ECE/SE 380 — Analog Control Systems

Fall 2021

Lab Manual v1.5

Rollen S. D'Souza

Revised by C. Caradima

Cover page image captured by SpaceX® on their STP-2 Mission, licensed under the Creative Commons Non-Commercial 2.0. See here for the original content. It is, of course, a photo of one of their self-landing booster rockets attempting a landing. If you are interested, see the discussion by Lars Blackmore on *Autonomous Precision Landing of Space Rockets* in this publication. It isn't very technical but you can find where to jump from there if you want a little more technical content.

* * *

A big thank you to

Carmen Caradima,

Prof. Christopher Nielsen,

Prof. Andrew Heunis,

ECE 380, Spring 2020 Students

and other ECE 380 lab and course instructors whose work has inspired much of this lab manual. This lab manual is a variation of past ECE 380 labs. Thank you to Prof. Derek Rayside whose general teaching style inspires the variations made in this lab.

* * *

Contents

List of Figures

List of Tables

Introduction

Welcome to your remote lab for ECE/SE 380, Introduction to Feedback Control. The lab portion of this course was designed to give you a hands-on feel for control using analog circuits. Given the current climate, this is clearly not possible in the way previously envisioned.

We persist. This lab is designed to run in MATLAB and Simulink. You will engage in understanding, developing, executing and analyzing Simulink diagrams that model and simulate control systems. Every lab is provided with brief motivation on how one can see this applied in a real world problem. I hope I can help connect the abstract block diagrams of Simulink with the real world for you.

Control systems is a very broad topic with a long history that even predates the ideas discussed in this course. This course and lab covers the very basic notion of control, the idea of the negative feedback loop, buttressed by the mathematical tools of linear dynamical systems.

MATLAB and Simulink

MATLAB is a desktop numerical and symbolic mathematics computing environment. We will not rely heavily on MATLAB scripting itself and instead use Simulink but you should be familiar with both environments. Please ensure that, when you are installing MATLAB, you install

(1) MATLAB,

(2) Simulink,

(3) the Control Systems Toolbox,

(4) the Simulink Control Design,

(5) the DSP System Toolbox,

(6) the Signal Processing Toolbox,

(7) the Statistics and Machine Learning Toolbox and

(8) the Symbolic Math Toolbox.

Simulink is an add-on to MATLAB that allows you to design and simulate systems (physical or otherwise) using block diagrams like in Figure 1.

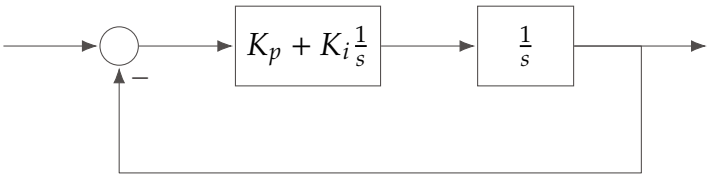
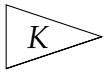


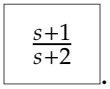
Figure 1: Example feedback diagram. The blocks are differential equations, expressed as a Laplace transfer function, that take an input signal and produce an output signal.

Figure 1 can be replicated perfectly in Simulink. That fact makes it easy for engineers to quickly design and verify control designs. Simulink even supports direct interaction with hardware but we will not utilize this feature.

The core of Simulink is the notion of a *block*, maps that take an input signal and produce an output signal. The blocks we'll be concerned with primarily are *gains* (multiplier) depicted in Simulink like



and *transfer functions* depicted in Simulink like



.

You may be provided template blocks to help you setup the Lab plant, the system we wish to control; it will then be your task to analyze this block, design a controller, and implement it by linking up blocks in the appropriate feedback architecture.

Lab Logistics and Deliverables

Lab deadlines can be found in your course outline and on LEARN. Read carefully. Times are provided in the active Eastern Time. Every lab requires you to complete a number of procedures described in green boxes like

Procedure 0.1

Here are some steps:

(1) Yes here is step 1,

(2) oh wow step 2,

(3) you got the idea.

and deliverables described in red boxes like

Deliverable 0.1

Here are some things you will need to save for your report.

Procedures are simply steps you should follow that will eventually result in one or more deliverables, namely something you must record and submit as part of your report. Note that the sequential number of the procedure may differ from the number of the deliverable. Procedures are, nevertheless, positioned and numbered such that they integrate well in the flow of information of specific sections. In other words, do not skip content trying to jump from a deliverable to what you may think is the corresponding procedure simply based on the procedure's number.

Every lab chapter ends with a final summative deliverable, usually consisting of a series of questions that you must answer. All the deliverables should be packaged up in an organized way; ideally they should be in your report in the order presented under headings directly corresponding to the deliverable label (1.1, 1.2, etc.).

We expect that the majority of your submissions will be typeset. However, for parts that involve a lot of formulas/diagrams/equations, it is acceptable to include **neatly hand-written parts**; make sure that they are **organized and legible**. Your final report must be submitted as one PDF document on Learn; it is the duty of both lab partners to verify that the PDF file has been rendered correctly. Your grade is entirely based on your report.

Occasionally, there has been confusion about what is allowed and what is not allowed in labs. You are responsible for knowing what constitutes an "academic offence" according to Policy 71 of the university. In particular, according to this policy:

1. Cheating is an academic offence. Cheating includes copying from another

student's work or allowing another student to copy from one's own work, submitting another person's work as one's own, fabrication of data and use of unauthorized aids.

2. Plagiarism (the act of presenting the ideas, words, or other intellectual property of another as one's own) is an academic offence. The use of other people's work must be completely and unambiguously acknowledged and referenced in all written material, including laboratory reports and computer programs.

In this course, labs are being done in groups of two students, subject to enrolment constraints. Each group must do their own analysis and their own design, do their own edits to the Matlab/Simulink files provided, and write up their own lab report. However:

1. You may talk to the lab instructor, the teaching assistant and the course instructor about any aspect of the lab.

2. You are allowed to consult with other students in the class or share *high-level ideas and approaches*, but *not* to share detailed analysis or detailed design results.

3. You may not obtain or look at lab reports (either in hardcopy or softcopy) written by other students, whether they are current students or former students of ECE/SE 380. You may not let any other student access any part of your lab report (either in hardcopy or softcopy).

4. In your report, you must completely and unambiguously acknowledge and reference any person, website, report, book, or notes that you used to help you with your work. You should reference this lab manual, and the course notes, for example.

5. You must include in each report, and sign, the following statement:

Declaration of Authorship

We acknowledge and promise that:

a) We are the sole authors of this lab report and associated simulation files/code.

b) This work represents our original work.

c) We have not shared detailed analysis or detailed design results, computer code, or Simulink diagrams with any other student.

d) We have not obtained or looked at lab reports from any other current or former student of ECE/SE 380, and we have not let any other student access any part of our lab work.

e) We have completely and unambiguously acknowledged and referenced all persons and aids used to help us with our work.

Student1 Name and Signature:
Student2 Name and Signature:

Every lab has its own grading scheme, but the following penalties apply equally to all labs:

- −5% for poor organization or hand-written parts that are messy or difficult to read,
- −5% for the submission not being in PDF format,
- −5% for not including the Declaration of Authorship shown above,
- −1% per hour within the first 24 hours past the submission deadline. After 24 hours, the penalty becomes −100%. If prior arrangements are made or a valid reason presented within one week from the missed deadline, the late penalty is waived. In no case will a lab report be accepted more than one week past the deadline. If a valid reason exists for being unable to hand in the lab report within the week following the deadline, then a solution tailored to each particular situation will be offered. **Contact the Lab Instructor for this.** All reports are due at `23:59 Eastern Time` on the due date shown in the Course Outline.

Minor mistakes are penalized quite steeply so check your work. Mistakes in discussions usually amount to theoretical mistakes in understanding; we will make an attempt to avoid double, or any other cascade, deduction.

LAB 1

System Identification Fundamentals

Low-pass filters are ubiquitous. They can be found in the hardware of audio processing chips, or the software of image editing tools. As you may recall from a course on Signals and Systems theory[1], a low-pass filter is a system that, generally speaking, rejects high-frequency signals while "passing through" low-frequency signals. In this lab you will be asked to analyze a low-pass filter block and determine its parameters via a time domain approach. This is not meant to be a difficult lab and is instead designed to give you a sense of the lab format. However, it will put to the test your knowledge of signals and systems. If you have forgotten your Laplace transforms, this is a good time to remind yourself of them!

In the real world, our models are rarely perfect. The techniques you learn here generalize nonetheless. Whenever you confront a novel system, devise a generic model that you think characterizes its behaviour and then come up with experiments that expose its characteristics; this gives you estimates of the generic model's parameters that best fit the actual system. We will encounter such a situation in a future lab.

1.1 Objectives

The primary objectives of this lab are to

(1) **Install** MATLAB, Simulink and required toolboxes. Alternatively, after establishing a VPN connection to the uW campus, **connnect** to `englab.uwaterloo.`

[1] If you haven't taken a course in Signals and Systems, speak to your Lab Instructor on knowledge you may be lacking.

ca, the portal for remote connection to several computers managed by Engineering Computing or by specific engineering departments. For the Engineering Computing (EngComp) machines, the available software is listed, almost all of them having Matlab 2019b available. Amongst the ECE machines, at least the ones named *ece-control* and *ece-public*, have Matlab 2019b installed.

(2) Understand how to **open** and **modify** a Simulink diagram and how to run a simulation.

(3) Understand how to **log** signals for future inspection.

(4) Understand how to **plot** figures and capture them.

(5) Understand the theory behind first-order low-pass systems.

(6) Understand how to **identify** a first-order low-pass system.

This lab may take a long time as it is very likely your first encounter with Simulink. As a result, I suggest you give yourselves time to work through the lab and the problems. I also recommend you spend some time by yourself exploring Simulink so as to reduce the amount of time you spend on future labs.

1.2 Experimental Procedure

This entire lab will be done using the "`Lab_1.slx`" Simulink model. First extract the "`Lab_1_Data.zip`" file available from Learn. Inside there you will find zip files created for each individual group. Extract the zip file associated to your group, and there you will find the relevant "`Lab_1.slx`" file.

In the Simulink model you will find a number of blocks already placed for you in an open-loop configuration:

- a signal generator,
- a summing junction,
- an adjustable gain block,
- a "plant", the system we are going to analyze, and
- a terminator.

The plant — labelled $P(s)$ in the diagram — can be assumed to be a transfer function taking the form

$$P(s) = \frac{bT}{s + aT} \quad ,$$

where $a, b > 0$ and $T \in \{10, 100\}$. Your goal is to use the techniques described in the following subsections to estimate the parameters a, b and T for your plant. This process is known as **system identification** and it is a critical precursor to control design. How can we design a controller if we don't even know the parameters of the system we wish to control?

Acquiring the Bandwidth in Time-Domain

The goal of this portion of the lab is to acquire the bandwidth of the given linear system in both open and closed-loop configuration. The result of this subsection will be the following deliverables.

Deliverable 1.1

Record your estimated DC gain and **capture** the generated figure with the cursors you used to measure the amplitudes. Your figure should like Figure 1.1.

Deliverable 1.2

At the bandwidth frequency, **capture a figure** of the input and output signal. It should look like Figure 1.2. Pay close attention to the time axis; it has been scaled to make the signal easy to measure and to visualize.

Deliverable 1.3

Repeat Deliverables 1.1 and 1.2 for the closed-loop system (described in Procedure 1.3).

What is the bandwidth? To understand and acquire it, we must first know what the DC gain of a system is.

Definition 1.1: DC Gain

Let $G(s)$ be a *stable* transfer function that maps an input signal $U(s)$ to an output $Y(s)$. The **DC gain** of $G(s)$ is $G(0)$ and, if $u(t) = A\mathbf{1}(t)$ for $A \in \mathbb{R}$, it

satisfies the relation

$$\lim_{t\to\infty} y(t) = AG(0).$$

Observe that the DC gain $G(0)$ is called precisely that because when we apply a square (DC-like; step) signal change, the system's output converges to the amplitude of the input times $G(0)$. The value that a signal converges to is called the **steady-state** value. Instead of computing the DC gain by evaluating the system's response to a step input, we take a different approach.

Procedure 1.1

You will acquire the DC gain of the plant $P(s)$. Follow these steps:

(1) **Open** the signal generator block, **set** the type of signal to be a sinusoidal wave, and set the **amplitude** to be 1 and the **frequency** to be 0.5 Hz. This signal is so low frequency that it emulates a constant signal. Note that you may use a lower frequency if you wish, like in Figure 1.1, but record the frequency you use.

(2) **Run** the script `generate_lab_1_plot.m` and inspect Figure 1, which depicts the input and output signal.

(3) **Measure the peak to peak amplitudes** of the input and output signals respectively when the system is in steady-state.

(4) **Estimate** the DC gain by dividing the output amplitude by the input amplitude. *Remember, we chose the input amplitude to be* 1*!*

(5) **Capture** the figure and save it. This produces Deliverable 1.1.

For low-pass systems, as we increase the frequency of the input, we will observe, the output amplitude *decreases*. The bandwidth frequency is the frequency that marks the point where we distinguish between the "low" frequency inputs a low-pass system sustains and the "high" frequency inputs a low-pass system rejects.

Definition 1.2: Bandwidth

Let $G(s)$ be a proper transfer function that maps an input signal $U(s)$ to an output $Y(s)$. Say the input is a sinusoid of the form $A\cos(\omega t)$ with amplitude A. In steady-state, the amplitude of $y(t)$ converges to a constant B. The **gain at frequency** ω is $\left\|G(j\omega)\right\| = \frac{B}{A}$. The **bandwidth (frequency)** of $G(s)$ is the

smallest frequency ω which satisfies

$$\frac{1}{\sqrt{2}} = \frac{\|G(j\omega)\|}{\|G(0)\|}.$$

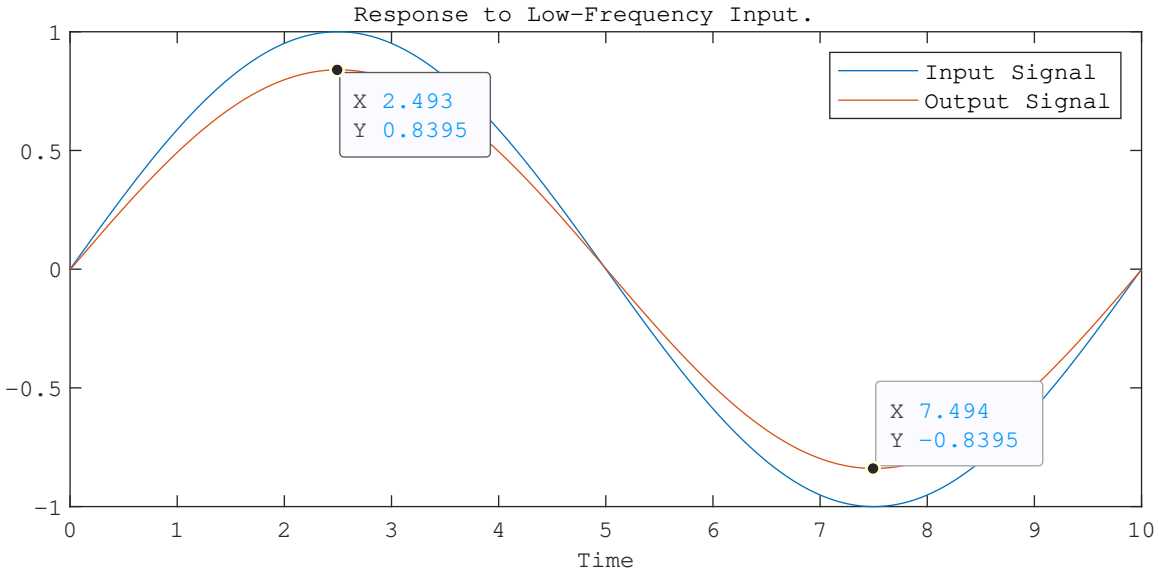


Figure 1.1: The low-frequency response to my low-pass plant $P(s)$. The gain at this low frequency is 0.8395. Can you see why? What is the DC gain approximately?

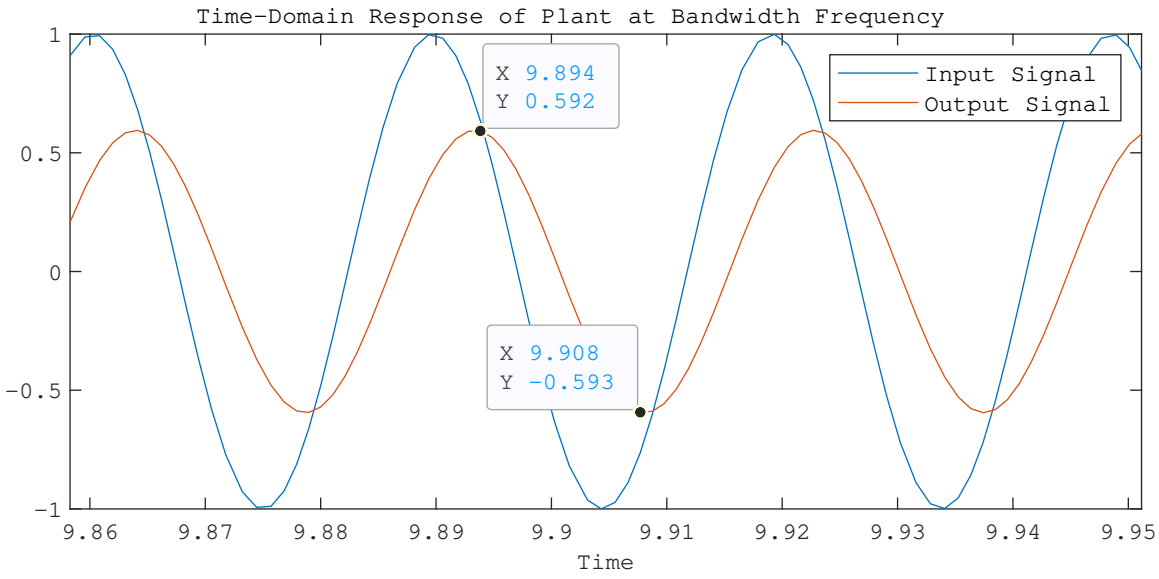


Figure 1.2: The response to my $P(s)$ at the bandwidth frequency. Can you see why this is the bandwidth frequency for my plant?

What is the definition above trying to characterize? The gain is the multiplier between the input amplitude and output amplitude. For a low-pass system, this gain decreases as a function of input frequency. The bandwidth frequency is the frequency where this gain has fallen, in comparison to the DC gain, by factor of $\sqrt{2}$. This is further demonstrated by the difference in response we observe between Figure 1.1 and Figure 1.2. Note how much smaller the output signal is in the response with higher frequencies in comparison to the response with lower frequencies.

