The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University

Velocity Control

HexaMotor

Experiment objectives:

- Introduction to DC motor dynamic equations.
- Coulomb Friction identification and handling
- System identification using open loop step response, model validation.
- System identification using frequency response, model validation.
- Velocity control implementation. Proportional, Proportional - Integral control loops.
- Controller performance under load.

1. Introduction

DC motors are electromechanical devices that convert electrical energy into mechanical energy, using principles of electromagnetism. They are known for their simple construction, high reliability, and ease of control. These features make them suitable for various applications, from small electronic devices to large industrial machinery.

1.1. DC motor equations

The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in Figure 1.

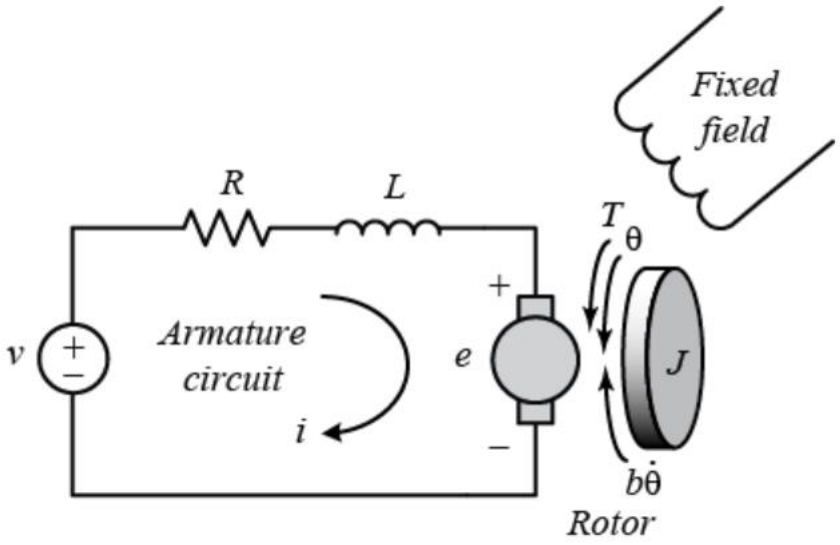


Figure 1: Electric equivalent circuit of a dc motor

V - Applied Voltage: The voltage applied across the motor's terminals. It is the input to the system and determines the driving force behind the motor.

i - Armature Current: The current flowing through the armature winding, responsible for creating the magnetic field that drives the motor.

R - Armature Resistance: The resistance of the armature winding, which opposes the flow of current.

L - Armature Inductance: The inductance of the armature winding, which determines how quickly the current can change in the coil.

$\boldsymbol{V}_{emf}$ - Back Electromotive Force (EMF): Voltage generated by the motor's rotation, opposing the applied voltage.

Kt - Torque Constant: A factor that relates the motor's torque output to its current. It depends on the motor's construction.

$\boldsymbol{K_e}$ - Back EMF Constant: A factor that relates the motor's back EMF to the angular velocity. It often equals the torque constant in many DC motors.

T - Torque: The twisting force exerted by the motor on its shaft.

$\mathbf{T}_{friction}$ - Friction Torque: The torque opposing the motor's rotation due to various frictional effects.

J - Moment of Inertia: A measure of the motor's resistance to changes in its rotational speed. It depends on the distribution of mass in the rotating parts.

ω - Angular Velocity: The rate of rotation of the motor's shaft, often the controlled output in velocity control systems.

B - Damping Coefficient: A parameter representing the viscous friction in the system. It characterizes how the frictional forces are proportional to the angular velocity.

$\mathbf{T}_{\mathrm{coulomb}}$ - Coulomb friction.

s - Complex Frequency (Laplace Variable): In the context of the transfer function, s is a complex variable used in the Laplace transform.

Voltage Equation:

1.1 $$V = \mathrm{Ri} + L\frac{\mathrm{di}}{\mathrm{dt}} + K_e\omega$$

where V is the applied voltage, R is the armature resistance, i is the armature current, L is the armature inductance, K_e is the back EMF constant, and ω is the angular velocity.

Torque Equation:

1.2 $$T = K_t\mathrm{i} - \mathrm{T}_{friction} - J\frac{\mathrm{d}\omega}{\mathrm{dt}}$$

where T is the torque, Kt is the torque constant, Tfriction is the friction torque, J is the moment of inertia, and $\frac{\mathrm{d}\omega}{\mathrm{dt}}$ is the angular acceleration.

Friction Model:

1.3 $$\mathrm{T}_{friction} = B\omega + \mathrm{T}_{\mathrm{coulomb}}$$

By substituting the expressions for torque and friction into the torque equation and solving for the differential equation relating the applied voltage to the angular velocity, we obtain:

1.4 $$LJ\frac{\mathrm{d}^2\omega}{\mathrm{dt}^2} + (RJ + BL)\frac{\mathrm{d}\omega}{\mathrm{dt}} + (\mathrm{BR} + K_tK_e)\omega = K_tV$$

Now, the transfer function relating the applied voltage V(s) to the angular velocity Ω(s) in the Laplace domain (assuming zero initial conditions) is:

1.5 $$G(s) = \frac{K_t}{LJs^2+(Rj+BL)s+(\mathrm{BR}+K_tK_e)}$$

1.2. First order approximation:

Armature Inductance (L): The inductive term $L\frac{\mathrm{di}}{\mathrm{dt}}$ in the voltage equation is often neglected. This is under the assumption that the armature inductance is small compared to the resistance, meaning that the electrical dynamics are much faster than the mechanical dynamics.

First-Order Transfer Function:

1.6 $$G(s) = \frac{K_t}{(RJ)s+(\mathrm{BR}+K_tK_e)}$$

This first-order model offers a simpler representation of the DC motor's dynamics and is useful for preliminary control design or when the details of the electrical subsystem are not critical. It captures the main characteristics of the motor's velocity response but omits some of the subtler dynamics present in the full second-order model.

For additional reading consider the following links:

DC motor velocity model:

https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling

System analysis:

https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemAnalysis

1.3. Coulomb friction

Coulomb friction is distinctive from viscous damping, as it remains constant regardless of the speed of the motor. It contributes to non-linearity in the system and can lead to challenges in maintaining precise control, especially at low speeds.

Measuring Coulomb friction in a DC motor is essential for designing effective control strategies. One common method to identify this friction involves applying a small, controlled voltage to the motor and observing the threshold voltage at which the motor begins to rotate. By relating this threshold voltage to the known torque-to-current relationship, the Coulomb friction torque can be quantified.

1.4. Step response modeling

Step response modeling, commonly referred to as the "bump test," is a straightforward examination grounded in the step response characteristics of a stable system. In this test, a step input is applied to the system, and the subsequent response is observed and recorded. To illustrate this concept, imagine a system defined by a particular transfer function:

1.7 $$G(s) = \frac{K}{\tau s+1}$$

The step response in the shown figure is generated using K=10 rad/V and $\tau = 0.1s$.

