EEE342 Feedback and Control Systems Lab 1

Mehmet Berk Şahin

Electrical and Electronics Engineering Department, Bilkent University, 06800 Ankara, Turkey

1. Introduction

In this lab work, by using the knowledge obtained in the preliminary work and by using the quantized step response of the open velocity loop with noise, PI (Proportional-Integral) controller was designed. This design was done by trial-and-error method. Lab consists of three part. In the first part, first order approximated data was used to identify the DC motor. In the second part, three different PI controllers were designed by using first order transfer function as described in Lab#1 Assignment. In the last part, the responses of the closed loops with 3 PI controllers were implemented on nonlinear model of the DC motor.

2. Laboratory Content

Part I

In this part, the response of complex nonlinear model and its first order approximation were plotted and compared. To achieve this, first, nonlinear model was downloaded from moodle and parameters (name, student ID and section) were entered. The response of the DC motor is below:

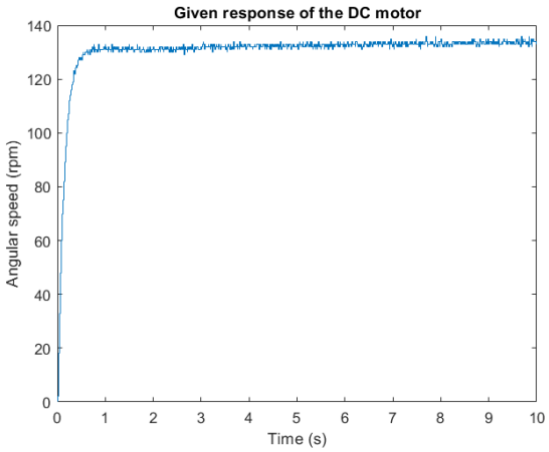

Fig. 1: Response of DC motor

The response is saved automatically to workspace. Then, by using the same approach done in preliminary work, the data taken from DC motor were used to acquire first order approximation. To achieve first order approximation, the transfer function which was derived in preliminary work was used. It is

$$G_p(s) = \frac{K}{\tau s + 1} \qquad (Eq.1)$$

where G_p is process function, τ is time constant and K is constant found in preliminary work. To find two unknowns, which are K and τ, two equations were needed. One of them is coming from first order approximated output function, which was derived by the input signal 9u(t) and process function given in equation 1. It is the following:

$$y(t) = 9K\left(1 - e^{-\frac{t}{\tau}}\right) \qquad (Eq.2)$$

so, K can be found by sending t to infinity. The second unknown, τ, can be found from any random value of t in response's steady state region. The latter calculation was done by calculator, but the former was done by MATLAB. The systems designed in Simulink in order to get two different data are given below:

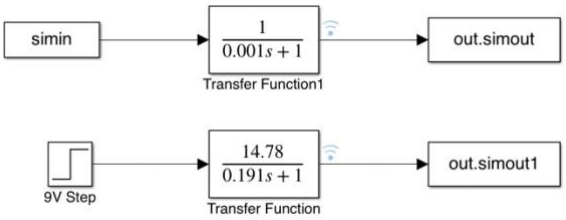

Fig. 2: Open-Loop Systems to Acquire Approximated and Filtered Data

Necessary calculations, which are mentioned above, taking the data from Simulink to workspace and plotting the results were done in the following MATLAB code.

