The Iby and Aladar Fleischman Faculty of Engineering

Tel Aviv University

Position Control

HexaMotor

Experiment objectives:

• Introduction to second order systems.

• Coulomb Friction identification and handling.

• System identification using position close loop step reponse.

• System identification using position close loop frequency reponse.

• Position control implementation based on second order system modeling.

• Lead-Lag compensator controller design and implementation.

• Force FeedForward example - compliant motor control.

1. Introduction

In the field of motor control, mastering both velocity and position control is essential for a wide range of applications. While the previous lab experiment delved into the nuances of velocity control and the fundamentals of a DC motor, this experiment pivots to emphasize position control. Position control, vital for accurate maneuvering and precise positioning of mechanisms, demands its own unique approach. This experiment outlines objectives ranging from system identification using step and frequency responses to the design and actual implementation of control strategies.

1.1. First order approximation

While in the previous experiment the full equations of the DC motor were presented and the first-order approximation was derived, in this experiment we will contrinue working with the first-order approximation of the DC motor.

First-Order Transfer Function:

1.1 $$G(s) = \frac{K_T}{(RJ)s+(BR+K_TK_b)} \;\; ; G(s) = \frac{K}{\tau s+1}$$

1.2. Coulomb friction

Recall from the previous experiment that Coulomb friction, which remains constant irrespective of motor speed, contributes to system non-linearity and can pose challenges in control implementation, particularly at low speeds. We identified this friction by applying a controlled voltage to the DC motor, observing the threshold voltage initiating rotation.

1.3. Second order systems

When discussing control systems, a second order system is frequently encountered. It can be described by the standard second-order transfer function:

1.2 $$T(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$$

Where:

K – is the system gain.

ω_n – is the undamped natural frequency.

ζ– is the damping ratio.

The system's response is primarily characterized by its damping ratio ζ and natural frequency ω_n

Several performance metrics can be derived from the above standard form:

Percent Overshoot (%OS):

It defines the amount by which the transient response exceeds the steady-state value. For a step response, the percent overshoot is given by: $OS = 100e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}$ It should be noted that overshoot occurs only when ζ<1 (underdamped case).

Rise Time (Tr):

The time it takes for the response to rise from 10% to 90% of its final value for an underdamped system. While there's no direct closed-form formula, for many systems with a damping ratio between 0.4 and 0.7, the rise time is approximately: $T_r \approx \frac{1.8}{\omega_n}$

Settling Time (Ts):

The time for the system response to stay within a certain percentage (usually 2% or 5%) of its final value. It's given by $T_s \approx \frac{4}{\zeta\omega_n}$ for a 2% settling time.

Peak Time (Tp):

The time it takes for the response to reach its first peak. For an underdamped system: $T_p = \frac{\pi}{\omega_d}$ where $\omega_d = \omega_n\sqrt{1-\zeta^2}$ is the damped frequency.

Natural Frequency (ω_n):

It's the frequency at which the system would oscillate if there were no damping (i.e., ζ=0).

1.4. PID controller

As previously discussed in our prior experimentation, the PID controller is a predominant feedback control algorithm extensively employed in industrial applications due to its efficiacy across diverse scenarios and its systematic tuning methodology.

1.3 $$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{\mathrm{d}e(t)}{\mathrm{dt}}$$

The PID controller is characterized by three fundamental components:

Proportional (P): This component yields a control action that is directly proportional to the instantaneous error. The magnitude of this response is governed by the proportional gain

Integral (I): Concerned with the aggregation of error over time, this action aims to nullify any persistent steady-state error.

Derivative (D): This term offers a control action based on the rate of error change, effectively anticipating potential future errors and generating a preemptive control response.

1.5. Lead-Lag Compensator

Lead and lag compensators are essential tools in control theory, particularly in the design of feedback control systems to meet transient and steady-state response specifications:

Lag: The general transfer function for a lag compensator can be described as:

1.4 $$G_{Lag} = \frac{1+\tau s}{1+a\tau s}$$

Where τ is the time constant and a>1.

Purpose: The primary purpose of a lag compensator is to improve the steady-state response of the system. This is done by adding a pole and a zero to the system. Since the zero is closer to the origin than the pole, the compensator will have a magnitude less than one for most frequencies, thereby reducing the gain of the system and the steady-state error.

Placement: The zero is placed to the left of the pole. The distance between them determines the magnitude of the phase boost, but a lag compensator will not introduce a significant phase lead.

Drawbacks: Lag compensators generally degrade the transient response because they tend to reduce the system bandwidth.

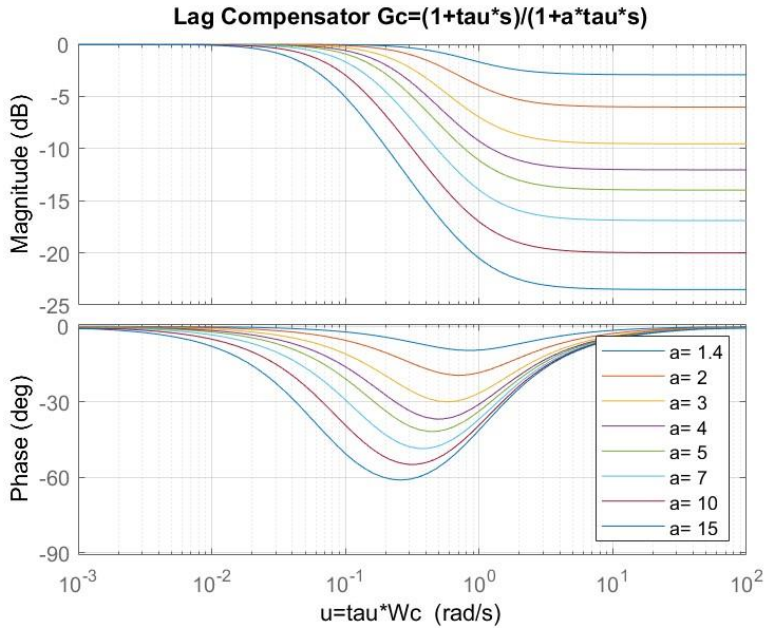

Figure 1 Lag Compensator plot using LeadLad.m scrip

Lead Compensator: The general transfer function for a lead compensator can be described as:

1.5 $$G_{Lead} = \frac{1+a\tau s}{1+\tau s}$$

Where τ is the time constant and a>1

Purpose: The primary purpose of a lead compensator is to improve the transient response of the system. By placing a pole and a zero in the left half plane, it introduces phase lead to the system, which can be used to reshape the root locus or push the closed-loop poles to desired locations.

Placement: The zero is placed to the right of the pole, introducing phase lead in the frequency range of interest. The amount of phase lead is determined by the distance between the pole and zero and their proximity to the imaginary axis.

Drawbacks: Because of the introduced phase lead, a lead compensator also increases the gain in some frequency range. This can make the steady-state error worse.

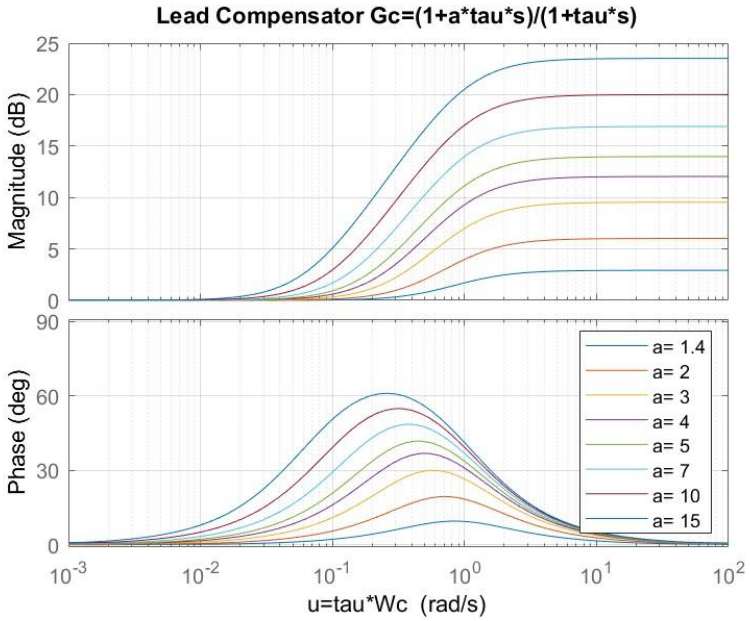

Figure 2 Lead Compensator plot using LeadLad.m scrip

Design Considerations:

Lag: When using a lag compensator, the goal is often to place the pole and zero sufficiently far from the imaginary axis, so they don't affect the transient response. You place the zero so that the magnitude of the system is unchanged at the crossover frequency.

Lead: When designing with a lead compensator, the goal is often to add enough phase lead to achieve desired transient characteristics without pushing the gain crossover frequency too high.

In practice, combining both lead and lag compensators can offer an optimal balance between steady-state and transient performance in many systems.

Graphical approach for Lead Lag design:

Run the "LeadLag.m"

- The script "LeadLag.m" generates plots associated with the lead-lag compensator design. This includes various curves representing different amounts of phase lead and the corresponding gains for different values of a (since a lead compensator's behavior is determined by its zero and pole locations, which are in turn determined by the parameter a and the time constant τ).

Open-Loop Bode Plot:

- Start by plotting the open-loop Bode plot of the system. From this plot, you'll identify the current crossover frequency ω_c and the existing phase margin.

Determine Required Phase Boost:

- Based on system specifications or desired closed-loop performance, determine how much additional phase you want to introduce.

Iterative Process for Design:

- Use the provided lead graph (generated by the script) to identify a point that provides the necessary phase boost.
- From the corresponding gain graph, note the gain added due to the lead compensator.
- This added gain will shift ω_c in the open-loop Bode plot.
- Now, with this new ω_c the phase requirements might be slightly different. So, you may need to iterate this process to ensure that the final selected values of phase and gain boosts provide the desired performance.

Calculate the Lead Compensator:

- Using the relationship and the equation provided: $G_{Lead}(s) = \frac{1+a\tau s}{1+\tau s}$
- And given: $\tau = \frac{u}{\omega_c}$ Where:
- τ is the time constant of the lead compensator.
- a is the gain boost and defines the curve.
- u is the specific point on the curve/graph which dictates the amount of phase lead and gain boost introduced.
- ω_c is the new crossover frequency after introducing the compensator.

Implement and Test:

- Once the lead compensator parameters are determined, you can then apply them to the system and validate if the system's response matches your design specifications. You may also need to iterate the design process if the initial compensator doesn't provide the desired results.

This process essentially allows you to design a lead compensator using graphical methods, taking advantage of the insight provided by Bode plots and the relationships between phase lead, gain boost, and compensator parameters.

