DEVELOPMENT OF AN UNDERGRADUATE LABORATORY COURSE IN CONTROL SYSTEMS

A Thesis
Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By
Elio Abiakel, B.S.E.C.E.

* * * * *

The Ohio State University
2003

Master’s Examination Committee:
Dr. Stephen Yurkovich, Adviser
Dr. Kevin M. Passino

Approved by

______________________________
Adviser
Department of Electrical Engineering
ABSTRACT

This thesis serves as a guideline for the development of a full one-term undergraduate laboratory course in control systems. The work presented here is mainly of experimental nature with emphasis on measurement, data acquisition and control design and analysis. The challenge behind this project was the adaptation of classical teaching and experiments in control systems to the state-of-the art implementation technologies used in the industry worldwide, while maintaining and improving the pedagogical aspects of the methodology previously followed in the Department of Electrical Engineering at The Ohio State University. First we introduce the hardware and software used in the laboratory, i.e. Quanser Consulting equipment, dSPACE computer cards and software packages, and Matlab and Simulink (Chapter 1). Then we present two laboratory experiments (Chapters 2 & 3) that serve as a tutorial introduction to these hardware and software and to digital signal processing. The next two experiments (Chapters 4 & 5) deal with system identification and basic control design using gain compensation. We then develop a series of experiments on DC servomotors to treat classical design methods such as root locus, lead, lag and PID control design (Chapters 6 through 8). We follow that with a set of experiments on more advanced apparatus such as flexible links and flexible
joints (Chapters 9 & 10). Chapters 11, 12, and 13 are dedicated to linear quadratic regulator (LQR) control design for the flexible joint, the flexible link and the two-degree-of-freedom helicopter; even though these chapters are not intended for use in an undergraduate laboratory course, their content can be used to develop demonstration experiments that can be presented at the beginning or the end of the term. Finally we conclude with general comments and recommendations for the future, along with an Appendix that contains an instructor’s guide.
Dedicated to my mother and to my sister
ACKNOWLEDGMENTS

I would like to thank my adviser, Professor Stephen Yurkovich, for his academic and financial support and for his patience in continuously supervising and correcting the Laboratories throughout the period of development.

I am also grateful to Daniel Clancy and Juliet Hurtig for allowing us to use some of their work in the previous version of the laboratory manual in order to develop our new lab.

I would also like to thank Professor Kevin M. Passino for being a member of my examination committee and for offering valuable suggestions to improve the Lab as well as providing ideas for some of the laboratories.

I thank Nicanor Quijano for helping me familiarize myself with dSPACE and the Quanser plants and for his contribution to some of the chapters of this document.

I thank Garo Zarikian for his useful help in creating and editing the list of figures and table of contents.

I also thank Bahar Jalali Farahani and Peng Yan for helping me teach EE557: Control Systems Technology Lab and for their quality feedback.

Finally I thank my mother, Thérèse, for, without her daily support I would have had no resolve.
VITA

January 1, 1978 ......................... Born - Beirut, Lebanon

2001 ................................. B.S. Electrical and Computer Engineering,

                        The Ohio State University

2001-present ......................... Graduate Teaching Associate,

                        The Ohio State University

FIELDS OF STUDY

Major Field: Electrical Engineering
# TABLE OF CONTENTS

| Abstract | ii |
| Dedication | iv |
| Acknowledgments | v |
| Vita | vi |
| List of tables | xii |
| List of figures | xiii |

Chapters:

1. Introduction ................................................................. 1

2. Software and Hardware ...................................................... 4

   2.1 Introduction ................................................................. 4

   2.2 A Simple Experiment Using Simulink and dSPACE ................. 6

       2.2.1 Frequency Domain System Identification of a “Black Box” .... 6

       2.2.2 Introduction to Simulink ........................................ 9

       2.2.3 Simulink Model .......................................................... 10

       2.2.4 The dSPACE Interface Board .................................... 14

       2.2.5 The Quanser Universal Power Module ......................... 15

       2.2.6 Connections .............................................................. 16

       2.2.7 dSPACE ControlDesk Environment ........................... 16

       2.2.8 Creating a New Experiment ....................................... 18

       2.2.9 Compiling ................................................................. 21

       2.2.10 Graphical User Interface ......................................... 22

   2.3 Running the Experiment and Gathering Data ....................... 27

       2.3.1 Real-Time Running and Data Capturing ....................... 27

       2.3.2 Capturing Data .......................................................... 28

       2.3.3 Frequency Sweep .................................................... 29
## Table of Contents

2.4 Post-Laboratory Exercises................................................................. 29

3. Sampled Data Systems................................................................................................ 31

3.1 Background ......................................................................................................... 31
  3.1.1 Overview .................................................................................................. 31
  3.1.2 Fourier Series ...................................................................................... 33
  3.1.3 Sampling of Continuous-Time Signals ............................................... 37
  3.1.4 The Z-plane .......................................................................................... 40
  3.1.5 Discrete Equivalents ........................................................................... 41
  3.1.6 Hardware Characteristics ................................................................... 43

3.2 Laboratory Preparation ..................................................................................... 44

3.3 Procedure .......................................................................................................... 47

3.4 Post-Laboratory Exercises ................................................................................ 49

4. The Quanser SRV-02 DC Servo................................................................................. 51

4.1 Modeling .......................................................................................................... 51

4.2 Transfer Function Estimation ........................................................................... 56

4.3 Laboratory Procedure ....................................................................................... 58
  4.3.1 The SRV-02ET DC Servomotor ......................................................... 58
  4.3.2 Connections ......................................................................................... 60
  4.3.3 Simulink Model .................................................................................. 61
  4.3.4 Compiling and Running ...................................................................... 62

4.4 Post-Laboratory Exercises ................................................................................ 64

5. Gain Compensation and Feedback for a DC Servo .................................................... 65

5.1 Background ...................................................................................................... 65
  5.1.1 Root Locus .......................................................................................... 65
  5.1.2 Root Locus Design .............................................................................. 67
  5.1.3 Bode Design ........................................................................................ 68
  5.1.4 Steady-State Error ............................................................................... 70

5.2 Laboratory Preparation ..................................................................................... 72

5.3 Laboratory Procedure ....................................................................................... 76
  5.3.1 Connections ......................................................................................... 76
  5.3.2 Simulink Diagram ............................................................................... 76
  5.3.3 ControlDesk and Results ..................................................................... 77

5.4 Post-Laboratory Exercises ................................................................................ 78

6. Lag Compensation for Speed Control of a DC Servo................................................. 79

6.1 Background ...................................................................................................... 79
  6.1.1 Overview ............................................................................................. 79
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>Helicopter Parameters</td>
<td>156</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>14</td>
</tr>
<tr>
<td>2.9</td>
<td>17</td>
</tr>
<tr>
<td>2.10</td>
<td>19</td>
</tr>
<tr>
<td>2.11</td>
<td>20</td>
</tr>
<tr>
<td>2.12</td>
<td>20</td>
</tr>
<tr>
<td>2.13</td>
<td>21</td>
</tr>
<tr>
<td>2.14</td>
<td>23</td>
</tr>
<tr>
<td>2.15</td>
<td>24</td>
</tr>
<tr>
<td>2.16</td>
<td>25</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>2.17</td>
<td>Plotter</td>
</tr>
<tr>
<td>2.18</td>
<td>Running Modes</td>
</tr>
<tr>
<td>2.19</td>
<td>Capture Settings</td>
</tr>
<tr>
<td>3.1</td>
<td>Digital Signal Processing System</td>
</tr>
<tr>
<td>3.2</td>
<td>A/D Converter</td>
</tr>
<tr>
<td>3.3</td>
<td>D/A Converter</td>
</tr>
<tr>
<td>3.4</td>
<td>Quantizer Characteristic for an A/D Converter</td>
</tr>
<tr>
<td>3.5</td>
<td>Zero-Order Hold</td>
</tr>
<tr>
<td>3.6</td>
<td>Triangular Waveform</td>
</tr>
<tr>
<td>3.7</td>
<td>Ideal Sampling and Reconstruction System</td>
</tr>
<tr>
<td>3.8</td>
<td>Closed-Loop System</td>
</tr>
<tr>
<td>3.9</td>
<td>Simulink Diagram</td>
</tr>
<tr>
<td>4.1</td>
<td>Block Diagram of Position Servo</td>
</tr>
<tr>
<td>4.2</td>
<td>Open-Loop Unit Step Response</td>
</tr>
<tr>
<td>4.3</td>
<td>SRV-02ET DC Servomotor</td>
</tr>
<tr>
<td>4.4</td>
<td>Low Gear Ratio</td>
</tr>
<tr>
<td>4.5</td>
<td>High Gear Ratio</td>
</tr>
<tr>
<td>4.6</td>
<td>Simulink Diagram</td>
</tr>
<tr>
<td>5.1</td>
<td>Closed-Loop Control System</td>
</tr>
<tr>
<td>5.2</td>
<td>Closed-Loop System</td>
</tr>
<tr>
<td>5.3</td>
<td>RItool Import</td>
</tr>
<tr>
<td>5.4</td>
<td>Design Constraint</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

In the 1990’s, the Control Systems Laboratory undergraduate course in the Department of Electrical Engineering at The Ohio State University was taught using mostly analog equipment like the ComDyna and data acquisition was basically limited to GPIB connections between oscilloscopes and computers; the lab had no “real” physical plants and digital control was still at an entry level. In 2001, due to the efforts of Professors Stephen Yurkovich and Kevin M. Passino, the lab was equipped with the latest technologies for control design and implementation. The innovations include:

- New Dell computers
- New physical plants: motors, links, joints, tanks, ball and beam, pendulums, cube… purchased from Quanser (http://www.quanser.com)
- DS1104 dSPACE controller cards for real-time control and dSPACE Solutions for Control software package for design and analysis (http://www.dspaceinc.com)
- Matlab Real-Time Workshop

The educational benefit brought by these innovations is the fact that students and researchers can now use Simulink to design a controller and interface it with the outside
plant; then, using a seamless combination of Matlab Real-Time Workshop (RTW) and dSPACE Real-Time Interface (RTI), the Simulink model is compiled into a C code and downloaded on the dSPACE 1104 card for real-time running. This process is extremely time-efficient since it allows the user to use a high-level block diagram simulation program like Simulink instead of spending hours on coding and debugging in a lower level language. Visualization and analysis of results can be conducted in dSPACE ControlDesk, a graphical user interface software that allows graphing, data capturing and real-time parameter tuning. Another advantage of ControlDesk is the compatibility of its captured data with Matlab.

Chapter 2 discusses in depth the software and hardware used in the laboratory. Chapters 2 through 10 are written in a lab manual format, i.e. they contain specific instructions on how to operate the equipment in order to perform specific experiments. Laboratory Preparation sections are exercises that each student must complete before performing that week’s actual Laboratory Procedure. The Laboratory Preparation contains theoretical questions that might require mathematical calculations and proofs, as well as computer simulation exercises of what will be seen on the lab bench equipment. Written reports based on this manual are required after each Laboratory Procedure. The sections labeled “Post-Laboratory Exercises” require the students to discuss and analyze the results of the experiment as well as answer some qualitative or quantitative questions. A characteristic immediately evident in each of the Laboratories is that the second section provides a theoretical background drawn from [1], [2], and [3] to explain the concepts behind the Laboratory. Another characteristic is that the section labeled “Laboratory
Procedure” contains a detailed step-by-step description of what the students should do in order to build the Simulink model and connect the hardware. We chose this methodology so that the students will have more time to think about control theory and so that the instructor can divide his/her time more efficiently among all the groups. However, this method should not be misused and the students are urged to completely understand each step and the underlying software and hardware settings.

In this document, the words “Lab”, “lab”, “course”, “Laboratory”, “laboratory” are related to the course EE557: Control Systems Technology Lab offered by the Department of Electrical Engineering at The Ohio State University. Nevertheless, the information presented herein may be used in any other setting without loss of generality.
This chapter serves as an introductory guide to data acquisition, sampled data systems, and control systems development and implementation using dSPACE® software. It also provides information about interfacing Quanser® plants to the dSPACE data acquisition cards. This tutorial is designed as a quick reference for a university laboratory course; more detailed documentation can be found in dSPACE manuals and Quanser documentation.

2.1 Introduction

This introduction is intended to be generic with respect to control system technology, focusing instead on introducing the hardware platform and software to be used in future Laboratories. For example, the only terms from control systems structure we will use here are: the plant (object to be controlled), the actuator (physical hardware mechanism that produces a control action on the plant), and sensor (physical device that produces measurements from the plant and actuators). In Appendix A we remind the student of some basic quantities associated with signals in a typical control system.
Throughout we will assume some elementary familiarity with Matlab and Simulink, and will freely refer to operations within each; however, we will essentially be giving an introduction to building control systems within the Simulink environment. It is anticipated that the user will frequently return to this introductory Laboratory until sufficient familiarity is achieved.

In order to control a plant, we need to measure certain variables of that plant and feed the measured signals to a digital computer. This is done via analog sensors (e.g. strain gage, tachometer, potentiometer, and so on) or digital encoders, and is commonly called data acquisition. Another important signal in a control system is the control input to the plant. This is a signal computed by the digital computer (controller) and fed to the actuator (e.g. servo, motor) in order to achieve desired tracking.

The hardware platform that takes sensor inputs, and sends out control signals, is the data acquisition card. In this Laboratory, and those to follow, we will use the dSPACE DS1104 Controller Board installed in the desktop computer for data acquisition and an interface board provided by Quanser to interface the DS1104 card with the plants. Since some measured signals are analog signals, the DS1104 card has analog-to-digital converters in order to make the signals compatible with the digital computer. It also has digital-to-analog converters so that the computed control signals can be fed to the analog plants.

The software platform for this setup is a combination of Simulink in which users can
develop controllers and dSPACE ControlDesk in which they can create graphical user interfaces. Figure 2.1 shows a general block diagram of all the experimental setups that we will be implementing:

![Control System Block Diagram](image)

Figure 2.1: Control System Block Diagram

2.2 A Simple Experiment Using Simulink and dSPACE

2.2.1 Frequency Domain System Identification of a “Black Box”

We will use Simulink and dSPACE in order to identify an unknown transfer function. Since this is an introductory experiment, we will simulate the plant using a transfer function contained in a Simulink block created and masked by the instructor. That is, you will not know the parameters of the transfer function, but must “identify” them through signal and system analysis; this, of course, is often a necessary task in many applications. In future laboratories, we will interface real plant hardware (experiments) to the dSPACE platform.
Let the general second order transfer function estimate with output, \( Y(s) \), and input, \( R(s) \), be represented as:

\[
\frac{Y(s)}{R(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}
\]

Figure 2.2: Frequency Magnitude Response of a Second-Order System

Figure 2.2 provides a typical Bode magnitude response for a control system with the important quantities appropriately marked. In this figure (see Appendix A) several quantities (most measurable) are displayed and defined in the following procedure, which
presents a general technique for obtaining a second-order transfer function estimate of a control system from a Bode magnitude response:

**Estimate $\zeta$:** Determine the magnitude of resonance, $M_r$, which is the resonance peak, $M_m$, minus the static gain magnitude, $M_0$, and then use the magnitude of resonance relationship given in Appendix A.

**Estimate $\omega_n$:** Determine one of the following quantities

(i) Undamped natural frequency, $\omega_n$, can be estimated directly by finding the point of intersection between the straight line asymptote for the static gain magnitude, $M_0$, and the high-frequency roll-off asymptote, $S_{\text{roll-off}}$, which must be $-40$ dB/decade for a second-order transfer function.

(ii) Resonance frequency, $\omega_r$, which is the frequency at which the resonance peak, $M_m$, occurs and then use the resonance frequency relationship from Appendix A.

**DC gain:** The DC gain is calculated using $K = 10^{\frac{M_0}{20}}$. This comes from the relationship $M_0 = 20 \log_{10} K$, which defines the dB static (DC) gain magnitude.

**Remark**

The high-frequency roll-off asymptote, $S_{\text{roll-off}}$, directly determines the order of the transfer function estimate for an all-pole system (no finite zeros). If $S_{\text{roll-off}} = -20$ dB/decade, then we can deduce that the system is first order and the single pole can be estimated directly by finding the point of intersection between the straight line asymptote for the static gain magnitude, $M_0$, and the high-frequency roll-off asymptote, $S_{\text{roll-off}}$. 
2.2.2 Introduction to Simulink

Open Matlab and type “simulink” at the command prompt. The Simulink Library Browser will appear as shown in Figure 2.3:

Figure 2.3: Simulink Library Browser
Simulink is a useful tool that allows you to simulate a system without the need of coding in lower level languages. It uses built-in blocks that can be interconnected by the user and a variety of built-in differential equations solvers for the model.

2.2.3 Simulink Model

Click on the “Open a Model” in the Simulink Library Browser toolbar. Select the file named lab1.mdl in the folder C:\EE557\LAB1. The extension “.mdl” denotes Simulink models. In this model you will see a block named “Black Box”. The objective of this experiment is to use frequency-domain system identification techniques in order to estimate the transfer function of the system contained in the black box. Any block that you need for this course will be available in the Simulink Library Browser. Simply find the block you need then drag and drop it in the Simulink model window. If you cannot find a block, simply type the name in the “Find” field and click on “Find”. Most of the blocks needed for this course will be under the “Simulink” and “dSPACE RTI1104” icons. The other icons represent more advanced toolboxes. Build the model for this experiment by following the instructions:

- Find the block named “Sine Wave” under Simulink Sources and drag and drop it in the model. Double-click on the block and set the frequency to 0.1 rad/sec. This frequency will be changed later in order to perform a frequency sweep from 0.1 to 100 rad/sec.
Now find the block named DAC in the “DS1104 MASTER PPC” subdivision under the “dSPACE RTI1104” icon and drag and drop it in the model. Double-click in the block and observe the following window:

![Figure 2.4: DAC Settings](image)

Select “Channel number 1” then click on the “Termination tab” and set the termination value to zero as shown in the figure below so that the output of the DAC is 0 V when the experiment stops running:

![Figure 2.5: DAC Settings](image)
- Find the block named “Gain” under the Simulink Math group and insert it between the sine wave and the DAC block. Double-click on the “Gain” block and change the gain value to 1/10. Connect the sine wave output to the gain input, then connect the gain output to the DAC block. This will output the sine wave signal on the first Analog Output channel of the data acquisition board. The gain of 0.1 is necessary because the dSPACE card always amplifies the output signal by 10; therefore, dividing the signal by 10 in software compensates for this scaling.

- Now find the block named ADC in the “DS1104 MASTER PPC” subdivision under the “dSPACE RTI1104” icon and drag and drop it in the model. Double-click in the block and select channel 5. Connect the output of the ADC to the input of the black box through a gain of 10 as done previously. The gain of 10 is necessary because the dSPACE card always divides the analog input signal by 10; therefore, multiplying the signal by 10 in software compensates for this scaling factor.

- Find a block called a Terminator (Simulink Sinks) and connect it to the output of the black box. Double-click on the connecting wire and name it “Response”. This will be clarified when we discuss the dSPACE software. Your simulink model should look similar to the model in figure 2.6.
Click on “Simulation” in the toolbar of the model window, then click on “Simulation Parameters”. Change the Stop time to 60. Then, under the “Solver” tab, change the “Solver options” to a Fixed step of 1 ms and an ODE5 solver as follows:
The ODE solver (one choice among several) represents a numerical integration routine Simulink will use to simulate your control system.

- Under the “Advanced” tab, change block reduction to Off as follows:

![Simulation Parameters](image)

Figure 2.8: Simulation Parameters

2.2.4 The dSPACE Interface Board

The dSPACE DS1104 Interface Board is used to interface signals to the dSPACE DS1104 card located in the computer tower. The following elements of the interface board are of importance to this and other laboratories to follow:
Eight analog inputs numbered from 0 to 7. When you select Analog Input Channel 1 in Simulink or dSPACE software, this corresponds to Channel 0 on the interface board. This is a source of confusion, therefore, always check your connections before you run an experiment.

- Eight analog outputs numbered from 0 to 7.
- Two 5-pin Din-stereo connectors used for encoders.
- Two SCSI 3 Female connectors, labeled P1A and P1B, to interface the DS1104 Interface Board to the dSPACE data acquisition card.