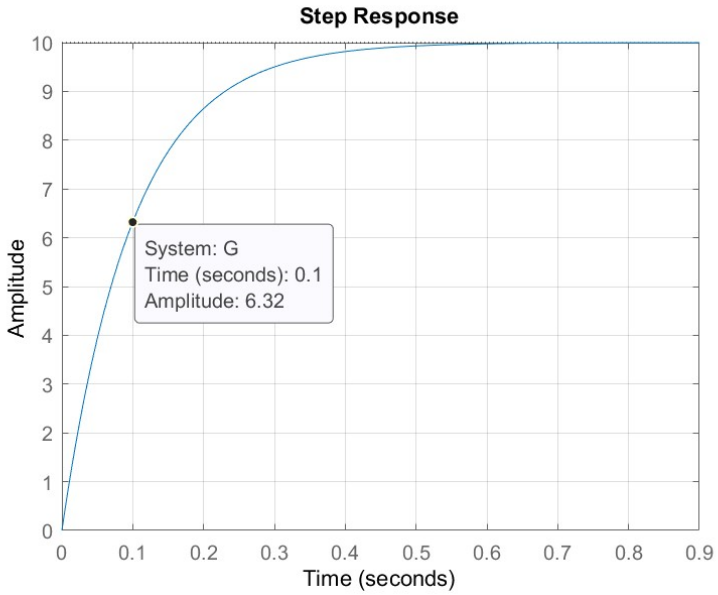


Figure 3: Step response of eq1.7 with K=10, τ=0.05s.

The time constant τ is a measure of how fast the system responds to a change in input. It's defined as the time it takes for the system's response to reach approximately 63.2% of its final value (steady-state value) after a step change in input.

To see why it is the case lets evaluate a first-order linear system that has the general form of a time constant τ and can be described by the following first-order differential equation:

1.8 $$\tau \frac{\mathrm{dy(t)}}{\mathrm{dt}} + \mathrm{y(t)} = Ku(t)$$

If the system is subjected to a step input of magnitude A, the response y(t) can be described by:

1.9 $$y(t) = KA(1 - e^{-\frac{t}{\tau}})$$

By evaluation the response at $t = \tau$ we get:

1.10 $$y(\tau) = KA(1 - e^{-\frac{\tau}{\tau}}) \approx KA \cdot 0.632$$

1.5. Frequency response modeling

System analysis with a sinusoidal input provides insights into the motor's frequency response, resonance behavior, and the effect of damping. The motor's transfer function can be used to investigate how different frequencies of the sinusoidal input are amplified or attenuated, revealing information about the stability, bandwidth, and transient response of the system.

When a sine wave is used as an input for a DC motor, the motor's output results in a sinusoid of the same frequency but is scaled and delayed. As illustrated in Figure 4, the length of one full period of the sinusoid is represented by t1, while t2 indicates the time delay between the input voltage signal, Vm, and the corresponding scaled output speed signal, ω_m.

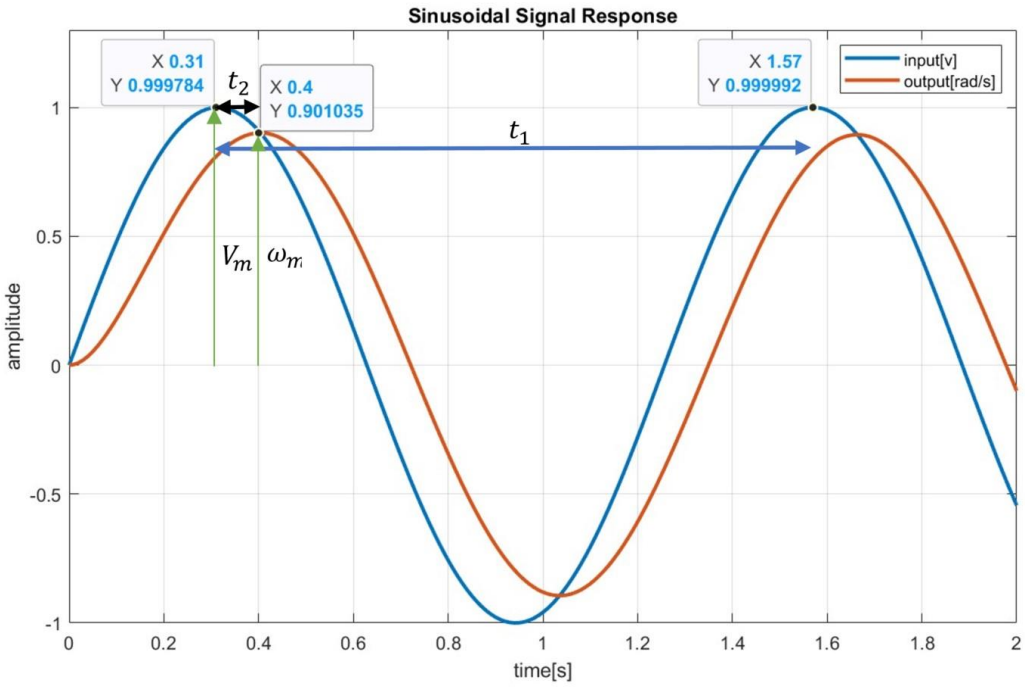


Figure 4: System response with a sinusoidal signal

The magnitude response of the resulting output changes with respect to the frequency of the applied sinusoid and can be used to find the time constant τ of the DC motor with the first order approximation. In addition, a bode plot of the system`s magnitude response can be obtained.

1.11 $$G(s) = \frac{\omega_m(s)}{V_m(s)} = \frac{K}{\tau s+1}$$

Substituting s = jω the magnitude of the frequency response at a frequency ω of the motor input voltage is then defined as

1.12 $$|G(s)| = \left|\frac{\omega_m(j\omega)}{V_m(j\omega)}\right| = \left|\frac{K}{\tau j\omega+1}\right| = \frac{K}{\sqrt{\tau^2\omega^2+1}}$$

The system's steady state (or low frequency) gain can then be obtained by setting $\omega = 0$, applying a constant signal:

1.13 $K = |G(0)|$

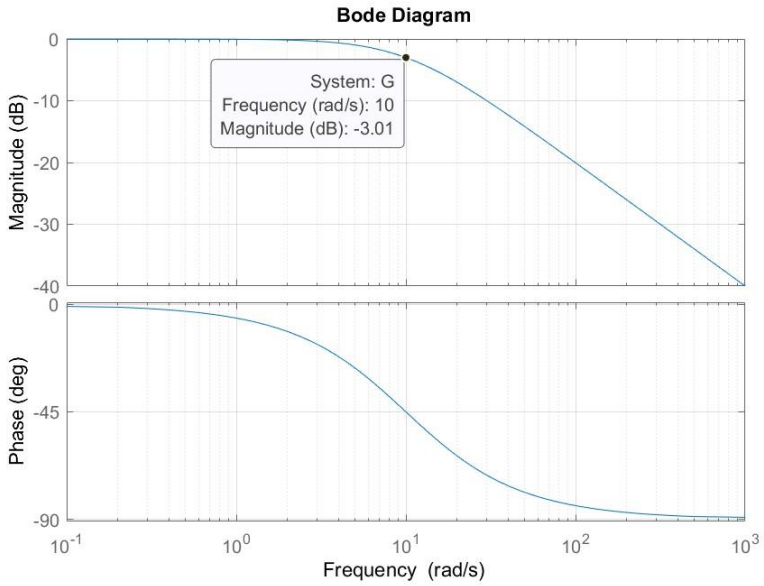
The phase delay of the system φ_d can be calculated as:

1.14 $\varphi_d = \varphi_{num} - \varphi_{den}$

When φ_{num} and φ_{den} represent the phase angle / delay of the numerator and denominator of the transfer function respectively.

