

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.

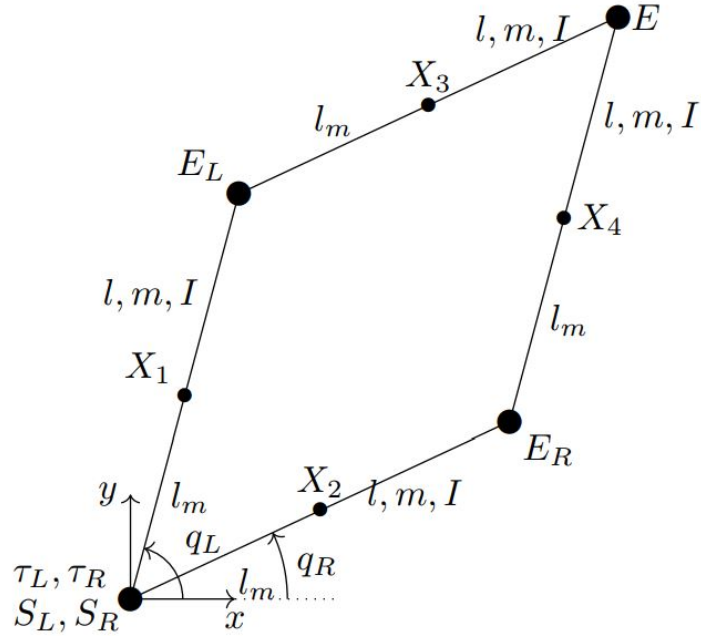


Figure 1: Parallel robot diagram

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.

$$H = \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix} \quad (1)$$

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

$$T = \frac{1}{2} \dot{q}^T H \dot{q} \quad (2)$$

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

$$U = 0 \quad (3)$$

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 the total kinetic and total potential energy of the system. Then the concept of Lagrange function is introduced in the form:

$$L \equiv T - U \quad (4)$$

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

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

Lagrange function according to the equation (4)

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

For the further calculations Kinetic energy was expanded

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

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

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

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

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.

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

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

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

The Lagrange equation for the robot is as follows.

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

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

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

```
1 %% Q2 - CONTROL
2
3 clear all
4 clc
5
6 syms q1 q2
7 m = 1; %[kg] mass
8 l = 0.2; %[m] length of the link
9 l.m = 0.1; %[m] distance from joint to center of mass
10 I = 0.01; %[kg/m^2]
11
12 alpha = 2*m*l.m^2 + m*l^2 + 2*I;
13 beta = 2*m*l*l.m*cos(q2-q1);
14 H = [alpha, beta; beta alpha]; % mass distribution matrix
15
16 dt = 1000;
17 T = 2;
18 t = 0:T/dt:T;
19 w = t/T;
20 % trajectory vector
21 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))];
22 % velocity vector
23 dxd = [(-3*(w.^4 - 2*w.^3 + w.^2)); ((-1.5*(w.^4 - 2*w.^3 + w.^2)))]';
24 B = dxd';
25 % acceleration vector
26 ddx = [(-1.5*(4*w.^3 - 6*w.^2 + 2*w)); ((-0.75*(4*w.^3 - 6*w.^2 + 2*w)))]';
27
28 q1 = deg2rad(60);
29 q2 = deg2rad(30);
30 q1_0 = deg2rad(60);
31 q2_0 = deg2rad(30);
32
33 dQ = [];
34 Q1 = [];
35 Q2 = [];
36 ddQ1 = [];
37 ddQ2 = [];
38
39
40 for i = 1:dt
41
42     % VELOCITY - DESIRED dQ
43     J = l*[-sin(q1), -sin(q2); cos(q1), cos(q2)];
44     dQ(:, i) = pinv(J) * B(i, :)';
45     dQ1(i) = dQ(1, i);
46     dQ2(i) = dQ(2, i);
47
48     % POSITION - DESIRED Q
49     Q1(i) = trapz(dQ1(1:i))*T/dt + q1_0;
50     Q2(i) = trapz(dQ2(1:i))*T/dt + q2_0;
```

```

51
52     q1 = Q1(i);
53     q2 = Q2(i);
54
55 end
56
57 for i = 2:dt
58
59     % ACCELERATION - DESIRED Q
60     ddQ1(i) = (dQ1(i) - dQ1(i-1))/ T/dt;
61     ddQ2(i) = (dQ2(i) - dQ2(i-1))/ T/dt;
62
63 end
64
65
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 %% FEEDBACK
68
69 new_Q1 = zeros(1,dt);
70 new_Q2 = zeros(1,dt);
71 new_dQ1 = zeros(1,dt);
72 new_dQ2 = zeros(1,dt);
73 new_ddQ1 = zeros(1,dt);
74 new_ddQ2 = zeros(1,dt);
75
76 new_Q1(1) = Q1(1);
77 new_Q2(1) = Q2(1);
78 new_dQ1(1) = dQ1(1);
79 new_dQ2(1) = dQ2(1);
80 new_ddQ1(1) = ddQ1(1);
81 new_ddQ2(1) = ddQ2(1);
82 new_Q1(2) = Q1(2);
83 new_Q2(2) = Q2(2);
84 new_dQ1(2) = dQ1(2);
85 new_dQ2(2) = dQ2(2);
86 new_ddQ1(2) = ddQ1(2);
87 new_ddQ2(2) = ddQ2(2);
88
89
90 tau1 = [];
91 tau2 = [];
92 new_ddQ = [];
93
94 K = 0.01; % [Nm]
95 k = 100; % [s]
96
97 Xfb(1,1) = xd(1,1);
98 Yfb(1,1) = xd(2,1);
99
100 for i = 2:dt
101
102     % TAU
103     e1 = Q1(i) - new_Q1(i);
104     e2 = Q2(i) - new_Q2(i);
105     de1 = dQ1(i) - new_dQ1(i);
106     de2 = dQ2(i) - new_dQ2(i);
107
108     tau1 = K*(e1 + k*de1);
109     tau2 = K*(e2 + k*de2);
110     Tau(i,:) = [tau1, tau2];
111
112
113     % NEW ACCELERATION - ddQ
114     alpha = 2*m*l_m^2 + m*l^2 + 2*I;
115     beta = 2*m*l*l_m*cos(new_Q2(i)-new_Q1(i));