2. Pre Lab

2.1.Second order system identification

Identifying the First Order Transfer Function of a DC Motor Using Closed-Loop Step Response in Position Feedback

In this method, the behavior of the DC motor under position feedback is contrasted with a canonical second-order system, described by $T(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$. The benefits of this comparison are twofold. Firstly, second-order systems are thoroughly understood and have been extensively studied, offering a benchmark against which the motor's dynamics can be evaluated. Secondly, essential performance metrics — including overshoot, peak time, rise time, and settling time — can be directly correlated to the system's damping ratio, ζ, and its natural frequency, ω_n.

Armed with these metrics from experimental observations and a known feedback gain, it's possible to determine the inherent characteristics of the DC motor, especially its time constant, τ, and gain, k_m.

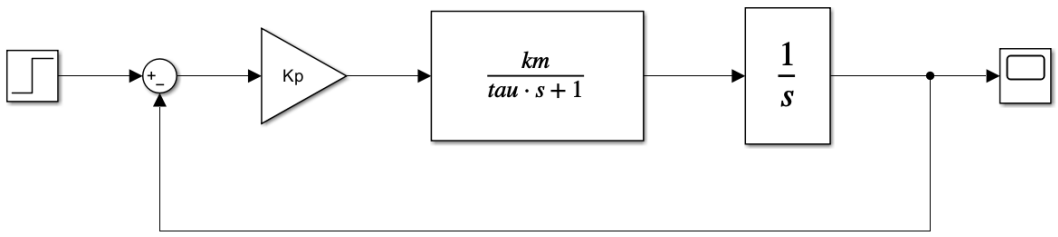

Figure 3 Simulink model representing close loop system

a. Given the system above, assuming the plant is a DC motor using its first order approximation $G(s) = \frac{k_m}{\tau s+1}$, write what the blocks represent and the units of the input and output signals.
b. Given the system above, what is the close loop transfer function:
c. Compare the transfer function derived in b. To the standard second order transfer function of the form $T(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$. Find representations of ζ and ω_n as a function of τ , k_m and *kp*
d. Using Matlab, plot the step response of the following first order system $G(s) = \frac{2.5}{0.1s+1}$, using the system represented above with gain of *kp*=10.
e. Based on the peak time and over-shoot performance metrices for a standard second order system. Find the values of ζ and ω_n.
f. Based on the equations derived in C verify the transfer function of the DC motor in the form of $G(s) = \frac{k_m}{\tau s+1}$

2.2. PD controller

Tuning a PD Controller for a DC Motor Using Canonical Second-Order Transfer Function Analysis

In this approach, the behavior of the DC motor, when coupled with a PD controller, is compared with a canonical second-order system described by $T(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$. Such a comparison offers two primary advantages. Firstly, the canonical second-order systems serve as a well-defined benchmark, given their extensive study and well-understood dynamics. Secondly, by mapping desired system response characteristics, such as overshoot (OS) and peak time, to the parameters of this second-order system, one can directly relate them to the controller gains, Kp and Kd.

Through this method, it becomes possible not only to tune the PD controller for a specific DC motor but also to have a systematic approach to achieving desired dynamic characteristics. By tailoring Kp and Kd based on the insights from the second-order comparison, one can design a controller that meets specific performance metrics, paving the way for optimal system operation.

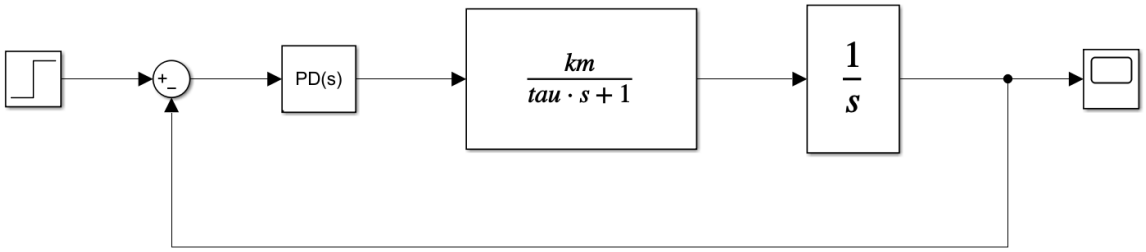

Figure 4 Simulink model representing close loop system

a. Given the system above, assuming the plant is a DC motor using its first order approximation $G(s) = \frac{k_m}{\tau s+1}$, and $PD(s)$ (is a proportional derevative controller) of the form: $PD(s) = k_p + k_d s$. What would be the close loop transfer function $T_1(s)$.
b. Compare the poles of the transfer function $T_1(s)$ to the poles of the standard second order system represented as $T_2(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$. And find representation of ζ and ω_n as a function of k_m, τ, k_p and k_d.
c. Find the values of ζ and ω_n which will result in a step response for the standard second order system $T_2(s) = \frac{K}{s^2+2\zeta\omega_n+\omega_n^2)}$ with the following criteria: %OS of 25% and $T_p = 0.20$ $[seconds]$.
d. Based on the equations found in b. And assuming the motor has a transfer function of $G(s) = \frac{2.5}{0.1s+1}$ find the values for k_p and k_d.
e. Simulate the system at a. (in matlab or simulink). using the found controller gains and dc motor transfer function from d. What can you say about its %OS, T_p ? why is it slightly different from the designed OS%, T_p. Plot the step response of the system and mark OS%, T_p.

