Imperial College London

DE3-ROB1 ROBOTICS 1

Tutorial 2: Dynamics and Control of a prallel robot

Marcin Laskowski

supervised by
Dr Petar Kormushev

November 12, 2017

1 Abstract

During the second Tutorial classes it was given a planar four-link parallel robot. Below, in Fig.1 is illustrated the diagram of thus robot.

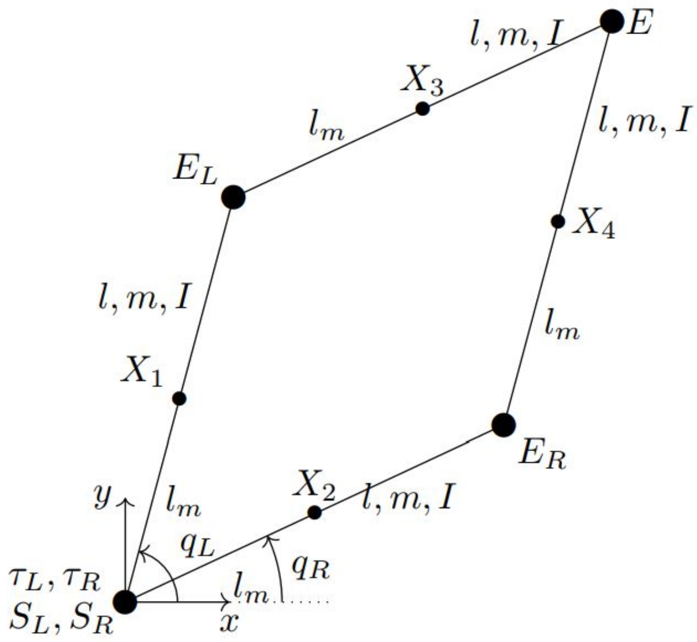

Figure 1: Parallel robot diagram

The goal of the tutorial was to calculate the Dynamic equation on the basis of Lagrange formulation and familiarise with two types of controllers: feedback and feedforward.

2 Dynamics

In the first task of the tutorial classes was determined the dynamics of such a system using the Lagrange formulation.

$$H = \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \tag{1}$$

where $\alpha = 2ml_m^2 + ml^2 + 2I$, $\beta = 2mll_m cos(q_R - q_L)$

Thank to given matrix H it was calculated Kinetic energy T, which is equal

$$T = \frac{1}{2}\dot{q}^T H \dot{q} \tag{2}$$

Due to robot works only in one plane (x and y), parallel to the ground the Potential energy is equal to zero.

$$U = 0 \tag{3}$$

First step to writing the dynamic equations is to choose the generalized coordinates q_L, q_R which fully determine the position of the system and T and U is defined as he total kinetic and total potential energy of the system. Then the concept of Lagrange function is introduced in the form:

$$L \equiv T - U \tag{4}$$

The form of dynamic equations of motion is written as follows:

$$\tau_B = \left[\frac{d}{dt}\right]\left(\frac{\delta L}{\delta \dot{q}}\right) - \left(\frac{\delta T}{\delta q}\right) \tag{5}$$

Lagrange function according to the equation (4)

$$L = T - U = T = \frac{1}{2}\dot{q}^T H \dot{q} \tag{6}$$

For the further calculations Kinetic energy was expanded

$$\frac{1}{2}\dot{q}^T H \dot{q} = \frac{1}{2}\begin{bmatrix} \dot{q_L} & \dot{q_R} \end{bmatrix} \begin{bmatrix} 2ml_m^2 + ml^2 + 2I & 2mll_m cos(q_R - q_L) \\ 2mll_m cos(q_R - q_L) & 2ml_m^2 + ml^2 + 2I \end{bmatrix} \begin{bmatrix} \dot{q_L} \\ \dot{q_R} \end{bmatrix} = \tag{7}$$

$$= \frac{1}{2}((2ml_m^2 + ml^2 + 2I)\dot{q_L}^2 + 2(2mll_m cos(q_R - q_L))\dot{q_L}\dot{q_R} + (2ml_m^2 + ml^2 + 2I)\dot{q_R}^2) \tag{8}$$

Determination of the corresponding derivatives according to the formula (5).

$$\left(\frac{\delta L}{\delta q}\right) = \left(\frac{\delta(\frac{1}{2}\dot{q}^T H \dot{q})}{\delta q}\right) = \left(\frac{\delta(2mll_m cos(q_R - q_L)\dot{q_L}\dot{q_R})}{\delta q}\right) = \begin{bmatrix} 2mll_m sin(q_R - q_L)(\dot{q_L}\dot{q_R}) \\ -2mll_m sin(q_R - q_L)(\dot{q_L}\dot{q_R}) \end{bmatrix} \tag{9}$$

Calculating derivative from the Kinetic energy T with respect to q it was important to notice that first and third part of the formula for Kinetic Energy simplified due to lack of q inside.
In the second part of the Dynamic equation it was used the chain rule.

$$\left[\frac{d}{dt}\right]\left(\frac{\delta L}{\delta \dot{q}}\right) = \dot{H}\begin{bmatrix} \dot{q_L} \\ \dot{q_R} \end{bmatrix} + H\begin{bmatrix} \ddot{q_L} \\ \ddot{q_R} \end{bmatrix} \tag{10}$$

$$\dot{H}\begin{bmatrix} \dot{q_L} \\ \dot{q_R} \end{bmatrix} = \begin{bmatrix} 2mll_m sin(q_R - q_L)(-\dot{q_R}^2 + \dot{q_L}\dot{q_R}) \\ 2mll_m sin(q_R - q_L)(\dot{q_L}^2 - \dot{q_L}\dot{q_R}) \end{bmatrix} \tag{11}$$

$$\left[\frac{d}{dt}\right]\left(\frac{\delta L}{\delta \dot{q}}\right) = H\begin{bmatrix} \ddot{q_L} \\ \ddot{q_R} \end{bmatrix} + \begin{bmatrix} 2mll_m sin(q_R - q_L)(-\dot{q_R}^2 + \dot{q_L}\dot{q_R}) \\ 2mll_m sin(q_R - q_L)(\dot{q_L}^2 - \dot{q_L}\dot{q_R}) \end{bmatrix} \tag{12}$$