2.2.5 The Quanser Universal Power Module

This module (UPM-2405) has the following connectors:

- From Analog Sensors: these are the inputs from analog signals (sensors, etc.). They use 6-pin Mini to Din / 6-pin Mini Din cables to connect the UPM-2405 to the analog source. The purpose of these connection points is to internally connect the analog signals to the analog input connector, “To A/D”.

- To A/D: This connector uses a 5-Pin Din-stereo / 4 × RCA cable to connect the analog inputs to the dSPACE card. We will refer to this cable as the “To A/D” cable.

- From D/A: This connector uses a 5-Pin Din-mono / RCA cable to connect the analog output from the dSPACE board to the plant. We will refer to this cable as the “D/A” cable.
• To Load: This connector uses a 6-pin Din / 4-pin Din black cable to connect the power amplifier to the actuator of the plant. We will refer to this cable as the “To Load” cable.

2.2.6 Connections

Since we are not using a physical plant in this experiment we will only need to connect the analog output numbered 0 to the analog input numbered 4 on the DS1104 interface board using an RCA/RCA cable. This is called “loop-back”.

2.2.7 dSPACE ControlDesk Environment

The purpose of this section is to familiarize the user with the various functions of dSPACE ControlDesk, as well as assist in implementing a first control system and saving data results using dSPACE. First, launch ControlDesk from the Windows desktop by double-clicking on the corresponding icon. The window in Figure 2.9 will appear.

ControlDesk is a user interface that allows the user to run simulations on different platforms such as the real-time DS1104 Controller Board or simply non-real-time applications like Simulink. For example, you can build a Simulink model that has no blocks related to dSPACE and then run the simulation in Matlab and interface it to the ControlDesk in order to explore the results. Although we will not be doing this type of simulation for this course, we will use Simulink to build models that include dSPACE elements such as D/A’s, A/D’s, encoders, etc., as we did in section 2.2.3, then we will
compile the models and download them on the DS1104 card to run them in real-time.

Finally, we will use the data capturing capabilities of ControlDesk to save the results and integrate them into Matlab for plotting purposes.

The ControlDesk window contains three main regions when in default settings: The Navigator (upper left corner), the general work area (upper right area) and the Tool window (bottom area).
The Navigator has three tabs: Experiment, Instrumentation and Platform. The platform tab shows the platforms to which ControlDesk can interface. The instrumentation tab shows open “Layouts” (graphical user interfaces created by the user) and the graphical instruments associated with those layouts. The Experiment tab shows the current open experiment and the associated files.

The Tool window has several tabs, the most important being the file selector tab that allows you to browse files and drag and drop applications to the platform tab. The extensions shown in the file selector are *.mdl (Simulink model files), *.ppc (compiled object files), *.sdf (system description files) and *.trc (variable description files). The files that can be dragged and dropped are *.ppc or *.sdf that have the same effect.

The general work area is used to display layouts.

2.2.8 Creating a New Experiment

To create a new experiment, click on the file menu in ControlDesk and choose “New Experiment”. You will see the window in Figure 2.10. Type a name for the experiment and choose C:\EE557\LAB1 as the working root directory. Other fields in the window above are optional and for documentation purposes only. The experiment will be saved with a *.cdx extension. We will associate all the files that we work with to this experiment so that when we save the experiment and close it, all the layouts (graphical user interfaces) and compiled applications from Simulink will be linked to the experiment and loaded with it if we choose to open it later.
Remark

Since the dSPACE card uses EPROM technology to store real-time applications, a previous program may still be downloaded on the card when you launch dSPACE ControlDesk. In order to clear the EPROM, right-click on the “ds1104” icon in the Navigator window (see Figure 2.9) and select “Clear Flash”. The window in Figure 2.11 will appear.
In this case, the application “helicopter.ppc” is loaded on the flash. Click on “Clear Application” and you should get the window in Figure 2.12, indicating that the memory is clear.

Figure 2.12: Cleared Flash
2.2.9 Compiling

You are now ready to compile your Simulink model and run the experiment:

- In the Simulink model window click on “Tools”, then click on “Real-Time Workshop” and select “build model” (you could also type ctrl+B as a shortcut to these steps). You might get the screen shown in Figure 2.13.

![Figure 2.13: RTI Task Configuration](image)

Select “Stop simulation” at the bottom of the screen and click continue. This will insure that the control algorithm always runs in real-time. If the sampling time is too small, i.e. smaller than the sum of the system time constant and the
computational time, there is an overrun and the simulation is stopped to prevent non-real-time operation. In that case, you will have to adjust the sampling time and re-compile.

- Wait while Matlab compiles the model into a C-code executable application that will be downloaded on the dSPACE card. Go to ControlDesk and observe the Tool window tabs; you should have a tab for the application that you downloaded. If you see other applications (previously downloaded applications) click on the corresponding tab and then right-click on the corresponding icon in the Tool window and select “close”.

2.2.10 Graphical User Interface

In order to visualize the real-time experiment and make it interactive, it is necessary to build a layout and associate it with the experiment. In order to do so, click on “File” in ControlDesk, then click on “New” and select “Layout”. The general work area of ControlDesk will appear with a new window as shown in Figure 2.14.
Figure 2.14: ControlDesk Layout

Note the Virtual Instruments panel on the right. It allows you to select from various instruments and then draw the instrument box in the Layout area. For this experiment, we are interested in the Data Acquisition panel. Click on the Data Acquisition button and observe the panel in Figure 2.15.
Click on Plotter and draw a plotter in the Layout area. The plotter will have a red border indicating that no variables have yet been assigned to the graph (Figure 2.16).

To assign variables to be plotted on the graph:

- Double-click on “Model Root” in the left half of the Tool window.
- Double-click on “Sine Wave” under “Model Root”.
- Select “Out1” in the right half of the Tool window and drag and drop it on the Y-axis of the graph.
- Do the same with the variable “Response”.
- Right-click on the plotter and select “properties”. Make the X-axis and Y-axis floating and add a grid.
Your plotter should now look like Figure 2.17.

Click on File and select “Add all opened files”. This will insure that all open layouts are associated with the experiment. Click on File and select “Save Experiment”
The next step is to get familiar with the different modes under which an experiment runs in ControlDesk. We can operate under three modes:

- **Edit Mode** (the default mode under which we create and edit layouts and capture data)
- **Test Mode** (used for testing the current layouts)
- **Animation Mode** (used to visualize the real-time progression of the experiment)

When we first compile the experiment, it automatically goes into Animation mode. When the experiment stops because the time preset in Simulink has expired, ControlDesk goes into Edit mode. However, if you need to stop the experiment before the simulation time expires, you need to switch to Edit mode first. Switching between modes is done by clicking on one of the icons in Figure 2.18.
2.3 Running the Experiment and Gathering Data

2.3.1 Real-Time Running and Data Capturing

Click on the “Play” button and immediately switch to animation mode and watch the signals vary in the plotter. If no signal is shown on the plotter, or if a signal is shown for a very short period of time, this probably implies that your capture settings are incorrect. Right-click on the plotter after the simulation stops and select “Edit Capture Settings”. Set the length to a number higher than your simulation time (Figure 12.19)

Re-run the experiment and you should be able to observe the signal for the duration of the experiment.
2.3.2 Capturing Data

In order to save the data and plot it in a more convenient Matlab plot, right click on the plotter after the simulation stops and click on “Edit Capture Settings”. Click on “Save” (Figure 2.19) and select the name of the *.mat file in which the data will be saved.

As one structure, dSPACE saves the plotted variables and the time vector in a *.mat file. Your next task is to convert this structure into an array of separate vectors so that you can plot each vector with respect to the time vector. Below is a sample Matlab script to achieve this:
load filename
v = filename

t = getfield(v.X(1), 'Data'); % time vector, X
y1 = getfield(v.Y(1), 'Data'); % variable 1, Y
y2 = getfield(v.Y(2), 'Data'); % variable 2, Y
plot(t,y1,t,y2)

2.3.3 Frequency Sweep

An important step in system identification is to analyze the “black box” for frequency content information. To do that, inputs of various frequencies (a “sweep”) are used, and the resulting outputs are analyzed. The procedure:

- Add a numeric input to your Layout from the virtual instrument panel and associate the variable “Frequency” of the Sine Wave to it.
- Run the experiment and change the frequency using the following values:
  [ 0.1, 2, 4, 6, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100] rad/sec. For each frequency, save the plot in a *.mat file and record the magnitudes of input and output in tabular format.

2.4 Post-Laboratory Exercises

Your report for this Laboratory should be compiled according the general guidelines outlined by your instructor. For this particular Laboratory, you should:

(i) Submit three plots of captured data as explained in Section 2.3.2
(ii) Compute the gain ratios of output over input in dB and plot the corresponding points on a Bode plot in Matlab. Connect the dots smoothly and obtain a Bode plot estimate.

(iii) Use the technique discussed in Section 2.2.1 to find an estimate of the transfer function of the Black Box.

(iv) Produce the Bode plot of the estimated transfer function and mark (on your plot) the measured points.
The objective of this chapter is to offer a brief introduction to digital signal processing. The effects of sampling an analog signal using an analog to digital (A/D) converter and then reconstructing this analog signal using a digital to analog (D/A) converter will be analyzed. As a result, the effects of aliasing and quantization errors will be demonstrated. This Laboratory will also review the basics of Fourier series analysis, examine the sampling of finite and infinite bandwidth signals, and will briefly address the concept of discrete-time equivalent approximation of a continuous-time filter.

3.1 Background

3.1.1 Overview

The incredible advances in the areas of microprocessor and computer technology have made analog compensation (control using analog computers, i.e. electric circuits, amplifiers, etc.) almost obsolete. Most of today's control systems are implemented using digital controllers. This type of control system is called a sampled data system, in which the controller is implemented on a digital computer, and is included in the closed loop
system. In order to develop and implement a sampled data control system, basic digital signal processing techniques must be thoroughly understood.

![Digital Signal Processing System](Figure 3.1: Digital Signal Processing System)

![A/D Converter](Figure 3.2: A/D Converter)

![D/A Converter](Figure 3.3: D/A Converter)
A typical digital signal processing system is shown in Figure 3.1 above. Figure 3.2 presents a block diagram representation for an A/D converter and Figure 3.3 depicts a similar block diagram representation for a D/A converter. In these figures, “ZOH” refers to zero-order hold operation, which will be discussed in more detail later.

In this configuration, an analog signal \( x(t) \) is sampled by the A/D converter, processed by the digital computer, and then transformed back into an analog signal \( \tilde{x}(t) \) by the D/A converter. This entire process is synchronized by a clock circuit whose speed is dictated by the sampling period \( T \). The objective of this digital signal processing system is to ensure that the important characteristics about the input signal \( x(t) \) are contained in the output signal \( \tilde{x}(t) \).

### 3.1.2 Fourier Series

The widespread applicability of the Fourier series representation for a signal stems from its ability to accurately represent most practical excitation signals. The Fourier series decomposes any signal into a sum of sinusoids or complex exponentials. For representing periodic signals, Fourier series are used, and for aperiodic signals, the Fourier transform is employed. The main concentration of this section will be on Fourier series and the truncated Fourier series. A Fourier series expansion for a periodic signal \( f(t) \) is given by the following:

\[
f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} b_k \sin(k\omega_0 t),
\]

\( a_0, a_k, b_k \)
where \( \frac{a_0}{2} \) is the average value of the signal \( f(t) \), \( a_k \) the harmonic component of \( f(t) \) due to the \( k^{th} \) cosine excitation \( \cos(k \omega_0 t) \), \( b_k \) is the harmonic component of \( f(t) \) due to the \( k^{th} \) sine excitation \( \sin(k \omega_0 t) \), and \( \omega_0 = \frac{2\pi}{T} \) is the fundamental frequency and \( T \) is the period.

The sufficient conditions such that the Fourier series representation is valid are as follows:

1. \( \int_T |f(t)|dt < \infty \)
2. For any finite interval, \( f(t) \) must have a finite number of minima and maxima and a finite number of finite discontinuities.

Neglecting any additional technical details as to when the Fourier series representation is valid, the Fourier coefficients are determined by the following equations:

\[
\begin{align*}
a_0 &= \frac{2}{T} \int_T f(t)dt \\
a_k &= \frac{2}{T} \int_T f(t)\cos(k \omega_0 t)dt \\
b_k &= \frac{2}{T} \int_T f(t)\sin(k \omega_0 t)dt,
\end{align*}
\]

where it is understood that integration is over the period \( T \).

The above computations for the Fourier coefficients can be simplified under the following conditions:

1. If \( f(t) \) is an even function, i.e., \( f(t) = f(-t) \), then the following simplifications hold:
2. If \( f(t) \) is an odd function, i.e., \( f(t) = -f(-t) \), then the following simplifications hold:

\[
a_k = \frac{4}{T} \int_{0}^{T} f(t) \cos(k\omega_0 t) \, dt
\]
\[
b_k = 0.
\]

In practice, it is impossible to obtain a representation for a signal based on an infinite number of sinusoidal excitations. Therefore, in a laboratory setting, the Fourier series representation is actually the truncated Fourier series representation given as follows:

\[
f_N(t) = \frac{a_0}{2} + \sum_{k=1}^{N} a_k \cos(k\omega_0 t) + \sum_{k=1}^{N} b_k \sin(k\omega_0 t),
\]

where \( N \) is some suitable integer. An alternate representation of the truncated Fourier series can be derived based on the trigonometric identity:

\[
x \cos(A) + y \sin(A) = r \cos(A + \phi),
\]

with \( r = \sqrt{x^2 + y^2} \) and \( \phi = \tan^{-1}\left( -\frac{y}{x} \right) \),

which yields the following:

\[
f_N(t) = A_0 + \sum_{k=1}^{N} A_k \cos(k\omega_0 t + \phi_k),
\]
where $A_0 = \frac{a_0}{2}, A_k = \sqrt{a_k^2 + b_k^2}$, and $\phi_k = \tan^{-1}\left(-\frac{b_k}{a_k}\right)$. In this context, $A_1$ is called the first (fundamental) harmonic, $A_2$ is called the second harmonic, etc.

Given a truncated Fourier series representation $f_N(t)$ for a signal $f(t)$, it is now important to define a figure of merit so that the quality of this approximation can be quantified mathematically.

**Average Power**: Given the periodic signal $f(t)$, the average power, $P_f$, contained in this signal is determined by:

$$P_f = \frac{1}{T} \int_0^T |f(t)|^2 \, dt.$$  

The average power for the truncated Fourier series is:

$$P_{f_N} = \frac{1}{T} \int_0^T |f_N(t)|^2 \, dt = A_0^2 + \sum_{k=1}^{N} \frac{A_k^2}{2}.$$  

The accuracy of the approximation can now be quantified using the following definition for the approximation error $\eta_N$ of the truncated Fourier series representation.

**Approximation Error**: Given the average power $P_f$ of a periodic signal $f(t)$, and the average power $P_{f_N}$ of the truncated Fourier series representation $f_N(t)$ for the signal, the normalized approximation error, $\eta_N$, is calculated according to:

$$\eta_N = \frac{P_f - P_{f_N}}{P_f}.$$  

Note that $P_f \geq P_{f_N}$. 

36
3.1.3 Sampling of Continuous-Time Signals

This section briefly discusses some of the issues involved in the sampling of continuous-time signals and highlights the primary sources of reconstruction errors, namely, aliasing and quantization. The Nyquist Sampling Theorem is valid for band-limited signals which contain no spectral content above some frequency \( f_n \) in Hz. In words, the Nyquist Sampling Theorem states that if the sampling frequency \( f_s \) is at least twice the bandwidth, \( f_n \), of the original signal, then there is no loss of information between the original signal and the sampled signal.

\[ \text{Nyquist Sampling Theorem:} \] Given a band-limited signal \( x(t) \), represented as follows:

\[
X(f) = 0, \quad |f| > f_n,
\]

with \( X(f) \) signifying the frequency response of the signal \( x(t) \), and \( f \) representing the frequency in Hz. The signal \( x(t) \) can be uniquely determined by its samples \( x[n] = x(nT) \)

if the sampling frequency, \( f_s = \frac{1}{T} \), satisfies the following relationship:

\[
f_s \geq 2f_n.
\]

Signals that exist for a finite period of time, i.e., time-limited signals, cannot be band-limited in frequency. Similarly, any band-limited signal in frequency must exist for an infinite period of time. In addition, many signals are neither time-limited nor band-limited. Whenever an infinite bandwidth signal is sampled or the Nyquist Sampling Theorem is violated for a band-limited signal, some of the high-frequency components within the sampled signal will overlap the frequency components of the original signal,
producing aliasing error in the reconstructed signal. Most practical signals have a frequency spectrum which decreases as the frequency increases. Thus, for most signals, the aliasing error can be diminished by selecting a sampling frequency, $f_s$, which is large enough such that the amplitudes of the overlapping high-frequency components are negligible. Another approach is to prefilter the original signal before sampling by using an anti-aliasing low pass filter which has the effect of band-limiting the signal.

The A/D converter, depicted in Figure 3.2, is a device that converts an input voltage (or current) into a binary code which represents the quantized amplitude value closest to the actual amplitude of the input signal. In addition, since the A/D conversion process is not instantaneous, the A/D system includes a sample and hold circuit to prevent the input value from changing during processing. The hold circuit is usually a zero-order hold (ZOH). The A/D converter is composed of a nonlinear device called a quantizer and a coder. The quantizer transforms the input sample, $x[n]$, into a finite set of prescribed amplitude values, $\hat{x}[n]$, according to:

$$\hat{x}[n] = Q(x[n]),$$

where $\hat{x}[n]$ is called the quantized sample. The quantizer can be composed of either uniformly or non-uniformly spaced quantization levels. For signal processing, the quantization steps are usually uniformly spaced, which is the case for the A/D converters utilized herein. Figure 3.4 depicts a typical uniform (linear) quantizer characteristic with sample rounding to the nearest quantization level. Notice that the quantizer pictured in Figure 3.4 is composed of an even number of quantization levels. Also observe that the number of positive and negative quantization levels is not equal, because there exists a
zero-amplitude quantization level. The number of quantization levels is, in general, even, resulting in a coder which performs binary coding. Many different binary coding schemes exist, but these schemes will not be discussed any further. The full-scale level of the A/D converter, $V_{\text{max}}$, shown in Figure 3.4, determines the resolution (step-size) between quantization levels according to the following equation:

$$\Delta_r = \frac{2V_{\text{max}}}{2^B},$$

where $B$ is the number of bits used to represent the quantized sample, and $2^B$ is the number of quantization levels. For the A/D converters used in this laboratory, $V_{\text{max}} = 10V$.

![Figure 3.4: Quantizer Characteristic for an A/D Converter](image)
3.1.4 The Z-plane

For the analysis of continuous-time systems, we use the Laplace transform which leads directly to the transfer function of a linear system, given the differential equations describing that system. For the analysis of discrete-time systems, the z-transform is defined by:

\[ F(z) = \sum_{k=0}^{\infty} f(k) z^{-k}, \]

where \( f(k) \) is the sampled version of the function \( f(t) \). Analogous to the continuous-time case, the delay property of the z-transform leads directly to the transfer function of a discrete system, given the difference equations describing that system. For example, a second order difference equation in the general form

\[ y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2) \]

can be transformed to

\[ Y(z) = (-a_1 z^{-1} - a_2 z^{-2})Y(z) + \left(b_0 + b_1 z^{-1} + b_2 z^{-2}\right)U(z) \]

which leads directly to the transfer function

\[ \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \]

Certain behavior results from different S-plane pole locations (continuous-time). That is, complex poles near the \( j\omega \)-axis result in oscillatory behavior, unstable behavior results from right-half-plane poles, and so on. A similar analysis holds for the Z-plane and discrete-time systems. In general, a pole (or zero) at \( s = -\sigma \) in the S-plane corresponds to a pole at \( z = e^{-\sigma T} \) in the Z-plane. In fact, this is true in general, and the S-plane is related
to the Z-plane according to $z = e^{sT}$ where $T$ is the sampling period. In general, all important features of the S-plane map into the Z-plane via this relationship, including the “stability boundary” (jω-axis in the S-plane), which becomes the unit circle in the Z-plane.

3.1.5 Discrete Equivalents

Given a continuous-time filter, $H(s)$, the problem for implementation on a digital computer is to find the best discrete equivalent $H(z)$. A digital implementation (discrete equivalent) requires that $u(t)$ be sampled at some sampling rate and that the computer output sample stream be “smoothed” somehow to provide a continuous $y(t)$. This smoothing operation is usually implemented with the ZOH shown in Figure 3.5.

