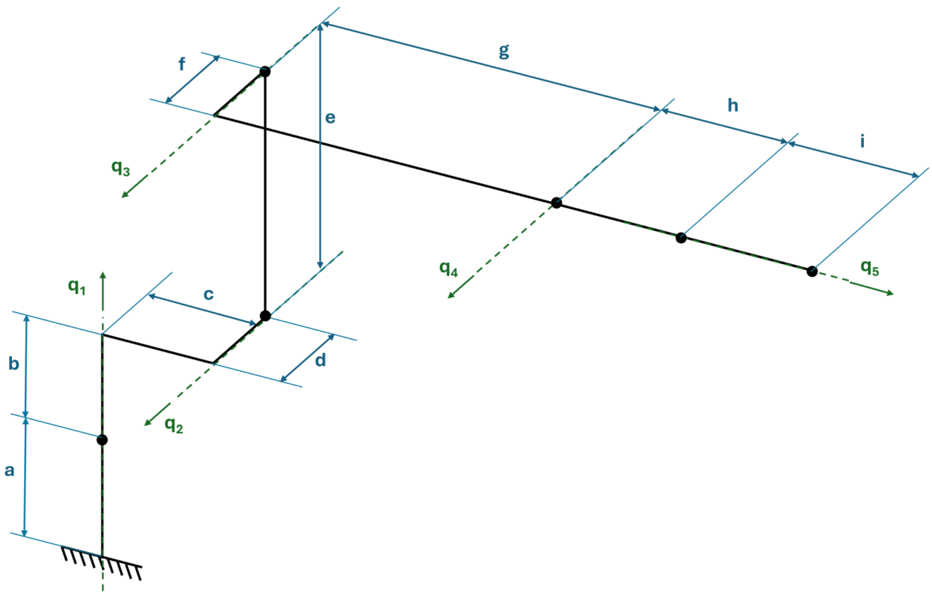


Figure 4.7: Distances used in the Denavit-Hartenberg convention

	a	b	c	d	e	f	g	h	i	l
Distance [mm]	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 $O_0x_0y_0z_0$

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

The positioning problem is solved on Matlab using the function denhar_en01.m and the result is shown in the following script.

```
for j = 1:length(time)


    % POSITIONING PROBLEM

    %Denavit-Huntberg covention
    A01o = denhar_en01( 0,      0,    a+b,     deg2rad(q1(j)));
    A12o = denhar_en01(pi/2,    c,    -d,    pi/2+deg2rad(q2(j)));
    A23o = denhar_en01( 0,      e,     0,   -pi/2+deg2rad(q3(j)));
    A34o = denhar_en01( 0,      g,     f,    pi/2+deg2rad(q4(j)));
    A45o = denhar_en01(pi/2,    0,     h,      deg2rad(q5(j)));
    A5EEo = denhar_en01( 0,      0,     i,             0);
    A5Po = denhar_en01( 0,      0,     l+i,             0);

    A02o = A01o * A12o;
    A03o = A02o * A23o;
    A04o = A03o * A34o;
    A05o = A04o * A45o;
    A0EEo = A05o * A5EEo;
    A0Po = A05o * A5Po;


    %wire frame in the initial position
    if j == 1
        wf_init = [0 A01o(1,4)  A01o(1,4) A01o(1,4)+c  A02o(1,4)...
                   A03o(1,4) A03o(1,4)   A04o(1,4) A05o(1,4) A0EEo(1,4);
                   0 A01o(2,4)   A01o(2,4) A01o(2,4)    A02o(2,4)...
                   A03o(2,4) A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
                   0 A01o(3,4)-b A01o(3,4) A01o(3,4)    A02o(3,4)...
                   A03o(3,4) A03o(3,4)   A04o(3,4) A05o(3,4) A0EEo(3,4)];

        A_joint1 = A01o;
        A_joint1(3,4) = A01o(3,4)-b;

        initial_cond = {wf_init,A_joint1,A02o,A03o,A04o,A05o,A0EEo}';
    end

    %wire frame in the final position
    if j == length(time)
        wf_fin = [0 A01o(1,4)   A01o(1,4) A01o(1,4)+c  A02o(1,4)...
                A03o(1,4) A03o(1,4)   A04o(1,4) A05o(1,4) A0EEo(1,4);
                  0 A01o(2,4)   A01o(2,4) A01o(2,4)    A02o(2,4)...
                  A03o(2,4) A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
                  0 A01o(3,4)-b A01o(3,4) A01o(3,4)    A02o(3,4)...
```

```
                    A03o(3,4) A03o(3,4)   A04o(3,4) A05o(3,4) A0EEo(3,4)];

        A_joint1 = A01o;
        A_joint1(3,4) = A01o(3,4)-b;

        final_cond = {wf_fin,A_joint1,A02o,A03o,A04o,A05o,A0EEo}';
    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.

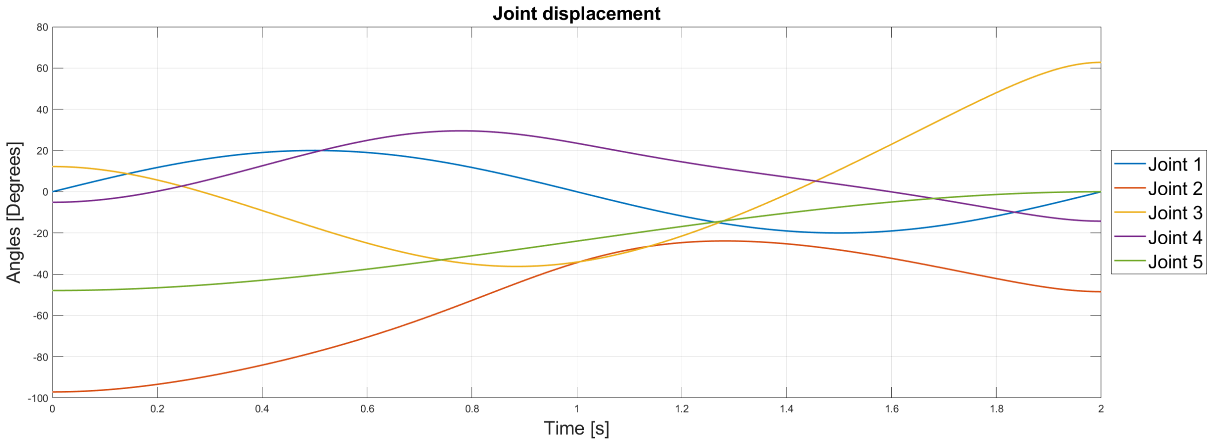


Figure 4.8: Joint angles plot

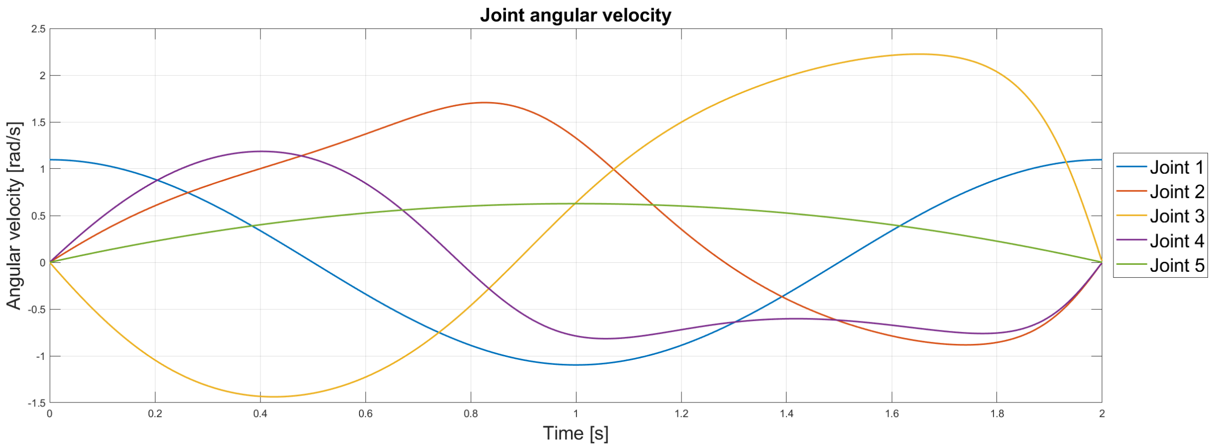


Figure 4.9: Joint angular velocities plot

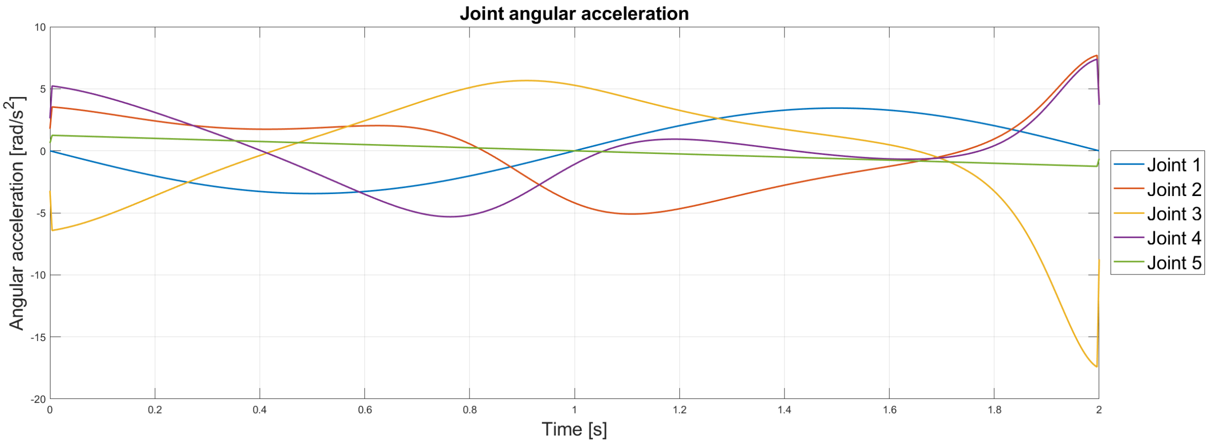


Figure 4.10: Joint angular accelerations plot

The Matlab script to obtain them is reported below.

```
%requirement 2 - ang,vel,acc plot
figure(3)
plot(time,q1,time,q2,time,q3,time,q4,time,q5,LineWidth=kin_lw)
hold on
grid on
title("Joint displacement",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
       "eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Angles [Degrees]",FontSize=fs)

figure(4)
plot(time,q1d,time,q2d,time,q3d,time,q4d,time,q5d,LineWidth=kin_lw)
grid on
title("Joint angular velocity",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
       "eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Angular velocity [rad/s]",FontSize=fs)

figure(5)
plot(time,q1dd,time,q2dd,time,q3dd,time,q4dd,time,q5dd,LineWidth=kin_lw)
grid on
title("Joint angular acceleration",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location",...
       "eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
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.

