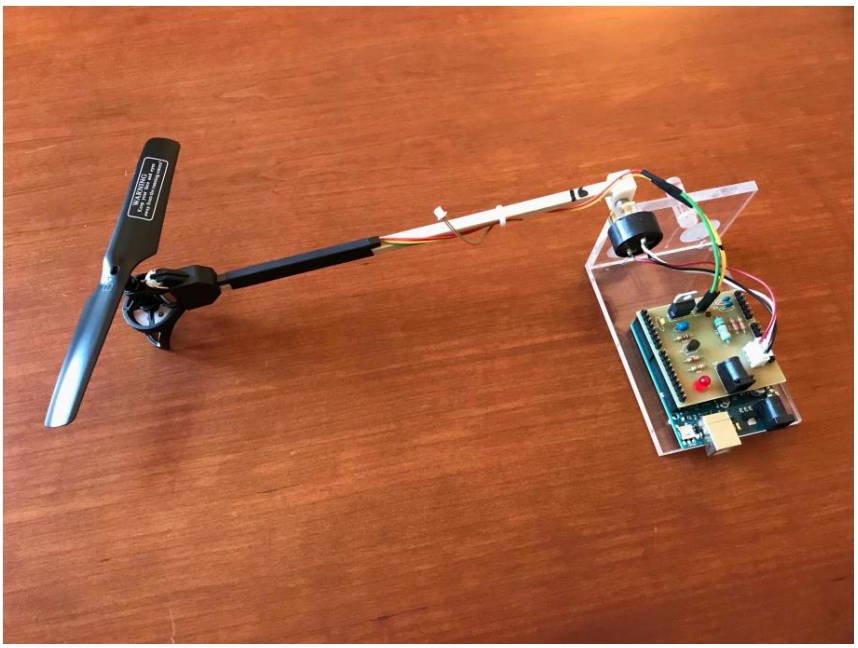
KOC UNIVERSITY

SPRING 2021

ELEC 304 – FEEDBACK CONTROL SYSTEMS LAB

FINAL PROJECT REPORT:

ARM ANGLE POSITIONING: ADAPTIVE PID CONTROLLER DESIGN



Instructor: Prof. Dr. Alper Demir
Student Name: Deniz Karadayı
ID: 59925

INTRODUCTION

Throughout the course, I will be investigating the arm angle positioning system with PID controller. At the end of my observations and experiment, I dedicated that this system cannot be perfectly controlled by a simple PID controller since it is not linear. Therefore, as a final task for this course, I have designed a such controller which can work better for this specific system. My design satisfies the minimum requirement of the project and does a little bit more.

THEORY

Reminder for the theory behind the system:

General driving non-linear formula of the system.

$$\frac{d^2\vartheta_a}{dt^2} = \ a.m_d - g.\cos(\vartheta_a) - \beta\frac{d\vartheta_a}{dt} \tag{5}$$

Closed Loop Difference equation with Friction in Z-Transform

$$G(z) = \frac{\vartheta_a(z)}{m_d(z)} = \ \frac{a.T_s^2.z^{-2}}{1 + (\beta.T_s - 2)z^{-1} + (1 - b.T_s^2 - \beta.T_s)z^{-2}} \tag{7}$$

Steady-State Tracking Error (SSTE)

Similar with the previous labs, I will find expressions for the desired angle ϑ_d and the steady-state angle ϑ_a^s in terms of other parameters for a case where SSTE = 0. This means that, as steps goes to infinity, different steps of the output angle will be equal to each other and will be eliminated in a manner. In the following equation, the colored terms will make each other zero, then ends up as the following one.

$$SSTE = \vartheta_a^{ss} - \vartheta_d = 0 \tag{8}$$

Transfer Functions of the PID Controller in Z-Domain

In the equation, $P = K_p$, $D = K_d/K_p$ and $I = K_i/K_p$.

non-filtered: $$C(z) = \frac{m_d(z)}{e(z)} = \ P\left(1 + I.T_s.\frac{1}{z-1} + \ D.\frac{1-z^{-1}}{T_s}\right) \tag{9.1}$$

filtered: $$C(z) = \frac{m_d(z)}{e(z)} = \ P\left(1 + I.T_s.\frac{1}{z-1} + \ D.\frac{N}{1 + N.T_s.\frac{1}{z-1}}\right) \tag{9.2}$$

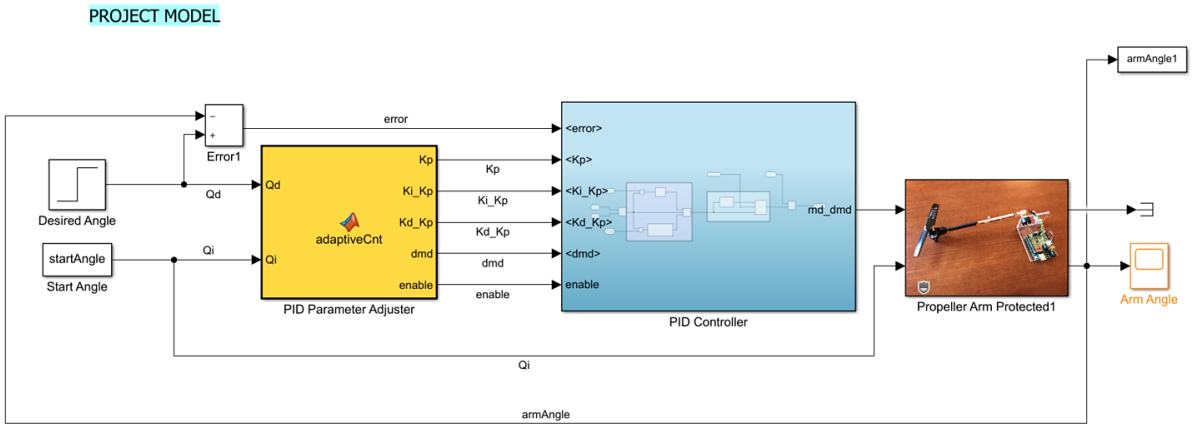
OBJECTIVE

Minimum requirements for the project were the followings.

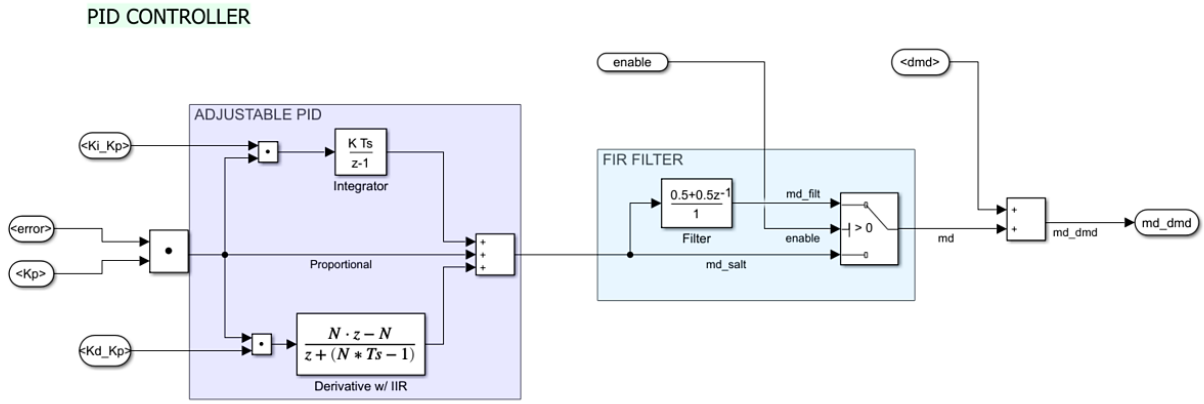
- Work well for 0 to 45 degrees step input, meaning that good transient response (low settling time and minimum overshoot) and steady-state response (low SSTE).
- Work well for 45 to 0 degrees step input, meaning that good transient response (low settling time and minimum overshoot) and steady-state response (low SSTE).
- Reaching to the most possible highest angle starting from 0 degree without collapsing the other side. Next, keeping its position for 300 s.