The Lagrange equation for the robot is as follows.

$$\tau_B = H\begin{bmatrix} \ddot{q_L} \\ \ddot{q_R} \end{bmatrix} + \begin{bmatrix} 2mll_m sin(q_R - q_L)(-\dot{q_R}^2 + \dot{q_L}\dot{q_R}) \\ 2mll_m sin(q_R - q_L)(\dot{q_L}^2 - \dot{q_L}\dot{q_R}) \end{bmatrix} - \begin{bmatrix} 2mll_m sin(q_R - q_L)(\dot{q_L}\dot{q_R}) \\ -2mll_m sin(q_R - q_L)(\dot{q_L}\dot{q_R}) \end{bmatrix} \tag{13}$$

$$\tau_B = H\begin{bmatrix} \ddot{q_L} \\ \ddot{q_R} \end{bmatrix} + 2mll_m sin(q_R - q_L)\begin{bmatrix} -\dot{q_R}^2 \\ \dot{q_L}^2 \end{bmatrix} \tag{14}$$

3 Control

3.1 Feedback control

In the second part it was given the desired trajectory of the endpoint. The task was to program robot dynamics and control using linear feedback controller.

3.1.1 Matlab code

Listing 1: Matlab code for the task 1

```
%% Q2 - CONTROL

clear all
clc

syms q1 q2
m = 1; %[kg]  mass
l = 0.2; %[m]  length of the link
l_m = 0.1; %[m] distance from joint to center of mass
I = 0.01; %[kg/m^2]

alpha = 2*m*l_m^2 + m*l^2 + 2*I;
beta = 2*m*l*l_m*cos(q2-q1);
H = [alpha, beta; beta alpha]; % mass distribution matrix

dt = 1000;
T = 2;
t = 0:T/dt:T;
w = t/T;
% trajectory vector
xd = [(0.273 - 0.2*(6*w.^5 - 15*w.^4 + 10*w.^3)); (0.273 - 0.1*(6*w.^5 - 15*w.^4 + 10*w.^3))
    ];
% velocity vector
dxd = [(-3*(w.^4-2*w.^3+w.^2)); ((-1.5*(w.^4-2*w.^3+w.^2)))];
B = dxd';
% acceleration vector
ddxd = [(-1.5*(4*w.^3 - 6*w.^2 + 2*w)); ((-0.75*(4*w.^3 - 6*w.^2 + 2*w)))];

q1 = deg2rad(60);
q2 = deg2rad(30);
q1_0 = deg2rad(60);
q2_0 = deg2rad(30);

dQ = [];
Q1 = [];
Q2 = [];
ddQ1 = [];
ddQ1 = [];


for i = 1:dt

    % VELOCITY - DESIRED dQ
    J = l*[-sin(q1), -sin(q2); cos(q1), cos(q2)];
    dQ(:,i) = pinv(J) * B(i,:)';
    dQ1(i) = dQ(1,i);
    dQ2(i) = dQ(2,i);

    % POSITION - DESIRED Q
    Q1(i) = trapz(dQ1(1:i))*T/dt + q1_0;
    Q2(i) = trapz(dQ2(1:i))*T/dt + q2_0;
```

```
    q1 = Q1(i);
    q2 = Q2(i);

end

for i = 2:dt

    % ACCELERATION - DESIRED Q
    ddQ1(i) = (dQ1(i) - dQ1(i-1))/ T/dt;
    ddQ2(i) = (dQ2(i) - dQ2(i-1))/ T/dt;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FEEDBACK

new_Q1 = zeros(1,dt);
new_Q2 = zeros(1,dt);
new_dQ1 = zeros(1,dt);
new_dQ2 = zeros(1,dt);
new_ddQ1 = zeros(1,dt);
new_ddQ2 = zeros(1,dt);

new_Q1(1) = Q1(1);
new_Q2(1) = Q2(1);
new_dQ1(1) = dQ1(1);
new_dQ2(1) = dQ2(1);
new_ddQ1(1) = ddQ1(1);
new_ddQ2(1) = ddQ2(1);
new_Q1(2) = Q1(2);
new_Q2(2) = Q2(2);
new_dQ1(2) = dQ1(2);
new_dQ2(2) = dQ2(2);
new_ddQ1(2) = ddQ1(2);
new_ddQ2(2) = ddQ2(2);


tau1 = [];
tau2 = [];
new_ddQ = [];

K = 0.01; %[Nm]
k = 100; %[s]

Xfb(1,1) = xd(1,1);
