POLITECNICO DI TORINO

MSc Automotive Engineering

A.Y. 2023/24

Numerical Modelling and Simulation part B

Laib projects

Carlo Vittorio Colucci 329703

Pierpaolo Placida 323197

Alessio Covetti 329876

January 26, 2024

Numerical Modelling and Simulation - Part B

Exercise 1

```
clc
clear all
close all
```

The triangle adopted for this project is carachterised by the following coordinates in the global reference system:

```
Ra_0 = [0 1 1]';
Rb_0 = [0 7 1]';
Rc_0 = [0 4 7]';
Po1_0 = [0 4 1]';
Rtriangle_0 = [Ra_0 Rb_0 Rc_0];
```

The first step is the representation of the global frame of reference (FOR):

```
rFOR0_0 = eye(3);
x0Axis = quiver3(0,0,0,rFOR0_0(1,1),rFOR0_0(2,1),rFOR0_0(3,1),'linewidth',5);
hold on
y0Axis = quiver3(0,0,0,rFOR0_0(1,2),rFOR0_0(2,2),rFOR0_0(3,2),'linewidth',5);
hold on
z0Axis = quiver3(0,0,0,rFOR0_0(1,3),rFOR0_0(2,3),rFOR0_0(3,3),'linewidth',5);
hold on
axis equal
```

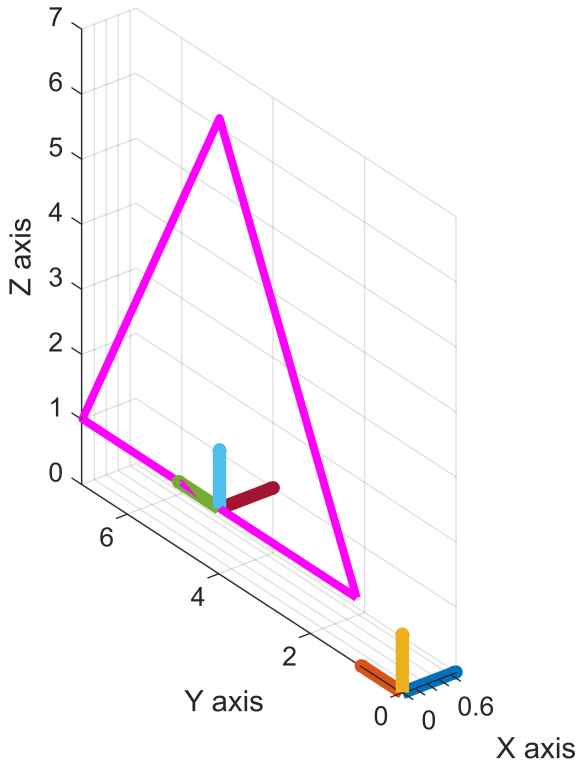
The reference triangle is represented in the space as well:

```
X = [Ra_0(1) Rb_0(1) Rc_0(1) Ra_0(1)];
Y = [Ra_0(2) Rb_0(2) Rc_0(2) Ra_0(2)];
Z = [Ra_0(3) Rb_0(3) Rc_0(3) Ra_0(3)];

plot3(X,Y,Z,'m','linewidth',3,'MarkerSize',2); hold on
```

The local FOR origin coordinates and orientation, with respect to the global FOR, are known. Thus, its axes are represented:

```
alpha = deg2rad(90);
ROTz0_90 = [cos(alpha) -sin(alpha) 0
        sin(alpha) cos(alpha) 0
        0 0 1];
ROTy0_90 = [cos(alpha) 0 sin(alpha)
        0 1 0
        -sin(alpha) 0 cos(alpha)];

rFOR1_1 = ROTy0_90 * ROTz0_90 * rFOR0_0;

x1Axis = quiver3(Po1_0(1),Po1_0(2),Po1_0(3),
rFOR1_1(1,1),rFOR1_1(2,1),rFOR1_1(3,1),'linewidth',5); hold on
```

```
y1Axis = quiver3(Po1_0(1),Po1_0(2),Po1_0(3),
rFOR1_1(1,2),rFOR1_1(2,2),rFOR1_1(3,2),'linewidth',5); hold on
z1Axis = quiver3(Po1_0(1),Po1_0(2),Po1_0(3),
rFOR1_1(1,3),rFOR1_1(2,3),rFOR1_1(3,3),'linewidth',5); hold on
axis equal; xlabel('X axis');   ylabel('Y axis');   zlabel('Z axis');   grid on;
zoom on
```



Question 1)

Starting from these data, the homogeneous matrix, decribing the orientation of the local FOR with respect to the global one, is retrieved:

```
A10 = ROTy0_90 * ROTz0_90 * eye(3)
```

```
A10 = 3×3
    0.0000   -0.0000    1.0000
    1.0000    0.0000         0
   -0.0000    1.0000    0.0000
```

```
A10o = [A10;zeros(1,3)];
translationVector = [Po1_0;1];
A10o = [A10o translationVector]
```

```
A10o = 4×4
    0.0000   -0.0000    1.0000         0
    1.0000    0.0000         0    4.0000
   -0.0000    1.0000    0.0000    1.0000
         0         0         0    1.0000
```

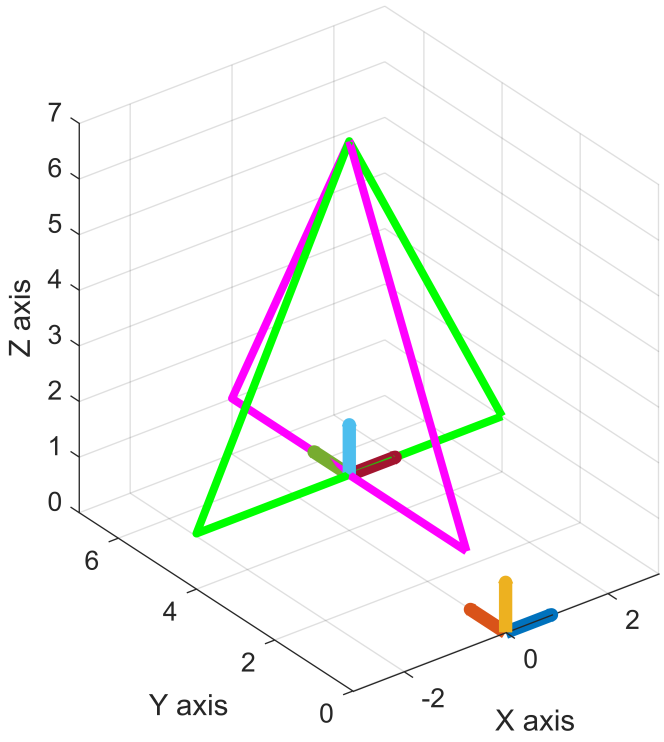
Question 3)

The first motion of the reference triangle is about the y-axis of the local FOR. The correspondent homogeneous matrix follows:

```
theta = deg2rad(90);
ROTy1_90 = [cos(theta) 0 sin(theta)
        0 1 0
        -sin(theta) 0 cos(theta)];
ROTy1_90o = [ROTy1_90; zeros(1,3)];
TranslationVectorROTy1_90o = [zeros(3,1); 1];
ROTy1_90o = [ROTy1_90o, TranslationVectorROTy1_90o];
B10o = A10o * ROTy1_90o;
```

To apply this rotation to the reference triangle, it's firstly necessary to retrieve its starting coordinate in the local FOR and then it's sufficient to multiply times the desried homogeneous matrix:

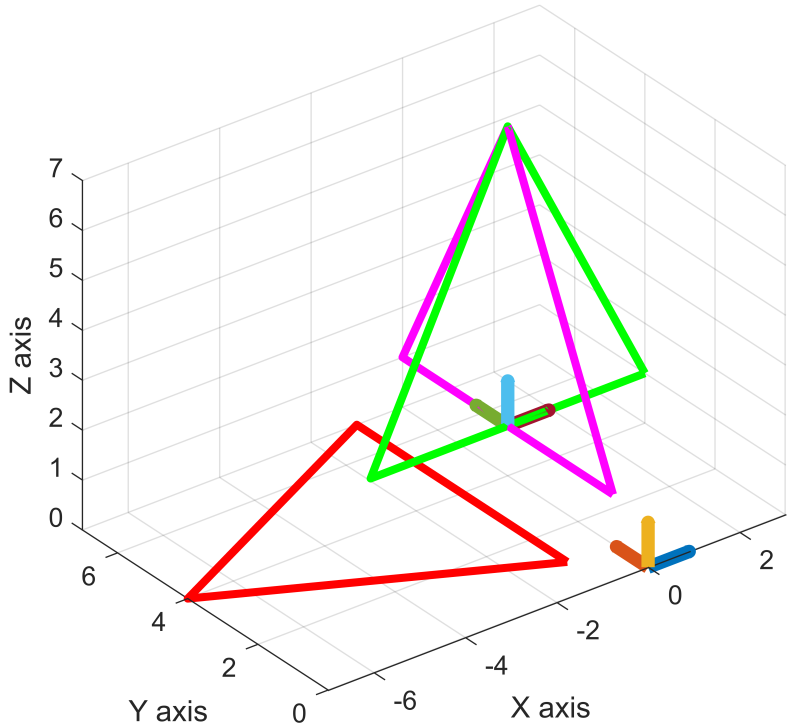
```
Qtriangle_0 = Rtriangle_0 - Po1_0;
Qtriangle_1 = inv(A10)*Qtriangle_0;
Qtriangle_1o = [Qtriangle_1; ones(1,3)];

RtrianglePrime_0o = B10o * Qtriangle_1o;
RtrianglePrime_0 = RtrianglePrime_0o(1:3,:);

Xprime = [RtrianglePrime_0(1,1) RtrianglePrime_0(1,2) RtrianglePrime_0(1,3)
RtrianglePrime_0(1,1)];
Yprime = [RtrianglePrime_0(2,1) RtrianglePrime_0(2,2) RtrianglePrime_0(2,3)
RtrianglePrime_0(2,1)];
Zprime = [RtrianglePrime_0(3,1) RtrianglePrime_0(3,2) RtrianglePrime_0(3,3)
RtrianglePrime_0(3,1)];

plot3(Xprime,Yprime,Zprime,'g','linewidth',3,'MarkerSize',2)
hold on
```



Question 4)