Procedure 1.2

You will acquire the bandwidth frequency of the plant $P(s)$. Follow these steps:

(1) **Predict** the amplitude of the steady-state output at the bandwidth frequency. As an example, if your DC gain is 0.8395 then the ratio between the input signal amplitude and the output signal at the bandwidth frequency is

$$\frac{0.8395}{\sqrt{2}}.$$

If my input signal has amplitude 1, then I should look for an output signal amplitude with the above value.

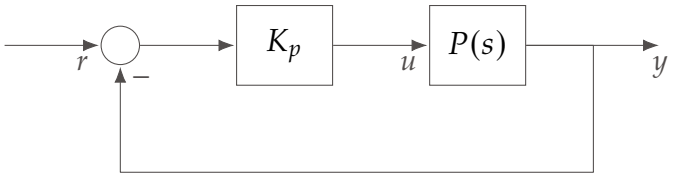


Figure 1.3: Closing the loop around the Plant $P(s)$ for Lab 1.

(2) **Open** the signal generator block, **set** the type of signal to be a sinusoidal wave, and set the **amplitude** to be 1 and the **frequency** to be 0.5 Hz.

(3) **Run** the script `generate_lab_1_plot.m` and inspect Figure 1, which depicts the input and output signal.

(4) **Measure the amplitudes** of the output signal when the system is in steady-state.

(5) If the amplitude isn't what you want, repeat Steps (2) through (4) with a different frequency. Increase or decrease by orders of magnitude to speed up the process!

(6) If the amplitude is what you predicted, **record** the frequency of the input you used to achieve the result. **Capture** this Figure and save it to produce Deliverable 1.2.

Having acquired the bandwidth for the open-loop system, we now investigate the closed-loop system.

Procedure 1.3

Now **close the loop** around the slider gain and plant $P(s)$. That is, connect your diagram so that it looks like Figure 1.3. Then **set** the gain K_p to 1. Repeat Procedures 1.1 and 1.2 for the closed-loop system depicted in Figure 1.3. Remember to acquire the Figures in both procedures for Deliverable 1.3.

Acquiring the Settling Time and Time Constant

In this section of Lab 1 we take a different approach to estimating the plant parameters. To establish you have completed this section, you must produce the following deliverables.

Deliverable 1.4

Capture the figure you used to measure the settling time and time constant in Procedure 1.4. **Show** your cursors.

Deliverable 1.5

Capture the figure you used to measure the settling time and time constant in Procedure 1.5. **Show** your cursors.

Instead of feeding in a sinusoid of low and high frequency, we acquire what is known as the **step response**. The step response of a plant $P(s)$ is the output $Y(s) = P(s)U(s)$ when $U(s)$ is the unit step function, i.e. $U(s) = \frac{1}{s}$. In practice, we often feed square waves as those are easier to construct. Note that square waves are really just a sequence of steps, from the minimum value to maximum value and back. There are two characteristic values we care about for a first-order step response. The 2%-settling time and the time constant.

Definition 1.3: The 2%-Settling Time

For a plant $P(s)$ and step input $U(s) = \frac{1}{s}$, the 2%**-Settling Time** is the time T it takes for the output $y(t)$ to reach within 2% of its steady-state value and stay within that 2% margin for all future time $t > T$.

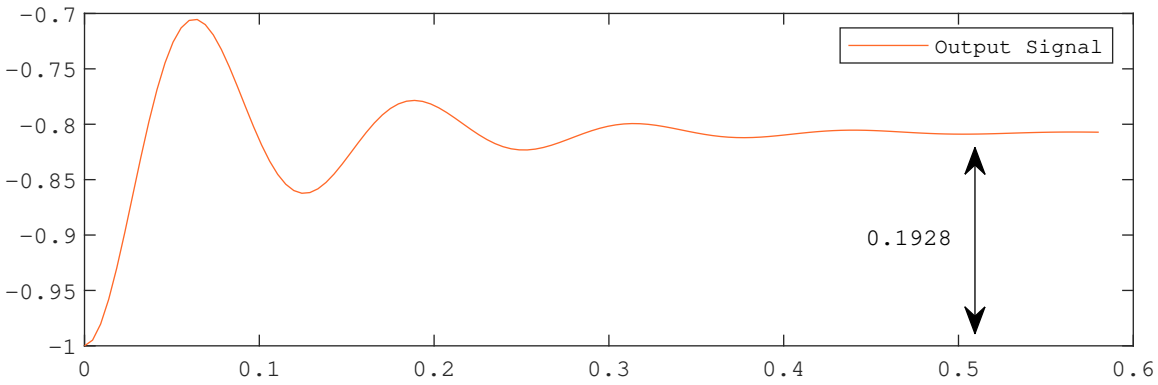
The settling time is a measure of long-term behaviour. The notion is depicted by Figure 1.4, which shows the time response of a *second-order* system. In Figure 1.4a the steady-state value is measured peak-to-peak, *from the base to the top* of the signal. Then 2% of the steady-state value is computed to define a cut-off point above and below the steady-state value, depicted as the broken lines in Figure 1.4b. Notice how the signal *stays* in the settling region after the settling time. The next characteristic of the step response worthy of measurement is the time constant. First let us define the standard first-order form for a first order transfer function.

Definition 1.4: Standard First-Order Form

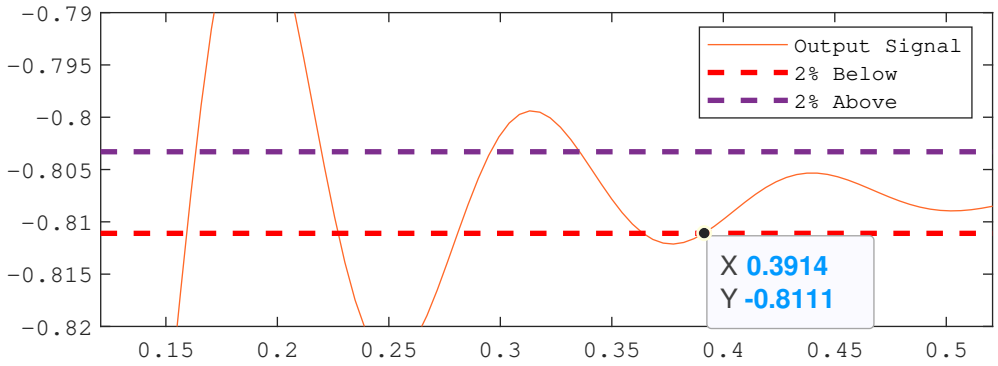
The standard first-order form for a strictly proper, first-order transfer function $P(s)$ is

$$P(s) = \frac{K}{\tau s + 1},$$

where $\tau, K \in \mathbb{R}$. The number τ is called the **time constant**.



(a) Complete step response.



(b) Zoomed-in view on the settling behaviour of signal.

Figure 1.4: The step response of some unknown plant with the 2%-settling time annotated in part (b). Observe how the the signal stays in the 2% region, demarcated by the broken lines, after the settling time of 391.4 ms is attained.

It can be verified that the number K in the standard first-order form is simply the DC gain. The time constant can be estimated experimentally in an easy way.

Definition 1.5: The Time Constant

For a first-order plant $P(s)$ and step input $U(s) = \frac{1}{s}$, the time constant is the time $\tau > 0$ it takes for the output $y(t)$ to reach $1 - e^{-1} \approx 63\%$ of its steady-state value.

The time constant is a measure of transient, short-term behaviour. Note that, unlike the settling time, the time constant only has meaning for a first-order system. You will derive the relationship as one of your deliverables.

Procedure 1.4

In this procedure you will acquire the settling time and time constant of your plant $P(s)$. First, **ensure** your plant is in open-loop configuration.

(1) **Open** the signal generator block and **set** the type to be a square wave. Also, **set** the frequency to a low value, such as 0.5 Hz and set the amplitude to 0.5. You will want to see atleast one step in the simulation but not too many more than that.

(2) **Run** the script `generate_lab_1_plot.m` and inspect Figure 1, which depicts the input and output signal.

(3) Using cursors, **measure** the steady-state value by measuring from *the base value of the signal to the steady-state point*.

(4) Using the above measurement, **calculate** the 63% point where the time constant is found and **calculate** the 2% margins where the settling time is found.

(5) Using cursors, **measure** the time constant and settling times.

(6) **Capture** your Figure and save it to produce Deliverable 1.4. Make sure to show your cursors.

Procedure 1.5

Repeat Procedure 1.4 for the closed-loop system described in Procedure 1.3. The Figure you generate in the last step produces Deliverable 1.5.

On Saturation or Clipping

In practice, unlike in simulation, the signals we can apply to a system are limited. This may be an inherent limitation, e.g. power supply limitations, or a self-imposed safety limitation, e.g. the angular velocity of an autonomous vehicle. Understanding where these hard constraints appear in your control system is important, as usually these constraints have *nonlinear* effects! Ideally, we want to remain in the region which has linear behaviour.

The only type of constraint you will encounter in these labs is the saturator constraint. A **saturator**, or **clipper** as it is known in electronics, limits a signal's maximum and minimum value to some preset constant. In this section, you will attempt to find this limit.

Procedure 1.6

Put your system in open-loop. **Set** the signal generator to generate a sinusoid with amplitude 1 at a low frequency. **Set** the gain K_p to 1 and **simulate**. **Increase** the gain K_p and **simulate** until you can observe the effects of saturation.

Deliverable 1.6

Capture the figure depicting the effects of saturation. **Record** the value of K_p where you had clipping occur.

1.3 Report Deliverable

Good job! You made it through Lab 1. You are required to submit a report that verifies you completed Lab 1 and that you understand the procedures you performed.

Preface your lab report with the Declaration of Authorship.

In addition to including

- Deliverable 1.1,
- Deliverable 1.2,
- Deliverable 1.3,
- Deliverable 1.4,
- Deliverable 1.5 and
- Deliverable 1.6

in your report, you are required to answer the questions of the following deliverable. Make sure to leverage your other deliverables in your answers!

Deliverable 1.7

(1) **Derive** a formula for the DC gain of $P(s)$ in terms of a, b, T.

(2) You know that your plant takes the form

$$P(s) = \frac{bT}{s + aT} \quad ,$$

for $a, b > 0$ and $T \in \{10, 100\}$. **Derive** a formula for the bandwidth of $P(s)$ in terms of $a, b, T.$, starting from the Definition 1.2.

(3) Using the formulas you derived in (1) and (2) and the estimates of your bandwidth and DC gain in Procedures 1.1 and 1.2, **estimate** a, b, T. Recall that T can only take on values 10 or 100.

(4) **Compare** the estimates of a, b, T made in (3) to your actual parameters. **Open** the "`Lab_1_Data.sldd`" to see your plant parameters. If there are discrepencies, explain why.

(5) **Find** a formula for the time constant of $P(s)$ in terms of a, b, T.

(6) **Find** the formula for the 2% settling time of $P(s)$ in terms of a, b, T. *Hint: You know $P(s)$, the input $U(s) = \frac{1}{s}$ and that the output is*

$$Y(s) = P(s)U(s).$$

Solve for an explicit expression for $y(t)$ in the time domain. Then find how long it takes to reach within 2% of the DC gain (the steady state-value for a unit step).

(7) Using the expressions found in (5) and (6) and the estimates of the steady-state value (as measured peak to peak), settling time and time constant in Procedure 1.4, **estimate** a, b, T.

(8) **Compare** the estimates of a, b, T made in (7) to your actual parameters.

(9) **Find** the transfer function from the input to the output when the system is in closed-loop, treating a, b, T, K_p as just unknown parameters. *Hint: For the closed-loop configuration, you can get an expression $Y(s) = G(s)R(s)$ where $G(s)$ is the closed-loop transfer function. Put $G(s)$ in standard first-order form.*

(10) **Discuss** how closing the loop, in Procedure 1.3, affected the bandwidth and DC gain. **Justify** your answer using the transfer function found in (9).

(11) **Discuss** how closing the loop, in Procedure 1.5, affected the settling time and time constant. **Justify** your answer using the transfer function found in (9).

(12) **Discuss** why we want to know the saturator limits. What would happen if we performed control design without verifying our control stayed within limits?

Grading Scheme

The grading scheme is shown in Table 1.1. The breakdown of your grade is shown per deliverable except in the case of the lab questions where it is shown per question.

Table 1.1: Grading scheme for Lab 1.

	Deliverable	Marks
	1.1	4
	1.2	4
	1.3	4
	1.4	4
	1.5	4
	1.6	4
Lab Subtotal		24
	1.7 (1)	1
	1.7 (2)	3
	1.7 (3)	3
	1.7 (4)	2
	1.7 (5)	3
	1.7 (6)	3
	1.7 (7)	2
	1.7 (8)	2
	1.7 (9)	2
	1.7 (10)	5
	1.7 (11)	5
	1.7 (12)	1
Report Subtotal		32
Total		56

LAB 2

Second-Order System Identification and Analysis

A large majority of systems we wish to control are physical systems. These systems are often modelled using Newton's laws

$$\text{mass}\,\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,\text{position} = \sum \text{natural forces} + \text{applied force},$$
$$\text{inertia}\,\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,\text{orientation} = \sum \text{natural torques} + \text{applied torque},$$

These systems — when they are linear — result in second-order transfer functions from the applied input (forces or torques) to the output (position and orientation). It follows that if we want to understand how to effectively control physical systems then we must first understand second-order systems and their properties!

In this lab you will explore the generic characteristics of a second-order linear system. You will discover the limitations of proportional error feedback control — the most naíve of control laws — when applied to a physical system. This motivates consideration of more complicated control strategies. Labs 4 and 5 will explore these options.

A word of caution. This is the final lab where it is possible to find out the parameters of your system by inspecting the data file. This, in principle, allows you to back-calculate what your procedures should tell you if the plant were a black box. This was intentional to allow you to check your work and ensure you understand how to perform the procedure. However, future labs will assume you've perfected this understanding. The parameters will be fairly well hidden like how it is sometimes in the `#RealWorld`*. So, make sure you understand how to perform the procedures!*

2.1 Objectives

The primary objectives of this lab are to

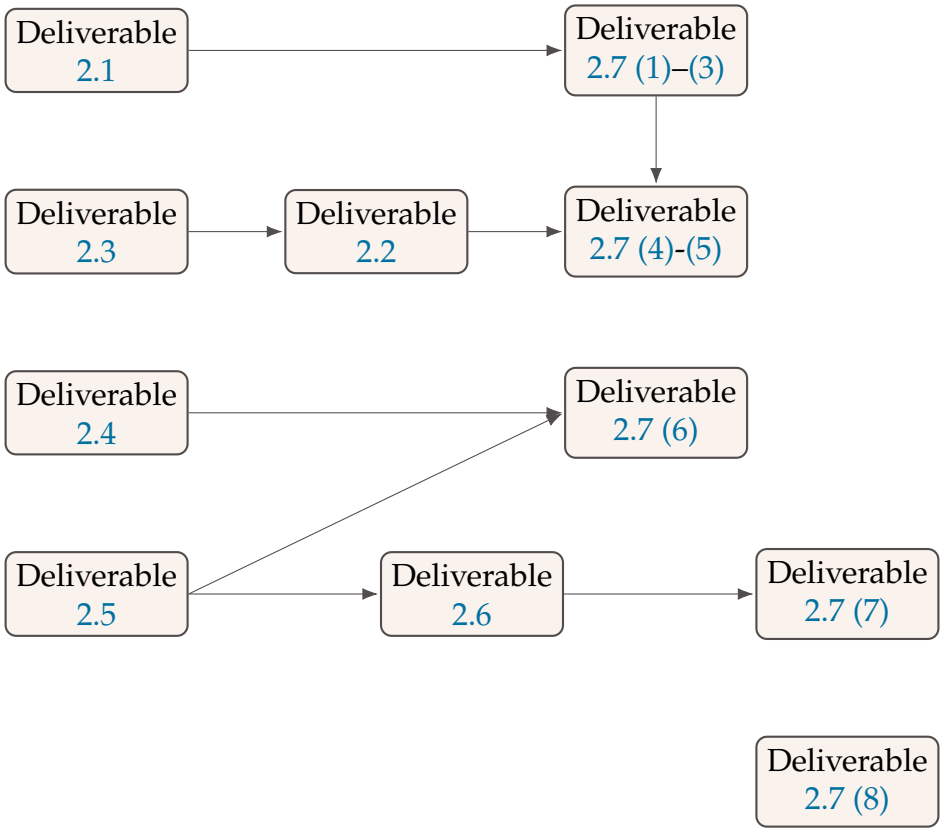
(1) **Learn** the characteristic properties of a standard second-order linear system.

(2) **Identify** the parameters of a standard second-order linear system.

(3) **Learn** how to acquire a Bode plot, the frequency response, and **understand** how to interpret it.

(4) **Explore** how a proportional control feedback affects the response of a second-order system.

Below is a dependency graph depicting the relationships between each of the deliverables. If an arrow flows from A to B, that means B depends on the partial or full completion of A.



2.2 Experimental Procedure

This entire lab will be done using the "`Lab_2.slx`" Simulink model. In there you will find a number of blocks already placed for you in an open-loop configuration:

- a step input,
- a summing junction,
- an adjustable gain block,
- a "fill-in", zero disturbance block,
- a "plant", the system we are going to analyze, and
- a terminator.

This time we will *not* use the signal generator and instead leverage the Model Linearizer App. Its usage is described in Appendix A.1. You may acquire the step response using the techniques described in Lab 1, but to acquire the frequency response you will have to use the Model Linearizer app. In the future, we will primarily rely on the Model Linearizer app as it will unify all your data collection requirements and is far easier to use than modifying a script and turning on/off logging. For this lab, the input signal starts off configured as r and the output signal as y. You will have to change it later in the lab. The plant — labelled $P(s)$ in the Simulink diagram — can be assumed to be a transfer function taking the form

$$P(s) = \frac{\hat{K}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

This form has a special name.

Definition 2.1: Standard Second-Order Form

The standard second-order order form for a strictly proper, second-order transfer function $P(s)$ with no zero is

$$P(s) = \frac{\hat{K}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

where ω_n is called the **natural frequency** and ζ is the **damping ratio**.