![Figure 3.5: Zero-Order Hold](image)
In general there is no exact solution to the problem of matching an approximation $H(z)$ to $H(s)$, since $H(z)$ is formed only from samples of a continuous signal, as opposed to a complete time history. Different techniques for approximation effectively make different assumptions about what happens between sample points. We will be investigating two different discrete equivalent approximations. For the sake of argument, consider the continuous-time filter transfer function

$$H(s) = \frac{a}{s + a}.$$ 

Two methods considered for finding a discrete-time equivalent of $H(s)$ are concerned with how integration (a continuous-time operation) is approximated. This approximation can be carried out in several ways; the two considered in this laboratory are approximating the area under a curve with a “forward looking” rectangle or a trapezoid:

Forward rectangle rule:

$$H_T(z) = \frac{aT}{(z - 1) + aT}$$

Trapezoid rule:

$$H_T(z) = \frac{2}{z - 1} \frac{aT}{z + 1} + \frac{2}{2}$$

This example illustrates a very important point. That is, the discrete transfer function equivalent is obtained from the given Laplace transform transfer function $H(s)$ by simple substitution of an approximation for $s$:

Forward rectangle rule:

$$s \leftarrow \frac{z - 1}{T}$$

Trapezoid rule:

$$s \leftarrow \frac{2}{T} \frac{z - 1}{z + 1}.$$
The trapezoid rule is also known as “Tustin's method”, "or the “bilinear transformation". The quality of the approximation is determined by several factors, most importantly the sampling rate \( f_s \). A decrease in sampling rate (increase in \( T = 1/f_s \)) means more time is available for control calculations. The best engineering choice is usually the slowest possible sampling rate that still meets all performance specifications. In terms of the simple transfer function \( H(s) \) above, a typical design goal in filter theory is to provide enough attenuation at half the sampling rate (\( f_s/2 \)) so that the noise above \( f_s/2 \), when aliased into lower frequencies by the sampler, will not be detrimental to performance. For a good reduction in the high frequency noise at \( f_s/2 \), the sampling rate is selected about 5 or 10 times higher than the filter breakpoint \( a \). The implication of this, if \( H(s) \) is a component of a larger (closed loop) system, is that sampling rates need to be on the order of 20 to 100 times faster than the system bandwidth, effectively providing a lower bound or sampling rate selection.

3.1.6 Hardware Characteristics

(i) Analog-to-Digital Converters (A/D) are devices which convert a voltage level to a digital word for use in the computer. That is, at each sample the A/D must convert a voltage level (for example, from a sensor) to the correct bit pattern and to hold that pattern until the next sample. The most common techniques for achieving this are counting schemes or successive approximation schemes. In counting methods, the input voltage is converted to a train of pulses (with frequency proportional to the voltage level), which are counted over a fixed period, resulting in a binary
representation of the voltage level. The successive approximation technique (usually faster) successively compares the input voltage to reference levels representing the various bits in the digital word.

(ii) Digital-to-Analog Converters (D/A) convert digital words from the computer to a voltage level for driving actuators or an oscilloscope for recording. The basic idea is that the binary bits are used to open or close switches, thereby routing some current through an appropriate network of resistors so that the correct voltage level is generated. Since no counting or iteration is required, D/A's are much faster than A/D's.

(iii) As we discussed in the first Laboratory, the data processing subsystem device used in this Lab is the DS1104 dSPACE card housed in an expansion slot of each computer. Each card has an interface board that has several channels of A/D, and several analog output channels (D/A's). Note that the input voltage range is limited to ±10V, and high levels at the input can damage the converter. Software to operate and access the A/D and D/A converters is available through Simulink.

3.2 Laboratory Preparation

(i) Given the triangular waveform, f(t), shown in Figure 3.6,

(a) Find the trigonometric Fourier series of the triangular waveform, f(t), over the period \(\left[-\frac{T}{2}, \frac{T}{2}\right]\).

(b) Find the average power, \(P_f\), of f(t).
(c) Find the average power $P_{i_2}$ for the first two harmonics of $f(t)$, and the average power $P_{i_6}$ for the first six harmonics of $f(t)$.

(d) Find the approximation error, $\eta_N$, for $N = 2$ and $N = 6$.

![Triangular Waveform](image)

Figure 3.6: Triangular Waveform

(ii) For the sampling system shown in Figure 3.7, let $x(t) = \cos(4\pi t) = \cos(2\pi f_0 t)$, with $f_0 = 2.0$ Hz. The signal $x(t)$ is being sampled using an ideal sampler,

\[
s(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\tau_s),
\]

with a sampling frequency of $f_s = 10$ Hz. The resultant signal is passed through a low-pass filter with a bandwidth of 5 Hz resulting in the output signal $y(t) = \cos(4\pi t)$. Letting $x(t) = \cos(2\pi f_{k_i} t)$ find two other input
frequencies, $f_{k_1}$ and $f_{k_2}$, called alias frequencies, which will result in

$$y(t) = \cos(4\pi t).$$

**Hint:** Graph all of the signals in the frequency domain.

![Figure 3.7: Ideal Sampling and Reconstruction System](image)

(iii) Given the closed loop system shown in Figure 3.8,

(a) Find the transfer function $G(z) = \frac{y(z)}{u(z)}$ if the dynamics of the plant are represented in the difference equation

$$y(k) = 1.2y(k-1) - 0.32y(k-2) + u(k-2).$$

(b) Find the value of $G_c(z) = K$ which will yield closed loop poles at $z = 0.6 \pm j0.8$.

![Figure 3.8: Closed-Loop System](image)
(iv) Indicate whether each of the following systems is stable or unstable and why.

(a) \( G(z) = 1 + 5z^{-1} \)

(b) \( G(z) = \frac{1 + 5z^{-1}}{1 + 0.7z^{-1}} \)

(c) \( G(z) = \frac{z^{-2}}{(1 + (0.65 + j0.8)z^{-1})(1 + (0.65 - j0.8)z^{-1})} \)

(v) Use the forward rectangle rule, \( s \sim \frac{z - 1}{T} \), and the trapezoidal rule, \( s \sim \frac{2z - 1}{T(z + 1)} \), to find the discrete-time equivalents \( H_F(z) \) and \( H_T(z) \) of the system \( H(s) = \frac{5}{s + 5} \) with \( f_s = 30 \) Hz. Convert the discrete-time equivalents \( H_F(z) \) and \( H_T(z) \) into the following form:

\[
H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \cdots}{1 + a_1z^{-1} + a_2z^{-2} + \cdots}.
\]

(vi) Use the forward rectangle rule to find the discrete-time equivalent \( H_F(z) \) of the system \( H(s) = \frac{10}{s(s + 25)} \) with \( T = 0.1 \) seconds. Is this a good approximation of the system? Justify your answer.

3.3 Procedure

The procedure for this lab is designed to exercise the capabilities of the data acquisition board, using Channel 5 of the A/D (12-bit).
Using a BNC/RCA cable prepared for this purpose, connect the output of the Analog Function Generator to the fifth analog input (Channel 5, 12-bit) on the DS1104 Interface board. Select the output of the Function Generator to be a triangular wave of amplitude 2 V and frequency 1 Hz.

Open Matlab and Simulink and create a simple Simulink model named lab2.mdl to acquire signals on ADC5. A sample of this is depicted below:

![Simulink Diagram](image)

Choose three different sampling frequencies, $f_s = 4$ Hz, $f_s = 12$ Hz, and $f_s = 1000$ Hz, and obtain a plot of the sampled output for each case using the ControlDesk acquiring capabilities in a layout of your own, similar to the previous Laboratory.

*Hint*: Choose the sampling time of the ZOH to be 10 times the step size of the simulation. For example, when $f_s = 4$ Hz, the sampling time of the ZOH must be set to 1/4 and the simulation step size should be 1/40. Also note that for high sampling rates such as 1000 Hz, you might need to downsample the acquired...
signal in the Capture Settings window (a factor of 10 is a good choice).

Downsampling simply means taking less data points per unit time.

- Select the sampling frequency \(f_s = 10\) Hz. As the input, select a sine wave, amplitude 2 V and frequency 0.5 Hz. Obtain a plot of the output. Increase the frequency of the input until the output looks like a 0.5 Hz sine wave; this should occur around 9.5 Hz and 10.5 Hz at the input. Record the two frequencies and obtain a plot of the output at both of these input frequencies.

3.4 Post-Laboratory Exercises

(i) Discuss the implications of aliasing and relate this concept with what you know about the frequency content of the triangular wave as determined in the Laboratory Preparation 1. Discuss your results and be sure to include a discussion on the accuracy of your results. Does the Nyquist Sampling Theorem hold for the triangular wave?

(ii) Discuss your results relative to the sampling process. If the input signal contains both the 9.5 Hz and 10.5 Hz components, what will happen to the output of the system? Can these components be distinguished in the output? Consideration of your answer to Laboratory Preparation 2 should help.

(iii) A numerical value can be represented with only limited precision in a digital computer. The stepsize drops by a factor of 2 for each additional bit. The effect of this limited precision shows up in the A/D conversion (which often has a smaller word size than the computer), multiplication truncation (experienced in this step),
and storage errors. Quantization error is related to a signal-to-noise ratio (SNR) that depends on the number of bits, B, and that is given by the following formula:

\[
\text{SNR} = (6 \text{ dB}) \times B
\]

The DS1104 interface board has another A/D with 16-bit word length. Compute the resolution and the SNR for both cases (12-bit and 16-bit) and compare the results.
CHAPTER 4

THE QUANSEN SRV-02 DC SERVO

The objective of this chapter is to perform time-domain system identification for the DC servomotor that is the main actuator for the Quanser SRV-02ET equipment series in the Lab. The time-domain system ID technique consists of applying a step input to the plant and measuring certain transient characteristics in order to estimate the transfer function.

4.1 Modeling

Before we attempt to identify a plant’s transfer function, it is always useful to refer to the physics of the plant and derive a mathematical model based on the governing differential equations. Once we determine a model that describes the plant as truly as possible, we are ready to conduct experiments (with this model structure in mind) and find out if the experimental model matches the theoretical transfer function. This leads us to a simple definition of “system identification” for our purposes: Deduce a model of the physical system from available measurements. Although modeling issues are beyond the
scope of this course, it is important to mention a few remarks that relate modeling and control.

A truth model is a set of equations that describe the dynamics of a system as well as the effects of disturbances, parameter variations and uncertainties, all of which are determined based on physical principles or system identification techniques. A design model is a simplified version of the truth model and is used for controller design. The design model can be obtained from the truth model by order reduction, linearization or other simplifications. A truth model is never a perfect representation of a physical system because it uses approximations and it ignores some characteristics of the plant. It follows that a design model can never represent the system perfectly since it is a simplified version of the truth model itself. To evaluate the accuracy of a model, we carry out experiments on the real plant and we perform simulations of the model in order to compare the results of the simulations to the actual plant data. This comparison allows us to correct and improve the model. When the design model does not exhibit the same essential properties as the truth model, the control engineer should consider modifying the design model to account for properties such as stability, rate of decay, etc. Since the controller design is based on the design model, the controller may have difficulty achieving its task if the design model is not close enough to the truth model.

The block diagram for the position Servo SRV-02 is shown in Figure 4.1. It consists of a DC servomotor with a built-in gearbox with 14:1 gear ratio. The output of the gearbox drives a potentiometer and an independent output shaft to which a load can be
attached. For this Laboratory, the servo is used in the low gear ratio (1:1) configuration (three identical 72-tooth external gears).

\[ \text{Figure 4.1: Block Diagram of Position Servo} \]

Now let us develop the state-space model and the transfer function of the motor. We start by writing the differential equations, using the following variables:

- \( \theta \) is the angle of the output shaft
- \( \omega_m \) is the angular velocity of the motor shaft
- \( \omega_l \) is the angular velocity of the output shaft
- \( K_g \) is the gear ratio
- \( T_m \) is the motor torque
- \( T_0 \) is the output torque after the gearbox
- \( J_m \) is the motor inertia
- \( J_l \) is the load inertia

The parameter values needed for modeling this servomotor are given in [4] as: \( R_m=2.6 \, \Omega \), \( K_m=0.00772 \, \text{V/rad-s}^{-1} \), \( K_g=14:1 \), \( J_m=3.87\times10^{-7} \, \text{kg-m}^2 \) and \( J_l=3\times10^{-5} \, \text{kg-m}^2 \).
The governing electrical equation is:

\[ V_{in} = I_m R_m + K_m \omega_m \]  \hspace{1cm} (1)

The governing mechanical equations are:

\[ \omega_m = K_g \omega_1 \]

\[ T_o = K_g T_m \]

\[ T_o = K_g T_m = K_g \left( J_m \dot{\omega}_m + \frac{J_l}{K_g} \dot{\omega}_l \right) = J_m K_g^2 \dot{\omega}_1 + J_l \dot{\omega}_1 = \left( J_m K_g^2 + J_l \right) \dot{\omega}_1 \]

Let \( J_{eq} = \left( J_m K_g^2 + J_l \right) \) be the equivalent inertia seen by the motor, which implies that

\[ T_o = J_{eq} \dot{\omega}_1 \]

Given the above, the torque-current relationship is:

\[ T_m = K_m I_m \Rightarrow I_m = \frac{T_m}{K_m} = \frac{T_o}{K_m K_g} = \frac{J_{eq} \dot{\omega}_1}{K_m K_g} \]

Equation (1) becomes:

\[ \frac{R_m}{K_m K_g} J_{eq} \dot{\omega}_1 + K_m K_g \omega_1 = V_{in} \]

Let \( x_1 = \theta, x_2 = \dot{\theta} = \omega_1 \) be the states of the system, \( u = V_{in} \) the input and \( y = \theta \) the output; then a state-variable representation is possible:

\[ \dot{x} = Ax + Bu \]

\[ y = Cx \]

where

54
\[
A = \begin{bmatrix} 0 & \frac{1}{(K_m K_g)^2} \\ 0 & -\frac{R_m J_{eq}}{R_m J_{eq}} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{K_m K_g}{R_m J_{eq}} \end{bmatrix}, \quad C = [1 \quad 0]
\]

Substituting for the physical parameters we get:

\[
A = \begin{bmatrix} 0 & 1 \\ 0 & -41.5769 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 384.6154 \end{bmatrix}
\]

Re-writing the differential equation as a function of \( \theta \) results in:

\[
\frac{R_m J_{eq}}{K_m K_g} \ddot{\theta} + K_m K_g \dot{\theta} = V_{in}
\]

Taking Laplace transforms of both sides, and using properties of Laplace transforms, leads to the following s-domain representation:

\[
\frac{R_m J_{eq}}{K_m K_g} s^2 \theta(s) + K_m K_g s \theta(s) = V_{in}(s)
\]

or, in terms of the transfer function from input \( V_{in}(s) \) to output \( \theta(s) \),

\[
G(s) = \frac{\theta(s)}{V_{in}(s)} = \frac{1}{s \left( \frac{R_m J_{eq}}{K_m K_g} s + K_m K_g \right)}
\]

Substituting for the parameters leads to

\[
G(s) = \frac{1}{s(0.0026s + 0.1081)}
\]

This transfer function indicates that the linearized plant has system poles at \( s=0, s=-41.58 \).
Pre-Lab Question 1:

Re-write the differential equation as a function of $\omega$, then take the Laplace transform of both sides and deduce the velocity transfer function:

$$G_\omega(s) = \frac{\Omega(s)}{V_{in}(s)}$$

4.2 Transfer Function Estimation

In this development, the velocity transfer function $G_\omega(s)$ is of first order; therefore we will simply perform system identification on the actual system to deduce $G_\omega(s)$ then integrate the result (i.e. multiply it by $1/s$) to obtain the position transfer function $G(s)$. In order to do so, we apply a step input to the motor and measure the settling time or rise time.

A general first-order transfer function estimate with output $Y(s)$, and input $R(s)$, can be represented as:

$$\frac{Y(s)}{R(s)} = G(s) = \frac{K}{s + \sigma}.$$  

With a step input $R(s) = \frac{1}{s}$, the output is $Y(s) = \frac{K}{s(s + \sigma)}$. Hence, the steady-state output, deduced from the final value theorem, is equal to $K$. The only remaining unknown is $\sigma$, and it is estimated using the 10% to 90% rise time method. This method consists of measuring the time it takes the response to rise from 10% to 90% of the steady-state value. Clearly, one can use only one point on the step response curve to calculate $\sigma$; however, it is more accurate to pick two points that are spread apart on the plot. The
output can be written as: \( y(t) = K\left(1 - e^{-\sigma t}\right)u(t) \). Let \( t_1 \) and \( t_2 \) be the times when \( y(t) \) is equal to 10% of \( K \) and 90% of \( K \), respectively. This translates to:

\[
0.1K = K\left(1 - e^{-\sigma t_1}\right) \Rightarrow e^{-\sigma t_1} = 0.9 \\
0.9K = K\left(1 - e^{-\sigma t_2}\right) \Rightarrow e^{-\sigma t_2} = 0.1
\]

Dividing the two equations we get:

\[
e^{\sigma(t_2 - t_1)} = 9 \Rightarrow \sigma = \frac{\ln 9}{t_2 - t_1}
\]

Pre-Lab Question 2:

Figure 4.2 shows a unit step response. Assume that this is a first-order response and apply the rise-time method to determine the transfer function of the system.
4.3 Laboratory Procedure

4.3.1 The SRV-02ET DC Servomotor

Figure 4.3 shows a backside view of the DC motor that we will use in this Laboratory\(^1\). The different ports on the back of the motor can be connected to the Universal Power Module described in Chapter 2 or directly to the DS1104 Interface.

---

\(^1\) Picture courtesy of Prof. John Watkins, Systems Engineering Department, U.S. Naval Academy [5].
board. Figure 4.4 shows the motor set up in low gear ratio and Figure 4.5 shows the high gear ratio setting. We will use the low gear ratio configuration in this Laboratory. Since our aim is to estimate the velocity transfer function, we will need to use the speed sensor attached to the motor, i.e. the tachometer. The encoder and potentiometer ports will remain unconnected because we are not interested in position sensing. The remaining port is the voltage input to the motor (top left).

![SRV-02ET DC Servomotor][5]

---

Figure 4.3: SRV-02ET DC Servomotor [5]
4.3.2 Connections

- Connect the tachometer port of the motor to the S3 port of the Universal Power Module (UPM) using an “analog sensor cable” (6-pin Mini to DIN/6-pin Mini DIN cable).
- Connect the input voltage port of the motor to the “To load” port of the UPM using a “to load cable” (6-pin DIN/ 4-pin DIN cable).

- Connect the D/A port of the UPM to the analog output 0 on the DS1104 interface board using a “D/A cable” (5-pin DIN-mono/ RCA cable).

  *Reminder:* analog output 0 on the DS1104 interface board corresponds to analog output #1 in Simulink.

- Connect the A/D port of the UPM to the analog input 4 on the DS1104 interface board using an “A/D cable” (5-pin DIN-stereo/ 4 × RCA cable).

  *Reminder:* analog input 4 on the DS1104 interface board corresponds to analog input #5 in Simulink.

4.3.3 Simulink Model

- Open Matlab and Simulink and create a Simulink model named lab3.mdl.

- Add a step input that starts at time t = 4 and has a final value of 4 Volts.

- Let the step input go to two paths:

  1) To a transfer function block that represents the theoretical transfer function of the motor.

  2) To the analog output #1 that is connected externally to the motor (don’t forget the DAC gain).

- Add an analog input block with the corresponding gain of 10 followed by a gain of 1/14 gain to account for the internal gear ratio.
- Set the step size of the solver to 1 ms, and the simulation time to 8 s.