In addition, it can be observed directly from the input output graphs for the sin signal. The phase shift can be expressed as:

1.15 $\varphi_d = -\frac{t_2}{t_1} \cdot 360^{\circ}$



The Bode plot includes a vital feature known as the cutoff frequency, denoted by ω_c. It's the frequency where the system's gain is 3 dB below its maximum steady-state gain. This 3 dB drop corresponds to the system's half-power point of the system, and defined as $10 \log_{10} 0.5$ leading to a value of approximately -3.01 dB, or equivalently a magnitude of approximately $1/\sqrt{2} \approx 0.70721 \approx 0.707$, or -3.01 dB in terms of the maximum system gain. The cutoff frequency is synonymous with the system's bandwidth and serves as an indicator of the speed at which the system reacts to an input.

1.6. PID controller

The PID controller, which stands for Proportional-Integral-Derivative, is one of the most widely used feedback control algorithms in industry. Its popularity stems from its effectiveness in a broad range of applications and its relatively straightforward tuning process.

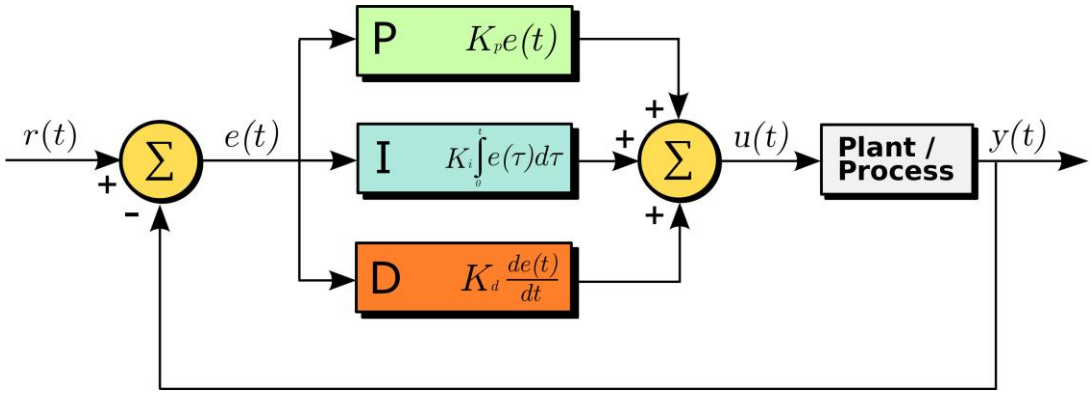


Figure 3: Block diagram of a PID controller

1.16 $$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

The PID controller consists of three main components:

Proportional (P): This part produces a control action that's directly proportional to the error between the setpoint and the measured process variable. The proportional gain Kp determines how aggressively the controller responds to an error.

Integral (I): The integral term is concerned with the accumulated error over time. If there is a persistent, steady-state error, the integral action will build up and eliminate it.

Derivative (D): The derivative action provides a control effort to counteract the rate of error change. It anticipates the future error by its rate of change and provides control action to prevent it from occurring.

For additional reading consider the following links:

Introduction: PID Controller Designhttps://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

1.6.1. Proportional speed control

In proportional control, the control action is directly proportional to the error between the desired set point and the actual measured value, in this case, the motor speed. The equation for the control action is given by: $u(t) = K_p e(t) = K_p(r(t) - y(t))$,

K_p is the proportional gain. $r(t) = \omega_d(t)$ is the desired angular velocity, $y(t) = \omega_m(t)$ is the measured angular velocity and $u(t) = v_m(t)$ is the control impute (the voltage applied to the motor).

The close loop transfer function can be derived

1.17 $$T(s) = \frac{Y(s)}{R(s)} = \frac{\mathrm{P(s)\cdot C(s)}}{1+\mathrm{P(s)\cdot C(s)}} = \frac{K\cdot K_p}{\tau s+1+K\cdot K_p}$$

Whereas P(s) is the first order approximation of the dc motor. And $\mathrm{C(s)} = K_p$ represent the controller transfer function.

1.6.1. Final value theorem

The Final Value Theorem states that if a system is stable and if the limit exists, the steady-state value of a time-domain function $f(t)$ can be found directly from its Laplace Transform $F(s)$. The theorem is given by:

1.18 $$\lim_{t\to\infty} f\,(t) = \lim_{s\to 0} s\cdot F(s)$$

1.6.2. Steady state error

Steady-state error is the difference between the desired and actual velocity of the DC motor once the transients have died down, and the system has reached its steady-state condition. The main reason for this steady-state error in proportional control lies in the nature of the control law itself. Since the control action is proportional to the error, when the error reduces to zero, so does the control input. In physical systems like a DC motor, there may be disturbances, such as friction, that require continuous control effort to overcome. If the control input becomes zero, these disturbances may cause the system to drift away from the desired set point.

Given a step input $R(s) = \frac{1}{s}$ The error transfer function can be derived as:

1.19 $$E(s) = R(s) - Y(s) = \; R(s) - \mathrm{T(s)\cdot R(s)} = \frac{1}{s} - \frac{K\cdot K_p}{s(\tau s+1+K\cdot K_p)}$$

By applying the final value theorem:

1.20 $$\lim_{t\to\infty} e\,(t) = \lim_{s\to 0} s\cdot E(s) = \lim_{s\to 0} s\left(\frac{1}{s} - \frac{K\cdot K_p}{s(\tau s+1+K\cdot K_p)}\right) = 1 - \frac{K\cdot K_p}{1+K\cdot K_p}$$

The steady state error is given by:

1.21 $$e_{ss} = 1 - \frac{K\cdot K_p}{1+K\cdot K_p}$$

2. Pre Lab

2.1. Step response modeling

a. Given the first order approximation of a dc motor, what would be the values of K and τ (give parametric solution).
b. Using Matlab, plot the step response of the following first order system $G(s) = \frac{1}{0.1s+1}$ Mark the time constant τ and the steady state gain K.

2.2. Frequency response modeling

a. Using Matlab plot the response of the system $G(s) = \frac{1}{0.1s+1}$ to various sin inputs (at list 5 different frequencies), calculate the gain change as a function of frequency from the sin responses. You can use the "lsim" function in Matlab to simulate a system response to an arbitrary input signal in the time domain. Or implement it in Simulink.
b. Approximate the cutoff frequency ω_c and plot the system`s sinusoidal response at this frequency.
c. Using the first order approximation $G(s) = \frac{K}{\tau s+1}$ and by evaluating the magnitude of the transfer function at the cutoff frequency ω_c , derive an expression to determine τ based on ω_c
Hint: $|G(\omega_c)| = \frac{1}{\sqrt{2}}|G(0)|$
d. Using the first order approximation $G(s) = \frac{K}{\tau s+1}$ and by evaluating the phase of the transfer function, derive an expression to determine τ in terms of the frequency input of the signal and the resulting phase delay.
Hint: $\varphi_d = Phase(\frac{K}{\tau s+1})\Big|_{s=j\omega}$

2.3. PID controller

a. Calculate the parametric steady state error for the given system $G(s) = \frac{1}{0.1s+1}$ in a close loop using a proportional gain controller Kp and a step response input.
b. Plot the response of the close loop system using Kp=10, compare the steady state value to the theoretical.