Yfb(1,1) = xd(2,1);

for i = 2:dt

    % TAU
    e1 = Q1(i) - new_Q1(i);
    e2 = Q2(i) - new_Q2(i);
    de1 = dQ1(i) - new_dQ1(i);
    de2 = dQ2(i) - new_dQ2(i);

    tau1 = K*(e1 + k*de1);
    tau2 = K*(e2 + k*de2);
    Tau(i,:) = [tau1, tau2];


    % NEW ACCELERATION - ddQ
    alpha = 2*m*l_m^2 + m*l^2 + 2*I;
    beta = 2*m*l*l_m*cos(new_Q2(i)-new_Q1(i));
```

```
    H = [alpha, beta; beta alpha];
    new_H = pinv(H);
    C = 2*m*l*l_m*sin(new_Q2(i)-new_Q1(i));

    part_of_eq(i,:) = Tau(i,:) - C*[-(new_dQ2(i))^2; (new_dQ1(i))^2]';
    new_ddQ(i,:) = new_H * part_of_eq(i,:)';
    new_ddQ1(i) = new_ddQ(i,1);
    new_ddQ2(i) = new_ddQ(i,2);
    new_ddQ1(i+1) = new_ddQ(i,1);
    new_ddQ2(i+1) = new_ddQ(i,2);


    % REAL VELOCTY
    new_dQ1(i+1) = new_dQ1(i) + new_ddQ1(i)*(T/dt);
    new_dQ2(i+1) = new_dQ2(i) + new_ddQ2(i)*(T/dt);


    % REAL POSITION
    new_Q1(i+1) = new_Q1(i) + new_dQ1(i)*(T/dt);
    new_Q2(i+1) = new_Q2(i) + new_dQ2(i)*(T/dt);

    % ACTUAL X AND Y
    Xfb(1,i) = l*cos(new_Q2(1,i))+l*cos(new_Q1(1,i));
    Yfb(1,i) = l*sin(new_Q2(1,i))+l*sin(new_Q1(1,i));

    % ROBOT POSITION
    RIGHTarm_fb(:,i)=l*[cos(new_Q2(1,i));sin(new_Q2(1,i))];
    LEFTarm_fb(:,i)=l*[cos(new_Q1(1,i));sin(new_Q1(1,i))];



end



%% PLOTS 2A

t = linspace(0,T,dt);

new_Q1(dt+1) = [];
new_Q2(dt+1) = [];
new_dQ1(dt+1) = [];
new_dQ2(dt+1) = [];
xd(:,dt+1) = [];



% DESIRED AND ACTUAL ANGLE - AGAINST TIME
figure(1)
subplot(1,2,1)
plot(t,rad2deg(Q1), 'k--'); hold on; plot(t, rad2deg(new_Q1), 'b');
title('DESIRED AND ACTUAL QL ANGLE');
legend('desired value', 'feedback');
xlabel('Time [s]');
ylabel('Angle [\circ]')
subplot(1,2,2)
plot(t,rad2deg(Q2), 'k--'); hold on; plot(t, rad2deg(new_Q2), 'b');
title('DESIRED AND ACTUAL QR ANGLE')
legend('desired value', 'feedback')
xlabel('Time [s]');
ylabel('Angle [\circ]')

% DESIRED AND ACTUAL ENDPOINT POSITIONS X AND Y DIRECTIONS - AGAINST TIME
figure(2)
subplot(1,2,1)
```

```
plot(t,xd(1,:), 'k—'); hold on; plot(t, Xfb, 'b');
title('DESIRED AND ACTUAL ENDPOINT X POSITION');
legend('desired value', 'feedback');
xlabel('Time [s]');
ylabel('Position [m]')
subplot(1,2,2)
plot(t,xd(2,:), 'k—'); hold on; plot(t, Yfb, 'b');
title('DESIRED AND ACTUAL ENDPOINT Y POSITION')
legend('desired value', 'feedback')
xlabel('Time [s]');
ylabel('Position [m]')


% DESIRED AND ACTUAL ENDPOINT TRAJECTORIES – IN THE X–Y PLANE
figure(3)
plot(xd(1,:),xd(2,:), 'k—'); hold on; plot(Xfb, Yfb, 'b');
title('DESIRED AND ACTUAL ENDPOINT TRAJECTORIES');
legend('desired value', 'feedback');
xlabel('x axis [m]');
ylabel('y axis [m]')
```