The project was supposed to handle the given requirement to be successful, and my design is perfectly doing its job for those cases.

DESIGN



The figure above shows my controller design for the project in general. It consists of a MATLAB function block, which is basically a function script returning PID controller parameters according to the start angle and desired angle inputs. This function block feeds the PID controller with its parameters. The next blue block is my PID controller fed by the MATLAB function block. The next figure shows the inside of this controller.



The first part of this controller is directly a PID controller whose parameters can be manipulated. The second part, there is a filter to eliminate noise before feeding the actual system. Since the system itself is noisy, its feedback is noisy as well. Therefore, PID controller send a much noisier feedback to the system as well. With this filter, I try to eliminate it a little bit. However, this is not always desiring for the system, so I control it from MATLAB function with an enable bit. Finally, I add the dmd to the final controller output which similarly controlled by MATLAB function as well.

The following script is my MATLAB function to adjust controller.

```
function [Kp, Ki_Kp, Kd_Kp, dmd, enable] = adaptiveCnt(Qd,Qi)
    % GENERAL CONTROLLER PARAMETERS
    Kp = 2;
    Kd_Kp = 0.2;
    Ki_Kp = 1.5;
    dmd = 0;
```

```
    enable = 1; % -enables overall filter for final controller output before feeding the system

    if(Qi==0)               % // All cases where Start Angle (Qi) is 0.
        if Qd >= Qi             % // For Positive Desired Angle (Qd)
            if(Qd >= 70)
                Ki_Kp = 0.00017551619222566l*Qd^2 - 0.046290072190463*Qd + 2.74258543457608;
            elseif (Qd >= 45)
                Ki_Kp = 0.00017551619222566l*Qd^2 - 0.046290072190463*Qd + 2.627632959313750;
            elseif (Qd >= 30)
                Ki_Kp = 0.00017551619222566l*Qd^2 - 0.046290072190463*Qd + 2.74258543457608;
                dmd = 8.921011874031939e-07*Qd^5 - 3.685173886516096e-05*Qd^4 -
0.001518186511475*Qd^3 + 0.068177750660973*Qd^2 - 1.184019962140772*Qd + 40.090697892711330;
            elseif (Qd < 30)
                dmd = 8.921011874031939e-07*Qd^5 - 3.685173886516096e-05*Qd^4 -
0.001518186511475*Qd^3 + 0.068177750660973*Qd^2 - 1.184019962140772*Qd + 40.090697892711330;
            end
        else                    % // For Negative Desired Angle (Qd)
            if(Qd>=-20)
               dmd = -2*Qd + 40;
            else
                dmd = 80;
            end
        end
    elseif Qi>0 && Qd==0    % // All cases where Qi is positive and Qd is 0.

        if Qi<50
            dmd = 7.195027195027213e-04*Qi^3 - 0.027668997668998*Qi^2 + 1.274203574203577*Qi +
40.418318158520410;
        elseif Qi<70
            dmd = 124.89;
        else
            dmd = 0.032252207125414*Qi^2 - 2.325646484252623*Qi + 1.307144824995270e+02;
            Ki_Kp = -1.264792436290815e-04*Qi^2 - 6.837392774397380e-04*Qi + 2.165825558825345;
        end

    elseif Qi == -90        % // All cases where Qi is -90.
        if Qd <= -70
            enable = 0;
            Ki_Kp = 3;
            Kp = 0.6;
        elseif Qd <= -40
            enable = 0;
            Ki_Kp = 5.925925925926097e-05*Qd^3 + 0.008190476190476*Qd^2 + 0.328518518518530*Qd +
6.022222222222408;
            Kp = 5.555555555555661e-05*Qd^3 + 0.007642857142857*Qd^2 + 0.380396825396832*Qd +
8.533333333333440;
        elseif Qd <= 30
            Ki_Kp = -6.313131313131334e-06*Qd^3 + 8.387445887445803e-05*Qd^2 -
0.012848124098124*Qd + 1.126623376623377;
            Kp = 2;
        else
            Ki_Kp = 2.727236199854714e-07*Qd^4 - 6.619203811799032e-05*Qd^3 +
0.005640287022121*Qd^2 - 0.207466259546331*Qd + 3.363611519226011;
            Kp = -5.798900495588756e-07*Qd^4 + 1.491761329295650e-04*Qd^3 - 0.013744414002193*Qd^2
+ 0.510560174649890*Qd - 4.502360006537398;
        end

    elseif (Qd<Qi)          % // All the rest cases where Qd<Qi.
        dQ = Qi-Qd;
        if Qd>80 && dQ<6
            dmd =0;
```

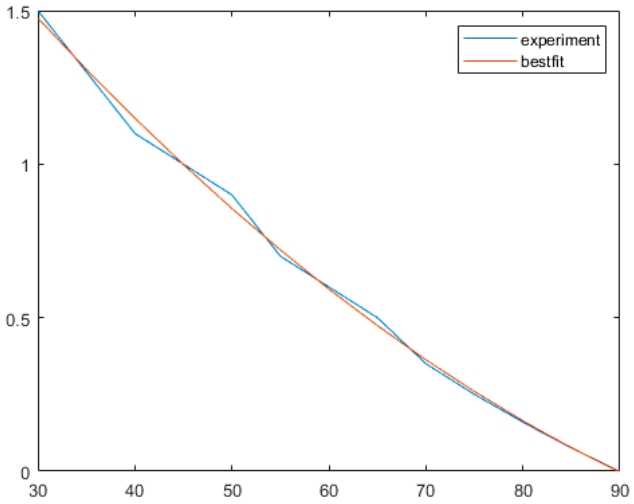
```
        elseif Qd>80 && dQ>6
            dmd = dQ-6;
        elseif Qd>70 && dQ<=10
            dmd = dQ + 15;
        elseif Qd>70 && dQ>10
            dmd = 1.5*dQ + 10;
        elseif Qd>60 && dQ<=10
            dmd = dQ+20;
        elseif Qd>60 && dQ>10
            dmd = dQ + 20;
        elseif Qd>50 && dQ<=10
            dmd = 1.1*dQ+25;
        elseif Qd>50 && dQ>10
            dmd = 0.8*dQ + 28;
        elseif Qd>40 && dQ<=10
            dmd = 1.5*dQ + 25;
        elseif Qd>40 && dQ>10
            dmd = 1.375*dQ + 21.25;
        elseif Qd>30 && dQ<=10
            dmd = 0.9*dQ + 35;
        elseif Qd>30 && dQ>10
            dmd = 1.12*dQ + 22.8;
        elseif Qd>20 && dQ<=10
            dmd = 1.2*dQ + 35;
        elseif Qd>10 && dQ>10
            dmd = dQ + 40;
        elseif Qd>10 && dQ>10
            dmd = 1.71*dQ + 32.86;
        elseif Qd>0 && dQ>10
            dmd = dQ + 40;
        elseif Qd>0 && dQ>10
            dmd = 1.625*dQ + 33.75;
        end

    elseif Qi>45 && Qd>Qi
        Kp = 1;
        Ki_Kp = 0.3;
    elseif Qi>20 && Qd>Qi
        Kp = 1.4;
        Ki_Kp = 0.6;
    elseif Qd>=70
        Kp = 1;
        Ki_Kp = 0.1;
    end

end
```

This script consists of Ki/Kp, Kp and dmd functions of start angle and desired angle. I tried to come up with first, second, third, fourth and fifth order functions of Qi and Qd for different intervals. I have run some test to find optimal value combinations of controller parameters for varying Qd and Qi values. My approach is based on my deductions from previous labs and further experiments. I collected data and use pollyfit() function of MATLAB to construct multiple continuous functions. First, I specify a general PID controller with parameters Ki/Kp = 1.5, Kd/Kp=0.2, Kp=2, dmd=0 and p=0.9 with FIR filter at the end. I initially assigned these values, then if it is necessary, I changed them with continuous functions according to the intervals of Qd and Qi. With this way, my controller does a very good job in general, especially for the project requirements.