The second motion is about the y-axis of the global FOR. In this case, multiplying the reference triangle coordinates, in the global FOR, times the correspondent rotation matrix, was enough:

```
theta = deg2rad(-90);
ROTy0_m90 = [cos(theta) 0 sin(theta)
        0 1 0
        -sin(theta) 0 cos(theta)];

RtriangleSecond_0 = ROTy0_m90 * Rtriangle_0;

Xsecond = [RtriangleSecond_0(1,1) RtriangleSecond_0(1,2) RtriangleSecond_0(1,3)
RtriangleSecond_0(1,1)];
Ysecond = [RtriangleSecond_0(2,1) RtriangleSecond_0(2,2) RtriangleSecond_0(2,3)
RtriangleSecond_0(2,1)];
Zsecond = [RtriangleSecond_0(3,1) RtriangleSecond_0(3,2) RtriangleSecond_0(3,3)
RtriangleSecond_0(3,1)];

plot3(Xsecond,Ysecond,Zsecond,'r','linewidth',3); hold on
```



The same rotation is applied also to the local system axes. Thus the FOR2 is defined as follows:

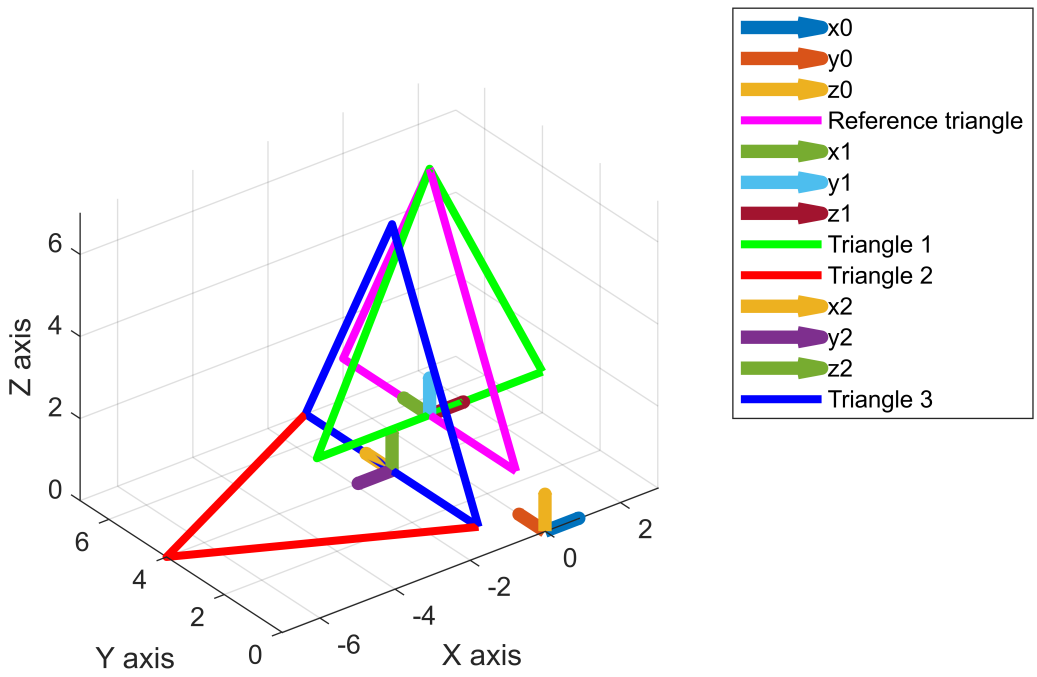
```
Po2_0 = ROTy0_m90 * Po1_0;
rFOR2_2 = ROTy0_m90 * rFOR1_1;
x2Axis = quiver3(Po2_0(1),Po2_0(2),Po2_0(3),
rFOR2_2(1,1),rFOR2_2(2,1),rFOR2_2(3,1),'linewidth',5); hold on
y2Axis = quiver3(Po2_0(1),Po2_0(2),Po2_0(3),
rFOR2_2(1,2),rFOR2_2(2,2),rFOR2_2(3,2),'linewidth',5); hold on
```

```
z2Axis = quiver3(Po2_0(1),Po2_0(2),Po2_0(3),
rFOR2_2(1,3),rFOR2_2(2,3),rFOR2_2(3,3),'linewidth',5); hold on
lgd=legend('x0','y0','z0','Reference triangle', 'x1', 'y1', 'z1', 'Triangle 1',
'Triangle 2', 'x2', 'y2', 'z2');
```

Question 5)

The final rotation to be applied is about the x-axis of the new FOR2:

```
ROTy0_m90o = [ROTy0_m90; zeros(1,3)];
TranslationVectorROTy0_m90o = [zeros(3,1); 1];
ROTy0_m90o = [ROTy0_m90o TranslationVectorROTy0_m90o];

A20o = ROTy0_m90o * A10o;
QtriangleSecond_0 = RtriangleSecond_0 - Po2_0;
QtriangleSecond_2 = inv(A20o(1:3,1:3))*QtriangleSecond_0;
QtriangleSecond_2o = [QtriangleSecond_2; ones(1,3)];

theta = deg2rad(90);
ROTx2_90o = [1 0 0 0
            0 cos(theta) -sin(theta) 0
            0 sin(theta) cos(theta) 0
            0 0 0 1];
B20o =  A20o * ROTx2_90o;
RtriangleThird_0o = B20o * QtriangleSecond_2o;
RtriangleThird_0 = RtriangleThird_0o(1:3,:);

XThird = [RtriangleThird_0(1,1) RtriangleThird_0(1,2) RtriangleThird_0(1,3)
RtriangleThird_0(1,1)];
YThird = [RtriangleThird_0(2,1) RtriangleThird_0(2,2) RtriangleThird_0(2,3)
RtriangleThird_0(2,1)];
ZThird = [RtriangleThird_0(3,1) RtriangleThird_0(3,2) RtriangleThird_0(3,3)
RtriangleThird_0(3,1)];

plot3(XThird,YThird,ZThird,'b','linewidth',3)
lgd=legend('x0','y0','z0','Reference triangle', 'x1', 'y1', 'z1', 'Triangle 1',
'Triangle 2', 'x2', 'y2', 'z2', 'Triangle 3');
```


x0
y0
z0
Reference triangle
x1
y1
z1
Triangle 1
Triangle 2
x2
y2
z2
Triangle 3
Z axis
6
4
2
0
Y axis
6
4
2
0
X axis
2
0
-2
-4
-6

Numerical Modelling and Simulation - Part B

Exercise 2

This exercise aims to represent the movements of a Scara Robot, given the trajectories of each joint. To solve the exercise the Denavit Hartenberg convention is exploited.

Input parameters for the evaluation of the homogeneous transformation matrices:

```
clear
close all
clc
load joints.mat

alfa1 = 0;  a1 = 0;     d1 = 25;
alfa2 = 0;  a2 = 50;    d2 = 15;
alfa3 = 0;  a3 = 10;    d3 = 10;

q1 = M(:, 1);   q2 = M(:, 2);   q3 = M(:, 3);
```

Here are defined the positions of each point with respect to the relative local system, used in D.H. convention. The transformation matrices are then retrieved. Moreover, the latter matrices, which will be exploited in plotting the final trajectories, are computed in the basement reference system.

```
Po0_0 = [0 0 0 1]';
Po1_1 = Po0_0;
Po2_2 = Po0_0;
Po3_3 = Po0_0;
Po0_1 = [0 0 -d1 1]';
PA_1 = [a2 0 0 1]';
PB_2 = [a3 0 0 1]';
PB_3 = [0 0 d3 1]';

mPo0_0moved = [];
mPo1_1moved = [];
mPo2_2moved = [];
mPo3_3moved = [];
mPo0_1moved = [];
mPA_1moved = [];
mPB_2moved = [];
mPB_3moved = [];

for i=1:length(M)

    capA10 = denhar_en01(0,a1,d1,q1(i));
    capA21 = denhar_en01(0,a2,d2,q2(i));
    capA32 = denhar_en01(0,a3,q3(i)-d3,0);

    capA20 = capA10*capA21;
    capA30 = capA20*capA32;

    Po0_1moved = capA10 * Po0_1;
```

```
    Po1_1moved = capA10 * Po1_1;
    PA_1moved = capA10 * PA_1;
    Po2_2moved = capA20 * Po2_2;
    PB_2moved = capA20 * PB_2;
    Po3_3moved = capA30 * Po3_3;
    PB_3moved = capA30 * PB_3;
```

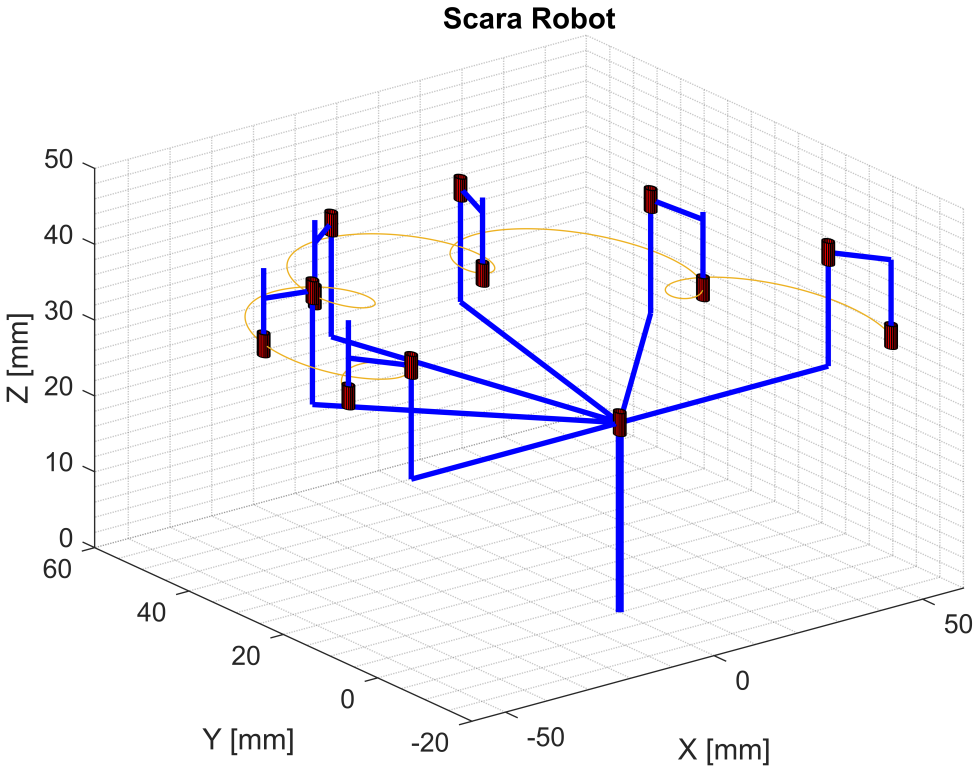
The remaining part aims to plot the trajectories of the End Effector, as well as the initial and final position of the Robot. Furthermore, also four intermediate positions of the assembly are plotted, in order to better visualize the complete movement performed by the Robot.

```
    if i==1 || i/200==round(i/200) || i==length(M)
        plot3([Po0_1moved(1) Po1_1moved(1)], [Po0_1moved(2) Po1_1moved(2)],
[Po0_1moved(3) Po1_1moved(3)],'b','linewidth',3); hold on
        plot3([Po1_1moved(1) PA_1moved(1)], [Po1_1moved(2) PA_1moved(2)],
[Po1_1moved(3) PA_1moved(3)],'b','linewidth',2)
        plot3([PA_1moved(1) Po2_2moved(1)], [PA_1moved(2) Po2_2moved(2)],
[PA_1moved(3) Po2_2moved(3)],'b','linewidth',2)
        plot3([Po2_2moved(1) PB_2moved(1)], [Po2_2moved(2) PB_2moved(2)],
[Po2_2moved(3) PB_2moved(3)],'b','linewidth',2)
        plot3([Po3_3moved(1) PB_3moved(1)], [Po3_3moved(2) PB_3moved(2)],
[Po3_3moved(3) PB_3moved(3)],'b','linewidth',2)

        joint_rev_01(1,3,20,capA10,'red')
        joint_rev_01(1,3,20,capA20,'red')
        joint_rev_01(1,3,20,capA30,'red')
    end
    mPo0_1moved = [mPo0_1moved Po0_1moved(1:3)];
    mPo1_1moved = [mPo1_1moved Po1_1moved(1:3)];
    mPA_1moved = [mPA_1moved PA_1moved(1:3)];
    mPo2_2moved = [mPo2_2moved Po2_2moved(1:3)];
    mPB_2moved = [mPB_2moved PB_2moved(1:3)];
    mPo3_3moved = [mPo3_3moved Po3_3moved(1:3)];
    mPB_3moved = [mPB_3moved PB_3moved(1:3)];

    pause(3e-3)
end
```