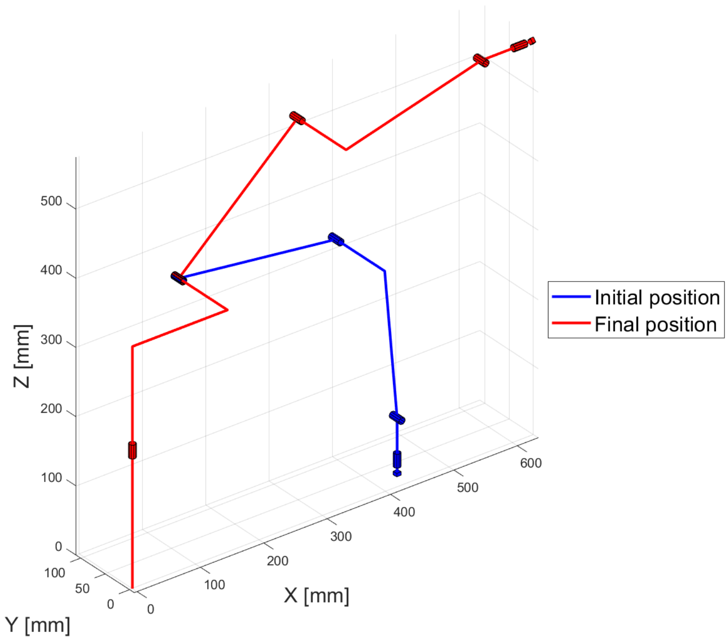


Figure 4.11: Wire frame (initial and final position)

The Matlab script to implement it is reported below.

```
%% Requirement 3 - PLOT POSITIONS AND TRAJECTORY

figure(2)
wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...
Color="b",LineWidth=wf_lw); %Initial position
hold on
wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...
Color="r",LineWidth=wf_lw); %Final position
hold on

t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
hold on
t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory

joint_rev_01(r_j,l_j,10,initial_cond{2},"b");
joint_rev_01(r_j,l_j,10,initial_cond{3},"b");
joint_rev_01(r_j,l_j,10,initial_cond{4},"b");
joint_rev_01(r_j,l_j,10,initial_cond{5},"b");
joint_rev_01(r_j,l_j,10,initial_cond{6},"b");
joint_rev_01(r_j,r_j,4,initial_cond{7},"b");

```

```
joint_rev_01(r_j,l_j,10,final_cond{2},"r");
joint_rev_01(r_j,l_j,10,final_cond{3},"r");
joint_rev_01(r_j,l_j,10,final_cond{4},"r");
joint_rev_01(r_j,l_j,10,final_cond{5},"r");
joint_rev_01(r_j,l_j,10,final_cond{6},"r");
joint_rev_01(r_j,r_j,4,final_cond{7},"r");

legend([wf_plot(1) wf_plot(2) t1 t2],{"Initial position", "Final position",...
"EE Trajectory", "CP Trajectory"},fontsize=fs)
xlabel("X [mm]","FontSize",fs)
ylabel("Y [mm]","FontSize",fs)
zlabel("Z [mm]","FontSize",fs)
axis equal
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.

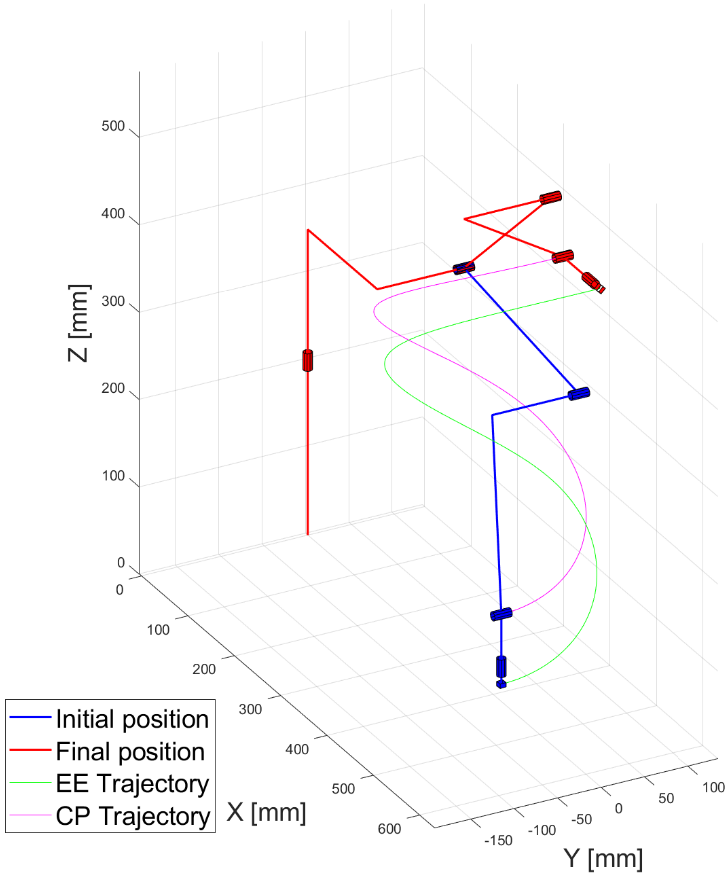


Figure 4.12: CP and EE trajectory plot

The Matlab implementation is reported.

```
%% Requirement 4 - Trajectory of CP and EE
for j = 1:length(time)

    %trajectory EE
    traj_EE(:,j) = A0EEo(1:3,4);

    %trajectory EE
    traj_CP(:,j) = A04o(1:3,4);

end

figure(2)
wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...
Color="b",LineWidth=wf_lw); %Initial position
hold on
wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...
Color="r",LineWidth=wf_lw); %Final position
hold on

t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
hold on
t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory

joint_rev_01(r_j,l_j,10,initial_cond{2},"b");
joint_rev_01(r_j,l_j,10,initial_cond{3},"b");
joint_rev_01(r_j,l_j,10,initial_cond{4},"b");
joint_rev_01(r_j,l_j,10,initial_cond{5},"b");
joint_rev_01(r_j,l_j,10,initial_cond{6},"b");
joint_rev_01(r_j,r_j,4,initial_cond{7},"b");

joint_rev_01(r_j,l_j,10,final_cond{2},"r");
joint_rev_01(r_j,l_j,10,final_cond{3},"r");
joint_rev_01(r_j,l_j,10,final_cond{4},"r");
joint_rev_01(r_j,l_j,10,final_cond{5},"r");
joint_rev_01(r_j,l_j,10,final_cond{6},"r");
joint_rev_01(r_j,r_j,4,final_cond{7},"r");

legend([wf_plot(1) wf_plot(2) t1 t2],{"Initial position", "Final position",...
"EE Trajectory", "CP Trajectory"},fontsize=fs)
xlabel("X [mm]","FontSize",fs)
ylabel("Y [mm]","FontSize",fs)
zlabel("Z [mm]","FontSize",fs)
axis equal
```

```
grid on)
```

4.3.5 Verify the kinematic function

The code of the Matlab function kinem_en.02.m is reported below.

```
% kinem_en02.m
% Function for recursive forward computation of kinematic variables of link i
% in terms of kinematic variables of link (i-1) and of d.o.f. in joint i.
%
% input variables:
% wim1 = angular velocity of link (i-1) expressed in frame (i-1)
% wim1p = angular acceleration of link (i-1) expressed in frame (i-1)
% vim1 = linear velocity of the origin of link (i-1) expressed in frame (i-1)
% vim1p = linear acceleration of the origin of link (i-1) expressed in frame (i-1)
% ki = unit vector k of joint i axis expressed in frame i
% lim1 = position vector of the origin i with respect to frame (i-1) expressed
%        in frame (i-1)
% bi = position vector of center of mass of link i with respect to frame i
%      expressed in frame i
% deltai = joint i parameter (1/0)
% qip = joint i d.o.f. velocity
% qipp = joint i d.o.f. acceleration
% iAim1 = rotation matrix (i)A(i-1) of frame (i-1) with respect to frame i
%
% Output variables:
% wi = angular velocity of link i expressed in frame i
% wip = angular acceleration of link i expressed in frame i
% vi = linear velocity of the origin i expressed in frame i
% vip = linear acceleration of the origin i expressed in frame i
% vGip = linear acceleration of the center of mass of link i expressed in frame i


function [wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,...
ki,lim1,bi,deltai,qip,qipp,iAim1)

%computation of angular velocity omega(i)
wi = iAim1*wim1+qip*(1-deltai)*ki;

%computation of linear velocity v(i) of origin Oi in frame i
vi=iAim1*vim1+iAim1*cross(wim1,lim1)+qip*deltai*ki;

%computation of linear velocity vG(i) of centre of mass in frame i
vGi=vi+cross(wi,bi);
```

```
%computation of angular acceleration omegadot(i)
wip = iAim1*wim1p+qipp*(1-deltai)*ki+qip*(1-deltai)*...
    cross(iAim1*wim1,ki);

%computation of linear acceleration vdot(i)
vip = iAim1*(vim1p+cross(wim1p,lim1)+cross(wim1,cross(wim1,lim1)))+...
    qipp.*deltai*ki+2*qip*deltai*cross(iAim1*wim1,ki);