The following scripts are short part of my function calculations written in the previous MATLAB function.

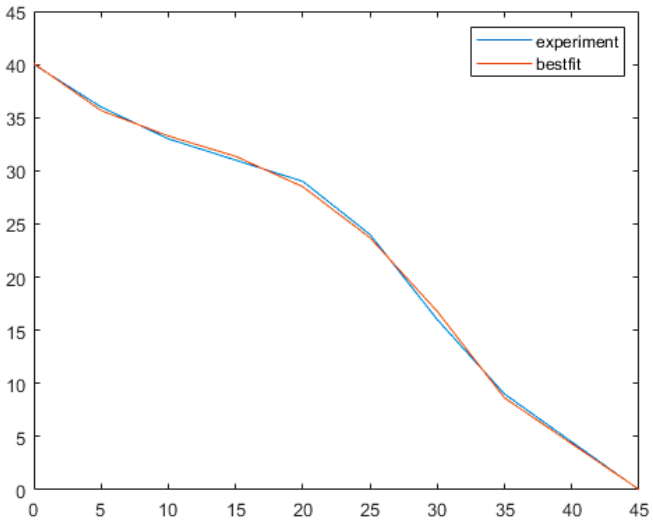
Best Ki/Kp values for Qi = 0, and varying Qd.

```
desAng = [30 35 40 45 50 55 60 65 70 75 80 82.5 85 86 87 88 89 89.5 89.7 89.8 89.83 ];
Ki_Kp_val = [1.5 1.3 1.1 1 0.9 0.7 0.6 0.5 0.35 0.25 0.16 0.117 0.076 0.06 0.045 0.03 0.0145 0.007 0.0034 0.001
0.000001];
pol = polyfit(desAng,Ki_Kp_val,2);
bestFit = pol(1)*desAng.^2 + pol(2)*desAng + pol(3);
figure
plot (desAng,Ki_Kp_val)
hold on
plot (desAng,bestFit)
legend('experiment','bestfit');
```



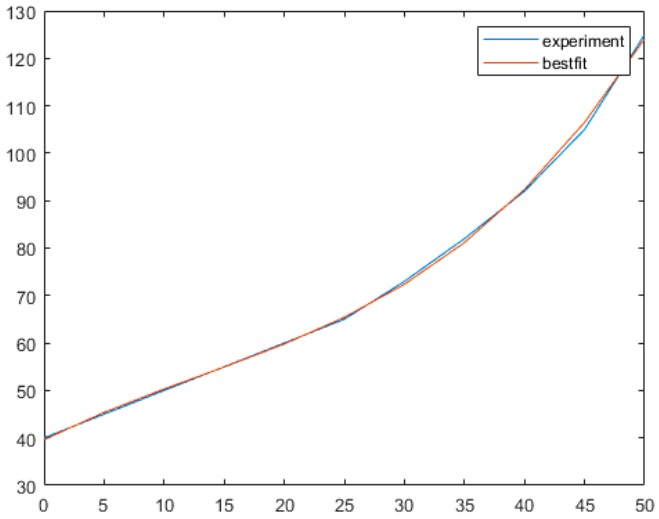
Best dmd values for Qi = 0, and varying Qd.

```
desAng = [45 35 30 25 20 15 10 5 0];
dmd = [0 9 16 24 29 31 33 36 40];
pol = polyfit(desAng,dmd,5);
bestFit =  pol(1)*desAng.^5 + pol(2)*desAng.^4 + pol(3)*desAng.^3 + pol(4)*desAng.^2 + pol(5)*desAng + pol(6);
figure
plot (desAng,dmd)
hold on
plot (desAng,bestFit)
legend('experiment','bestfit');
```



Best dmd values for Qd = 0, and varying Qi.

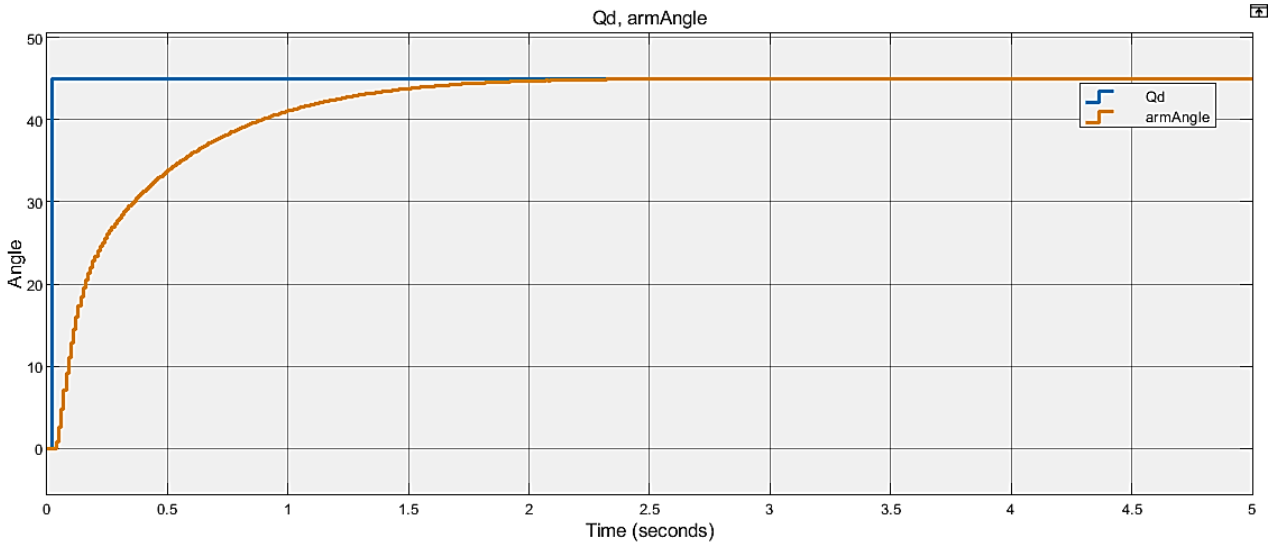
```
startAng = [0 5 10 15 20 25 30 35 40 45 50];
dmd = [40 45 50 55 60 65 73 82 92 105 125];
pol = polyfit(startAng,dmd,3);
bestFit = pol(1)*startAng.^3 + pol(2)*startAng.^2 + pol(3)*startAng + pol(4);
figure
plot (startAng,dmd)
hold on
plot (startAng,bestFit)
legend('experiment','bestfit');
```



ANALYSIS

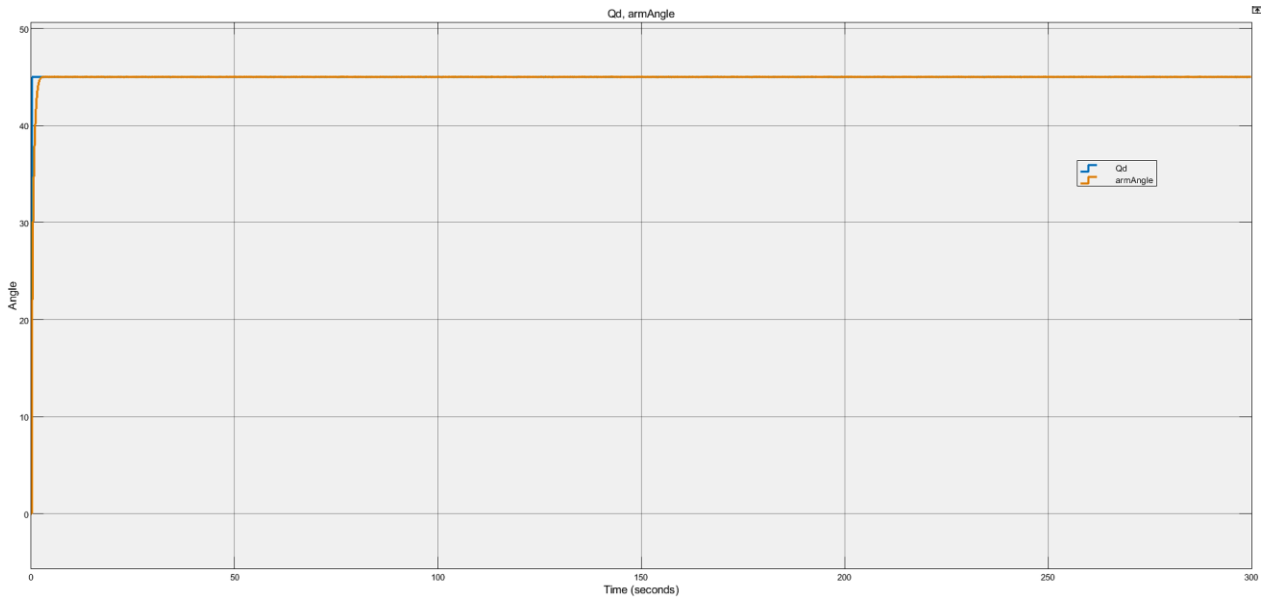
- 0 to 45 Step Input and Response

  In this test, there is no overshoot, and settling time is 2.4 secs.



