EEE342 Feedback Control Systems-Preliminary Work 1
First-Order Approximate Transfer Function Estimation of a DC Motor Using Simulink Filtering and Experimental Data Analysis

Ezgi Demir

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

1. Introduction

This lab aims to determine a first-order approximate transfer function of a DC motor by analyzing its angular velocity response to a step input. Due to measurement noise, a low-pass filter is applied in Simulink before system identification. The motor's steady-state gain and time constant are estimated from the filtered response to derive the transfer function, which is then compared with the original data for validation. Furthermore, the effects of Proportional (P) and Proportional-Integral (PI) controllers on system response are studied, focusing on overshoot, settling time, and peak time. The necessity of the integral controller in reducing steady-state error for velocity control applications is also discussed.

2. Laboratory Content

Part 1

The MATLAB workspace file prelab1_response.mat is downloaded from Moodle and includes time and velocity data sampled at 10 kHz from a DC motor simulation. A 6V step input is applied to the motor, and its angular velocity is recorded. In this section, the objective is to determine the first-order approximate transfer function of the DC motor. We willl start with, the motor torque. Equations starting from Eq.1 are the proof of how we ended up with a basic first order transfer function.

Km is a function of the permeability of the magnetic field, where La and Ra represent the inductance and resistance, respectively, and Vb(s) is the back electromotive-force voltage proportional to the motor speed.

$$K_m = K_1 K_f I_f \quad (Eq.1)$$

$$T_m(s) = \left(K_1 K_f I_f\right) I_a(s) \quad (Eq.2)$$

$$therefore, \qquad K_m I_a(s) \quad (Eq.3)$$

$$V_b(s) = K_b \omega(s) \quad (\text{Eq.}5)$$

$$\omega(s) = \frac{T_m}{Js+b} = \frac{K_m I_a(s)}{Js+b} \quad (Eq.6)$$

Note that, J represents the inertia of the load and b is friction constant.

$$G(s) = \frac{\omega(s)}{V_a(s)} = \frac{\frac{K_m I_a(s)}{Js+b}}{I_a(s)(sL_a + R_a) + \frac{K_m I_a(s)}{Js+b} K_b} \quad (Eq.7)$$

$$= \frac{K_m}{(Js+b)(sL_a + R_a) + K_b K_m}$$

We say,

$$K = \frac{K_m}{bR_a + K_m K_b}, and\ \tau = \frac{R_a J}{bR_a + K_m K_b} \quad (Eq.8)$$

Making the assumption that $L_a \ll R_a$. We can ignore L_a. Therefore our transfer function turns into this and a basic first order transfer function in a Laplace form will look like this if we also make the above changes,

$$G(s) = \frac{K}{\tau s + 1} \quad (Eq.9)$$

Where K represents the steady state gain and, τ represents the time constant which are 2 unknown variables. Starting from now on, we will continue the first order transfer function we derive and call this Eq. and start new calculations.

$$G(s) = \frac{K}{\tau s + 1} \quad (Eq.1)$$

As we are asked the input voltage of DC motor is 6V. So the input voltage and output voltage in time domain will look like as follows,

$$V_a(t) = 6u(t) \quad (Eq.2)$$

$$y(t) = g(t) * V_a(t) \quad (Eq.3)$$

Then again in the Laplace domain we get,

$$Y(s) = G(s).V_a(s) \quad (Eq.4)$$

$$Y(s) = \frac{6K}{(\tau s + 1)s} \quad (Eq.5)$$

I need a partial fraction expansion to compute Eq.5. therefore,

$$k_1 = (\tau s + 1)Y(s) \quad (Eq.6)$$

$$k_2 = sY(s) \quad (Eq.7)$$

$$\mathrm{y(s)} = 6K\left(\frac{1}{s} - \frac{1}{(s+\frac{1}{\tau})}\right) \quad (Eq.8)$$

$$\mathrm{y(t)} = 6K\left(1 - e^{-\frac{t}{\tau}}\right)u(t) \quad (Eq.9)$$

To determine K, the steady-state response will be utilized, which corresponds to the system's behavior as time approaches infinity. The steady-state value can be obtained from the given data by using the MATLAB command round(mean(y)). It is 98 as we take 2<t<10 .If we evaluate t at ∞ we get 6K,

$$K = \frac{98}{6} = 16.3333 \quad (Eq.10)$$

Then, using any point in output data we derive the time constant, we choose arbitrary value from the transient part which is around 0<t<1. So any value between this will give us same result. However, let us calculate when t = 0.05. The corresponding value of the data given us was 27.5291 at t = 0.05. So we plug it in eq. 9. Desmos calculation of Eq.9 will give us below result when we took K as we calculated in Eq.10,

$$\frac{-0.05}{\left(\ln\left(\left(1 - \frac{27.5291}{98}\right)\right)\right)} = 0.1516219209$$