```

```

116 H = [alpha, beta; beta alpha];
117 new_H = pinv(H);
118 C = 2*m*l*l_m*sin(new_Q2(i)-new_Q1(i));
119
120 part_of_eq(i,:) = Tau(i,:) - C*[-(new_dQ2(i))^2; (new_dQ1(i))^2]';
121 new_ddQ(i,:) = new_H * part_of_eq(i,:)';
122 new_ddQ1(i) = new_ddQ(i,1);
123 new_ddQ2(i) = new_ddQ(i,2);
124 new_ddQ1(i+1) = new_ddQ(i,1);
125 new_ddQ2(i+1) = new_ddQ(i,2);
126
127
128 % REAL VELOCITY
129 new_dQ1(i+1) = new_dQ1(i) + new_ddQ1(i)*(T/dt);
130 new_dQ2(i+1) = new_dQ2(i) + new_ddQ2(i)*(T/dt);
131
132
133 % REAL POSITION
134 new_Q1(i+1) = new_Q1(i) + new_dQ1(i)*(T/dt);
135 new_Q2(i+1) = new_Q2(i) + new_dQ2(i)*(T/dt);
136
137 % ACTUAL X AND Y
138 Xfb(1,i) = l*cos(new_Q2(1,i))+l*cos(new_Q1(1,i));
139 Yfb(1,i) = l*sin(new_Q2(1,i))+l*sin(new_Q1(1,i));
140
141 % ROBOT POSITION
142 RIGHTarm_fb(:,i)=l*[cos(new_Q2(1,i));sin(new_Q2(1,i))];
143 LEFTarm_fb(:,i)=l*[cos(new_Q1(1,i));sin(new_Q1(1,i))];
144
145
146
147 end
148
149
150
151 %% PLOTS 2A
152
153 t = linspace(0,T,dt);
154
155 new_Q1(dt+1) = [];
156 new_Q2(dt+1) = [];
157 new_dQ1(dt+1) = [];
158 new_dQ2(dt+1) = [];
159 xd(:,dt+1) = [];
160
161
162
163 % DESIRED AND ACTUAL ANGLE – AGAINST TIME
164 figure(1)
165 subplot(1,2,1)
166 plot(t,rad2deg(Q1), 'k—'); hold on; plot(t, rad2deg(new_Q1), 'b');
167 title('DESIRED AND ACTUAL QL ANGLE');
168 legend('desired value', 'feedback');
169 xlabel('Time [s]');
170 ylabel('Angle [\circ]');
171 subplot(1,2,2)
172 plot(t,rad2deg(Q2), 'k—'); hold on; plot(t, rad2deg(new_Q2), 'b');
173 title('DESIRED AND ACTUAL QR ANGLE');
174 legend('desired value', 'feedback');
175 xlabel('Time [s]');
176 ylabel('Angle [\circ]');
177
178 % DESIRED AND ACTUAL ENDPOINT POSITIONS X AND Y DIRECTIONS – AGAINST TIME
179 figure(2)
180 subplot(1,2,1)

```

```

181 plot(t,xd(1,:), 'k—'); hold on; plot(t, Xfb, 'b');
182 title('DESIRED AND ACTUAL ENDPOINT X POSITION');
183 legend('desired value', 'feedback');
184 xlabel('Time [s]');
185 ylabel('Position [m]')
186 subplot(1,2,2)
187 plot(t,xd(2,:), 'k—'); hold on; plot(t, Yfb, 'b');
188 title('DESIRED AND ACTUAL ENDPOINT Y POSITION')
189 legend('desired value', 'feedback')
190 xlabel('Time [s]');
191 ylabel('Position [m]')
192
193
194 % DESIRED AND ACTUAL ENDPOINT TRAJECTORIES – IN THE X-Y PLANE
195 figure(3)
196 plot(xd(1,:),xd(2,:), 'k—'); hold on; plot(Xfb, Yfb, 'b');
197 title('DESIRED AND ACTUAL ENDPOINT TRAJECTORIES');
198 legend('desired value', 'feedback');
199 xlabel('x axis [m]');
200 ylabel('y axis [m]')