%computation of linear acceleration vGdot(i)
vGip = vip+cross(wip,bi)+cross(wi,cross(wi,bi));
```

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

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

$$ {}^i\vec{v}_i = {}^i\vec{v}_{i-1} + {}^i\vec{\omega}_{i-1} \times {}^i l_{i-1} + \dot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i \tag{4.3} $$

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

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

$$ \dot{\vec{v}}_i = {}^i\dot{\vec{v}}_{i-1} + {}^i\dot{\vec{\omega}}_{i-1} \times {}^i l_{i-1} + {}^i\vec{\omega}_{i-1} \times \left[{}^i\vec{\omega}_{i-1} \times {}^i l_{i-1}\right] + \ddot{q}_i \cdot \delta_i \cdot {}^i\vec{k}_i + 2 \cdot \dot{q}_i \cdot \delta_i \cdot {}^i\vec{\omega}_{i-1} \times {}^i\vec{k}_i \tag{4.6} $$

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

Function inputs as $\vec{\omega}_{i-1}$, $\dot{\vec{\omega}}_{i-1}$, $\vec{v}_{i-1}$, $\dot{\vec{v}}_{i-1}$, 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.

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

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

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

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

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 **??**.

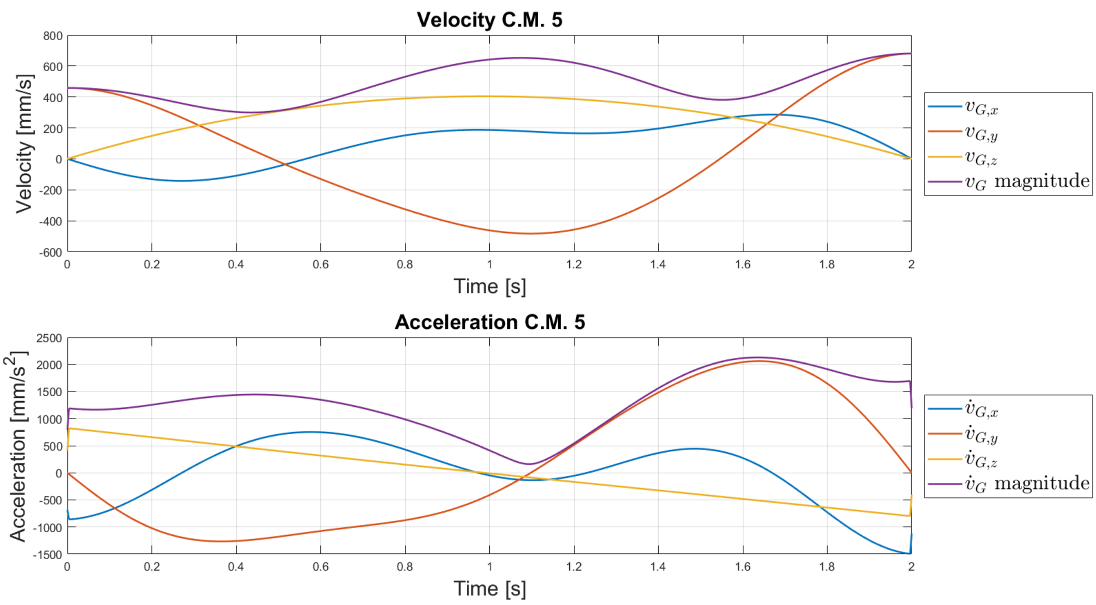


Figure 4.13: Kinematics of centers of mass 5

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

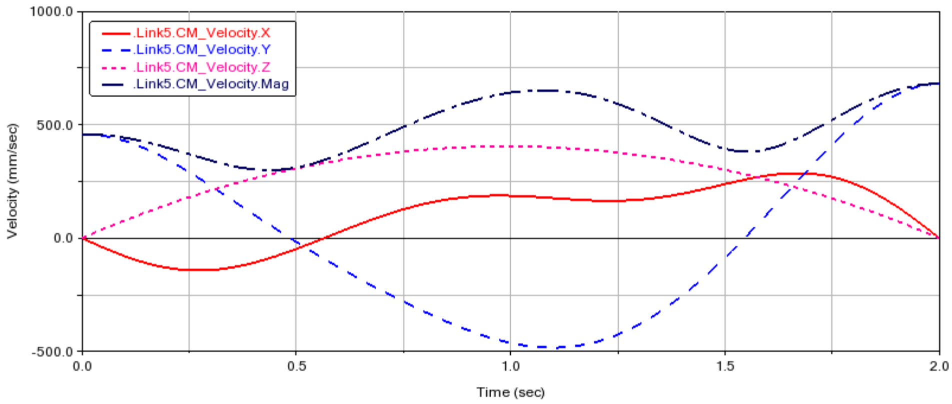


Figure 4.14: Adams simulation of body 5 CM velocity

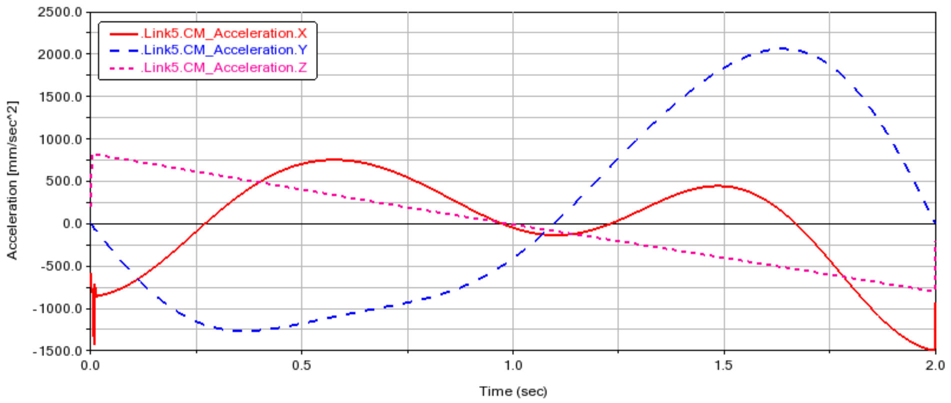


Figure 4.15: Adams simulation of body 5 CM acceleration

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

```
%% Requirement 6 - Speeds and accelerations of CM 5

for j = 1:length(time)

    % body 1
    OO01 = A01o(1:3,4);
    A10 = inv(A01o);
    A10 = A10(1:3,1:3);
    k1 = [0 0 1]';
    [w1(:,j),w1p(:,j),v1(:,j),vG1(:,j),v1p(:,j),vG1p(:,j)]=...
        kinem_en02([0 0 0]',[0 0 0]',[0 0 0]',[0 0 0]',k1,OO01,...
        CM1_1(1:3),delta(1),q1d(j),q1dd(j),A10);
```

```

    % body 2
    O1O2 = A12o(1:3,4);
    A21 = inv(A12o);
    A21 = A21(1:3,1:3);
    k2 = [0 0 1]';
    [w2(:,j),w2p(:,j),v2(:,j),vG2(:,j),v2p(:,j),vG2p(:,j)]=...
        kinem_en02(w1(1:3,j),w1p(1:3,j),v1(1:3,j),v1p(1:3,j),k2,O1O2,...
        CM2_2(1:3),delta(2),q2d(j),q2dd(j),A21);

    % body 3
    k3 = [0 0 1]';
    O2O3 = A23o(1:3,4);
    A32 = inv(A23o);
    A32 = A32(1:3,1:3);
    [w3(:,j),w3p(:,j),v3(:,j),vG3(:,j),v3p(:,j),vG3p(:,j)]=...
         kinem_en02(w2(1:3,j),w2p(1:3,j),v2(1:3,j),v2p(1:3,j),k3,O2O3,...
         CM3_3(1:3),delta(3),q3d(j),q3dd(j),A32);

    % body 4
    O3O4 = A34o(1:3,4);
    A43 = inv(A34o);
    A43 = A43(1:3,1:3);
    k4 = [0 0 1]';
    [w4(:,j),w4p(:,j),v4(:,j),vG4(:,j),v4p(:,j),vG4p(:,j)]=...
        kinem_en02(w3(1:3,j),w3p(1:3,j),v3(1:3,j),v3p(1:3,j),k4,O3O4,...
        CM4_4(1:3),delta(4),q4d(j),q4dd(j),A43);

    % body 5
    O4O5 = A45o(1:3,4);
    A54 = inv(A45o);
    A54 = A54(1:3,1:3);
    k5 = [0 0 1]';
    [w5(:,j),w5p(:,j),v5(:,j),vG5(:,j),v5p(:,j),vG5p(:,j)]=...
        kinem_en02(w4(1:3,j),w4p(1:3,j),v4(1:3,j),v4p(1:3,j),k5,O4O5,...
        CM5_5(1:3),delta(5),q5d(j),q5dd(j),A54);


    %Body Payload
    O5OP = A5Po(1:3,4);
    AP5 = inv(A5Po);
    AP5 = AP5(1:3,1:3);
    kP = [0 0 1]';
    [wP(:,j),wPp(:,j),vP(:,j),vGP(:,j),vPp(:,j),vGPp(:,j)]=...
        kinem_en02(w5(1:3,j),w5p(1:3,j),v5(1:3,j),v5p(1:3,j),kP,O5OP,...
```

```
        CMP_P(1:3),delta(6),qPd(j),qPdd(j),AP5);

    %velocity and acceleration of body 5 CM
    vG5_0(:,j) = A05o(1:3,1:3)*(vG5(:,j));
    vG5p_0(:,j) = A05o(1:3,1:3)*(vG5p(:,j));

end

%plot setup
vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
legend_f = 18;
plot_s = 1.5;

%figure plot
%kinematic of body 5 CM plot
vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
figure(6)
subplot(2,1,1)
plot(time,vG5_0(1,:),time,vG5_0(2,:),time,vG5_0(3,:),time,vG5_mag,LineWidth=kinCM_lw)
grid on
title("Velocity C.M. 5",FontSize=fs)
legend("$v_{G,x}$","$v_{G,y}$","$v_{G,z}$","$v_{G}$ magnitude", 'Interpreter',...
        'latex',"Location","eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Velocity [mm/s]",FontSize=fs)

subplot(2,1,2)
plot(time,vG5p_0(1,:),time,vG5p_0(2,:),time,vG5p_0(3,:),time,vG5p_mag,...
     LineWidth=kinCM_lw)
grid on
title("Acceleration C.M. 5",FontSize=fs)
legend("$ \dot{v}_{G,x} $","$ \dot{v}_{G,y} $","$ \dot{v}_{G,z} $",...
       "$ \dot{v}_{G} $ magnitude", 'Interpreter','latex',"Location",...
       "eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Acceleration [mm/s^2]",FontSize=fs)

```

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.

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

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

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

The computed inertia tensors are reported below.

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

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

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

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

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

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

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

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

```
%% Exercise 2 - Requirement 1 - INERTIA MATRICES

IG0 = (m0/12).*[200^2+160^2      0           0;
                    0     200^2+245^2      0;
                    0          0      245^2+160^2].*(10^(-6));

IG1 = (m1/12).*[230^2+246.5^2      0             0;
                    0        280^2+230^2        0;
                    0             0      246.5^2+280^2].*(10^(-6));

IG2 = (m2/12).*[   54^2+80^2        0           0;
                      0       350^2+54^2       0;
                      0            0      350^2+80^2].*(10^(-6));

IG3 = (m3/12).*[120^2+187.5^2      0           0;
                    0        120^2+390^2      0;
                    0             0      390^2+187.5^2].*(10^(-6));

IG4 = (m4/12).*[110^2+95^2      0          0;
                    0      95^2+110^2     0;
                    0          0      95^2+95^2].*(10^(-6));

IG5 = (m5).*[(60^2)/16+(20^2)/12           0               0;
                    0            (60^2)/16+(20^2)/12      0;
                    0                      0          (60^2)/8].*(10^(-6));


IGP = (mP/12).*[80^2+30^2        0            0;
                    0      230^2+30^2        0;
                    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 **??**.

$$[Fi, Mi, ti] = dynam\_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii, iAip1, iA0) \quad (4.22)$$

