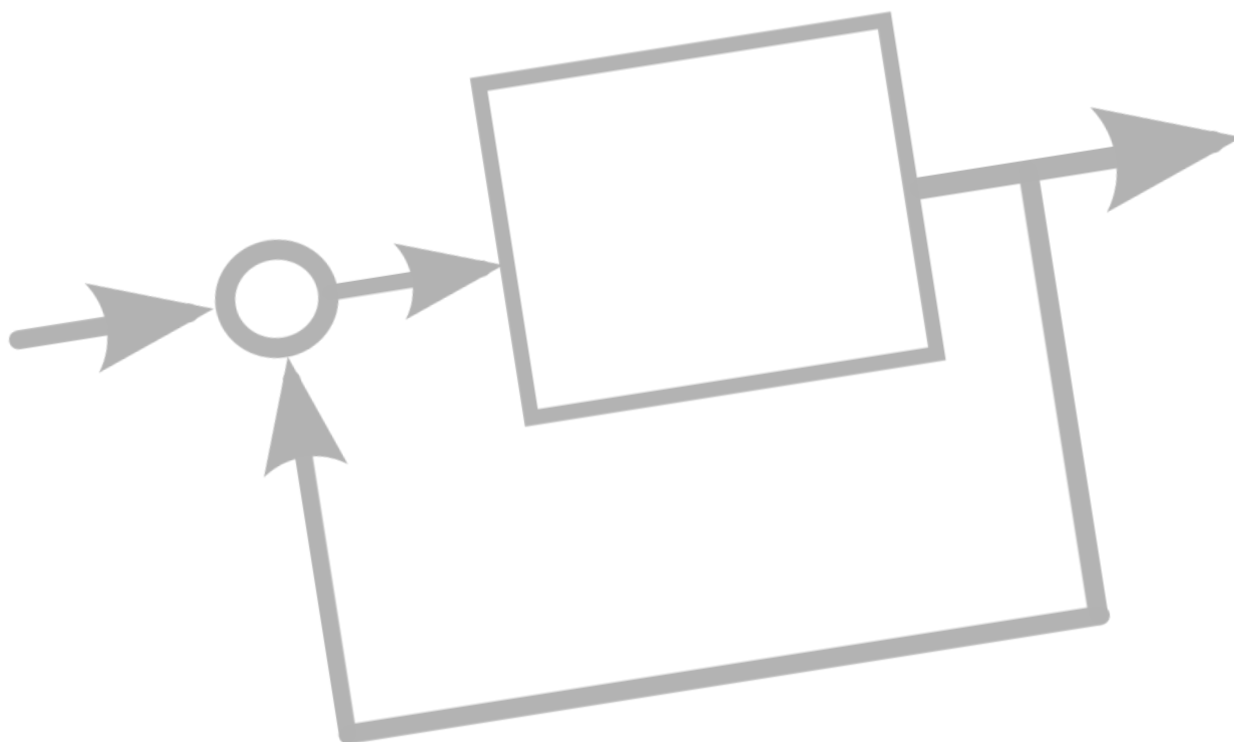




MIE404 – CONTROL SYSTEMS

LAB 3 – AIRFOIL POSITION CONTROL

2021



Pre-Reading: Please read this lab fully before attending!

Laboratory Objectives

The objective of this laboratory is to design and develop a hardware-in-the-loop position control strategy using for an airfoil in a small wind tunnel. This is meant to simulate a rudder on an aircraft. In this lab, you will gain a better understanding of the principles of system modeling, Proportional-Integral-Derivative (PID) Control, and controller design using structured tuning methods. In particular you will:

- Practice first principles of modeling
- Design a PID rudder angle controller
- Integrate the controller into a hardware loop

Introduction

Aircraft control surfaces are actuated to change aerodynamic characteristics and allow the aircraft to maneuver. The aircraft rudder is one of these control surfaces and is located in the tail section as shown below. The deflection of the rudder causes a side force, which creates a moment about the center of gravity and changes direction of the plane nose. This control surface is used in conjunction with the ailerons to change the direction of an aircraft.