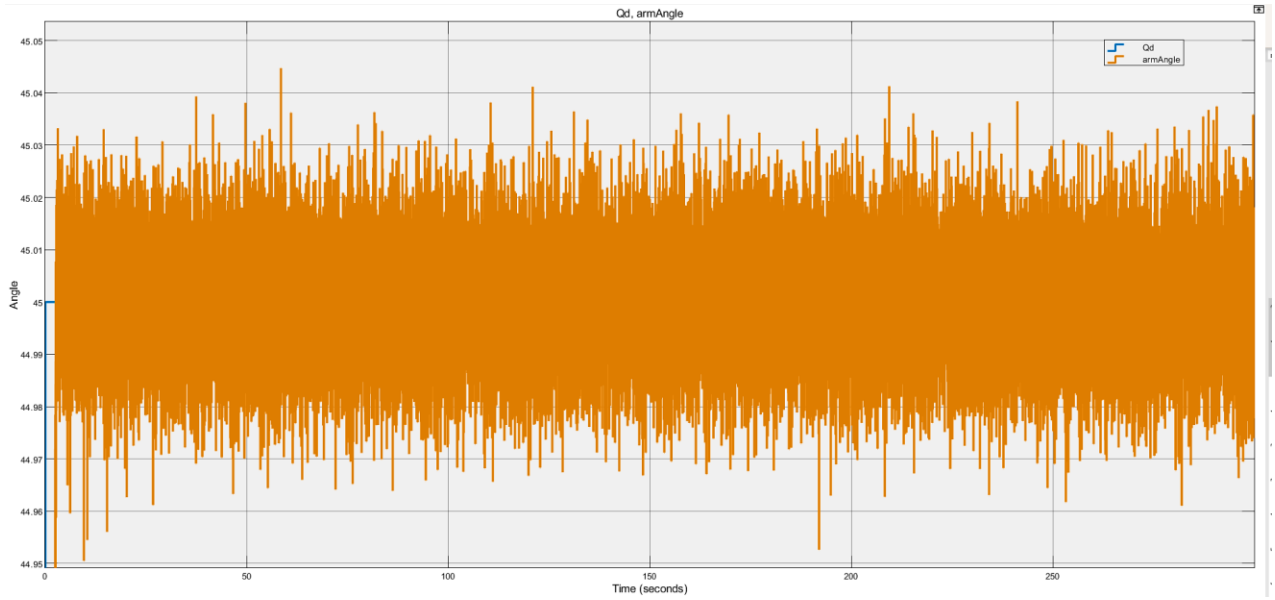
- 0 to 45 Step Input and Response for 300 secs

There is no clear steady-state error except the noise of the system. The system is stable for long trials.



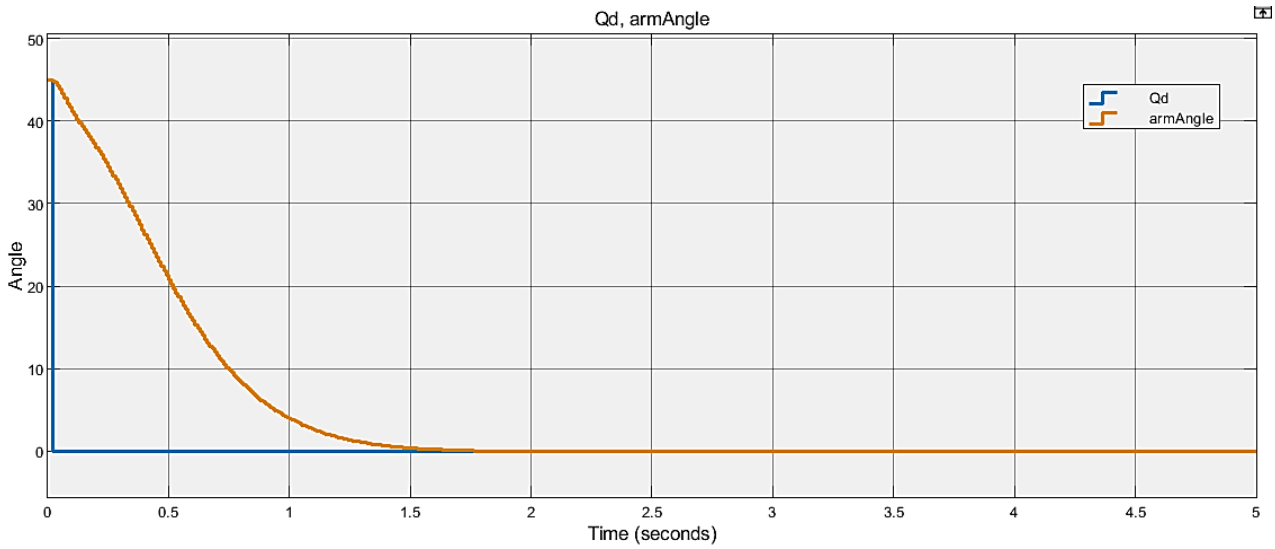
- 0 to 45 Step Input and SSTE of Response

Here, we can see the noise of the system, the greatest error is 0.05 degrees which is an extreme case for even noise. The average noise is ±0.25. However, when we take average of the response there is no SSTE overall.



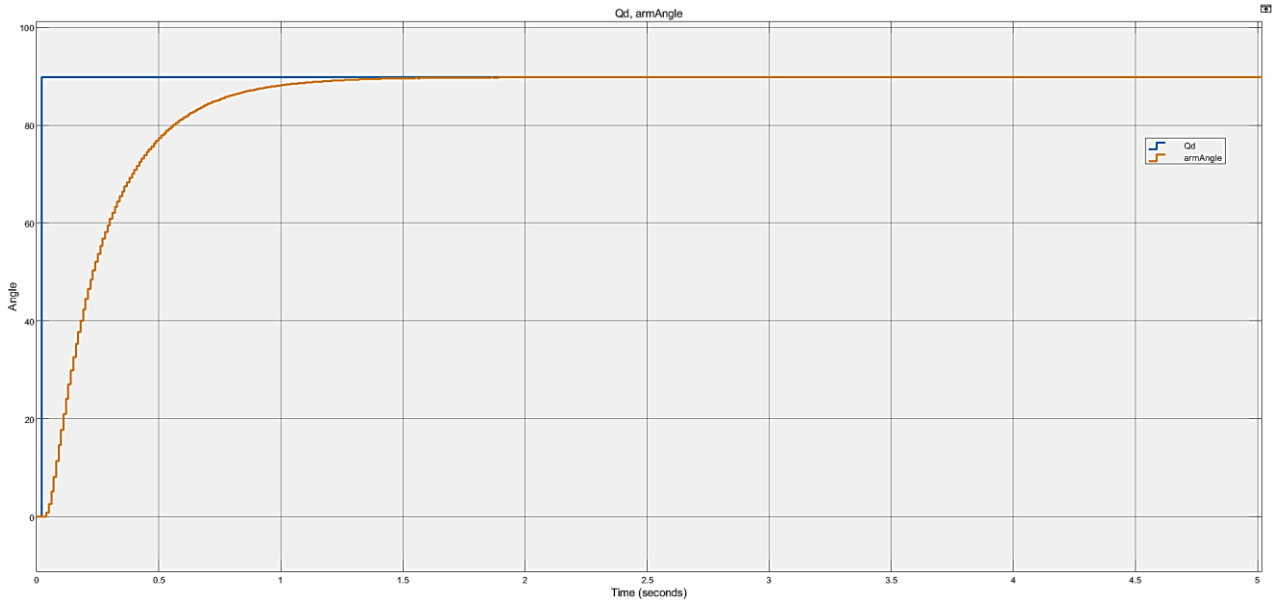
- 45 to 0 Step Input and Response

  In this test, there is no overshoot, and settling time is 1.8 secs.