Fig. 1: Desmos calculation for τ

Therefore, our first order approximation of DC motor's transfer function will look like this,

$$G(s) \cong \frac{16.3333}{1+0.1516s} \quad (Eq.11)$$

The pole of the approximation transfer function is approximately −6.5963.

$$|\Re\{p_{TF}\}| = 6.5963 < 100 = \frac{|\Re\{\mathrm{pLP\ F}\}|}{10} \quad (Eq.12)$$

We conclude that the pole of the transfer function is dominant over low-pass filter. Because we are given,

$$H_{LPF} = \frac{1}{1+0.001s} \quad (Eq.13)$$

Since the dataset is given noisy we are asked to reduce these fluctuations, a low-pass filter is applied (Eq.13), eliminating the high-frequency components and producing smoother data with reduced noise. We use above filter parameters as we are asked.

The applied low-pass filter is shown below. See my simulink block diagram(Fig. 5) and my MATLAB code to filter out data given us.

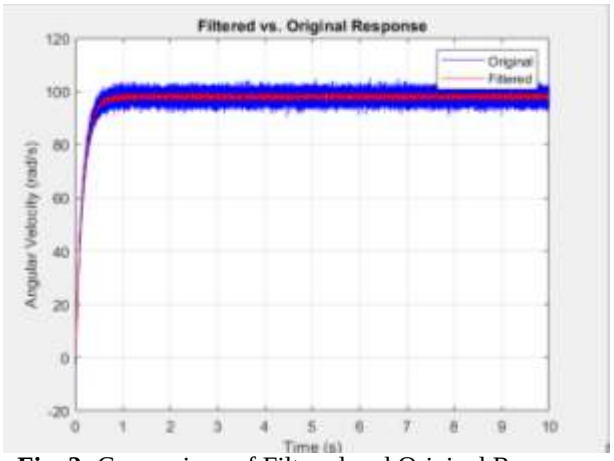

Fig. 2: Comparison of Filtered and Original Response

The red plot is filtered data and blue plot is noisy one.