The primary goal of this experiment is to (1) estimate ζ,ω_n, and $\hat{K}$, (2) explore the characteristics of your system and (3) see how the characteristics change under

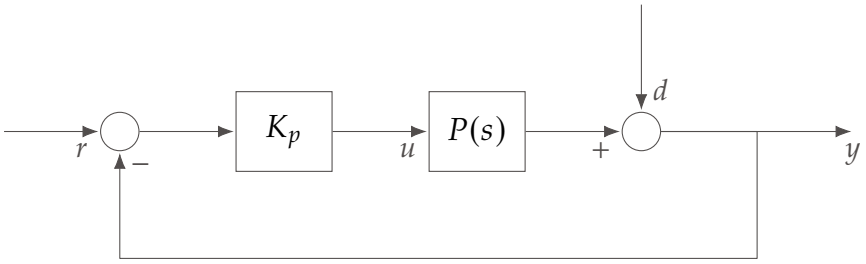


Figure 2.1: Closing the loop around the Plant $P(s)$ for Lab 2. Note that the loop is closed *after* the disturbance has been added to the output.

a proportional error controller. When analyzing the *closed-loop system*, we put our system in the form depicted by Figure 2.1.

Measuring the Characteristics of a Second-Order System

In Lab 1, you learned about the DC gain, the bandwidth, and settling time of a first-order system. We will use these characteristics, as well as one more new one, to help characterize a second-order plant. Your deliverable for this section is

Deliverable 2.1

Capture a figure showing your system's open-loop step response. **Measure** and **record**

(1) the steady-state value y_{ss},

(2) the time-to-peak value T_{peak} and the peak value y_{max}, and

(3) the 2% settling time T_s.

The figure included in your report must have **cursors** at the peak value and steady-state value.

You are already familiar with most of these measurements. The new measurements are the peak related measurements. For us, the following definition suffices.

Definition 2.2: Time-To-Peak and Overshoot

Let $G(s)$ be a proper, stable transfer function that maps an input signal $u(t)$ to an output $y(t)$. Let $u(t)$ be the (not necessarily unit) step function. The peak value of $y(t)$ is the maximum value $y(t)$ obtains, mathematically expressed by[a]

$$y_{\max} := \max_{\tau \in \mathbb{R}} \left| y(\tau) \right|$$

and the time it obtains the peak value, known as **time-to-peak** is defined as

$$T_{\text{peak}} := \operatorname{argmax}_{\tau \in \mathbb{R}} \left| y(\tau) \right|.$$

Then, if y_{ss} is the steady-state value of $y(t)$, the **percent overshoot** is defined as

$$\%\text{OS} := \frac{\left| \left| y_{\max} \right| - \left| y_{\text{ss}} \right| \right|}{\left| y_{\text{ss}} \right|}.$$

[a] normally sup is used but this suffices for our purposes.

Figure 2.2 depicts the measurements you perform to acquire the percent overshoot. You measure the maximum value of the output signal and the steady-state value of the signal. Then you compute the relative error between the maximum value and the settling value of the output.

Procedure 2.1

In this procedure you will capture a variety of characteristics of your provided second-order system to achieve the goal of Deliverable 2.1 and to eventually characterize the parameters a,b and K.

(1) **Ensure** your system is in the open-loop configuration.

(2) **Ensure** that you indicate the signal before the summing junction is an input signal (Input Perturbation) and the signal after the second summing junction is indicated as an output signal (Output Measurement). Refer to Appendix A.1 for more information on how to do so.

(3) **Open** the Model Linearizer app and **capture** a step response. Refer to Appendix A.1 on how to do so.

(4) **Measure** and **record** the following parameters of the output signal:

- the steady-state value y_{ss},

- the time-to-peak value T_{peak} and the peak value y_{max},
- the 2% settling time T_s.

Exploring Proportional Error Feeedback

For this section you will explore how proportional error feedback affects the behaviour of a second-order system. You will complete the following deliverables.

Deliverable 2.2

Capture a single step response of your closed-loop system with a gain $K_p \neq 1$.

Deliverable 2.3

Fill three rows of the table

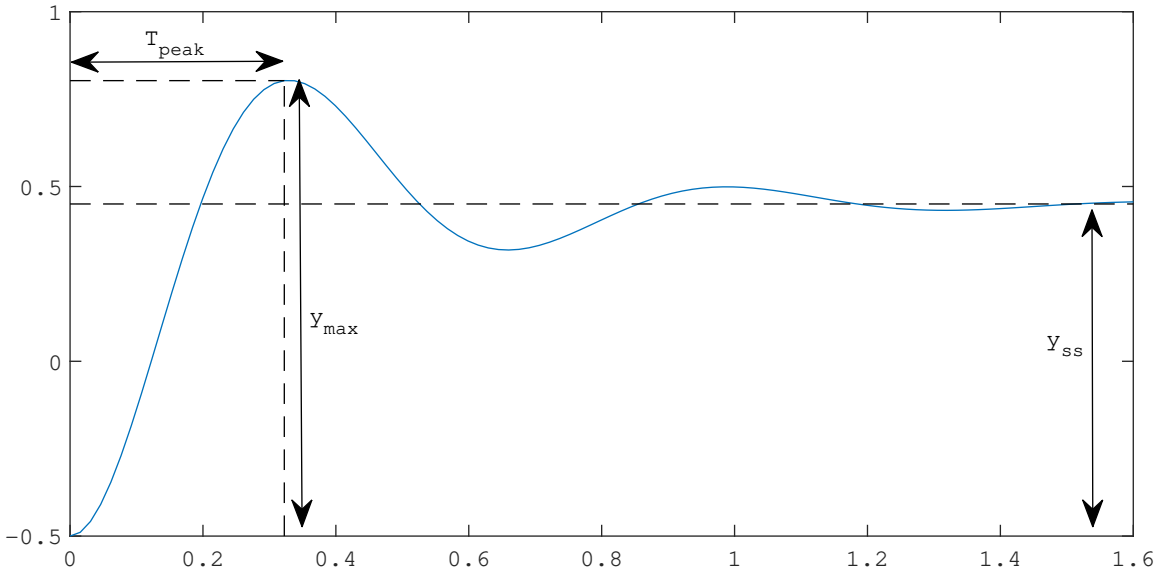


Figure 2.2: The maximum peak occurs at time T_{peak} with amplitude y_{max}. The steady-state value is shown as y_{ss}.

K_p Gain	y_{ss} Steady-State Value	y_{max} Peak Value	T_{peak} Time-To-Peak	%OS Overshoot
1				
Any				
Any				

for the closed-loop system. Note that the overshoot is computed from y_{ss} and y_{max}.

Procedure 2.2

In this procedure you will explore a variety of gains K_p.

(1) **Ensure** your system is in the closed-loop configuration as depicted in Figure 2.1.

(2) Choose a sequence of three gains to test. One of these gains must be $K_p = 1$. You may choose any value that is allowed by the slider gain provided to you in the Simulink Diagram.

(3) For each of your chosen gains K_p, **compute** the step response and **fill** the relevant row of the table in Deliverable 2.3

Acquiring the Bandwidth in the Frequency-Domain

Once again we will acquire the bandwidth of our system. We will do so for *both* the closed and open-loop configurations. This time, however, we will do so using the frequency response. In particular, you are asked to complete the following deliverables.

Deliverable 2.4

Capture a figure of the Bode plot for the open-loop system. **Include** a cursor at the bandwidth frequency on the magnitude plot and a cursor where you estimated the DC gain.

Deliverable 2.5

Capture a figure of the Bode plot for the closed-loop system with unity gain ($K_p = 1$). **Include** a cursor at the bandwidth frequency on the magnitude plot.

First of all, let us briefly review what a Bode plot depicts. Given a *real, rational and stable* transfer function $G(s)$ and (co)sinusoidal input $U(s) = \frac{s}{s^2+\omega^2}$ of frequency ω, the output $y(t)$ converges towards the signal

$$y_{\text{ss}}(t) = \left\|G(j\omega)\right\| \cos(\omega t + \measuredangle G(j\omega)).$$

Sometimes we say that the signal $y_{\text{ss}}(t)$ is the output of $G(s)$ in *steady state*. Recognizing that the input was $u(t) = \cos(\omega t)$, we observe that the output in steady state is an amplified/attenuated and phase shifted version of the input.

The amplification or attenuation of the signal $u(t)$ is determined by $\left\|G(j\omega)\right\|$. This is what is plotted in the **Magnitude Plot** of a Bode plot. Normally the standard units of this multiplicative factor is decibels, i.e.

$$\left\|G(j\omega)\right\|_{\text{dB}} = 20\log_{10}\left(\left\|G(j\omega)\right\|\right).$$

Remember when reading a Bode plot, such as in MATLAB, you will often have to convert from the decibel value $\left\|G(j\omega)\right\|_{\text{dB}}$ on the left to the unitless magnitude $\left\|G(j\omega)\right\|$ on the right. We will not concern ourselves with the phase plot in this lab but normally the units of the phase shift is degrees.

The magnitude plot can be used to measure the DC gain and bandwidth frequency. You will recall from Lab 1 that the DC gain can be approximated by the gain of $G(s)$ at low sinusoidal frequencies. This amounts to looking at the value that $\left\|G(j\omega)\right\|$ tends towards as $\omega \to 0$. Further you will recall from Definition 1.2 that the bandwidth frequency ω_{bw} of $G(s)$ solves the equation

$$\frac{1}{\sqrt{2}} = \frac{\left\|G(j\omega_{\text{bw}})\right\|}{\|G(0)\|}.$$

Converting our expression to be in decibels yields the approximate relation

$$\left\|G(j\omega_{\text{bw}})\right\|_{dB} = \|G(0)\|_{dB} - 3\,\text{dB}.$$

This suggests that to measure the bandwidth frequency, it suffices to look for the point where the magnitude plot $\left\|G(j\omega)\right\|_{dB}$ crosses the value 3 dB below the DC gain $\|G(0)\|_{dB}$.

Procedure 2.3

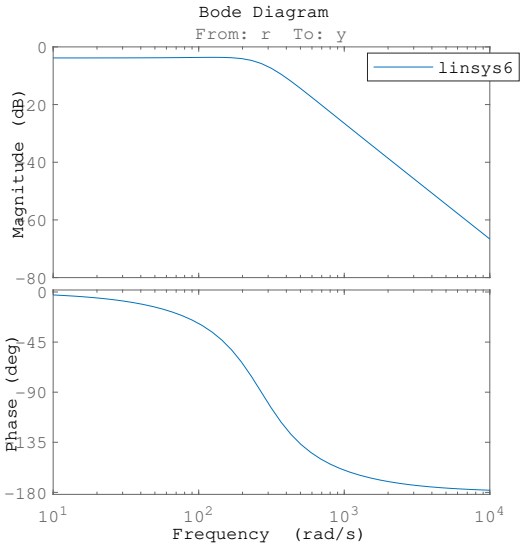
(1) **Ensure** your system is in the open-loop configuration. **Ensure** the gain K_p is set to 1.

(2) **Open** the Model Linearizer App.

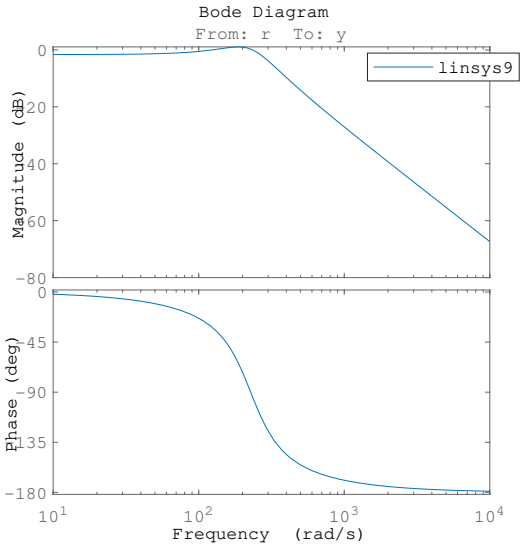
(3) **Acquire** a Bode plot. Your Bode plot will probably look a lot like Figure 2.3. Note that, for some of you, you may have a little peak in the magnitude plot like in Figure 2.3b; this is normal!

(4) **Measure** the DC gain on the Magnitude plot using a cursor. The DC gain is the value that the Magnitude Plot tends to as $\omega \to 0$.

(5) **Measure** the frequency ω at which the gain (magnitude plot) drops to a value 3 dB below the DC gain measured in the previous step. This frequency is called the **bandwidth**.



(a) A fairly well damped second-order system.



(b) A much less damped, $\zeta << 0.707$, second-order system.

Figure 2.3: Sample Bode plots of a second-order system.

Procedure 2.4

(1) **Put** your system is in the closed-loop configuration as depicted by Figure 2.1. **Ensure** the gain K_p is set to 1.

(2) Repeat steps (3)–(5) of Procedure 2.3.

Disturbance Rejection

You have almost completed the experimental part of this lab. This section explores how the closed-loop affects disturbance rejection. Disturbances are signals that we haven't modelled *or* cannot be modelled and therefore cannot account for. Naturally, large enough disturbances affect our ability to control systems. The beauty of control is that we can design systems that mitigate disturbances we do not explicitly model or know about! Here we will find that our system can mitigate some types of disturbance signals, but not all. The only deliverable for this section is

Deliverable 2.6

Include a Bode plot illustrating the response from the *disturbance input signal* $d(t)$ to the output signal $y(t)$. **Include** a cursor on the magnitude plot at *the bandwidth frequency of the closed-loop system.* (The frequency found in Deliverable 2.5)

To fulfill this deliverable, you must acquire a Bode plot of the input to output relationship between d and y.

Procedure 2.5

For this procedure, you will change the input signal. You will remove the "`Input Perturbation`" annotation from the reference r and place it on the disturbance d.

(1) **Ensure** the system is in the closed-loop configuration depicted in Figure 2.1. **Set** the gain K_p to 1.

(2) **Remove** the "`Input Perturbation`" annotation from the reference signal. This amounts to repeating the process described in Appendix A.1, i.e., right-click the signal wire r and select `Linear Analysis`

`Points/Input Perturbation` so that the input perturbation icon disappears from the signal wire r.

(3) **Add** the "`Input Perturbation`" annotation to the the disturbance signal wire d.

(4) **Open** the Model Linearizer App and **acquire** a Bode plot. *Expect the Bode plot to look substantially different than your previous Bode plots.*

2.3 Report Deliverable

Good job! You made it through Lab 2. You are required to submit a report that verifies you completed Lab 2 and that demonstrates you understand the procedures you performed.

Preface your lab report with the Declaration of Authorship.

In addition to including

- Deliverable 2.1,
- Deliverable 2.2,
- Deliverable 2.3,
- Deliverable 2.4,
- Deliverable 2.5 and
- Deliverable 2.6

in your report, you are required to answer the questions of the following deliverable. Make sure to leverage your other deliverables in your answers!

Deliverable 2.7

(1) Using the characteristics collected in Step (4) of Procedure 2.1, **determine** the parameters $\hat{K}$, ω_n and ζ that describe your plant $P(s)$ in the standard second-order form

$$P(s) = \frac{\hat{K}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

Hints: This time you drove the system with a unit step input. The approximate 2% settling-time for an underdamped, standard second-order system is approximated by the expression

$$T_{2\%} \approx \frac{4}{\zeta\omega_n}$$

and the percent overshoot is given by the formula

$$\%\text{OS} = 100e^{\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)}.$$

(2) It can be shown that the time-to-peak, another measurement you acquired, is equal to

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}.$$

Using your already acquired estimate of ζ, **estimate** ω_n using the time-to-peak. Based on your estimates, **explain** which you would rather use to estimate ω_n : the time-to-peak or the 2% settling time.

(3) For the closed-loop diagram of Figure 2.1, **compute** the transfer function from r to y. **Compute** the damping ratio, DC gain, and natural frequency of the closed-loop second-order system in terms of your plant's $\hat{K}$, ω_n and ζ and the gain K_p.

(4) Using the table filled out in Deliverable 2.3 and the formulas you derived in (3), **discuss** how changing the gain K_p affects the steady-state value y_{ss}, the time-to-peak T_p and the percent overshoot %OS of the closed-loop second-order step response. Also **discuss** how the settling time theoretically changes in the closed-loop as a function of K_p using the estimate formula shown in (1). ***Ensure*** *you discuss every characteristic you've collected; if there is no trend, say so; if the trend is complicated (not simply linear in K_p), say so.*

(5) By leveraging your answer for (4), **explain** why proportional error feedback control may not always be sufficient to control a physical system.

(6) Using the bandwidth frequencies found in Procedures 2.3 and 2.4, **state** how closing the loop changed the bandwidth frequency.

(7) Leveraging Deliverable 2.6, **discuss** how well the closed-loop system rejects disturbance signals. *Your answer should depend on the disturbance input frequency ω and the closed-loop bandwidth frequency which you marked on your plot.*

(8) **Derive** the closed-loop transfer function from the disturbance d to the output y. How do the disturbance rejection properties change as you increase K_p? Do they improve or degrade? *You will receive full marks if you provide a theoretical justification. You will want to produce a transfer function from d to y and it will have the form*

$$\frac{s^2 + As + B}{s^2 + As + D}$$

where A, B, D depend on the plant parameters and K_p. K_p should only appear in D. You can then split the above system into two subsystems

$$\left[s^2 + As + B\right]\left[\frac{1}{D}\frac{D}{s^2 + As + D}\right]$$

and recognize that K_p only affects the Bode plot of the latter system. This means it suffices to analyze the frequency response of

$$\frac{1}{D}\frac{D}{s^2 + As + D}$$

as K_p varies. You can put this subsystem into standard second-order form to aid in your analysis. You already should be able to argue how the DC gain and natural frequency of this new subsystem is affected by K_p, so it remains to connect this knowledge with how the Bode plot of this subsystem would be affected.

An empirical argument (showing the Bode plot for multiple K_p) will receive at most half the marks.

A correct answer without proof of any sort will receive 1 mark.

Grading Scheme

The grading scheme is shown in Table 2.1. The breakdown of your grade is shown per deliverable except in the case of the lab questions where it is shown per question.

Table 2.1: Grading scheme for Lab 2.

	Deliverable	Marks
	2.1	4
	2.2	4
	2.3	4
	2.4	4
	2.5	4
	2.6	4
Lab Subtotal		24
	2.7 (1)	3
	2.7 (2)	3
	2.7 (3)	3
	2.7 (4)	4
	2.7 (5)	2
	2.7 (6)	2
	2.7 (7)	3
	2.7 (8)	4
Report Subtotal		24
Total		48

LAB 3

Bringing it Together: Lateral Motion Control

Imagine you are a young, aspiring, junior engineer at VenX, a private company in an alternate Universe, manufacturing a mining probe on Venus[1] Because of your junior status, they have only asked you to design a simple controller that ensures the landing module lands at the correct location. The controller you design controls the level of horizontal thrust to achieve a desired position relative to some point on the surface. You may assume that all units are in standard units. Distances are measured in meters, time is measured in seconds, mass is measured in kilograms and all other units are such that they are consistent.