```

3.1.2 Desired and actual angles against time

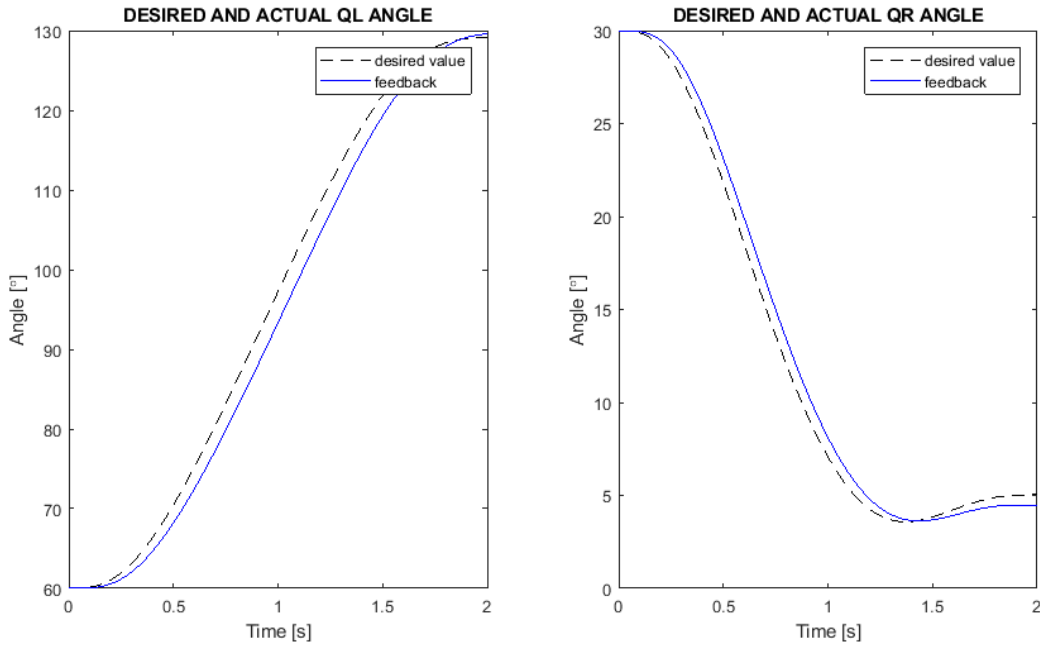


Figure 2: Plot of the desired and actual angle against time

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

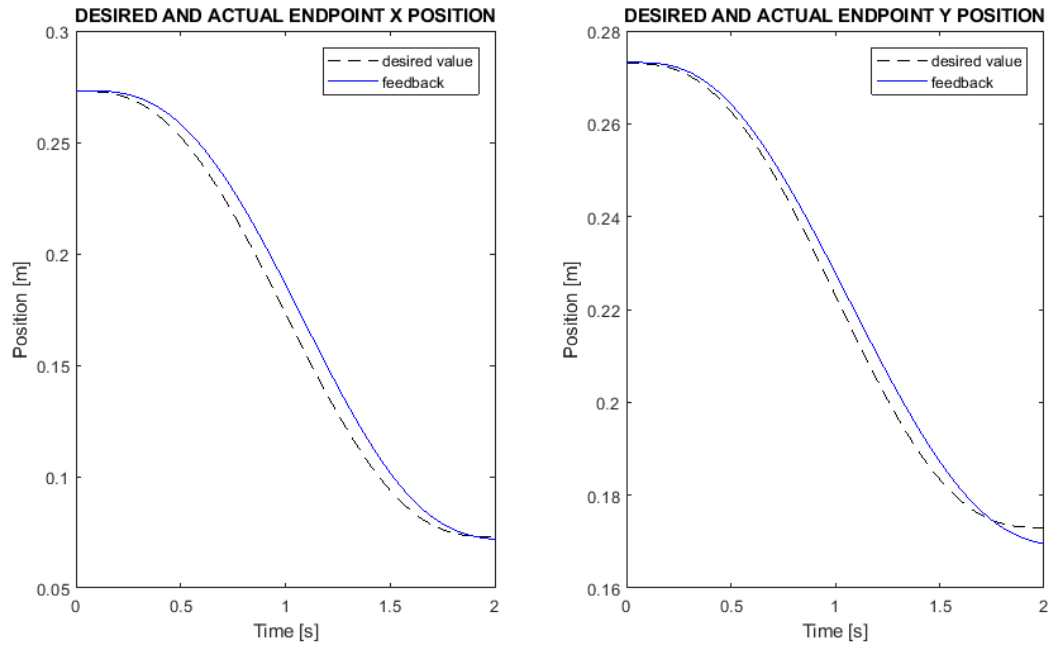


Figure 3: Plot of the desired and actual endpoint positions x and y directions against time