3.1.2 Desired and actual angles against time

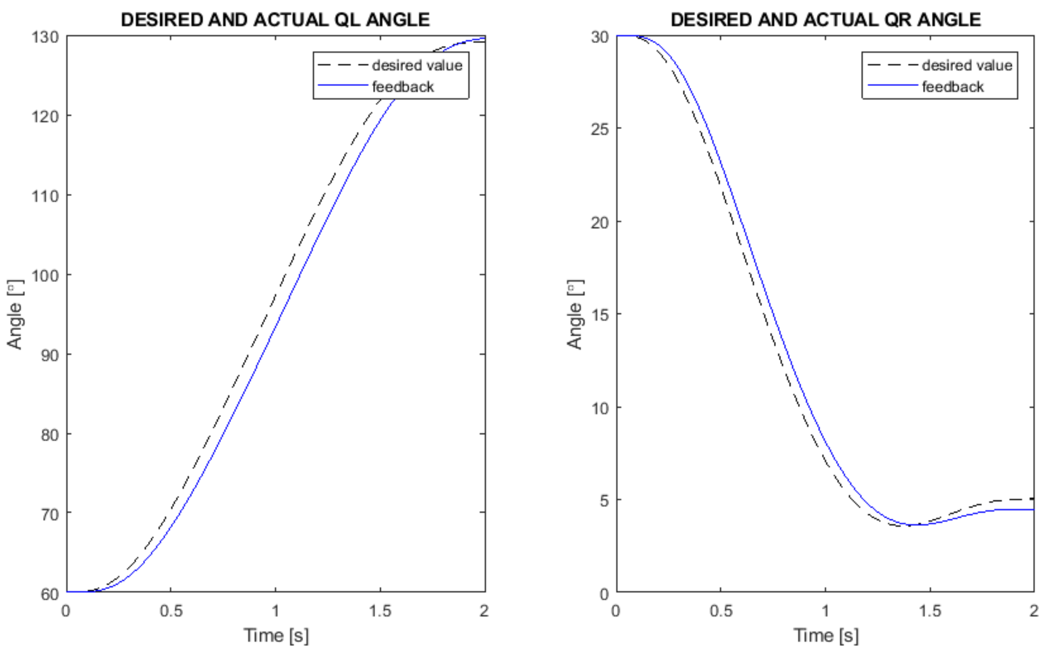

Figure 2: Plot of the desired and actual angle against time

3.1.3 Desired and actual endpoint positions in x and y directions against time

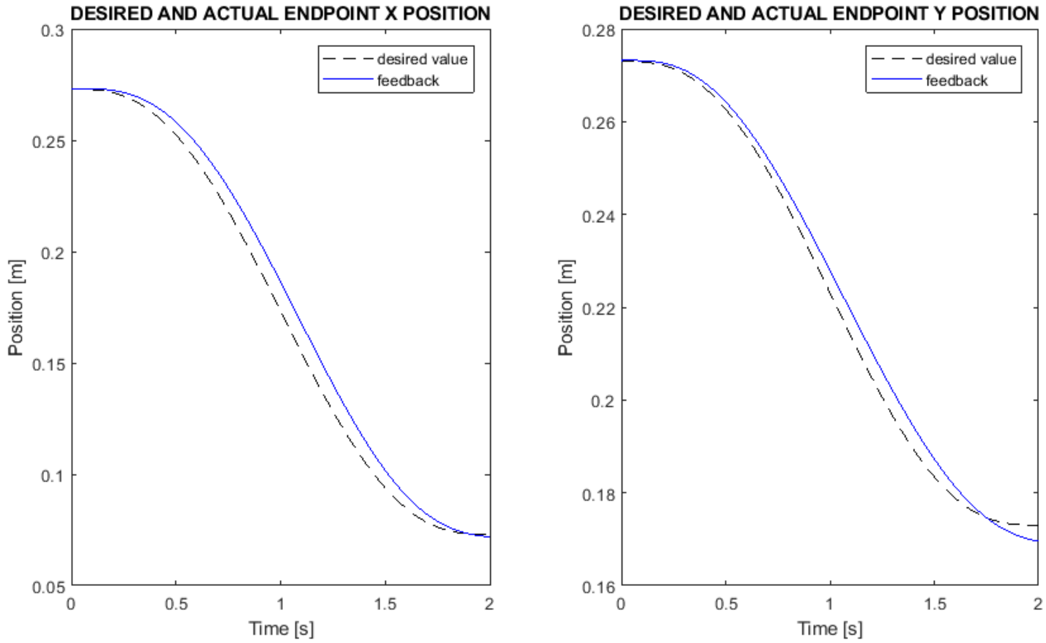

Figure 3: Plot of the desired and actual endpoint positions x and y directions against time