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 (Fip1 and Mip1) are null for the payload and the orientation matrix (${}^{i}A_{i+1}$) 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:

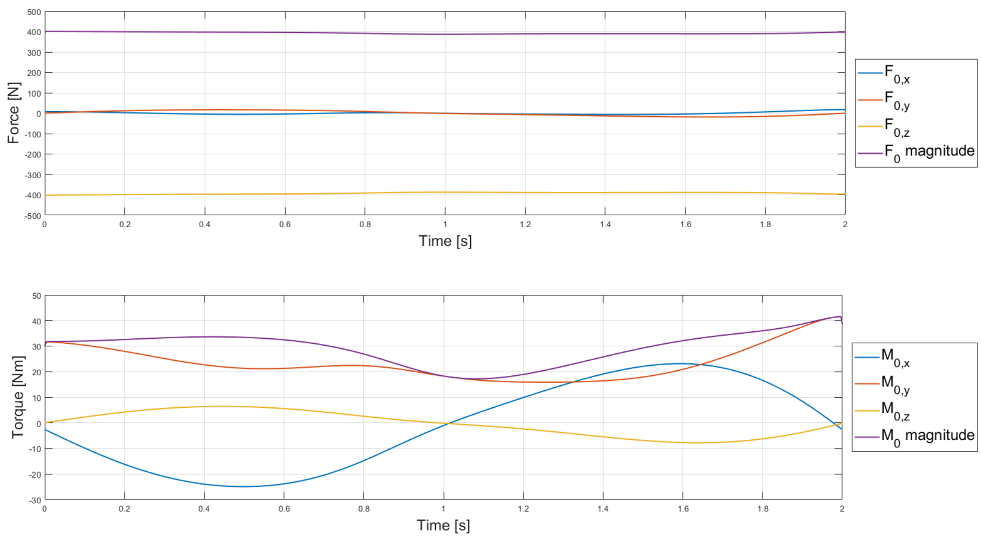


Figure 4.16: Reacting forces and torques produced by the floor

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

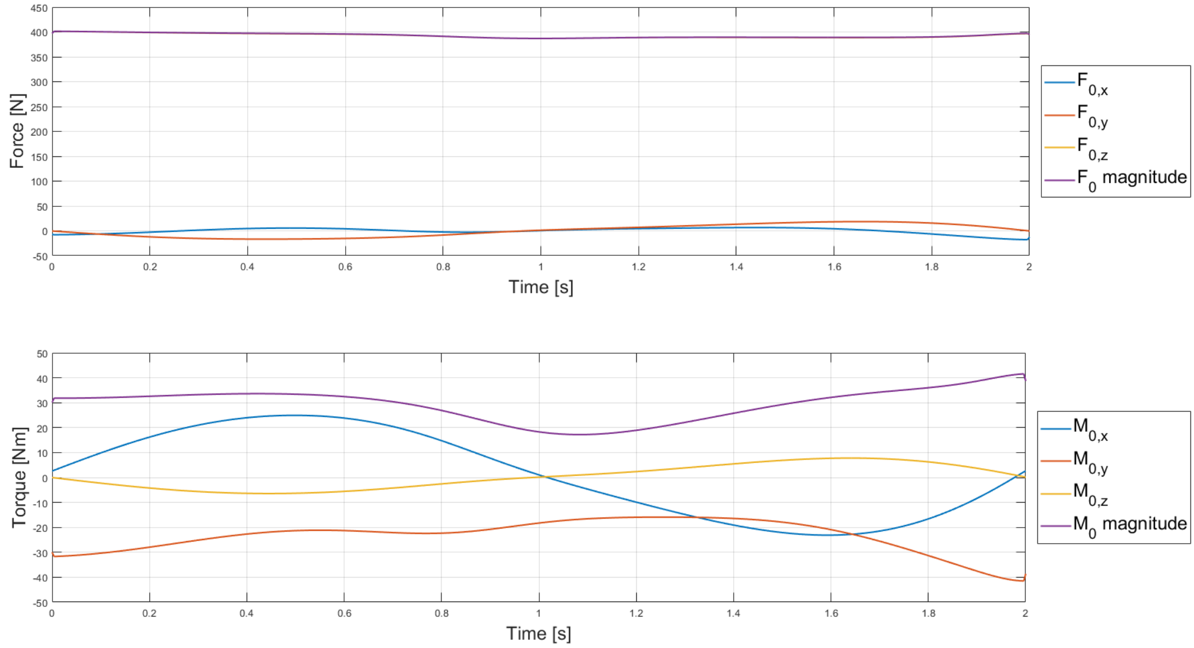


Figure 4.17: Forces and torques of by the body 0

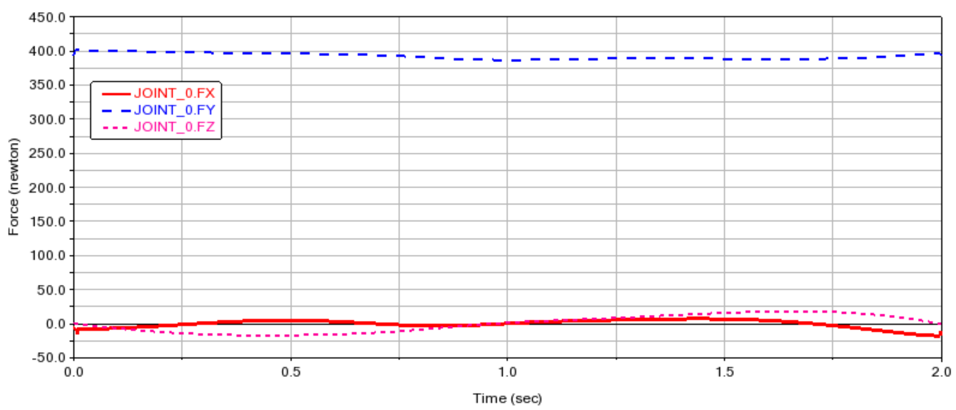


Figure 4.18: Adams simulation of body 0 forces

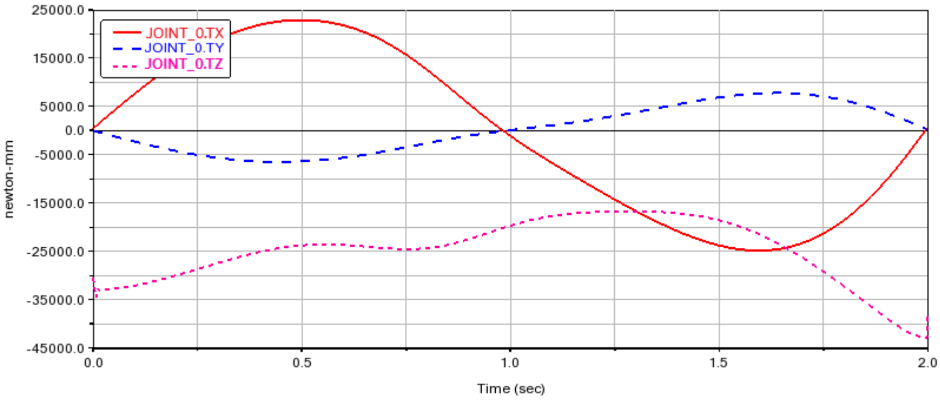


Figure 4.19: Adams simulation of body 0 torques

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 τ_i in function of time is reported and compared with the Adams simulation.

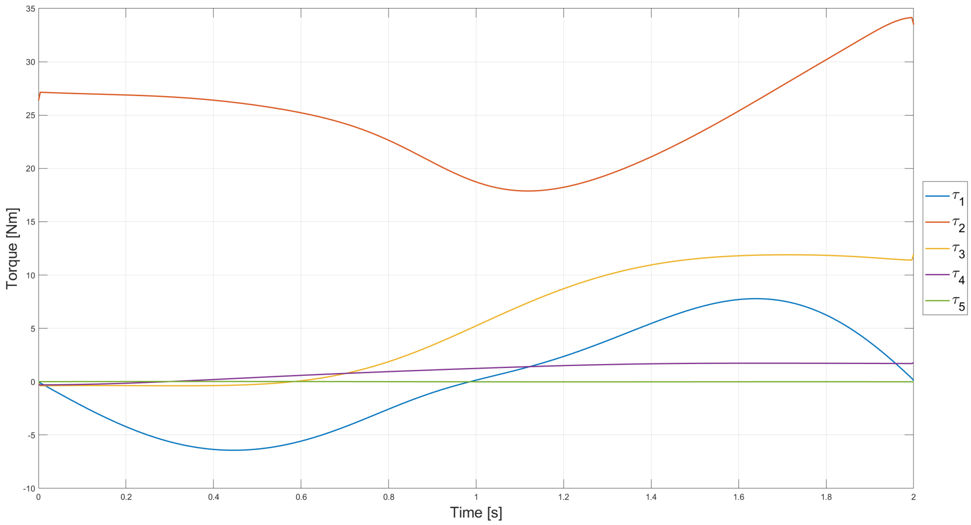


Figure 4.20: Actuator torque in function of time

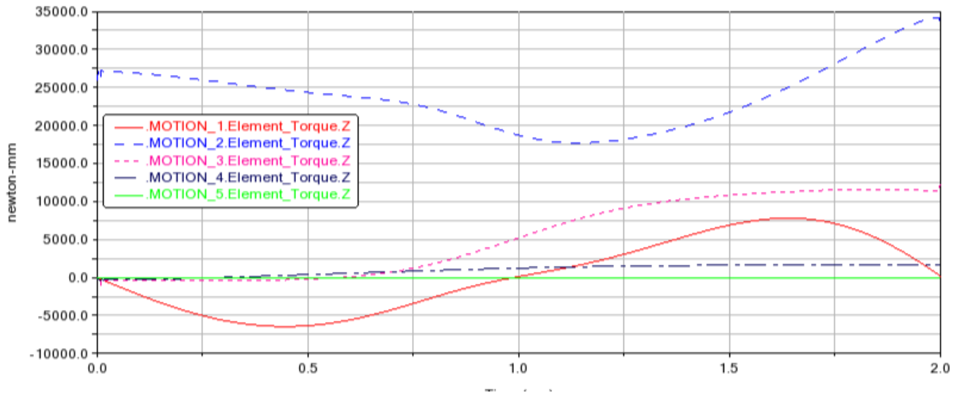


Figure 4.21: Adams simulation of actuator torque

The Matlab implementation is reported below.

```
%% Exercise 2 - Requirement 2 - INVERSE DYNAMIC PROBLEM
for j = 1:length(time)
    %Payload
    lP = [1 1 1]';
    AP0=inv(A0Po(1:3,1:3));
    [FP(:,j),MP(:,j)] = dynam_en02([0 0 0]',[0 0 0]',...
        mP,vGPp(:,j).*10^(-3),wP(:,j),wPp(:,j),lP,...
        CMP_P.*10^(-3),IGP,eye(3,3),AP0);

    %Body 5
    l5 = A5Po(1:3,4);
    A50 = inv(A05o(1:3,1:3));
    [F5(:,j),M5(:,j)] = dynam_en02(FP(:,j),MP(:,j),...
        m5,vG5p(:,j).*10^(-3),w5(:,j),w5p(:,j),l5.*10^(-3),...
        CM5_5.*10^(-3),IG5,A5Po(1:3,1:3),A50);

    %Body 4
    l4 = A45o(1:3,4);
    A40 = inv(A05o(1:3,1:3));
    [F4(:,j),M4(:,j)] = dynam_en02(F5(:,j),M5(:,j),...
        m4,vG4p(:,j).*10^(-3),w4(:,j),w4p(:,j),...
        l4.*10^(-3),CM4_4.*10^(-3),IG4,A45o(1:3,1:3),A40);

    %Body 3
    l3 = A34o(1:3,4);
    A30 = inv(A03o(1:3,1:3));
    [F3(:,j),M3(:,j)] = dynam_en02(F4(:,j),M4(:,j),...
        m3,vG3p(:,j).*10^(-3),w3(:,j),w3p(:,j),...
        l3.*10^(-3),CM3_3.*10^(-3),IG3,A34o(1:3,1:3),A30);

    %Body 2
```