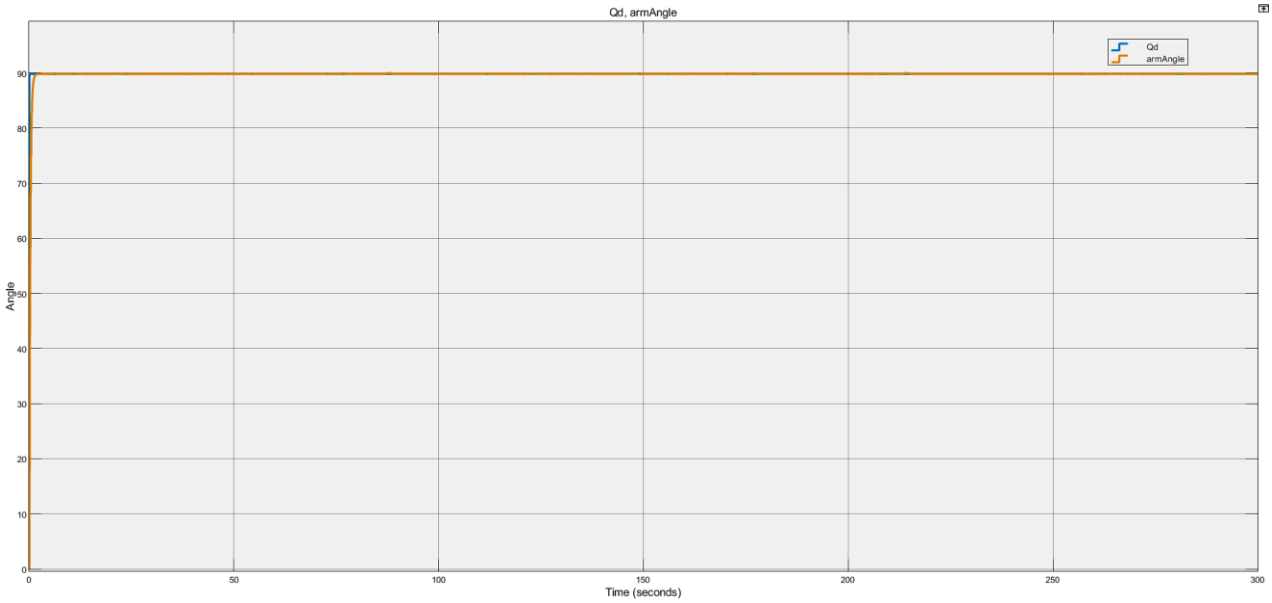
- 0 to 89.815 Step Input and Response

The highest angle that the arm can be stabilized at from 0 is 89.815 degree for my design. There is no overshoot and cannot be in fact since it is not tolerable at that angle.



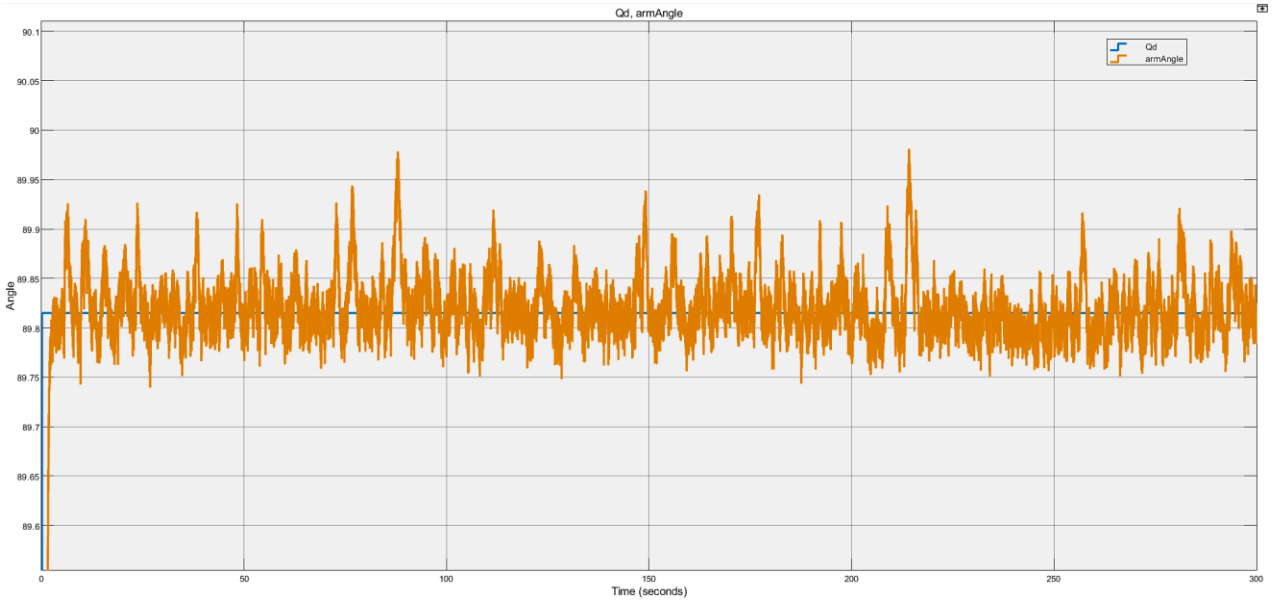
- 0 to 89.815 Step Input and Steady-State Analysis

  The system is stabilized for a long time without collapsing the other side.



- 0 to 89.815 Step Input and Noise of Response

At this angle, it I very hard keep arm at there since a little noise can collapse the system and gravity is no longer be helping in stabilizing the system. The average noise is approximately 0.35 but there is sudden jumps reaching up to 0.16 so that the arm angle becomes 89.96. However, the system is table anyway with this step input around 89.82.



Additional Working Ranges

My controller also works well for the step input given below.

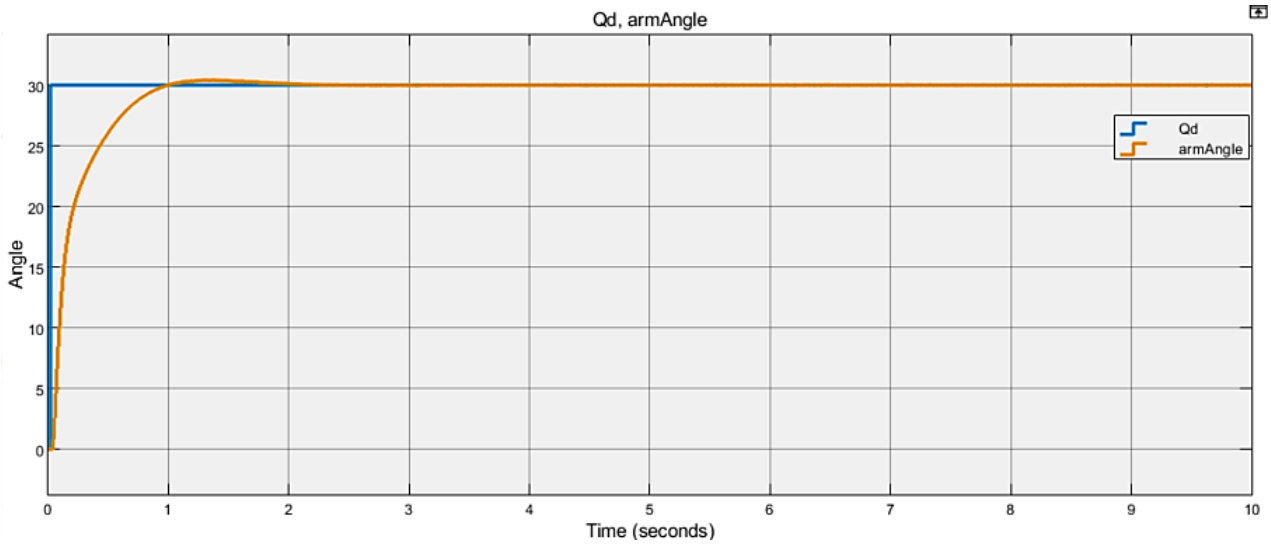
- Starting from 0 and going up.
- Starting from 0 and going down.
- Starting from positive angle and settles at zero.
- Starting from any angle and goes down (for all desired angles below start angle).
- Starting from -90 and goes up 80 degrees.
- Starting from negative angle and goes 0.