3.1 Objectives

(1) **Use** the skills you learned in Labs 1 and 2 to identify parameters of a black-box system.

(2) **Design** a proportional inner-loop controller that stabilizes the velocity and controls it to a desired value.

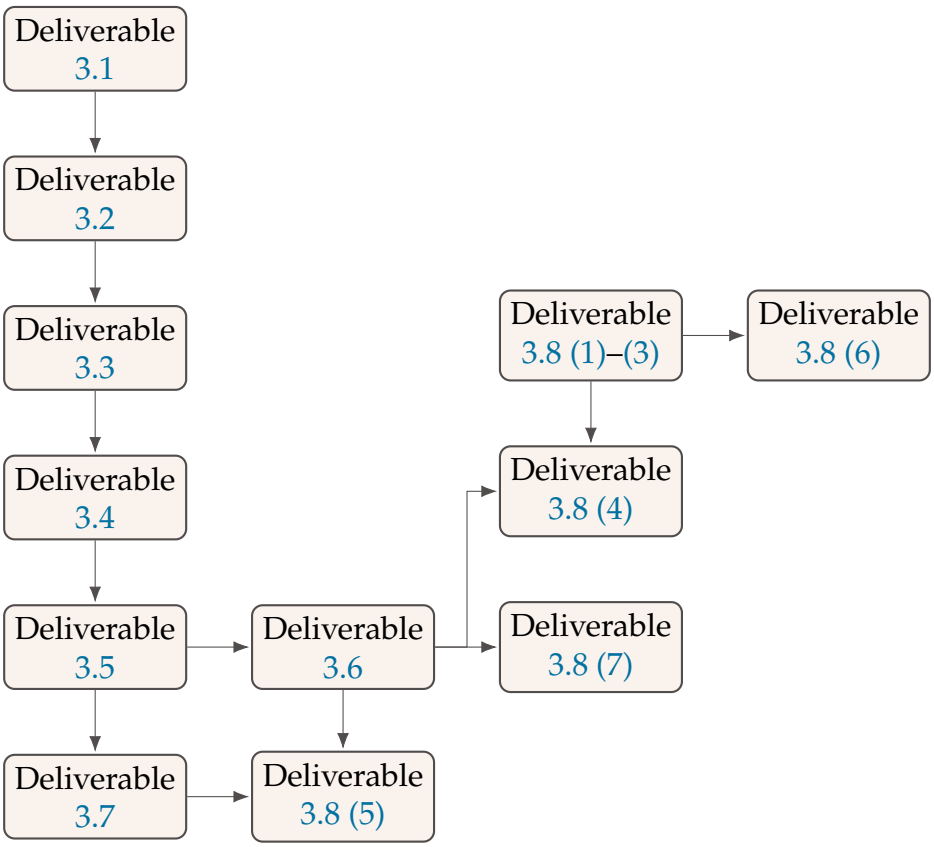
(3) **Design** a proportional outer-loop controller that stabilizes the relative position of the landing module with respect to a set point on the ground.

(4) **Compare and contrast** your design with that of Lab 2.

[1]This society isn't particularly concerned about the environmental impact.

(5) **Appreciate** how your controller deals with disturbances and unmodelled dynamics.

Unfortunately this lab is quite serial. Here is a dependency graph outlining the order in which you should progress through the lab. Note that the lab report questions (1)–(3) are not dependent on any of the simulations. It suffices to have read the lab, understand the components and how they connect.



3.2 Experimental Procedure

Unlike previous labs, you will now work with two Simulink models. Moreover, this lab will involve a larger number of moving parts; you will introduce three feedback interconnections!

In Part I you will stabilize the physical plant that maps input forces to an output horizontal velocity. You will treat velocity as the measurement (output). Once stabilized, you will design a controller in Part II that, again using velocity as a measurement, ensures perfect tracking of a reference velocity. We call this the

inner loop controller since it corresponds to the inner most loops of the system (ignoring the stabilizing loop). Finally, in Part III you will design a controller that uses the inner loop and negative feedback to achieve perfect tracking to a desired reference position. The end result is that you will have a closed-loop system that stabilizes the nonlinear landing module dynamics to achieve a desired horizontal position against disturbances!

Part I: Stabilize and Identify the Plant

Your deliverables for this subsection are:

Deliverable 3.1

Choose and **record** a gain value K_1 for your stabilizing gain.

Deliverable 3.2

Acquire the step response from the signal labelled $w(t)$ to the signal $v(t)$ when the signal $v(t)$ is connected to the G_1 loop summing junction (aptly named "G_1 Loop").

Deliverable 3.3

Measure the time constant τ_1 of your stabilized plant using the step response acquired in Deliverable 3.2.

This section concerns the Part I area of the Simulink model

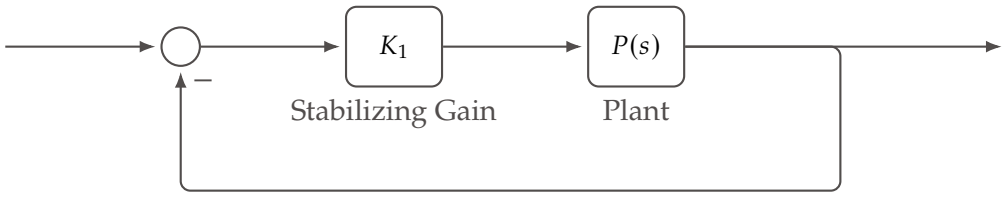
`Lab_3_Velocity_Controlled_Landing_Module.slx`.

In this lab, your plant is a simulated physical system. The system has nonlinear dynamics, but we will treat it as a linear system and you will observe the power of linear control. First, in order to even design a controller to achieve our objectives, we must stabilize the system. The reason for this is quite simple. A physical system normally integrates the input force into velocity (and then position) and, unless there are external forces such as friction, the plant is of the form

$$P(s) = \frac{1}{Ms}$$

where $M > 0$ is the mass of the landing module. The step response of this system is unbounded. It is true that there is friction in your model, but it is often hard

to model friction in reality, so we will take the above equation as our model. To stabilize the plant, close the loop in the following way



with a randomly chosen gain $K_1 > 0$. The gain you choose is up to you but make sure you record it.

Procedure 3.1

This procedure simply reiterates the steps above.

(1) **Close** the loop as described above. You should connect the signal labelled v to the summing junction labelled "G_1 Loop."

(2) **Choose** and **set** a random positive gain for K_1. The block is named "Stabilizing Gain" in the Simulink model. **Record** the chosen gain for Deliverable 3.1.

(3) **Acquire** a step response from the signal $w(t)$ to the signal $v(t)$. Remember from Lab 2 that you need to set w as an input perturbation and v as an output measurement. **Save** the figure for Deliverable 3.2.

(4) **Measure** the time constant of this system using the acquired step response. We will call this time constant τ_1. **Record** τ_1 for Deliverable 3.3.

Part II: Reference Velocity Control Design

Upon performing Procedure 3.1, you notice that the step response is extremely slow. It took more than a minute of simulated time to achieve a speed of 1 m/s (3.6 km/h)! If this is the response speed of our system, we will never be able to control it to a desired position. In this section, you will solve this problem by designing a proportional error feedback controller that controls to a *desired velocity* but with a much better time constant. Your deliverable for this section is

Deliverable 3.4

Acquire a (unit) step response from the signal $v_r(t)$ to the signal $v(t)$. **Ensure** you have cursors that indicate the following:

- the time constant is $\frac{1}{\sqrt{50}} \approx 0.1414s$.
- the DC gain, or steady-state value, is 1.

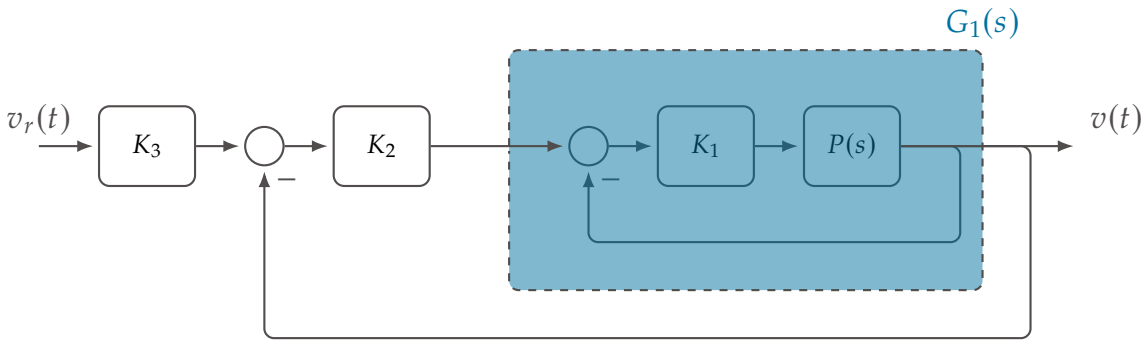
This section concerns the Part II area of the Simulink model

`Lab_3_Velocity_Controlled_Landing_Module.slx`.

The now stabilized plant, as a transfer function from $w(t)$ to $v(t)$, will be referred to as $G_1(s)$ and is depicted below in the block diagram. Verify for yourself that

$$G_1(s) = \frac{1}{\tau_1 s + 1}$$

where τ_1 is the time constant of $G_1(s)$ (as estimated in Deliverable 3.3). Close the loop around $G_1(s)$ with a proportional error feedback controller like so



Let's denote by $G_2(s)$, the transfer function from $v_r(t)$ to $v(t)$. This transfer function takes the form:

$$G_2(s) = \frac{V(s)}{V_r(s)} = \bar{K}\,\frac{1}{\tau_2 s + 1}.$$

Your task is to ensure that $G_2(s)$ has the desired time constant of $\frac{1}{\sqrt{50}}$ and has a DC gain of 1.

Procedure 3.2

In this procedure you use another closed-loop to accelerate the response of your system, thereby making it usable to do position step tracking.

(1) **Close** the loop as described above. You should connect the signal labelled v to the summing junction labelled "G_2 Loop."

(2) **Derive** symbolic expressions for $\bar{K}$ and for τ_2, using the expression of $G_2(s)$.

(3) **Calculate** the value of K_2 to achieve the specification $\tau_2 = \frac{1}{\sqrt{50}}$.

(4) **Calculate** the gain K_3 such that $G_2(s)$ has a DC gain of 1.

(5) **Set** the respective blocks corresponding to K_2 and K_3 in the Simulink diagram.

(6) **Acquire** the step response from the signal $v_r(t)$ to the signal $v(t)$. Remember again to change which signal is the input perturbation from Part I. Your input is now $v_r(t)$.

(7) **Place** cursors on the time constant and steady-state value to prove you met the specification. **Save** your figure for Deliverable 3.4.

Part III: Reference Position Control Design

You have made it to the final experimental part of this lab. In this part, you will ensure perfect tracking of a constant position reference by closing the loop around our newly stabilizing velocity control system. Your deliverables for this part are

Deliverable 3.5

Acquire the step response from the signal labelled $r(t)$ to the signal $x(t)$ when the signal $x(t)$ is connected to the G_3 loop summing junction (aptly named "G_3 Loop"). This is the step response of the transfer function $G_3(s)$ depicted in the Simulink model.

Deliverable 3.6

Run the "`visualize_landing.m`" script and **save** both Figures that are generated.

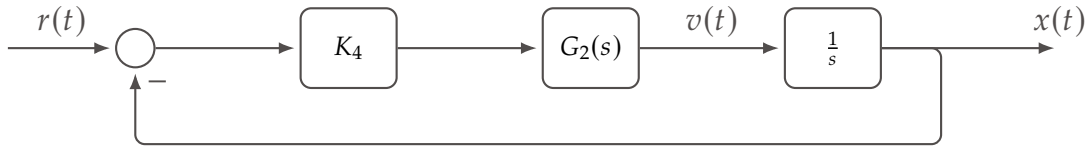
Deliverable 3.7

Acquire the Bode plot from the signal labelled $d(t)$ to the signal $x(t)$.

This section concerns the Part III area of the Simulink model

`Lab_3_Position_Controlled_Landing_Module.slx`.

You will notice that the other Simulink model you worked with is labelled $G_2(s)$ in this model. Since you have designed a controller that achieves a desired reference velocity, we leverage it in this model to design an outer loop controller that achieves a desired position. Close the loop to arrive at a diagram



If you performed Part II correctly, the transfer function $G_2(s)$ takes the form

$$G_2(s) = \frac{1}{\frac{1}{\sqrt{50}}s + 1}.$$

Calculate the transfer function from $r(t)$ to $x(t)$, denoted $G_3(s)$ in the Simulink model.

Procedure 3.3

You will make a specific choice for K_4 in this procedure and then simulate the entire non-linear system.

(1) **Close** the loop as described above. You should connect the signal $x(t)$ to the G_3 loop summing junction (aptly named "G_3 loop").

(2) **Determine** and **set** the gain K_4 so that the system $G_3(s)$ is critically damped. To be clear, with your choice of K_4 you should find that $\zeta = 1$ when $G_3(s)$ is put in standard form.

(3) **Acquire** the (unit) step response from the signal $r(t)$ to the signal $x(t)$. Remember to go into the other model file and remove any annotations you have used there before setting new ones in this model file. **Save** your figure for Deliverable 3.5.

(4) **Run** the "`visualize_landing.m`" script and **save** both Figures that are generated for Deliverable 3.6. *Do not be alarmed that the behaviour does not look like the step response.*

Now that you have designed a control system that regulates position and simulated the entire nonlinear system, you are almost ready to do analysis. We require you to generate one more plot, a Bode plot, that will help you in analyzing how well your system deals with the approximate model. In the following procedure, be very careful where you place your annotations, and make sure to save your diagram before running the Model Linearizer. You must complete the previous procedure before starting this final procedure.

Procedure 3.4

In this procedure, you will acquire the Bode plot from the input disturbance force $d(t)$ to the output position $x(t)$.

(1) **Ensure** you completed Procedure 3.3.

(2) **Remove** all annotations from all signals in *both* Simulink models.

(3) In Part I of the Simulink model

`Lab_3_Velocity_Controlled_Landing_Module.slx`.

set the input perturbation annotation for the signal $d(t)$ that is added into the signal right before the Plant block. $d(t)$ is the signal coming out of the "Disturbance Signal" block. Make sure to **save** your changes before proceeding.

(4) **Set** the output measurement annotation for the signal $x(t)$ right after the integrator block (found in the "Position" Simulink model).

(5) **Acquire** the Bode plot and **save** the figure for Deliverable 3.7.

3.3 Report Deliverable

You made it through the Lab 3 experiment! As usual, you are expected to submit a report demonstrating that you completed the lab and that you understand the tasks performed.

Preface your lab report with the Declaration of Authorship.

In addition to including

- Deliverable 3.1,
- Deliverable 3.2,
- Deliverable 3.3,
- Deliverable 3.4,
- Deliverable 3.5,
- Deliverable 3.6 and
- Deliverable 3.7

in your report, you are required to answer the questions of the following deliverable. Make sure to leverage your other deliverables in your answers!

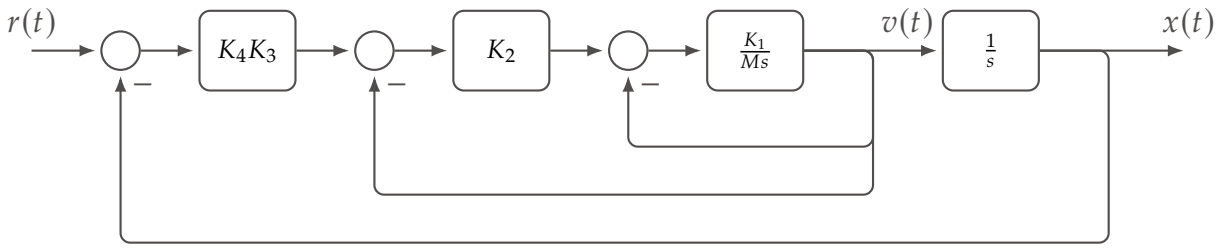


Figure 3.1: Entire closed-loop architecture for Lab 3.

Deliverable 3.8

(1) The entire closed-loop system $G_3(s)$ can be described by the diagram in Figure 3.1. **Derive** the transfer function from $r(t)$ to $x(t)$ in terms of the *symbols* K_1, K_2, K_3, K_4, and M. Leave your answer in symbolic form. *It is not required, but it is wise to verify your answer against the transfer functions discussed throughout the lab. They should be consistent.*

(2) You should find that the transfer function that you derived in (1) was a second-order transfer function with no zero. **Write** the transfer function of (1) in standard second order form by deducing what its DC gain $\hat{K}$,

natural frequency ω_n and damping ratio ζ are in terms of the symbols K_1, K_2, K_3, K_4, and M. Leave your answer in symbolic form.

(3) Unlike in Lab 2, you can change the settling time of the final, second order transfer function from $r(t)$ to $x(t)$. Leveraging the formula that estimates the 2% settling time

$$T_{2\%} \approx \frac{4}{\zeta\omega}$$

for a transfer function in standard form **identify** the two gains that affect the settling time of your transfer function $G_3(s)$ (use the standard form derived in (2)). *Aside: You may assume for this question that your system is underdamped. It is true that we designed for a critically damped system and, in that case, the formula does not apply. However, we will gloss over this technicality.*

(4) To produce Deliverable 3.6, you simulated the true response of your controller against the complete, nonlinear system; not a linear approximation of it! The Model Linearizer performs a linear approximation of the nonlinear differential equation found in the Plant block. It then gives you the step response of the corresponding linear system's transfer function. As a result, you designed against only an approximation of the true system. This explains why you will have observed a modest to large overshoot of the target horizontal position before the landing module settles. That doesn't mean linear system theory is completely useless, however.

Using the transfer function $G_3(s)$ found in (2), **determine** a gain (not a particular value, the symbol) that you can change to reduce the overshoot *without affecting the settling time*. **Justify** your explanation using the relevant overshoot formula and make sure to include whether you would increase or decrease that gain. *It is not required for your submission but I highly recommend you give your answer a try in simulation. You might be surprised!*

(5) As mentioned in (4), to produce Deliverable 3.6 you simulated your control design against the real system. The engineers at VenX included in that model a variable wind force that increases in magnitude as you get closer to the ground *and* has some sinusoidal variability. As part of

the specification, the engineers would like to know just how large of a force they can afford.

Since wind is just an additional force to the input force $u(t)$, we can model it as $d(t)$; $d(t)$ is an input disturbance. Assume

$$d(t) = B + A\sin(\pi t),$$

where $B, A > 0$.