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

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

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

    t1 = [0 0 1]*M1(:,j);
    t2 = [0 0 1]*M2(:,j);
    t3 = [0 0 1]*M3(:,j);
    t4 = [0 0 1]*M4(:,j);
    t5 = [0 0 1]*M5(:,j);

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

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

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

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

```
ylabel("Force [N]",fontsize=fs)
legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

subplot(2,1,2)
plot(time,-1.*M0(1,:),time,-1.*M0(2,:),time,-1.*M0(3,:),time,M0_mag,LineWidth=dyn_lw)
xlabel("Time [s]",fontsize=fs)
ylabel("Torque [Nm]",fontsize=fs)
legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

% Force on the body 0
figure(9)
subplot(2,1,1)
plot(time,F0(1,:),time,F0(2,:),time,F0(3,:),time,F0_mag,LineWidth=dyn_lw)
xlabel("Time [s]",fontsize=fs)
ylabel("Force [N]",fontsize=fs)
legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

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

Appendix A

Matlab full codes

A.1 Report 2 - Transformation Matrix

```
clear all
clc



%triangle points

P1 =  [0 1 1]';
P2 = [0 7 1]';
P3 = [0 4 7]';




%positioning matrix of triangle

T0o = [P1,P2,P3,P1;...
        1, 1, 1, 1];

%Coordinates of mobile reference system origin

P01 = [0 4 1]';

%define versors of reference systems

scale = 1;

%versor sys 0

i0 = [1 0 0]' * scale;
j0 = [0 1 0]' * scale;
```

```
k0 = [0 0 1]' * scale;

%versors sys 1

i1 = [0 1 0]' * scale;
j1 = [0 0 1]' * scale;
k1 = [1 0 0]' * scale;

%positioning matrix A01

A01 = [i1 j1 k1];




% homogeneus matrix A01o

A01o = [A01,P01;...
        0 0 0 1];

%rotation of triangle about y1 of 90

theta1 = pi/2;

rotYo90 = rotYo(theta1);

B01o=  A01o*rotYo90;

A10o = inv(A01o);

T1o = A10o * T0o;

T0_rotated1 = B01o * T1o;

%trajection of rotation 1

angles = linspace(0,theta1,1000);

for i=1:length(angles)
    trajP1_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P1;1]);
    trajP2_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P2;1]);
    trajP3_rot1(i,:) = (A01o*rotYo(angles(i))) * (A10o * [P3;1]);
end


```

```
%rotation of triangle about y0 oh theta2
theta2 = -pi/2;

A20o = A10o;

T2o = T1o;

A02o = A01o;

rotYom90 = rotYo(theta2);

C02o=  rotYom90 * A02o;

T0_rotated2 = C02o * T2o;

%trajection of rotation 2

angles_2 = linspace(0,theta2,1000);

for i=1:length(angles)
    trajP1_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P1;1]);
    trajP2_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P2;1]);
    trajP3_rot2(i,:) = (rotYo(angles_2(i))*A02o) * (A20o * [P3;1]);
end

%rotation of triangle about x2 oh theta3
theta3 = pi/2;

rotXo90 = rotXo(theta3);

D02o = C02o * rotXo90;

C20o = inv(C02o);

T2 = C20o * T0_rotated2;

T0_rotated3 = D02o * T2;

%trajection of rotation 2

angles_3 = linspace(0,theta3,1000);

for i=1:length(angles)
    trajP1_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
```

```
    T0_rotated2(:,1));
    trajP2_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
    T0_rotated2(:,2));
    trajP3_rot3(i,:) = (C02o*rotXo(angles(i))) * (C20o * ...
    T0_rotated2(:,3));
end



%% triangular plot

% plot settings
vector_lw = 3;
triang_lw = 1;
legend_fs = 18;
label_fs = 18;
text_fs = 18;

X_trang = [P1(1,1) P2(1,1) P3(1,1) P1(1,1)];
Y_trang = [P1(2,1) P2(2,1) P3(2,1) P1(2,1)];
Z_trang = [P1(3,1) P2(3,1) P3(3,1) P1(3,1)];

%initial plot

figure(1)

plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
legend("Initial position",fontsize=legend_fs)
axis equal
grid on


%plot versors

figure(2)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
```

```
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
hold on

quiver3(PO1(1),PO1(2),PO1(3),i1(1),i1(2),i1(3),Color=[1 0.5 0.5],...
LineWidth=vector_lw)
hold on
quiver3(PO1(1),PO1(2),PO1(3),j1(1),j1(2),j1(3),Color = "#77AC30",...
LineWidth=vector_lw)
hold on
quiver3(PO1(1),PO1(2),PO1(3),k1(1),k1(2),k1(3),Color = "#4DBEEE",...
LineWidth=vector_lw)
xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
text(PO1(1)-0.2,PO1(2)-0.5,PO1(3)-0.5,"P_{O_1}","FontSize",text_fs)
legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
fontsize=legend_fs)
axis equal
grid on


%plot rotation 1
figure(3)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
hold on

quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
```

```
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
Color = "#4DBEEE",LineWidth=triang_lw)
hold on

plot3(trajP1_rot1(:,1),trajP1_rot1(:,2),trajP1_rot1(:,3),"g")
hold on
plot3(trajP2_rot1(:,1),trajP2_rot1(:,2),trajP2_rot1(:,3),"g")
hold on
plot3(trajP3_rot1(:,1),trajP3_rot1(:,2),trajP3_rot1(:,3),"g")

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
text(P1(1)-0.2,P1(2)-0.2,P1(3)-0.2,"P_1","FontSize",text_fs)
text(P2(1)+0.1,P2(2)+0.1,P2(3)+0.1,"P_2","FontSize",text_fs)
text(P3(1)+0.1,P3(2)+0.1,P3(3)+0.1,"P_3","FontSize",text_fs)
text(PO1(1)-0.2,PO1(2)-0.5,PO1(3)-0.5,"P_{O_1}","FontSize",text_fs)
legend("X_0","Y_0","Z_0","Initial position","X_1","Y_1","Z_1",...
"Rot1 90° about y_1","trajectories",fontsize=legend_fs)
axis equal
grid on

%plot rotation 2
figure(4)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"--b",LineWidth=triang_lw)
hold on

quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
```

```
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
Color= "#D95319",LineWidth=triang_lw)
hold on

plot3(trajP1_rot2(:,1),trajP1_rot2(:,2),trajP1_rot2(:,3),"g")
hold on
plot3(trajP2_rot2(:,1),trajP2_rot2(:,2),trajP2_rot2(:,3),"g")
hold on
plot3(trajP3_rot2(:,1),trajP3_rot2(:,2),trajP3_rot2(:,3),"g")

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)

legend("X_0","Y_0","Z_0","Initial position","X_2","Y_2","Z_2",...
"Rot2 -90° about y_0","trajectories",fontsize=legend_fs)
axis equal
grid on

%plot rotation 2
figure(5)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b--",LineWidth=triang_lw)
hold on

plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),"--",...
LineWidth= triang_lw, Color= "#D95319")
hold on

quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,1),D02o(2,1),D02o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
```

```
hold on
quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,2),D02o(2,2),D02o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(D02o(1,4),D02o(2,4),D02o(3,4),D02o(1,3),D02o(2,3),D02o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)
hold on

plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
LineWidth= triang_lw, Color="#EDB120")
hold on

plot3(trajP1_rot3(:,1),trajP1_rot3(:,2),trajP1_rot3(:,3),"g")
hold on
plot3(trajP2_rot3(:,1),trajP2_rot3(:,2),trajP2_rot3(:,3),"g")
hold on
plot3(trajP3_rot3(:,1),trajP3_rot3(:,2),trajP3_rot3(:,3),"g")

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)

legend("X_0","Y_0","Z_0","Initial position","Rot.2 on Y_0 -90°",...
"X_2","Y_2","Z_2","Rot.3 on x_2 90°","trajectories",fontsize=legend_fs)
axis equal
grid on

%Total Plot

figure(5)

quiver3(0,0,0,i0(1),i0(2),i0(3),"r", LineWidth=vector_lw)
hold on
quiver3(0,0,0,j0(1),j0(2),j0(3),"g", LineWidth=vector_lw)
hold on
quiver3(0,0,0,k0(1),k0(2),k0(3),"b", LineWidth=vector_lw)
hold on

plot3(X_trang,Y_trang,Z_trang,"b",LineWidth=triang_lw)
hold on


plot3(T0_rotated1(1,:),T0_rotated1(2,:),T0_rotated1(3,:),...
Color = "#4DBEEE",LineWidth=triang_lw)
hold on
```