Bonus Questions:

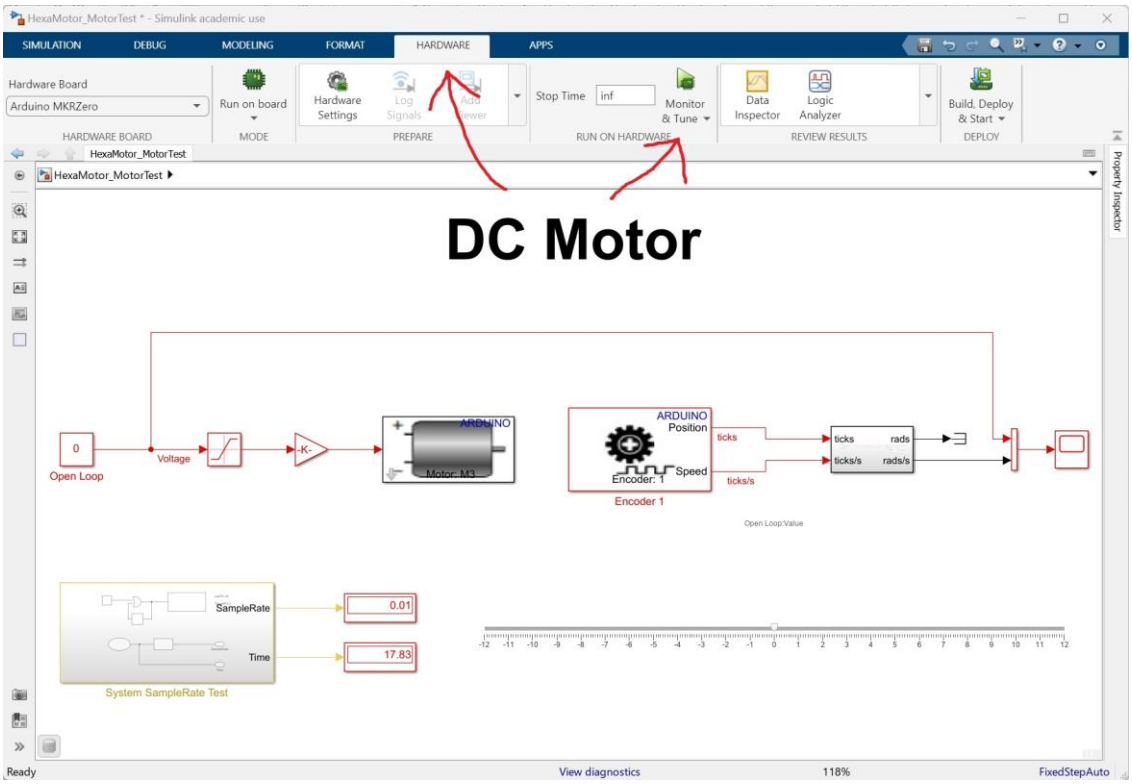
a. Calculate the transfer function of a closed loop velocity controlled dc motor with a PI Controller using the first order approximation for the dc motor transfer function.
b. Calculate the steady state error for the above.
c. For the given system: $G(s) = \frac{1}{0.1s+1}$ Design a PI controller (using any method) with rise time $0.2s > T_r > 0.1s$, overshoot of up to 15% and settling time of up to 0.5s.
d. Consider the DC motor equations, what will happen to the time constant τ in case an additional rotating mass will be added to the motor. Hint: consider what will happen to the inertia **J.**

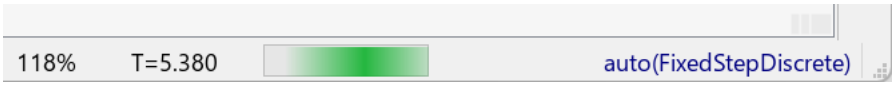
3. In Lab

3.1. Introduction

Welcome to the lab! Before we dive into the experiments, it's crucial to familiarize yourself with the hardware we'll be working with. The setup you will be working with is called HexaMotor. It includes a DC motor with an encoder connected to an Arduino MKR motor controller. This can be controlled using Matlab Simulink. Additional information is available in the appendix of this manual.

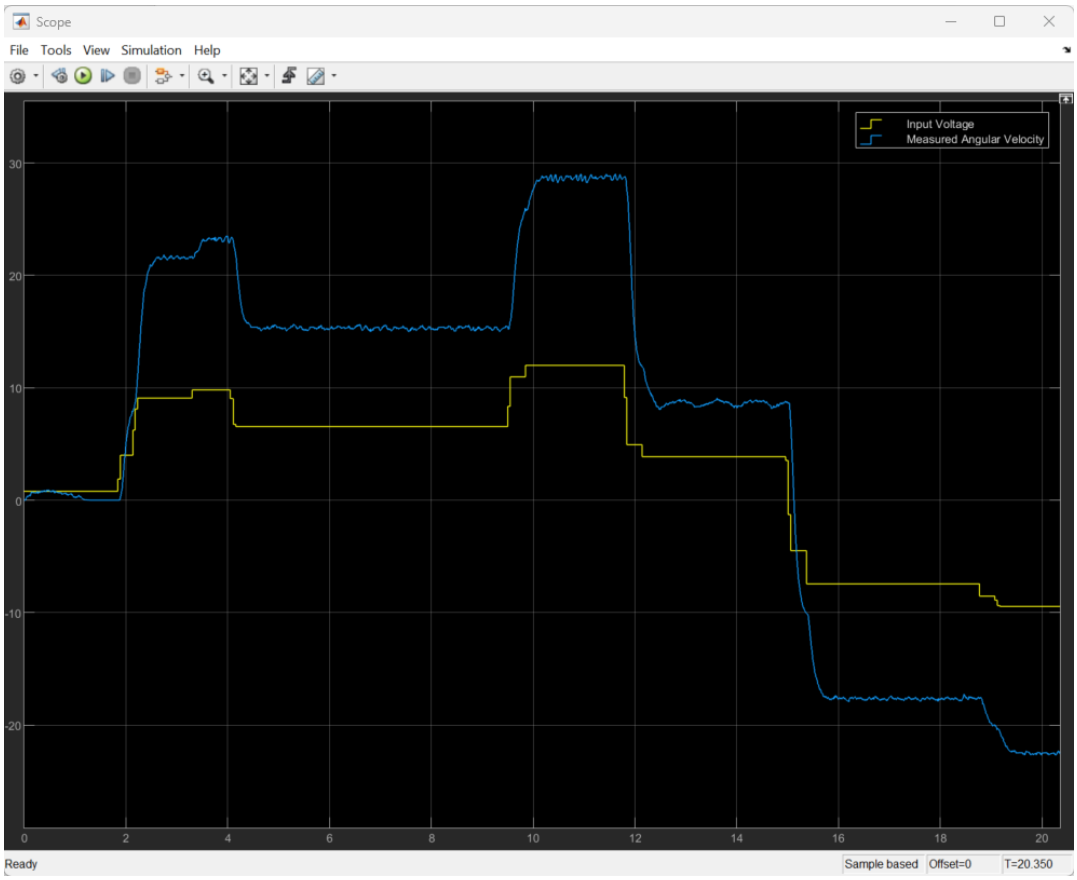
- Start by opening Matlab R2022b
- Navigate to …\Documents\MATLAB
- Create a new folder with your name and copy the lab materials into it
- Open "HexaMotor_MotorTest.slx" Simulink model.
- Prior to running the model make sure the setup is connected to the PC through USB cable and the power supply is connected to the socket.
- Press the On/Off Button at the back and make sure the light is on.
- Make sure the rotating mass is present.
- Under hardware you will find the Monitor&Tune button, press it, if everything is properly set it will compile the Simulink model and upload a C++ code to the micro controller. When the process has finished the button with be switch to a stop button and at the button bar a simulation time will start counting.