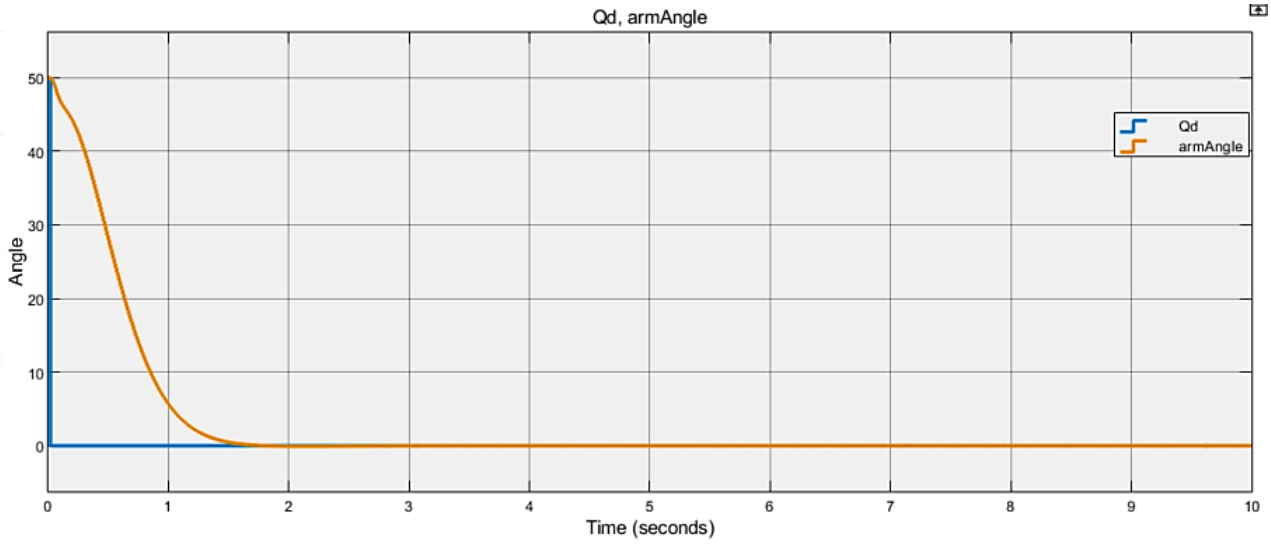
- Starting from negative angle and goes up to another negative angle.

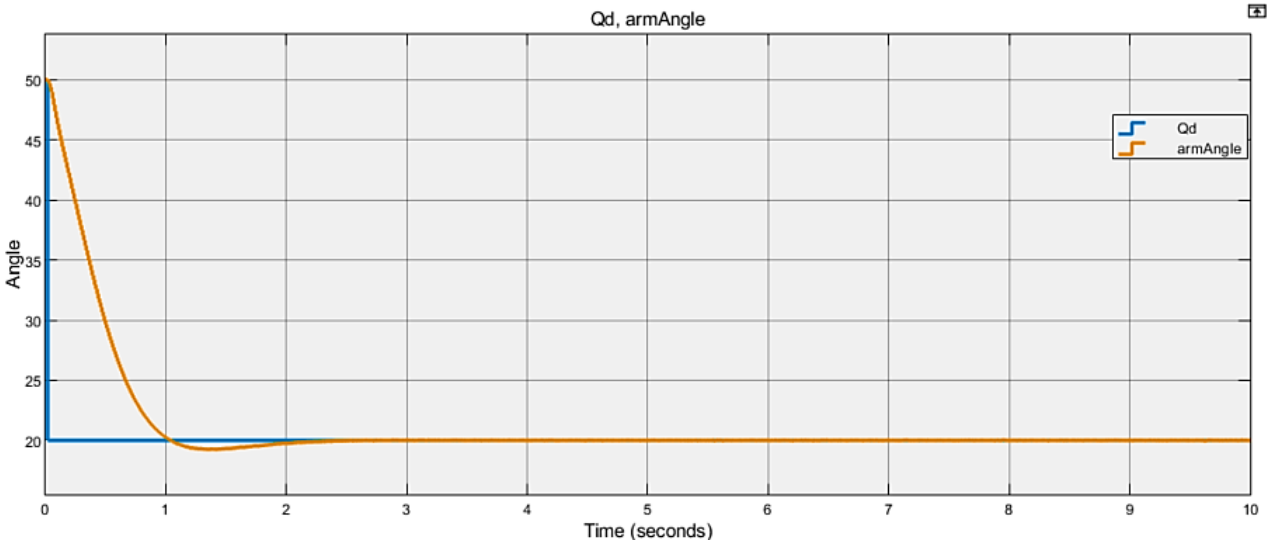
My controller also works pretty okay for the step input given below.

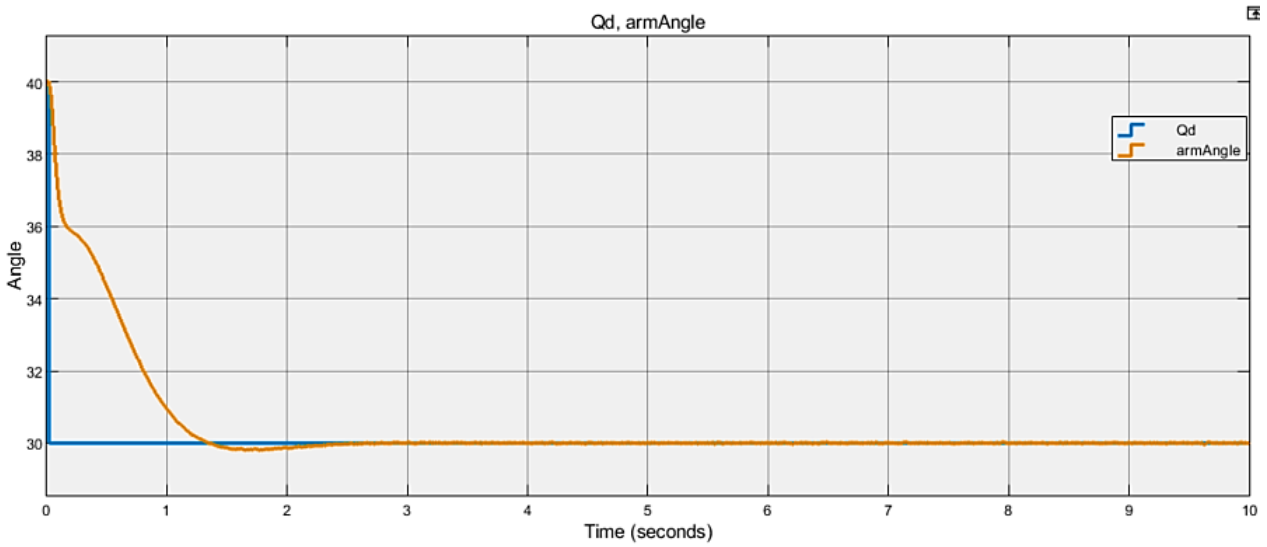
- Starting from positive angle greater than 0 and goes up.
- Starting from negative angle greater than -90 and goes large positive angles.