Figure 4.6 below shows the detailed Simulink model described above:

![Simulink Diagram]

**Figure 4.6: Simulink Diagram**

4.3.4 Compiling and Running

- Compile the model and create a new experiment in ControlDesk called Lab3. Add a layout that contains a plotter showing the measured output of the real plant in Volts.
- When the experiment stops, save the results in a *.mat file as described in Laboratory #1.
The tachometer is the sensor used in this experiment. It is a speed sensor with a gain that enters in the overall open-loop transfer function. Therefore, the data that we acquired from the step response characterizes the following transfer function:

\[
\frac{\Omega}{V} = \frac{K\sigma}{s + \sigma} = \frac{K_{\text{tach}}K_{\text{motor}}}{s + \sigma},
\]

whereas we are interested in the motor transfer function:

\[
\frac{\Omega}{V} = \frac{K_{\text{motor}}\sigma}{s + \sigma}.
\]

Perform the following steps\(^2\) in order to determine the value of \(K_{\text{tach}}\) (in V/rad/s):

- Align the screw of the center gear at zero degrees.
- In the Simulink model, change the step input time to 7 s.
- Compile and run the experiment.
- Reduce the end simulation time until the motor makes only one revolution.
- Record the stop time and let \(t_{\text{stop}}\) be the time elapsed from the rising edge of the step input to the simulation stop time.
- Save the result in a *.mat file. Assuming that the signal reaches steady state in this time interval, compute the mean of the signal and call it \(v_{\text{avg}}\).

**Hint:** You can use the Matlab plot statistics tool, provided that you plot the range of interest only.

This method basically computes an approximation of the sensor constant by using an average approach. The sensor gain is the ratio of the average voltage per average angular velocity during one revolution of the DC servomotor.

---

\(^2\) Procedure due to Prof. John Watkins [5].
4.4  Post-Laboratory Exercises

(i) Using the acquired plot, compute the values of $K$ and $\sigma$ as described in Section 4.2.

*Note*: remember that the step input used was 4.

(ii) Compute the value of $K_{\text{tach}}$ as follows:

- Compute the average velocity during one revolution: $\omega_0 = \frac{2\pi}{t_{\text{stop}}}$.

- Compute $K_{\text{tach}}$ using the mean, $v_{\text{avg}}$ computed earlier: $K_{\text{tach}} = \frac{v_{\text{avg}}}{\omega_0}$.

(iii) Compute the value of $K_{\text{motor}}$ and write the motor transfer function. Compare this estimate with the model derived in the pre-lab.
The objective of this chapter is to employ cascade gain compensation in a unity feedback configuration to adjust the damping of a closed loop system. Choice of compensating gains will be determined via computer-aided design using root locus and frequency response techniques.

5.1 Background

In this laboratory we employ both the root locus and frequency response methods for designing gain compensation in a unity feedback configuration.

5.1.1 Root Locus

Consider the control system shown in Figure 5.1 which has a closed loop transfer function (from r to y) given by

\[
\frac{Y(s)}{R(s)} = \frac{G}{1 + GH}
\]

The quantity GH is called the “loop transfer function” and in general is represented by
where \( N(s) \) and \( D(s) \) are polynomials in the complex variable \( s \), and \( K \) is the loop gain factor. Thus,

\[
\frac{Y(s)}{R(s)} = \frac{G}{1 + KN/D} = \frac{GD}{D + KN}
\]

and the closed loop poles are roots of the characteristic equation

\[
D(s) + KN(s) = 0.
\]

The locations of the closed loop poles change as \( K \) varies, and the locus of the poles (roots of the characteristic equation) plotted in the S-plane as a function of \( K \) is called a “root locus”. Note that when \( K = 0 \), the poles are the roots of \( D(s) \), which are the poles of the loop transfer function (“open loop poles”). Also, as \( K \) gets larger the closed loop poles approach the roots of \( N(s) \) (the open-loop zeros) since the \( KN(s) \) term dominates.
the characteristic equation in that case. The bottom line: as $K$ is increased from 0 to $\infty$, the loci of the closed loop poles begin at the open loop poles and move to (and end at) the open loop zeros.

5.1.2 Root Locus Design

The root locus method can be used effectively for feedback control design because of the way it graphically displays the closed loop poles as a function of the gain factor. Gain factor compensation, the subject of this Laboratory, is the design accomplished by choosing a value of $K$ which results in “satisfactory” closed loop behavior. If system specifications (such as specified gain margin or closed loop damping) cannot be met with simple gain compensation, another form of compensation must be designed to alter the root locus as required; this is the subject of future Laboratories.

Two important concepts in classical control design are the gain and phase margin. The gain margin is the factor by which the open loop plant $G(s)$ can be multiplied before the closed loop system becomes unstable. The gain margin is easily determined from the root locus according to

$$\text{gain margin} = \text{value of } K \text{ at } j\omega\text{-axis crossover} = K_{cr}.$$  

Note that the gain margin is infinite if the root locus remains in the left-half $S$-plane for all $K$. The value for $K$ at the $j\omega$-axis crossover is often referred to as the “critical gain”, or $K_{cr}$ (we use this terminology and notation in later laboratories). The phase margin is the amount of phase which may be added to the system before the system
becomes unstable. Using root locus conditions, the “angle-criterion” is employed to find the point $j\omega_c$, the “gain crossover frequency”, such that

$$|GH(j\omega_c)| = 1,$$

phase margin $= 180^\circ + \angle GH(j\omega_c)$.

Note that these expressions are simply the magnitude and phase relationships corresponding to the characteristic equation: $1 + GH(s) = 0$, or $GH(s) = -1$.

Finally, the damping ratio, a very important design parameter, can be determined easily from the root locus. The gain factor $K$ required to give a specified closed loop damping ratio $\zeta$ for a given loop transfer function is found by first drawing a line from the origin at an angle of $\pm 0 = \cos^{-1} \zeta$. Then the gain factor at the point of intersection with the root locus is the required value of $K$. Again, for calculations all frequencies are in rad/s.

5.1.3 Bode Design

Simply stated, design of feedback control systems via Bode techniques amounts to shaping the Bode magnitude and phase plots to satisfy closed loop system specifications. In this laboratory, we can only give an introduction to design techniques using Bode analysis. Using this foundation, the interested student may wish to pursue more advanced material to fully realize the power of such methods.

Typical specifications are conveniently expressed in the frequency domain, and include the gain and phase margin and, for steady-state response specifications, error
constants. Often times adjustment of the gain factor $K$ is sufficient to satisfy system specifications. This can be done in a somewhat direct manner with Bode plots, or sometimes it is more convenient to use the so-called “Bode gain” $K_B$ which may be expressed as

$$K_B = \frac{K \prod_{i=1}^{m} z_i}{\prod_{i=1}^{n} p_i}$$

where the (-$p_i$) and (-$z_i$) are the finite poles and zeros, respectively, of the loop transfer function. An increase or decrease of (positive) $K$ (or $K_B$) shifts the magnitude plot up or down, and does not affect the phase plot. Thus, since the shape of the log-magnitude curve does not change, but rather the entire curve is shifted, a simple way to adjust for a change in $K$ (or $K_B$) is to merely change (shift) the magnitude scale of the plot. For example, if $K$ (or $K_B$) is doubled, the magnitude plot would be shifted up by $20\log_{10}2 = 6.02$ dB. The effects of dynamic compensators on the Bode plot will be treated in upcoming Laboratories.

The “relative stability” measures, gain margin and phase margin, are described in terms of the system open loop frequency response (that is, the frequency response of the loop transfer function), and are therefore easily determined from the Bode plots of $G(s)H(s)$:

(i) Since 0 dB corresponds to a magnitude of 1, the gain margin is the number of dB that $|GH(j\omega)|$ is below 0 at $\omega_\pi$, the “phase crossover frequency” (note that $\angle GH(j\omega_\pi)=180^\circ$).
(ii) The phase margin is the number of degrees $\angle GH(j\omega_c)$ is above $-180^\circ$ at $\omega_c$, the “gain crossover frequency” (note that $|GH(j\omega_c)| = 1$).

With sufficiently accurate plots, the gain and phase margins may be estimated from the actual data.

5.1.4 Steady-State Error

One of the primary benefits of using a feedback system is that an error quantity is established and regulated (minimized). For the no-unity feedback system shown previously in Figure 5.1, the error quantity, $e$, is driven to zero by the action of feedback. However, this error quantity does not correspond to the system error, $r - y$, which is also referred to as the tracking error of the closed loop system. For a unity feedback system, $H = 1$, and thus $e = r - y$, which is the system error for the closed loop system. The system error for a unity feedback configuration is therefore represented in the S-domain as follows:

$$E(s) = R(s) - Y(s) = \frac{R(s)}{1 + G(s)}$$

The steady-state error, $e_{ss}$, is determined by using the Final Value Theorem:

$$E_{ss}(s) = \lim_{s \to 0} s E(s) = \lim_{s \to 0} s \frac{R(s)}{1 + G(s)}.$$

The steady-state error, $e_{ss} = \lim_{t \to \infty} e(t)$, depends directly on the system input, $r$. This dependency can be made explicit by representing the steady-state error in terms of an
error constant that is defined for various test inputs, \( r \). The error constant and steady-state error, \( e_{ss} \), for several typical system test inputs are defined as follows:

(i) Position Error Constant \( K_p \): measure of the steady-state error when the input, \( r \), is a unit step (\( R(s) = \frac{1}{s} \)),

\[
K_p = \lim_{s \to 0} G(s)
\]

\[
e_{ss} = \frac{1}{1 + K_p}
\]

(ii) Velocity Error Constant \( K_v \): measure of the steady-state error when the input is a unit ramp function (\( R(s) = \frac{1}{s^2} \)),

\[
K_v = \lim_{s \to 0} s G(s)
\]

\[
e_{ss} = \frac{1}{K_v}
\]

(iii) Acceleration Error Constant \( K_a \): measure of the steady-state error when the input is a unit parabolic function (\( R(s) = \frac{1}{s^3} \)),

\[
K_a = \lim_{s \to 0} s^2 G(s)
\]

\[
e_{ss} = \frac{1}{K_a}
\]

The general form for the forward path transfer function, \( G(s) \), is given as follows:

\[
G(s) = \frac{N(s)}{s'(s + p_1)\ldots(s + p_n)}
\]
where N(s) is the numerator polynomial, \((-p_i)\) are the system poles, and \(\ell\) is an integer indicating the system type number. For a type-zero system \((\ell = 0)\) with the typical test inputs mentioned above, only the steady-state error to a step input is finite. For a type-one system \((\ell = 1)\) with the typical test inputs mentioned above, finite steady-state errors occur for a step input, \(e_{ss} = 0\), and for a ramp input. For a closed loop step response, the steady-state error is \(e_{ss} = r_{ss} - y_{ss}\), where \(r_{ss} = \lim_{t \to \infty} r(t)\) is the steady-state value of the input and \(y_{ss} = \lim_{t \to \infty} y(t)\) is the steady-state value of the output.

5.2 Laboratory Preparation

![Figure 5.2: Closed-Loop System](image)

Given the closed loop unity feedback system in Figure 5.2 with the transfer function from input voltage to angular position of the SRV-02 DC servo,

\[
G(s) = \frac{\theta(s)}{V_{in}(s)} = \frac{1}{s(0.0026s + 0.1081)}
\]
and given the reference input $\theta_d$, the error signal is $e = \theta - \theta_d$. In this laboratory the compensator is a proportional gain: $G_c(s) = K$, and the goal is to achieve position tracking on the DC servomotor.

(i) What is the “system type number” of the model transfer function? Compute the three error constants and the corresponding steady state errors for a step, a ramp and parabolic input.

(ii) Use the Matlab command “rltool” to plot the root locus of this unity feedback system. The “rltool” command is a very powerful tool for single-input single-output (SISO) classical control design. When you type “rltool” at the Matlab command prompt, the SISO design GUI will load. Before you start using the GUI, make sure you define your system in the workspace by typing the following commands:

```matlab
>> num = 1;
>> den = [0.0026 0.1081 0];
>> sys = tf(num,den);
```

Now click on “File” and select “Import” and the screen of Figure 5.3 will appear.
Select your system under “SISO Models”, and click on the arrow that imports it to G (the plant). Note the other blocks are equal to unity, which agrees with our closed loop unity feedback control system. Click OK when you are done, and you will observe a root locus plot in the main window. Get a plot of the root locus of the system.

(iii) Right click on the root locus plot and select “Design Constraints/ New”. Our design constraint for this Laboratory is going to be a damping ratio greater than or equal to $1/\sqrt{2}$ (which translates into a percent overshoot that is less than 4.33%), specified as shown in Figure 5.4.
Pan the roots on the root locus (i.e., slide the cursor using the mouse) until you hit the border of the constraints region. This gives you the maximum value of the gain $K$ for which the damping ratio is greater than $\frac{1}{\sqrt{2}}$. Record that value.

(iv) With the value of $K$ that you obtained in the previous step, click on “Tools/Loop Responses” (or “Analysis” depending on the Matlab version) and select both “closed-loop step” and “control signal step” in order to observe a plot of the closed loop step response and the control signal response. Explore the plots and find out how to add useful annotations, and then get a printout. What are the values of steady state error, percent overshoot, rise time and settling time? Derive the theoretical values of these characteristics and compare.
5.3 Laboratory Procedure

5.3.1 Connections

Since we are performing position control on the servomotor, we will need to use the encoder for position sensing. Also, remember that we are using the motor in low gear ratio, with an inertia added on the top. Use the following procedure:

- Connect the UPM “To Load” port to the motor input.
- Connect the UPM “From D/A” port to analog output 0.
- Connect the encoder port of the motor (see Fig. 3.3) to the ENC1 on the interface board.

5.3.2 Simulink Diagram

- Open Matlab and Simulink and create a new model called Lab4.mdl. Set the simulation parameters to a fixed step size of 1 ms and a simulation time of 30 seconds.
- The model should appear as shown in Figure 5.5.
- The unit step input should start at time \( t = 10 \).
- The Rate Limiter block reduces the slope of the step input so as to avoid a very high frequency component input to the motor, which could potentially cause damage. Set the slew rates to \( \pm 1000 \).
- The gain \( K \) is the proportional gain used for compensation. Initially, use the gain that you obtained in your preliminary design.
- The Saturation block limits the control input so as to avoid a high voltage input to the motor, which could also cause damage. Set the saturation limits to \( \pm 5 \text{ V} \).
- The “Enc position” block allows us to acquire the motor shaft position from the encoder. Select Encoder 1, and include an “Encoder Master Setup” block that is necessary for the latter block.

Note that the output of the encoder is in counts, and we need to convert it to radians in order to compare it with the setpoint and produce an error signal. We know that the encoder has a counter that outputs 1024 counts \( (2^{10}) \) for a full revolution \( (360^\circ) \); therefore, the gain to go from counts to degrees is \( 360/1024 \), and it should be included in the Simulink diagram.

5.3.3 ControlDesk and Results

- Open dSPACE ControlDesk and create a new experiment called Lab4, then compile the Simulink model.
Create a new layout and add it to the experiment. Include a plot of the input and output.

Add a numerical input that controls the gain from the dSPACE GUI. You will need to change this gain until you obtain the best response (less than 5% overshoot if possible), since the actual experimental results will differ from the theoretical derivations.

Acquire the data from the plot to a *.mat file. Save the experiment

5.4 Post-Laboratory Exercises

(i) From the saved data, evaluate the steady state error, rise time, settling time, percent overshoot.

(ii) Compare those values with the results you obtained in the pre-laboratory assignment.

(iii) What is the value of the gain you finally tuned in dSPACE? How did you tune this gain?

(iv) Explain why the value of the gain is different than the one found in the pre-laboratory.
CHAPTER 6

LAG COMPENSATION FOR SPEED CONTROL OF A DC SERVO

The objective of this chapter is to utilize computer-aided design tools with techniques from Bode and root locus design methods, to design and subsequently implement lag compensation in a unity feedback configuration in order to perform speed control on the SRV-02 DC servomotor.

6.1 Background

6.1.1 Overview

The general unity feedback configuration to be investigated is shown in Figure 6.1. The general compensation network, $G_c(s)$, is cascaded with the plant, $G(s)$, in order to provide a desired closed loop response. The transfer function for a general first order cascade compensation network is given by:

$$G_c(s) = \tilde{K}_c \frac{1 + \alpha s}{1 + \alpha \tau s} = K_c \frac{s + a}{s + b}.$$  

Higher order compensators can be formed by simply cascading first order compensators together. For this laboratory, the compensation embodied in $G_c(s)$ is that of a lag compensation network.
A lag compensator has a low pass filter characteristic; that is, the pole \( b \) is closer than the zero \( a \) to the \( j\omega \)-axis (\(|a| > |b|\) or \( \alpha > 1 \)). If the effect of the lag compensator's zero is negligible, that is, \(|a| \gg |b|\), and the pole of the lag compensator is near zero, then, for frequencies \( \omega \ll |a| \), the lag compensator approximates a pure integrator:

\[
\lim_{b \to 0} G_c(s) \approx \frac{K_c a}{s}.
\]

![Figure 6.1: Cascade lag compensation closed loop system](image)

The effects of lag compensation on the overall (closed-loop system) performance can be summarized as follows:

(i) decrease in bandwidth

(ii) increase in error constant (thus a decrease in steady-state error)

(iii) improvement in gain or phase margin

(iv) more sluggish system (dominant time constant increased)

When the magnitude of resonance, \( M_r \), and the accompanying resonant frequency, \( \omega_r \), of a unity feedback control system are considered acceptable, but the steady-state error, \( e_{ss} \),
is too large, an effective scheme is to increase the gain (thereby improving the error constants) without significantly changing other (acceptable) system characteristics. This can be done with a cascade lag compensator in a unity feedback configuration. As an example, consider the case of a type 0 system where improvement in steady-state accuracy is desired, which is the case for the servo system speed control problem.

The velocity transfer function modeled in Chapter 2 is given by

$$G_v(s) = \frac{1}{0.0026s + 0.1081}$$

Without the compensator, the position error constant is

$$K_p = \lim_{s \to 0} G(s),$$

whereas with the compensator in the loop, the result is

$$\tilde{K}_p = \lim_{s \to 0} G_c(s)G(s) = \tilde{K}_c K_p.$$  

The compensator gain $\tilde{K}_c$ directly determines the position error constant since

$$\tilde{K}_c = \frac{\tilde{K}_p}{K_p}$$

6.1.2 Bode Design

Ideally, by our objectives, we would like to increase $\tilde{K}_c$, and therefore $\tilde{K}_p$, while having a negligible effect on $M_r$ and $\omega_r$. Recall from our discussion in Chapter 5 that the closed loop characteristic equation (denominator of the closed loop transfer function set equal to zero) is given by

$$1 + G_cG(s) = 0$$

81
and determines the closed loop poles (roots of the characteristic equation). Thus, we can write this expression as

\[ |G_c(s) G(s)| = 1 \]
\[ \angle[G_c(s) G(s)] = -180^\circ \]

simply because \( G_c(s)G(s) = -1 \). This magnitude and angle criterion represents the critical stability boundary. Returning to the discussion above, then, the lag compensator (in addition to the desirable of effect of allowing an increase in error constant) has a negative angle which moves the original log magnitude plot to the left, or closer to the \((-180^\circ, 0 \text{ dB})\) point. This has a destabilizing effect and reduces \( \omega_r \). To limit this effect, \( G_c(s) \) is designed to introduce only a small angle, generally less than \(-5^\circ\), at the original resonant frequency \( \omega_r \). Ordinarily, \( G_c(s) \) is selected so that the magnitude of its angle is 5° or less at the original phase margin frequency \( \omega_c \); recall that \( \omega_c \) is the frequency where the magnitude is one, and is precisely where the phase margin is defined. The log magnitude and phase angle expressions for \( G_c(s) \) with \( K_c = 1 \) are:

\[
20 \log_{10}|G_c(j\omega)| = 20 \log_{10}|1 + j\omega\tau| - 20 \log_{10}|1 + j\omega\alpha\tau|
\]
\[
\angle G_c(j\omega) = \angle(1 + j\omega\tau) - \angle(1 + j\omega\alpha\tau)
\]

Figure 6.2 shows a typical family of frequency response curves for a lag compensator as a function of \( \alpha \) with \( K_c = 1 \). An inspection of those curves reveals much about the attenuation and phase angle contribution of lag compensators. Observe that as \( \alpha \to \infty \), the lag compensator approximates a pure integrator.
6.1.3 Root Locus Design

In keeping with our discussion above, we recall that the primary objective of the lag compensation scheme is to increase the gain and/or phase margin while reducing the steady-state error. This can be accomplished by using root locus techniques in the system design. The basic idea is to achieve the change in error constant while maintaining (approximately) the location of dominant poles. The following procedure may be utilized...
using the root locus to design lag cascade compensation. As a first step, the compensator pole and zero should be placed very close together. Because in the root locus the closed loop poles approach the open loop zeroes as the gain increases, placing the compensator pole and zero close together has the effect of leaving the overall root locus relatively unchanged. If the angle contributed by $G_c(s)$ at the original closed loop dominant pole is less than about 5° (just a rule of thumb), the new locus will be only slightly displaced. This implies that the “dominant” pole(s) will be essentially unchanged, satisfying the design goal that the transient response be left unchanged. On the other hand, the values of $K_p$ and $\tilde{K}_p$ differ (approximately) by a factor of $\alpha$ ($\tilde{K}_p \approx \alpha K_p$). That is, the gain required to produce the new poles (approximately equal to the original poles) increases by the factor $\alpha$, which is the ratio of the compensator pole and zero. In summary, the necessary conditions for the design of $G_c(s)$ are

1. the pole and zero are close together

2. the ratio $\alpha$ must approximately equal the desired increase in gain.