Despite being the final request to plot the End Effector trajectory, it is possible through these commands also to visualize the trajectories of each reference system defined in the beginning.

```
% plot3(mPo0_1moved(1,:),mPo0_1moved(2,:),mPo0_1moved(3,:)); hold on
% plot3(mPo1_1moved(1,:),mPo1_1moved(2,:),mPo1_1moved(3,:)); hold on
% plot3(mPA_1moved(1,:),mPA_1moved(2,:),mPA_1moved(3,:)); hold on
% plot3(mPo2_2moved(1,:),mPo2_2moved(2,:),mPo2_2moved(3,:)); hold on
% plot3(mPB_2moved(1,:),mPB_2moved(2,:),mPB_2moved(3,:)); hold on
% plot3(mPB_3moved(1,:),mPB_3moved(2,:),mPB_3moved(3,:)); hold on
plot3(mPo3_3moved(1,:),mPo3_3moved(2,:),mPo3_3moved(3,:)); hold on
grid minor
title("Scara Robot")
xlabel("X [mm]")
```

```
ylabel("Y [mm]")
zlabel("Z [mm]")
```



Numerical Modelling and Simulation - Part B

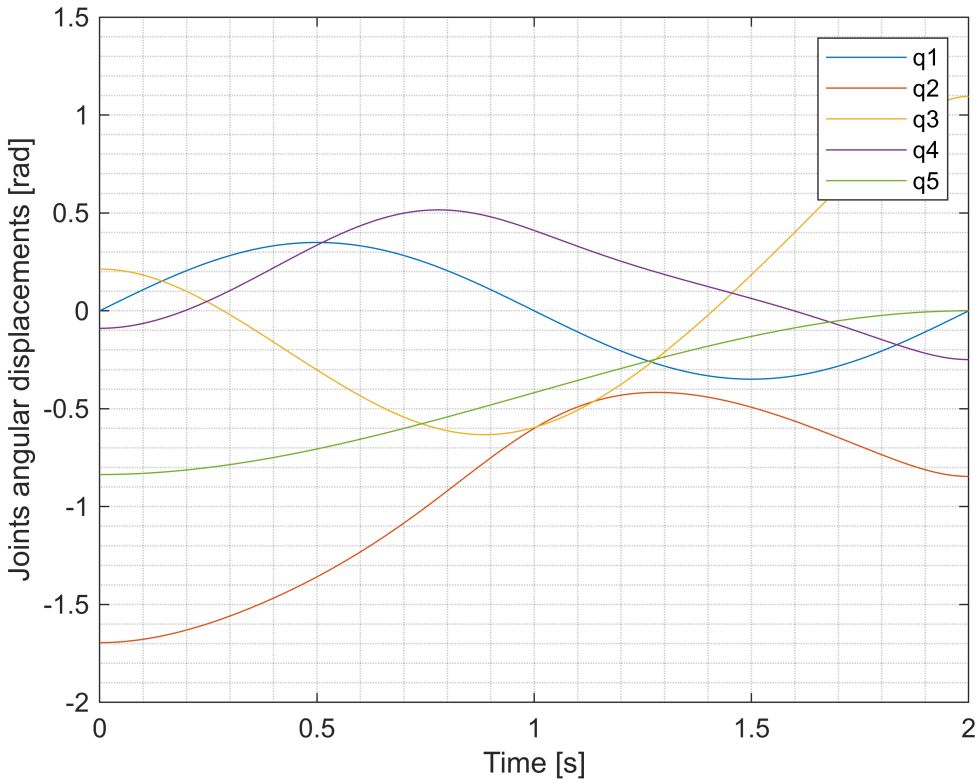
Exercise 3

```
clc
clear
close all
load("trajectory1.mat")
```

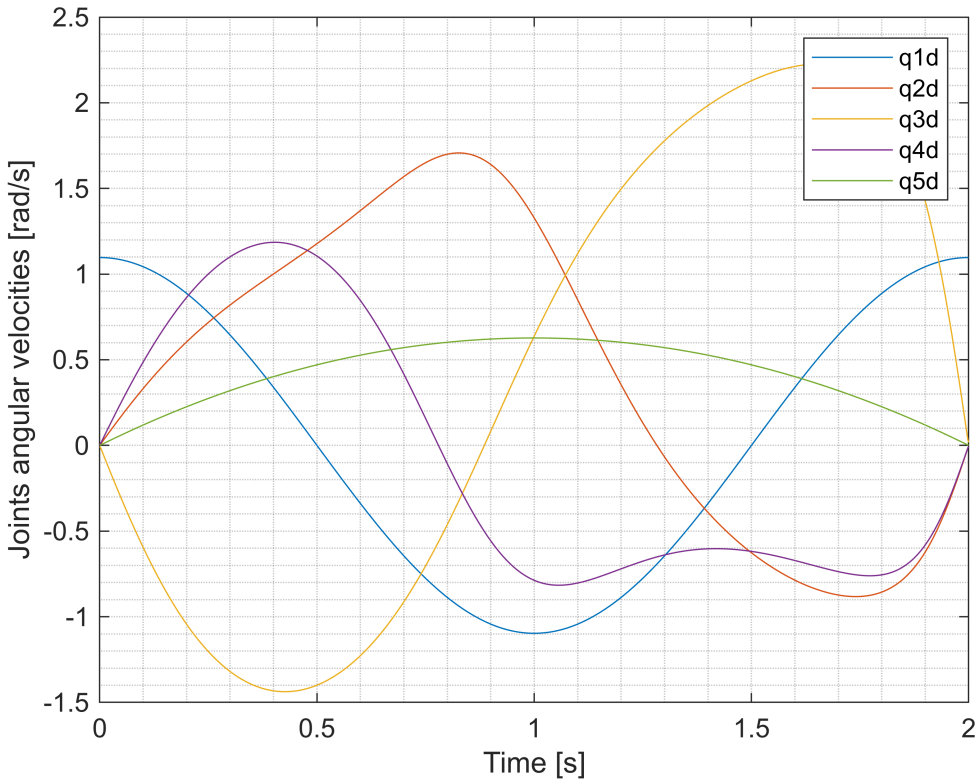
Question 2:

The joints movements are accurately described by the trajectory file, that contains the joints speed and acceleration too:

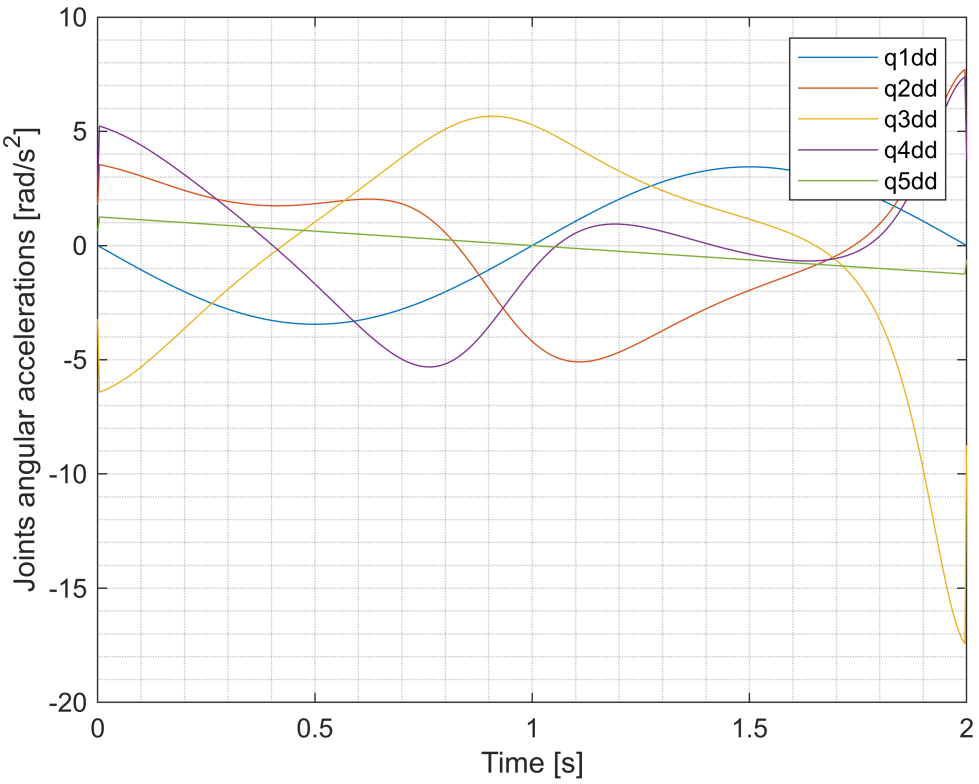
```
q1 = deg2rad(q1vec);    % [deg]
q2 = deg2rad(q2vec);
q3 = deg2rad(q3vec);
q4 = deg2rad(q4vec);
q5 = deg2rad(q5vec);
q6 = zeros(1,length(q1));

q1d = deg2rad(q1dvec);  % [deg/s]
q2d = deg2rad(q2dvec);
q3d = deg2rad(q3dvec);
q4d = deg2rad(q4dvec);
q5d = deg2rad(q5dvec);
q6d = zeros(1,length(q1));

q1dd = deg2rad(q1ddvec);    % [deg/s^2]
q2dd = deg2rad(q2ddvec);
q3dd = deg2rad(q3ddvec);
q4dd = deg2rad(q4ddvec);
q5dd = deg2rad(q5ddvec);
q6dd = zeros(1,length(q1));

figure;
plot(timevec, q1); hold on
plot(timevec, q2); hold on
plot(timevec, q3); hold on
plot(timevec, q4); hold on
plot(timevec, q5); grid minor
xlabel('Time [s]')
ylabel('Joints angular displacements [rad]')
lgd = legend('q1','q2', 'q3', 'q4','q5');
```




```
figure;
plot(timevec, q1d); hold on
plot(timevec, q2d); hold on
plot(timevec, q3d); hold on
plot(timevec, q4d); hold on
plot(timevec, q5d); grid minor
xlabel('Time [s]')
ylabel('Joints angular velocities [rad/s]')
lgd = legend('q1d','q2d', 'q3d', 'q4d','q5d');
```




```
figure;
plot(timevec, q1dd); hold on
plot(timevec, q2dd); hold on
plot(timevec, q3dd); hold on
plot(timevec, q4dd); hold on
plot(timevec, q5dd); grid minor
xlabel('Time [s]')
ylabel('Joints angular accelerations [rad/s^2]')
lgd = legend('q1dd','q2dd', 'q3dd', 'q4dd','q5dd');
```




```
o0_0 = [0 0 0]'; o1_1 = o0_0; o2_2 = o0_0; o3_3 = o0_0; o4_4 = o0_0; o5_5 = o0_0;
o6_6 = o0_0;  % [m]
A_0 = [0 0 0.200]';
B_1 = [0.150 0 0]';
C_2 = [0.250 0 0]';
EE_5 = o5_5;
PL_6 = [-0.115 0 0]';

nMovements = length(q1);    % [-]
nLinks = 6;
```