2.3. Lead-Lag compensator

Graphical Approach to Lead-Lag Compensator Design for DC Motor Performance Enhancement Using Bode Plots

By graphically analyzing the Bode plots, one can visually discern the system's frequency response shortcomings. The "lead" in the compensator can be tailored to boost phase margin and speed up system response, while the "lag" part can be optimized to improve steady-state accuracy. Through a methodical graphical approach, not only is it possible to pinpoint where compensation is needed but also to sketch the compensator's frequency response right on the Bode plot, ensuring an intuitive and interactive design process.

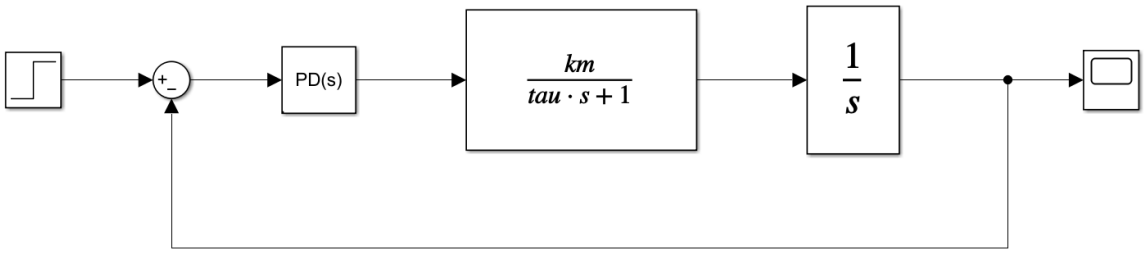

Figure 5 Simulink model representing close loop system

a. Given the system above, and using the values from the previous section, plot in matlab the open loop bode of the system and evaluate the ω_c (crossover frequency) and phase margin of the system.
b. Design a lead lag compensator to fix the Phase margin to $60°$. Add to the bode plot in a. the compesated bode and mark the new ω_c , Phase margin. Plot the step responses from both of the systems on the same graph. Mark the OS%, T_p. Of both. (use the graphical method or any other method of your choice).

3. In Lab

3.1. Introduction

Welcome back to the lab! Before diving into our experiments, please take a moment to reacquaint yourself with our hardware. You'll be using the HexaMotor setup, which features a DC motor paired with an encoder and linked to an Arduino MKR motor controller. You can operate it through Matlab Simulink. For more details, refer to 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.** (same configuration as in the velocity control experiment)
- 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.

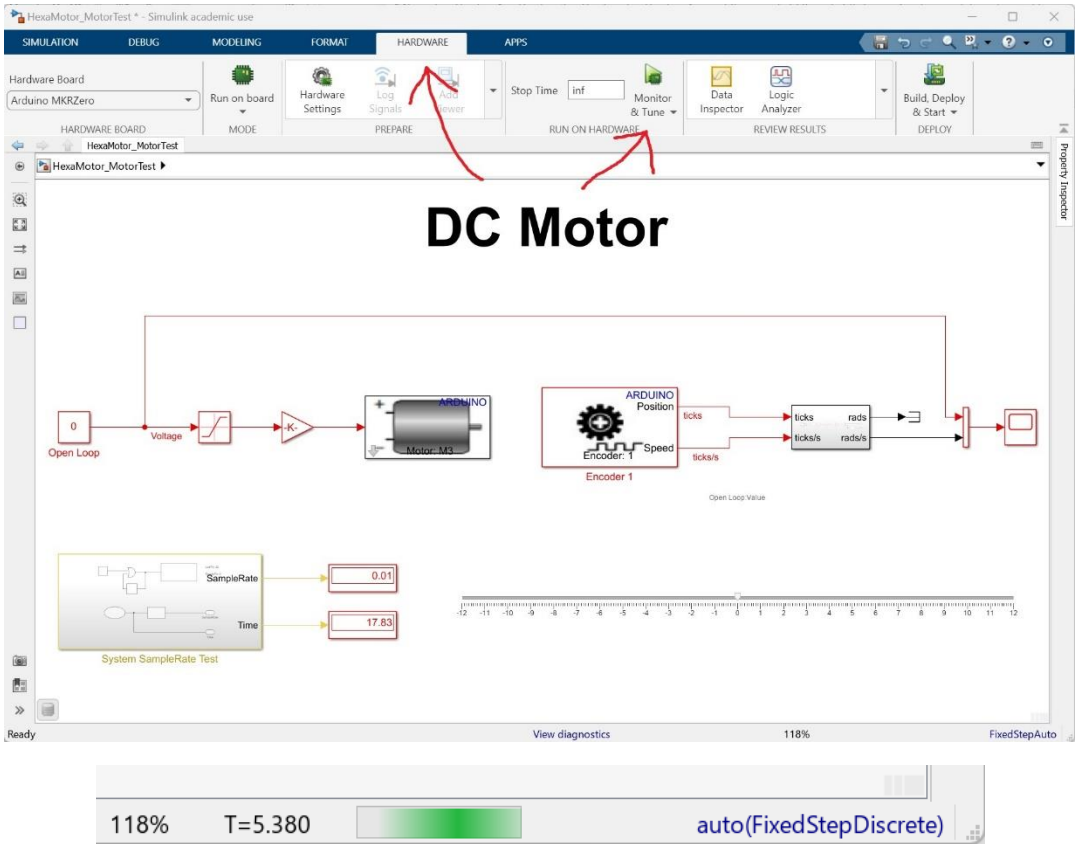

Figure 6 Simulink model for interaction with the HexaMotor Setup

3.2. Revisiting Coulomb Friction

Let's delve deeper into the system by reexamining the Coulomb friction present. Often termed 'dry friction', Coulomb friction consistently opposes motion and remains unaffected by the varying velocity between surfaces. Within a brushed DC motor's framework, this is perceived as the Dead-Zone. This Dead-Zone denotes the area where, despite voltage being administered, the motor remains static. This immobility arises primarily when the exerted voltage or torque falls short of conquering the system's innate static friction. Pinpointing and rectifying this Dead-Zone is pivotal for the motor's seamless operation, especially during its initial phases and at slower speeds.