These requirements can be achieved by placing the pole and zero of $G_c(s)$ very close to the origin. Although these statements are easiest to apply to a type-0 system, the same conditions apply to higher system types.
6.2 Laboratory Preparation

(i) Given the closed loop system shown in Figure 6.1 with the transfer function of the lag compensator, \( G_c(s) = \frac{K_c}{s + \frac{1}{\tau}} \), perform the following:

(a) Show that \( \tau = \frac{1}{a} \), \( \alpha = \frac{a}{b} \), and \( \tilde{K}_c = \alpha K_c \).

(b) Observation of Figure 6.2 demonstrates that the gain decrease (attenuation) in dB due to the lag compensator with \( \tilde{K}_c = 1 \)

\[ \lim_{\omega \to \infty} 20 \log_{10} \left| G_c(j\omega) \right| = -20 \log_{10} \alpha. \]

Prove this.

(ii) Given the closed loop system shown in Figure 6.1 with the transfer function of the plant, \( G(s) = \frac{1}{0.0026s + 0.1081} \) specified in Chapter 4, and given the following desired form for the transfer function of the compensator, \( G_c(s) = \frac{s/a + 1}{s/b + 1} \), design a lag compensator using Bode techniques, according to the following design objective.

**Bode Design Objective:** Design a lag compensator that will alter the phase response of the original plant by increasing the phase margin and thus allowing for more uncertainty regarding system phase.
(a) Plot the Bode plot (open-loop) of the uncompensated system, that is, $G_c(s) = 1$, and find the gain margin, and the phase margin (see Matlab command “margin”).

(b) A compensator is to be designed such that the closed loop system has a desired phase margin of $125^\circ$. Allowing for an extra $5^\circ$ of phase lag from the compensator, determine the desired crossover frequency, $\omega_{c_{des}}$, where the phase of $G(s)$ equals $-50^\circ$, $(-180^\circ+125^\circ+5^\circ)$.

(c) Place the zero of $G_c(s)$ one decade below $\omega_{c_{des}}$ ($a = \omega_{c_{des}}/10$) in order to keep the compensator phase lag contribution to less than $5^\circ$.

(d) Measure the gain, $K_{\omega_{des,\alpha}}$, of $G(s)$ in dB at $\omega_{c_{des}}$. Because the lag compensator $G_c(j\omega)$ contributes an attenuation of $-20 \log_{10} \alpha$ in dB, as was shown 1.(b), and because it is desired that the compensated magnitude plot cross the 0 dB line at the frequency $\omega_{c_{des}}$, the following equation must be satisfied:

$$-K_{\omega_{des,\alpha}} = -20 \log_{10} \alpha.$$ 

Equivalently, $K_{\omega_{des,\alpha}} = \alpha$. Solve for $b$.

(e) Obtain an open loop Bode plot for the compensated system using the loop transfer function $G_c(s)G(s)$.

(f) Obtain a closed loop step response for the system shown in Figure 6.1 using the lag compensator $G_c(s)$ just designed.

(g) Let $G_c(s) = 1$ and obtain the closed loop step response.
Given the closed loop system shown in Figure 6.1 with the transfer function of the plant, \( G(s) = \frac{1}{0.0026s + 0.1081} \) specified in Chapter 4, and given the following desired form for the transfer function of the compensator, \( G_c(s) = \frac{\bar{K}_c s^a + 1}{s^b + 1} \), design a lag compensator using root locus techniques, according to the following design objective.

**Root Locus Design Objective:** Design a lag compensator which decreases the steady-state error of the original plant and simultaneously shapes the time response of the original plant.

(a) Find the error constant, \( K_p \), of the uncompensated plant.

(b) Choose the desired ratio of the lag compensator pole and zero, \( \alpha = 20 \). Since the objective is to decrease the steady-state error while having a negligible effect on the closed loop response, select \( a \) and \( b \) such that \( a < 1 \) and the desired \( \alpha \) is achieved.

(c) Using the “rltool” command, plot the root locus for the compensated system, that is, with \( G_c = K_c \frac{s^a + 1}{s^b + 1} \). Shape the time response of the original plant by increasing the value of \( K_c \) until you achieve the best transient response (minimize steady-state error while keeping the input to the motor under \( \pm 5V \)). Record this value of \( K_c \).
Hint: you can monitor the closed loop step responses while you change the
gain in the SISO design GUI.

(d) Obtain a closed loop step response for the system shown in Figure 6.1 using
the lag compensator $G_c(s)$ just designed. Be sure to show the steady-state
error, and the error constant, $\bar{K}_p$, for the compensated system $G_c(s)G(s)$ to
check that the design objectives have been met.

6.3 Laboratory Procedure

6.3.1 Connections

Since we are performing speed control on the servomotor, we will need to use the
tachometer for velocity sensing. Also, remember that we are using the motor in low gear
ratio, with an inertia added on the top. Use the following procedure (see Figure 4.3):

- Connect the UPM “To Load” port to the motor input.
- Connect the UPM “From D/A” port to analog output 0 (Channel 1 in software).
- Connect the tachometer to the S3 “From Analog Sensors” UPM port.
- Connect the UPM “To A/D” port to analog input 4 (Channel 5 in software).

6.3.2 Simulink Diagram

- Open Matlab and Simulink and create a new model called Lab5.mdl. Set the
  simulation parameters to a fixed step size of 1 ms and a simulation time of 15
  seconds.
- The model should appear as shown in Figure 6.3.

![Simulink Diagram](image)

**Figure 6.3: Simulink Diagram**

- The step input should start at time $t = 4$ and have a final value of 4.
- Set the slew rate of the rate limiter to $\pm 1000$.
- Set the saturation limits to $\pm 5$.
- The Lag Compensator block is the transfer function of the compensator designed in the pre-laboratory. We will implement both the Bode design and the root locus design methods next.

### 6.3.3 ControlDesk and Results

- Open dSPACE ControlDesk and create a new experiment called Lab5, then compile the Simulink model using the compensator obtained in the pre-laboratory from the Bode design method.
- Create a new layout and add it to the experiment. Include a plot of the input and output.
- Acquire the data from the plot to a *.mat file.
- Change the compensator transfer function and use the one obtained from the root locus method. Compile the model and acquire the plot to a new *.mat file.
  
  **CAUTION:** In case high frequency switching occurs and you hear a “buzz” in the motor, immediately stop the experiment and reduce the compensator gain. No compensator gain should be greater than 5.
- Save the experiment

### 6.4 Post-Laboratory Exercises

(i) Plot the acquired data of both the Bode design method and the root locus method.

(ii) Compare those plots with the plots you obtained in the pre-laboratory assignment.

(iii) What can you add to the Bode design method presented in section 5.3 in order to eliminate the steady-state error in practice? **Hint:** refer to section 5.1.3.

**Note:** The Bode design method was presented here in its most basic and concise form with the intent of giving the student an insight into frequency-domain design; this technique could be pursued to a greater extent and give results similar to other classical design techniques.

(iv) Explain why, in the prelab, the system response remains relatively slow despite the high gain that you obtained from the root locus design procedure and that you used to reduce steady-state error.
CHAPTER 7

LEAD COMPENSATION FOR POSITION CONTROL OF A DC SERVO

The objective of this chapter is to utilize computer-aided design tools with techniques from Bode and root locus design methods, to design and subsequently implement lead compensation in a unity feedback configuration in order to perform position control on the SRV-02 DC servomotor.

7.1 Background

7.1.1 Overview

The general unity feedback configuration to be investigated is shown in Figure 7.1.

Figure 7.1: Cascade Lead Compensation Closed-Loop System
Recall the transfer function for a general first order cascade compensation network:

\[
G_c(s) = \tilde{K}_c \frac{1 + \tau s}{1 + \alpha \tau s} = K_c \frac{s + a}{s + b}.
\]

The compensation embodied in \(G_c(s)\) for this Laboratory is that of a lead compensator. A lead compensator has a high pass filter characteristic; that is, the zero \(a\) is closer than the pole \(b\) to the \(j\omega\)-axis (\(|b| > |a|\), or \(\alpha < 1\)). If the effect of the lead compensator's pole is negligible, that is, \(|b| \gg |a|\), and the zero of the lead compensator is near zero, then for frequencies \(\omega \ll |b|\), the lead compensator approximates a pure differentiator:

\[
\lim_{a \to 0} G_c(s) \approx \frac{K_c}{b}s
\]

The effects of lead compensation can be summarized as follows:

(i) increase in system bandwidth

(ii) decrease in error constant

(iii) improvement in gain or phase margin

(iv) faster system response

There are many methods and “rule-of-thumb” guidelines for placement of the pole and zero of the compensator, some of which we will address in the sections to follow. In general, lead compensation can have the effect of increasing the phase margin for a given system. To do this, some phase must be added to the point where the phase margin is defined (where \(\omega = \omega_c\), the “gain crossover frequency” where \(20 \log_{10} |G_c(s)G(s)| = 0\)), without adding any gain (which would subsequently change \(\omega_c\)). If the zero is made too
large (in magnitude), then the phase margin will not be significantly improved. If the zero is made too small, then the additional gain may make the closed-loop system unstable.

7.1.2 Bode Design

The log magnitude and phase angle expressions for $G_c(s)$ with $\tilde{K}_c = 1$ are:

$$20 \log_{10}|G_c(j\omega)| = 20 \log_{10}|1 + j\omega\tau| - 20 \log_{10}|1 + j\omega\alpha\tau|$$

$$\angle G_c(j\omega) = \angle(1 + j\omega\tau) - \angle(1 + j\omega\alpha\tau)$$

![Figure 7.2: Magnitude and Phase vs $\omega\tau$ for a Lead Compensator](image)

Figure 7.2: Magnitude and Phase vs $\omega\tau$ for a Lead Compensator
Figure 7.2 shows a typical family of frequency response curves for a lead compensator as a function of $\alpha$ with $K_c = 1$. Inspection of this figure reveals that the lead network is basically a high pass filter. Note also that a “lead angle” is introduced in the frequency range from $\omega = 1/\tau$ to $\omega = 1/(\alpha \tau)$. Thus, a lead network, because of its characteristic lead-angle, can be used to increase the bandwidth of a system. Observe that as $\alpha \to 0$, the lead compensator approximates a pure differentiator.

Design using the Bode log magnitude and phase plots is carried out by adjusting the phase margin and the phase margin frequency (also known as the gain crossover frequency $\omega_c$). To see this, assume the gain associated with $G(s)$ has been adjusted to yield a desired peak resonance magnitude ($M_r$). The inherent characteristic of the lead compensator is that it introduces a positive angle over a relatively narrow bandwidth; thus, increasing $\omega_c$ will also increase $\omega_r$. By properly selecting $\tau$, which in turn selects the pole and zero location for a given $\alpha$, the gain crossover frequency can be increased. Comparing the compensator angle curve (Figure 7.2) with the angle curve of the original system can lead to a selection of $\tau$. That is, the location of the $G_c(s)$ angle curve must be such as to produce the specified (or, desired) phase margin at the highest possible frequency (thereby determining $\tau$). The gain of $G_c(s)G(s)$ must be increased so that the log magnitude curve has a value of 0 dB at the “new” gain crossover frequency. For a given $\alpha$, the further to the right the compensator curves are placed (see Figure 7.2), the smaller $\tau$ will be and the larger the gain of the compensated system.
Procedures such as the above will be used in this Laboratory to add phase to a system to increase the phase margin. A rule-of-thumb criterion, for Type-1 or higher systems, is to select $\tau$ equal to or slightly less than the largest time constant of $G(s)$. For a Type-0 system, the rule-of-thumb is to select $\tau$ equal to or slightly less than the second largest time constant of the original system.

7.1.3 Root Locus Design

Recall that by making $\alpha$ sufficiently small, the location of the pole is far to the left and has a small effect on the “important” part of the root locus. As a basic guideline, for a Type-0 system a good time response, with larger gain, is often obtained by selecting the compensator zero so that it cancels (or is close to) the first or second largest real pole of the original system (if there exist multiple real poles). For a Type-1 or higher system, placement of the compensator zero to cancel the largest real pole (excluding the pole at zero) often results in a good transient response. Other schemes are possible for using the root locus for design of lead compensation, all basically having to do with the phase contribution at the dominant closed loop poles. We omit discussion of such techniques here, since the procedure in this Laboratory combines several ideas for lead compensator design.
7.2 Laboratory Preparation

(i) Given the closed loop system shown in Figure 7.1 with the transfer function of the lead compensator, \( G_c(s) = \frac{1 + \tau s}{1 + \alpha \tau s} = K_c \frac{s + a}{s + b} \), perform the following:

(a) Show that the frequency \( \omega_m \) of maximum phase lead is given by:

\[
\omega_m = \frac{1}{\tau \sqrt{\alpha}} = \sqrt{ab}.
\]

(b) Bonus problem: given the expression for \( \omega_m \) proven in (a), and the maximum phase lead \( \phi_m = \angle G_c(j\omega_m) \) of the compensator, show that

\[
\sin \phi_m = \frac{1 - \alpha}{1 + \alpha} = \frac{b/a - 1}{b/a + 1}.
\]

(c) Observation of Figure 7.2 demonstrates that the gain increase in dB due to the lead compensator with \( \tilde{K}_c = 1 \) is

\[
20 \log_{10} |G_c(j\omega_m)| = -10 \log_{10} \alpha.
\]

Prove this.

(ii) Given the closed loop system shown in Figure 7.1 with the transfer function of the plant, \( G(s) = \frac{1}{s(0.0026s + 0.1081)} \), and given the transfer function of the compensator, \( G_c(s) = \frac{s/a + 1}{s/b + 1} \), design a lead compensator using Bode techniques, according to the following design objective.
**Bode Design Objective:** Design a lead compensator that will increase “robustness” of the original plant to phase uncertainty by increasing the phase margin.

(a) Produce the open loop Bode plot of the uncompensated system, that is, $G_c(s) = 1$, and find the gain margin, $GM_{\text{orig}}$ and the phase margin, $PM_{\text{orig}}$.

(b) A compensator is to be designed such that the open loop system has a desired phase margin, $PM_{\text{des}}$, of 90°. Define the phase margin frequency as $\omega_{\text{pm}}$. The goal is to have the compensator $G_c(s)$ contribute a phase lead of $\phi_m$ at approximately $\omega_{\text{pm}}$ in order to give the closed loop system the desired phase margin of $PM_{\text{des}} = 90°$. Allowing for an extra 5° of phase lead from the compensator, determine the maximum phase lead of the compensator:

\[
\phi_m = PM_{\text{des}} - PM_{\text{orig}} + 5°
\]

(d) Using the result given in (i),(b) above, solve for $\alpha$.

(e) Because the lead compensator contributes a maximum phase of $\phi_m$ at $\omega_{\text{pm}}$, it is necessary that the compensated system have a 0 dB crossing at $\omega_m$. Since $G_c(j\omega)$ contributes a gain of $-10\log_{10} \alpha$ dB at $\omega_m$, as was shown in (i),(c), choose $\omega_m$ to be the frequency where the uncompensated system has a gain equal to $10\log_{10} \alpha$ dB. Solve for $a$ and $b$.

(f) Obtain an open loop Bode plot for the compensated system and verify the new phase margin.
(g) Obtain a closed loop step response for the system shown in Figure 7.1 using the lead compensator \( G_c(s) \) just designed.

(h) Let \( G_c(s) = 1 \) and obtain the closed loop step response.

(iii) Given the closed loop system shown in Figure 7.1 with the transfer function of the plant, \( G(s) = \frac{1}{s(0.0026s + 0.1081)} \), and given the transfer function of the compensator, \( G_c(s) = \tilde{K}_c \frac{s/a + 1}{s/b + 1} \), design a lead compensator using root locus techniques according to the following design objective.

**Root Locus Design Objective**: Design a lead compensator that “speeds up” the time response of the original plant and simultaneously shapes the time response of the original plant.

(a) “Speed up” the time response of the original plant by shifting the root locus farther to the left. The best approach for meeting this objective is to cancel the “slowest” pole of the open loop system using the lead compensator zero, \( a \). Select \( \alpha = 0.7 \). Solve for \( a \) and \( b \).

(b) Plot the root locus for the compensated system, that is, with

\[
G_c(s) = \tilde{K}_c \frac{s/a + 1}{s/b + 1}
\]

Shape the time response of the original plant by finding the value of \( \tilde{K}_c \) such that the dominant poles have a damping ratio of
$\zeta = 1/\sqrt{2}$. Plot the resultant closed loop pole locations for your value of $\tilde{K}_c$ on the root locus.

(c) Obtain a closed loop step response for the system shown in Figure 7.1 using the lead compensator $G_c(s)$ just designed.

7.3 Laboratory Procedure

7.3.1 Connections

Since we are performing (angular) position control on the servomotor, we will need to use the encoder for angular position sensing. Also, remember that we are using the motor in low gear ratio, with an inertia added on the top. Use the following procedure:

- Connect the UPM “To Load” port to the motor input.
- Connect the UPM “From D/A” port to analog output 0.
- Connect the encoder port of the motor (see Fig. 4.3) to the ENC1 on the interface board.

7.3.2 Simulink Diagram

- Open Matlab and Simulink and create a new model called Lab6.mdl. Set the simulation parameters to a fixed step size of 1 ms and a simulation time of 30 seconds.
- The model should appear as in Figure 7.3.
- The unit step input should start at time $t = 10$.

- The Rate Limiter block reduces the slope of the step input so as to avoid a very high frequency component input to the motor, which could potentially cause damage. Set the slew rates to $\pm 1000$.

- The Saturation block limits the control input so as to avoid a high voltage input to the motor, which could also cause damage. Set the saturation limits to $\pm 5$ V.

- The “Enc position” block allows us to acquire the motor shaft position from the encoder. Select Encoder 1, and include an “Encoder Master Setup” block that is necessary for the latter block.

Note that the output of the encoder is in cycles, and we need to convert it to degrees in order to compare it with the setpoint and produce an error signal. We know that the encoder has a counter that outputs 4 counts/cycle and we determine empirically that a full revolution ($360^\circ$)
yields 4096 counts ($2^{12}$). Therefore, the gain to go from cycles to degrees is $360 \times 4 / 4096$, and it should be included in the Simulink diagram.

- The Lead Compensator block is the transfer function of the compensator designed in the pre-laboratory.

7.3.3 ControlDesk and Results

- Open dSPACE ControlDesk and create a new experiment called Lab6, then compile the Simulink model using the compensator obtained in the pre-laboratory from the Bode design method.
- Create a new layout and add it to the experiment. Include a plot of the input and output.
- Acquire the data from the plot to a *.mat file.
- Change the compensator transfer function and use the one obtained from the root locus method. You might need to tune the compensator gain in order to get the best response.
  - Save the experiment

7.4 Post-Laboratory Exercises

(i) Plot the acquired data of both the Bode design method and the root locus method. Show steady state error, rise time, settling time and overshoot on each plot.

(ii) Compare those values with those of the plots you obtained in the pre-laboratory assignment.
(iii) In the root locus design procedure, we picked \( \alpha = 0.7 \); what discrepancies between simulation and practical implementation would you have expected had we selected a smaller value, for instance \( \alpha = 0.1 \)? In other terms, would you expect to obtain the same system responses on the actual motor as the ones obtained during the design procedure? Explain.
CHAPTER 8

TUNING A PID CONTROLLER

The objective of this chapter is to investigate the Proportional-Integral-Derivative (PID) type of control law. Model-based tuning rules will be investigated for designing a PID controller to perform position control of the SRV-02 DC motor with modeled and unmodeled dynamics.