MATLAB Code:

```
simin = DCmotor.out ;
figure();
plot(out.simout, 'green')
xlabel("Time (s)");
ylabel("Angular speed (rpm)");
title("Filtered Signal and Approximated Signal");
hold on
data = out.simout.Data';
data2 = out.simout.Data;
datass = [data(20001:100001)]; % steady-state values
gain = round(mean(datass));    % Gain = 9K
plot(out.simout1,'black');
legend("Filtered Signal", "First Order Approximated
Transfer Function");
hold off
```

Following graph is obtained:

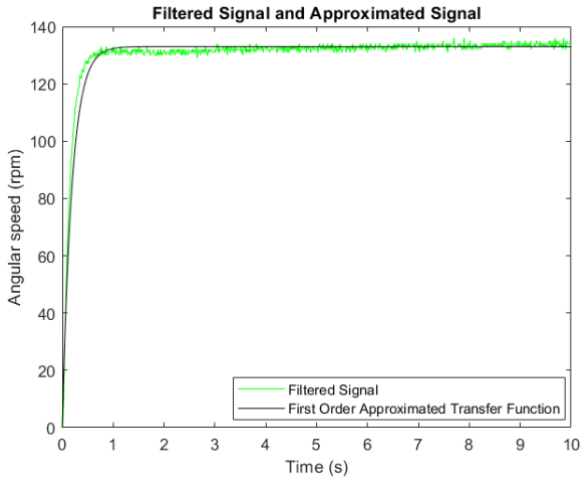

Fig. 3: Comparison of Filtered Signal and First Order Approximated Signal

As it can be seen in Figure 3, filtered signal and first order approximated signal are quite different. The reason for this is filtered signal is taken from nonlinear complex model. Even though it is low pass filtered, it is distorted due to its nonlinearity. However, if first order approximation is done on the data, it is linearized and becomes smooth.

Part II

In this part, PI (Proportional Integral) controller was designed. Design criteria for the controller was the following:

1. Steady state error is 0.
2. Maximum percentage overshoot is less than 10%
3. Settling time is less than 0.8 seconds (%2 error bound)

To achieve these, steps mentioned in Lab#1 Assignment were followed. Initially, coefficients $[K_i, K_p]$ are adjusted to [0.5, 0.5]. Thus, following system was designed in Simulink:

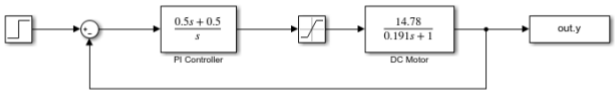

Fig. 4: Unity Feedback System With PI Controller

This design did not meet the criteria so $[K_i, K_p]$ coefficients were adjusted until given criteria were met. It was found to be met when $K_i = 1$ and $K_p = 0.2$. Its graph is the following:

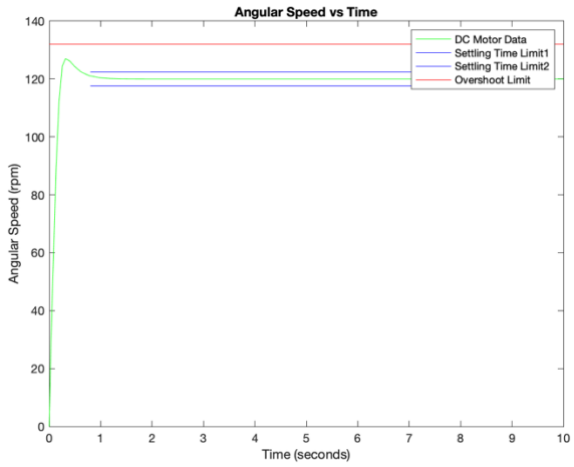

Fig. 5: DC Motor Data Meeting Criteria with Coefficients $[\boldsymbol{K_i}, \boldsymbol{K_p}]$ = [1, 0.2]

As it can be seen in Figure 5, there are three lines. The red line is for 10% overshoot criteria and blue lines are 2% error bound. Since it is required that settling time is less than 0.8 seconds, blue lines are drawn from 0.8 second.