3.1.4 Desired and actual endpoint trajectories in x-y plane

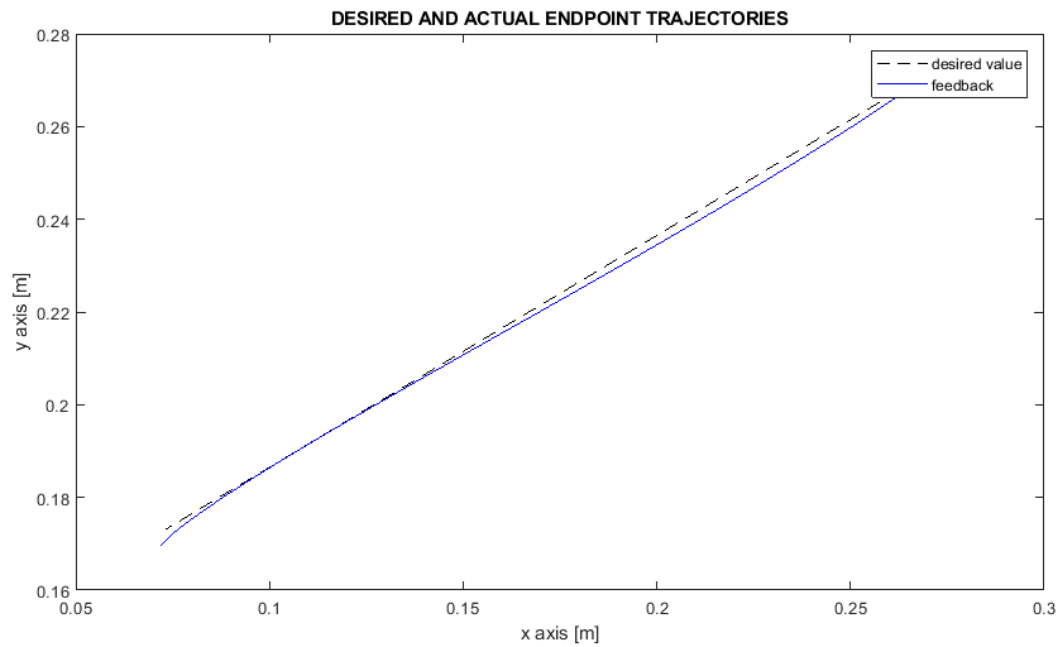


Figure 4: Plot of the desired and actual endpoint trajectories in the x-y plane

3.1.5 Conclusion after changing values of the parameter K

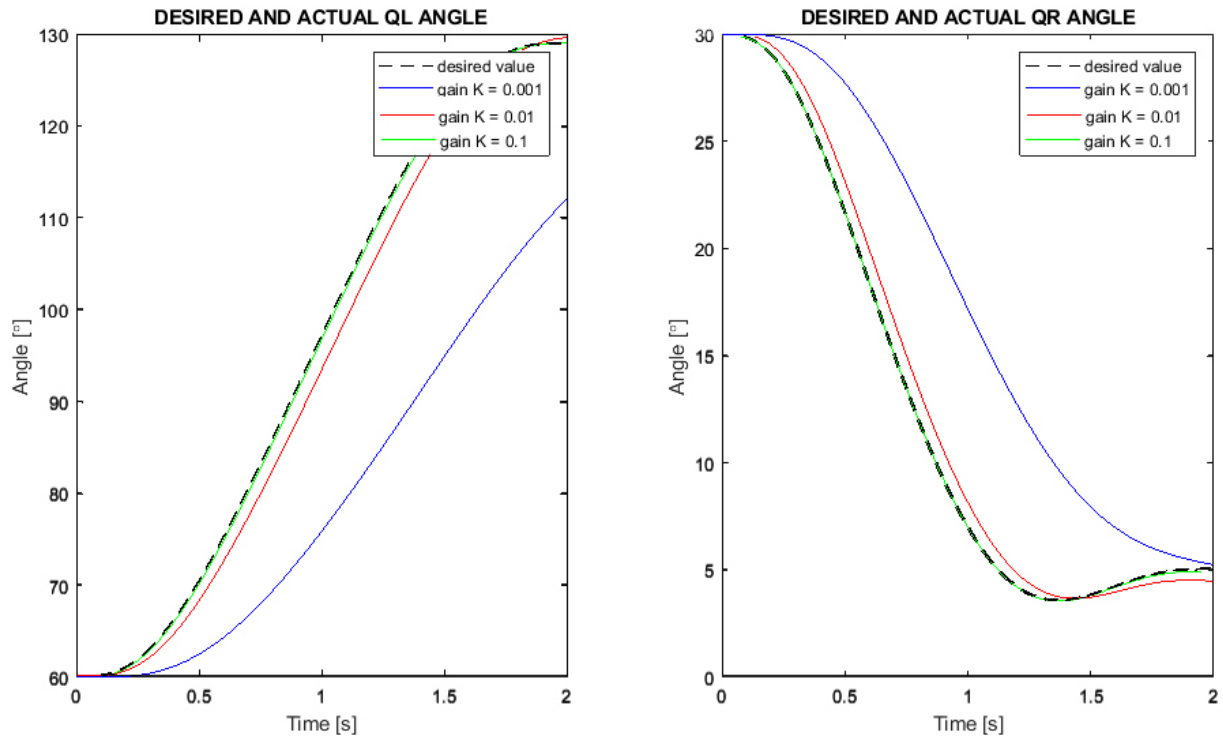


Figure 5: Different values of parameter K

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

```

1 %% Q2 - CONTROL
2
3 clear all
4 clc
5
6 syms q1 q2
7 m = 1; %[kg] mass
8 l = 0.2; %[m] length of the link
9 l.m = 0.1; %[m] distance from joint to center of mass
10 I = 0.01; %[kg/m^2]
11
12 alpha = 2*m*l.m^2 + m*l^2 + 2*I;