8.1 Background

8.1.1 Overview

Figure 8.1: Cascade PID Compensation Closed-Loop System
The previous chapters have investigated the use of gain, lag, and lead compensation in cascade connection with the plant for a unity feedback configuration. This Laboratory combines the effects of all three types of compensation into one control law which has proportional gain, integral action (lag), and derivative action (lead). The compensation embodied in $G_c(s)$ of Figure 8.1 for this Laboratory is that of a PID compensator.

PID control is the most commonly used compensation scheme in industrial control systems. From a practical point of view, the PID formulation is appealing because of its simplicity and familiarity; it has, in fact, become somewhat of a standard in many application areas, particularly process control systems. In a unity feedback configuration, the PID controller is the summation of the system error, the integral of the system error, and the derivative of the system error with each error quantity being scaled by its corresponding proportional, integral, or derivative gain. Because of the relation of proportional, integral and derivative gains to actual physical quantities, engineering intuition plays a much larger role in system analysis and design as compared to some more complicated control design strategies. The ability to tune the PID controller gains allows for increased flexibility in the compensator design. The desired compensator may be composed of any combination of the proportional, integral or derivative gain terms, or the compensator may be composed of only one of these individual gain terms. For example, only a proportional gain compensator (P) may be desired, or a proportional plus integral (PI) controller, or a proportional plus derivative (PD) controller. Finally, a large advantage of the PID tuning strategy is that little a priori knowledge of the plant is
required. That is, typically a formal dynamical model of the plant is not necessary in order to design and implement a PID controller.

The transfer function $G_c(s)$ of the PID controller is usually given by

$$G_c(s) = K_p \left( 1 + \frac{1}{\tau_i s} + \tau_d s \right)$$

where $K_p$ is the proportional gain, $\tau_i$ is the time constant associated with the integral action, $\tau_d$ is the time constant associated with the derivative action. For the unity feedback configuration of Figure 8.1, the system error, $e(t)$, which is the error between the reference input and the plant output, is the input to the PID controller, and the output of the PID controller takes the form

$$u(t) = K_p \left[ e(t) + \frac{1}{\tau_i} \int_{-\infty}^{t} e(t) dt + \tau_d \frac{de(t)}{dt} \right].$$

A convenient form for representing these dynamics in a transfer function is

$$\frac{U(s)}{E(s)} = G_c(s) = K_p + \frac{K_i}{s} + K_d s,$$

where $K_p$ is the proportional gain, $K_i$ is the integral gain, and $K_d$ is the derivative gain. The gains $K_p$, $K_i$, and $K_d$ are referred to as the PID controller parameters, since they essentially determine the form of the compensator.

### 8.1.2 Tuning Rules

If a mathematical model of the plant is available, it is always possible to apply various classical control design techniques, such as root locus or Bode plots, in order to
determine controller parameters which meet desired design specifications. On the other hand, if the plant is so complex that an accurate mathematical model cannot be derived easily, these classical control design techniques become difficult to apply in designing a general linear controller. The alternative is to apply experimentally proven techniques for choosing the controller parameters, resulting in at least a starting point from which the controller can then be tuned.

In the early 1940's, Ziegler and Nichols published work on the topic of tuning rules for determining PID controller gains based on observable behaviors of the system, such as the transient response of a given plant. The motivation was primarily for application to systems where the mathematical model of the plant is not known precisely, so that tuning of the PID parameters can be performed on site via experimental analysis with the plant. These tuning rules are equally applicable to systems with known models.

In this Laboratory, we will use classical control design techniques for two reasons: first, we have a valid model for the servomotor and second, the open-loop step response of the servomotor does not exhibit the behavior necessary for applying the Ziegler-Nichols tuning rules.

8.2 Laboratory Preparation: PID Control Design

(i) Write the transfer function of the PID controller in the following form:

\[
\frac{V_{in}(s)}{E(s)} = K \left( \frac{s^2 + 2\zeta \omega_n s + \omega_n^2}{s} \right),
\]

where the quantities \( K \), \( 2\zeta \omega_n \) and \( \omega_n^2 \) are functions of \( K_p \), \( K_d \) and \( K_i \).
(ii) Determine the ratios $\frac{K_p}{K_d}$ and $\frac{K_i}{K_d}$ so that the zeros of the PID controller are on the $\zeta = \frac{1}{\sqrt{2}}$ line with $\omega_n = 50$ rad/sec. What are the values of those zeros? What are the poles of the compensator?

(iii) In Matlab, create the transfer function (for angular position) of the motor:

$$G(s) = \frac{1}{s(0.0026s + 0.1081)}.$$

Using the ‘rltool’ command, import the system and add the compensator zeros and poles found in part (ii). Make sure you change the compensator format to “zero/pole/gain” as shown in Figure 8.1.

![SISO Tool Preferences](image)

**Figure 8.2: Compensator Format**
(iv) Find the value of $K$ on the root locus that will place the closed loop poles relatively close to the open loop zeros (which are nothing but the zeros of the PID controller). A good proximity would be to place the closed loop poles within a circle of radius 5 of the open loop zeros. Specify the three closed loop pole locations for this value of $K$. Compute the values of $K_p$, $K_i$ and $K_d$.

(v) Plot the closed loop step response and show on the same plot the values of overshoot, rise time, settling time and steady state.

(vi) The next step consists of fine tuning the PID gains in order to obtain better transient characteristics than the initial design. It is necessary to note that in simulation one can virtually achieve any desired specifications, however, in implementation the control engineer is faced by practical issues such as physical limitations, actuator saturation, and the like. High controller gains result in unrealistically large control inputs which exceed the operating range of the actuators. Hence the introduction of a new constraint in control design, namely the control effort. In this Lab, the input voltage to motor has a range of $\pm 5V$. Therefore, the magnitude of the control input should not exceed 5. In order to monitor the control input you will need to build the Simulink diagram of Figure 8.3.
- Enter the values of $K_p$, $K_i$ and $K_d$ that you obtained in question 4 in the PID block.
- Change the ‘save format’ of the Workspace blocks to ‘Array’.
- Set the unit step time to 0.
- Set the simulation time to 1 sec.
- Run the simulation and then, in Matlab, plot the variable $u$ versus $t$. Note the initial value of $u$. Is it in the range of $\pm 5V$?
- Now start reducing the proportional gain until the control input remains in the valid range. Once you meet this control energy constraint, tune the PID gains in an ad hoc fashion to find the best trade-off between overshoot and settling time (less than 2% overshoot and less than 0.1 sec settling time).
- Tuning hints: Start with a proportional gain that is less than 5 and keep in mind that in most applications the derivative gain is 1 or 2 orders of
magnitude smaller than the proportional gain, and the integral gain is smaller than the proportional gain.

- Produce a step response plot and a control input plot for at least three iterations including the final iteration. Record the values of the PID gains, overshoot and 2% settling time on each plot.

8.3 Laboratory Procedure

8.3.1 Derivative Filtering

Implementing the derivative term \( K_d \dot{e} \) poses the following problem:

The error signal \( e \) often contains high frequency components due to sharp transitions in step inputs or to measurement noise from sensors such as tachometers and encoders. In both cases, the error signal exhibits sharp corners which translate into a very large derivative term since \( K_d \dot{e} = K_d (\dot{r} - \dot{y}) \). In this equation, if the reference input \( r \) has a sharp corner (step) or if the measured signal \( y \) has high frequency noise, the derivative will be very large and will yield a very large control input from the PID controller. In order to solve this problem, we first include a rate limiter after the step input to reduce the infinite slope. Next, we assume that the reference input remains constant and we write the derivative equation as follows:

\[
K_d \dot{e} = -K_d \dot{y}.
\]

Hence, we have eliminated the reference input from the PID equation and kept the derivative of the output signal that will allow the PID controller to maintain its predictive
capabilities. The solution to the problem of noise in the output signal $y$ consists of passing the derivative term through a low-pass filter of the form:

$$\text{LPF} \equiv \frac{a}{s + a}$$

The overall derivative component of the PID controller becomes then

$$K_d s \cdot \frac{a}{s + a} = \frac{K_d s}{(1/a)s + 1}.$$

A value of $a = 200$ is a reasonable choice for the low-pass filter with the measuring equipment in this Lab.

8.3.2 Simulink Diagram

- Open Matlab and Simulink and create a new model called lab7.mdl. Set the simulation parameters to a fixed step size of 1 ms and a simulation time of 30 seconds.
- The model should appear as shown in Figure 8.4.

Figure 8.4: Simulink Diagram
- The unit step input should start at time $t = 10$.
- Set the slew rates of the rate limiter to $\pm 1000$.
- Set the saturation limits to $\pm 5$ V.
- Plug in values for $K_p$, $K_i$ and $K_d$. Remember the tuning hints in the Laboratory procedure.

8.3.3 ControlDesk and Results

- Open dSPACE ControlDesk and create a new experiment called Lab7, then compile the Simulink model.
- Create a new layout and add it to the experiment. Include a plot of the input and output.
- Perform several iterations until you reach the best possible transient characteristics (less than 2% overshoot and less than 0.5 sec settling time). Include plots for three iterations.

8.4 Post-Laboratory Exercises

(i) Plot the acquired data in Matlab plots.
(ii) Include on each plot the values of the PID gains, overshoot and 2% settling time.
(iii) Compare the experimental results with those obtained in the pre-lab.
(iv) Explain why it is practically impossible to implement a PID without a derivative filter. *Hint:* refer to the transfer function of the PID controller and analyze it using basic signal and systems theory.
CHAPTER 9

PID CONTROL FOR A FLEXIBLE JOINT

The objective of this chapter is to design a PID controller for endpoint position control in the face of flexibility effects for a flexible joint mounted on the SRV-02 DC servomotor. For this we will first present the model of the joint system and then tune the PID gains to come up with the best position tracking while minimizing endpoint oscillation.

9.1 Model

The joint of a robotic manipulator is an important concern in modeling and control for industrial and space applications. In the early days of robotics, robotic arm joints were rigid and they posed two problems: first, they were unsafe for the products on which they were acting, especially when quick and precise movement was required; second, they were unsafe for the robots themselves in case of accidental collisions with external objects. Thus, often flexibility is “designed into” the mechanism, such as in the flexible joint studied in this chapter, and other times flexibility might be a consequence of demands on structure weight, such as for the flexible link studied in the next chapter.
Modeling and controlling these flexible manipulators is much more difficult than modeling and controlling rigid structures. To simplify matters somewhat, we will assume a linearized model for the purpose of this laboratory. The flexible joint shown in Figure 9.1 is a fourth-order system. There are nine different ways the springs can be attached and there are three different springs each with a different spring constant in order to create various flexibility effects. The default connection that we will use in this Laboratory is the one shown in Figure 9.2, and the constants used in this Laboratory correspond to this particular connection. A mass can be attached at the tip of the joint to study the effects of disturbances and plant parameter variations.

Figure 9.1: Flexible Joint [4]
Let us define some useful variables before we write the state variable model:

- $\theta$ is the motor shaft position, measured using an encoder.
- $\alpha$ is the angular deflection of the joint’s arm, i.e. the difference between the angular position of the tip and the angular position of the motor’s output shaft. It is measured using an encoder.
- $K_{\text{stiff}}$ is a linear approximation of the joint stiffness. For the default connection (spring type 2 and anchors A-3, see Figure 9.2) the value of $K_{\text{stiff}}$ is $1.6108$ Nm/rad.
- $J_{\text{hub}}$ is the total inertia of the motor, with value $0.0021$ Kg.m$^2$.
- $J_{\text{load}}$ is the inertia of the arm, with value $0.0059$ Kg.m$^2$.
- $R_m$ is the armature resistance of the motor, with value $2.6$ $\Omega$.
- $K_m$ is one of the motor torque constants, with value $0.00767$ V/rad/sec.
- $K_g$ is the gear ratio of the motor. In the high gear ratio configuration, $K_g = 70$.

The fourth-order linearized differential equation representing the dynamics of the plant can be written in a first-order matrix differential equation, representing the state variable description of the system. The state variable representation is a compact notation combining four coupled first-order differential equations each describing the dynamics of one state of the plant. We define the state vector, $x$, for the flexible joint as follows:

$$
x = \begin{bmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix}.
$$
Let $y$ be the output of the system and $u$ the voltage input to the motor ($V_{in}$). The state variable model is of the form:

\[
\dot{x} = Ax + Bu \\
y = Cx
\]
The derivation of the model equations from physics involves mathematical modeling and linearization techniques that are beyond the scope of this Lab. The following matrices will comprise the design model for the system:

\[
A = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & -K \frac{K^2}{J_{\text{hub}}} & 1 \\
0 & -K \frac{K^2}{J_{\text{hub}}} & 0 & 0 \\
0 & 0 & K \frac{K^2}{R_{m}J_{\text{hub}}} & 0
\end{bmatrix},
B = \begin{bmatrix}
0 \\
0 \\
-\frac{K^2}{R_{m}J_{\text{hub}}} \\
-\frac{K^2}{R_{m}J_{\text{hub}}}
\end{bmatrix},
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}.
\]

9.2 Laboratory Preparation

(i) Write the four first-order differential equations as a function of the states and of the physical constants. Do not substitute the numerical values for the constants.

(ii) Looking at the matrix C, what are the outputs that we are interested in? Do we need to measure the third output? Why?

(iii) In Matlab, enter the values of the matrices A, B and C by substituting the numerical values of the constants. Using the command “ss2tf” write the transfer functions from the input to each output. Make sure to specify the input and output for each transfer function and ignore any terms that are extremely small compared to the other terms.

(iv) Construct the Simulink model shown in Figure 9.3 and plug-in the necessary transfer functions from part (iii):

- Change the ‘save format’ of the Workspace blocks to ‘Array’.
- Set the unit step time to 0.
- Set the simulation time to 3 sec.
- In the PID block set the integral gain to 0. Start with a logical choice for \( K_p \) (one that takes into account the input voltage constraint of the motor, that is \( \pm 5\text{V} \)). Remember that a rule of thumb is that the derivative gain is 1 or 2 orders of magnitude smaller than the proportional gain.

![Simulink Design Model](image)

**Figure 9.3: Simulink Design Model**

- Tune the gains until you obtain a step response for the total angular position \((\theta + \alpha)\) that has less than 2% overshoot and less than 0.6 sec settling time (2% criterion). Plot the step response for \( \theta + \alpha \) and indicate the values of the gains on the plot, as well as the percent overshoot and the settling time. Your plot should look like Figure 9.4. Also include plots of the response of the tip deflection \( \alpha \) and of the motor input.
9.3 Laboratory Procedure

9.3.1 Connections

Connect the motor in the position control configuration then connect the encoder port located on the back of the joint to the second encoder input on the DS1104 interface board.
9.3.2 Simulink Diagram and ControlDesk

- Open Matlab and Simulink and create a new model called Lab8.mdl. Set the simulation parameters to a fixed step size of 1 ms and a simulation time of 30 seconds. Turn off block reduction.

- The model should appear as in Figure 9.5.

Let the unit step input should start at time \( t = 10 \).

- Add a safety stop condition to the Simulink diagram.

*Hint: use the blocks “relational operator”, “stop simulation”, and “constant”.*

- Start with the values of \( K_p \) and \( K_d \) found in the pre-laboratory. Can you eliminate steady-state error with these values? Add integral control if needed in order to achieve a smaller steady-state error.
• Perform several iterations until you reach a steady-steady error of less than 0.5% and a peak overshoot of less than 2%. The settling time should be less than 0.6 seconds.

• Save the plot acquired in your final iteration and record the corresponding values of the PID gains.

9.4 Post-Laboratory Questions

(i) Plot the acquired data and show the gain values, the steady-state error and the percent overshoot on the plot.

(ii) Why was PD control alone (without the integral term) enough to obtain a satisfactory response in the preliminary design?

(iii) Based on your answer to the previous question, why would you need to add integral control during implementation?

(iv) The design method proposed in the pre-laboratory allows you to control only the combined angular position, i.e. \( \theta + \alpha \). Can you think of a different way to approach this control problem?

*Hint:* Think in terms of the four states of the system.
CHAPTER 10

PID CONTROL FOR A FLEXIBLE LINK

The objective of this chapter is to position the tip of a rotary flexible link as quickly as possible with minimal vibrations using a PID controller. For this, we will first develop the model of the (motor + link) system and then tune the PID gains to come up with the best position tracking while minimizing endpoint oscillation.

10.1 Model

In addition to the motivations presented in section 9.1, another important factor has fueled the advances in controlling robots with flexible links: the need for large and lightweight structures in space-based robotic applications. Because payloads into space are very costly, large robot arms must be lightweight. Thus, a direct consequence is the inherent flexibility of such devices, which must be accounted for.

The flexible link shown in Figure 10.1 is a fourth-order system. The motor shaft position, $\theta$, is measured using an encoder. The angular deflection at the tip, $\alpha$, is measured by a strain gage analog sensor that is calibrated to output one volt per inch of
The output that we are trying to control is the total deflection of the link, i.e. \( \theta + \alpha \). The state variable model has the following variables:

\[
\begin{bmatrix}
\theta \\
\alpha \\
\dot{\theta} \\
\dot{\alpha}
\end{bmatrix}
\]

\[
u = V_{in}, \quad x =
\begin{bmatrix}
\theta \\
\alpha \\
\dot{\theta} \\
\dot{\alpha}
\end{bmatrix}
\]

Figure 10.1: Rotary Flexible Link

We write the state variable model as:

\[
\dot{x} = Ax + Bu
\]

\[
y = Cx
\]
where \( A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 2165 & -53.6 & 0 \\ 0 & -2797 & 53.6 & 0 \end{bmatrix} \), \( B = \begin{bmatrix} 0 \\ 0 \\ 99 \\ -99 \end{bmatrix} \) and \( C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \).

10.2 Laboratory Preparation

(i) Looking at the matrix \( C \), what are the outputs that we are interested in? Do we need to measure the third output? Why?

(ii) In Matlab, enter the values of the matrices \( A \), \( B \) and \( C \) and use the command “ss2tf” to write the transfer function from the input to each output. Make sure to specify the input and output for each transfer function and ignore any terms that are extremely small compared to the other terms.

(iii) Construct the Simulink model of Figure 10.2 where you will plug-in the necessary transfer functions from part (ii):

![Simulink Design Model](image)

Figure 10.2: Simulink Design Model
- Change the ‘save format’ of the Workspace blocks to ‘Array’.
- Set the unit step time to 0.
- Set the simulation time to 3 sec.
- In the PID block set the integral gain to 0. Start with a logical choice for \( K_p \) (one that takes into account to input voltage constraint of the motor, that is ±5V). Remember that a rule of thumb is that the derivative gain is one or two orders of magnitude smaller than the proportional gain.
- Tune the gains until you obtain a step response for the total angular position \((\theta+\alpha)\) that has zero steady-state error, less than 0.6 sec settling time (2% criterion) and no overshoot. Plot the step response for \(\theta+\alpha\) and indicate on the plot the values of the gains as well as the settling time.
- Also include plots of the response of the tip deflection, \(\alpha\), and of the motor input.

10.3 Laboratory Procedure

10.3.1 Connections

Connect the motor in the position control configuration then connect the strain gauge port located on the back of the link via the UPM to the fifth analog input on the DS1104 interface board.
10.3.2 Simulink Diagram and ControlDesk

- Open Matlab and Simulink and create a new model called Lab9.mdl. Set the simulation parameters to a fixed step size of 1 ms and a simulation time of 30 seconds. Turn off block reduction.

- The model should appear as in Figure 10.3.

![Simulink Diagram](image)

Figure 10.3: Simulink Diagram

- Add a safety stop condition to the Simulink diagram.

\textit{Hint}: use the blocks “relational operator”, “stop simulation”, and “constant”.

- You might need to add a constant to adjust for the offset in the measurement of ‘alpha’.

- Start with the values of $K_p$, $K_i$ and $K_d$ found in the pre-laboratory and tune the gains until you achieve a steady-steady error of less than 1%, an overshoot of less than 5% and a settling time of less than 0.5 sec.
Plot the variables ‘input’, ‘output’ and ‘alpha’.

Save the plot acquired in your final iteration and record the corresponding values of the PID gains.

10.4 Post-Laboratory Questions

(i) Plot the acquired data and show the gain values, the steady-state error, the percent overshoot and the settling time on the plot.

(ii) Explain how we obtained the strain gauge gain.