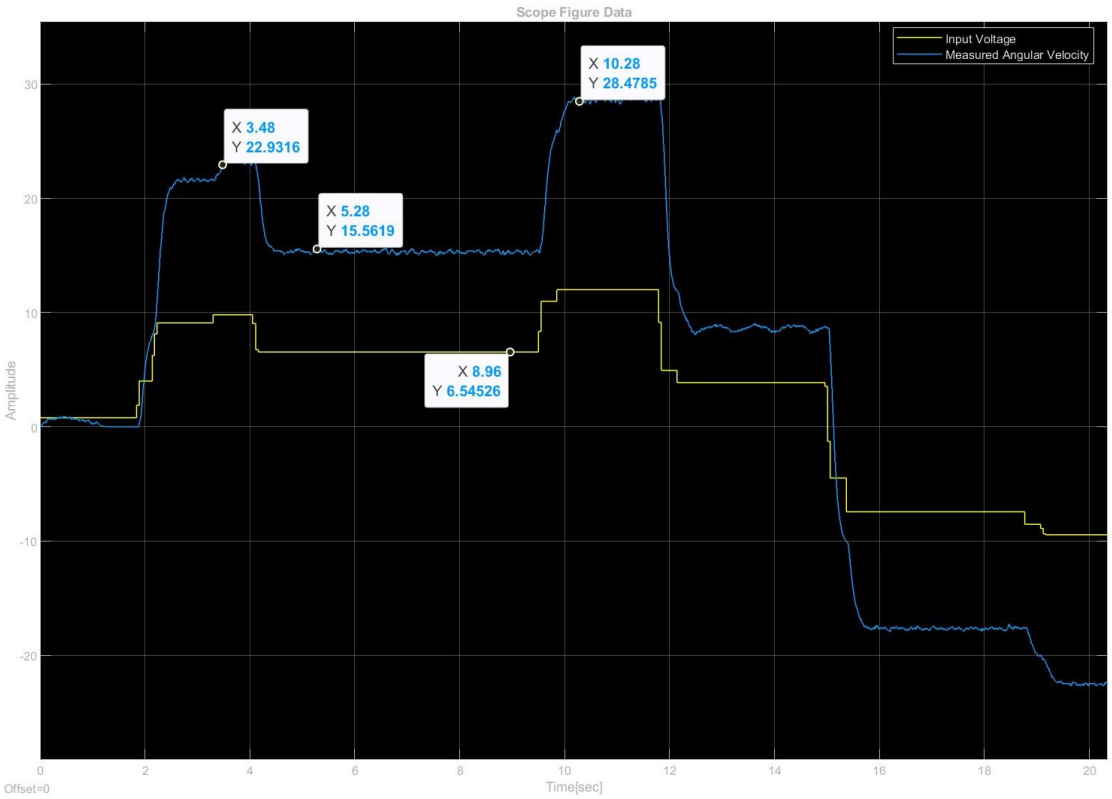


The Simulink model acts as an interface, allowing the interaction with the hardware and monitoring it`s signals.

- Double click on the scope, it will open the signals from the system. The angular velocity measured from the motor`s encoder. And the applied voltage to the motor.
- Change the open loop voltage, either by writing the open loop block values between [-12 ..12] or using the slide bar.



- On the button left you will notice a block named System SampleRate, the block outputs the simulation sample time and the measured time, and in additional performs I/O toggling which can be check with external hardware to verify the actual Sample Rate of the system.
- And at the top you near the Stop / Monitor & Tune button there is Stop Time, which can be change from inf to any number and will stop the simulation.
- You can export the scope to a figure by pressing File→Print To Figure, in the figure it is possible to make measurements by selecting Tools→Data Tips and clicking the plot. If you press Shift while selecting a data point it will add an additional data point. Then the data points can be moved around. Explore the figure and scope options and save a plot for the report. A good plot contains Title, Lables, Legend, units.



- In addition, the scope data is exported to the Matlab environment under the structure variable ScopeData when the simulation is stopped. You can access the data using the following commands:
  input = ScopeData.time;
  input = ScopeData.signals.values(:,1);
  output = ScopeData.signals.values(:,2);
  Or plot it directly using:
  plot(ScopeData.time,ScopeData.signals.values)
- In addition, it is possible to save the data for later processing by writing:
  save record_name ScopeData
  And retrieving it back to Matlab either by double clicking the saved file or writing:
  load('record_name.mat')

Note that you may notice some oscillation at constant voltage, that is due to non-linearity of the friction. If you measure the phenomena you may notice that its frequency is the same as the rotation angle frequency.

For the Report:

a. Present a plot using a figure from the scope simulation
b. Include a plot generated in the Matlab environment, using the ScopeData variable. Don't forget to include legend, Title, Lables and Grid to the plot. You can check how to add this by typing "help plot" in the Matlab environment.
c. Write the max velocity of the motor (at 12V, -12V)

3.2. Coulomb friction

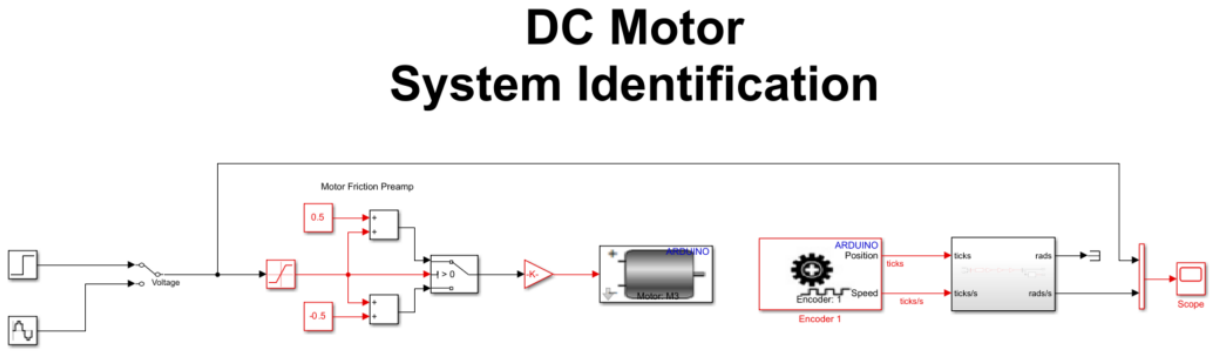
We will begin our analysis by identifying the Coulomb friction within the system. Coulomb friction, often referred to as 'dry friction,' is a constant magnitude friction that opposes motion and is independent of the relative velocity between surfaces. In the context of a brushed DC motor, this phenomenon manifests as the Dead-Zone. The Dead-Zone is a region in which the motor doesn't produce any movement even though a voltage is applied. This is primarily because the applied voltage or torque is not sufficient to overcome the inherent static friction of the system. It's crucial to identify and compensate for this Dead-Zone to ensure smooth operation and accurate control of the motor, particularly during its startup and low-speed operations

- Using the same "HexaMotor_MotorTest.slx" Simulink model. Find the minimum voltage values required to start the motor motion. You can change the slider range if you wish to use it to find the minimal values.
  You can push the motor a bit with your hand and try to find the least resistance angle. (due to the non-linearity in the friction mentioned earlier).
- Write those values for next section.

