Lab 01 Report - System Identification

Oğuz Altan

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

1. Introduction

In this laboratory assignment, we use DC motor driven with Arduino to get hardware data from motor. In the preliminary report, we were asked to estimate the DC motor parameters using transfer functions of components in the diagram and simulate the DC motor parameters on MATLAB Simulink. In the laboratory work, we find DC motor parameters from the hardware experiments changing the LPF parameters and compare the simulation result with actual hardware result. We get different position data from the motion of DC motor for different LPF parameters in the diagram. Then, we use settling time and overshoot estimations to estimate DC motor parameters and see how changing the LPF parameters change DC motor model parameters. Finally, we check the accuracy doing the comparison between the simulation result and hardware result.

2. Laboratory Content

In the preliminary report, we have calculated the generalized transfer function of the whole block diagram as:

$$G(s) = \frac{K}{\tau s + 1} \qquad H(s) = \frac{T_{LPF}G(s)}{1 + T_{LPF}G(s)}$$

$$H(s) = \frac{\frac{K_1}{\tau_1 s + 1}\frac{K}{\tau s + 1}}{1 + \frac{K_1}{\tau_1 s + 1}\frac{K}{\tau s + 1}}$$

$$H(s) = \frac{K_1 K}{(\tau_1 s + 1)(\tau s + 1) + K_1 K}$$

For the position output T(s) we use the voltage source, where A denotes the amplifier parameter:

$$T(s) = \frac{\frac{6H(s)A}{s}}{1 + \frac{6H(s)A}{s}}$$

$$T(s) = \frac{\frac{6AK_1K}{s(\tau_1 s + 1)(\tau s + 1) + sK_1K}}{1 + \frac{6AK_1K}{s(\tau_1 s + 1)(\tau s + 1) + sK_1K}}$$

$$T(s) = \frac{6AK_1K}{s(\tau_1 s + 1)(\tau s + 1) + sK_1K + 6AK_1K}$$

$$T(s) = \frac{6AK_1K}{\tau_1\tau s^3 + (\tau_1 + \tau)s^2 + (K_1K + 1)s + 6AK_1K}$$

$$T(s) = \frac{w_0^2}{(s^2 + 2\zeta w_0 s + w_0^2\,)(\tau_x s + 1)}$$

Equating the two equations, we find the equalities below:

$$\tau_x = \tau_1\tau a$$

$$2\zeta w_0\tau_x + 1 = (\tau_1 + \tau)a$$

$$2\zeta w_0\tau_1\tau a + 1 = (\tau_1 + \tau)a$$

$$2\zeta w_0 + w_0^2\tau_x = (K_1K + 1)a$$

$$2\zeta w_0 + w_0^2\tau_1\tau a = (K_1K + 1)a$$

$$w_0^2 = 6AK_1Ka$$

These fundamental equations are used to calculate:

$$Ka = \frac{w_0^2}{6AK_1}$$

$$2\zeta w_0\tau_1\tau a + 1 = (\tau_1 a + \tau a)$$

$$\tau a(2\zeta w_0\tau_1 - 1) = (\tau_1 a - 1)$$

$$\tau a = \frac{\tau_1 a - 1}{2\zeta w_0\tau_1 - 1}$$

$$2\zeta w_0 + w_0^2\tau_1\tau a = (K_1Ka + a)$$

$$\tau a = \frac{(K_1Ka + a - 2\zeta w_0)}{w_0^2\tau_1}$$

From the equality of $\tau a's$ in the equations:

$$\tau a = \frac{\tau_1 a - 1}{2\zeta w_0\tau_1 - 1} = \frac{(a + K_1Ka - 2\zeta w_0)}{w_0^2\tau_1}$$

$$(a + K_1Ka - 2\zeta w_0)(2\zeta w_0\tau_1 - 1) = \ (w_0^2\tau_1)(\tau_1 a - 1)$$

$$(2\zeta w_0\tau_1 a - a + 2\zeta w_0\tau_1K_1Ka - K_1K - 4\zeta^2w_0^2\tau_1 + 2\zeta w_0) = (w_0^2\tau_1^2 a - w_0^2\tau_1)$$