(The length of the link is $L = 0.45$ m).

(iii) Explain why we have a minus sign for the $K_d$ path in the final summing junction.

(iv) Explain why the response is not as smooth as that of the flexible joint in the previous Laboratory.

(v) Draw comparison and contrast between the results of this chapter and the previous chapter.
In this experiment, inspired from [6], we will use full state feedback and design a linear quadratic regulator (LQR) to position the tip of a rotary flexible joint as quickly as possible with minimal vibrations. “Full state feedback” is achieved by using encoders to measure the position of the DC servo output shaft and the deflection of the joint; a derivative filter produces the angular rate of the joint while a tachometer provides a measurement of the angular speed of the motor. At the end of the chapter, we design observers to substitute for derivative filtering and estimate angular speeds.

11.1 Model

The importance of flexible joints was highlighted in section 9.1. The state variable model of the flexible joint is as follows:
\[
A = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & \frac{K_{\text{stiff}}}{J_{\text{hub}}} & 0 \\
0 & -\frac{K_{\text{stiff}}}{J_{\text{hub}}} \left( J_{\text{load}} + J_{\text{hub}} \right) & \frac{-K_m^2K_g^2}{R_mJ_{\text{hub}}} & 0 \\
0 & \frac{J_{\text{hub}}}{J_{\text{load}}} & \frac{K_m^2K_g^2}{R_mJ_{\text{hub}}} & 0 \\
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
0 \\
0 \\
\frac{K_mK_g}{R_mJ_{\text{hub}}} \\
\frac{R_mJ_{\text{hub}}}{R_mJ_{\text{hub}}} \\
\end{bmatrix}.
\]

Assuming that we can produce all states, \( C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}. \)

With the parameters provided in [4], Matlab gives:

\[
A = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 767.05 & -52.8 & 0 \\
0 & -1040.07 & 52.8 & 0 \\
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
0 \\
98.33 \\
-98.33 \\
\end{bmatrix}.
\]

Using Matlab command ‘ctrb’, we find the controllability matrix of the system and we compute its rank:

\[
>> \text{rank(ctrb(A,B))}
\]

The controllability matrix has full rank, hence the system is controllable.

11.2 LQR Design

The Simulink model in Figure 11.1 along with the Matlab command ‘lqr’ allow to test various designs.
Starting with the initial conditions \[
\begin{bmatrix}
\pi \\
0 \\
0 \\
0
\end{bmatrix}
\] and choosing \( R=1 \) and \( Q = 
\begin{bmatrix}
90 & 0 & 0 & 0 \\
0 & 90 & 0 & 0 \\
0 & 0 & 90 & 0 \\
0 & 0 & 0 & 90
\end{bmatrix}
\]
, we obtain the optimal response that drives the joint to the zero position as fast as possible while maintaining the input voltage in the range of \( \pm 5V \). The feedback gain resulting from this LQR design is:

\[
K = \begin{bmatrix}
9.49 & -138.37 & 9.5 & -3.51
\end{bmatrix}
\]

Figures 11.2 and 11.3 show the responses of the states and the control input respectively.
Figure 11.2: LQR Simulation Results
11.3 Observer Design

11.3.1 Observability and Observer Equations

Using the Matlab function ‘obsv’, we find the observability matrix for different combinations of sensors and we compute its rank. We note that the system is observable for any combination of sensors, in particular, whenever we measure the output shaft position only. In this case, the output matrix is $C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ and the observer equations are:
\[
\dot{x} = A\dot{x} + Bu + L(y - \dot{y})
\]

\[
\hat{y} = C_i \hat{x}
\]

The error dynamics are:

\[
\dot{e} = \dot{x} - \dot{\hat{x}} = (Ax + Bu) - (A\dot{x} + Bu + L(y - \dot{y})) = A(x - \dot{x}) - L(C_i x - C_i \hat{x}) = (A - LC_i)(x - \dot{x}) = (A - LC_i)e
\]

11.3.2 Observer Gain Selection Via Pole Placement

From the properties of linear systems, when the pair \((A,C)\) is observable, the pair \((A^T,C^T)\) is controllable. On the other hand, \(\text{eig}(A^T-C^TK) = \text{eig}(A^T-C^TK)^T = \text{eig}(A-KTC)\).

This means that in order to design an observer for the pair \((A,C)\), it suffices to design a controller gain for the pair \((A^T,C^T)\) and then transpose that gain to get the observer gain vector.

11.3.3 Reduced-Order Observer

Since we can measure \(\theta, \alpha\) and \(\dot{\theta}\), let us design a reduced order observer to estimate only \(\dot{\alpha}\). First, regroup the states in the state vector \(x\) to get the state vector \(s\):

\[
x = \begin{bmatrix} \theta & \alpha & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T.
\]

\[
s = \begin{bmatrix} \dot{\alpha} & \theta & \alpha & \dot{\theta} \end{bmatrix}^T.
\]

Therefore,

\[
s = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x = T_i x.
\]

134
Note that the new output matrix is

\[
C = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}.
\]

Now use the following transformation:

\[
v = \begin{bmatrix}
s_1 \\
y \\
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
s_1 \\
s_2 \\
\end{bmatrix} = T_2s,
\]

where \( s_1 = \dot{\alpha} \) and \( s_2 = [\theta \quad \alpha \quad \dot{\theta}]^T \).

Therefore,

\[
\dot{v} = (T_2 T_1 A T_1^{-1} T_2^{-1})v + T_2 T_1 Bu;
\]

so that with appropriate parameter values we have

\[
\dot{v} = \begin{bmatrix}
0 & 0 & -1040.07 & 52.8 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 0 & 767.05 & -52.8 \\
\end{bmatrix}v + \begin{bmatrix}
-98.33 \\
0 \\
0 \\
98.33 \\
\end{bmatrix}u = \Phi v + \Gamma u.
\]

Let \( s_1^* = s_1 + Ly \). Then we have

\[
\dot{s}_1^* = (\Phi_{11} + L\Phi_{21})s_1^* + (\Phi_{12} + L\Phi_{22})y + (\Gamma_1 + L\Gamma_2)u.
\]

The error dynamics are:

\[
\dot{x}_1^* = (\Phi_{11} + L\Phi_{12})x_1^*.
\]
Using the same techniques, we design the gain $L$ and deduce the value of $s_1$ from the estimated $s_1^*$ to obtain an estimate for $\dot{\alpha}$. Figure 11.4 shows the observer convergence for all four states.

![Figure 11.4: Observer Response](image)

11.4 Implementation Without Observer

(i) To acquire the position of the motor shaft we use the encoder position (channel 1) and encoder master setup block. The velocity is acquired using the MUX ADC
block (channel 4) that reads from the tachometer. Both blocks are followed by the appropriate gains shown in Figure 11.5.

Figure 11.5: Motor Outputs

(ii) To acquire the angle $\alpha$, we use the encoder position block as well. However, to get the derivative of $\alpha$, we use derivative filter with transfer function $\frac{150s}{s+150}$.

The blocks along with the corresponding gains are shown in the figure below:

Figure 11.6: Flexible Joint Outputs
(iii) The blocks in part (i) were combined under one block called motor outputs. The blocks in part (ii) were combined under one block called joint outputs. The outputs from those two blocks are fed to a multiplexer to form the state vector that is multiplied by the state feedback gain and output on the DAC for actuation. Then a safety stop time block and a safety stop angle are added for fail-safe purposes. In addition to the usual blocks that are used in those two subsystems, the Absolute Value block is used in the safety stop angle block to ensure an angle limit in the positive and negative directions; this is represented in Fig 11.7.

![Figure 11.7: Safety Stops](image)

The reference signal is supplied by a repeating sequence block that outputs a square wave of amplitude 30 deg and frequency 0.05 Hz. The full model diagram is shown in Figure 11.8.
(iv) The values of the gain obtained in the design phase were adequate for an acceptable transient response but they left the system with much steady-state error. This is due to the low penalty that we assigned to the position state $\theta$.

Having in mind the undesired steady state error, we reduced the penalty on control (represented by the value of $R$) and increased the penalty on position error (represented by the first element of the $Q$ matrix). After several iterations, the best results, shown in Figure 11.9, were obtained with following tuned values:
The corresponding gain vector is: 

\[ K = \begin{bmatrix} 9.2195 & -0.5074 & 0.8903 & 0.2081 \end{bmatrix}. \]

Figure 11.9: LQR Response
11.5 Implementation With Observer

The model with the observer is the same model used previously, but with the addition of the observer block that takes the input voltage and the measured position as its inputs and computes a state vector estimate as its output. However, in this case, the state feedback comes from the output of the observer because we assume that not all the states are measured. Figure 11.10 shows the model with observer state feedback.

Figure 11.10: Simulink Diagram with Observer
The observer block diagram is given in Figure 11.11.

The matrices $A$ and $B$ are the same as in the design phase, because in principle, an observer should be a replica of the plant model, with the addition of output error feedback that will allow convergence given arbitrary initial conditions. The matrix $D$ is zero (no feed-forward). The matrix $C$ (here depicted as $C_1$) is just a row vector consisting of the motor position measurement: $C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$. Figure 11.12 shows the observer error system responses.
Figure 11.12: Observer Errors

The response (plant output) in Figure 11.13 has a steady-state error of less than 1% when compared to the response with no observer. This mismatch is caused by the time required for convergence of the observer, and any errors that exist in the observer response.
Figure 11.13: LQR Response with Observer

Figure 11.14 compares the responses with and without an observer. Note the delay and steady state error in the observer response.
Figure 11.14: LQR Response with and without Observer
CHAPTER 12

LQR CONTROL FOR A FLEXIBLE LINK

In this experiment, we will use full state feedback and design a linear quadratic regulator (LQR) to position the tip of a rotary flexible link as quickly as possible with minimal vibrations. “Full state feedback” is achieved by using a tachometer and an encoder to measure the angular velocity and the position of the DC servo output shaft respectively; a strain gage measures the deflection at the tip and a derivative filter produces the angular rate of the link angle.

12.1 Model

The importance of flexible links was highlighted in section 9.1. The following state variable equation represents a linearized model of the flexible link with the SRV-02 plant:

\[ \dot{x} = Ax + Bu \]

\[ y = Cx \]

where

\[ x = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^T \]
\( \theta \) is the angular position of the DC servo

\( \alpha \) is the angular deflection of the link

\( u \) is the input voltage \( V_{in} \)

\[
A = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1059.1 & -57.6 & 0 \\
0 & -1463.4 & 57.6 & 0
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
0 \\
107.2 \\
-107.2
\end{bmatrix} \quad \text{and} \quad C = [1 \ 1 \ 0 \ 0]
\]

The system is controllable ( \( \text{rank}(\text{ctrb}(A,B)) = 4 \) ).

12.2 Integral Control via State Augmentation

The flexible link system is subject to vibration disturbances. In order to reject those disturbances and achieve better steady-state tracking, we need to introduce integrators in the loop. We augment the state variable representation with the new state \( \theta_i \) that is the integral of \( \theta \). The state variable equation becomes:

\[
\dot{x} = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & \theta \\
0 & 0 & 0 & 1 & 0 & \alpha \\
0 & 1059.1 & -57.6 & 0 & 0 & \dot{\theta} \\
0 & -1463.4 & 57.6 & 0 & 0 & \dot{\alpha} \\
1 & 0 & 0 & 0 & 0 & \theta_i
\end{bmatrix}
+ \begin{bmatrix}
0 \\
0 \\
107.2 \\
-107.2 \\
0
\end{bmatrix} u
\]
12.3 LQR Design and Simulation

Figure 12.1 below shows the simulation diagram of the system.

![Simulation Model](image)

Using Matlab command ‘lqr’ with $R=2$ and $Q = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, we obtain the following gain vector: $K = [7.18 \ 0.3 \ 0.586 \ 0.166 \ 0.707]$
Starting with zero initial conditions, we plot the closed-loop response to a 30-degree step input (0.5236 rad) and the control input response; the result is shown in Figure 12.2, where the control variable ($V_{in}$) is shown in Figure 12.3.

Figure 12.2: State Response
12.4 Connections

- The DC Servo will be used in high gear ratio.
- The position of the DC Servo will be acquired through encoder #1
- The velocity of the DC Servo will be acquired through the MUX ADC #3 from the tachometer.
The link deflection will be acquired through the MUX ADC # 4 from the strain gage (the strain gage outputs one volt per inch of deflection and the link length is 48.26 cm, therefore, the strain gage gain is 0.052632)

12.5 Simulink Diagram and Results

Figure 12.4 below shows the Simulink diagram that will be compiled and downloaded on the dSPACE card.

![Simulink Diagram](image)

Figure 12.4: Simulink Diagram
After tuning the LQR gains we obtain the following results:

\[
R=1, \quad Q = \begin{bmatrix}
10 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.6
\end{bmatrix}
\quad \text{and} \quad K = \begin{bmatrix}
3.44 & 0.138 & 0.183 & 0.128 & 1.26
\end{bmatrix}.
\]

The response to a 30-degree step is plotted in Figure 12.5.

![Figure 12.5: LQR Response](image-url)
12.6 Discussion

The methods explored in chapters 11 and 12 could be used to develop a term project that would encourage students to improve their knowledge of state variable representation as well as learn the concepts of state-feedback and LQR control design. Since the topic of observers is somewhat advanced for undergraduate students, the instructor could explain the philosophy behind observers and provide the students with pre-designed observer blocks that they can insert in their Simulink diagrams in order to compare the results with and without observers.
CHAPTER 13

TWO-DEGREE-OF-FREEDOM HELICOPTER EXPERIMENT

The Quanser 2 DOF Helicopter (see Figure 13.1) is a nonlinear MIMO system. It has two measurable outputs: pitch and yaw. The two inputs to the plant are voltages applied respectively to the front motor driving the pitch propeller and to the back motor driving the yaw propeller. First, using a linearized model, we design a linear quadratic regulator (LQR) for pitch and yaw position control. Full-state feedback is achieved by using two encoders to measure the angles of pitch and yaw, and derivative filtering to produce pitch rate and yaw rate measurements. Next, we develop integral control by augmenting the state-space system matrix to include integral effects on both pitch and yaw outputs. Proper modeling and gain assignments will guarantee zero steady-state error in the face of step inputs and disturbances.
13.1 Model

The helicopter model is a 2-input 2-output nonlinear system. However, a simplified 4th order design model, provided by Quanser [4], is presented below:

\[
\begin{bmatrix}
\dot{p} \\
\dot{y} \\
\ddot{p} \\
\ddot{y}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
p \\
y \\
\dot{p} \\
\dot{y}
\end{bmatrix} + \begin{bmatrix}
0 & 0 \\
0 & 0 \\
\frac{-L_b K_{ff}}{2 J_{pp}} & \frac{-K_{tb}}{J_{pp}} \\
\frac{-K_{ff}}{J_{yy}} & \frac{L_b K_{fb}}{2 J_{yy}}
\end{bmatrix} \begin{bmatrix}
V_p \\
V_y \\
\dot{V}_p \\
\dot{V}_y
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
G_d \\
0
\end{bmatrix},
\]

where

- \( p \) is the pitch angle relative to the initial position (0 is the position when the helicopter is at rest).
- \( y \) is the yaw the angle relative to the initial position (positive is defined clockwise).
- \( V_p \) is the pitch motor voltage.
- $V_y$ is the yaw motor voltage.
- $G_d$ is a gravitational constant disturbance (It will be neutralized with integral control later).
- $L_b$ is the distance between the centers of the propellers.
- $J_{pp}$ and $J_{yy}$ are the moments of inertia of the body about the pitch and yaw axes.
- $K_{ff}$ is the front motor constant.
- $K_{fb}$ is the back motor constant.
- $K_{tb}$ and $K_{tf}$ are the motor coupling constants, i.e. the effect of one propeller on the other. Note the “-“ signs in the system matrix, denoting that each degree of freedom affects the other negatively. Fortunately, the coupling constants are much smaller than the motor constants and the controller takes advantage of that in order to compensate for those mutual disturbances.

Table 13.1 summarizes the numerical values needed for this model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_b$</td>
<td>0.4064 (m)</td>
</tr>
<tr>
<td>$J_{pp}$</td>
<td>0.030706 (kg.m$^2$)</td>
</tr>
<tr>
<td>$J_{yy}$</td>
<td>0.030706 (kg.m$^2$)</td>
</tr>
<tr>
<td>$K_{ff}$</td>
<td>0.8722 (N/V)</td>
</tr>
<tr>
<td>$K_{fb}$</td>
<td>0.4214 (N/V)</td>
</tr>
<tr>
<td>$K_{tb}$</td>
<td>0.01 (Nm/V)</td>
</tr>
<tr>
<td>$K_{tf}$</td>
<td>0.02 (Nm/V)</td>
</tr>
</tbody>
</table>

Table 13.1: Helicopter Parameters
13.2 Integral Control via State Augmentation

In order to reject the gravitational disturbances and achieve better steady-state tracking, we need to introduce integrators in the loop. We augment the state variable model with the two new states, $p_I$ and $y_I$, that are the integrals of $p$ and $y$, respectively.

The new state variable equation is then:

$$
\begin{bmatrix}
\dot{p} \\
\dot{y} \\
\ddot{p} \\
\ddot{y} \\
p_I \\
y_I
\end{bmatrix}
= \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
p \\
y \\
\dot{p} \\
\dot{y} \\
p_I \\
y_I
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
L_b & K_{ff} & -K_{fb} & 0 & 0 & 0 \\
2J_{pp} & J_{pp} & 0 & 0 & 0 & 0 \\
-K_{ff} & J_y & -L_b & K_{fb} & 0 & 0 \\
-J_y & 2J_{yy} & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
V_p \\
V_y
\end{bmatrix}.
$$

The LQR gains are determined using the Matlab command ‘lqr’.

13.3 Connections

- Place the helicopter in an area where it can pitch and yaw freely without touching any other equipment.
- Make sure the base of the helicopter support is stable (place a folded paper between the base and the table if needed).
- Use the following diagram to connect the experiment (make sure you use the To Load cable labeled ‘5’ for the front motor and the To Load cable labeled ‘3’ for the back motor).
- Please refer to Figure 13.2.
13.4 Simulink Diagram and Results

(i) The first block in the Simulink diagram is the Helicopter I/O block:

- Use an encoder block to acquire the pitch angle. Assume that the initial position of the helicopter is 0 and use the gain of $4 \times 360/4096 = 0.35156$ deg/cycle at the output of the encoder.

- Use an encoder block to acquire the yaw angle. Assume that the initial position of the helicopter is 0 and use the gain of $-2 \times 360/4096 = 0.17578$ deg/cycle at the output of the encoder. (Caution: make sure you include the
minus sign. The back propeller voltage and the rotation of the helicopter have opposite signs)

- Use derivative filtering to obtain pitch and yaw rate (suggested filter pole: -100)

- Use a DAC block for the front motor voltage. Since the UPM-2405 cable labeled ‘5’ amplifies the voltage by 5, include a UPM gain of 1/5. Then add a saturation block of ±5 just as a safety precaution to protect the motor. After that, add the usual DAC gain of 1/10.

- Use a DAC block for the back motor voltage. Since the UPM-1503 cable labeled ‘3’ amplifies the voltage by 3, include a UPM gain of 1/3. Then add a saturation block of ±5 just as a safety precaution to protect the motor. After that, add the usual DAC gain of 1/10.

- The helicopter block should then have two inputs (front and back voltages) and four outputs (pitch, yaw, pitch rate and yaw rate)

- The detailed Simulink diagram of the Helicopter block is given in Figure 13.3.

- Create two subsystems for angle safety stop. The two subsystems should stop the simulation whenever the pitch and yaw angles exceed a reasonable safe value. They should also allow the user to ‘land’ the helicopter smoothly whenever commands are entered in dSPACE (see Figures 13.4 and 13.5).
Figure 13.3: Helicopter I/O block

Figure 13.4: Pitch Safety Stop
(ii) The second block is the Command block:

- Use step blocks to create a pitch command of 45 degrees at time $t=5$ and 30 degrees at time $t=40$.
- Use step blocks to create a yaw command of 20 degrees at time $t=20$.
- Now use switches to allow real-time user command from dSPACE.
- Use rate limiters to limit the slope of the step commands to $\pm 1000$ (This is important because sudden step voltages to the motors should be avoided since they could cause currents as high as 5 A to flow into the armature).
- The command block should then have two outputs: pitch command and yaw command. Details are shown in Figure 13.6.
(iii) The final block is the Controller block:

- This block has four inputs: pitch error, yaw error, pitch rate and yaw rate. Details are shown in Figure 13.7.
- It has two outputs: front voltage and back voltage.
- Use the values of the LQR gain matrix inside this block to form gain banks that multiply the four states.
- Include the output integrators (additional two states) in this block and multiply them by the corresponding bank of gains (see Figure 13.8). Saturate the resulting products by $\pm 10$: this is an ad hoc integrator anti-windup scheme. Find the best value of saturation by trial and error.
- Sum up all the products and connect them to the two outputs (this constitutes state feedback in the form $u = Kx$).