3.3. Step response modeling

In this section we will use the bump test to estimate the system`s transfer function. We will use in two methods, one with initial voltage 0, and the other with initial voltage of 2 volts.

- Open the "HexaMotor_System_Identification.slx" Simulink model. Make sure the switch is set to step input.
- Update the Motor Friction Preamp gains with the values found in the previous section.
- Adjust the step values to step at 1 second, from initial 0-volt to 7-volt step, update the Stop time to 5 seconds and record the response.
- Run the experiment again this time with a step from initial 2-volt to 9-volt step. Record the response.



For the Report:

a. Find the transfer function from both experiments recording using the first order approximation. If the difference is more than 10% check your calculations. Remember that the step size is 7 volts. And in the second experiment the initial angular velocity isn`t zero. Consider in which cases it is preferred to perform one or the other.

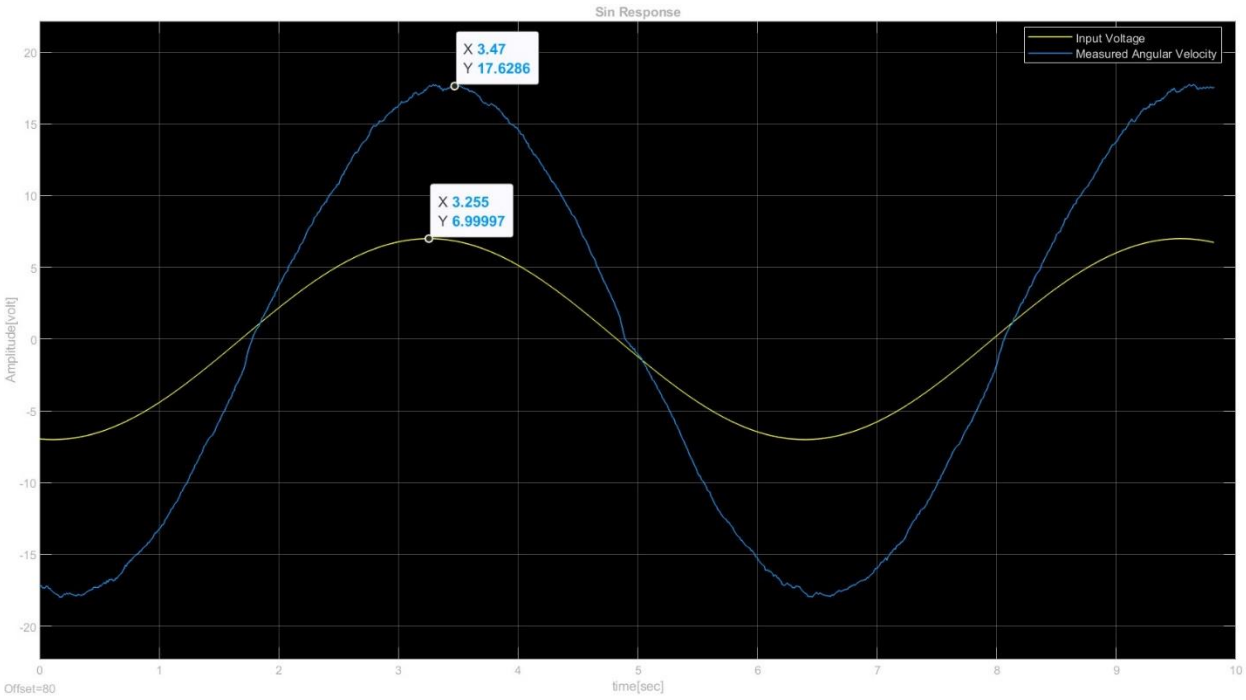
b. Create a plot in Matlab which contains a step response from 0 volts to 7 volts from the recordings and as a simulation using the step function in Matlab for the identified transfer function.
Hint: Use the following commands to simulate a step response similar to the experiment. Don't forget to include title. Legend etc. to the plot.

```
plot(ScopeData.time,ScopeData.signals.values)
hold on
plot(ScopeData.time+1,step(T,ScopeData.time)*7)
```

3.4. Frequency response modeling

In this section we will generate a bode plot for the system and derive the transfer function using its frequency response to several input frequencies.

- Using the same "HexaMotor_System_Identification.slx" Simulink model. Move the switch to a sin input. And update the stop time to inf.
- Update the parameters of the input Amplitude:7 Frequency (rad/sec): 1 rad/sec and run the model, notice the system response.
- Measure the time delay and magnitude of the response.



For the Report:

a. Fill up the following table make sure to record at list 7 frequencies at the range of [0..30] rad/sec
Hint: you can change the frequencies while the model is running and print to figure while it runs.
Gain $= \frac{Max\ velocity}{Input\ Amplitude}$, $\omega = 0$ can be calculated from the step response.

Frequency (Rad/sec)	Input Amplitude (V)	Max velocity (rad/s)	Time Delay (s)	Gain (rad/s/v)	Gain(dB).	Phase (degree)
0						
…..						
30						

b. Approximate the ω_c by evaluating the frequency at which the amplitude $|G(\omega_c)| = \frac{1}{\sqrt{2}}|G(0)|$. store the response for that frequency and calculate the time constant τ. How close it is to the τ calculated from the step response? Which method will give a more precise result?
Hint: Pre lab question 2.2 c.

c. Plot a bode plot using the data from the table.
Hint: use the following Matlab commands:

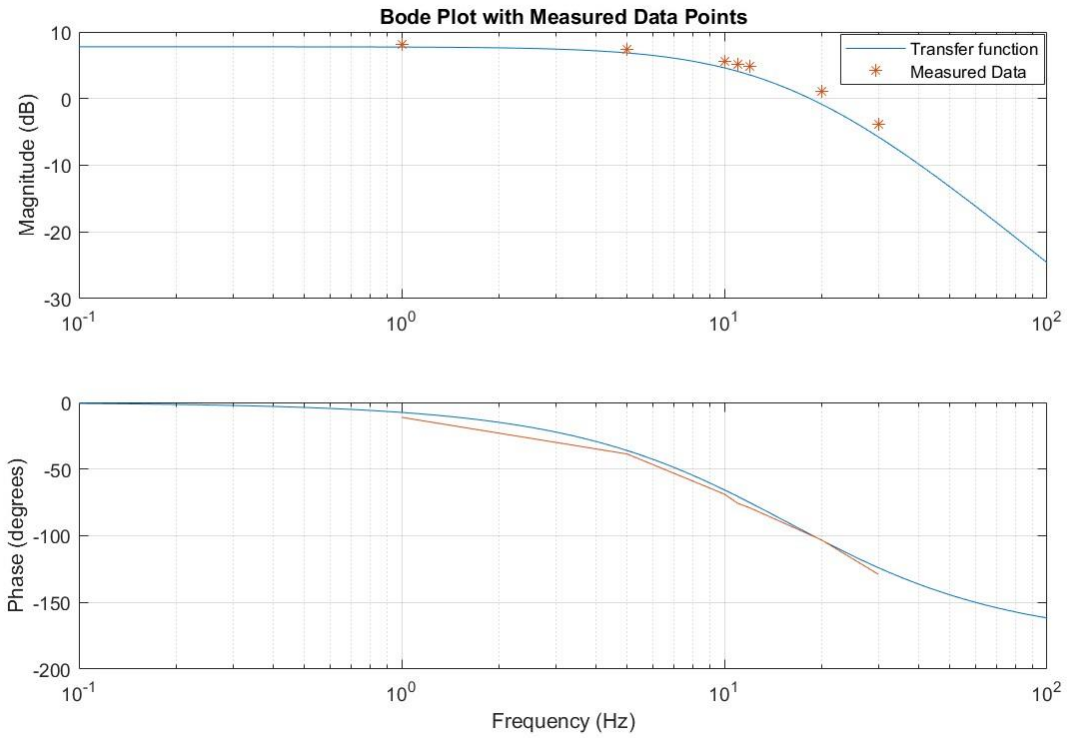
```
% Create a frequency response data object
% units: amplitudes [rad/s/v] ,  frequencies [Rad/sec]
freq_data = idfrd(amplitudes, frequencies, 0);
% Estimate transfer function (change '1' to the desired order of the system)
sys = tfest(freq_data, 1)

% plot bode like plot, with the added points
s=tf('s');
w = logspace(-1,2,100);
[mag,phase] = bode(sys*(1/(0.05*s+1)),w); % added a low pass filter present

figure(1);
subplot(2,1,1)
semilogx(w,20*log10(squeeze(mag)))
hold on
semilogx(frequencies, 20*log10(amplitudes), '*');

subplot(2,1,2)
semilogx(w,squeeze(phase))
hold on
semilogx(frequencies,phases)
```

Example Plot



Note: The resulting plot contains a phase up to -180 degrees due to the presence of a low pass filter for the velocity measurements at $\tau = 0.05$

d. How close the transfer function derived from the bode estimation to the one found from the step response. What do you think, which one is more precise? which was faster to achieve?

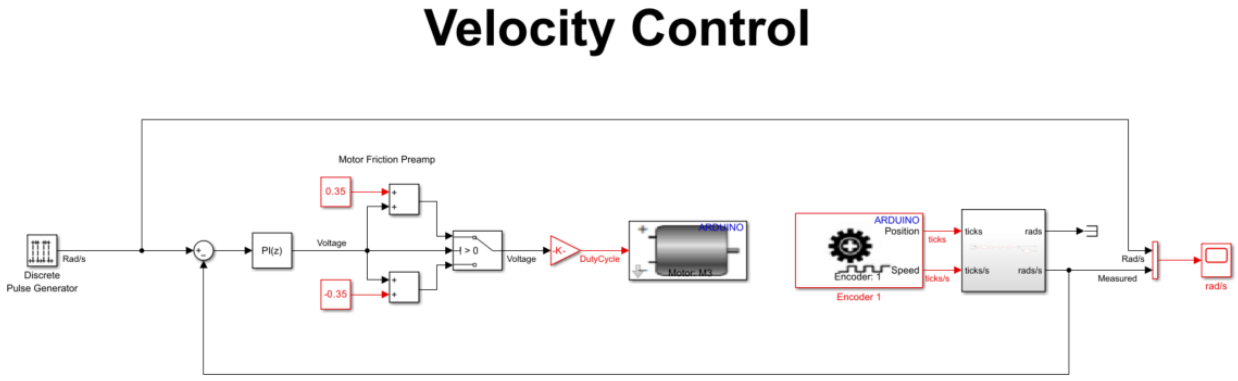
3.5. PID controller

In velocity control setups for motors, a PI (Proportional-Integral) controller is commonly used. While the PID controller includes all three terms — Proportional, Integral, and Derivative — the PI controller only uses the Proportional and Integral terms. The reason for excluding the Derivative term in velocity control is due to the nature of the application. In many motor control scenarios, the Derivative term can introduce noise or lead to excessive sensitivity to rapid changes. Given the objectives of velocity control, the PI controller strikes a balance between performance and stability, delivering smooth and consistent motor speed without the potential drawbacks of the Derivative term.

3.5.1. Unity feedback

We will start by measuring the steady state error from a unity feedback and comparing the results to a simulated system based on the model found in previous sections.

- Open the "HexaMotor_Velocity_Control.slx" Simulink model. Make sure the switch is set to step input.
- Update the Motor Friction Preamp gains with the values found in the first section.
- Adjust the square wave Pulse Generator a period of 4 seconds pulse width of 2 seconds and delay of 1 second, set the amplitude to 15 rad/s and update the Stop time of the simulation to 7 seconds.
- Update the PI controller to proportional gain Kp=1 and integral gain Ki=0
- Run the experiment.



For the Report:

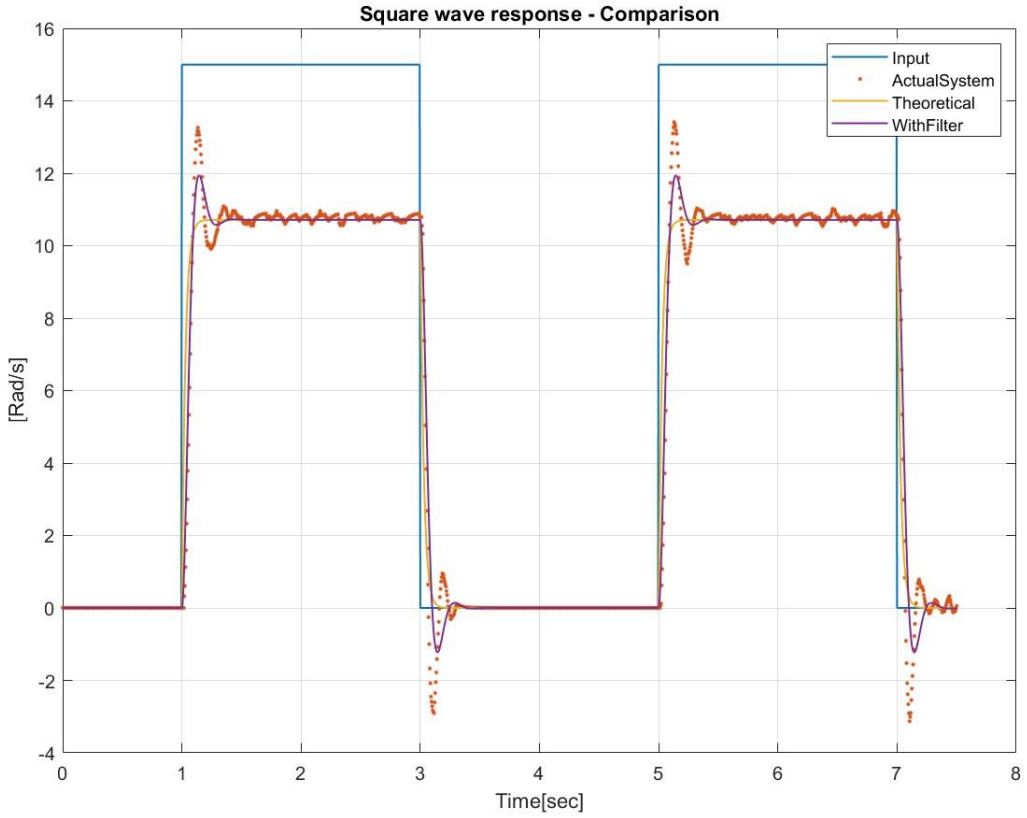
a. Compare the steady state error to the theoretical one

b. Using the recording of the experiments create a plot in Matlab containing the response from the experiment and simulated results from the plant found in section 3.4. You may notice that the responses aren`t exactly the same, that is due to non-linearity's in the actual setup and to the presence of the low pass filter for the velocity measurement. You may also add simulated results including the low pass filter for comparison. ($LPF(s) = \frac{1}{0.05s+1}$).