In addition to that, two unity feedback system with coefficients $K_p = 5K_i$ and $5K_p = K_i$, were designed. These are given below:

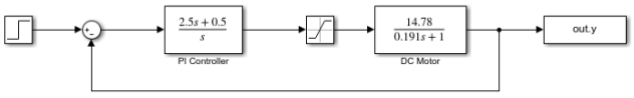

Fig. 6: Unity Feedback PI Controller System With Coefficients $[\boldsymbol{K_i}, \boldsymbol{K_p}]$ = [0.5, 2.5]

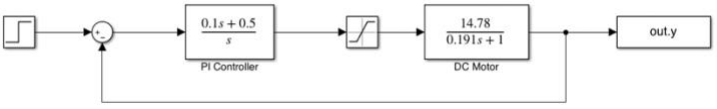

Fig. 7: Unity Feedback PI Controller System With Coefficients $[\boldsymbol{K_i}, \boldsymbol{K_p}]$ = [0.1, 0.5]

The plot of the systems in Figure 7 and 8 are given below:

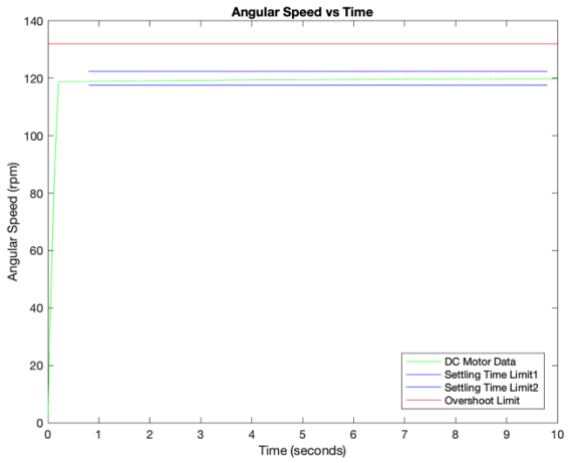

Fig. 8: DC Motor Data Meeting Criteria with Coefficients $[K_i, K_p] = [0.5, 2.5]$

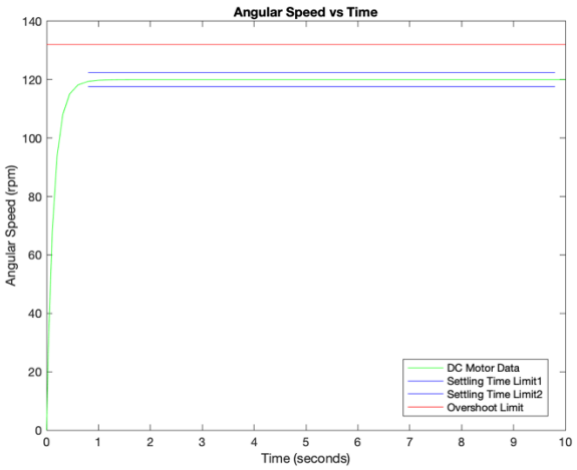

Fig. 9: DC Motor Data Meeting Criteria with Coefficients $[K_i, K_p] = [0.5, 0.1]$

Clearly, both signals given in Figure 8 and 9 are met the criteria. The work done in preliminary work is observed again. From Figure 8 and 9, it seems that when $5K_i = K_p$

signal goes smoother. However, if the reverse is true, it goes sharper. They both reach the same steady state value that shows, coefficients do not affect steady state value. The MATLAB code to achieve these plots is given below:

MATLAB Code:

```
figure();
plot(out.y, 'green');
hold on;
upperbound = 120+120*0.02;
t1 = (0.8:10);
upperbound = ones(1,length(t1))*upperbound;
lowerbound = 120-120*0.02;
lowerbound = ones(1,length(t1))*lowerbound;
plot(t1,upperbound,'b');
hold on
plot(t1,lowerbound,'b');
hold on
t2 = (0:10);
overshot = 132;
overshot = ones(1,length(t2))*overshot;
plot(t2,overshot,'r');
legend("DC Motor Data","Settling Time
Limit1","Settling Time Limit2","Overshoot Limit")
xlabel('Time (seconds)');
ylabel('Angular Speed (rpm)');
title('Angular Speed vs Time')
```

Also, note that in calculation of time constant, the data at given t is taken from the response. The response data of DC motor and the value taken to calculate time constant can be seen below:

Time	Data:1
0.0495	29.5379
0.0496	29.7620
0.0497	29.9881
0.0498	30.2163
0.0499	30.4462
0.0500	30.6777

Table 1: Data Taken From DC Motor Response To Calculate Time Constant

Part III

In this part, PI controller is tested on nonlinear model of the DC motor. “DCmotor.mat” was downloaded to the workspace and final design of PI controller in part 2 is implemented to nonlinear model of the DC motor. Note that the input, r(t), is equal to 120 rpm. The plot is the following

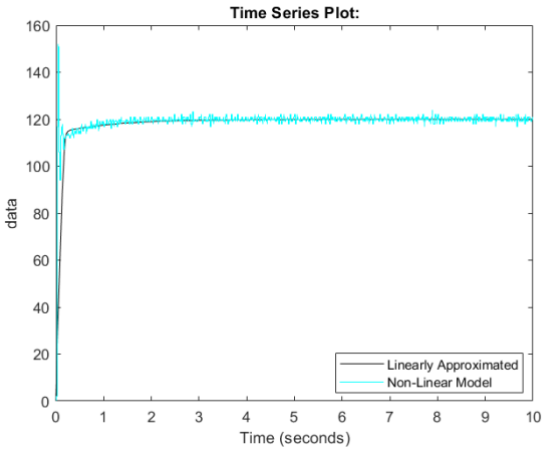

Fig. 9: Comparison of Linearly Approximated Signal and Non-Linear Model

MATLAB Code for the plot in Figure 9 is

```
figure();
approximated = out.y; %while running this part comment
%this
plot(approximated, 'black');
hold on
plot(out.y,'cyan');
hold on
legend("Linearly Approximated","Non-Linear Model");
xlabel('Time (Seconds)');
ylabel('data');
title('Time Series Plot');
hold off;
```

More detailed MATLAB code is provided in zip folder. As it can be seen in Figure 9, linear model does not exceed the steady state value. Even if it exceeds it, it is very small and negligible. However, in nonlinear case, signal exceeds the steady state value too much in its transient response. Furthermore, it is so noisy that in steady state, there are distortions which can clearly be seen. The distortions are generated due to nonlinearity of the signal and since the black signal is linearly approximated, there is no distortion.

Derivation of Equation (1) from Preliminary Work:

Motor Torque of DC motor [1]:

$$T_m(s) = \left(K_1 K_f I_f\right) I_a(s) \text{ (Eq. 3)}$$

$$= K_m I_a(s) \text{ (Eq. 4)}$$

$$where\ K_m = K_1 K_f I_f \text{ (Eq. 5)}$$

K_m is a function of the permeability of the magnetic field. The armature current depends on the input voltage given to the armature as follows

$$V_a(s) = (R_a + L_a s) I_a(s) + V_b(s), \quad \text{(Eq. 6)}$$

where $V_b(s)$, L_a, and R_a are the back electromotive-force voltage proportional to the motor speed, inductance, and resistance respectively. Thus,

$$V_b(s) = K_b w(s), \quad (Eq. 7)$$

where w(s) is the Laplace transform of angular velocity. From physical unknowns, we know that

$$w(s) = \frac{T_m}{Js + b} = \frac{K_m I_a(s)}{Js + b} \quad (Eq. 8)$$

Where J is the inertia of the load and b is friction constant. Note that equations 3-6 are DC motor's Kirchhoff equations.