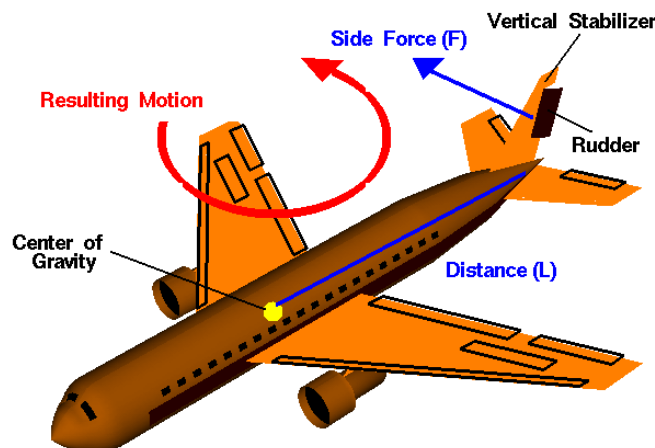


Figure 1: Impact of Rudder Deflection on Aircraft Dynamics (Credit NASA)

In these aircraft, the side force and turning moment is proportional to the deflection angle of the rudder. As such, precision control of this angle is critical for smooth turns and aircraft performance.

For this laboratory, you will examine control of a simulated rudder in a benchtop wind tunnel. To control the position, you will develop a PID controller to regulate the angle and the aerodynamic forces. The controller that you will design takes the rudder angle command and the actual rudder angle from an encoder signal. The PID controller then outputs the appropriate motor voltage, V_m , to actuate the motor which drives the rudder in the wind tunnel. The wind speed in the tunnel can vary and this variability acts as a disturbance which the controller must be able to reject. A block diagram of the overall system is shown in Figure 2.

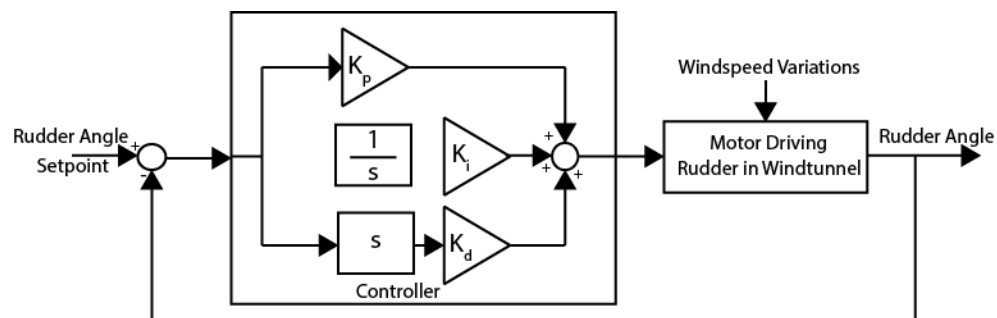


Figure 2: Rudder Control Loop

Background

THE PID CONTROLLER

The PID controller is one of the most common control algorithms. For position control, it combines the error reduction of proportional control, the steady state error reduction of integral control and the overshoot elimination of derivative control. The proportional control term tracks the instantaneous error, the integral control term integrates the error over time to increase the control signal to the level where the steady state error disappears, and the derivative term predicts the response of the system based on the slope of the response. This builds upon the controller developed in Lab 2 though the addition derivative control term to reduce overshoot and improve the settling time.

The linear behavior of a PID controller in the time-domain can be described by:

$$u(t) = k_p(r(t) - y(t)) + k_i \int_0^t (r(t) - y(t)) dt + k_d \frac{d}{dt}(r(t) - y(t)) \quad 1$$

where $u(t)$ is the control signal, $r(t)$ is the reference signal, and $y(t)$ is the measured process output. The behavior of the controller is governed by the proportional gain (k_p), the integral gain (k_i), and the derivative gain (k_d).

In this lab, you will examine the effect of these terms in sequence. You will first implement the proportional controller to get the system to respond quickly to changes in the reference set point. Proportional control will speed the response of a system, but typically also can cause the closed-loop system to have low damping, leading to high values of overshoot and oscillations. To help eliminate the overshoot and oscillations for the rudder control, a derivative control term will be added. Finally, you will see the effect of the wind adding a disturbance to the system. To eliminate the effect of the disturbance and achieve low steady-state errors, an integral control term will be added. Finally, you will have a chance to explore how varying the wind speed effects your system response.