Hint: use the following Matlab commands: (do not forget to update the transfer function and add grid, axis labels, title to the plot).

% Generate plots from recordings and simulated transfer function

```
s=tf('s');
G = 2.5/(0.1*s+1);
LPF = 1/(0.05*s+1);
figure(1)
plot(ScopeData.time,ScopeData.signals.values(:,1))
hold on
plot(ScopeData.time,ScopeData.signals.values(:,2),'.');
[y,t] = lsim(G/(1+G),ScopeData.signals.values(:,1),ScopeData.time);
plot(t,y);
[y,t] = lsim(G*LPF/(1+G*LPF),ScopeData.signals.values(:,1),ScopeData.time);
plot(t,y);
```



3.5.2. PI Controller manual tuning

In this section, we will discuss how to tune the parameters of a PI (Proportional-Integral) controller to achieve a desired response.

- Begin by setting the integral gain to 0 and adjust the proportional gain within the range of [0 to 2] until the response exhibits a slight overshoot or reaches the desired rise time.
- Introduce the integral gain to minimize the steady-state error and ensure the response meets the desired peak time, overshoot, and settling requirements. If needed, decrease the proportional gain.
- Fine-tune both the Proportional and Integral gains as required to fulfill the design specifications.

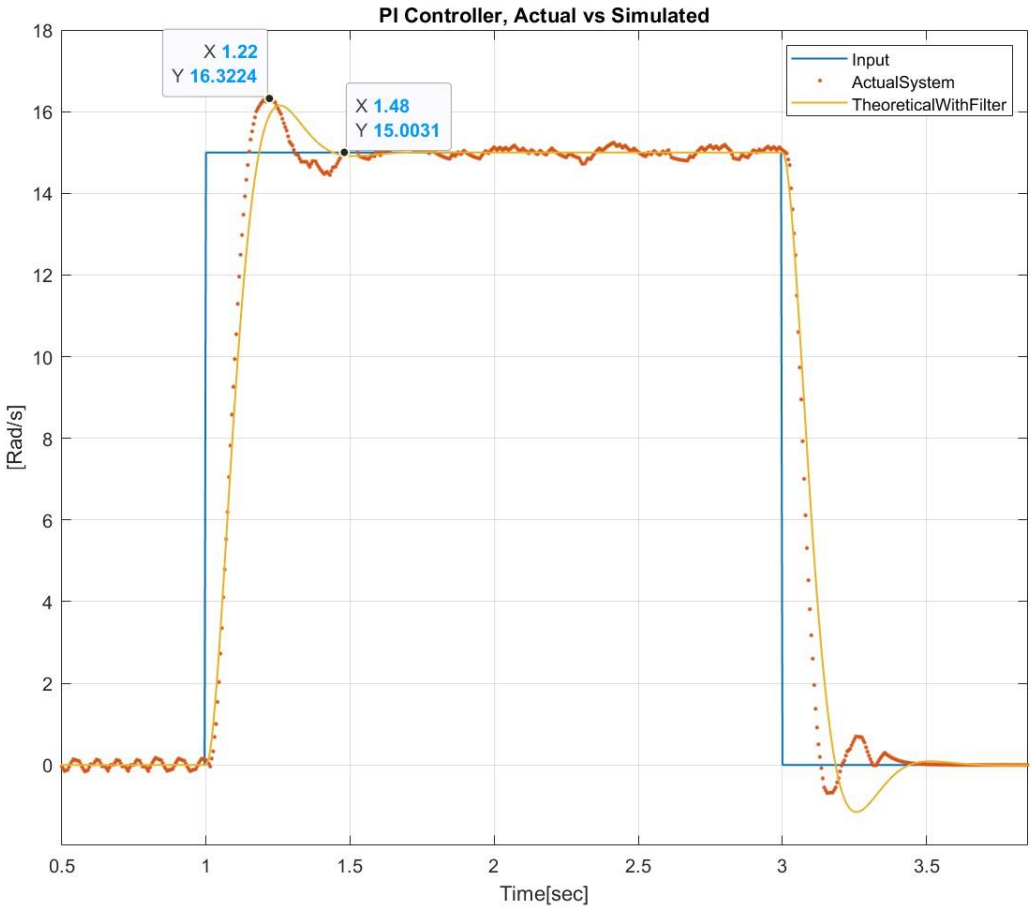
Adjust the parameter to get the following control specifications:

$0.3s > \mathrm{T}_{PeakTime} > 0.15s\,,\ \mathrm{T}_{Settling} < 0.5s\,,\ Overshoot < 15\%$

Hint: you can set the Stop time of the simulation to 'inf' while tunning the parameters.

For the Report:

a. Plot the response of the system (optional with the simulated theoretical model). Mark the important criteria and calculate overshoot of the system.



Note: The resulting simulated plot differs slightly from the actual response. This sifference arises because the simulation does not account for all the non-linearities of the actual setup. Additionally, we approximated the system as a first-order system, but ideally, it should have been estimated as a second-order system due to the presence of the low-pass filter, whose pole is not significantly distant from the motor's pole.

4. Appendix

Github Page: https://github.com/TALs-Education/HexaMotor

4.1. DC Motor

This gearmotor consists of a low-power, 12 V brushed DC motor combined with a 20.4:1 metal spur gearbox, and it has an integrated 48 CPR quadrature encoder on the motor shaft, which provides 979.62 counts per revolution of the gearbox's output shaft.

Product page: https://www.pololu.com/product/4883

Data Sheet: https://www.pololu.com/file/0J1829/pololu-25d-metal-gearmotors.pdf

4.2. Arduino boards

The MKR Motor Carrier is an MKR add-on board designed to control servo, DC, and stepper motors. The Carrier can also be used to connect other actuators and sensors via a series of 3-pin male headers.

MKR Zero: https://docs.arduino.cc/hardware/mkr-zero

MKR Motor Carrier: https://store-usa.arduino.cc/products/arduino-mkr-motor-carrier

4.3. Simulink Support package

With MATLAB and Simulink Support Packages for Arduino® hardware, you can use MATLAB and Simulink to interactively communicate with your Arduino. Simulink also enables you to perform model deployment for standalone operations on Arduino boards.

MathWorks: https://www.mathworks.com/hardware-support/arduino.html

4.4. 3D Model – Fusion360

The setup is design in Fusion360 as an open source project, for educational purposes. Available at:

Fusion360: https://a360.co/3TU1BYP