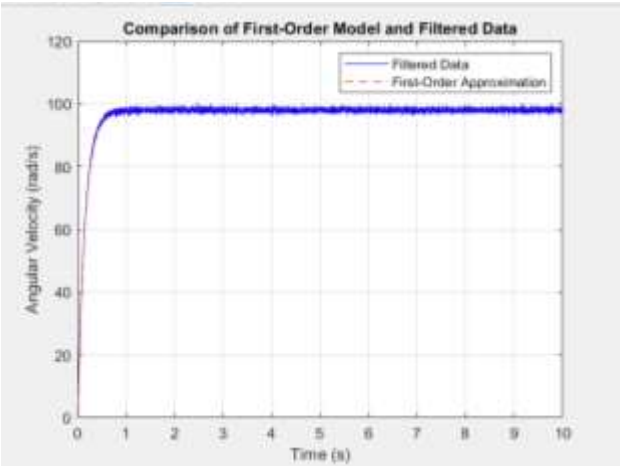

Fig. 3: Comparison of Basic First Order Model and our Simulink Filtered Data

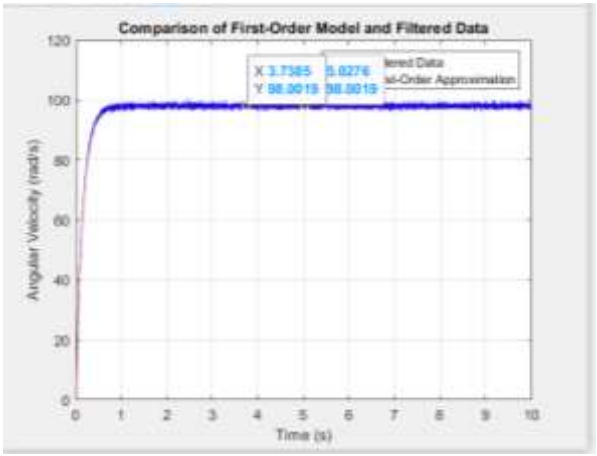

Fig. 4: Steady state reaches 98 rad/s as we found via round(mean(y)) command

Fig. 5: Simulink Block Diagram of the Filtered Data

Matlab Code

```
% Define the index range
start_idx = 20001;  % Corresponding to t = 20.001
(assuming a 10 kHz sampling rate)
end_idx = 100001;   % Corresponding to t = 100.001

% Compute the mean of y in this range and round it
rounded_mean = round(mean(y(start_idx:end_idx)));
y(.05/.0001+1); %to find y at .05

Va_s = double(6);

simulationVariables= [t, y];
data_input = [t, 6 * ones(size(t))];

plot(t, y, 'b');
xlabel('Time (s)');
ylabel('Angular Velocity (rad/s)');
title('Original DC Motor Response');
grid on;

out = sim('Lab1_filter');
filtered_output = out.simout;
filter_time = filtered_output.Time; data_filtered=
filtered_output.Data;

figure;
plot(t, y, 'b', filter_time, data_filtered,'r');
legend('Original', 'Filtered');
xlabel('Time (s)');
ylabel('Angular Velocity (rad/s)');
title('Filtered vs. Original Response');
grid on;

% Estimate K from mean response in interval 2 < t <
10
idx = (filter_time > 2) & (filert_time < 10);
K = mean(data_filtered(idx)) / 6; % Since input vol-
tage is 6V
% Estimate tau using an early time point
t_tau = 0.1;
omega_tau = interp1(filter_time, data_filtered,
t_tau);
tau = -t_tau / log(1 - omega_tau / (K * 6));

% Display results
fprintf('First-Order Transfer Function Estimation:
G(s) = %.3f / (%.3fs + 1)\n', K, tau);

sys = tf(K, [tau 1]); % First-order transfer func-
tion
[y, time_simul] = step(sys * 6, time); % Step res-
ponse

% Plot comparison
figure;
plot(filter_time, data_filtered, 'b', time_simul, y,
'r--');
legend('Data-Filtered', 'Approximation of First Or-
der');
xlabel('Time (s)');
ylabel('Angular Velocity (rad/s)');
title('Filtered Dats and First-Order Suggested Model
Comparison');
grid on;
```

Part 2. Proportional Controller (P) and Proportional Integral Controller (PI)

In velocity control applications, such as controlling the speed of a DC motor, external disturbances (e.g., friction, load variations) can cause a steady-state error when using only a P controller. Since a P controller alone cannot fully compensate for this error, an integral controller is necessary. The integral term accumulates the steady-state error and applies a corrective action to drive it to zero, ensuring the motor maintains the desired speed despite disturbances.

A P controller generates a control signal proportional to the error, which helps reduce rise time and enhance system response speed. However, it does not eliminate steady-state error. Additionally, it may lead to increased overshoot and potentially cause instability if not properly tuned.

An I controller, on the other hand, introduces an integral action that continuously sums past errors. The integral term eliminates steady-state error by adjusting the control signal based on accumulated deviations over time. However, excessive integral action can increase settling time and overshoot, potentially leading to instability and oscillations.

A PI controller combines the advantages of both P and I controllers. This combination ensures faster response while reducing steady-state error. The proportional term provides immediate correction, while the integral term addresses any persistent error over time.
When designing a PI controller, the proportional (P) controller can be included to reduce overshoot, while the

integral component is introduced to eliminate steady-state error. Implementing a P controller decreases rise time, but after reaching a certain reduction in steady-state error, further increasing Kp value primarily results in system overshoot. If the P controller is too aggressive, it can lead to oscillations. A PI controller effectively eliminates steady-state error; however, excessive integral action can still cause overshoot and longer settling times, negatively affecting the system's stability. Where we can see the trade-off's of the engineering world.

3. Conclusion

This lab effectively demonstrated the modeling of a DC motor system using a first-order approximation. The system's gain and time constant were determined from the filtered response data, and a comparison with the original response verified the accuracy of the approximation. The dominant pole condition was evaluated to ensure reliability. Additionally, the impact of P and PI controllers on system performance was analyzed, emphasizing the importance of integral action in eliminating steady-state error for velocity control.

REFERENCES

1. R. H. Bishop and R. C. Dorf, Modern Control Systems: International Edition, 10th ed. Upper Saddle River, NJ: Pearson, 2005.
2. https://www.desmos.com/scientific?lang=tr
3. Arif Bülent Özgüler, Spring 2025 Lecture Notes