Lab Apparatus

In this lab you will be using the hardware shown in the figure below. The apparatus consists of a miniature wind tunnel with an airfoil inside the test section. The airfoil is designed to mimic the rudder of a traditional airplane. A large fan mounted downstream of the test section draws air into the wind tunnel. A DC motor is used to rotate the airfoil inside the wind tunnel. The DC motor is connected to a rotary encoder via a timing belt. The airfoil's angle is measured by the encoder. Power switches for both the airfoil positioning motor and wind tunnel fan are located at the far left of the apparatus. When either the positioning motor or fan is turned on, a red LED will illuminate above their respective switches. Only turn on the motor or fan when in use; avoid powering either when not needed. The input power plug is located at the right rear of the unit. When power is applied, a green LED will illuminate below the "POWER" label.

The system is controlled by an Arduino compatible microcontroller located at the front right. The PID controller you design will use feedback from the encoder to calculate the error and then output the control signal to the DC motor as a varying voltage from 12 to -12 volts. The microcontroller can also vary the speed of the wind tunnel fan, thereby adjusting the airflow in the test section.

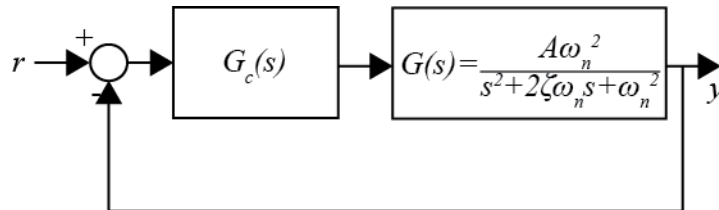


Pre-Lab Questions

Question 1: The transfer function of a PID controller can be written as follows:

$$G_c(s) = \frac{K_d s^2 + K_p s + K_i}{s}$$

When used with a second order system as shown below, write out the closed loop transfer function <deliverable>.



Question 2: If K_d and K_i are both zero, write out the approximation for settling time and percent overshoot <deliverable>.

Lab Overview

The purpose of this lab is to develop a PID controller to control the angle of a DC motor-driven airfoil. You will be provided with a miniature wind tunnel apparatus designed to simulate an airfoil positioning system with or without airflow. The transfer function of the plant (DC motor rotating airfoil) has been determined to be the following:

Plant	
Airfoil Motor without airflow	Airfoil Motor WITH airflow
$G_s = \frac{7841}{s^2 + 23.25s}$	$G_s = \frac{6411}{s^2 + 27.87s + 264.2}$

Given the transfer functions above, it can be seen that turning on the wind tunnel fan to create airflow changes the response of the plant. This is as expected as intuitively we would imagine that air deflected by an airfoil would produce a force proportional to the angle of the airfoil. We can treat this force as a disturbance in our system. The first goal of this lab is to develop a PD controller that can position the airfoil within a set of design requirements without any airflow in the wind tunnel.

The second goal of this lab will be to develop a PID controller that can position the airfoil with the same design requirements as before, but this time with the additional disturbance of airflow in the wind-tunnel.

Lab Procedure

Step 1 – Use Simulink to design Proportional controller

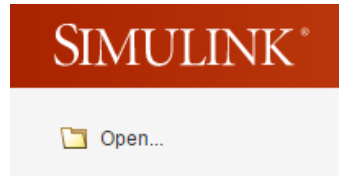
(20 min.)

Complete

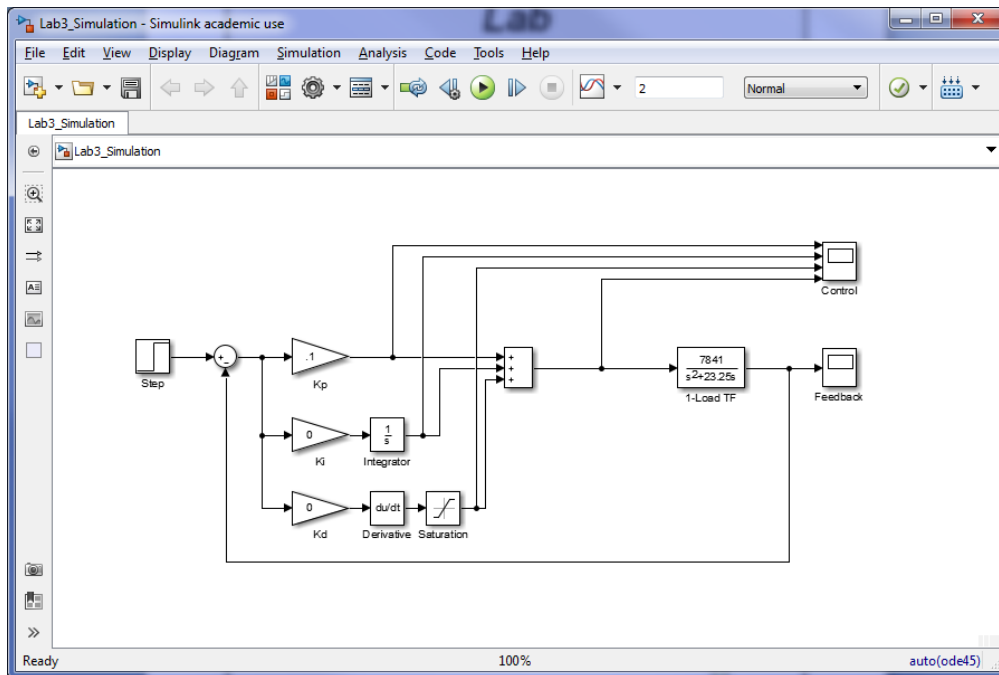
To start this lab, you will use the Matlab Simulink to model the system and design a Proportional controller. To begin, we will open a model of the **airfoil plant without airflow** and a PID controller.