```

quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,1),B01o(2,1),B01o(3,1),...
        Color=[0.6 0 0], LineWidth=vector_lw)
hold on
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,2),B01o(2,2),B01o(3,2),...
        Color = [0.08 0.3 0], LineWidth=vector_lw)
hold on
quiver3(B01o(1,4),B01o(2,4),B01o(3,4),B01o(1,3),B01o(2,3),B01o(3,3),...
        Color = [0 0 0.6], LineWidth=vector_lw)
hold on

plot3(T0_rotated2(1,:),T0_rotated2(2,:),T0_rotated2(3,:),...
LineWidth=triang_lw,Color=        "#D95319")
hold on

plot3(T0_rotated3(1,:),T0_rotated3(2,:),T0_rotated3(3,:),...
LineWidth=triang_lw,Color="#EDB120")
hold on

quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,1),C02o(2,1),C02o(3,1),...
        Color=[1 0.5 0.5], LineWidth=vector_lw)
hold on
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,2),C02o(2,2),C02o(3,2),...
        Color = "#77AC30", LineWidth=vector_lw)
hold on
quiver3(C02o(1,4),C02o(2,4),C02o(3,4),C02o(1,3),C02o(2,3),C02o(3,3),...
        Color = "#4DBEEE", LineWidth=vector_lw)

xlabel("X",FontSize=label_fs)
ylabel("Y",FontSize=label_fs)
zlabel("Z",FontSize=label_fs)
legend("X_0","Y_0","Z_0","Initial position","Rot.1 on Y_1 90°",...
"X_1","Y_1","Z_1","Rot.2 on Y_0 -90°","Rot.3 on X_2 90°","X_2",...
"Y_2","Z_2",fontsize=legend_fs)
axis equal
grid on
```

A.2 Report 3 - SCARA robot

```
clear all
clc
close all

```

```
%% DATA

load("joints.mat")

P000 = [0 0 0]';

q1 = M(:,1);   %[rad]
q2 = M(:,2);   %[rad]
q3 = M(:,3);   %[mm]

d1 = 25;   %[mm]
d2 = 15;   %[mm]
d3 = 10;   %[mm]

a1 = 50;   %[mm]
a2 = 10;   %[mm]

%% POSITIONING PROBLEM

i=1;                  %initial position
%i=length(q1);        %final position

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
A23o = denhar_en01(0,a2,-d3+q3(i),0);

A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

%ficticius matrix for body 3 joint
A03o_joint = A02o;
A03o_joint(1:2,4) = A03o(1:2,4);

% TRAJECTORY
for j = 1:length(q1)
    A01oj = denhar_en01(0,0,d1,q1(j));
    A12oj = denhar_en01(0,a1,d2,q2(j));
```

```
    A23oj = denhar_en01(0,a2,-d3+q3(j),0);
    A03oj = A01oj * A12oj * A23oj;
    if j==1
        P303o_init = inv(A03oj)*A03oj(:,4);
    end
    P003j = A03oj * P303o_init;
    traj_EF(:,j) = P003j;
end

%Plot setup
radius = 1;
height_c = 3;
frame_lw = 3;
s = 5;
ref_lw = 3;
text_size = 18;
rf0_color = "#77AC30";
rf1_color = "#EDB120";
rf2_color = "#7E2F8E";
rf3_color = "#4DBEEE";
wf_color = "b";
%wf_color = "r";

figure(1)
%wire frame
wf = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),wf_color,...
     LineWidth=frame_lw);
hold on

%plot ref. 0
rf0_1 = quiver3(P000(1),P000(2),P000(3),1*s,0*s,0*s,...
        LineWidth=ref_lw,Color=rf0_color); %x0
hold on
rf0_2 = quiver3(P000(1),P000(2),P000(3),0*s,1*s,0*s,...
        LineWidth=ref_lw,Color=rf0_color); %y0
hold on
rf0_3 = quiver3(P000(1),P000(2),P000(3),0*s,0*s,1*s,...
        LineWidth=ref_lw,Color=rf0_color); %z0
hold on

%plot ref. 1
rf1_1 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,1)*s,A01o(2,1)*s,A01o(3,1)*s,...
        LineWidth=ref_lw,Color=rf1_color); %x1
hold on
```

```
rf1_2 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,2)*s,A01o(2,2)*s,A01o(3,2)*s,...
        LineWidth=ref_lw,Color=rf1_color); %y1
hold on
rf1_3 = quiver3(P001(1),P001(2),P001(3),...
        A01o(1,3)*s,A01o(2,3)*s,A01o(3,3)*s,...
        LineWidth=ref_lw,Color=rf1_color); %z1
hold on

%plot ref. 2
rf2_1 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,1)*s,A02o(2,1)*s,A02o(3,1)*s,...
        LineWidth=ref_lw,Color=rf2_color); %x2
hold on
rf2_2 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,2)*s,A02o(2,2)*s,A02o(3,2)*s,...
        LineWidth=ref_lw,Color=rf2_color); %y2
hold on
rf2_3 = quiver3(P002(1),P002(2),P002(3),...
        A02o(1,3)*s,A02o(2,3)*s,A02o(3,3)*s,...
        LineWidth=ref_lw,Color=rf2_color); %z2
hold on

%plot ref. 3
rf3_1 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,1)*s,A03o(2,1)*s,A03o(3,1)*s,...
        LineWidth=ref_lw,Color=rf3_color); %x3
hold on
rf3_2 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,2)*s,A03o(2,2)*s,A03o(3,2)*s,...
        LineWidth=ref_lw,Color=rf3_color); %y3
hold on
rf3_3 = quiver3(P003(1),P003(2),P003(3),...
        A03o(1,3)*s,A03o(2,3)*s,A03o(3,3)*s,...
        LineWidth=ref_lw,Color=rf3_color); %z3
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf_color);
hold on
joint_rev_01(radius,height_c,20,A02o,wf_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf_color,A03o);
```

```

%plot settings
grid on
axis equal
xlabel("X [mm]","FontSize",text_size)
ylabel("Y [mm]","FontSize",text_size)
zlabel("Z [mm]","FontSize",text_size)
legend([wf rf0_1 rf1_1 rf2_1 rf3_1],...
{"Wire Frame","rf. 0","rf. 1","rf. 2","rf. 3"},fontsize=text_size)

%%TOTAL PLOT
i=1;

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
A23o = denhar_en01(0,a2,-d3+q3(i),0);

A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

A03o_joint = A03o;

A03o_joint(3,4) = A02o(3,4);

%Plot setup
radius = 1;
height_c = 3;
frame_lw = 3;
traj_lw = 1;

ref_lw = 3;
text_size = 18;
wf1_color = "b";
wf2_color = "r";
traj_color = "g";


```

```
figure(2)
%wire frame
wf1 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
      wf1_color,LineWidth=frame_lw);
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf1_color);
hold on
joint_rev_01(radius,height_c,20,A02o,wf1_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf1_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf1_color,A03o);
hold on

i=length(q1);

A01o = denhar_en01(0,0,d1,q1(i));
A12o = denhar_en01(0,a1,d2,q2(i));
A23o = denhar_en01(0,a2,-d3+q3(i),0);

A02o = A01o * A12o;
A03o = A02o * A23o;

P001 = A01o(1:3,4);
P002 = A02o(1:3,4);
P003 = A03o(1:3,4);

wire_frame = [P000(1) P001(1) P002(1) P002(1) P003(1) P003(1);...
              P000(2) P001(2) P002(2) P002(2) P003(2) P003(2);...
              P000(3) P001(3) P001(3) P002(3) P002(3) P003(3)];

A03o_joint = A03o;

A03o_joint(3,4) = A02o(3,4);

figure(2)
%wire frame
wf2 = plot3(wire_frame(1,:),wire_frame(2,:),wire_frame(3,:),...
      wf2_color,LineWidth=frame_lw);
hold on

%joints
joint_rev_01(radius,height_c,20,A01o,wf2_color);
```

```
hold on
joint_rev_01(radius,height_c,20,A02o,wf2_color);
hold on
cube(A03o_joint(1:3,4),radius+0.5,wf2_color,A03o_joint);
hold on
cube(A03o(1:3,4),radius+0.5,wf2_color,A03o);
hold on
traj = plot3(traj_EF(1,:),traj_EF(2,:),traj_EF(3,:),...
       LineWidth=traj_lw,Color=traj_color);

%plot settings
grid on
axis equal
xlabel("X [mm]","FontSize",text_size)
ylabel("Y [mm]","FontSize",text_size)
zlabel("Z [mm]","FontSize",text_size)
legend([wf1 wf2 traj],{"Wire Frame initial pos.",...
       "Wire Frame final pos.","Trajectory"},...
       fontsize=text_size)
```

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

```
%% SETUP
clear all
clc
close all
format long e

%% DATA

data = load("trajectory.mat");

a = 200;
b = 150;
c = 150;
d = 100;
e = 250;
f = 100;
g = 220;
h = 60;
i = 20;
l = 40;

q1 = data.q1vec;
q2 = data.q2vec;
q3 = data.q3vec;
q4 = data.q4vec;
q5 = data.q5vec;

q1d = data.q1dvec.*(pi/180);
q2d = data.q2dvec.*(pi/180);
q3d = data.q3dvec.*(pi/180);
q4d = data.q4dvec.*(pi/180);
q5d = data.q5dvec.*(pi/180);

q1dd = data.q1ddvec.*(pi/180);
q2dd = data.q2ddvec.*(pi/180);
q3dd = data.q3ddvec.*(pi/180);
q4dd = data.q4ddvec.*(pi/180);
q5dd = data.q5ddvec.*(pi/180);

time = data.timevec;

qPd = zeros(length(time),1);
qPdd = zeros(length(time),1);
```

```

CM0_0 = [-140 0 100]';
CM1_1 = [80 0 35]';
CM2_2 = [125 0 -27]';
CM3_3 = [55 0 100]';
CM4_4 = [-7.5 -12.5 5]';
CM5_5 = [0 0 10]';
CMP_P = [0 0 15]';

m0 = 18;
m1 = 10.5;
m2 = 2;
m3 = 6;
m4 = 2;
m5 = 0.5;
mP = 1;

delta = zeros(6,1);

%% INERTIA MATRIX

IG0 = (m0/12).*[200^2+160^2      0           0;
                     0     200^2+245^2      0;
                     0          0      245^2+160^2].*(10^(-6));

IG1 = (m1/12).*[230^2+246.5^2      0             0;
                     0        280^2+230^2        0;
                     0             0      246.5^2+280^2].*(10^(-6));

IG2 = (m2/12).*[   54^2+80^2        0           0;
                       0       350^2+54^2       0;
                       0            0      350^2+80^2].*(10^(-6));

IG3 = (m3/12).*[120^2+187.5^2      0           0;
                     0        120^2+390^2      0;
                     0             0      390^2+187.5^2].*(10^(-6));

IG4 = (m4/12).*[110^2+95^2      0          0;
                     0      95^2+110^2     0;
                     0          0      95^2+95^2].*(10^(-6));

IG5 = (m5).*[(60^2)/16+(20^2)/12           0                0;
                     0            (60^2)/16+(20^2)/12      0;
                     0                      0           (60^2)/8].*(10^(-6));

```

```
IGP = (mP/12).*[80^2+30^2        0            0;
                     0      230^2+30^2        0;
                     0           0       230^2+80^2].*(10^(-6));