Initialization of all the matrices to collect useful data for further steps or for the visualization of the results:

```
alphaM = [];
aM = [];
dM = [];
thetaM = [];
mo4_0 = [];
mEE_0 = [];

vG11 = []; vG22 = []; vG33 = []; vG44 = []; vG55 = []; vG66 = [];

vG1p1 = []; vG2p2 = []; vG3p3 = []; vG4p4 = []; vG5p5 = []; vG6p6 = [];

vG15 = []; vG25 = []; vG35 = []; vG45 = []; vG55 = []; vG65 = [];
```

```
MAGvG15 = []; MAGvG25 = []; MAGvG35 = []; MAGvG45 = []; MAGvG55 = []; MAGvG65 = [];

vG1p5 = []; vG2p5 = []; vG3p5 = []; vG4p5 = []; vG5p5 = []; vG6p5 = [];

MAGvG1p5 = []; MAGvG2p5 = []; MAGvG3p5 = []; MAGvG4p5 = []; MAGvG5p5 = []; MAGvG6p5
= [];

MATw1 = []; MATw2 = []; MATw3 = []; MATw4 = []; MATw5 = []; MATw6 = [];

MATw1p = []; MATw2p = []; MATw3p = []; MATw4p = []; MATw5p = []; MATw6p = [];

matF0 = []; matF1 = []; matF2 = []; matF3 = []; matF4 = []; matF5 = []; matF6 = [];

matM0 = []; matM1 = []; matM2 = [];  matM3 = []; matM4 = []; matM5 = []; matM6 = [];

matMagF0 = []; matMagF1 = []; matMagF2 = []; matMagF3 = []; matMagF4 = []; matMagF5
= []; matMagF6 = [];

matMagM0 = []; matMagM1 = []; matMagM2 = [];  matMagM3 = []; matMagM4 = [];
matMagM5 = []; matMagM6 = [];

tau1 = []; tau2 = []; tau3 = []; tau4 = []; tau5 = [];
```

Question 7:

To compute the inertia tensors, the dimensions of the different bodies of the robot are collected:

```
body_0=[245,160,200];   % [mm]
body_1=[280,246.5,230];
body_2=[350,80,54];
body_3=[390,120,187.5];
body_4=[95,95,110];
body_5=[30,20,0]*0.001; % radius and length
payload=[230,80,30];
body_dimensions=[body_0;body_1;body_2;body_3;body_4;body_5;payload]*0.001;  % [m]
```

Also the center of mass positions of the different bodies are defined here, while they will be exploited later on:

```
b0 = [-0.0525 0 0.1]';  % [m]
b1=[0.080,-0.0235,-0.035]';
b2=[0.125,0,0]';
b3=[0.055,0,0]';
b4=[0,-0.0125,0]';
b5=[0,0,-0.010]';
bPL=[0,0,0]';

CM_positions = [b0 b1 b2 b3 b4 b5 bPL];

m=[18,10.5,2,6,2,1,1]; %kg
```

```
body=7;
```

Here the adopted procedure for the computation of the 6 inertia tensors follows:. The function to compute the inertia is appended in the appendix.

```
I_all=[];
I_matrix=eye(3);
for i = 1: body

    if i~=6

[I_xx,I_yy,I_zz]=inertia(m(i),body_dimensions(i,1),body_dimensions(i,2),body_dimensi
ons(i,3));
        vector=[I_xx,I_yy,I_zz];
        I_matrix=diag(vector);

    end

    if i==6
        [I_xx,I_yy,I_zz]= inertia_cylinder(m(i),body_5(1),body_5(2));
        vector=[I_xx,I_yy,I_zz];
        I_matrix=diag(vector);
    end

    I_all=[I_all, I_matrix];

end
I0=I_all(:,1:3);
I1=I_all(:,4:6);
I2=I_all(:,7:9);
I3=I_all(:,10:12);
I4=I_all(:,13:15);
I5=I_all(:,16:18);
I6=I_all(:,19:21);
```

Once all the necessary parameters have been retrieved and the utility matrices have been initialized, it's possible to compute the kinematic of the robot during its different movements:

```
figure;
axis equal
for i=1:nMovements
    alpha = [0 pi/2 0 0 pi/2 0]';
    a = [0 0.15 0.25 0.22 0 0]';
    d = [0.35 -0.127 0.127 0 0.08 0.055]';
    theta = [q1(i) pi/2+q2(i) -pi/2+q3(i) pi/2+q4(i) q5(i) 0]';
```

Question 1:

The transformation homogeneous matrices, to describe the movement from a reference frame to another one (considering the different joints), are retrieved adopting the Denavit Hartenberg convention. The function to retrieve the matrices is appended in the appendix.

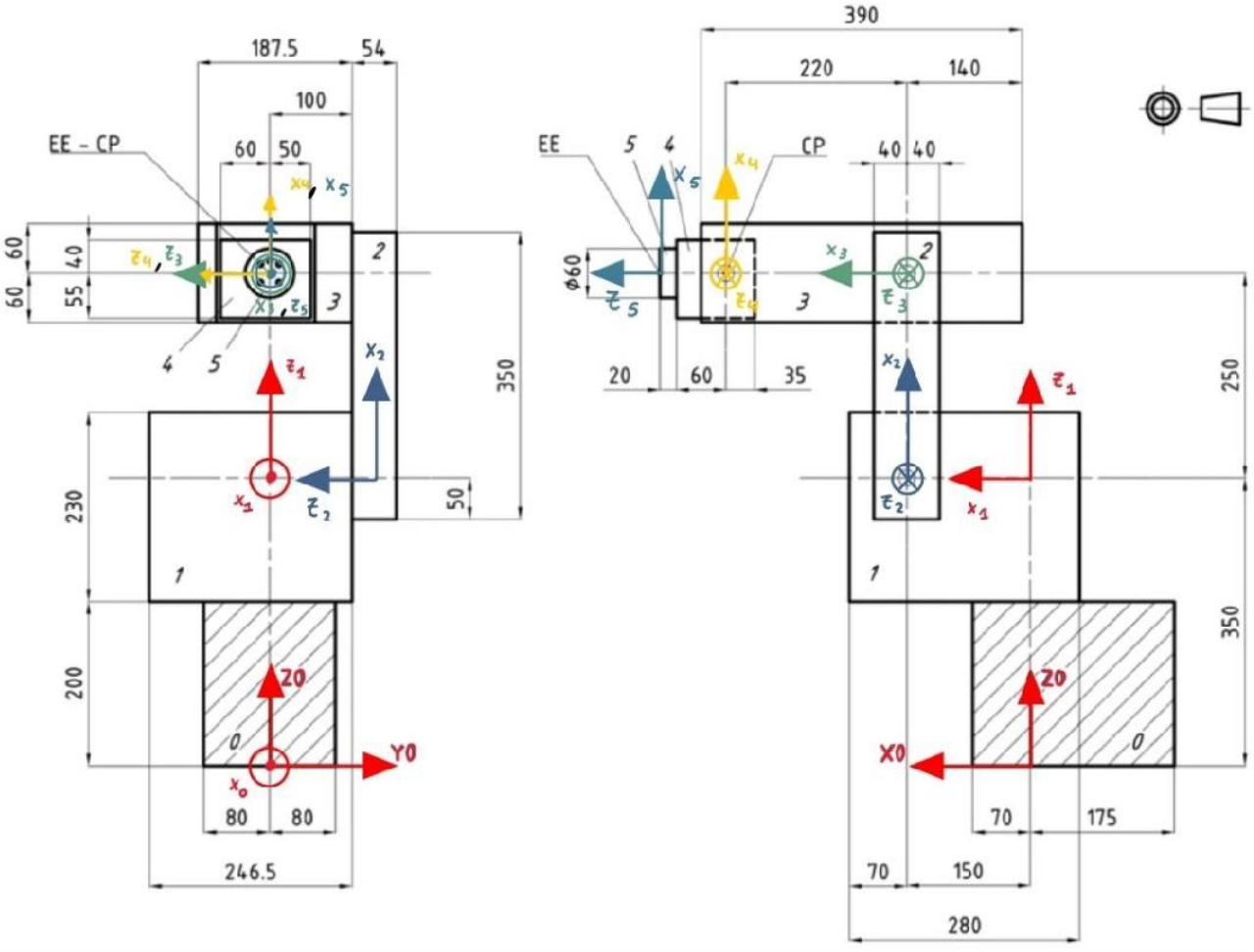


Figure 2

```
    [capA10, capA20, capA30, capA40, capA50, capA60, capA21, capA32, capA43,
capA54, capA65] = transformationMatrices(alpha, a, d, theta);
```

Question 3:

By means of these matrices it's possible to track the different robot characteristic points in the space, referred to the global reference system:

```
    o1_0h = capA10 * [o1_1; 1];
```

```
    B_0h = capA10 * [B_1; 1];
    o2_0h = capA20 * [o2_2; 1];
    C_0h = capA20 * [C_2; 1];
    o3_0h = capA30 * [o3_3; 1];
    o4_0h = capA40 * [o4_4; 1];
    o5_0h = capA50 * [o5_5; 1];
    EE_0h = capA50 * [EE_5; 1];
    o6_0h = capA60 * [o6_6; 1];
    PL_0h = capA60 * [PL_6; 1];

    o1_0 = o1_0h(1:3); B_0 = B_0h(1:3); o2_0 = o2_0h(1:3); C_0 = C_0h(1:3); o3_0
= o3_0h(1:3); o4_0 = o4_0h(1:3); o5_0 = o5_0h(1:3); o6_0 = o6_0h(1:3); EE_0 =
EE_0h(1:3); PL_0 = PL_0h(1:3);

    if i==1 | i/400==round(i/400) | i==nMovements
        plot3([o0_0(1) A_0(1)], [o0_0(2) A_0(2)], [o0_0(3)
A_0(3)],'b','linewidth',3); hold on
        plot3([A_0(1) o1_0(1)], [A_0(2) o1_0(2)], [A_0(3)
o1_0(3)],'g','linewidth',3); hold on
        plot3([o1_0(1) B_0(1)], [o1_0(2) B_0(2)], [o1_0(3)
B_0(3)],'g','linewidth',3); hold on
        plot3([B_0(1) o2_0(1)], [B_0(2) o2_0(2)], [B_0(3)
o2_0(3)],'g','linewidth',3); hold on
        plot3([o2_0(1) C_0(1)], [o2_0(2) C_0(2)], [o2_0(3)
C_0(3)],'k','linewidth',3); hold on
        plot3([C_0(1) o3_0(1)], [C_0(2) o3_0(2)], [C_0(3)
o3_0(3)],'m','linewidth',3); hold on
        plot3([o3_0(1) o4_0(1)], [o3_0(2) o4_0(2)], [o3_0(3)
o4_0(3)],'m','linewidth',3); hold on
        plot3([o4_0(1) o5_0(1)], [o4_0(2) o5_0(2)], [o4_0(3)
o5_0(3)],'b','linewidth',3); hold on
        plot3([o5_0(1) o6_0(1)], [o5_0(2) o6_0(2)], [o5_0(3)
o6_0(3)],'y','linewidth',3); hold on
        plot3([o6_0(1) PL_0(1)], [o6_0(2) PL_0(2)], [o6_0(3)
PL_0(3)],'y','linewidth',3); hold on

        joint_rev_01(0.005,0.02,20,capA10,'red')
        joint_rev_01(0.005,0.02,20,capA20,'red')
        joint_rev_01(0.005,0.02,20,capA30,'red')
        joint_rev_01(0.005,0.02,20,capA40,'red')
        joint_rev_01(0.005,0.02,20,capA50,'red')
        pause(1)    % Pause used to animate the robot movements
    end
    xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]'); grid minor
```

Question 4:

The movements of the wrist center and of the end effector are collected into matrices to be later plotted:

```
    mo4_0 = [mo4_0 o4_0(1:3)];
```

```
    mEE_0 = [mEE_0 EE_0(1:3)];
```

To express the robot characteristic points kinematic in the reference frame 5, it's necessary to compute the rotation matrices of different frames relative to it:

```
    capA53 = capA43 * capA54;
    capA52 = capA32 * capA53;
    capA51 = capA21 * capA52;
    A15 = capA51(1:3,1:3)';
    A25 = capA52(1:3,1:3)';
    A35 = capA53(1:3,1:3)';
    A45 = capA54(1:3,1:3)';
    A65 = capA65(1:3,1:3);
```

Question 6:

For each joint, the direct kinematic procedure is followed, and the data of interest are collected:

```
    for j=1:nLinks
        if j==1
            wim1 = [0 0 0]';
            wim1p = [0 0 0]';
            vim1 = [0 0 0]';
            vim1p = [0 0 0]';
            ki = [0 0 1]';
            lim1 = o1_0;
            bi = CM_positions(:,j+1);
            deltai = 0;
            qip = q1d(i);
            qipp = q1dd(i);
            iAim1 = capA10(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG11 = [vG11, vGi];
            vG1p1 = [vG1p1, vGip];

            vG15 = [vG15, A15 * vGi];
            MAGvG15 = [MAGvG15, norm(A15 * vGi)];

            vG1p5 = [vG1p5, A15 * vGip];
            MAGvG1p5 = [MAGvG1p5, norm(A15 * vGip)];

            MATw1 = [MATw1, wi];
            MATw1p = [MATw1p, wip];

        elseif j==2
            wim1 = wi;
            wim1p = wip;
            vim1 = vi;
```

```
            vim1p = vip;
            lim1h = inv(capA10) * o2_0h;
            lim1 = lim1h(1:3);
            bi = CM_positions(:,j+1);
            qip = q2d(i);
            qipp = q2dd(i);
            iAim1 = capA21(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG22 = [vG22, vGi];
            vG2p2 = [vG2p2, vGip];

            vG25 = [vG25, A25 * vGi];
            MAGvG25 = [MAGvG25, norm(A25 * vGi)];

            vG2p5 = [vG2p5, A25 * vGip];
            MAGvG2p5 = [MAGvG2p5, norm(A25 * vGip)];

            MATw2 = [MATw2, wi];
            MATw2p = [MATw2p, wip];

        elseif j==3
            wim1 = wi;
            wim1p = wip;
            vim1 = vi;
            vim1p = vip;
            lim1h = inv(capA20) * o3_0h;
            lim1 = lim1h(1:3);
            bi = CM_positions(:,j+1);
            qip = q3d(i);
            qipp = q3dd(i);
            iAim1 = capA32(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG33 = [vG33, vGi];
            vG3p3 = [vG3p3, vGip];

            vG35 = [vG35, A35 * vGi];
            MAGvG35 = [MAGvG35, norm(A35 * vGi)];

            vG3p5 = [vG3p5, A35 * vGip];
            MAGvG3p5 = [MAGvG3p5, norm(A35 * vGip)];

            MATw3 = [MATw3, wi];
            MATw3p = [MATw3p, wip];

        elseif j==4
            wim1 = wi;
```

```
            wim1p = wip;
            vim1 = vi;
            vim1p = vip;
            lim1h = inv(capA30) * o4_0h;
            lim1 = lim1h(1:3);
            bi = CM_positions(:,j+1);
            qip = q4d(i);
            qipp = q4dd(i);
            iAim1 = capA43(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG44 = [vG44, vGi];
            vG4p4 = [vG4p4, vGip];

            vG45 = [vG45, A45 * vGi];
            MAGvG45 = [MAGvG45, norm(A45 * vGi)];

            vG4p5 = [vG4p5, A45 * vGip];
            MAGvG4p5 = [MAGvG4p5, norm(A45 * vGip)];

            MATw4 = [MATw4, wi];
            MATw4p = [MATw4p, wip];

        elseif j==5
            wim1 = wi;
            wim1p = wip;
            vim1 = vi;
            vim1p = vip;
            lim1h = inv(capA40) * o5_0h;
            lim1 = lim1h(1:3);
            bi = CM_positions(:,j+1);
            qip = q5d(i);
            qipp = q5dd(i);
            iAim1 = capA54(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG55 = [vG55, vGi];
            vG5p5 = [vG5p5, vGip];

            MAGvG55 = [MAGvG55, norm(vGi)];

            MAGvG5p5 = [MAGvG5p5, norm(vGip)];

            MATw5 = [MATw5, wi];
            MATw5p = [MATw5p, wip];

        elseif j==6
            wim1 = wi;
```

```
            wim1p = wip;
            vim1 = vi;
            vim1p = vip;
            lim1h = inv(capA50) * o6_0h;
            lim1 = lim1h(1:3);
            bi = CM_positions(:,j+1);
            qip = q6d(i);
            qipp = q6dd(i);
            iAim1 = capA65(1:3,1:3)';

[wi,wip,vi,vGi,vip,vGip]=kinem_en02(wim1,wim1p,vim1,vim1p,ki,lim1,bi,deltai,qip,qipp
,iAim1);
            vG66 = [vG66, vGi];
            vG6p6 = [vG6p6, vGip];

            vG65 = [vG65, capA65(1:3,1:3) * vGi];
            MAGvG65 = [MAGvG65, norm(A65 * vGi)];

            vG6p5 = [vG65, capA65(1:3,1:3) * vGip];
            MAGvG6p5 = [MAGvG6p5, norm(A65 * vGip)];

            MATw6 = [MATw6, wi];
            MATw6p = [MATw6p, wip];
        end
    end
```

Question 8:

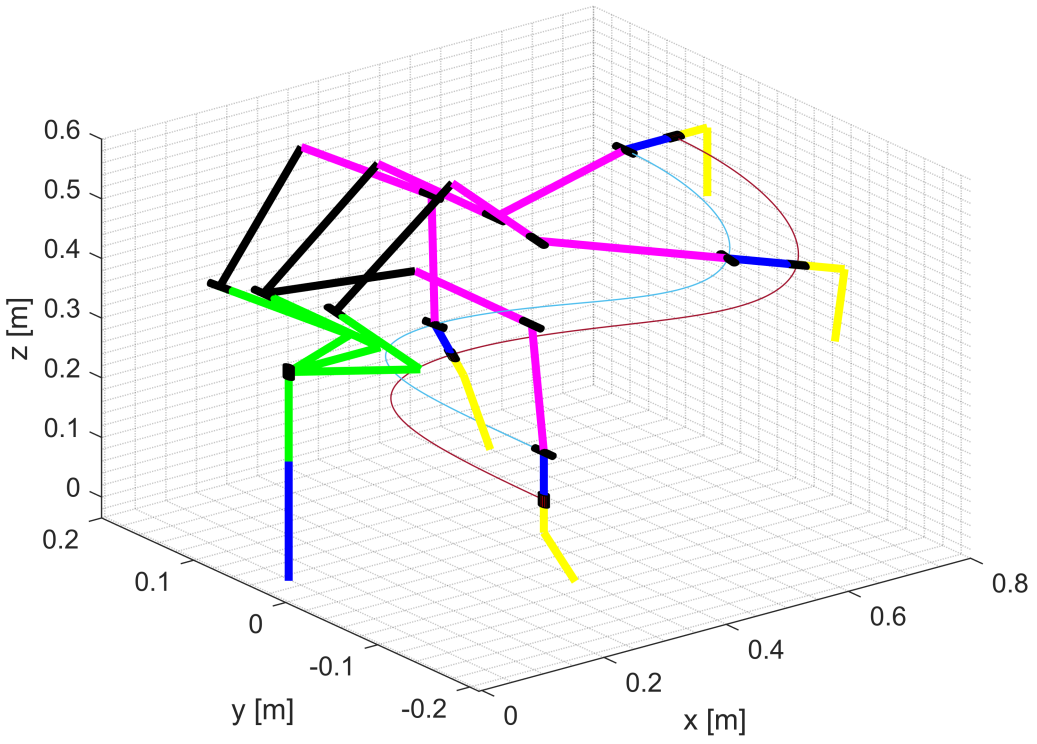
For the computation of the loads and momentums acting on each joint, the reverse procedure is followed. Also the evaluation of the torques that the joints must provide are calculated:

```
    for k=1:nLinks+1
            k = (nLinks+1)-k;
            deltai = 0;
            if k==6
                Fip1 = [0 0 0]';
                Mip1 = [0 0 0]';
                mi = 1;
                vGip = vG6p6(1:3,i);
                wi = MATw6(1:3,i);
                wip = MATw6p(1:3,i);
                li = [0 0 0]';
                bi = CM_positions(:,k+1);
                Ii = I6;
                iAip1 = eye(3);
                iA0 = capA60(1:3,1:3)';
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF6 = [matF6, Fi];
                matM6 = [matM6, Mi];
```

```
                matMagF6 = [matMagF6, norm(Fi)];
                matMagM6 = [matMagM6, norm(Mi)];

            elseif k==5
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 0.5;
                vGip = vG5p5(1:3,i);
                wi = MATw5(1:3,i);
                wip = MATw5p(1:3,i);
                lih = inv(capA50) * o6_0h;
                li = lih(1:3);
                bi = CM_positions(:,k+1);
                Ii = I5;
                iAip1 = A65;
                iA0 = capA60(1:3,1:3)';
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF5 = [matF5, Fi];
                matM5 = [matM5, Mi];
                matMagF5 = [matMagF5, norm(Fi)];
                matMagM5 = [matMagM5, norm(Mi)];
                tau5 = [tau5, ki' * Fi *deltai + ki' * Mi * (1-deltai)];

            elseif k==4
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 2;
                vGip = vG4p4(1:3,i);
                wi = MATw4(1:3,i);
                wip = MATw4p(1:3,i);
                lih = inv(capA40) * o5_0h;
                li = lih(1:3);
                bi = CM_positions(:,k+1);
                Ii = I4;
                iAip1 = capA54(1:3,1:3);
                iA0 = capA40(1:3,1:3)';
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF4 = [matF4, Fi];
                matM4 = [matM4, Mi];
                matMagF4 = [matMagF4, norm(Fi)];
                matMagM4 = [matMagM4, norm(Mi)];
                tau4 = [tau4, ki' * Fi *deltai + ki' * Mi * (1-deltai)];

            elseif k==3
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 6;
                vGip = vG3p3(1:3,i);
```

```
                wi = MATw3(1:3,i);
                wip = MATw3p(1:3,i);
                lih = inv(capA30) * o4_0h;
                li = lih(1:3);
                bi = CM_positions(:,k+1);
                Ii = I3;
                iAip1 = capA43(1:3,1:3);
                iA0 = capA30(1:3,1:3)';
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF3 = [matF3, Fi];
                matM3 = [matM3, Mi];
                matMagF3 = [matMagF3, norm(Fi)];
                matMagM3 = [matMagM3, norm(Mi)];
                tau3 = [tau3, ki' * Fi *deltai + ki' * Mi * (1-deltai)];

            elseif k==2
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 2;
                vGip = vG2p2(1:3,i);
                wi = MATw2(1:3,i);
                wip = MATw2p(1:3,i);
                lih = inv(capA20) * o3_0h;
                li = lih(1:3);
                bi = CM_positions(:,k+1);
                Ii = I2;
                iAip1 = capA32(1:3,1:3);
                iA0 = capA20(1:3,1:3)';
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF2 = [matF2, Fi];
                matM2 = [matM2, Mi];
                matMagF2 = [matMagF2, norm(Fi)];
                matMagM2 = [matMagM2, norm(Mi)];
                tau2 = [tau2, ki' * Fi *deltai + ki' * Mi * (1-deltai)];

            elseif k==1
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 10.5;
                vGip = vG1p1(1:3,i);
                wi = MATw1(1:3,i);
                wip = MATw1p(1:3,i);
                lih = inv(capA10) * o2_0h;
                li = lih(1:3);
                bi = CM_positions(:,k+1);
                Ii = I1;
                iAip1 = capA21(1:3,1:3);
                iA0 = capA10(1:3,1:3)';
```

```
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF1 = [matF1, Fi];
                matM1 = [matM1, Mi];
                matMagF1 = [matMagF1, norm(Fi)];
                matMagM1 = [matMagM1, norm(Mi)];
                tau1 = [tau1, ki' * Fi *deltai + ki' * Mi * (1-deltai)];

            elseif k==0
                Fip1 = Fi;
                Mip1 = Mi;
                mi = 18;
                vGip = [0 0 0]';
                wi = [0 0 0]';
                wip = [0 0 0]';
                li = o1_0;
                bi = CM_positions(:,k+1);
                Ii = I0;
                iAip1 = capA10(1:3,1:3);
                iA0 = eye(3);
                [Fi,Mi]=dynam_en02(Fip1, Mip1, mi, vGip, wi, wip, li, bi, Ii,
iAip1, iA0);
                matF0 = [matF0, Fi];
                matM0 = [matM0, Mi];
                matMagF0 = [matMagF0, norm(Fi)];
                matMagM0 = [matMagM0, norm(Mi)];
            end
    end
end

plot3(mo4_0(1,:),mo4_0(2,:),mo4_0(3,:)); hold on
plot3(mEE_0(1,:),mEE_0(2,:),mEE_0(3,:)); hold on
```

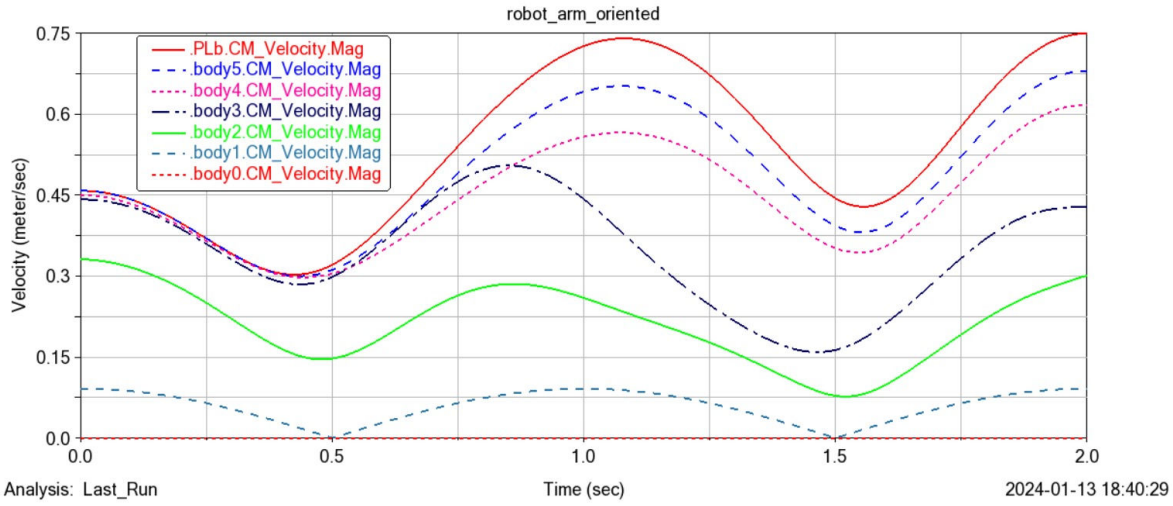


The wire frame represented above depicts the robot movements in the space for different time intervals, in particular, its initial and final positions are represented. Also the trajectories of the wrist center and of the end effector are traced as well.

```
figure;
plot(timevec, MAGvG15);hold on
plot(timevec, MAGvG25);hold on
plot(timevec, MAGvG35);hold on
plot(timevec, MAGvG45);hold on
plot(timevec, MAGvG55);hold on
plot(timevec, MAGvG65);grid minor
xlabel('Time [s]'); ylabel('Speed [m/s]');
lgd = legend('MAGvG15','MAGvG25', 'MAGvG35', 'MAGvG45','MAGvG55','MAGvG65');
```

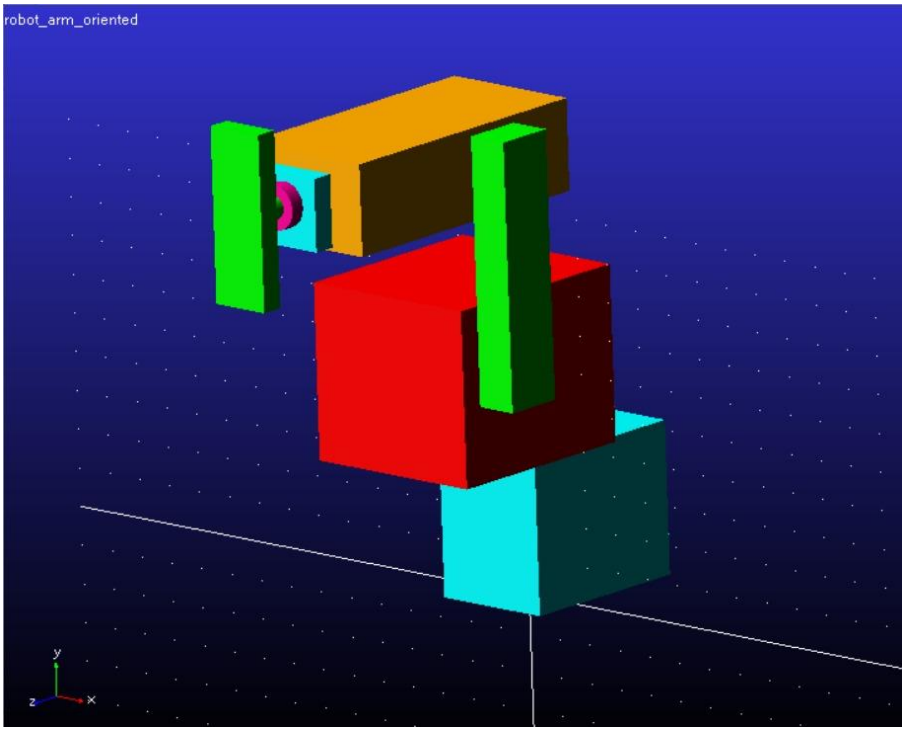
The plot above shows the center of mass movements of each body belonging to the robot during itìs considered working operation.