- Start Simulink by typing the following into the command window:

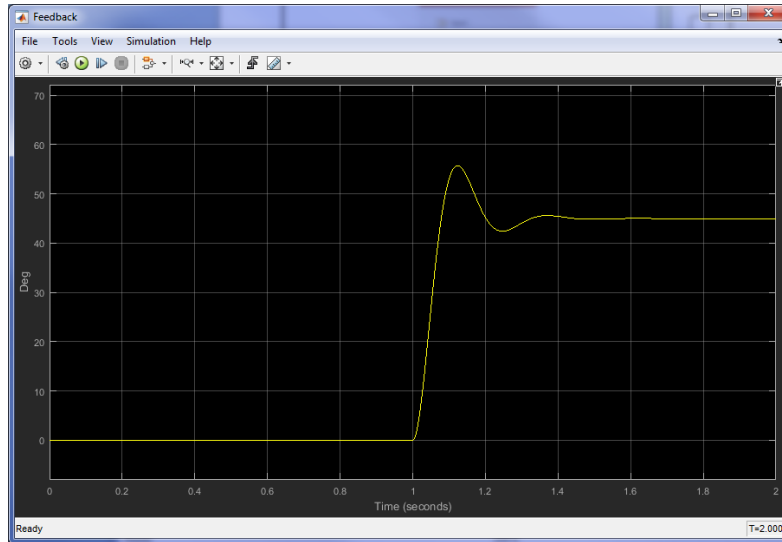
```
>>simulink
```
- Select “Open” and open the file “Lab3_Simulation.slx”. The file is available for download on Blackboard.



- The following Simulink model should open (note that the plant is the airfoil motor without airflow):



- Open the “Feedback” scope, and press play  to run the simulation. The scope should display the output shown on the next page.



- Use the cursor measurements to calculate the rise time (10%-90%) and % overshoot.
- Adjust the proportional gain, K_p , to produce a rise time of 40ms.
- Calculate the % overshoot.

Question 3: Plot the response of the system for the final value of K_p chosen. What is your chosen K_p ? What is the rise time and % overshoot? How do these agree with the estimate you derived in Question 2 when using the airfoil without wind transfer function <deliverable>.



Step 2 – Setup hardware

(10 min)

You will now connect the Airfoil Wind Tunnel to the computer. Refer to the image below for identification of components for the remainder of the lab.