- 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

Let's now transition to our next experiment: exploring the first order transfer function of a DC motor using the closed-loop step response in position feedback.

- Open the "HexaMotor_Position_Control.slx" Simulink model. Make sure the switch is set to Pulse Generator input.
- Update the Motor Friction Preamp gains with the values found in the previous section.
- Adjust the PD controller to proportional only with *kp*=10 and update stop time to 30 seconds.
- Run the experiment

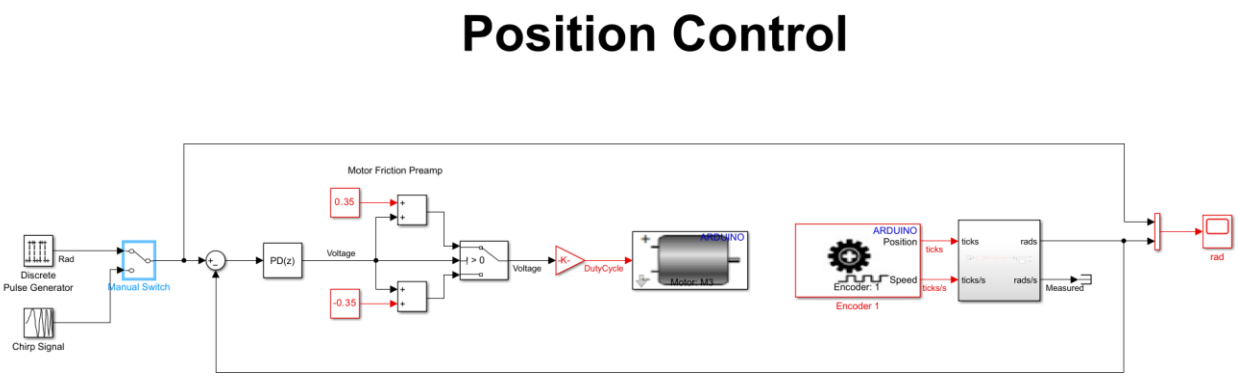

Figure 7 Simulink model : "HexaMotor_Position_Control.slx"

For the Report:

a. Using the closed-loop step response, estimate the DC motor's transfer function following the method previously described and practiced in the pre-lab section.

b. Create a plot in Matlab which contains the response from the recordings and as a simulation using the lsim function in Matlab for the identified transfer function. Mark peak time and over-shoot values for both signals.

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');
Kp = 10;
G = 2.5/(0.1*s+1)/s; % update the transfer function values
figure(1)
plot(ScopeData.time,ScopeData.signals.values(:,1))
hold on
plot(ScopeData.time,ScopeData.signals.values(:,2),'.');
[y,t] = lsim(Kp*G/(1+Kp*G),ScopeData.signals.values(:,1),ScopeData.time);
plot(t,y);
hold off
```

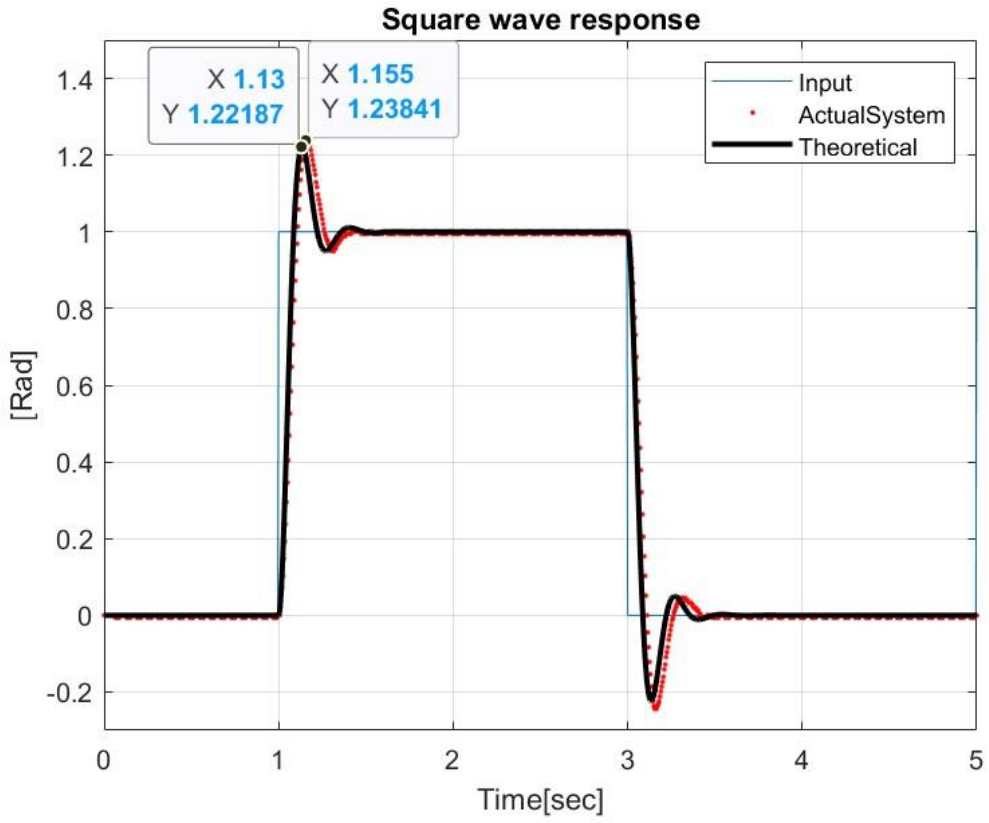

Figure 8 Square wave response, actual system and simulated model

3.4. Frequency response modeling

In this section, we'll utilize a frequency response from a chirp signal to determine the second order system's transfer function. From this, we can then derive the first order transfer function of the DC motor, employing the same approach as outlined in the preceding section.

- Using the same "HexaMotor_Position_Control.slx" Simulink model. Move the switch to a chirp input.
- Run the experiment.

For the Report:

a. Utilize the closed-loop chirp response to estimate the second order system's transfer function using the 'tfest' function in MATLAB. Then, estimate the DC motor's transfer function based on the approach we previously outlined and practiced in the pre-lab section.
Hint: use the following Matlab commands:
grid, axis labels, title to the plot).