The same results have been obtained also by means of a simulation of the same robot model by the Adams Car software:

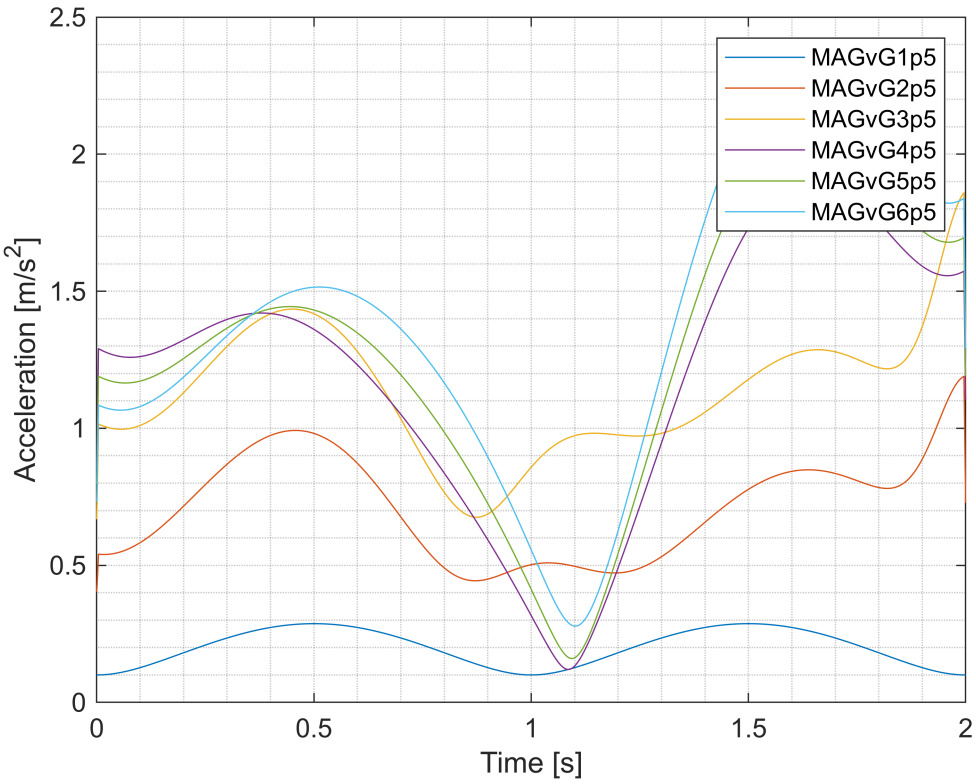


The results obtained are exactely the same, highlighting the effectiveness of the studied model.

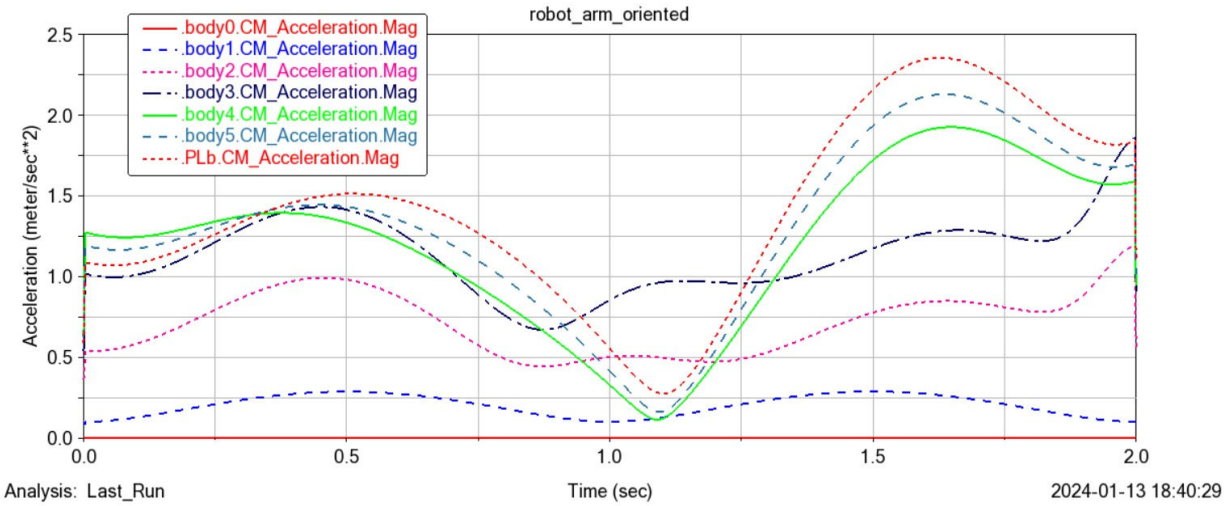
Here is provided a figure representing the equivalent model designed in Adams View.




```
figure;
plot(timevec, MAGvG1p5);hold on
plot(timevec, MAGvG2p5);hold on
plot(timevec, MAGvG3p5);hold on
plot(timevec, MAGvG4p5);hold on
plot(timevec, MAGvG5p5);hold on
plot(timevec, MAGvG6p5);grid minor
xlabel('Time [s]'); ylabel('Acceleration [m/s^2]');
lgd = legend('MAGvG1p5','MAGvG2p5', 'MAGvG3p5', 'MAGvG4p5','MAGvG5p5','MAGvG6p5');
```

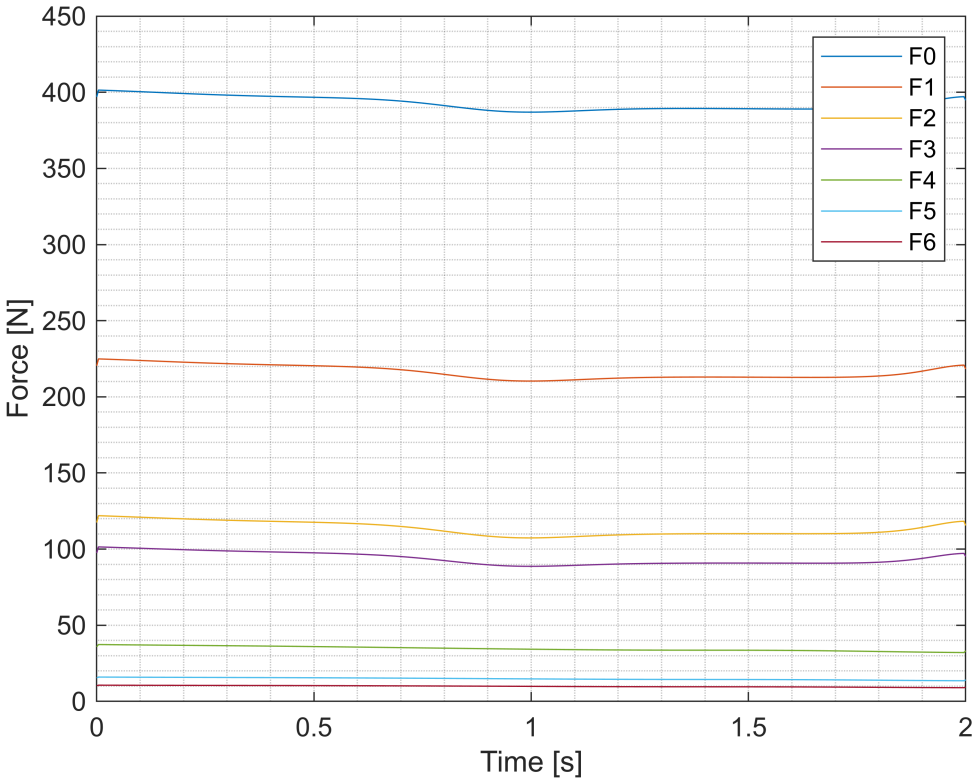


As well as for the centres of mass velocities, the same results for the relative linear accelerations are retrieved (both by Matlab and by Adams Car):



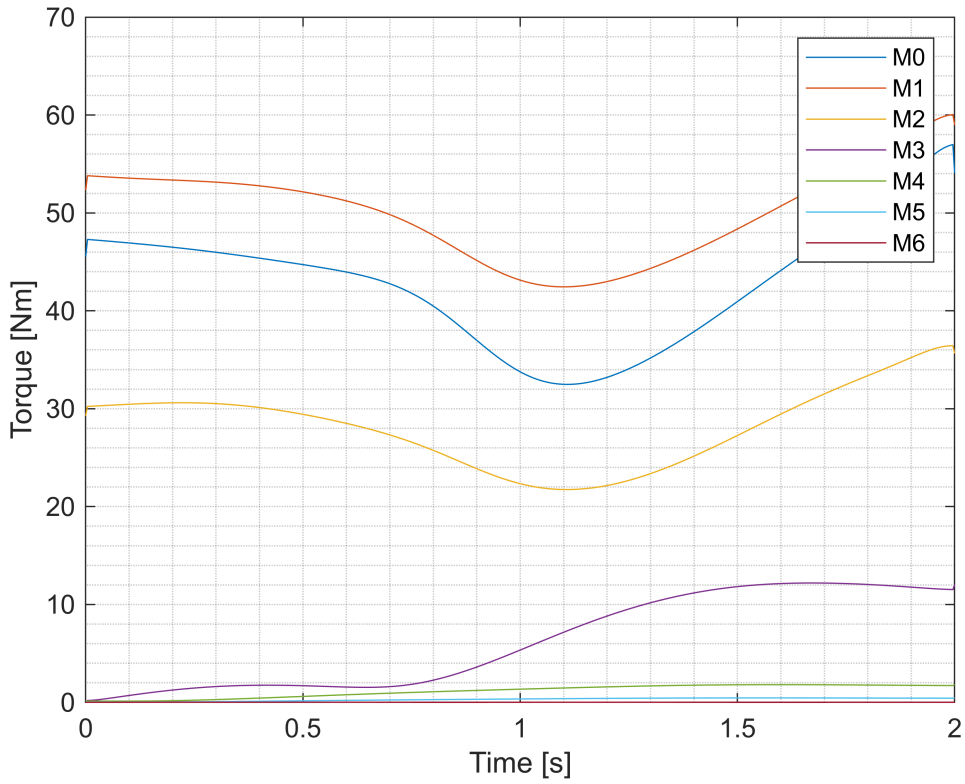

```
figure;
plot(timevec, matMagF0);hold on
plot(timevec, matMagF1);hold on
plot(timevec, matMagF2);hold on
plot(timevec, matMagF3);hold on
plot(timevec, matMagF4);hold on
plot(timevec, matMagF5);hold on
plot(timevec, matMagF6);grid minor
```

```
xlabel('Time [s]'); ylabel('Force [N]');
lgd = legend('F0','F1','F2', 'F3', 'F4','F5','F6');
```



The forces acting on each joint follow an interesting trend: moving closer to the basement, the loads that the joins withstand increase as expected for structural reasons.