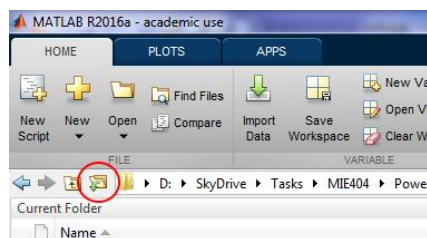


Complete

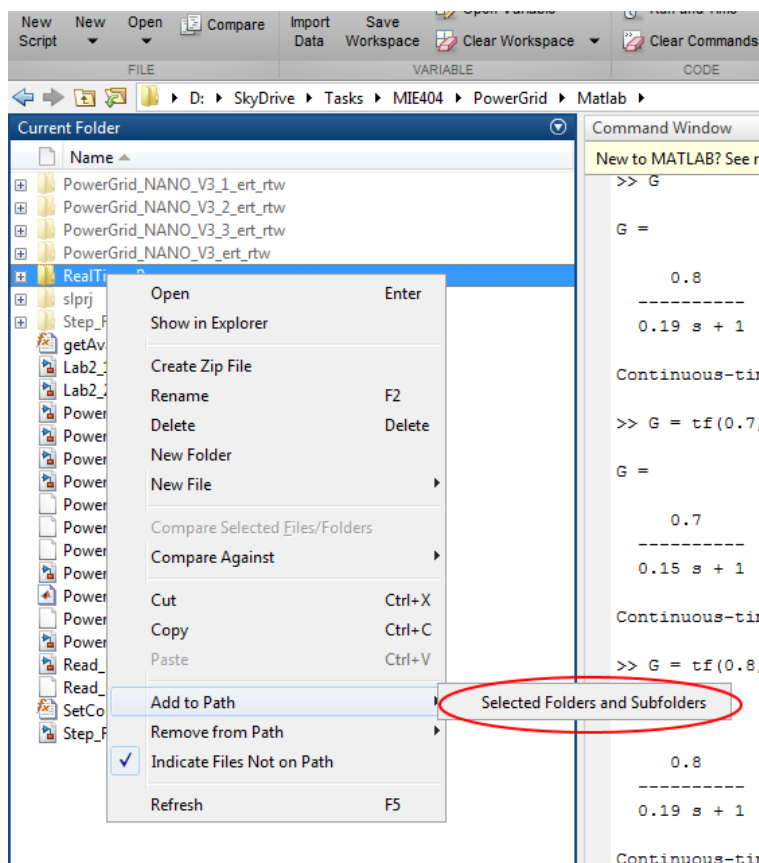
- Connect the power supply into the input jack at the top right of the board.
- Plug the power adapter into the power bar. A green LED should illuminate under the “POWER” label. If the LED does not illuminate, ask a TA for assistance.
- Connect the microcontroller to your computer using the supplied USB cable. Once the computer detects the microcontroller, the green “PWR” led and red “TX” LED should illuminate. If one or both of these LEDs fail to illuminate, ask a TA for assistance.

You will be using a Simulink model (Airfoil_Read_Data) to control the wind tunnel hardware and acquire data. We will now configure the model.

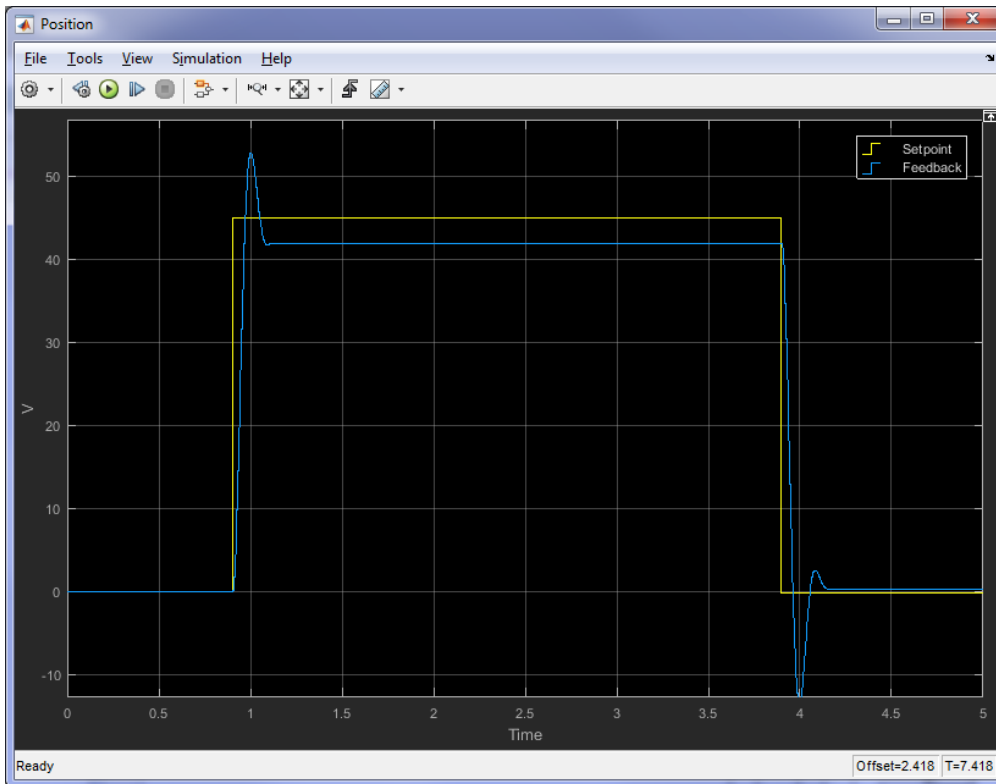
- Download the lab files from Blackboard and copy to a local folder.
- Set the active directory to the location where you copied the files for this lab. Click this button to set the active directory:



- Include the **RealTime_Pacer** directory to the path. In the folder view, right-click the directory and select **Add to Path > Selected Folders and Subfolder**



- Open the **Airfoil_Read_Data** Simulink model by double-clicking on the file.
- Open the **position** scope (**do not resize or maximize the scope window**)
- Turn the airfoil positioning motor **ON** by pressing the “Motor” button on the wind tunnel apparatus. A red LED above the button should illuminate.
- Rotate the airfoil 360 Deg by turning the pulley at the base of the motor. When the apparatus is not used for a period of time, the timing belt deforms and can lead to errors in the response. Turning the belt to a different position alleviates this issue.
- Align the airfoil to 0 DEG by turning the pulley at the base of the motor.
- Run the **Airfoil_Read_Data** model. After an initial delay, the chart should begin to update and the airfoil will rotate. If Matlab generates an error, or the chart does not update, ask a TA for assistance. A sample system output is shown on the next page.



- If the software and hardware behaves as expected, stop the model by pressing the stop icon.

Step 3 – Verify Proportional controller on hardware

(15 min)

Complete

You will now use the airfoil apparatus as the physical plant to verify your Simulink model. Note that due to variation between plants (as is the case in every real-world system), expect that your results will be close but not necessarily identical to the model. If your results are within a reasonable margin of error (comment on your definition of reasonable in your report), accept the results and continue with the lab.

- Enter the **Kp** value you determined in the **Lab3_Simulation** model into the **Kp** constant block in the **Airfoil_Read_Data** model. Ensure the **position** scope display is visible.
- Align the airfoil to 0 DEG by turning the pulley at the base of the motor.
- Start the model. The scope will provide real-time data from the apparatus.
- Use the cursor measurement tool to measure the **rise time** and **% overshoot**; note the results for the question below. Use the system's steady-state position as the reference value. (Do not use the setpoint).

- Ensure the cursor trace selection is set to "Feedback"



- Use the zooming tools as needed to get accurate measurements.



- If your system rise time is significantly different (>20% error) from the simulation, spend some time tuning the P controller to achieve the desired design criteria.

Question 4: Plot the response of the system. What is your final Kp value? What is the rise time and % overshoot? Does this match the system requirements? Does it match the simulated results? What are the sources of error? <deliverable>

☐
☐
☐
☐
☐
☒

Step 4 – Reduce overshoot by implementing PD control

(20 min)

Complete

You will now add a derivative control term to your feedback system. The derivative term will slow the system response and lower overshoot.

- Return to the **Lab3_Simulation** Simulink model.
- Adjust **Kp** and **Kd** to achieve the following design requirements:
(start with Kd = 0.003):

Rise Time: 40ms

Overshoot: <10%

Remember: Increasing Kp decreases rise time and increases overshoot
Increasing Kd decreases overshoot and slightly increases rise time

- Use the cursor measurement tool to measure the **rise time** and **% overshoot**; note the results for the question below.

Question 5: Plot the response of the system. What are your Kp and Kd values? What is the rise time and % overshoot? How did the derivative term affect the system? Did you need to adjust Kp also? <deliverable>

☐
☐
☐
☒

Step 5 – Verify PD controller on hardware	(15 min)	Complete
<p>You will now use the airfoil apparatus as the physical plant to verify your Simulink model. If your results are within a reasonable margin of error (comment on your definition of reasonable in your report), accept the results and continue with the lab.</p> <ul style="list-style-type: none"> Enter the Kp and Kd values you determined in the Lab3_Simulation model into the Kp and Kd constant block in the Airfoil_Read_Data model. Ensure the position scope display is visible. Start the model. The scope will provide real-time data from the apparatus. Use the cursor measurement tool to measure the rise time and % overshoot; note the results for the question below. Use the system's steady-state position as the reference value. (Do not use the setpoint). If your system rise time is significantly different from the simulation, spend some time tuning the PD controller to achieve the desired design criteria. <p>Question 6: Plot the response of the system. What is the rise time and % overshoot? Does this match the system requirements? Does it match the simulated results? What are the sources of error? <deliverable></p>		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>