```
% Estimate transfer function
s=tf('s');
data = iddata(ScopeData.signals.values(:,2),ScopeData.signals.values(:,1),0.005);
T=tfest(data,2,0)
```

b. Create a plot in Matlab which contains the response from the recordings and as a simulation using the lsim function in Matlab for the identified transfer function.

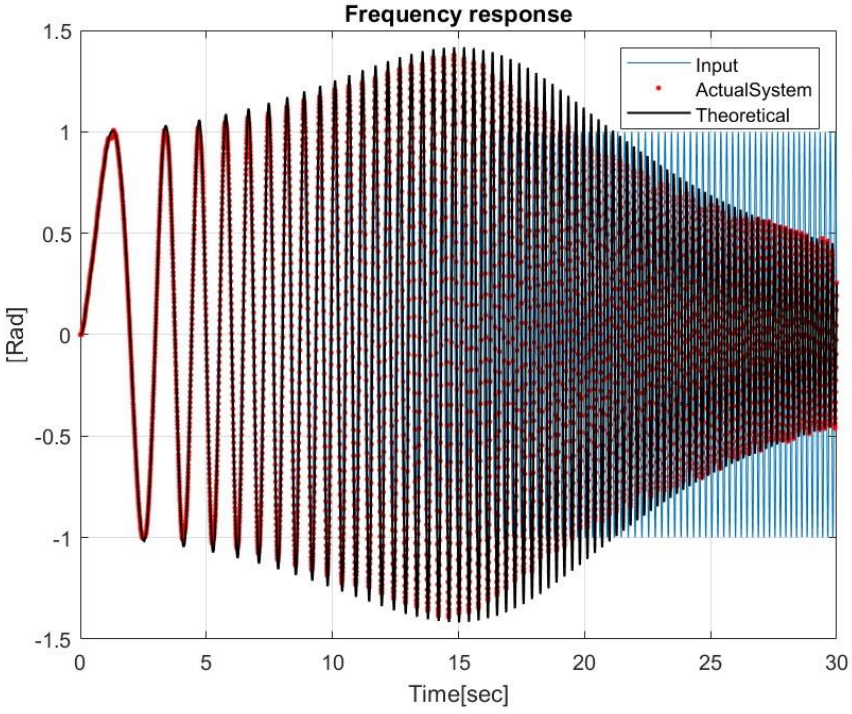

Figure 9 Frequency response, actual system and simulated model

3.5. PD controller

In this section, drawing on the approach used in the pre-lab section and based on the transfer function previously determined using the frequency response, we will harness the characteristics of a canonical second-order transfer function. Our objective is to design a PD controller tailored to meet specific criteria: an overshoot (OS) of 25% and a peak time (Tp) of 0.15 seconds.

- Using the same "HexaMotor_Position_Control.slx" Simulink model. Move the switch to Pulse Generator input.
- using the approach from the pre lab section calculate the Kp and Kd. Of the PD controller and update its values in simulink, (Use the motor transfer function found using the frequency response system identifcation method).
- Run the experiment.

For the Report:

a. Create a plot in Matlab which contains the response from the recordings and as a simulation using the lsim function in Matlab for the identified transfer function and the calculated PD controller. Mark peak time and over-shoot values for both signals.

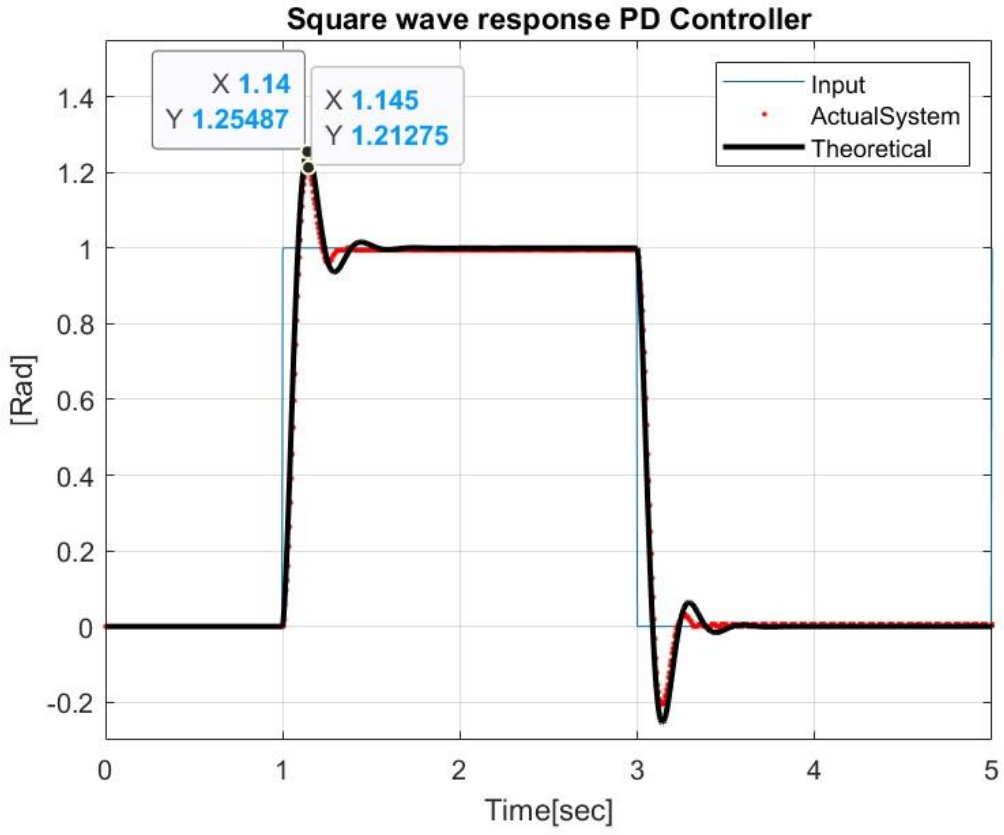

Figure 10 Square wave response with a PD controller, actual system and simulated model

3.6. Lead-Lag Compensator