Now, the scalar **a** becomes:

$$a = \frac{w_0^2 - 6Aw_0^2\tau_1 - 2\zeta w^3\tau_1 + 24A\zeta^2 w^2\tau_1 - 12A\zeta w_0}{6A(2\zeta w_0\tau_1 - 1 - w_0^2\tau_1^2)}$$

As we find a, we can write K and τ as below:

$$K = \frac{w_0^2}{6AKa_1}$$

$$\tau = \frac{(K_1K + 1)a - 2\zeta w_0}{w_0^2\tau_1 a}$$

These parameters construct the general case for the estmatng DC motor model parameters by changing the LPF parameters.

Using the MATLAB code, attached to the end of report, we calculate and find all of the necessary coefficients and constants and then show the plots for each experiment.

Experiments

Changing the LPF parameters, we have calculated 3 different DC motor model parameter estimations and plotted the position versus time plot for each model and experiment.

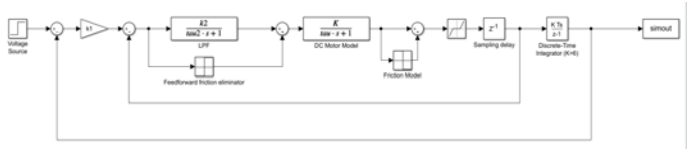

Fig. 1: Simulink Diagram

First Experiment: $C_p = 9$ $LPF = \frac{0.01}{0.08s+1}$

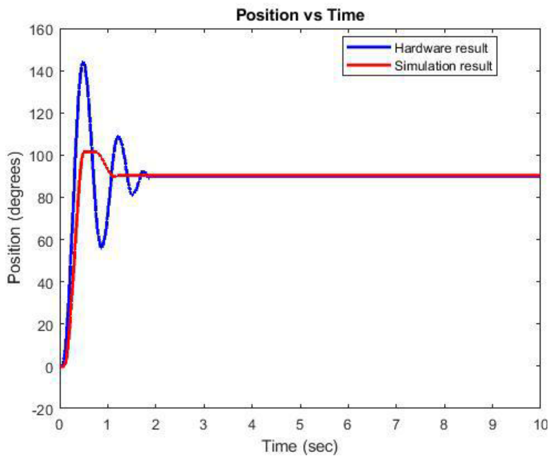

Fig. 2: Figure of the first experiment

For the first experiment, natural frequency and damping coefficient are found as:

$w_0 = 14.9 \qquad \zeta = 0.112$

Observing this plot, it can be said that due to the nonlinearity, settling time and overshoot of the simuliaton is lower than the hardware test.

Second Experiment: $C_p = 2$ $LPF = \frac{0.01}{0.6s+1}$

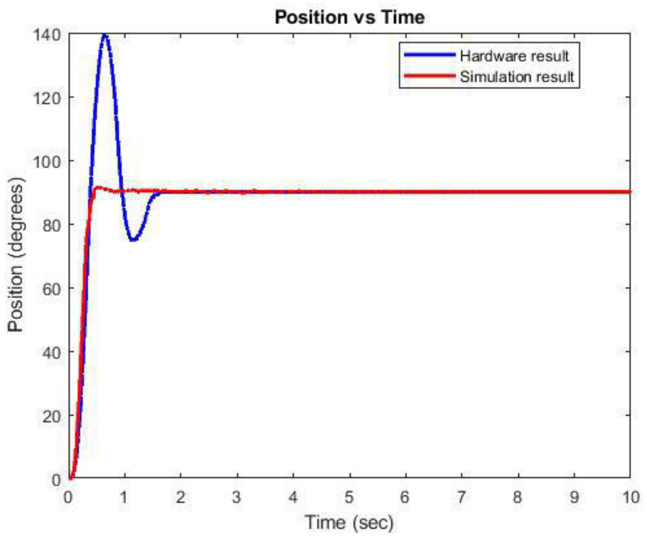

Fig. 3: Figure of the second experiment

For the second experiment, natural frequency and damping coefficient are found as:

$w_0 = 12.9 \qquad \zeta = 0.19$

Third Experiment: $C_p = 2$ $LPF = \frac{0.01}{0.05s+1}$

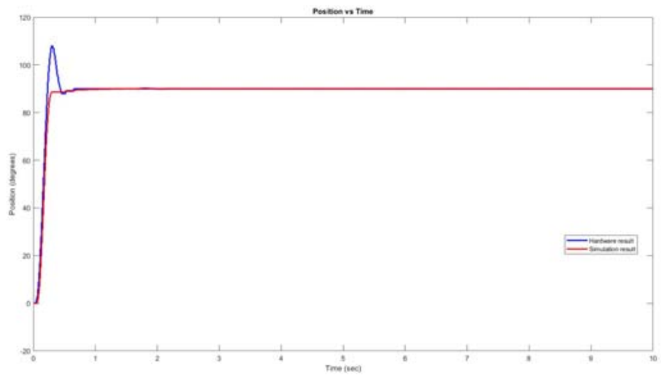

Fig. 4: Figure of the third experiment

For the third experiment, natural frequency and damping coefficient are found as:

$$w_0 = 3.86 \qquad \zeta = 0.65$$

Again observing the Figure 4, above, due to nonlinearity, overshoot is very low, nearly zero.

Bode Plot

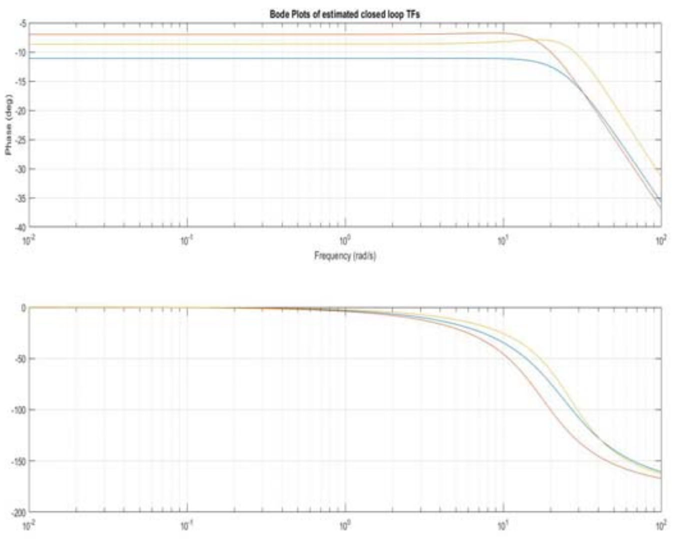

Fig. 5: Bode plot of estimated closed loop TFs

Analyzing the bode plot, it can be said that for the magnitude plot, for the natural frequencies, magnitude is highest. For the phase plot, real part is the dominant component for the frequencies lower than natural frequency. For frequencies higher that natural frequency, importance and effect of imaginary part increases.

3. Conclusion

In this laboratory assignment, our aim was to estimate the DC motor model parameters and transfer function. We used Arduino to control the DC motor and read the position data of DC motor using MATLAB and Simulink.

After, we got the data provided by DC motor for various LPF parameters and gain. The natural frequencies and damping coefficients are found using overshoot and settling time approximations. With these parameters, new models are simulated and the result of these simulations are compared with the hardware result, namely with the data provided by DC motor as hardware. This comparison shows the quality of our approximations and calculations of our parameters.

As a last part, we plotted the bode diagram of closed loop transfer functions, magnitude and phase plots of each transfer function on the same plot.

There happened some less important errors and issues that can be observed on the plots. These are due to the nonlinearity of the system and our assumption of second order dominance and relative approximations. In addition, as the motor is a mechanical hardware, it is not expected from it to behave perfectly and ideally. Therefore, there may occur physical problems such as friction, overheating, also communication problems with Arduino and so on. Thus, although the simulation outputs are not ideal and identical with the hardware output, nevertheless our approximations are very close to the exact values and parameters.

Also, observing the behavior of parameter K, it can be said that when K and gain of the amplifier increases, the overshoot increases because the time constant increase which results in the increase of settling time. This show the relation between these parameters.