Using the Bode plot you acquired in Deliverable 3.7, **justify** why you did not notice large sinusoidal disturbances in your response acquired in Deliverable 3.6. **Specifically determine** how large the constant A would have to be to observe a sinusoid of amplitude 1 m in the output.

(6) The software developers at VenX want your controller to have $x(t)$, your horizontal position, track perfectly any constant reference position $r(t)$.

Does your control system (treating $r(t)$ as the input and $x(t)$ as the output) do this? If yes, does it matter what the constants K_1, K_2, K_3, K_4, and M are? **Justify** using the theory learned in the course. *You may assume that K_1, K_2, K_3, K_4 and M are positive.*

(7) The philosophy of this control design was to (1) design a controller that regulates a desired velocity fast enough by controlling the input force and (2) design a controller that regulates a desired position by controlling the desired velocity. This is a very common technique used in aerospace since it breaks the control design into smaller, easier chunks that aim to solve different problems. With this philosophy in mind, the data you have at hand, and the answers to all the previous questions, answer the following question as best as you possibly can:

Your supervisor at the company VenX was alarmed by your Deliverable 3.6. They think the acceleration imparted by your controller is too large and could damage delicate components that are deployed on descent. How could you decrease the acceleration imparted by your controllers? *This is a conceptual question. We are looking for you to think about which gains to change and why. We are not looking for you to pick exact numbers. You may justify your answer using physical intuition (physics) or any formulas that apply.*

Grading Scheme

The grading scheme is shown in Table 3.1. The breakdown of your grade is shown per deliverable except in the case of the lab questions where it is shown per question.

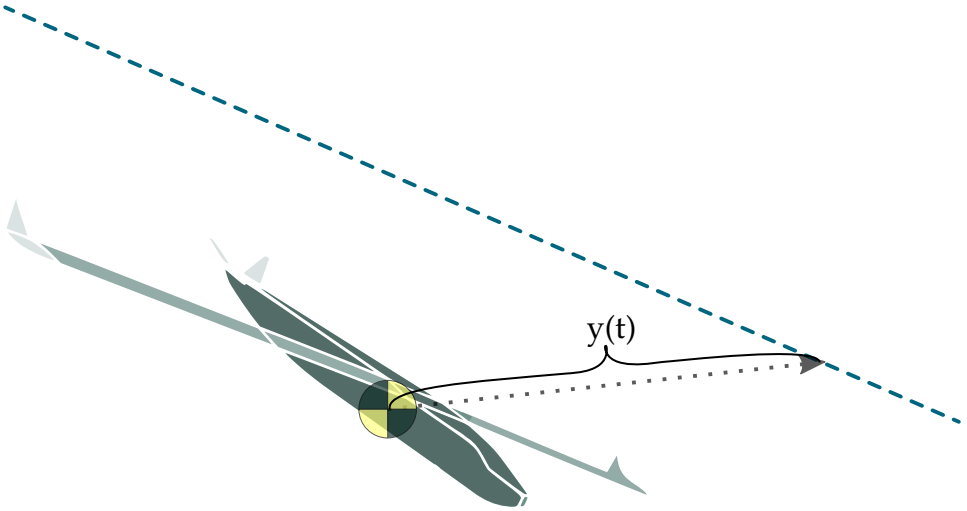
Table 3.1: Grading scheme for Lab 3.

	Deliverable	Marks
	3.1	1
	3.2	4
	3.3	2
	3.4	8
	3.5	4
	3.6	4
	3.7	4
Lab Subtotal		27
	3.8 (1)	4
	3.8 (2)	4
	3.8 (3)	4
	3.8 (4)	4
	3.8 (5)	3
	3.8 (6)	3
	3.8 (7)	3
Report Subtotal		25
Total		52

LAB 4

PID Analysis

You are now a slightly more experienced junior engineer at VenX. This time, you are designing an unmanned aerial vehicle that is dropped into the atmosphere and performs powered glides along predefined paths over the surface.



The purpose is to do surveillance[1] of the planet's surface to map out the landscape.

The controller you design controls the turning rate (rad/s) and the output you measure is the distance to the straight line path $y(t)$ (m). The controller must track a prespecified distance away from this path (constant references). In practice the

[1] Don't worry. The CEO of VenX, Lone Dusk, has assured you that the FBI are not using your technology for surveilling citizens in protests. Should you trust him? Probably not. But in this lab, you live in a utopia where people and the government are very honourable.

transfer function from the input to $y(t)$ is highly nonlinear, but the senior engineers have designed an inner loop controller that makes the plant $P(s)$ look like

$$P(s) = \frac{1}{s^2}.$$

Take `ECE 486`[2] if you are interested in how to do this.

4.1 Objectives

The goals of this lab are to

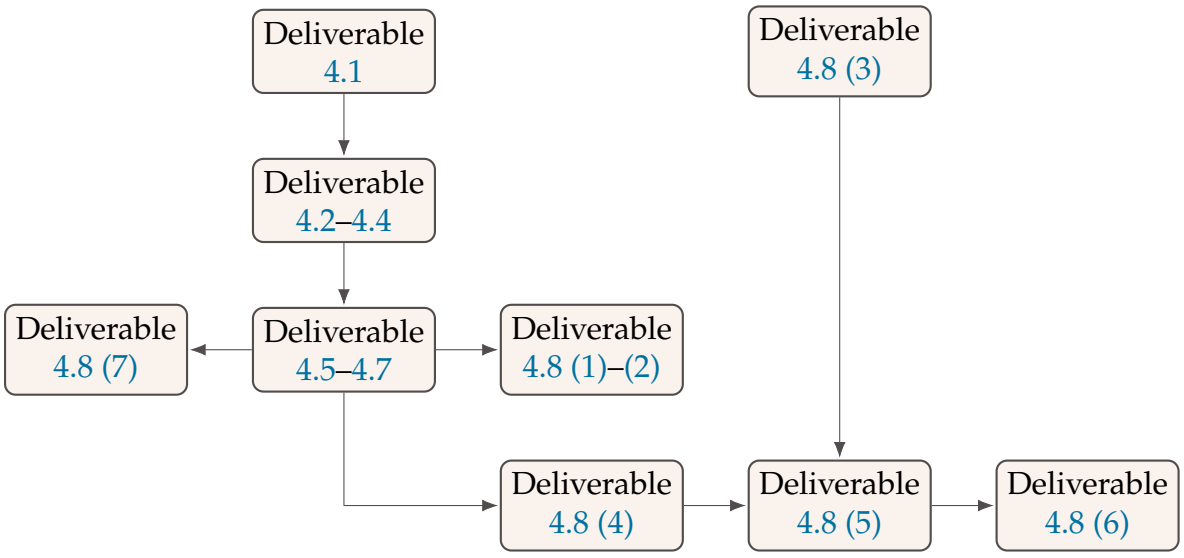
(1) **Learn** how the derivative and integral terms of a PID controller affect performance (desireably and undesireably).

(2) **Tune** a proportional-integral-derivative (PID) controller.

(3) **Learn** how to use the root locus to assist in control design.

(4) **Recognize** how zero dynamics can be introduced by controllers and how they can severely impact performance and analysis.

The deliverable dependency graph is



[2] In the robotics course you will learn to just cancel the nonlinearities out.

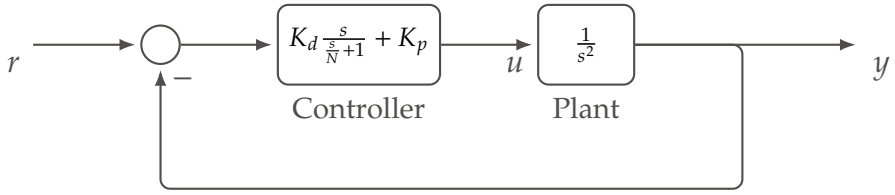
4.2 Experimental Procedure

This lab is split into three parts. There is only one Simulink model. In Part I you will analyze how the derivative term affects the stability of your system. In Part II you will analyze how the integral term affects the performance of your system. Finally in Part III you make an incremental change to your PID controller to improve your system's performance qualitatively. The plant for this lab will be

$$P(s) = \frac{1}{s^2}.$$

Part I: The Proportional Derivative (PD) Controller

The closed-loop system under PD control takes the form



where N is called the filter coefficient and K_p and K_d are the proportional (P) and derivative (D) gains. If you compute the closed-loop transfer function $G(s)$ from $r(t)$ to $y(t)$ you will find

$$G(s) = \frac{\left(NK_d + K_p\right) s + NK_p}{s^3 + Ns^2 + \left(NK_d + K_p\right) s + NK_p}.$$

As long as $N, K_p, K_d > 0$ the closed-loop system is stable. Note that we have a zero appearing in the transfer function. Assuming that the zero dynamics are fast, i.e. $K_p \gg K_d$, and we have an ideal differentiator, i.e. $N \to \infty$, we can approximate the system by the transfer function

$$\tilde{G}(s) = \frac{K_p}{s^2 + K_d s + K_p}.$$

It then should not surprise you that the intuition used for designing the gains in Lab 2 applies somewhat to the gains K_p and K_d. Of course, a real PD controller has the effects of a zero and the effects of the practical differentiator to consider. Your deliverable for this section is to

Deliverable 4.1

Choose and **record** a pair of gains K_p and K_d.

using the above intuition. The next procedure describes how to go about this.

Procedure 4.1

In this procedure you will determine a pair K_p and K_d.

(1) **Ensure** the loop of the Simulink model is closed as in the block diagram above.

(2) **Pick** a random $0 < K_p < 10$ and $0 < K_d < 10$.

(3) **Run** the "visualize_uav.m" MATLAB script and **inspect** Figure 2. If there are persistent oscillations in your response or if they don't dissipate around 30 s, **repeat** step (2) with a different choice. Aim to have oscillations that have a magnitude much less than 1 by 40 s. Use your understanding of Lab 3 to motivate whether you should increase or decrease K_d or K_p.

(4) Once you have settled on a choice for K_p and K_d, **run** the "visualize_uav.m" MATLAB script and **inspect** Figure 2. **Observe** that there is a constant steady-state error.

Part II: Introducing an Integral Component

The reference to the control system is 1 m. That is, engineers would like the drone to be able to perfectly track a constant distance away from the path. Unfortunately there is some steady-state error, so we do not achieve this specification. We would like to eliminate the observed steady-state error. How do we do this? First we must determine why this error appears. It is fair to ask: "If our plant already has an integrator, how is it that there is constant steady-state error?" In this simulation there is a constant *input* disturbance similar to the $d(t)$ of Lab 3. As you verified in Lab 3, even an integrator in the plant cannot perfectly reject this input disturbance and can only attenuate it under proportional error feedback. If we introduce a pole at the origin, i.e. an integrator, in the controller we can perfectly reject this error. Unfortunately, there is a constraint on how large the integrator term can be. Your deliverable for this part is to simply choose an integrator gain.

Deliverable 4.2

Choose and **record** a gain $K_i > 0$ that perfectly rejects the step disturbance while ensuring closed-loop stability.

To do this we will make use of the root locus method. As such, you are required to also produce a relevant root locus.

Deliverable 4.3

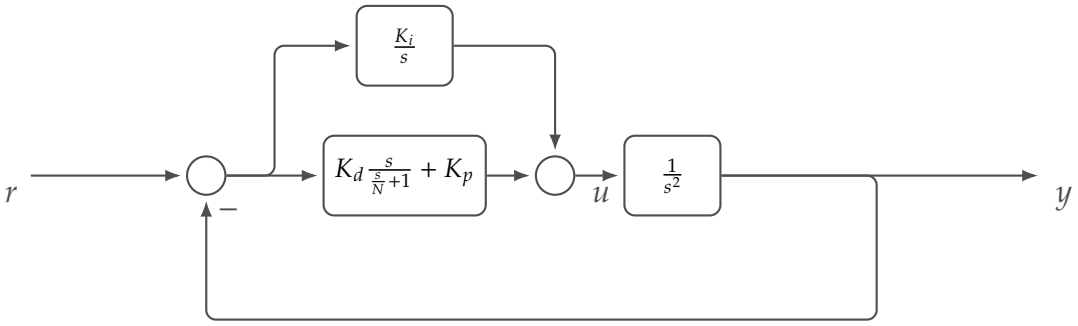
Acquire the root locus generated by the "part_2.m" MATLAB script. **Place** a cursor on a point where the system is unstable so that you recognize how large K_i is allowed to be. **Save** the figure.

Finally you will acquire the complete system response.

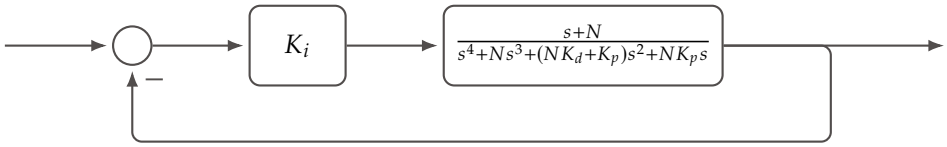
Deliverable 4.4

Acquire Figure 2 by running the "visualize_uav.m" MATLAB script and **save** it.

Your complete PID control system looks like



where you have determined N,K_d and K_p already. The closed-loop poles of this system[3] are the same as the poles of the closed-loop system



[3]Notice what I do not say. I am **not** saying the closed-loop zeros or the DC gain are the same.

It follows that if we plot the (positive gain) root locus of

$$\frac{s+N}{s^4+Ns^3+(NK_d+K_p)s^2+NK_ps}$$

and choose a gain so that the poles of this system meet our desired specifications, we can use this very same gain as K_i in our closed-loop system to produce the same poles that meet the same specifications. The next procedure guides you through these steps.

Procedure 4.2

In this procedure you will produce a root locus and make an initial choice for the gain K_i.

(1) **Modify** the "part_2.m" MATLAB script so that the variables `Kp` and `Kd` match your chosen gains from Part I. **Run** the "part_2.m" MATLAB script. **Inspect** Figure 3.

(2) **Observe** that if K_i is too large, the system will be rendered unstable. **Place a cursor** on one point of the root locus where the system is marginally stable (crosses from stable to unstable). **Save** this Figure for Deliverable 4.3.

(3) **Take note** of the gain depicted by your cursor in the previous step. If K_i takes on a value larger than this gain, your system will be rendered unstable!

(4) **Pick any** value of K_i between 0 and the maximum gain defined by the previous step.

(5) **Read** the cursor to find out what the gain value is. **Record** this gain value as the value you choose for K_i producing Deliverable 4.2.

Now that you have chosen K_i you are ready to test out whether it worked. There is no deliverable for the next procedure as it is only a verification that the previous procedures went smoothly.

Procedure 4.3

You will now verify your choice of K_i.

(1) **Set** the integrator gain with your chosen value of K_i.

(2) **Run** the "visualize_uav.m" MATLAB script and **inspect** Figures 1 and 2. **Observe** that the UAV continues to track the path but now does so without steady-state error. **Save** Figure 2 for Deliverable 4.4.

Part III: Tune your Controller

Now that we have eliminated the steady-state error, you may observe more extreme oscillations, large overshoot or other unusual behaviour! In this part, you will make an attempt to improve the performance of your controller in any way you desire. Your deliverable for this section is

Deliverable 4.5

Acquire the root locus plots generated by the "part_3.m" MATLAB script with your PID gains determined at the end of Part II. **Save** the figures.

and you will use this information, alongside your other intuitions, to guide another choice of gains.

Deliverable 4.6

Determine a new set of gains K_p, K_i and K_d that in some way improve the performance of your UAV. This may be in reduced overshoot, or reduced settling time, for example. You may consider any characteristic you consider undesireable.

Finally, you will acquire and save Figure 2 generated by "visualize_uav.m" to prove you have achieved perfect tracking.

Deliverable 4.7

Acquire Figure 2 by running the "visualize_uav.m" MATLAB script and **save** it to depict the improved performance. Attach readings to your plot to substantiate the improved response dynamics. If there is no improvement, **acquire** at least three simulations (of three different gain choices) producing Figure 2 and **save all of them**. **Record** the changes you made.

Procedure 4.4

There is no explicit procedure to tune your controller but this high-level series of steps can help you make your initial changes.

(1) **Modify** the "part_3.m" MATLAB script so that the variables `Kp`, `Kd` and `Ki` match the gains you chose in Part II. **Run** the "part_3.m" MATLAB script. **Save** the figures (4, 5, 6) for Deliverable 4.5.

(2) **Inspect** the root locus plots generated. Each root locus depicts your current closed-loop poles with filled black circles. You can therefore see how changing any gain would change the location of the closed-loop poles and thereby change the behaviour of your final closed-loop system. You can also see exactly what a pole will contribute to your system by clicking on the root locus to produce a cursor.

(3) Guided by the location of the poles currently and the root locus plots produced, **decide** on a gain to change to improve at least one characteristic of your system. **Make** the change to that gain in the Simulink model.

(4) **Acquire** Figure 2 by running the "visualize_uav.m" MATLAB script and **verify** that you did improve the characteristic you set out to improve. If successful, **save** it for Deliverable 4.7 and **record** your gain choices for Deliverable 4.6. If not successful explore the range of parameters available to see what changes you can make happen. *If no improvement is possible, save three simulations (Figure 2, in particular) of **at least three distinct choices you trialed**. **Record** what those choices were.*

4.3 Report Deliverable

Another lab nearly complete! As usual, you are expected to submit a report demonstrating that you completed the lab and that you understand the tasks performed.

Preface your lab report with the Declaration of Authorship.

In addition to including

- Deliverable 4.1,
- Deliverable 4.2,

- Deliverable 4.3,
- Deliverable 4.4,
- Deliverable 4.5,
- Deliverable 4.6 and
- Deliverable 4.7

in your report, you are required to answer the questions of the following deliverable. Make sure to leverage your other deliverables in your answers!