Step 6 – Reduce steady-state error by implementing PID control	(20 min)	Complete
<p>You will now add a integral control term to your feedback system. The integral term is a summation of system error over time and is used to remove steady-state error. In a Type 1 system such as this, the theoretical steady-state error to a unit step input is 0. We will introduce a steady-state error to the system by moving air through the wind-tunnel which will act upon the airfoil. The airflow will resist the rotation of the airfoil and produce a reactionary torque, resulting in steady-state error.</p> <ul style="list-style-type: none"> Open the Lab3_Simulation_Wind Simulink model. The plant in this model simulates the airfoil with the wind tunnel turned on. Set Kp and Kd to the values you found in the Lab3_Simulation model. Open the "Feedback" scope. Run the model and observe the system response and steady-state error. Adjust Kp, Ki, and Kd to achieve the following design requirements: (start with $K_i = 1$): Rise Time: 40ms Overshoot: <10% SS Error: <1% after 0.5s <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>Remember: Increasing Kp decreases rise time and increases overshoot Increasing Kd decreases overshoot and slightly increases rise time Ki is used to eliminate steady state error</p> </div> <p>NOTE: Do not exceed the design criteria. Attempt to tune the system to match these criteria as closely as possible.</p>		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

- Use the cursor measurement tool to measure the **rise time**, **% overshoot**, and **SS error decay time**; note the results for the question below.

Question 7: Plot the response of the system. What are your K_p , K_d , and K_i values? What is the rise time, % overshoot, and SS error decay time? How did the integral term affect the system? Did you need to adjust K_p and/or K_d also? <deliverable>

☐
☐

Step 5 – Verify PID controller on hardware

(15 min)

Complete

You will now use the airfoil and wind tunnel apparatus as the physical plant to verify your Simulink model. If your results are within a reasonable margin of error (comment on your definition of reasonable in your report), accept the results and continue with the lab.

- Enter the **K_p** , **K_i** , and **K_d** values you determined in the **Lab3_Simulation_Wind** model into the **K_p** , **K_i** , and **K_d** constant block in the **Airfoil_Read_Data** model. Ensure the **position** scope display is visible.
- Align the airfoil to 0 DEG by turning the pulley at the base of the motor.
- Turn on the fan by pressing the “Fan” button. The red LED above the button will illuminate and the fan will start.
- Start the model. The scope will provide real-time data from the apparatus.
- Turn the wind tunnel fan **OFF**. Do not leave the fan running when not in use.
- Use the cursor measurement tool to measure the **rise time**, **% overshoot**, and **SS error decay time**; note the results for the question below.
- If your system response is significantly different from the simulation, spend some time tuning the PID controller to achieve the desired design criteria.
- If you made significant adjustments to the controller parameters, return to the **Lab3_Simulation_Wind** model and simulate your new **K_p** , **K_i** , **K_d** values.

Question 8: Plot the response of the system. What is the rise time, % overshoot, and SS error decay time? Does this match the system requirements? Does it match the simulated results? What are the sources of error? <deliverable>

☐
☐
☐
☐
☐
☐
☐
☐
☐
☐

Just For Fun	Complete
<p>If time permits, you can delve deeper into the design of this system. Up to this point, you have designed a PID control system that meets the design criteria for the airfoil positioning system under one specific condition (45 deg rotation and max airflow).</p> <ul style="list-style-type: none"> • Using your Kp, Ki, Kd values, test the hardware with the wind tunnel fan turned OFF. What is the resulting system response? • Attempt to design a PID controller that meets or exceeds the design requirements for both cases (fan is on and off) • Attempt to simulate your newly designed PID controller in Lab3_Simulation_Wind. Does it match? What are the possible sources of error? (hint: look at the "Control" scope) 	<input data-bbox="1328 411 1386 470" type="checkbox"/> <input data-bbox="1328 506 1386 564" type="checkbox"/> <input data-bbox="1328 600 1386 659" type="checkbox"/> <input data-bbox="1328 695 1386 753" type="checkbox"/>