- **Hint**: Use the following matrices in your first LQR iteration

$$Q = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

![Figure 13.7: Controller Block Details](image-url)
Figure 13.8: Integration Details

Figure 13.9 shows the complete Simulink block diagram of the experiment.
Figure 13.10 shows the results of a step of 45 degrees for pitch and a step of 20 for yaw. Note the effect of coupling following the rising edge of each step.

13.5 Discussion

The results obtained in the previous section were better than the sample results provided by Quanser in [4]. This is due to a finer tuning of the LQR gains. We could
achieve better tracking and disturbance rejection if we design nonlinear observers for the pitch and yaw rates and apply nonlinear control.

The 2 DOF Helicopter is currently used as a demonstration to motivate the students at the beginning of the term. It could be integrated in the laboratory as an independent project in which the students would work on the current model and experiment with tuning the LQR gains.
14.1 Summary

In this thesis, we have developed an undergraduate control systems laboratory course in the form of nine experiments using the dSPACE DS1104 controller card with Simulink and Real-Time Workshop to control physical plants provided by Quanser Consulting Inc. The experiments relate a broad range of fundamental concepts in control theory and help students build a practical knowledge of the Simulink and dSPACE software packages for control design and analysis. After taking the course developed in this thesis as a first lab course in control systems, students should become familiar with the most important concepts in control design, measurement and data acquisition.

We also introduced three independent experiments that use more advanced control principles in order to emphasize the importance of state-feedback, observer design, and LQR design. Those three experiments (LQR control for flexible joint, flexible link and helicopter) could be used in an undergraduate laboratory course provided that students possess the necessary background in state-space methods and state-feedback analysis and design.
14.2 Recommendations for the Future

The materials discussed in this document, although very descriptive and educational, are mainly meant to teach undergraduate students a first lesson in measurement and control. Throughout the whole document, we did not emphasize dynamical modeling issues and the problem of nonlinearities in the plants such as deadzones (due to gear backlash), hysteresis, and saturation. These topics would constitute a very good ground for the development of a graduate-level course where students would concentrate more on system identification and mathematical modeling and on developing more advanced controllers such as nonlinear controllers, auto-tuning PID’s and other robust controller techniques.
APPENDIX A

TRANSIENT CHARACTERISTICS

Let a general second order transfer function with output, $Y(s)$, and input, $R(s)$, be represented as:

$$\frac{Y(s)}{R(s)} = G(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Figure A1: s-Plane Diagram for a Complex Pole
The figure above gives a generic S-plane diagram for a pair of complex conjugate poles. The system quantities annotated in the figure are also used below in important control system quantities utilized in the evaluation of the closed loop performance. These same quantities in are also employed when estimating transfer functions.

DC gain

Undamped natural frequency

Damping ratio

Damping constant

Frequency of oscillation

System time constant

Time to peak overshoot

Steady-state output

Peak overshoot

Percent overshoot

ε% settling time

Rise time
Resonance frequency

\[ \omega_r = \omega_0 \sqrt{1 - 2\zeta^2}, \quad \zeta \leq \frac{1}{\sqrt{2}} \]

Magnitude of resonance

\[ M_r = \frac{1}{2\zeta \sqrt{1 - \zeta^2}}, \quad \zeta \leq \frac{1}{\sqrt{2}} \]

(note: \( M_r \) is not in dB units here)
Preliminary Note

The dSPACE software installation CD’s needed for this Lab are accompanied with a floppy disk that checks the presence of a valid hardware key that is physically attached to the parallel port on the back of each station. The software will not run properly unless the hardware key is connected at all times. Each dSPACE version requires a specific version of Matlab and will not install unless it finds the corresponding installation of Matlab on the hard drive.

Laboratory #1

In this Laboratory, the TA must begin with a general introduction about control systems and the terminology used in control theory. He/she should give a brief historical background talking about analog control and then explain the concepts of measurement and sensing for the purpose of control with emphasis on the importance of signal interfacing. The DC servomotor can be used at the beginning to explain the principal of closed-loop feedback control: the TA would connect the DC servomotor as in a regular
experiment and show the students how to close the loop through the controller board (digital computer) while explaining the purpose of feedback and the benefits of digital control.

The TA should send an email to the class some time before the first Laboratory asking them to read the Lab manual and to bring a zip disk to save acquired data for the post-laboratory assignment. A zip disk will be needed for all Laboratories. At the end of Laboratory #1, the TA should ask the students to read the Lab manual for the next week, prepare the pre-laboratory assignment and write the results that they judge important for the next Laboratory procedure on a separate sheet of paper that they will keep during the Laboratory session.

Before going to the Lab, the TA should read and prepare the Laboratory first, then read the following important notes.

Initialization files

Laboratory #1, in addition to introducing dSPACE and Simulink, implements the concept of frequency sweeping for system identification. The TA must explain the principal of frequency response of linear time invariant systems. Because the servomotor used in this Lab does not support high frequency input, a second order plant is simulated in Simulink and the corresponding black box is masked and saved in a file named lab1.mdl. Another *.m file containing the parameters of the transfer function is saved in the same folder. Those two files must always remain in the same folder.
(C:\EE557\LAB1) and the TA should make sure to erase all work done in a previous section and re-copy the two files for the next section. These two files are found on the CD prepared for the TA in EE557.

Paths and working directories

It is very important that all work be done in one single folder. For instance, in Laboratory #1, this folder would be C:\EE557\LAB1. The working directory of Matlab should be set to this path and all files should be saved in this folder. When an experiment is created in ControlDesk, it should be saved under the working root C:\EE557\LAB1 for Laboratory #1 and so forth. A problem could arise if paths are confused, and it is often a large source of confusion for both the students and the TA. This problem is related to *.sdf files and will be treated in the next note.

*.sdf Files

The executable files that are built after compiling a Simulink model have the “sdf” extension. They are the files that are downloaded on the DS1104 Controller Board for real-time execution. Referring to Fig. 2.14, if we click on the file selector tab in the bottom of the Tool Window, we will see a file listing similar to a Windows Explorer list. Whenever a model is compiled, the corresponding sdf file is automatically downloaded on the DS1104 Controller board; however, one can always find the file under the file selector tab and drag and drop it on the DS1104 icon under the platform tab in the navigation window. Sometimes, when different paths are used for the same experiment
two sdf files are generated and opened simultaneously in the Tool Window and it may become very difficult to discern which one is running on the controller board. If this situation occurs, simply right-click on the sdf file tabs in the Tool Window and click “close”. Then make sure all the relevant files are located in the same folder and load the sdf file again.

*Note: After each Lab session, the TA should erase all the files saved in the EE557 folder to reduce the risk of duplicate file conflicts.*

**sdf file variables**

Every time a Simulink model is compiled and the corresponding sdf file loaded, all variables of the Simulink model will appear in the Tool Window under the sdf file tab. Under the main root of the sdf file are the general variables that are part of every sdf file. Under the Model Root are the variables specific to each Simulink model, i.e. the block names and the block parameters. When you click on Model Root, a listing of the wire labels also appears in the right-hand part of the Tool window. It is up to the user to choose whether to label the wires or just use the block names themselves for GUI purposes.

*Note: Every Simulink variable is split into two variables in dSPACE. For instance, a gain named K in Simulink, will have a corresponding variable under the Model Root in the Tool window. If you click on that variable you will notice two variables in the right-hand part of the Tool Window: one is called “out” and represents the output of the gain*
block that would be used as a signal for plotting, the other is the value of the gain that
would be associated with a numerical input block in order to change it in real-time (see
numerical input block section in Laboratory #2 below).

The On/Off Button

As mentioned in the Laboratory procedure, real-time execution can be stopped from
ControlDesk, before the Simulink simulation time expires, by clicking on the stop button
in the toolbar. However, this does not guarantee that DAC voltage will be set to zero. In
fact, the termination “set to 0 volts” parameter is not executed unless the Simulink model
stops the simulation. Therefore, it is necessary to implement a manual function in the
ControlDesk layout in order to isolate the controller board from the outside world at any
desired time. This is done by adding an On/Off button found in the virtual instrument
menu (this block usually has two On/Off buttons, and one of them should be removed by
right-clicking on the block and going to its properties). Then, after having compiled the
model, the variable “simstate”, found under the root of the model’s sdf file tab in the Tool
window, should be associated with the On/Off button. The On/Off button can then be
used only when the experiment is run in animation mode and its use has to follow this
order:

- While in animation mode, click on the On/Off button to stop the experiment at
  any time.
- Click on the stop button in the toolbar.
- Switch to edit mode and hence be ready for another run.
Note that this button really works as an Off button only since you cannot use it to
restart the experiment; instead, you have to run the experiment by clicking the play
button from the toolbar again. This can be avoided by using a Radio Button but we chose
to keep the On/Off button mainly for safety reasons.

The “edit capture settings” Window

Section 2.3.1 explains the use of the “edit capture settings” window. The length of
data capturing, as mentioned in the Laboratory procedure, should be at least equal to the
Simulink simulation time. In practice, however, two to three seconds are lost while
switching from edit mode to animation mode and they can be accounted for by increasing
the capture window length (in the case of Laboratory #1, 65 would be a good choice).
The “edit capture settings” window can be accessed by right clicking on the plot and
clicking on properties. A faster way to open it is to click on the # button found on the
right side of the Tool Window (See Fig 2.14). In some instances, a glitch in the dSPACE
software prevents the user from seeing the “edit capture settings” window because it is
docked behind other windows and is very hard to distinguish. If that happens (usually it is
hidden behind the instrument selector window), just close all the windows, find the “edit
capture settings” window and undock it by moving it to the center. To open the other
windows that you closed, click on View in the toolbar, select “Controlbars” and activate
the desired windows.
The DS1104 Interface Board

The interface board is connected to the dSPACE card via two cables that have their corresponding sockets labeled PB1 and PB2 on the interface board. If, when the two cables are plugged, the red LED is not on, this indicates that the fuse on the board must be replaced. The TA should then replace it before the session. Fuses on the interface board are installed near the encoder inputs and they will be damaged if the encoder cable is plugged unevenly. To avoid fuse damage, all the leads of the encoder plug must be inserted simultaneously.

Laboratory #2

In this Laboratory, the TA should first explain the principle of sampling and the reason behind the need for sampling. Then he/she should present the concept of reconstruction and follow that by a frequency-domain analysis of aliasing (Nyquist theorem and such). A time-domain analysis of aliasing is crucial for the students to understand the meaning of this concept. Often times, time-domain analysis of aliasing is neglected in undergraduate courses because it cannot be very well described quantitatively, nevertheless, a qualitative approach is enough for the scope of this course. The TA could draw a sinusoid and sample it “slowly”, then reconnect the dots and show how a sinusoid of smaller frequency is obtained. A note should be made about the equivalency between “connecting the dots” and the action of a low-pass filter in the frequency domain. If asked, the TA should then draw an RC circuit and explain how a reconstruction low-pass filter works.
Function Generators

There are two types of function generators (FG) in the Lab: digital and analog. The digital FG’s are precise and provide frequencies and amplitudes exactly as dialed. The analog FG’s are less accurate and their output signal needs to be visualized on the oscilloscope before being fed to the acquisition board. If you notice a malfunction in the analog FG’s, clean the switches by repeatedly pushing them in and out while the FG is off; also, make sure the ‘duty’ knob is turned counterclockwise all the way.

Numeric Input Block

In this Laboratory, the students need to change the frequency of a sinusoid to perform a frequency sweep. It can be tedious and time-consuming to change this frequency in Simulink since they will need to re-compile every time a change is made to the original model. The benefit of dSPACE ControlDesk is that the user can change parameters directly from the GUI layout without having to re-compile. The TA should show the students how to add a “numerical input” block from the Virtual Instrument Selector and associate the sine wave frequency with it. Note that every time the experiment is run, the numerical input block will have the value of the frequency that is saved in the original Simulink model. To change it, the user must double-click inside the block and type the desired value, then type Enter (the value can also be changed by using the up and down arrows of the block, but it is not as practical). Note that, in future Laboratories, the students will need to tune controller gains in that fashion.
Laboratory #3

This Laboratory is self-explanatory; the only part that the TA should explain carefully is the procedure to determine the tachometer constant. A confusion arises from the fact that the measured ADC output is in Volts and cannot be compared to the theoretical output in rad/s, moreover, it cannot be used to determine the motor transfer function; it can only be used to determine an intermediate transfer function and the goal is then to compute the tachometer gain in order convert the output from Volts to rad/s and arrive to the final transfer function. The method to determine the tachometer gain relies on averaging. First, the average angular speed is calculated manually by allowing the motor to spin one rotation and measuring the elapsed time. Then, using the acquired plot (in Volts), the students should compute the corresponding average voltage output; to do so, the data vectors must be truncated manually in Matlab so as to include only the range from the instant the voltage start going above zero to the stop time $t_{stop}$. Then, using the “Data Statistics” function under “Tools” in the toolbar of the Matlab figure, the average or mean voltage can be read and shown on the Matlab plot.

Laboratory #4

This is the first experiment where the students perform closed loop feedback control. The TA should mention that the main objective of control systems is precise tracking or regulation in the face of disturbances. It should be made clear that even though it is often omitted from the desired specification, zero steady-error is an obvious requirement (a
tolerance of ±1% will be allowed in some instances). In this Laboratory, since there is only one degree of freedom, namely the gain K, the students are asked to achieve the “best transient response”. Whenever there is only one parameter to change, no specifications will be given and the best response is left to the judgment of the students and the TA.

Due to the nature of the equipment, different responses can be obtained every time the experiment is run, even though the same gain is used. One way to deal with this problem is to align the inertial load with the zero mark every time before running; this will ensure that the experiment is run under the same conditions each time. This behavior is not normal since the transfer function of the motor should be the same regardless of the initial conditions; the investigation of this problem is beyond the scope of this course.

At this point, if it was not already done, the TA should introduce the use of a “display” block in the layout. This block can be found under the Virtual Instruments panel and can be used to display any variable numerically, for example the steady-state error to a step input.

**Caution:** the TA should verify the Simulink diagram before compilation. No gain should exceed a value of 5 in this laboratory or in any other laboratory!

Laboratory #5

This Laboratory introduces a compensator with a zero, a pole and a gain. The pre-lab assignment requires the students to derive two lag compensators, one using the Bode design method and the other using root locus techniques. Once they determine the
transfer functions of those two compensators, the only values they have to tune in the Lab
are the compensator gains. Note that the Bode design method does not eliminate steady
state error because it only deals with improving the phase margin of the system; in order
to eliminate steady state error, a gain is needed with the compensator and this is done in
the root locus design method.

The TA should also warn the students about a common mistake before they start
working on the pre-laboratory assignment: Bode analysis is done on open loop systems.
Adding a compensator does not mean that the new Bode plot should be that of the closed-
loop system but it is simply that of the open-loop cascaded system (compensator &
plant).

Also note that the goal of this experiment is to perform lag control and therefore there
is no need to achieve fast settling time, therefore aggressive gains should be avoided in
favor of a small gain that assures only zero steady-state error.

Caution: the TA should verify the Simulink diagram before compilation. No gain
should exceed a value of 5 in this Laboratory or in any other Laboratory!

Laboratory #6

The procedure is similar to that of the previous Laboratory. The pre-lab assignment
asks students to prove certain formulas. It is very important that the TA make it clear to
the students beforehand that proving (or showing or demonstrating) is not verifying the
equation, or plugging-in and checking, or referencing a book; it is a formal logical
procedure that uses as a hypothesis the equalities given in the question and any
mathematical knowledge, and that follows mathematical and logical implications and equivalences to arrive to the conclusion or the relation that is to be proven. The relation to be proven cannot be used during the process of the proof, i.e. the consequent cannot be assumed. The only assumption that can be made is the negation of the consequent, which will lead to a reductio ad absurdum proof (not necessary in any pre-lab).

Laboratory #7

Part (iv) of the pre-lab will yield extremely high gains, which is very unrealistic. The purpose of part (vi) is to explain to the students that PID gain tuning is mainly ad hoc. The Simulink model in part (vi) is needed because the controller output (motor input) step response cannot be obtained automatically through the rlttool design GUI due to the non-causality of the PID controller’s transfer function. The specification given in part 6 (2% overshoot, 0.1 sec settling and certainly zero steady state error) can be achieved in the preliminary design without the use of integral gain, i.e. simply with a PD controller, because the transfer function of the plant is of type 1 and hence naturally has zero steady state error to a step. A choice of $K_p = 4.1$ and $K_d = 0.055$ will meet the specifications, even though adding a $K_i$ gain of about 0.1 will not alter the response significantly. We had to use a type 1 transfer function because it represents the motor on which the PID controller will be applied; if we refer to Laboratory #4, we see that a simple gain compensation is enough to achieve nearly perfect control. We chose to keep all PID gains for the sake of generality and for educating the student in PID tuning. Another important reason for keeping all three gains is that the motor exhibits steady-state error (noticeable
under the effect of disturbances) due to nonlinearities caused by gear backlash. Integral control will help reduce that steady-state error and achieve better disturbance rejection.

The TA should explain the effect of proportional, integral and derivative control laws. The main tuning strategy is to start with choosing $K_p$ while maintaining the other gains equal to zero. Then $K_i$ will help reduce steady-state error and $K_d$ will help reduce overshoot. The three gains are linked by a trade-off relationship. The TA should then let the students tune the PID controller. This will familiarize them with the tuning process and, if they ask for the intervention of the TA, he/she can help them fine tune the controller.

Later in Laboratories 8 and 9, students will have the chance to explore the differences between PD and PID controllers.

Laboratory #8

Since we only have two apparatus for this experiment, the TA might need to split the class into two shifts, depending on the number of students. Before the Lab session, the TA must set up the motors in high gear ratio (Fig 4.5) and mount the flexible joints. It is extremely important that the flexible joint be fastened tightly so that no movement in the vertical direction is allowed; only movement of the joint in the horizontal plane is permissible.

Before compiling the Simulink model, the students should keep the power supply turned off because the DAC might initially have a constant output fed to the motor, which
will cause the apparatus to spin continuously. An On/Off button should be added and used in animation mode to isolate the motor from the dSPACE card. Once this procedure is accomplished, the power supply can be turned on again and the experiment can start.

During the Lab session, the joint must be brought to its initial position after each iteration so as to avoid tangling of the sensor wires.

Laboratory #9

If the number of students in a section required making two shifts in week 8, the same groups will rotate shifts so that everybody will have had both shifts. Before the Lab session, the TA should dismount the flexible joints from Laboratory 8 and mount the flexible links in the same fashion.

Again, before compiling the Simulink model, the students should keep the power supply turned off because the DAC might initially have a constant output fed to the motor, which will cause the apparatus to spin continuously. An On/Off button should be added and used in animation mode to isolate the motor from the dSPACE card. Once this procedure is accomplished, the power supply can be turned on again and the experiment can start. The link must be brought to its initial position after each iteration so as to avoid tangling of the sensor wires.

After the end of Laboratory 9, the TA should set up the motors in low gear ratio (Fig 4.4) in preparation for the next quarter. All the files in the C:\EE557 folder must be deleted and the initialization files of Laboratory #1 must be restored from the TA CD.
Lab Practical

The last week of classes is dedicated to a Lab practical that serves as a final examination for EE557. The aim of the Lab practical is to test the knowledge of the students in classical control design, Simulink and the dSPACE interface. The TA can design the final in two ways:

(i) Pose a control design problem similar to the Laboratories of the course, i.e. ask the students to implement a certain type of control on the DC servomotor.

(ii) Create an arbitrary plant in Simulink (transfer function block) and ask the students to use channel loop-back, i.e. connect the D/A to the A/D like in the first Laboratory and design a controller for the system.
BIBLIOGRAPHY