Deliverable 4.8

(1) Like in Lab 3, the primary output was a (relative) position $y(t)$. Unlike Lab 3, we do not observe the velocity explicitly but, instead, use the (practical) differentiator in the PID controller to estimate velocity (derivative of the relative position is velocity) and apply that in closed-loop feedback. Unfortunately, the explicit use of a differentiator in parallel to the proportional error feedback introduced zeros in the final transfer function $G(s)$ from $r(t)$ to $y(t)$. In particular

$$G(s) = \frac{(NK_d + K_p)s^2 + (NK_p + K_i)s + NK_i}{s^4 + Ns^3 + (NK_d + K_p)s^2 + (NK_p + K_i)s + NK_i}.$$

For your final choice of K_p, K_i and K_d, **determine** where the zeros of $G(s)$ are located. *You can inspect the MATLAB scripts or Simulink model to find the filter coefficient N.*

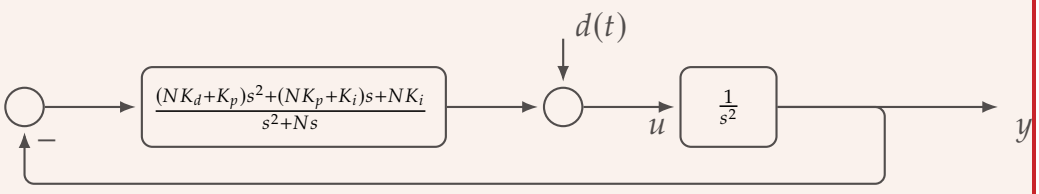
(2) If the zeros are sufficiently fast (i.e. larger negative real part) compared to the poles, we can approximate the behaviour of the system as simply the system where the zeros are evaluated at $s = 0$. You may assume that "sufficiently fast" means that the zeros is at least five times farther away in real part than any of the poles.

Using the zeros you determined in (1), **determine** if the zeros affected your system's response. **Acquire** the step response of the $G(s)$ in (1) and of

$$\tilde{G}(s) = \frac{NK_i}{s^4 + Ns^3 + (NK_d + K_p)s^2 + (NK_p + K_i)s + NK_i}.$$

[a] Compare these and determine **where** in the output response (Figure 2 generated by the last procedure) would the zeros affect your system.

(3) The PD controller of Part I produced a steady-state constant error. This was explained, in Part II, as being due to a step input disturbance to the plant $P(s)$. **Prove** that if $K_i \neq 0$ then step input disturbances to the plant are perfectly rejected. To do so, first find the transfer function from $d(t)$ to $y(t)$ in the diagram



and use this transfer function to show that the output $y(t)$ under the step input $d(t) = A\mathbf{1}(t)$, for $A \in \mathbb{R}$, converges to zero.

(4) Using the root loci of Deliverable 4.5, **explain** how the gains K_p, K_i and K_d theoretically affect the performance (overshoot, settling time, time-to-peak) of your system as you vary them. Ensure you discuss each gain as well as discuss if those gains affect the qualities you discuss universally (does increasing a certain gain always result in the quality improving?).

(5) The engineers at VenX claim that there are no input disturbances. Input disturbances, in this context, are caused by poor mechanical trimming of the aircraft (it doesn't "naturally" fly straight). Suppose you agreed with this statement[b]. However, they argue that you should include the integrator term *anyway* since they claim it improves the performance of your controller by guaranteeing that you converge onto the path (no steady-state error).

Using the statement you proved in (3), any other relevant deliverables, your experience in this lab and the definition of $P(s)$, **determine** if it is true that you should keep the integrator term and **explain** your reasoning. Ensure to explain both the positives and negatives of including an integrator term.

Note: Full marks are given to answers that correctly identify all the impacts of introducing an integrator.

(6) Repeat question (5) except assuming that the plant was

$$P(s) = \frac{s+1}{s^2+2s+3}.$$

instead of $\frac{1}{s^2}$. In what way would it change your answer?

(7) **State** what characteristic you aimed to improve in Part III. **Why** did you choose to change the gains you changed? A theoretical justification is not required but you should explain yourself clearly. If no change improved the characteristic, explain what changes you tried, why you tried them, and why you think they did not work.

[a] I provide a script `q2.m` to help you do this.
[b] at the end of the day, you aren't a mechanical/aerospace engineer

Grading Scheme

The grading scheme is shown in Table 4.1. The breakdown of your grade is shown per deliverable except in the case of the lab questions where it is shown per question.

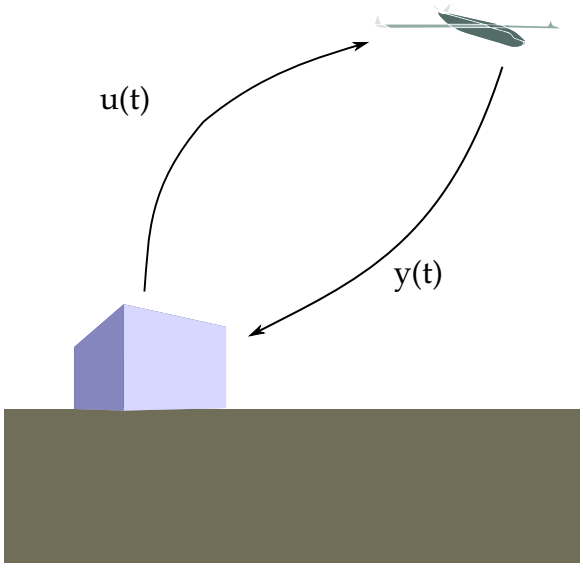
Table 4.1: Grading scheme for Lab 4.

	Deliverable	Marks
	4.1	2
	4.2	2
	4.3	2
	4.4	2
	4.5	3
	4.6	6
	4.7	3
Lab Subtotal		20
	4.8 (1)	2
	4.8 (2)	2
	4.8 (3)	4
	4.8 (4)	9
	4.8 (5)	5
	4.8 (6)	2
	4.8 (7)	3
Report Subtotal		27
Total		47

LAB 5

Lead and Lag Compensator Design

Your exploration into the line-following, unmanned aerial vehicle (UAV), developed in Lab 4, convinced VenX engineers that the inner-outer loop structure, described in Lab 3, was much easier to work with. This is the design they went with instead[1].



Recall that the drone achieved perfect step-tracking of a desired *distance* away from a line segment in space. In other words, if the input to that closed-loop system was equal to $2 \cdot \mathbf{1}(t)$ then the drone would track the line segment with an error of 2 m

[1] In `ECE 488` and `ECE 481`, you will learn about an idea known as state feedback. This is how I did this design.

in steady-state. Engineers did this to provide the ability to tune an offset from the pre-planned path stored in the drone.

Engineers would like to use this input to adjust the distance remotely with a control law located at a remotely deployed ground station. The ground station monitors weather patterns and information relayed by specialists on Earth to decide on a desired offset from the preplanned path. The station then communicates with the drone. The station sends reference commands and receives an observed output distance from the path. The observed output distance is used in closed-loop feedback on the ground station to update what the commanded reference offset should be.

Normally, proportional error feedback would preserve the stability of an already stable *second-order* system. Unfortunately, communication delays can actually destabilize such a system. Your task is to design a control law that achieves the desired specifications:

- track a constant reference with 2% error and
- preserve closed-loop stability under the presence of communication delays.

5.1 Objectives

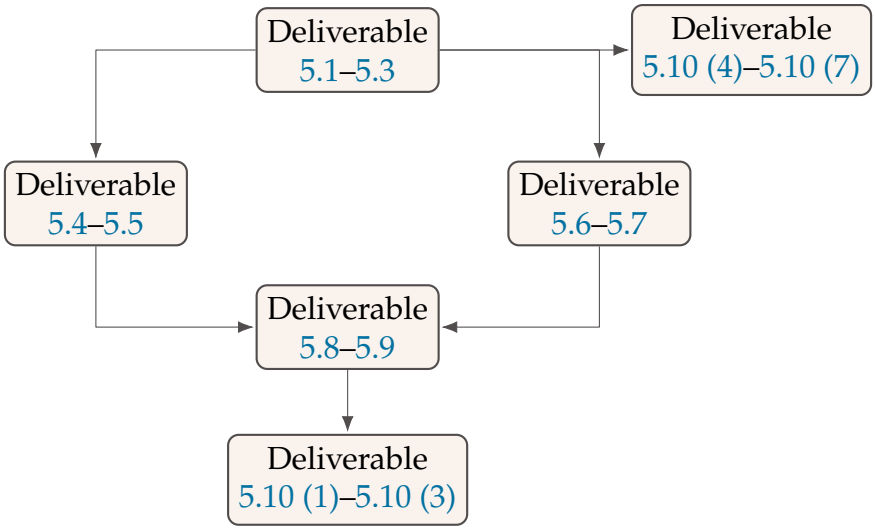
The goals of this lab are to:

(1) **Practice** the Lead and Lag Design Procedures as you've learned them in your course.

(2) **Learn** how delays in the loop can affect stability.

(3) **Learn** how a phase margin specification determines the maximum delay $L(s)$ can afford before losing stability.

The deliverable dependency graph is the following:



5.2 Experimental Procedure

This lab is split into four parts. In Part I you will calculate a proportional error gain K to achieve the tracking specification and then observe how closed-loop stability is lost upon introducing a delay. In Part II and Part III you will design a lag and lead compensator respectively, using the Bode plot acquired in Part I, to stabilize the closed-loop system even in the presence of the delay. You will verify this using a Nyquist plot. Finally in Part IV you will observe the lag and lead compensators stabilize the simulated system.

Preamble: Delays

A "perfect" delay of τ units of time from an input signal $u(t)$ to an output signal $y(t)$ is determined by the static equation

$$y(t) = u(t - \tau)$$

The value of the output y at time t is given by the value of the input signal u at a time τ units prior. Take the Laplace transform of both sides[2] and simplify to find

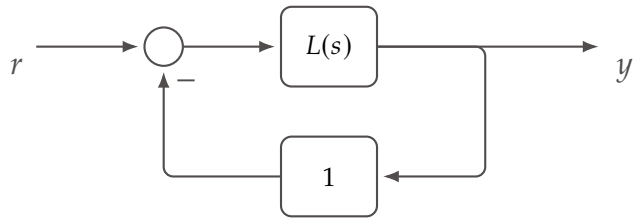
$$Y(s) = e^{-\tau s} U(s)$$

It is in this sense that one can view $e^{-\tau s}$ as the transfer function of the delay operator[3]. This gives a frequency-domain version of the time delay. Suppose a

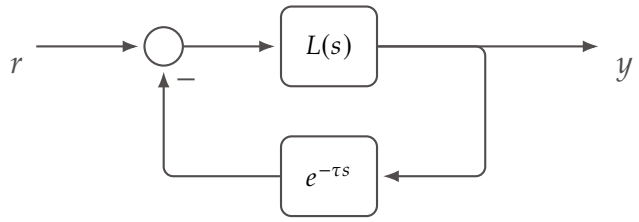
[2] and suppose the integrals converge.

[3] This is an irrational transfer function. Take `ECE 481` to see how we can approximate this transfer function and incorporate into a more formal design philosophy.

loop transfer function $L(s)$ satisfies the Nyquist criterion. Specifically imagine that the closed-loop system



is stable. Is it true that the closed-loop system remains stable under a delay of τ seconds?
Recognizing that $e^{-\tau s}$ is the transfer function of the delay operator, we could equivalently ask if the closed-loop system



is stable. But how do we check the stability of this system?
Since the closed-loop transfer function is not composed solely of real rational polynomials, we *cannot* use the Routh-Hurwitz stability criterion. Fortunately, the Nyquist criterion still applies[4]. If you compare the Nyquist plot of $L(s)$ with the Nyquist plot of $e^{-\tau s}L(s)$ you should observe some similarities; in fact, a number of points on the latter Nyquist plot look like a rotated version of the former Nyquist plot. This is depicted by Figure 5.1. This notion, if put formally[5], allows us to translate the earlier delay specification into the following phase margin specification:

> The loop transfer function $L(s)$ has a phase margin strictly greater than $\tau\omega_{\text{gc}}$ where ω_{gc} is the gain-crossover frequency in rad/s and τ is the delay in the loop in s.

This is the primary specification you will meet in this lab to accommodate the delay. Note that this specification only makes sense when $L(s)$ is already stable in closed-loop.

[4]Why? That is probably outside the scope of this course. Pursue a MASc in control theory and you'll find out.

[5]If you do not see why, do not be alarmed. This lab will directly translate the specification for you.

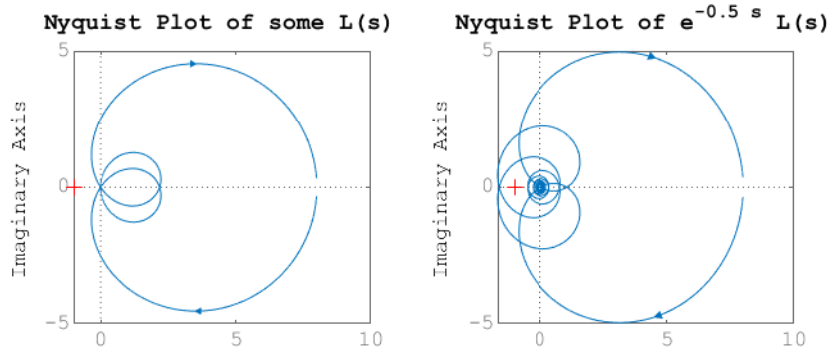


Figure 5.1: Nyquist plots of an unknown loop transfer function $L(s)$ (left) and of the same transfer function delayed by 500 ms (right).

In practice, attempting to change the phase margin can also result in a changed gain-crossover and thereby change what delays $L(s)$ can afford. The lag compensator in particular changes the gain-crossover in order to increase the phase margin. The lead compensator often results in a slightly higher gain-crossover frequency.

Part I: Delays and Stability

Consider the simple proportional feedback loop where $P(s)$ are the dynamics of the controlled line-following drone.

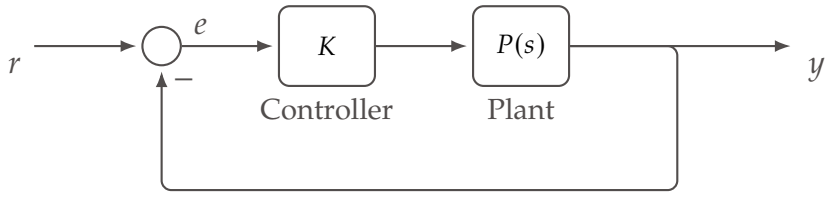


Figure 5.2: Block diagram of proportional controller and plant without delay.

The controller is designed in two stages: first the gain K is determined, and then the dynamic portion of the compensator is designed. Your first task is to

Deliverable 5.1

Calculate the loop gain $K > 0$ so that the steady-state step tracking error is less than or equal to 2%. From the range of gains K, **choose** a value for K that lies within 10% of the smallest gain that satisfies the steady-state step

tracking error requirement.

You will then acquire a Nyquist plot to verify stability of the closed-loop system when there are no delays as well as acquire a Nyquist plot to demonstrate instability when the delay is introduced.

Deliverable 5.2

Acquire two Nyquist plots (using the Model Linearizer app) so that

- one demonstrates the stability of the gain-compensated closed-loop system when there is no delay and
- the other demonstrates the instability of the gain-compensated closed-loop system with a delay.

Finally, you will need to acquire a Bode plot of the undelayed open-loop system in order to perform the lead and lag design procedures.

Deliverable 5.3

Acquire a Bode plot (using the Model Linearizer app) of the undelayed open-loop system. **Place** a cursor at the gain-crossover frequency on the magnitude plot.

To complete these tasks, follow the next procedure.

Procedure 5.1

In this procedure you will determine a gain K and acquire the required Nyquist plots. You will also get a chance to simulate the system.

(1) Assume the plant $P(s)$ has a DC gain 1. **Calculate** a gain K so that the steady-state tracking error of the system shown in Figure 5.2 for a unit-step input is less than or equal to 2%.From the range of gains K, **choose** a value for K that lies within 10% of the smallest gain that satisfies the steady-state step tracking error requirement.

(2) **Change** the variable `K` *in the MATLAB script* "procedure_5_1.m" to match your chosen gain. **Run** the script.

(3) **Open** the "Lab_5_Model_System.slx" Simulink model and **ensure** that

the switch is pointed towards the path without the "Delay."

(4) The model should already be correctly configured with the "Open-Loop Input" set to the signal $e(t)$ and the "Open-Loop Output" set to signal $y(t)$. **Verify** this.

(5) **Open** the Model Linearizer App and **acquire** a Nyquist plot. This produces part of Deliverable 5.2. Assuming $P(s)$ has no unstable poles, **verify** that you have closed-loop stability.

(6) **Open** the Model Linearizer App and **acquire** a Bode plot. **Identify** the 0 dB gain-crossover and the phase margin. **Place** a cursor on the gain-crossover frequency on the magnitude plot. This completes Deliverable 5.3.

(7) If you are curious, **run** "procedure_5_1_simulate.m" to see how the closed-loop system behaves without a delay.

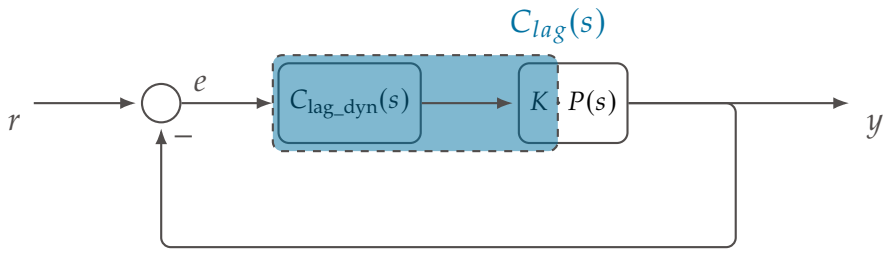
(8) **Open** the "Lab_5_Model_System.slx" Simulink model and **ensure** that the switch is pointed towards the path **with** the "Delay."

(9) **Open** the Model Linearizer App and **acquire** a Nyquist plot. This completes Deliverable 5.2. Assuming $P(s)$ has no unstable poles, **assess** how did the closed-loop stability change.

(10) If you are curious, **run** "procedure_5_1_simulate.m" to see how the closed-loop system behaves with the delay. Expect the system to behave undesirably; it is even possible that the simulation stops early if the drone travels backwards!

Part II: Design a Lag controller

One way to increase the phase margin of a loop transfer function is to introduce a lag controller. Lag controllers decrease the gain so as to change the gain-crossover; changing the gain-crossover in turn changes the phase margin. In this part the closed-loop system takes the form



where

$$C_{lag}(s) = K \cdot C_{lag\_dyn}(s) = K \cdot \frac{\alpha T s + 1}{T s + 1}, \quad with \quad 0 < \alpha < 1,\ T > 0.$$

is a lag compensator. Note that dynamic portion of the lag compensator, $C_{\text{lag\_dyn}}$ has 0 dB DC gain and K is the compensator gain determined in Deliverable 5.1. Your task is to find α and T, by completing the lag compensator design.