%% Solution

for j = 1:length(time)


    % POSITIONING PROBLEM

    %Denavit-Huntberg covention
    A01o = denhar_en01( 0,      0,    a+b,     deg2rad(q1(j)));
    A12o = denhar_en01(pi/2,    c,    -d,    pi/2+deg2rad(q2(j)));
    A23o = denhar_en01( 0,      e,     0,   -pi/2+deg2rad(q3(j)));
    A34o = denhar_en01( 0,      g,     f,    pi/2+deg2rad(q4(j)));
    A45o = denhar_en01(pi/2,    0,     h,      deg2rad(q5(j)));
    A5EEo = denhar_en01( 0,      0,     i,             0);
    A5Po = denhar_en01( 0,      0,     l+i,            0);

    A02o = A01o * A12o;
    A03o = A02o * A23o;
    A04o = A03o * A34o;
    A05o = A04o * A45o;
    A0EEo = A05o * A5EEo;
    A0Po = A05o * A5Po;


    %wire frame in the initial position
    if j ==1
        wf_init = [0 A01o(1,4)   A01o(1,4) A01o(1,4)+c  A02o(1,4)   A03o(1,4)...
                   A03o(1,4)   A04o(1,4) A05o(1,4) A0EEo(1,4);
                   0 A01o(2,4)   A01o(2,4) A01o(2,4)    A02o(2,4)   A03o(2,4)...
                   A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
                   0 A01o(3,4)-b A01o(3,4) A01o(3,4)    A02o(3,4)   A03o(3,4)...
                   A03o(3,4)   A04o(3,4) A05o(3,4) A0EEo(3,4)];

        A_joint1 = A01o;
        A_joint1(3,4) = A01o(3,4)-b;

        initial_cond = {wf_init,A_joint1,A02o,A03o,A04o,A05o,A0EEo}';
    end

```

```
    %wire frame in the final position
    if j == length(time)
        wf_fin = [0 A01o(1,4)   A01o(1,4) A01o(1,4)+c  A02o(1,4)   A03o(1,4)...
                  A03o(1,4)   A04o(1,4) A05o(1,4) A0EEo(1,4);
                  0 A01o(2,4)   A01o(2,4) A01o(2,4)    A02o(2,4)   A03o(2,4)...
                  A03o(2,4)-f A04o(2,4) A05o(2,4) A0EEo(2,4);
                  0 A01o(3,4)-b A01o(3,4) A01o(3,4)    A02o(3,4)   A03o(3,4)...
                  A03o(3,4)   A04o(3,4) A05o(3,4) A0EEo(3,4)];

        A_joint1 = A01o;
        A_joint1(3,4) = A01o(3,4)-b;

        final_cond = {wf_fin,A_joint1,A02o,A03o,A04o,A05o,A0EEo}';
    end

    %trajectory EE
    traj_EE(:,j) = A0EEo(1:3,4);

    %trajectory EE
    traj_CP(:,j) = A04o(1:3,4);



    % KINEMATIC PROBLEM

    % body 1
    O0O1 = A01o(1:3,4);
    A10 = inv(A01o);
    A10 = A10(1:3,1:3);
    k1 = [0 0 1]';
    [w1(:,j),w1p(:,j),v1(:,j),vG1(:,j),v1p(:,j),vG1p(:,j)]=...
            kinem_en02([0 0 0]',[0 0 0]',[0 0 0]',[0 0 0]',...
            k1,O0O1,CM1_1(1:3),delta(1),q1d(j),q1dd(j),A10);



    % body 2
    O1O2 = A12o(1:3,4);
    A21 = inv(A12o);
    A21 = A21(1:3,1:3);
    k2 = [0 0 1]';
    [w2(:,j),w2p(:,j),v2(:,j),vG2(:,j),v2p(:,j),vG2p(:,j)]=...
            kinem_en02(w1(1:3,j),w1p(1:3,j),v1(1:3,j),v1p(1:3,j),...
            k2,O1O2,CM2_2(1:3),delta(2),q2d(j),q2dd(j),A21);

    % body 3
```

```
    k3 = [0 0 1]';
    O2O3 = A23o(1:3,4);
    A32 = inv(A23o);
    A32 = A32(1:3,1:3);
    [w3(:,j),w3p(:,j),v3(:,j),vG3(:,j),v3p(:,j),vG3p(:,j)]=...
            kinem_en02(w2(1:3,j),w2p(1:3,j),v2(1:3,j),v2p(1:3,j),...
            k3,O2O3,CM3_3(1:3),delta(3),q3d(j),q3dd(j),A32);

    % body 4
    O3O4 = A34o(1:3,4);
    A43 = inv(A34o);
    A43 = A43(1:3,1:3);
    k4 = [0 0 1]';
    [w4(:,j),w4p(:,j),v4(:,j),vG4(:,j),v4p(:,j),vG4p(:,j)]=...
            kinem_en02(w3(1:3,j),w3p(1:3,j),v3(1:3,j),v3p(1:3,j),...
            k4,O3O4,CM4_4(1:3),delta(4),q4d(j),q4dd(j),A43);

    % body 5
    O4O5 = A45o(1:3,4);
    A54 = inv(A45o);
    A54 = A54(1:3,1:3);
    k5 = [0 0 1]';
    [w5(:,j),w5p(:,j),v5(:,j),vG5(:,j),v5p(:,j),vG5p(:,j)]=...
            kinem_en02(w4(1:3,j),w4p(1:3,j),v4(1:3,j),v4p(1:3,j),...
            k5,O4O5,CM5_5(1:3),delta(5),q5d(j),q5dd(j),A54);


    %Body Payload
    O5OP = A5Po(1:3,4);
    AP5 = inv(A5Po);
    AP5 = AP5(1:3,1:3);
    kP = [0 0 1]';
    [wP(:,j),wPp(:,j),vP(:,j),vGP(:,j),vPp(:,j),vGPp(:,j)]=...
            kinem_en02(w5(1:3,j),w5p(1:3,j),v5(1:3,j),v5p(1:3,j),...
            kP,O5OP,CMP_P(1:3),delta(6),qPd(j),qPdd(j),AP5);

    %velocity and acceleration of body 5 CM
    vG5_0(:,j) = A05o(1:3,1:3)*(vG5(:,j));
    vG5p_0(:,j) = A05o(1:3,1:3)*(vG5p(:,j));


    %DYNAMIC PROBLEM

    %Payload
    lP = [1 1 1]';
```

```
    AP0=inv(A0Po(1:3,1:3));
    [FP(:,j),MP(:,j)] = dynam_en02([0 0 0]',[0 0 0]',mP,vGPp(:,j).*10^(-3),...
                 wP(:,j),wPp(:,j),lP,CMP_P.*10^(-3),IGP,eye(3,3),AP0);

    %Body 5
    l5 = A5Po(1:3,4);
    A50 = inv(A05o(1:3,1:3));
    [F5(:,j),M5(:,j)] = dynam_en02(FP(:,j),MP(:,j),m5,vG5p(:,j).*10^(-3),...
                w5(:,j),w5p(:,j),l5.*10^(-3),CM5_5.*10^(-3),IG5,A5Po(1:3,1:3),A50);

    %Body 4
    l4 = A45o(1:3,4);
    A40 = inv(A04o(1:3,1:3));
    [F4(:,j),M4(:,j)] = dynam_en02(F5(:,j),M5(:,j),m4,vG4p(:,j).*10^(-3),...
                w4(:,j),w4p(:,j),l4.*10^(-3),CM4_4.*10^(-3),IG4,A45o(1:3,1:3),A40);

    %Body 3
    l3 = A34o(1:3,4);
    A30 = inv(A03o(1:3,1:3));
    [F3(:,j),M3(:,j)] = dynam_en02(F4(:,j),M4(:,j),m3,vG3p(:,j).*10^(-3),...
                w3(:,j),w3p(:,j),l3.*10^(-3),CM3_3.*10^(-3),IG3,A34o(1:3,1:3),A30);

    %Body 2
    l2 = A23o(1:3,4);
    A20 = inv(A02o(1:3,1:3));
    [F2(:,j),M2(:,j)] = dynam_en02(F3(:,j),M3(:,j),m2,vG2p(:,j).*10^(-3),...
                w2(:,j),w2p(:,j),l2.*10^(-3),CM2_2.*10^(-3),IG2,A23o(1:3,1:3),A20);