3.1.4 Desired and actual endpoint trajectories in x-y plane

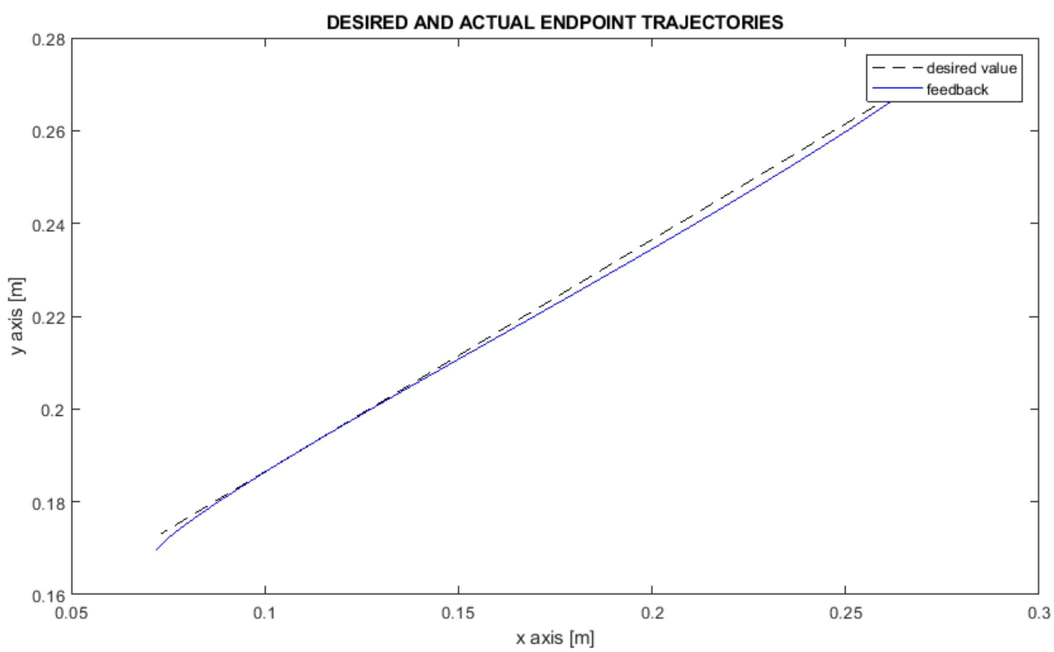

Figure 4: Plot of the desired and actual endpoint trajectories in the x-y plane

3.1.5 Conclusion after changing values of the parameter K

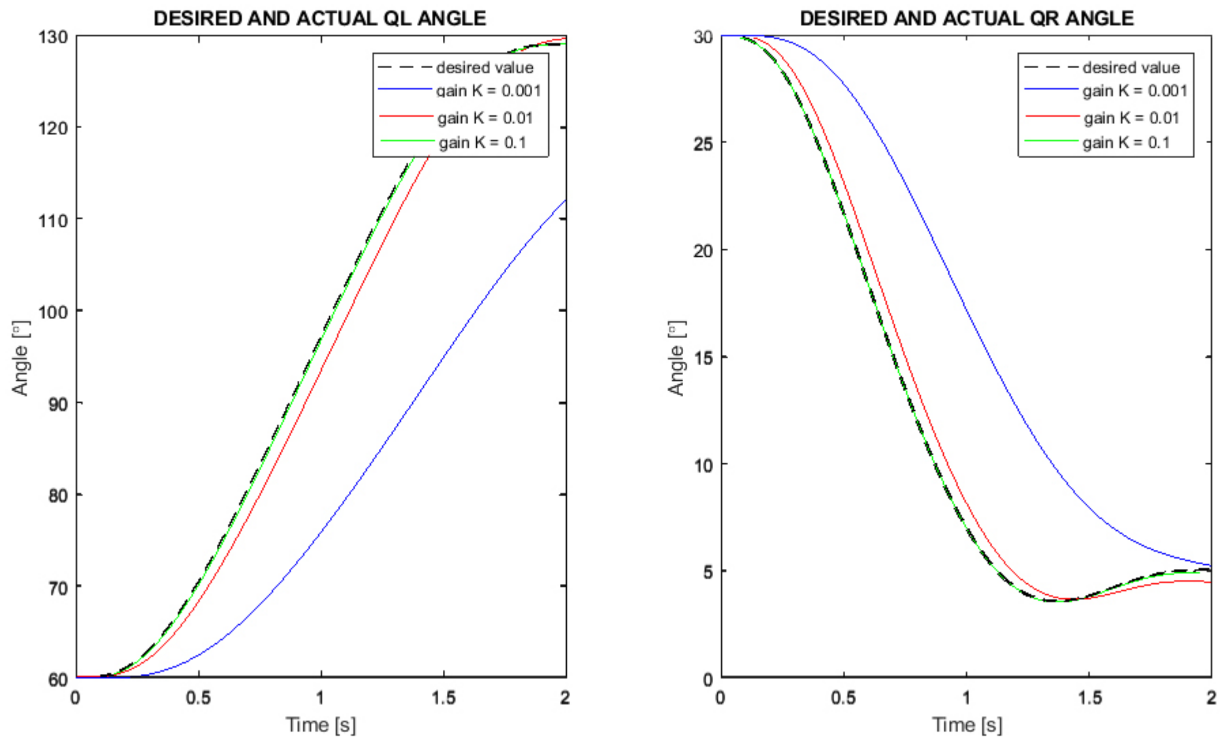

Figure 5: Different values of parameter K

Plot above shows that greater value of the parameter K gives bigger precision to the robot, while too low value causes a large delay, slows down reactions giving greater freedom.

3.2 Feedforward control

To compensate the effect of dynamics it was used feedforward controller together with feedback controller.