Deliverable 5.4

Let ω_{gc} denote the 0 dB gain-crossover frequency (in rad) of $K \cdot P(s)$.; this is the value you determined using the Bode plot from Deliverable 5.3. **Design** a lag compensator (determine α and T) so that the Bode plot of the compensated open-loop system has a phase margin strictly greater than $\frac{12\omega_{gc}}{100}$ rad.
You must show the use of the design equations for full marks. If your design relies on readings from a Bode plot, include the specific plot to show where your design values came from.
Note that the lag compensator design steps and/or notations from the lecture notes may differ from the notation used in the lab manual and associated scripts. You are encouraged to use the design procedure from your lecture notes. Present your resulting compensator parameters in **the same format** as shown in the lab manual.

You will then acquire a Nyquist plot to demonstrate that the delayed system has been stabilized with the lag compensator.

Deliverable 5.5

Acquire a Nyquist plot demonstrating the lag-compensated system *with delay* is closed-loop stable.

Procedure 5.2

In this procedure you will design a lag compensator, using the content you learned in your course, and simulate the delayed closed-loop system.

(1) The engineers at VenX have determined that the compensated closed-loop system must have a phase margin strictly greater than

$$\frac{12\omega_{gc}}{100} \text{ rad.}$$

where ω_{gc} is the gain-crossover frequency found for $K \cdot P(s)$ in Deliverable 5.3.

(2) Following the lag compensator design steps from lectures, **determine** α and T such that the lag-compensated system meets the phase-margin specification. This produces Deliverable 5.4.

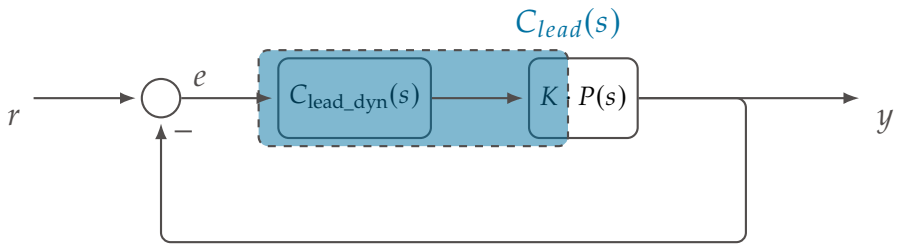
(3) **Change** the variable `alpha` and `T` *in the MATLAB script* "procedure_5_2.m" to match your chosen parameters. **Run** the script.

(4) **Open** the "Lab_5_Model_System.slx" Simulink model and **ensure** that the switch is pointed towards the path **with** the "Delay."

(5) **Open** the Model Linearizer App and **acquire** a Nyquist plot. This completes Deliverable 5.5. Assuming $P(s)$ has no unstable poles, **verify** that you do have closed-loop stability.

Part III: Design a Lead controller

Another way to increase the phase margin of a loop transfer function is to introduce a lead compensator. Lead compensators add phase near the gain-crossover frequency. In this part the closed-loop system takes the form



where

$$C_{lead}(s) = K \cdot C_{lead\_dyn}(s) = K \cdot \frac{\alpha T s + 1}{T s + 1}, \quad with \quad \alpha > 1,\ T > 0.$$

is a lead compensator. Note that the dynamic portion of this compensator, $C_{\text{lead\_dyn}}$, has a unit DC gain and K is the compensator gain determined in Deliverable 5.1. Your task is to find α and T in terms of a phase margin specification and the Bode plot of $K \cdot P(s)$ — determined by Deliverable 5.3 — to complete the lead compensator design.

Deliverable 5.6

Let ω_{gc} denote the 0 dB gain-crossover frequency (in rad) of $K \cdot P(s)$; this is the value you determined using the Bode plot from Deliverable 5.3. **Design** a lead compensator (determine α and T) so that the Bode plot of the compensated open-loop system has a phase margin strictly greater than $\frac{12\omega_{\text{gc}}}{100}$ rad.
You must show the use of the design equations for full marks. If in the design process you used readings from a Bode plot, include the specific plot to show where your design values came from.
Note that the lead compensator design steps and/or notations from the lecture notes may differ from the notation used in the lab manual and scripts. You are encouraged to use the design procedure from your lecture notes. Present your resulting compensator parameters in **the same format** as shown in the lab manual.

You will then acquire a Nyquist plot to demonstrate that the delayed system has been stabilized with the Lead compensator.

Deliverable 5.7

Acquire a Nyquist plot demonstrating the lead-compensated system *with delay* is closed-loop stable.

Procedure 5.3

In this procedure you will design a lead compensator, using the content you learned in your course, and simulate the delayed closed-loop system.

(1) The engineers at VenX have determined that the compensated closed-

loop system must have a phase margin strictly greater than

$$\frac{12\omega_{\text{gc}}}{100} \text{ rad.}$$

Using the Bode plot for $K \cdot P(s)$ in Deliverable 5.3, **determine** the required phase addition to meet this specification.

(2) Following the lead compensator design steps from lectures, **determine** α and T to meet this phase-margin specification. This produces Deliverable 5.6.

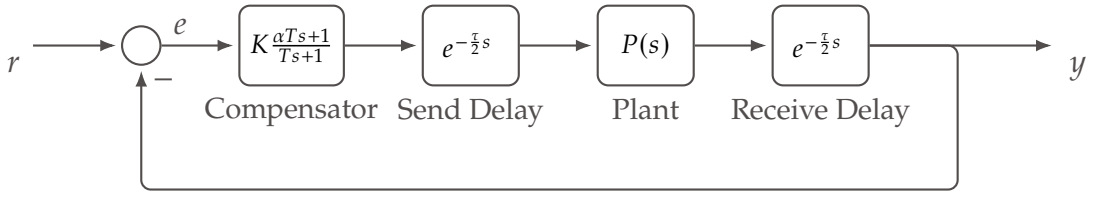
(3) **Change** the variables `alpha` and `T` *in the MATLAB script* "procedure_5_3.m" to match your chosen parameters. **Run** the script.

(4) **Open** the "Lab_5_Model_System.slx" Simulink model and **ensure** that the switch is pointed towards the path **with** the "Delay."

(5) **Open** the Model Linearizer App and **acquire** a Nyquist plot. This completes Deliverable 5.7. Assuming $P(s)$ has no unstable poles, **verify** that you do have closed-loop stability.

Part IV: Simulating the Real System

In Parts II and III you designed and verified lag and lead compensators that ensure closed-loop stability of a plant with delay. This was verified using the Nyquist plots of Deliverable 5.5 and 5.7. In this part, you will simulate the real closed-loop system with delays. The true closed-loop system takes the form



The "Send Delay" captures the amount of time the plant has to wait to see an update from the compensator on the ground while the "Receive Delay" captures the amount of time the compensator has to wait to see an updated output. From a networking perspective, this is lower bounded by the ping time[6]. If you designed your Lag and Lead compensators correctly in Parts II and III, then you should expect

[6] since it ignores any computational delays.

them to perform adequately. Remember, stability is the primary characteristic of concern in this discussion. Do not be alarmed if the settling time or overshoot is undesirable.

There are two deliverables for this part. First you will simulate the real system against the lag compensator to produce

Deliverable 5.8

Acquire Figure 542 generated by "procedure_5_4_simulate.m."

Second, you will simulate the real system against the lead compensator to produce

Deliverable 5.9

Acquire Figure 552 generated by "procedure_5_5_simulate.m."

To do so, follow the next couple procedures.

Procedure 5.4

In this procedure you will simulate the real closed-loop system with your lag compensator.

(1) **Run** "procedure_5_4_simulate.m."

(2) **Save** Figure 542. **Verify** the simulation ran to completion (240 s) and that the error settles. If there are oscillations, they should decay. This produces Deliverable 5.8.

Procedure 5.5

In this procedure you will simulate the real closed-loop system with your lead compensator.

(1) **Run** "procedure_5_5_simulate.m."

(2) **Save** Figure 552. **Verify** the simulation ran to completion (240 s) and that the error settles. If there are oscillations, they should decay. This produces Deliverable 5.9.

5.3 Report Deliverable

You are almost done the lab component of this course! As usual, you are expected to submit a report demonstrating that you completed the lab and that you understand the tasks performed.
Preface your lab report with the Declaration of Authorship.
In addition to including

- Deliverable 5.1,
- Deliverable 5.2,
- Deliverable 5.3,
- Deliverable 5.4,
- Deliverable 5.5,
- Deliverable 5.6,
- Deliverable 5.7,
- Deliverable 5.8,
- Deliverable 5.9,

in your report, you are required to answer the questions of the following deliverable. Make sure to leverage your other deliverables in your answers!

Deliverable 5.10

(1) In the lead and lag design procedures you may have had to add more phase (or alter the gain-crossover location) than what the phase margin specification demanded. Acquire a Bode plot of your lead and lag compensators to **explain why** this can happen. Explain what the other side-effects of the lead/lag controllers are besides their intended effects (e.g. does the lead controller only add phase?).

(2) **Describe** the qualitative differences between the final closed-loop response of your lag-compensated and lead-compensated systems. Touch on characteristics like

- the overshoot,
- the settling time and

- the time to peak.

Use the simulations in Deliverables 5.8 and 5.9 in your discussion.

(3) **Describe** how you would choose between a lead compensator or lag compensator.

Ensure to leverage the Nyquist plots in Deliverables 5.5 and 5.7 as well as your answers to (1) and (2) in your discussion.

(4) Using the uncompensated phase margin ϕ and gain-crossover ω_{gc} of $K \cdot P(s)$ — both of which can be found by looking at the Bode plot of Deliverable 5.3 — **compute** the maximum delay the uncompensated system could afford before becoming unstable.

(5) A local WiFi network usually sees a round trip time for network packets of around 1 ms to 3 ms.

Would your uncompensated system $K \cdot P(s)$ be stable if the feedback loop involved communication over a similar wireless network with comparable delays? If not, would the compensated system be stable?

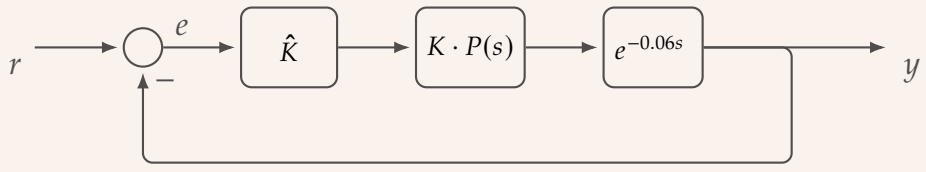
(6) The delay in the real system of Part IV was a total of 60 ms: there was a send delay of 30 ms and a receive delay of 30 ms.

In 4G/LTE communication networks the mean round trip time of a packet is around 50 ms.

Would your uncompensated system be stable if the feedback loop involved communication over a similar wireless network with comparable delays? If not, would the compensated system be stable?

(7) Using the Nyquist plot of the *delayed* system acquired in Deliverable 5.2, **determine** if there exists an additional proportional feedback gain $\hat{K} \in \mathbb{R}$ that would stabilize the system $K \cdot P(s)$ even in the presence of the 60 ms delay.

To be clear: Can you find a gain $\hat{K}$ that renders the closed-loop system



stable?

If it exists, does this closed-loop system have a larger or smaller steady-state error in comparison to the undelayed $KP(s)$.

What does this suggest about the effect of delays on performance?

Grading Scheme

The grading scheme is shown in Table 5.1. The breakdown of your grade is shown per deliverable except in the case of the lab questions where it is shown per question.

Table 5.1: Grading scheme for Lab 5.

	Deliverable	Marks
	5.1	2
	5.2	4
	5.3	2
	5.4	8
	5.5	2
	5.6	8
	5.7	2
	5.8	2
	5.9	2
Lab Subtotal		32
	5.10 (1)	3
	5.10 (2)	3
	5.10 (3)	2
	5.10 (4)	3
	5.10 (5)	3
	5.10 (6)	6
	5.10 (7)	6
Report Subtotal		26
Total		58

APPENDIX A

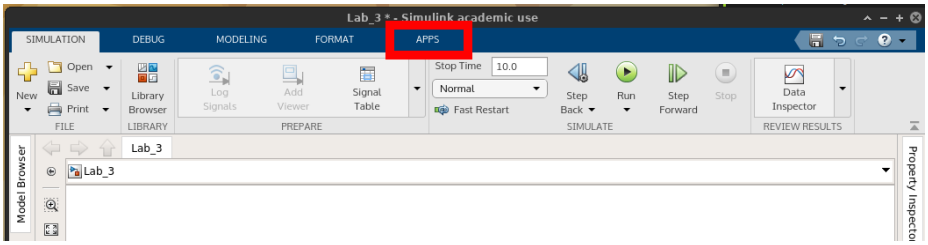
On Simulink

A.1 Model Linearizer Tools

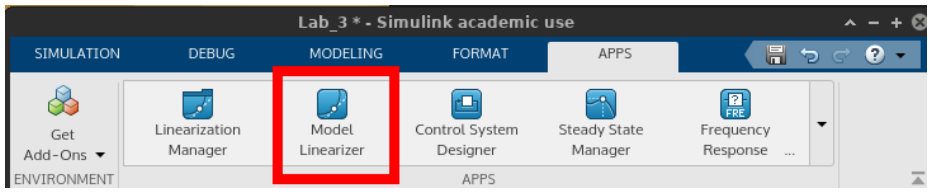
The model linearizer tools is an "App" in Simulink that allows an engineer to (1) linearize a system and (2) perform linear system analyses such as Bode plots and Step Responses. We won't be needing the first feature since our systems are already linear. This section is a brief tutorial on how to use the tools to acquire step responses, Bode plots and Nyquist plots.

The Model Linearizer App

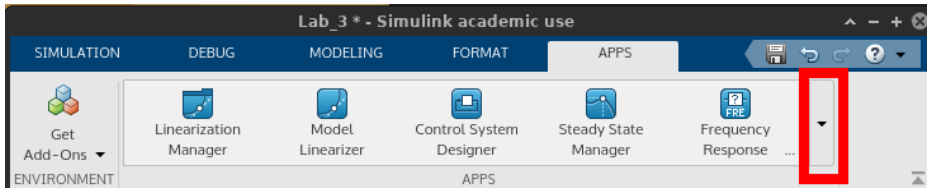
To open the Model Linearizer App, look to the top of your Simulink Model window and click the "`APPS`" button like so



Then proceed to opening the application by clicking the "`Model Linearizer`" button



If you cannot find it located in the toolbar depicted above, press the dropdown on the far right of the toolbar



to view a larger list of options. If you still cannot find it, you must not have the `Simulink Control Design` toolbox installed. Ensure you've installed the toolboxes listed in the Introduction. Upon opening the Model Linearizer app, you will have a window that looks like Figure A.1.

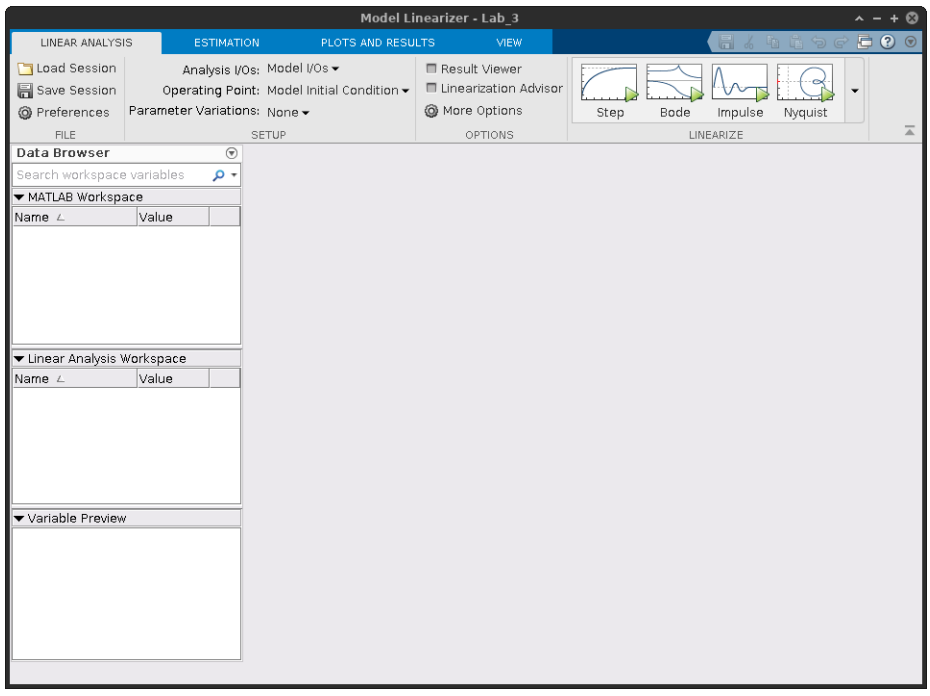
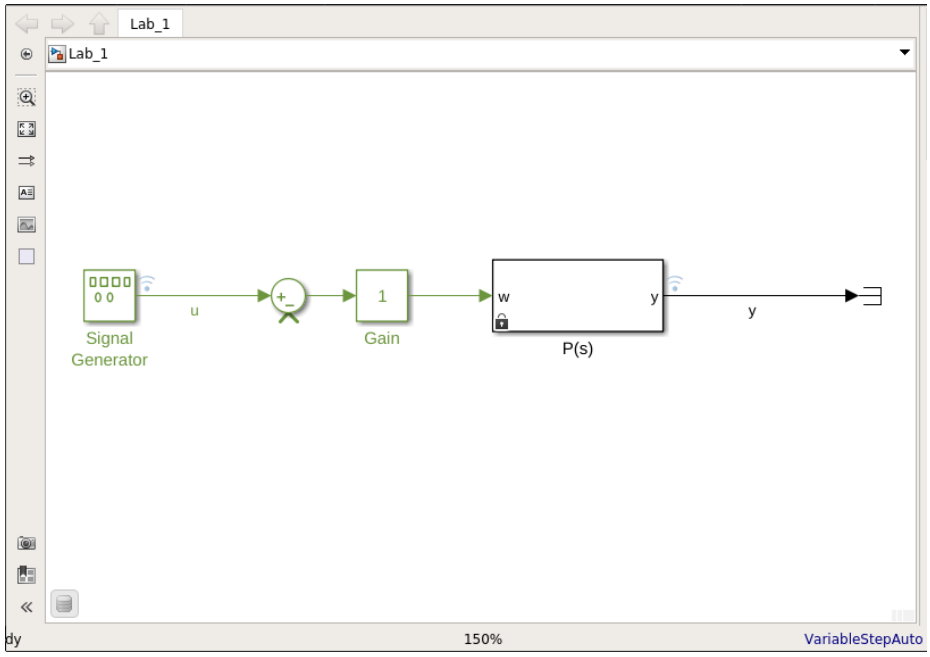


Figure A.1: What the Model Linearizer application looks like when you first open it.