```

```

13 beta = 2*m*l*1.m*cos(q2-q1);
14 H = [alpha, beta; beta alpha]; % mass distribution matrix
15
16 dt = 1000;
17 T = 2;
18 t = 0:T/dt:T;
19 w = t/T;
20 % trajectory vector
21 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))
22 ];
23 % velocity vector
24 dxd = [(-3*(w.^4-2*w.^3+w.^2)); ((-1.5*(w.^4-2*w.^3+w.^2)))]';
25 B = dxd';
26 % acceleration vector
27 ddx = [(-1.5*(4*w.^3 - 6*w.^2 + 2*w)); ((-0.75*(4*w.^3 - 6*w.^2 + 2*w)))]';
28
29 q1 = deg2rad(60);
30 q2 = deg2rad(30);
31 q1_0 = deg2rad(60);
32 q2_0 = deg2rad(30);
33
34 dQ = [];
35 Q1 = [];
36 Q2 = [];
37 ddQ1 = [];
38 ddQ2 = [];
39
40 for i = 1:dt
41
42     % VELOCITY - DESIRED dQ
43     J = l*[-sin(q1), -sin(q2); cos(q1), cos(q2)];
44     dQ(:,i) = pinv(J) * B(i,:)';
45     dQ1(i) = dQ(1,i);
46     dQ2(i) = dQ(2,i);
47
48     % POSITION - DESIRED Q
49     Q1(i) = trapz(dQ1(1:i))*T/dt + q1_0;
50     Q2(i) = trapz(dQ2(1:i))*T/dt + q2_0;
51
52     q1 = Q1(i);
53     q2 = Q2(i);
54
55 end
56
57 for i = 2:dt
58
59     % ACCELERATION - DESIRED Q
60     ddQ1(i) = (dQ1(i) - dQ1(i-1))/ T/dt;
61     ddQ2(i) = (dQ2(i) - dQ2(i-1))/ T/dt;
62
63 end
64
65 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66 %% FEEDBACK
67
68 new_Q1 = zeros(1,dt);
69 new_Q2 = zeros(1,dt);
70 new_dQ1 = zeros(1,dt);
71 new_dQ2 = zeros(1,dt);
72 new_ddQ1 = zeros(1,dt);
73 new_ddQ2 = zeros(1,dt);
74
75 new_Q1(1) = Q1(1);

```

```

77 new_Q2(1) = Q2(1);
78 new_dQ1(1) = dQ1(1);
79 new_dQ2(1) = dQ2(1);
80 new_ddQ1(1) = ddQ1(1);
81 new_ddQ2(1) = ddQ2(1);
82 new_Q1(2) = Q1(2);
83 new_Q2(2) = Q2(2);
84 new_dQ1(2) = dQ1(2);
85 new_dQ2(2) = dQ2(2);
86 new_ddQ1(2) = ddQ1(2);
87 new_ddQ2(2) = ddQ2(2);
88
89
90 tau1 = [];
91 tau2 = [];
92 new_ddQ = [];
93
94 K = 0.01; %[Nm]
95 k = 100; %[s]
96
97 Xfb(1,1) = xd(1,1);
98 Yfb(1,1) = xd(2,1);
99 Xff(1,1) = xd(1,1);
100 Yff(1,1) = xd(2,1);
101
102 for i = 2:dt
103
104     % TAU
105     e1 = Q1(i) - new_Q1(i);
106     e2 = Q2(i) - new_Q2(i);
107     de1 = dQ1(i) - new_dQ1(i);
108     de2 = dQ2(i) - new_dQ2(i);
109
110     tau1 = K*(e1 + k*de1);
111     tau2 = K*(e2 + k*de2);
112     Tau(i,:) = [tau1, tau2];
113
114
115     % NEW ACCELERATION - ddQ
116     alpha = 2*m*l_m^2 + m*l^2 + 2*I;
117     beta = 2*m*l*l_m*cos(new_Q2(i)-new_Q1(i));
118     H = [alpha, beta; beta, alpha];
119     new_H = pinv(H);
120     C = 2*m*l*l_m*sin(new_Q2(i)-new_Q1(i));
121
122     part_of_eq(i,:) = Tau(i,:) - C*[-(new_dQ2(i))^2; (new_dQ1(i))^2]';
123     new_ddQ(i,:) = new_H * part_of_eq(i,:)';
124     new_ddQ1(i) = new_ddQ(i,1);
125     new_ddQ2(i) = new_ddQ(i,2);
126     new_ddQ1(i+1) = new_ddQ(i,1);
127     new_ddQ2(i+1) = new_ddQ(i,2);
128
129
130     % REAL VELOCITY
131     new_dQ1(i+1) = new_dQ1(i) + new_ddQ1(i)*(T/dt);
132     new_dQ2(i+1) = new_dQ2(i) + new_ddQ2(i)*(T/dt);
133
134
135     % REAL POSITION
136     new_Q1(i+1) = new_Q1(i) + new_dQ1(i)*(T/dt);
137     new_Q2(i+1) = new_Q2(i) + new_dQ2(i)*(T/dt);
138
139     % ACTUAL X AND Y
140     Xfb(1,i) = l*cos(new_Q2(1,i))+l*cos(new_Q1(1,i));
141     Yfb(1,i) = l*sin(new_Q2(1,i))+l*sin(new_Q1(1,i));