3.2.1 Matlab code

Listing 2: Matlab code for the task 1

```
%% Q2 - CONTROL

clear all
clc

syms q1 q2
m = 1; %[kg]  mass
l = 0.2; %[m]  length of the link
l_m = 0.1; %[m] distance from joint to center of mass
I = 0.01; %[kg/m^2]

alpha = 2*m*l_m^2 + m*l^2 + 2*I;
```

```
beta = 2*m*l*l_m*cos(q2-q1);
H = [alpha, beta; beta alpha]; % mass distribution matrix

dt = 1000;
T = 2;
t = 0:T/dt:T;
w = t/T;
% trajectory vector
xd = [(0.273 - 0.2*(6*w.^5 - 15*w.^4 + 10*w.^3)); (0.273 - 0.1*(6*w.^5 - 15*w.^4 + 10*w.^3))];
% velocity vector
dxd = [(-3*(w.^4-2*w.^3+w.^2)); ((-1.5*(w.^4-2*w.^3+w.^2)))];
B = dxd';
% acceleration vector
ddxd = [(-1.5*(4*w.^3 - 6*w.^2 + 2*w)); ((-0.75*(4*w.^3 - 6*w.^2 + 2*w)))];

q1 = deg2rad(60);
q2 = deg2rad(30);
q1_0 = deg2rad(60);
q2_0 = deg2rad(30);

dQ = [];
Q1 = [];
Q2 = [];
ddQ1 = [];
ddQ1 = [];


for i = 1:dt

    % VELOCITY - DESIRED dQ
    J = l*[-sin(q1), -sin(q2); cos(q1), cos(q2)];
    dQ(:,i) = pinv(J) * B(i,:)';
    dQ1(i) = dQ(1,i);
    dQ2(i) = dQ(2,i);

    % POSITION - DESIRED Q
    Q1(i) = trapz(dQ1(1:i))*T/dt + q1_0;
    Q2(i) = trapz(dQ2(1:i))*T/dt + q2_0;

    q1 = Q1(i);
    q2 = Q2(i);

end

for i = 2:dt

    % ACCELERATION - DESIRED Q
    ddQ1(i) = (dQ1(i) - dQ1(i-1))/ T/dt;
    ddQ2(i) = (dQ2(i) - dQ2(i-1))/ T/dt;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FEEDBACK

new_Q1 = zeros(1,dt);
new_Q2 = zeros(1,dt);
new_dQ1 = zeros(1,dt);
new_dQ2 = zeros(1,dt);
new_ddQ1 = zeros(1,dt);
new_ddQ2 = zeros(1,dt);

new_Q1(1) = Q1(1);
```

```
new_Q2(1) = Q2(1);
new_dQ1(1) = dQ1(1);
new_dQ2(1) = dQ2(1);
new_ddQ1(1) = ddQ1(1);
new_ddQ2(1) = ddQ2(1);
new_Q1(2) = Q1(2);
new_Q2(2) = Q2(2);
new_dQ1(2) = dQ1(2);
new_dQ2(2) = dQ2(2);
new_ddQ1(2) = ddQ1(2);
new_ddQ2(2) = ddQ2(2);


tau1 = [];
tau2 = [];
new_ddQ = [];

K = 0.01; %[Nm]
k = 100; %[s]

Xfb(1,1) = xd(1,1);
Yfb(1,1) = xd(2,1);
Xff(1,1) = xd(1,1);
Yff(1,1) = xd(2,1);

for i = 2:dt

    % TAU
    e1 = Q1(i) - new_Q1(i);
    e2 = Q2(i) - new_Q2(i);
    de1 = dQ1(i) - new_dQ1(i);
    de2 = dQ2(i) - new_dQ2(i);

    tau1 = K*(e1 + k*de1);
    tau2 = K*(e2 + k*de2);
    Tau(i,:) = [tau1, tau2];


    % NEW ACCELERATION - ddQ
    alpha = 2*m*l_m^2 + m*l^2 + 2*I;
    beta = 2*m*l*l_m*cos(new_Q2(i)-new_Q1(i));
    H = [alpha, beta; beta alpha];
    new_H = pinv(H);
    C = 2*m*l*l_m*sin(new_Q2(i)-new_Q1(i));

    part_of_eq(i,:) = Tau(i,:) - C*[-(new_dQ2(i))^2; (new_dQ1(i))^2]';
    new_ddQ(i,:) = new_H * part_of_eq(i,:)';
    new_ddQ1(i) = new_ddQ(i,1);
    new_ddQ2(i) = new_ddQ(i,2);
    new_ddQ1(i+1) = new_ddQ(i,1);
    new_ddQ2(i+1) = new_ddQ(i,2);


    % REAL VELOCTY
    new_dQ1(i+1) = new_dQ1(i) + new_ddQ1(i)*(T/dt);
    new_dQ2(i+1) = new_dQ2(i) + new_ddQ2(i)*(T/dt);


    % REAL POSITION
    new_Q1(i+1) = new_Q1(i) + new_dQ1(i)*(T/dt);
    new_Q2(i+1) = new_Q2(i) + new_dQ2(i)*(T/dt);

    % ACTUAL X AND Y
    Xfb(1,i) = l*cos(new_Q2(1,i))+l*cos(new_Q1(1,i));
    Yfb(1,i) = l*sin(new_Q2(1,i))+l*sin(new_Q1(1,i));
```

```
    % ROBOT POSITION
    RIGHTarm_fb(:,i)=l*[cos(new_Q2(1,i));sin(new_Q2(1,i))];
    LEFTarm_fb(:,i)=l*[cos(new_Q1(1,i));sin(new_Q1(1,i))];



end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FEEDBACK + FEEDFORWARD

TauFB = Tau';

new2_Q1 = zeros(1,dt);
new2_Q2 = zeros(1,dt);
new2_dQ1 = zeros(1,dt);
new2_dQ2 = zeros(1,dt);
new2_ddQ1 = zeros(1,dt);
new2_ddQ2 = zeros(1,dt);

new2_Q1(1) = Q1(1);
new2_Q2(1) = Q2(1);
new2_dQ1(1) = dQ1(1);
new2_dQ2(1) = dQ2(1);
new2_ddQ1(1) = ddQ1(1);
new2_ddQ2(1) = ddQ2(1);
new2_Q1(2) = Q1(2);
new2_Q2(2) = Q2(2);
new2_dQ1(2) = dQ1(2);
new2_dQ2(2) = dQ2(2);
new2_ddQ1(2) = ddQ1(2);
new2_ddQ2(2) = ddQ2(2);

new2_tau1 = [];
new2_tau2 = [];