In this section, we'll introduce a lead-lag compensator to enhance the motor's response. Employing the approach adopted in the pre-lab section, our aim is to refine the system's phase margin to 60°.

- Open the "HexaMotor_LeadLag_Control.slx" Simulink model. Make sure the switch is set to Pulse Generator input.
- Using the approach from the pre lab section calculate the necessary Lead compensator and update its values in the simulink Lead-Lag block.
- Update the PD controller values to the ones found in the lab
- Run the experiment.

For the Report:

a. Present your calculations and illustrate the variations in the system's Bode plot.
b. Create a plot in Matlab which contains the response from the recordings and as a simulation using the lsim function in Matlab. Mark peak time and over-shoot values for both signals.

Bonus: Redo the experiments and record the response of the system with the Pulse generator signal for the following controllers:

- Proportional gain only *kp*=10,
- PD controller found in the lab.
- LeadLag Controller with the PD controller.

Plot on the same plot the response from the 3 recordings. Make your conclusion regarding the performance of the designed controller.

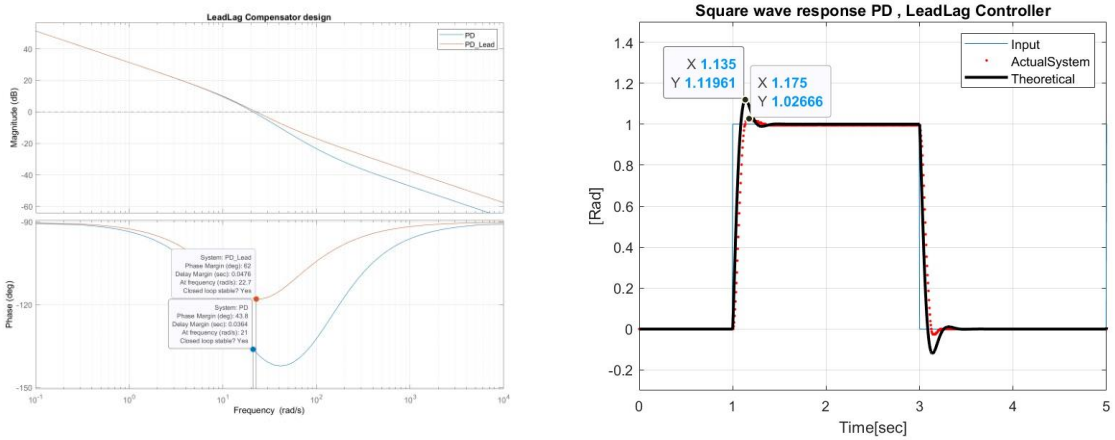

Figure 11 LeadLag design and system response

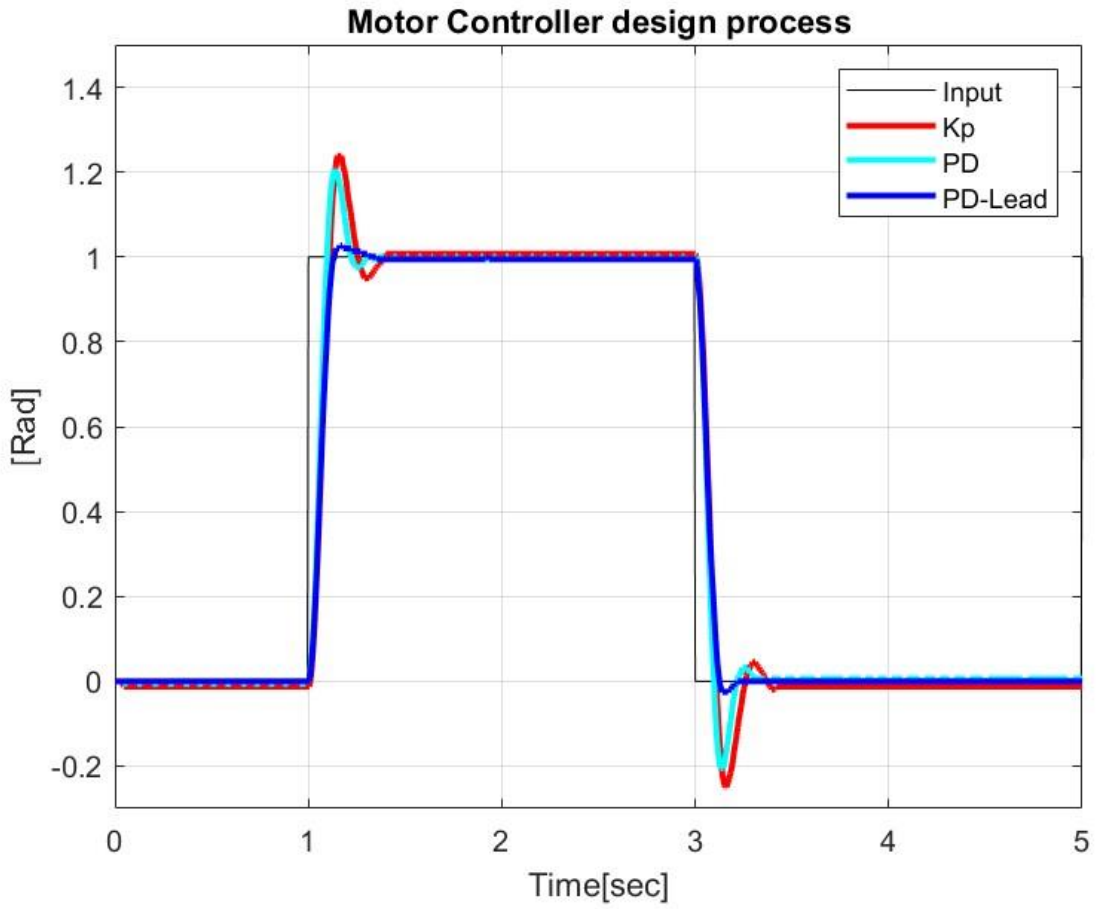

Figure 12 Controller design process, Square wave response of various controllers

3.7. Force FeedForward example - compliant motor control

In the realm of motor control, Force FeedForward - compliant motor control presents an innovative approach to system interaction. At the heart of this setup lies a DC motor integrated with position control feedback. But what sets this arrangement apart is its responsiveness to user interaction. When a user makes contact with