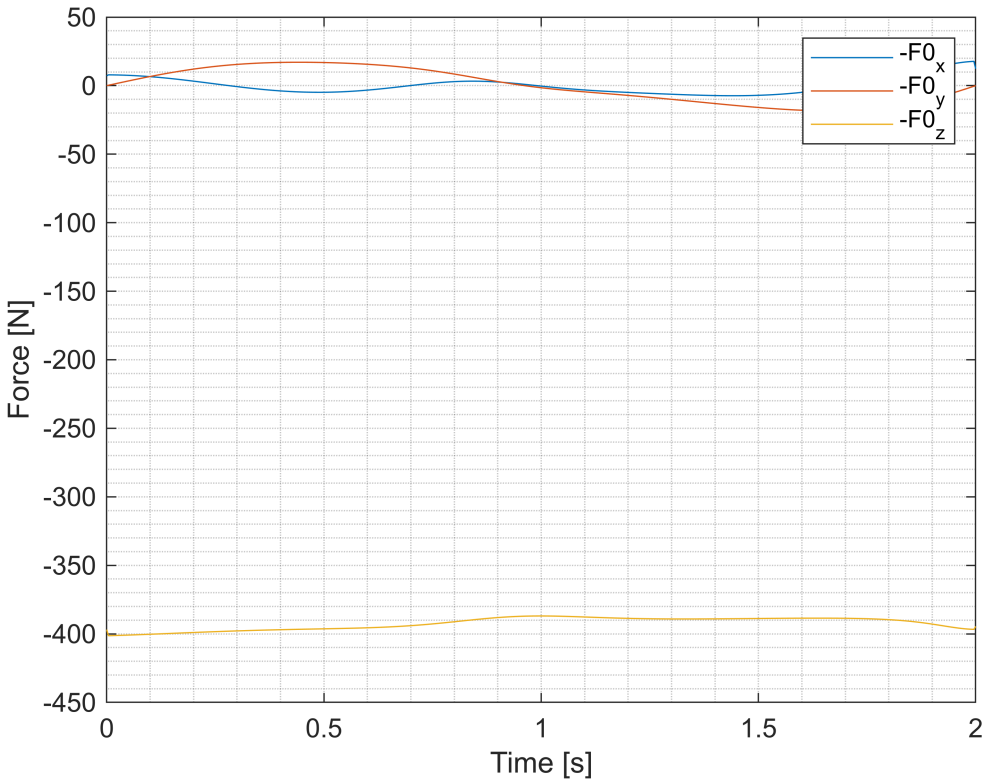
```
figure;
plot(timevec, matMagM0);hold on
plot(timevec, matMagM1);hold on
plot(timevec, matMagM2);hold on
plot(timevec, matMagM3);hold on
plot(timevec, matMagM4);hold on
plot(timevec, matMagM5);hold on
plot(timevec, matMagM6);grid minor
xlabel('Time [s]'); ylabel('Torque [Nm]');
lgd = legend('M0','M1','M2', 'M3', 'M4','M5','M6');
```



The same reasonement holds for the momentums as well.

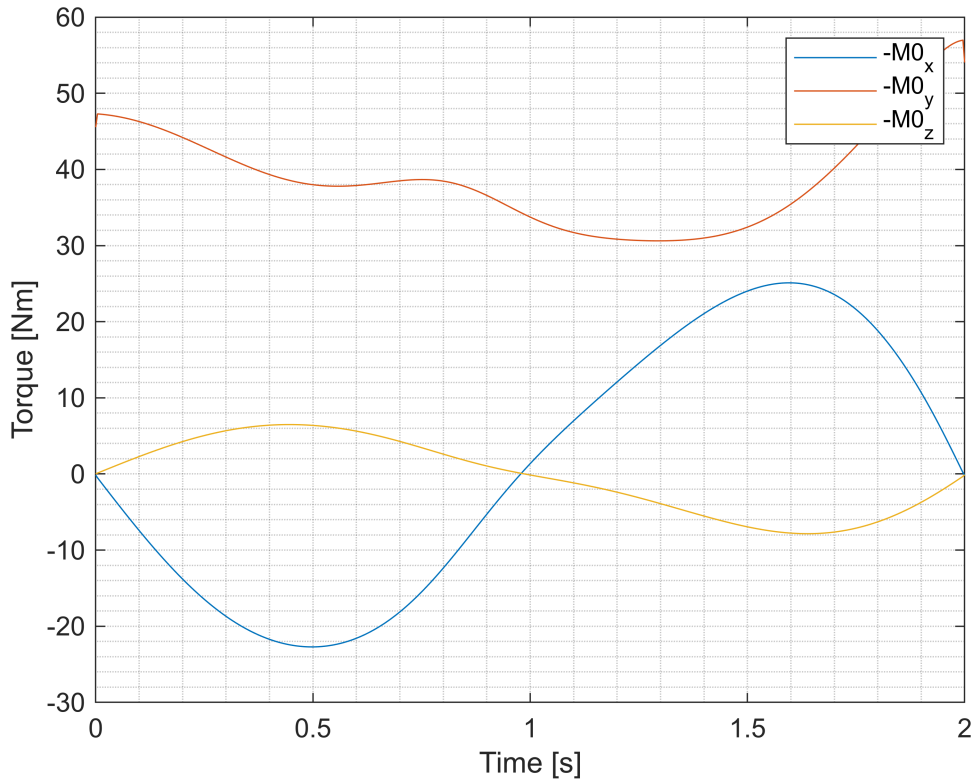
Question 9:

```
figure;
plot(timevec, -matF0(1,:)); hold on
plot(timevec, -matF0(2,:)); hold on
plot(timevec, -matF0(3,:)); grid minor
xlabel('Time [s]'); ylabel('Force [N]');
lgd = legend('-F0_x','-F0_y','-F0_z');
```



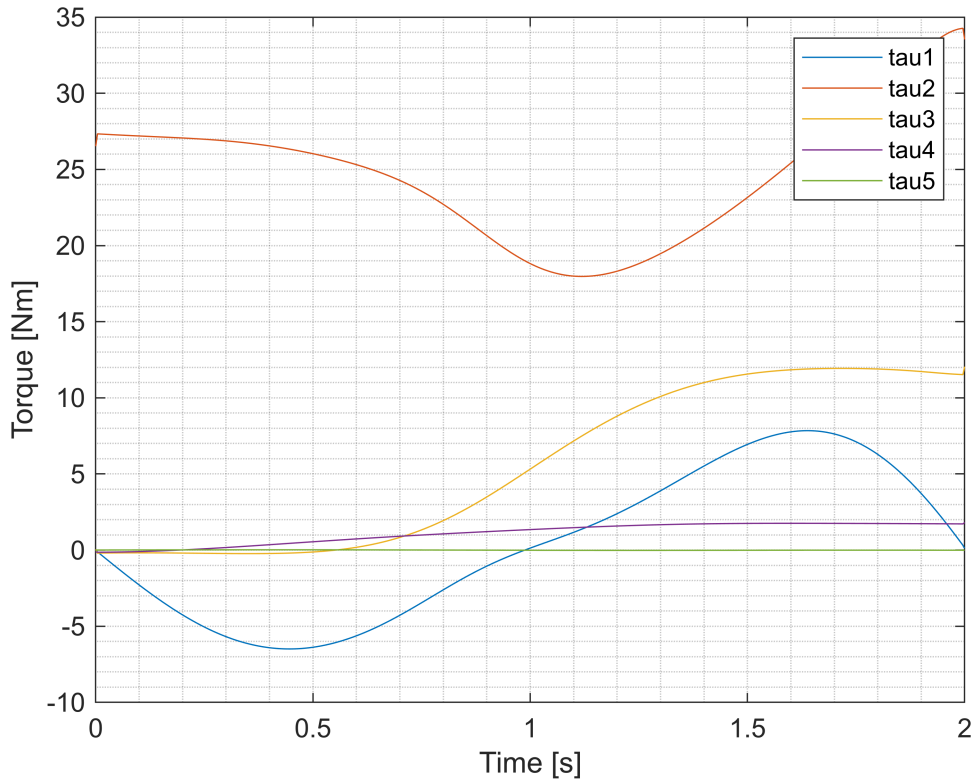
The most relevant load case to be analysed, as expected from the previous represnetation, is the one acting on the basement. The highest force contribution is related the one directed along the z-axis. This result is quite expected, considering the overall static load that the basement must withstand.

```
figure;
plot(timevec, -matM0(1,:)); hold on
plot(timevec, -matM0(2,:)); hold on
plot(timevec, -matM0(3,:)); grid minor
xlabel('Time [s]'); ylabel('Torque [Nm]');
lgd = legend('-M0_x','-M0_y','-M0_z');
```



Something of different happens for the momentum, where all the components attend a considerable contribution. The highest component is directed along the y-axis.

```
figure;
plot(timevec, tau1);hold on
plot(timevec, tau2);hold on
plot(timevec, tau3);hold on
plot(timevec, tau4);hold on
plot(timevec, tau5);grid minor
xlabel('Time [s]'); ylabel('Torque [Nm]');
lgd = legend('tau1','tau2','tau3', 'tau4', 'tau5');
```



Finally the trend of the torques that the joints must provide are depicted above. The electric motors embedded in the robot will enact to produce such these torque profiles for the correct functioning of the system.

Appendix

```
function [Ixx,Iyy,Izz] =  inertia(mass,x_local,y_local,z_local)

    Ixx=(1/12)*mass*(y_local.^2+z_local.^2);
    Iyy=(1/12)*mass*(x_local.^2+z_local.^2);
    Izz=(1/12)*mass*(y_local.^2+x_local.^2);

end

function [Ixx,Iyy,Izz] =  inertia_cylinder(mass,radius,length)

    Izz=(1/2)*mass*(radius^2);
    Iyy=(1/12)*mass*(3*radius^2+length^2);
    Ixx=(1/12)*mass*(3*radius^2+length^2);

end

function [capA10, capA20, capA30, capA40, capA50, capA60, capA21, capA32, capA43,
capA54, capA65]=transformationMatrices(alpha, a, d, theta)
    capA10 = denhar_en01(alpha(1), a(1), d(1), theta(1));
    capA21 = denhar_en01(alpha(2), a(2), d(2), theta(2));
```

```
    capA32 = denhar_en01(alpha(3), a(3), d(3), theta(3));
    capA43 = denhar_en01(alpha(4), a(4), d(4), theta(4));
    capA54 = denhar_en01(alpha(5), a(5), d(5), theta(5));
    capA65 = denhar_en01(alpha(6), a(6), d(6), theta(6));

    capA20 = capA10*capA21;
    capA30 = capA20*capA32;
    capA40 = capA30*capA43;
    capA50 = capA40*capA54;
    capA60 = capA50*capA65;
end
```