new2_TauFB = [];


for i = 2:dt

    % TAU
    J_dot = l*[-cos(Q1(i))*dQ1(i), - cos(Q2(i))*dQ2(i); -sin(Q1(i))*dQ1(i), -sin(Q2(i))*dQ2(i)];
    J = l*[-sin(Q1(i)), -sin(Q2(i)); cos(Q1(i)), cos(Q2(i))];
    J = pinv(J);

    feedback(:,i) = J * ([ddxd(1,i);ddxd(2,i)] - J_dot*[dQ1(1,i); dQ2(1,i)]);

    alpha_2 = 2*m*l_m^2 + m*l^2 + 2*I;
    beta_2 = 2*m*l*l_m*cos(Q2(i)-Q1(i));
    H_des = [alpha_2, beta_2; beta_2 alpha_2];
    tauFF1 = H_des(1,1)*feedback(1,i)+H_des(1,2)*feedback(2,i) + 2*m*l*l_m*sin(Q2(i)-Q1(i))*(-(dQ2(i))^2);
    tauFF2 = H_des(2,1)*feedback(1,i)+H_des(2,2)*feedback(2,i) + 2*m*l*l_m*sin(Q2(i)-Q1(i))*((dQ1(i))^2);
    TauFF(:,i) = [tauFF1; tauFF2];


    % ERROR
    new2_e1 = Q1(i) - new2_Q1(i);
    new2_e2 = Q2(i) - new2_Q2(i);
    new2_de1 = dQ1(i) - new2_dQ1(i);
```

```
    new2_de2 = dQ2(i) - new2_dQ2(i);

    new2_tau1 = K*(new2_e1 + k*new2_de1);
    new2_tau2 = K*(new2_e2 + k*new2_de2);
    new2_TauFB(:,i) = [new2_tau1, new2_tau2];


    TauFinal(:,i) = TauFF(:,i) + new2_TauFB(:,i);
    TauFinalT = TauFinal';

    % NEW ACCELERATION - ddQ
    alpha_3 = 2*m*l_m^2 + m*l^2 + 2*I;
    beta_3 = 2*m*l*l_m*cos(new2_Q2(i)-new2_Q1(i));
    H = [alpha_3, beta_3; beta_3 alpha_3];
    new_H = pinv(H);
    delta = new_H(1,1);
    gamma = new_H(1,2);
    C = 2*m*l*l_m*sin(new2_Q2(i)-new2_Q1(i));

    new2_Tau(i,:) = TauFinalT(i,:) - C*[-(new2_dQ2(i))^2; (new2_dQ1(i))^2]';
    new2_ddQ(i,:) = new_H * new2_Tau(i,:)';
    new2_ddQ1(i) = new2_ddQ(i,1);
    new2_ddQ2(i) = new2_ddQ(i,2);


    % REAL VELOCTY
    new2_dQ1(i+1) = new2_dQ1(i) + new2_ddQ1(i)*(T/dt);
    new2_dQ2(i+1) = new2_dQ2(i) + new2_ddQ2(i)*(T/dt);


    % REAL POSITION
    new2_Q1(i+1) = new2_Q1(i) + new2_dQ1(i)*(T/dt);
    new2_Q2(i+1) = new2_Q2(i) + new2_dQ2(i)*(T/dt);


    % ACTUAL X AND Y
    Xff(1,i) = l*cos(new2_Q2(1,i))+l*cos(new2_Q1(1,i));
    Yff(1,i) = l*sin(new2_Q2(1,i))+l*sin(new2_Q1(1,i));


end




%% PLOTS 2B

t = linspace(0,T,dt);

new_Q1(dt+1) = [];
new_Q2(dt+1) = [];
new_dQ1(dt+1) = [];
new_dQ2(dt+1) = [];
new2_Q1(dt+1) = [];
new2_Q2(dt+1) = [];
new2_dQ1(dt+1) = [];
new2_dQ2(dt+1) = [];
xd(:,dt+1) = [];



% DESIRED AND ACTUAL ANGLE - AGAINST TIME
figure(4)
subplot(1,2,1)
```

```
plot(t,rad2deg(Q1), 'k--'); hold on; plot(t, rad2deg(new_Q1), 'b'); hold on; plot(t, rad2deg(new2_Q1), 'r');
title('DESIRED AND ACTUAL QL ANGLE');
legend('desired value', 'feedback', 'feedforward + feedback');
xlabel('Time [s]');
ylabel('Angle [\circ]')
subplot(1,2,2)
plot(t,rad2deg(Q2), 'k--'); hold on; plot(t, rad2deg(new_Q2), 'b'); hold on; plot(t, rad2deg(new2_Q2), 'r');
title('DESIRED AND ACTUAL QR ANGLE')
legend('desired value', 'feedback', 'feedforward + feedback')
xlabel('Time [s]');
ylabel('Angle [\circ]')

% DESIRED AND ACTUAL ENDPOINT POSITIONS X AND Y DIRECTIONS - AGAINST TIME
figure(5)
subplot(1,2,1)
plot(t,xd(1,:), 'k--'); hold on; plot(t, Xfb, 'b'); hold on; plot(t, Xff, 'r');
title('DESIRED AND ACTUAL ENDPOINT X POSITION');
legend('desired value', 'feedback', 'feedforward + feedback');
xlabel('Time [s]');
ylabel('Position [m]')
subplot(1,2,2)
plot(t,xd(2,:), 'k--'); hold on; plot(t, Yfb, 'b'); hold on; plot(t, Yff, 'r');
title('DESIRED AND ACTUAL ENDPOINT Y POSITION')
legend('desired value', 'feedback', 'feedforward + feedback')
xlabel('Time [s]');
ylabel('Position [m]')


% DESIRED AND ACTUAL ENDPOINT TRAJECTORIES - IN THE X-Y PLANE
figure(6)
plot(xd(1,:),xd(2,:), 'k--'); hold on; plot(Xfb, Yfb, 'b'); hold on; plot(Xff, Yff, 'r');
title('DESIRED AND ACTUAL ENDPOINT TRAJECTORIES');
legend('desired value', 'feedback', 'feedforward + feedback');
xlabel('x axis [m]');
ylabel('y axis [m]')
```