```

```

142
143 % ROBOT POSITION
144 RIGHTarm_fb(:,i)=l*[cos(new_Q2(1,i));sin(new_Q2(1,i))];
145 LEFTarm_fb(:,i)=l*[cos(new_Q1(1,i));sin(new_Q1(1,i))];
146
147
148
149 end
150
151
152
153 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154 %% FEEDBACK + FEEDFORWARD
155
156 TauFB = Tau';
157
158 new2_Q1 = zeros(1,dt);
159 new2_Q2 = zeros(1,dt);
160 new2_dQ1 = zeros(1,dt);
161 new2_dQ2 = zeros(1,dt);
162 new2_ddQ1 = zeros(1,dt);
163 new2_ddQ2 = zeros(1,dt);
164
165 new2_Q1(1) = Q1(1);
166 new2_Q2(1) = Q2(1);
167 new2_dQ1(1) = dQ1(1);
168 new2_dQ2(1) = dQ2(1);
169 new2_ddQ1(1) = ddQ1(1);
170 new2_ddQ2(1) = ddQ2(1);
171 new2_Q1(2) = Q1(2);
172 new2_Q2(2) = Q2(2);
173 new2_dQ1(2) = dQ1(2);
174 new2_dQ2(2) = dQ2(2);
175 new2_ddQ1(2) = ddQ1(2);
176 new2_ddQ2(2) = ddQ2(2);
177
178 new2_tau1 = [];
179 new2_tau2 = [];
180 new2_TauFB = [];
181
182
183 for i = 2:dt
184
185 % TAU
186 J_dot = l*[-cos(Q1(i))*dQ1(i), -cos(Q2(i))*dQ2(i); -sin(Q1(i))*dQ1(i), -sin(Q2(i))*dQ2(
187 i)];
188 J = l*[-sin(Q1(i)), -sin(Q2(i)); cos(Q1(i)), cos(Q2(i))];
189 J = pinv(J);
190
191 feedback(:,i) = J * ([ddxd(1,i);ddxd(2,i)] - J_dot*[dQ1(1,i); dQ2(1,i)]);
192
193 alpha_2 = 2*m*l_m^2 + m*l^2 + 2*I;
194 beta_2 = 2*m*l*l_m*cos(Q2(i)-Q1(i));
195 H_des = [alpha_2, beta_2; beta_2 alpha_2];
196 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))
197 *(-(dQ2(i))^2);
198 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))
199 *((dQ1(i))^2);
200 TauFF(:,i) = [tauFF1; tauFF2];
201
202 % ERROR
203 new2_e1 = Q1(i) - new2_Q1(i);
204 new2_e2 = Q2(i) - new2_Q2(i);
205 new2_de1 = dQ1(i) - new2_dQ1(i);

```

```

204     new2_de2 = dQ2(i) - new2_dQ2(i);
205
206     new2_tau1 = K*(new2_e1 + k*new2_de1);
207     new2_tau2 = K*(new2_e2 + k*new2_de2);
208     new2_TauFB(:,i) = [new2_tau1, new2_tau2];
209
210
211     TauFinal(:,i) = TauFF(:,i) + new2_TauFB(:,i);
212     TauFinalT = TauFinal';
213
214     % NEW ACCELERATION - ddQ
215     alpha_3 = 2*m*l_m^2 + m*l^2 + 2*I;
216     beta_3 = 2*m*l*m*cos(new2_Q2(i)-new2_Q1(i));
217     H = [alpha_3, beta_3; beta_3 alpha_3];
218     new_H = pinv(H);
219     delta = new_H(1,1);
220     gamma = new_H(1,2);
221     C = 2*m*l*m*sin(new2_Q2(i)-new2_Q1(i));
222
223     new2_Tau(i,:) = TauFinalT(i,:) - C*[-(new2_dQ2(i))^2; (new2_dQ1(i))^2]';
224     new2_ddQ(i,:) = new_H * new2_Tau(i,:)';
225     new2_ddQ1(i) = new2_ddQ(i,1);
226     new2_ddQ2(i) = new2_ddQ(i,2);
227
228
229     % REAL VELOCITY
230     new2_dQ1(i+1) = new2_dQ1(i) + new2_ddQ1(i)*(T/dt);
231     new2_dQ2(i+1) = new2_dQ2(i) + new2_ddQ2(i)*(T/dt);
232
233
234     % REAL POSITION
235     new2_Q1(i+1) = new2_Q1(i) + new2_dQ1(i)*(T/dt);
236     new2_Q2(i+1) = new2_Q2(i) + new2_dQ2(i)*(T/dt);
237
238
239     % ACTUAL X AND Y
240     Xff(1,i) = l*cos(new2_Q2(1,i))+l*cos(new2_Q1(1,i));
241     Yff(1,i) = l*sin(new2_Q2(1,i))+l*sin(new2_Q1(1,i));
242
243
244 end
245
246
247
248
249 %% PLOTS 2B
250
251 t = linspace(0,T,dt);
252
253 new_Q1(dt+1) = [];
254 new_Q2(dt+1) = [];
255 new_dQ1(dt+1) = [];
256 new_dQ2(dt+1) = [];
257 new2_Q1(dt+1) = [];
258 new2_Q2(dt+1) = [];
259 new2_dQ1(dt+1) = [];
260 new2_dQ2(dt+1) = [];
261 xd(:,dt+1) = [];
262
263
264
265 % DESIRED AND ACTUAL ANGLE - AGAINST TIME
266 figure(4)
267 subplot(1,2,1)