Setting up your System: Input to Output

Now that you know how to open up the Model Linearizer, we must prepare the system we want to analyze. The Model Linearizer tool doesn't know what we consider an "input" and what we consider an "output" just from our Simulink diagram. But you do! Whenever you want to take a step response, you know where you input the step and what measurements you want to observe. The goal of the following steps is to show you how to give the Model Linearizer App this information.

First pull up the Simulink model of concern. I'll use the Lab 1 diagram as an example,



Suppose we wanted to analyze the relationship between the signals labelled u and y. Right-click the signal wire labelled u and hover over "Linear Analysis Points" as depicted in Figure A.2. This context menu has a number of options of which we will only use two. Since we would like to indicate to the Model Linearizer App that this wire is where the input appears, we select "`Input Perturbation`." We repeat the *same exact* process for the desired output signal wire y except we select "`Output Measurement`." The result is depicted in Figure A.4. *Note the little annotations above the signal.* A "down" arrow denotes an input and an "up" arrow denotes an output signal. You will have to repeat this process whenever you want to change where you apply your input or when you want to observe a different output. Note that the placement of the "`Input Perturbation`" and "`Output Measurement`" annotations

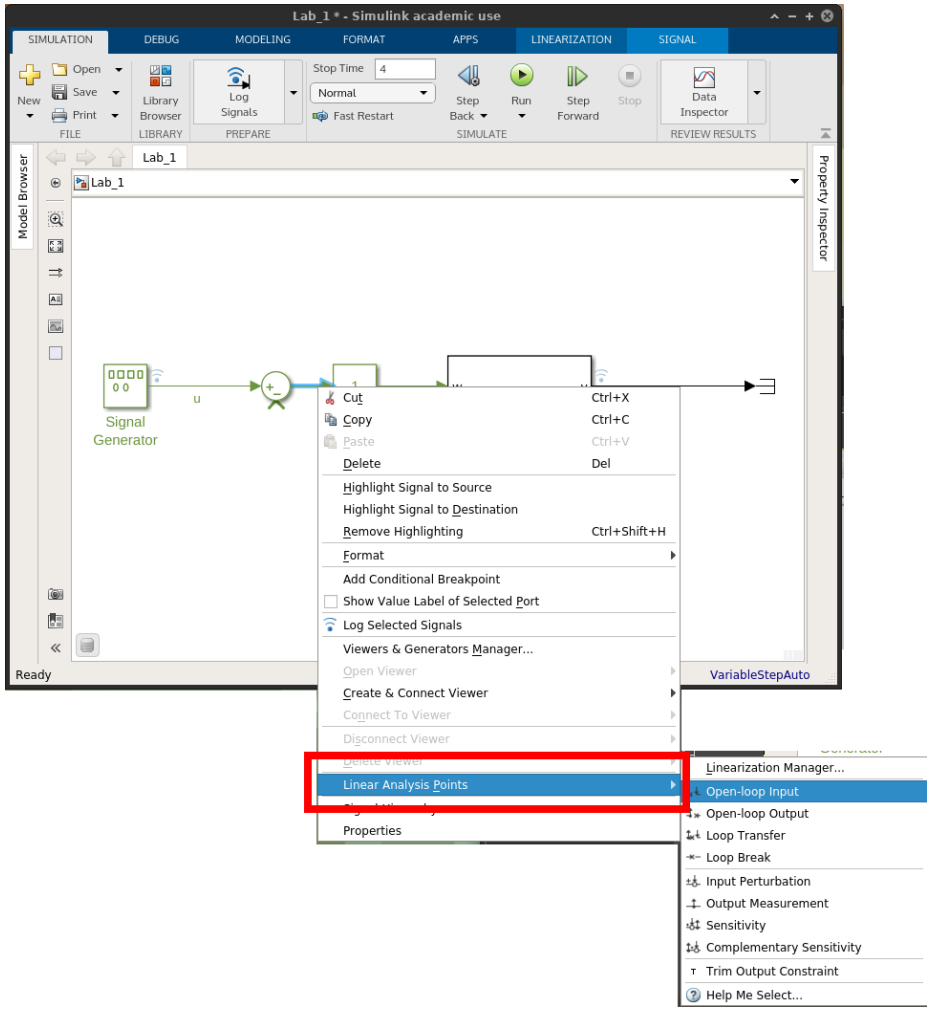


Figure A.2: Signal context menu and Linear Analysis points submenu.

is not arbitrary; it is intentional! This allows us to quickly change the configuration of the system, closing the loop for example, and get closed-loop measurements without having to change our annotations!

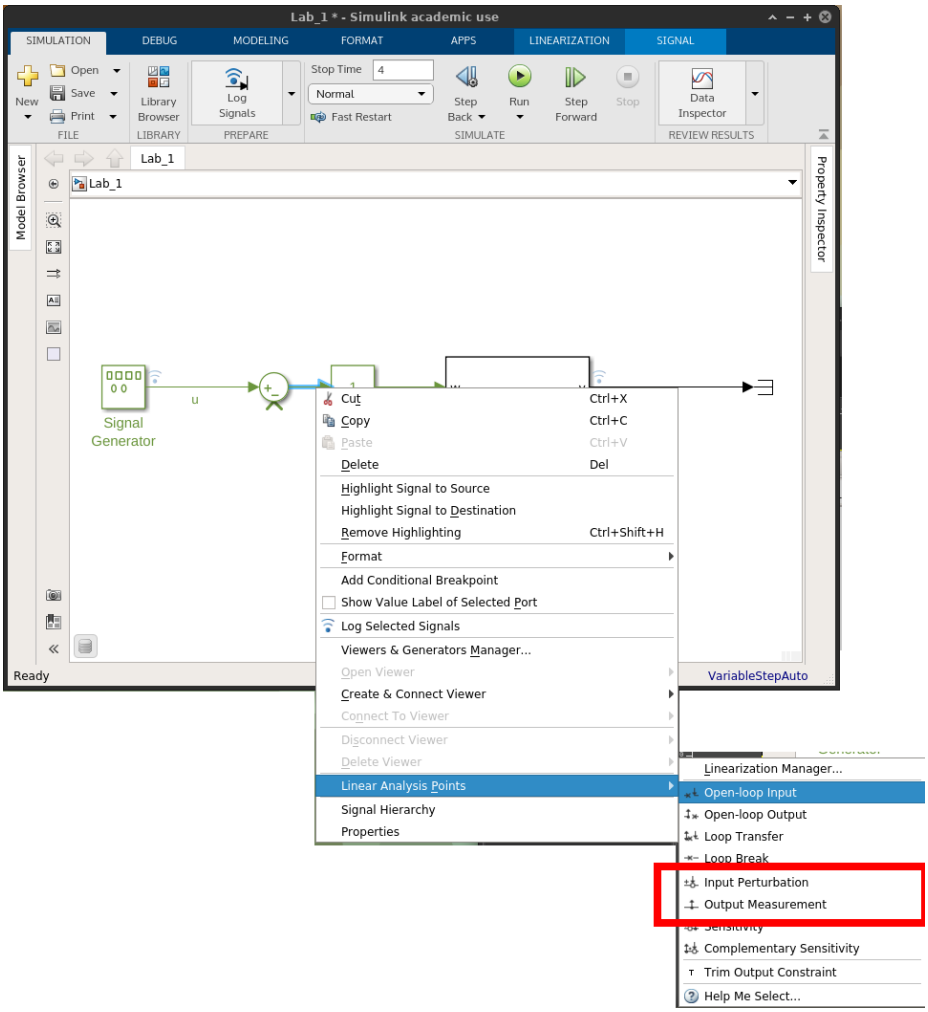


Figure A.3: The Linear Analysis context submenu.

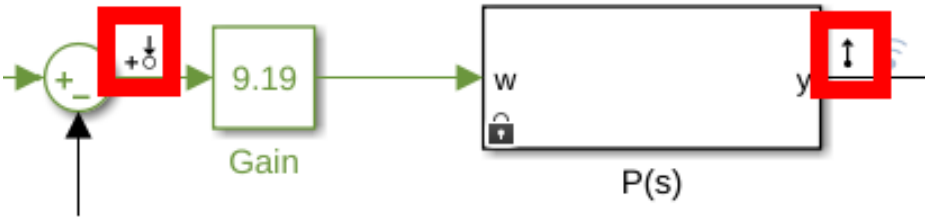
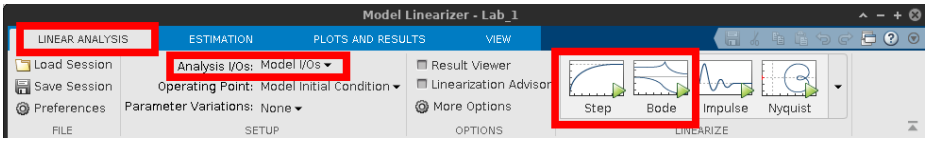


Figure A.4: The result after setting up an input signal and output measurement signal.

Acquiring Plots

Having indicated to the Model Linearizer app the input and output signal, you are now ready to acquire a step response. Open the Model Linearizer. Look at the top bar of the Model Linearizer and confirm that it looks as depicted below:



Try pressing the "`Step Response`" button! You should get a response like that of Figure A.5. Great! Now try clicking on a point of the curve. You will then get something we call a **data tip** or **cursor**, shown in Figure A.6. You can drag this cursor (the black dot) around to a specific point on the curve, allowing you to get accurate readings. Use this feature! Similarly, one can capture a Bode plot, as depicted in Figure A.7, by pressing the "`Bode`" button.

Now that we can acquire these plots, how do we *save* them? Notice that above every plot is a name. For example, Figure A.7 is in a tab named "Bode Plot 1." With this selected, you will see in the top bar a larger tab with the same name. This is depicted in Figure A.8. In that tab, you can then press "`Print to Figure`." Upon doing so, you have a Matlab Figure pop-up, which then gives you the option — under `File/Save As` — to export a `.png` or other image file.

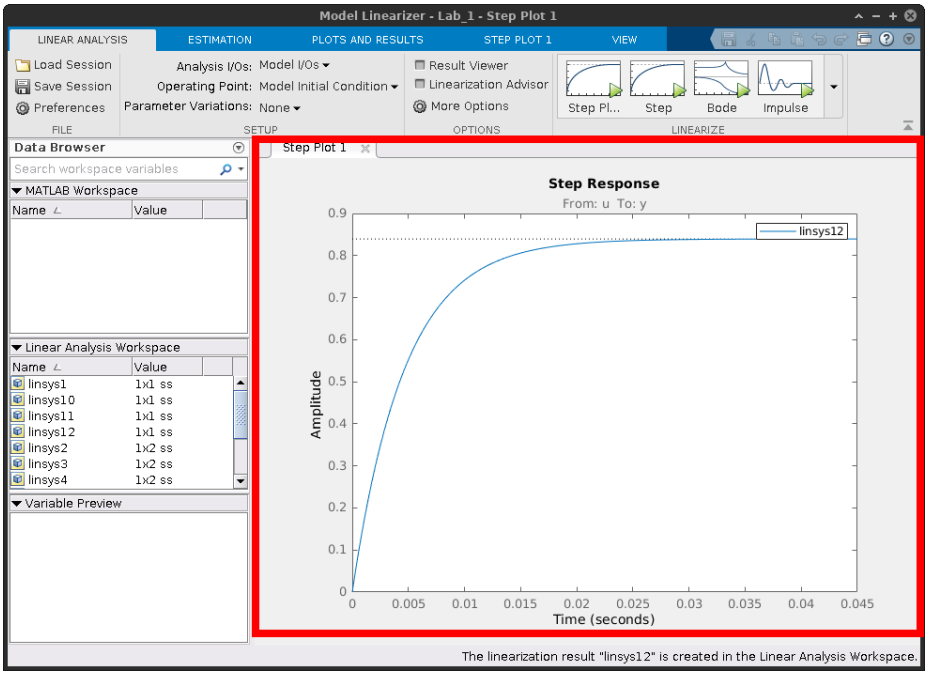


Figure A.5: Capturing a step response in the Model Linearizer app.

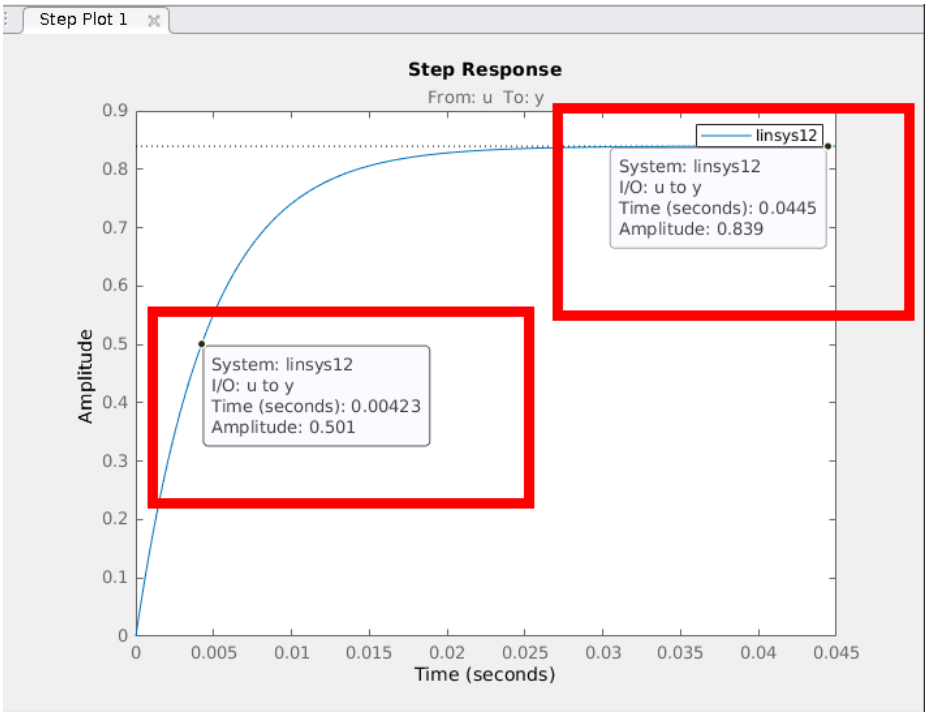


Figure A.6: I put two data tips, one near the settling value and another at just over 0.5 in amplitude.

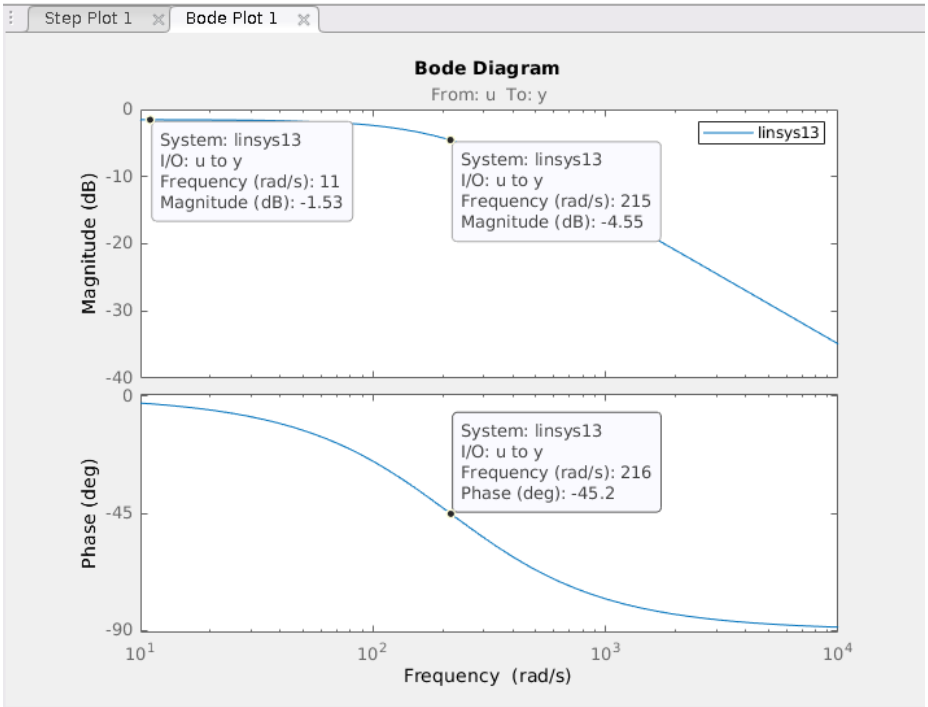


Figure A.7: A bode plot captured in the Model Linearize app. Do you know what special points I've marked on the Bode plot? Name them. Why are they important?

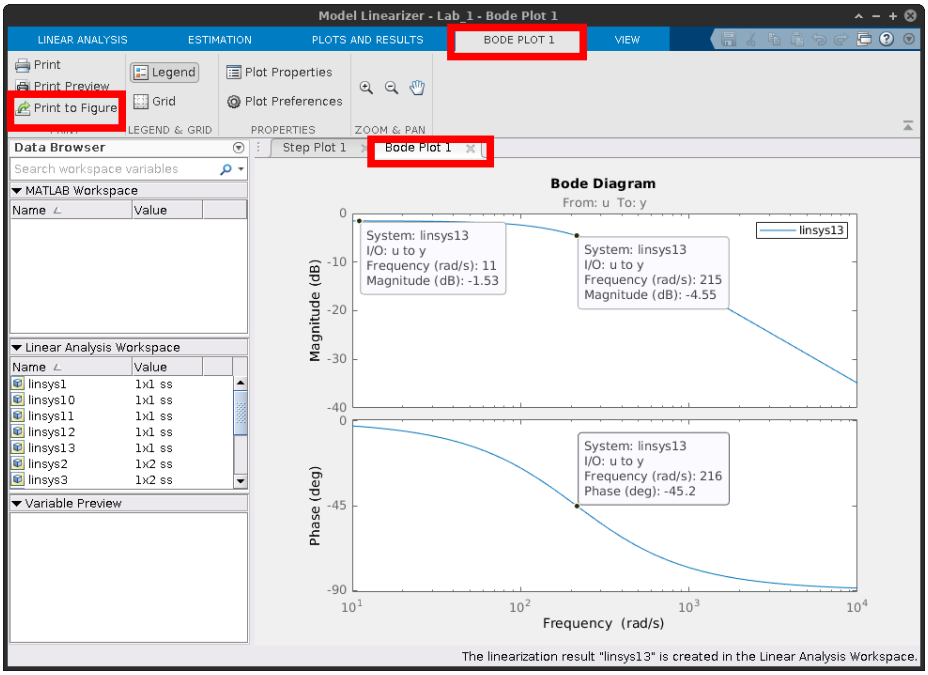


Figure A.8: This screenshot depicts how to turn a data acquisition into a Matlab Figure.