the force sensors, the signal generated is instantly channeled to the motor voltage. This direct coupling not only ensures real-time response but also empowers the user with the ability to rotate the motor freely, creating a seamless and intuitive interaction between the user and the motor system. This approach promises enhanced user experience and broader applications in areas demanding delicate or human-centered motor control.

- Open the "HexaMotor_Force_FF.slx" Simulink model. Make sure the switch is set to Pulse Generator input.
- Replace the rotating mass with the force sensor rod, as in the image on the cover of this manual.
- Run the experiment.
- Switch the Force Sensors on and off during the experiment.
- Touch the force rod and feel the compliance of the motor when the force sensors are enabled.
- Adjust the gains for optimal performance. In the range of [10-50]

Force Feed Forward - Compliant Motor Control

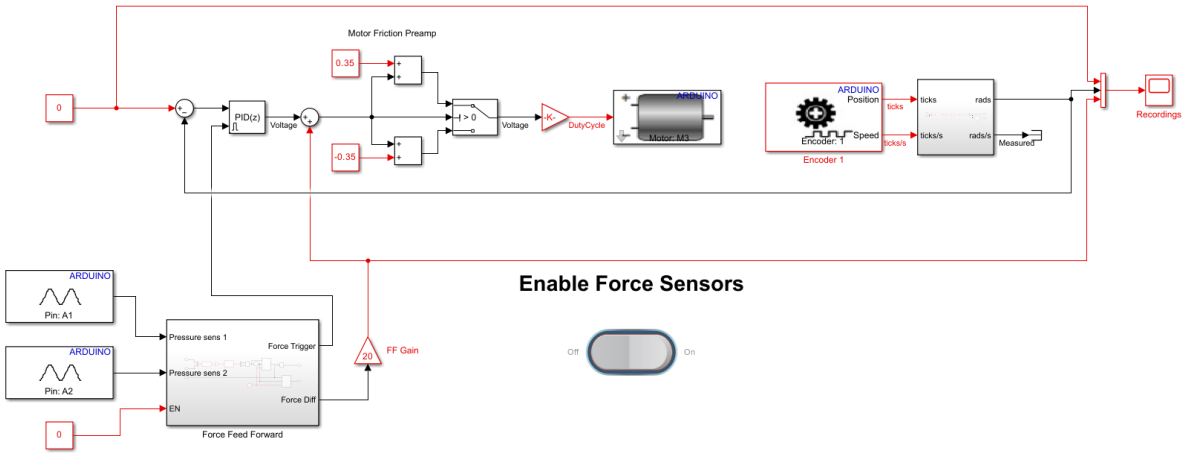

Figure 13 Simulink model "HexaMotor_Force_FF.slx"

Some points to consider about the setup:

You might observe that the controller has transitioned to a PID controller, which is more apt for stable positioning. This is due to the integral component's ability to accumulate and counteract external disturbances. However, when the force sensor is enabled and actuated, it overrides the integral component of the controller and produces a feedforward voltage proportional to its readings. It's worth noting that the force sensors are not calibrated and lack linearity, so their measurements are unitless.

Replace the force sensor rod with the rotating mass.

Turn off the system

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