By combining equation 6, 7 and 8, following equation is acquired:

$$G_p(s) = \frac{w(s)}{V_a(s)}$$

$$= \frac{\frac{K_m I_a(s)}{Js + b}}{I_a(s)(sL_a + R_a) + K_b \frac{K_m I_a(s)}{Js + b}} \quad (Eq. 9)$$

$$= \frac{K_m}{(sL_a + R_a)(Js + b) + K_b K_m} \quad (Eq. 10)$$

The reason for the name $G_p(s)$ is because it is process function. Note that before writing these equations, disturbance is ignored since it is negligible, and the purpose of this work is doing a first order approximation, so it is intentionally ignored. Additionally, to turn the equation 8 to first order approximation, inductance is assumed to be 0, $L_a = 0$. Hence,

$$G_p(s) \cong \frac{K_m}{R_a(Js + b) + K_b K_m} \quad (Eq. 11)$$

$$= \frac{K_m}{R_a Js + bR_a + K_b K_m} \quad (Eq. 12)$$

$$= \frac{\frac{K_m}{bR_a + K_b K_m}}{\frac{R_a Js}{bR_a + K_b K_m} + 1} \quad (Eq. 13)$$

$$= \frac{K_1}{\tau_1 s + 1} \quad (Eq. 14)$$

where $K_1 = \frac{K_m}{bR_a + K_b K_m}$, and $\tau_1 = \frac{R_a J}{bR_a + K_b K_m}$ (Eq. 15), (Eq. 16)

$$V_a(t) = 9u(t) \quad (Eq. 17)$$

Eq. 17 is given by DC motor simulation, the input voltage to DC motor is 9V. Therefore, the output voltage is

$$y(t) = g_p(t) * v_a(t) \quad (Eq. 18)$$

Symbol * denotes convolution operation. In Laplace domain, it corresponds to multiplication.

$$Y(s) = G_p(s) V_a(s). \quad (Eq. 19)$$

Thus, by equation 12 and 15,

$$Y(s) = \frac{9K_1}{(\tau_1 s + 1)s} \quad (Eq.20)$$

Because the Laplace transform of unit step is 1/s. To get the inverse Laplace transform of the equation 18, partial fraction expansion will be done.

$$Y(s) = \frac{9K_1}{(\tau_1 s + 1)s} = \frac{k_1}{s} + \frac{k_2}{(\tau_1 s + 1)} \quad (Eq.21)$$

$$k_1 = sY(s)|_{s=0} = 9K_1, \qquad (Eq.22)$$

$$k_2 = (\tau_1 s + 1)Y(s)|_{s=-1/\tau_1} = -9K_1\tau_1 \quad (Eq.23)$$

$$Y(s) = 9K_1\left(\frac{1}{s} - \frac{1}{\left(s + \frac{1}{\tau_1}\right)}\right). \quad (Eq.24)$$

Finally, by using the Laplace transformation in table [1],

$$y(t) = 9K_1\left(1 - e^{-\frac{t}{\tau_1}}\right)u(t) \quad (Eq.25)$$

There are two unknowns. To find K_1, steady state response will be used, which is as time goes to infinity. Steady state can be found from the data given by writing the command round(mean(y)) to Matlab. It is 133 so

$$\lim_{t\to\infty} 9K_1\left(1 - e^{-\frac{t}{\tau_1}}\right) = 9K_1 = 133\ (Eq.26), (Eq.27)$$

$$K_1 = \frac{97}{6} = 14.78\ (Eq.28)$$

After that, a sample point, which is in steady state, is chosen and put into the equation. Time constant is found to be 0.191s. Sample point chosen can be seen in Table 1.

3. Conclusion

The purpose of this lab work was to design PI controllers by using trial and error method. To identify the DC motor, first order approximation was done on received data. Then, three different PI controllers were designed by using first order transfer function explained in Lab#1 Assignment part 2. Lastly, closed loop PI controllers implemented on nonlinear model of the DC motor and their responses were checked. All the criteria were met given in the second part. There were very small errors for steady state, but I asked this issue to TA and he said no problem. In this lab, I learnt how to derive first order approximation transfer function, how to design unity feedback control system, how to link the workspace with Simulink and how to link the DC motor simulation with a system.

REFERENCES

1. R. H. Bishop and R. C. Dorf, *Modern Control Systems: International Edition*, 10th ed. Upper Saddle River, NJ: Pearson, 2005.