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

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

    t1 = [0 0 1]*M1(:,j);
    t2 = [0 0 1]*M2(:,j);
    t3 = [0 0 1]*M3(:,j);
    t4 = [0 0 1]*M4(:,j);
    t5 = [0 0 1]*M5(:,j);

```

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

end


%% PLOTS

%Plots setup
wf_lw =1.5;
r_j = 5;
l_j = 20;
fs = 18;
kin_lw = 1.5;
kinCM_lw = 1.5;
dyn_lw = 1.5;


%Wire frame plot
figure(1)
wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...
             Color="b",LineWidth=wf_lw); %Initial position
hold on
wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...
             Color="r",LineWidth=wf_lw); %Final position

joint_rev_01(r_j,l_j,10,initial_cond{2},"b");
joint_rev_01(r_j,l_j,10,initial_cond{3},"b");
joint_rev_01(r_j,l_j,10,initial_cond{4},"b");
joint_rev_01(r_j,l_j,10,initial_cond{5},"b");
joint_rev_01(r_j,l_j,10,initial_cond{6},"b");
joint_rev_01(r_j,r_j,4,initial_cond{7},"b");

joint_rev_01(r_j,l_j,10,final_cond{2},"r");
joint_rev_01(r_j,l_j,10,final_cond{3},"r");
joint_rev_01(r_j,l_j,10,final_cond{4},"r");
joint_rev_01(r_j,l_j,10,final_cond{5},"r");
joint_rev_01(r_j,l_j,10,final_cond{6},"r");
joint_rev_01(r_j,r_j,4,final_cond{7},"r");

legend([wf_plot(1) wf_plot(2)],{"Initial position", "Final position"},fontsize=fs)
xlabel("X [mm]","FontSize",fs)
ylabel("Y [mm]","FontSize",fs)
zlabel("Z [mm]","FontSize",fs)
axis equal
grid on
```

```
%wire frame+trajectories plot
figure(2)
wf_plot(1) = plot3(initial_cond{1}(1,:),initial_cond{1}(2,:),initial_cond{1}(3,:),...
             Color="b",LineWidth=wf_lw); %Initial position
hold on
wf_plot(2) = plot3(final_cond{1}(1,:),final_cond{1}(2,:),final_cond{1}(3,:),...
             Color="r",LineWidth=wf_lw); %Final position
hold on

t1 = plot3(traj_EE(1,:),traj_EE(2,:),traj_EE(3,:),"g"); %EE trajectory
hold on
t2 = plot3(traj_CP(1,:),traj_CP(2,:),traj_CP(3,:),"m"); %CP trajectory

joint_rev_01(r_j,l_j,10,initial_cond{2},"b");
joint_rev_01(r_j,l_j,10,initial_cond{3},"b");
joint_rev_01(r_j,l_j,10,initial_cond{4},"b");
joint_rev_01(r_j,l_j,10,initial_cond{5},"b");
joint_rev_01(r_j,l_j,10,initial_cond{6},"b");
joint_rev_01(r_j,r_j,4,initial_cond{7},"b");

joint_rev_01(r_j,l_j,10,final_cond{2},"r");
joint_rev_01(r_j,l_j,10,final_cond{3},"r");
joint_rev_01(r_j,l_j,10,final_cond{4},"r");
joint_rev_01(r_j,l_j,10,final_cond{5},"r");
joint_rev_01(r_j,l_j,10,final_cond{6},"r");
joint_rev_01(r_j,r_j,4,final_cond{7},"r");

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


%requirement 2 - ang,vel,acc plot
figure(3)
plot(time,q1,time,q2,time,q3,time,q4,time,q5,LineWidth=kin_lw)
hold on
grid on
title("Joint displacement",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...
```

```
        fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Angles [Degrees]",FontSize=fs)

figure(4)
plot(time,q1d,time,q2d,time,q3d,time,q4d,time,q5d,LineWidth=kin_lw)
grid on
title("Joint angular velocity",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...
        fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Angular velocity [rad/s]",FontSize=fs)

figure(5)
plot(time,q1dd,time,q2dd,time,q3dd,time,q4dd,time,q5dd,LineWidth=kin_lw)
grid on
title("Joint angular acceleration",FontSize=fs)
legend("Joint 1","Joint 2","Joint 3","Joint 4","Joint 5","Location","eastoutside",...
        fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Angular acceleration [rad/s^2]",FontSize=fs)


%kinematic of body 5 CM plot
vG5_mag = sqrt(vG5_0(1,:).^2+vG5_0(2,:).^2+vG5_0(3,:).^2);
vG5p_mag = sqrt(vG5p_0(1,:).^2+vG5p_0(2,:).^2+vG5p_0(3,:).^2);
figure(6)
subplot(2,1,1)
plot(time,vG5_0(1,:),time,vG5_0(2,:),time,vG5_0(3,:),time,vG5_mag,LineWidth=kinCM_lw)
grid on
title("Velocity C.M. 5",FontSize=fs)
legend("$v_{G,x}$","$v_{G,y}$","$v_{G,z}$",...
        "$v_{G}$ magnitude", 'Interpreter','latex',"Location",...
        "eastoutside",fontsize=fs)
xlabel("Time [s]",FontSize=fs)
ylabel("Velocity [mm/s]",FontSize=fs)

subplot(2,1,2)
plot(time,vG5p_0(1,:),time,vG5p_0(2,:),time,vG5p_0(3,:),time,vG5p_mag,...
     LineWidth=kinCM_lw)
grid on
title("Acceleration C.M. 5",FontSize=fs)
legend("$ \dot{v}_{G,x} $","$ \dot{v}_{G,y} $","$ \dot{v}_{G,z} $",...
       "$ \dot{v}_{G} $ magnitude", 'Interpreter',...
       'latex',"Location","eastoutside",fontsize=fs)
```

```
xlabel("Time [s]",FontSize=fs)
ylabel("Acceleration [mm/s^2]",FontSize=fs)


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

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

figure(8)
subplot(2,1,1)
plot(time,-1.*F0(1,:),time,-1.*F0(2,:),time,-1.*F0(3,:),time,F0_mag,LineWidth=dyn_lw)
xlabel("Time [s]",fontsize=fs)
ylabel("Force [N]",fontsize=fs)
legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

subplot(2,1,2)
plot(time,-1.*M0(1,:),time,-1.*M0(2,:),time,-1.*M0(3,:),time,M0_mag,LineWidth=dyn_lw)
xlabel("Time [s]",fontsize=fs)
ylabel("Torque [Nm]",fontsize=fs)
legend("M_{0,x}","M_{0,y}","M_{0,z}","M_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

% Force on the body 0
figure(9)
subplot(2,1,1)
plot(time,F0(1,:),time,F0(2,:),time,F0(3,:),time,F0_mag,LineWidth=dyn_lw)
xlabel("Time [s]",fontsize=fs)
ylabel("Force [N]",fontsize=fs)
legend("F_{0,x}","F_{0,y}","F_{0,z}","F_{0} magnitude","Location","eastoutside",...
        fontsize=fs)
grid on

subplot(2,1,2)
```

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

Appendix B

Matlab functions

B.1 Rotation about y-axis

```
function [hom_rotation_matrix] = rotYo(theta)

hom_rotation_matrix = [cos(theta)       0    sin(theta)     0;...
                            0           1         0         0;...
                       -1*sin(theta)    0    cos(theta)     0;...
                            0           0         0         1];
end
```

B.2 Rotation about x-axis

```
function [hom_rotated_matrix_X] = rotXo(theta)

hom_rotated_matrix_X = [     1          0           0           0;...
                             0      cos(theta)  -sin(theta)     0;...
                             0      sin(theta)  cos(theta)      0;...
                             0          0           0           1];
end
```

B.3 Denavit-Hartenberg convention

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

```
% alfa = twist angle [rad]
% a = offset distance [m]
% d = length [m]
% teta = angle [rad]
%
% Output variables:
% A = transformation matrix of the reference frame i with respect to the reference
%frame (i-1)

function A=denhar_en01(alfa,a,d,teta)

A=[cos(teta)             -sin(teta)            0             a
   sin(teta)*cos(alfa)   cos(teta)*cos(alfa)   -sin(alfa)    -d*sin(alfa)
   sin(teta)*sin(alfa)   cos(teta)*sin(alfa)   cos(alfa)     d*cos(alfa)
   0                     0                     0             1];
```

B.4 Revolution joint representation

```
% joint_rev_01.m
% Funtion to plot a revolute joint like a cylinder
%
% Input variables:
% R = cylinder radius [m]
% H = cylinder height [m]
% N = points numero punti sulla circonferenza
% A = 4x4 homogenious matrix of the reference frame attached to the cylinder,
%     with cylinder axis alligned with the 3rd reference axis and
%     cylinder centre located at the reference origin
% col = color ('blu','red','green',yellow',...)

function [h]=joint_rev_01(R,H,N,A,col)

[xc,yc,zc1] = cylinder(R, N);
zc=zc1*H-H/2;
base1=A*[xc(1,:); yc(1,:);zc(1,:);ones(1,N+1)];
base2=A*[xc(2,:); yc(2,:);zc(2,:);ones(1,N+1)];
xc0=[base1(1,:);base2(1,:)];
yc0=[base1(2,:);base2(2,:)];
zc0=[base1(3,:);base2(3,:)];
surface(xc0,yc0,zc0,'FaceColor', col)

end
```

B.5 Kinematic equations

```
% kinem_en02.m
% Function for recursive forward computation of kinematic variables of link i
% in terms of kinematic variables of link (i-1) and of d.o.f. in joint i.
%
% input variables:
% wim1 = angular velocity of link (i-1) expressed in frame (i-1)
% wim1p = angular acceleration of link (i-1) expressed in frame (i-1)
% vim1 = linear velocity of the origin of link (i-1) expressed in frame (i-1)
% vim1p = linear acceleration of the origin of link (i-1) expressed in frame (i-1)
% ki = unit vector k of joint i axis expressed in frame i
% lim1 = position vector of the origin i with respect to frame (i-1) expressed
%        in frame (i-1)
% bi = position vector of center of mass of link i with respect to frame i expressed
%      in frame i
% deltai = joint i parameter (1/0)
% qip = joint i d.o.f. velocity
% qipp = joint i d.o.f. acceleration
% iAim1 = rotation matrix (i)A(i-1) of frame (i-1) with respect to frame i
%
% Output variables:
% wi = angular velocity of link i expressed in frame i
% wip = angular acceleration of link i expressed in frame i
% vi = linear velocity of the origin i expressed in frame i
% vip = linear acceleration of the origin i expressed in frame i
% vGip = linear acceleration of the center of mass of link i expressed in frame i


function[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,...
                                    deltai,qip,qipp,iAim1)

%computation of angular velocity omega(i)
wi = iAim1*wim1+qip*(1-deltai)*ki;

%computation of linear velocity v(i) of origin Oi in frame i
vi=iAim1*vim1+iAim1*cross(wim1,lim1)+qip*deltai*ki;

%computation of linear velocity vG(i) of centre of mass in frame i
vGi=vi+cross(wi,bi);

%computation of angular acceleration omegadot(i)
wip = iAim1*wim1p+qipp*(1-deltai)*ki+qip*(1-deltai)*cross(iAim1*wim1,ki);

%computation of linear acceleration vdot(i)
```

```
vip = iAim1*(vim1p+cross(wim1p,lim1)+cross(wim1,cross(wim1,lim1)))+...
      qipp.*deltai*ki+2*qip*deltai*cross(iAim1*wim1,ki);

%computation of linear acceleration vGdot(i)
vGip = vip+cross(wip,bi)+cross(wi,cross(wi,bi));
```

B.6 Dynamic equations

```
% dynam_en02.m
% Function for recursive backward computation of dynamic balance of link i
% in terms of wrench in joint (i+1).
%
% Input variables:
% Fip1 = resulting force exerted in joint (i+1) by link i expressed in frame (i+1)
% Mip1 = resulting moment exerted in joint (i+1) by link i expressed in frame (i+1)
% mi = mass of link i
% vGip = linear acceleration of the center of mass of link i expressed in frame i
% wi = angular velocity of link i expressed in frame i
% wip = angular acceleration of link i expressed in frame i
% li = position vector of origin i+1 with respect to frame i expressed in frame i
% bi = position vector of center of mass of link i with respect to frame i expressed
%      in frame i
% Ii = inertia matrix of link i about its center of mass coordinate frame expressed
%      in frame i
% iAip1 = rotation matrix (i)A(i+1) of frame (i+1) with respect to frame i
%
% Output variables:
% Fi = resulting force exerted in joint i by link i-1 expressed in frame i
% Mi = resulting moment exerted in joint i by link i-1 expressed in frame i

function [Fi,Mi,ti]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii, iAip1, iA0)

%computation of force Fi
Fi = iAip1*Fip1+mi*vGip-mi*iA0*[0 0 -9.81]';

%computation of moment Mi
Mi = iAip1*Mip1-cross(Fi,bi)-cross(iAip1*Fip1,(li-bi))+Ii*wip+cross(wi,Ii*wi);
```