MATLAB Code

Lab1.m

```
clc;
close all;
clear all;
% load('prelab1_posData.mat');
% load('9-008-pos.mat');
 load('2-005-pos.mat');
% load('2-06-pos.mat');
% plot(position);
samples = getdatasamples(position, [1:1001]);
[maxSample, indexMax] = max(samples);
indexMax = indexMax / 100;
steady = double(samples(length(samples)));
M = double(maxSample) / 90;
s2u = steady * 1.02;
s2d = steady * 0.98;
for k = 36:500 % prelim -- 195, 008 -- 152, 06-- 117, 005
-- 36
   if double(samples(k)) <= s2u && double(samples(k))
>= s2d
      settling = (k - 1) / 100;
      break
   end
end
clear s2u s2d k
output = ['T_peak = ', num2str(indexMax)]; disp(output);
output = ['T_settling = ', num2str(settling)]; disp(output);
disp(" ");
```

```
zeta = abs(log(M)) / sqrt((pi^2) + (log(M))^2);
omega_natural = 4 / (zeta * settling);
output = ['zeta = ', num2str(zeta)]; disp(output);
output = ['omega = ', num2str(omega_natural)];
disp(output); disp(" ");

K1 = 0.1;
tau1 = 0.05;
amp = 2;

first = omega_natural ^ 2;
second = -6 * amp * first * tau1;
third = -2 * zeta * (omega_natural ^ 3) * tau1;
fourth = 24 * amp * (zeta ^ 2) * first * tau1;
fifth = -12 * amp * zeta * omega_natural;
d_first = 12 * amp * zeta * omega_natural * tau1;
d_second = -6 * amp;
d_third = -6 * amp * (omega_natural ^ 2) * (tau1 ^ 2);
alpha = (first + second + third + fourth + fifth) / (d_first
+ d_second + d_third);

K = (omega_natural ^ 2) / (6 * K1 * amp * alpha);
tau = ((tau1 * alpha) - 1) / (alpha * ((2 * zeta *
omega_natural * tau1) - 1));
tau_x = tau1 * tau * alpha;
output = ['alpha = ', num2str(alpha)]; disp(output);
output = ['K = ', num2str(K)]; disp(output);
output = ['tau_x = ', num2str(tau_x)]; disp(output);
output = ['tau = ', num2str(tau)]; disp(output); disp(" ");
clear output first second third fourth fifth d_first
d_second d_third M samples
```

- **Plot_g.m**

```
figure; plot(position,'b','LineWidth',2);
hold on; plot(pos_sim,'r','LineWidth',2);
xlabel('Time (sec)'); ylabel('Position (degrees)');
title('Position vs Time');
legend('Hardware result','Simulation result', 'Location',
'best');
```

- **Bode.m**

```
% % Initial Parameters
Cp = [1, 1, 1]; % change this Cp(1) --> first Cp etc.
% LPF = K_LPF/(t_LPF*s+1)
K_LPF = [0.01, 0.1, 0.1]; % change this
t_LPF = [0.08, 0.6, 0.05]; % change this
% P = Km/(tm*s+1)
Km = [41.857431678528650, 32.472131621434222,
12.041188559436800]; % change this
tm = [0.030664965569721, 0.043706000000000,
0.097113281786871]; % change this
% Plot Bode Diagram of all 3 CLs
figure;
for k=1:3
[w,Tp] =
freq_data(Cp(k),K_LPF(k),t_LPF(k),Km(k),tm(k))
subplot(2,1,1);
semilogx(w,20*log10(abs(Tp))); grid on; hold on;
subplot(2,1,2);
semilogx(w,unwrap(angle(Tp))*180/pi); grid on; hold
on;
end
subplot(2,1,1); ylabel('Mag (dB)');
title('Bode Plots of estimated closed loop TFs');
subplot(2,1,1); ylabel('Phase (deg)');
xlabel('Frequency (rad/s)');
function [w,Tp] = freq_data(Cp,K_LPF,t_LPF,Km,tm)
no_of_samples = 500;
w = logspace(-2,2,no_of_samples);
s = 1j*w;
Pv = Km./(tm*s+1);
Cv = K_LPF./(t_LPF*s+1);
Tv = (Pv.*Cv)./(1+Pv.*Cv);
Tp = Cp*Tv./(1+Cp*Tv);
end
```