3.2.2 Desired and actual angles against time

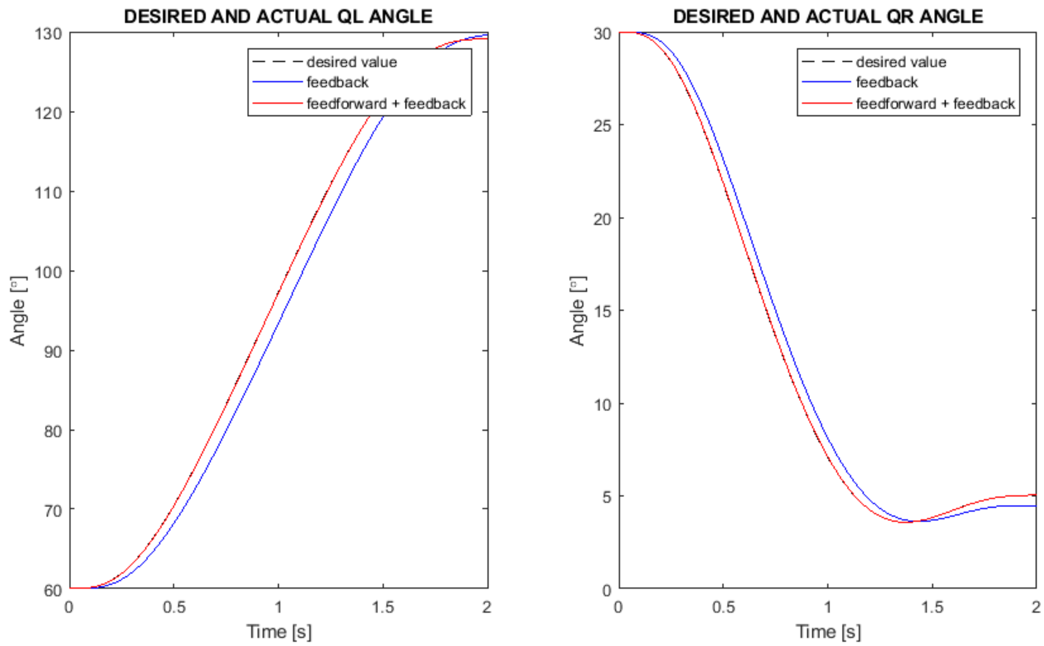

Figure 6: Plot of the desired and actual angle against time

3.2.3 Desired and actual endpoint positions in x and y directions against time

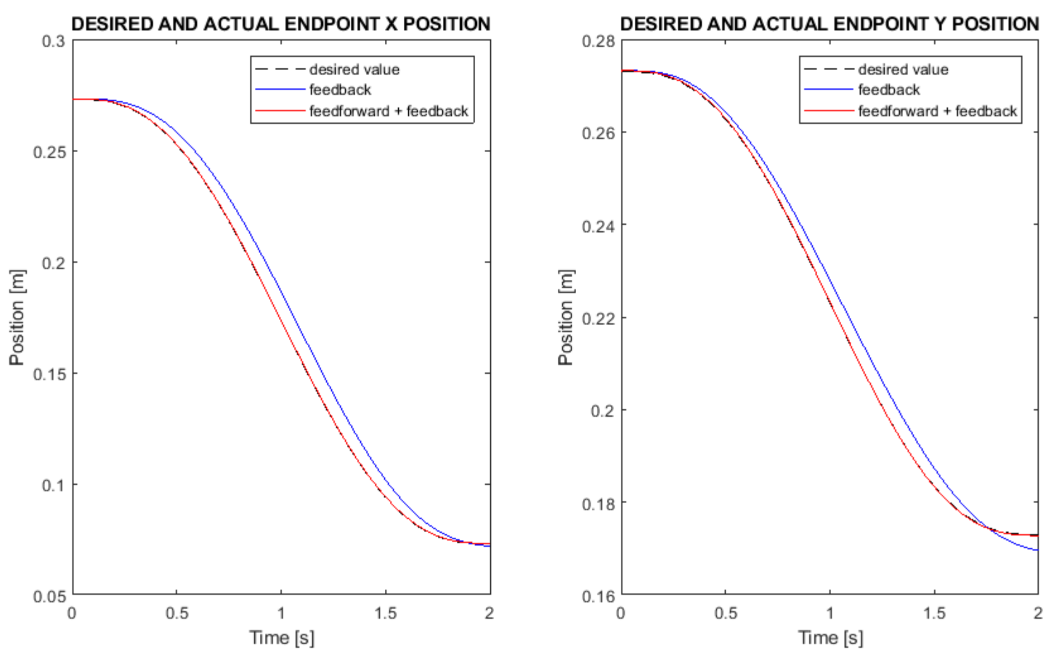

Figure 7: Plot of the desired and actual endpoint positions x and y directions against time

3.2.4 Desired and actual endpoint trajectories in x-y plane

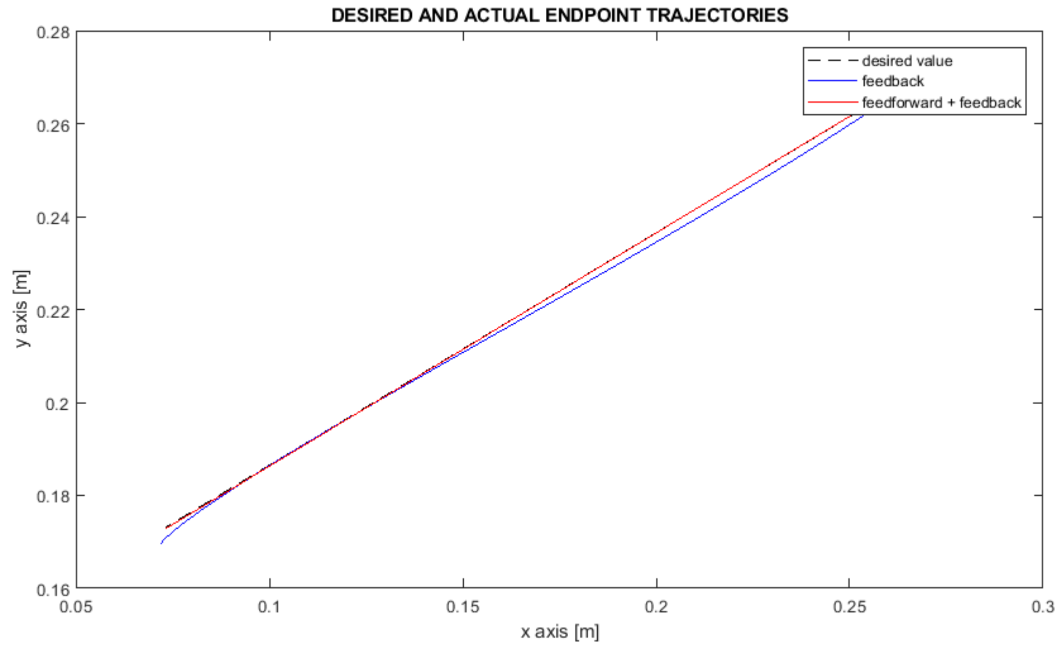

Figure 8: Plot of the desired and actual endpoint trajectories in the x-y plane

3.2.5 Improved through Feedforward controller

Feedback controller measures the error between the the desired value and real value of the angle of the robot. It responds to what happened in the past and that is why it will never goes the same path as the desired one. While the feedforward on the basis of the desired values and implemented values about the parameters of the structure of the robot can predict the behaviour of the robot in the next step. Thank to that it is possible to completely eliminate the error between the current and desire value.