```

```

268 plot(t,rad2deg(Q1), 'k—'); hold on; plot(t, rad2deg(new_Q1), 'b'); hold on; plot(t, rad2deg
    (new2_Q1), 'r');
269 title('DESIRED AND ACTUAL QL ANGLE');
270 legend('desired value', 'feedback', 'feedforward + feedback');
271 xlabel('Time [s]');
272 ylabel('Angle [\circ]');
273 subplot(1,2,2)
274 plot(t,rad2deg(Q2), 'k—'); hold on; plot(t, rad2deg(new_Q2), 'b'); hold on; plot(t, rad2deg
    (new2_Q2), 'r');
275 title('DESIRED AND ACTUAL QR ANGLE')
276 legend('desired value', 'feedback', 'feedforward + feedback')
277 xlabel('Time [s]');
278 ylabel('Angle [\circ]')
279
280 % DESIRED AND ACTUAL ENDPOINT POSITIONS X AND Y DIRECTIONS – AGAINST TIME
281 figure(5)
282 subplot(1,2,1)
283 plot(t,xd(1,:), 'k—'); hold on; plot(t, Xfb, 'b'); hold on; plot(t, Xff, 'r');
284 title('DESIRED AND ACTUAL ENDPOINT X POSITION');
285 legend('desired value', 'feedback', 'feedforward + feedback');
286 xlabel('Time [s]');
287 ylabel('Position [m]')
288 subplot(1,2,2)
289 plot(t,xd(2,:), 'k—'); hold on; plot(t, Yfb, 'b'); hold on; plot(t, Yff, 'r');
290 title('DESIRED AND ACTUAL ENDPOINT Y POSITION')
291 legend('desired value', 'feedback', 'feedforward + feedback')
292 xlabel('Time [s]');
293 ylabel('Position [m]')
294
295
296 % DESIRED AND ACTUAL ENDPOINT TRAJECTORIES – IN THE X-Y PLANE
297 figure(6)
298 plot(xd(1,:),xd(2,:), 'k—'); hold on; plot(Xfb, Yfb, 'b'); hold on; plot(Xff, Yff, 'r');
299 title('DESIRED AND ACTUAL ENDPOINT TRAJECTORIES');
300 legend('desired value', 'feedback', 'feedforward + feedback');
301 xlabel('x axis [m]');
302 ylabel('y axis [m]')