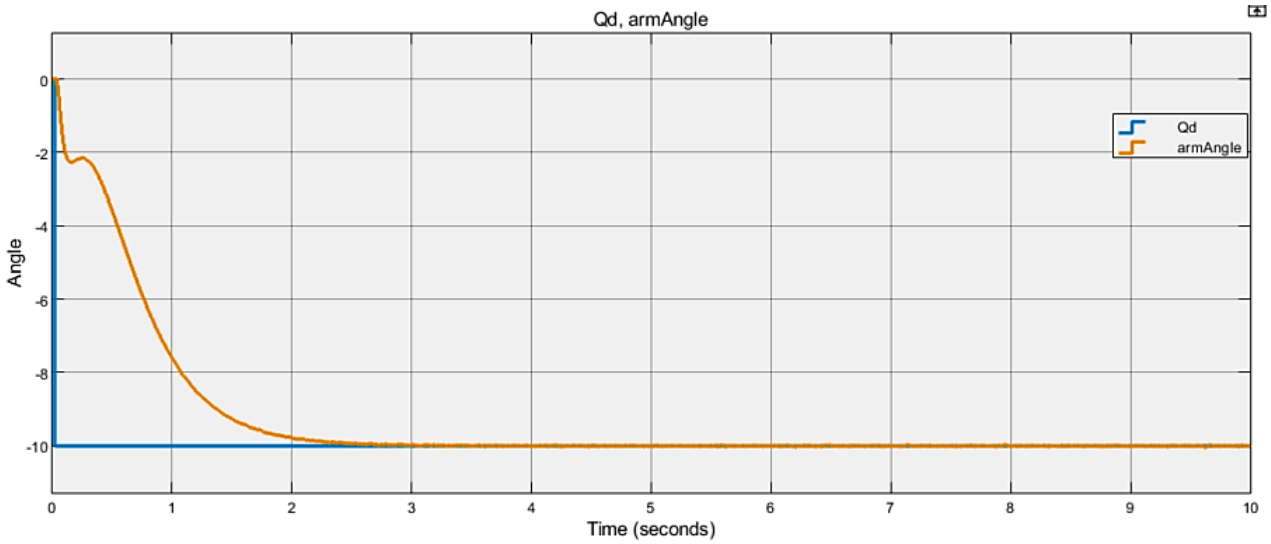
Sample Trials of the System:

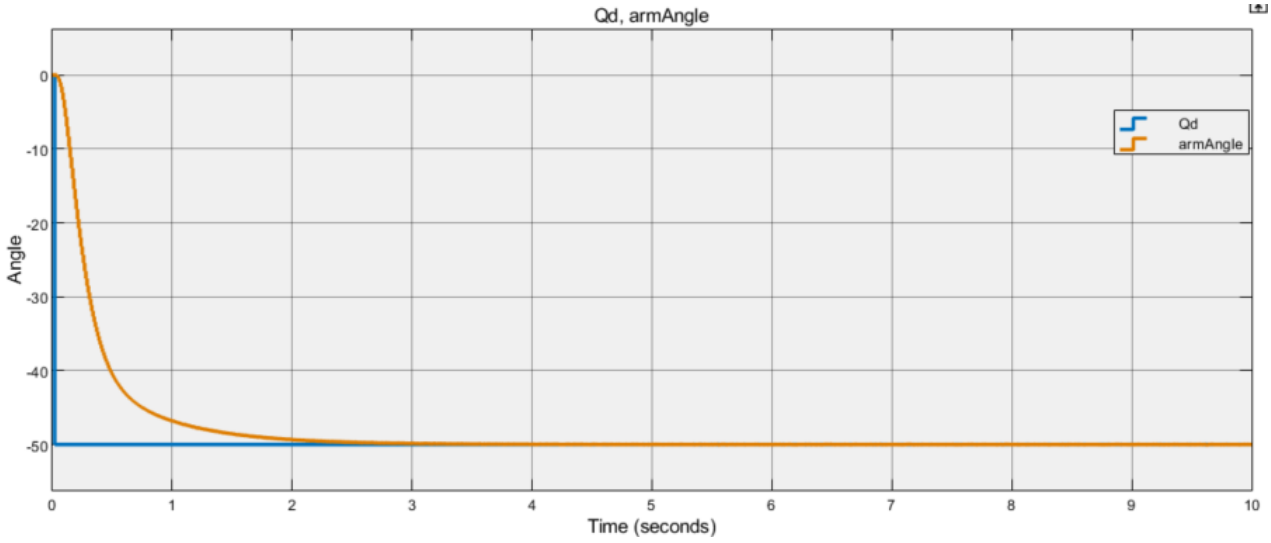


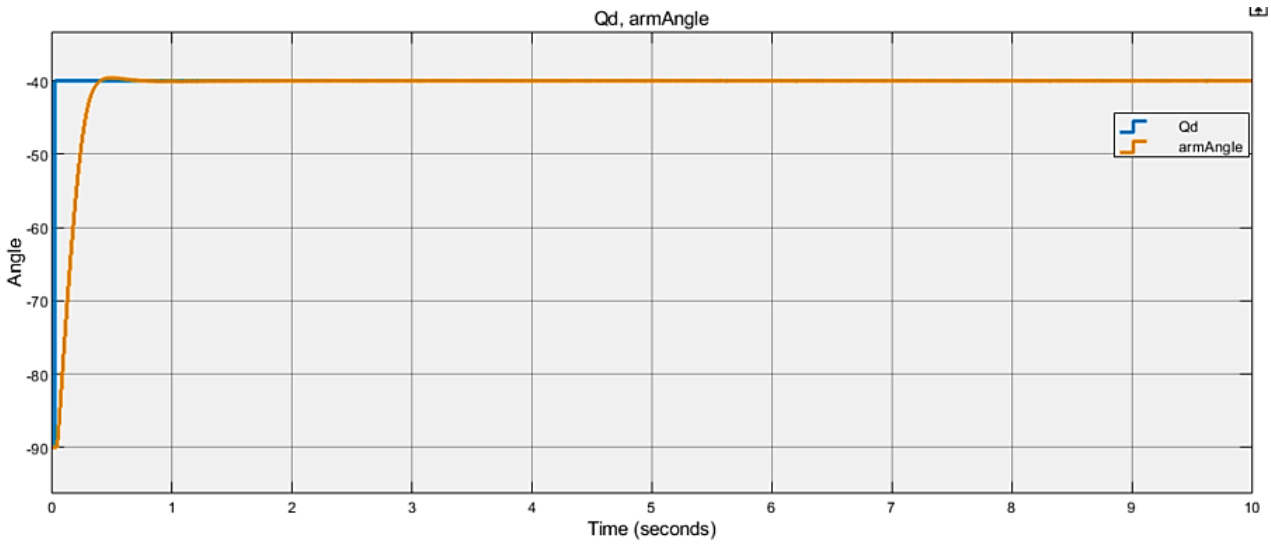


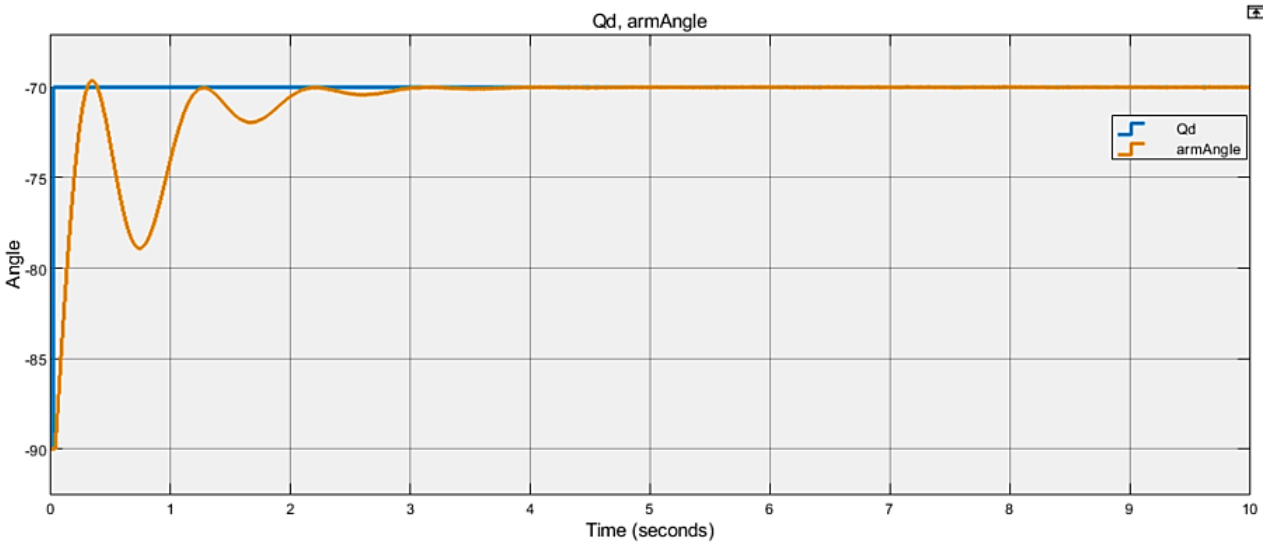


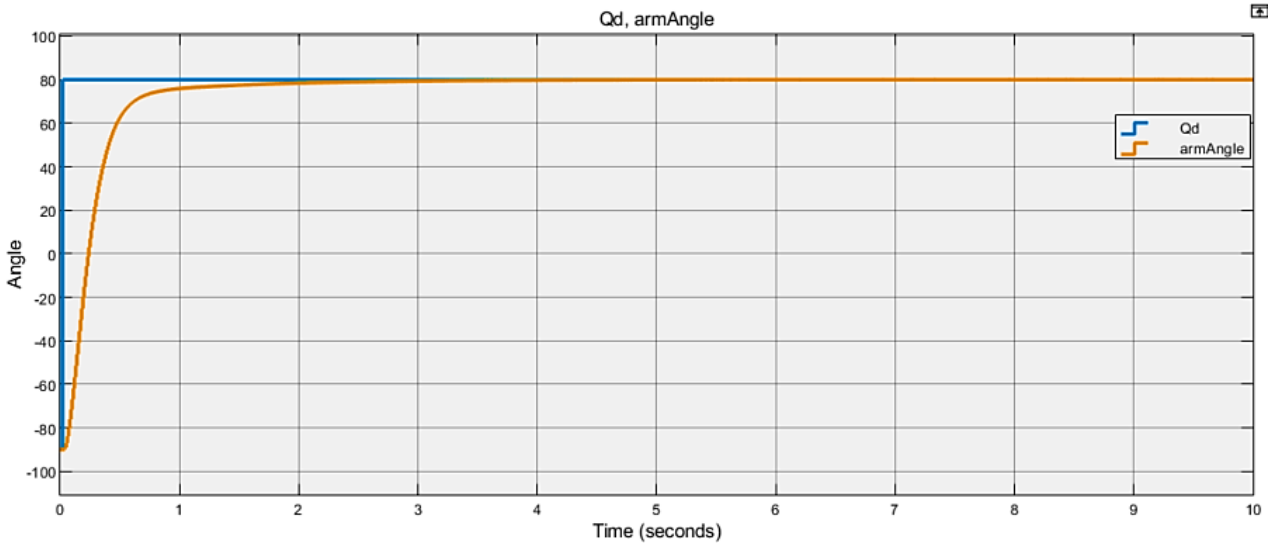

Qd, armAngle
Angle
Time (seconds)
Qd
armAngle

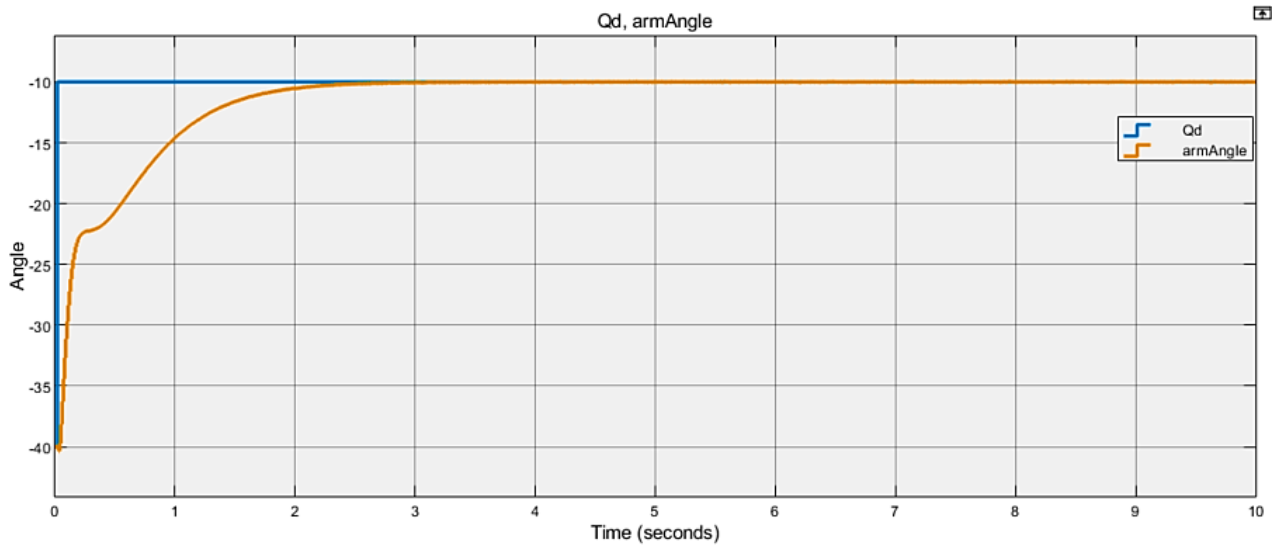

Qd, armAngle
Angle
Time (seconds)
Qd
armAngle


Qd, armAngle
Angle
Time (seconds)
Qd
armAngle


Qd, armAngle
Qd
armAngle
Angle
-40
-50
-60
-70
-80
-90
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)


Qd, armAngle
Qd
armAngle
Angle
-70
-75
-80
-85
-90
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)


Qd, armAngle
Qd
armAngle
Angle
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)



Script to Run and Test the System

The following script is used to run the system automatically. Qd_fin is Desired Angle, and Qd_init is Start Angle of the system. They should be defined by the user to observe response. Independent parameters should in the workspace for Simulink to operate and should not be changed in value.

```
clear all
close all
% independent parameters
deltaT = 0.01;
Ts = deltaT;
p = 0.9;
N = p/Ts;

% user defined parameters
Qd_init = 00;
startAngle = Qd_init;
Qd_fin = 45;
%Qd_fin = 89.815; % MAX value form 0 to go

% Simulink Run
sim('final_project.slx');
```

CONCLUSION

In this course, I have been working on this system from the beginning. I believe that I had a quite good knowledge about PID controller. Since this system is non-linear, I need to think complicated to design an optimal controller by using my acquired knowledge on this course. In the system, I still use a PID control, but its parameters can be inherently changed according to inputs without any interruptions of the designer once the run begins. It is not still perfect but is goof to use for most of the cases.

REFERENCES

- ELEC 304 Lecture Notes, Prof. Dr. Alper Demir, Koç University, May 2021
- ELEC 304 Project Handouts, Prof. Dr. Alper Demir, Koç University, June 2021
- ELEC 304 Laboratory 5 Handouts, Prof. Dr. Alper Demir, Koç University, May 2021
- ELEC 304 Laboratory 5 Preliminary Report, Deniz Karadayı, Koç University, May 2021