```

3.2.2 Desired and actual angles against time

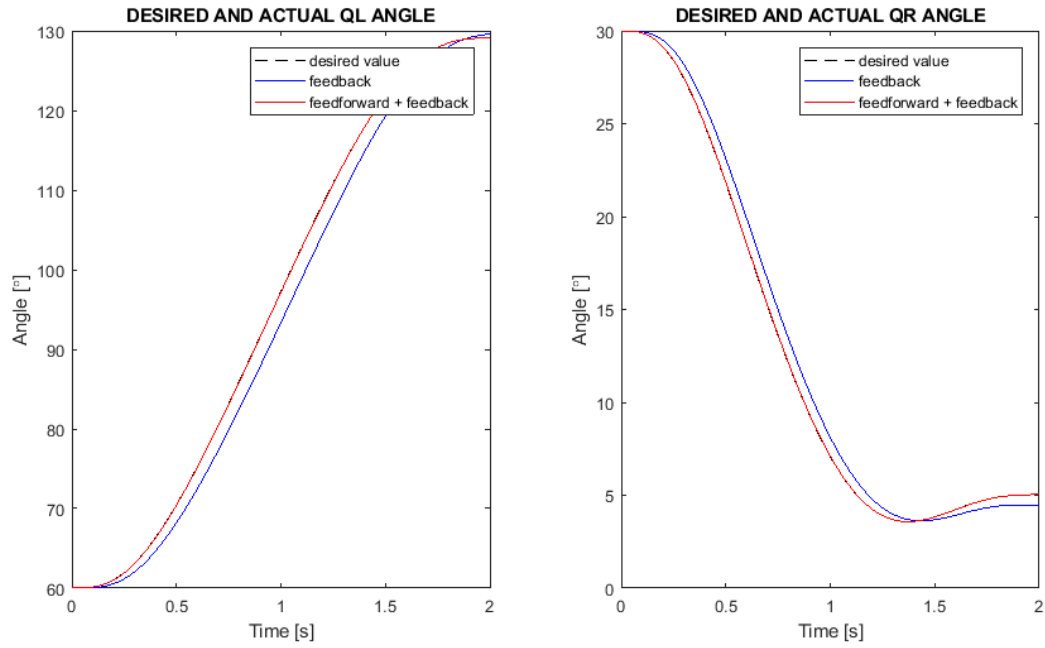


Figure 6: Plot of the desired and actual angle against time

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

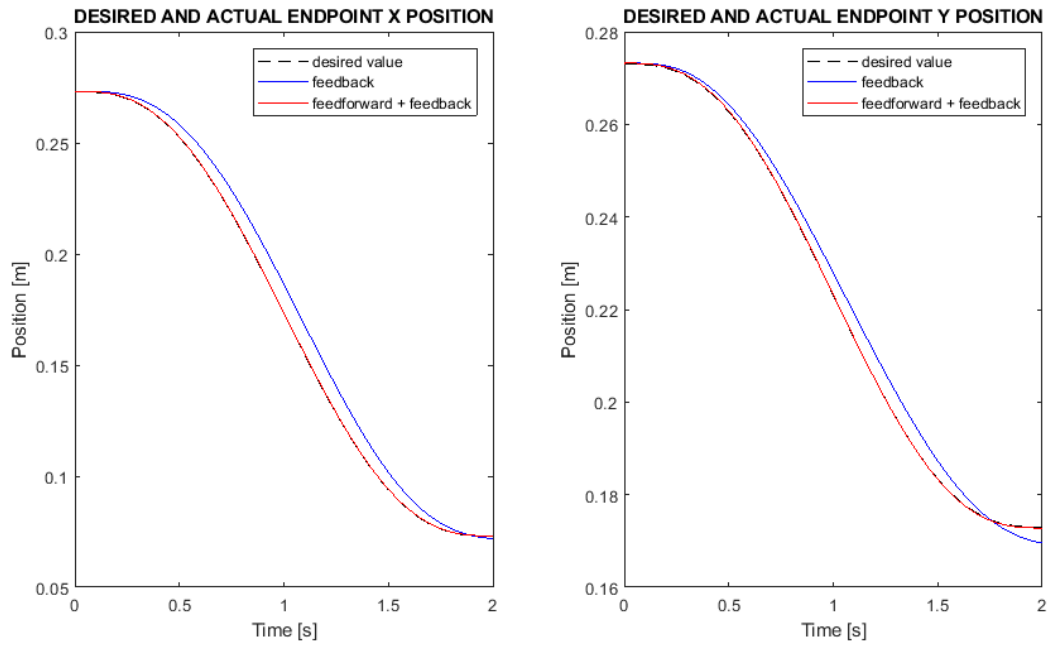


Figure 7: Plot of the desired and actual endpoint positions x and y directions against time

3.2.4 Desired and actual endpoint trajectories in x-y plane

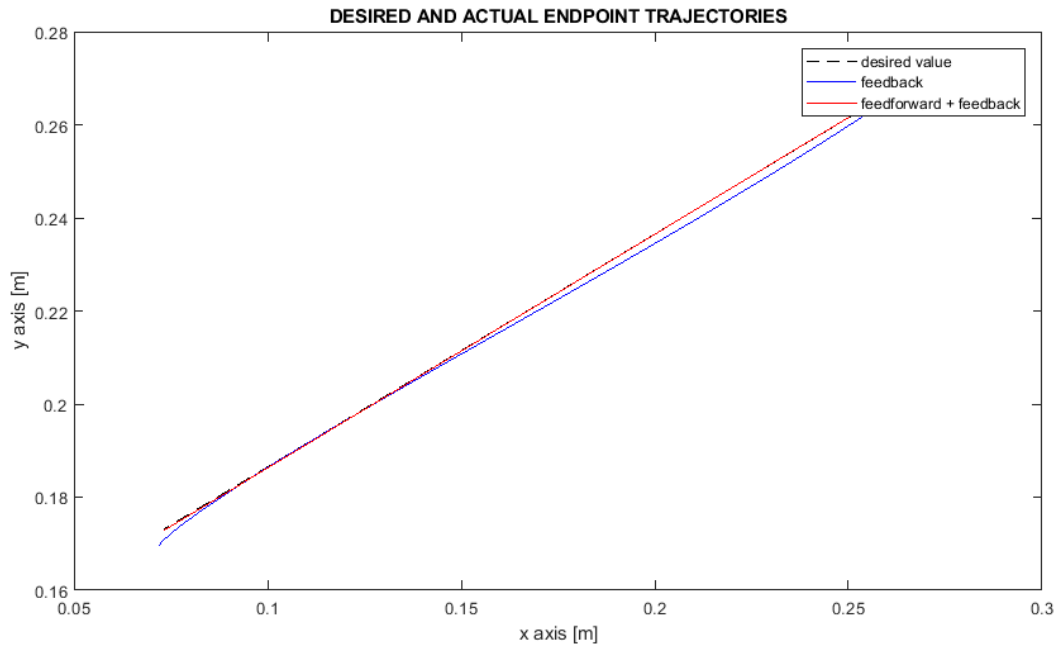


Figure 8: Plot of the desired and actual endpoint trajectories in the x-y plane

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.