

ECE 557 Lab 1:

A Tutorial Introduction to Data Acquisition, dSPACE and Simulink

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 1/12/12

Abstract: This tutorial served as an introductory guide to data acquisition and control systems development and implementation using dSPACE software, as well as providing information about Quanser plants to the dSPACE data acquisition cards. The tutorial was designed as a quick reference for a university laboratory course, with more information available if needed. We found the black box plant has transfer function

$$G_{p_tuned} = \frac{1000}{s^2 + 5s + 1000}$$

1. Introduction

The purpose of this lab was to become familiar with the operation of Simulink, dSPACE, and data acquisition using these programs. We used Simulink and dSPACE to identify an unknown transfer function in a Simulink block created and masked by the instructor. We were not given a black box as plant, and were given the task of identifying them through signal and system analysis. The example control system block diagram is shown in Figure 1.1 of the lab manual.

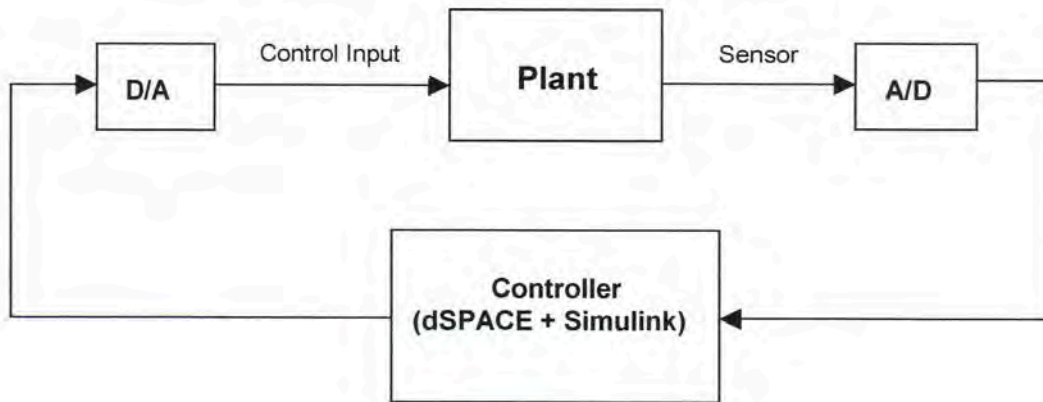


Figure 1.1 – Control System Block Diagram

2. Procedure

We used the procedure lined out in the Lab 1 section of the assigned lab manual, constructing the given Simulink model in Figure 1.4 of the lab manual.

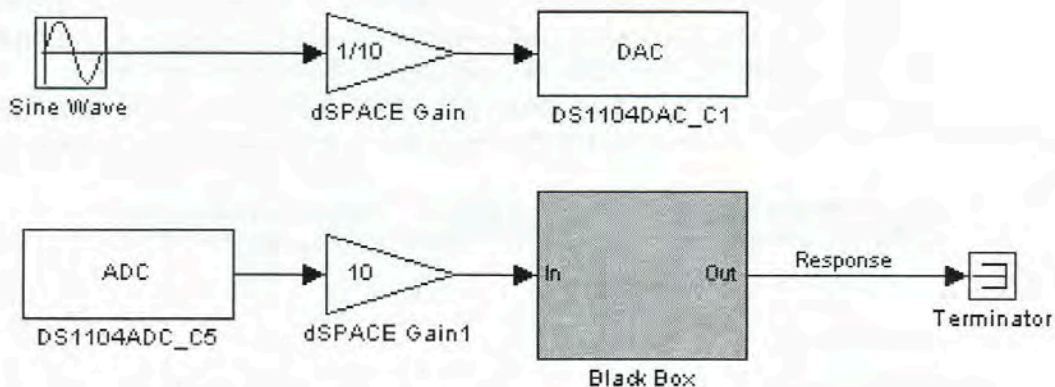
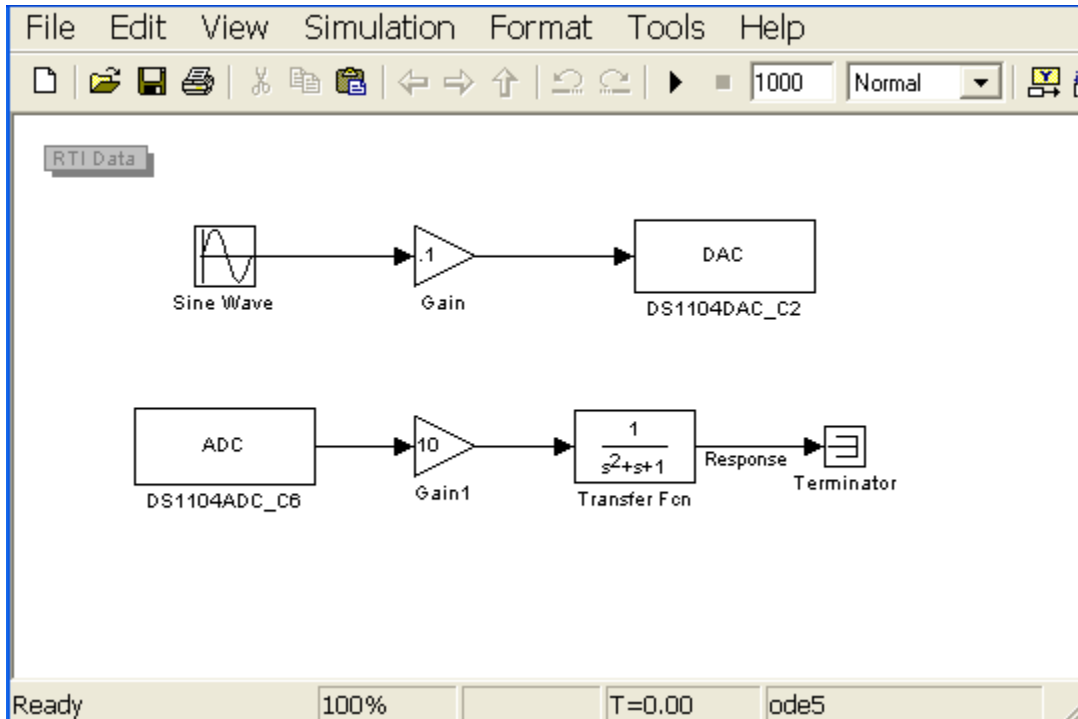


Figure 1.4 – Simulink Model

Our constructed model is shown below:



We constructed the model in Simulink, set our simulation parameters, connected it to the dSPACE interface board, ran the model in the ControlDesk Environment, compiled, then ran the experiment and gathered our data. Once we had our model's graphs, we ran them through a Matlab script to print out in more convenient plots.

3. Theoretical Results

We are modeling the plant using two-pole model (second order transfer function).

Standard form of such model and corresponding bode plot are as shown below.

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

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

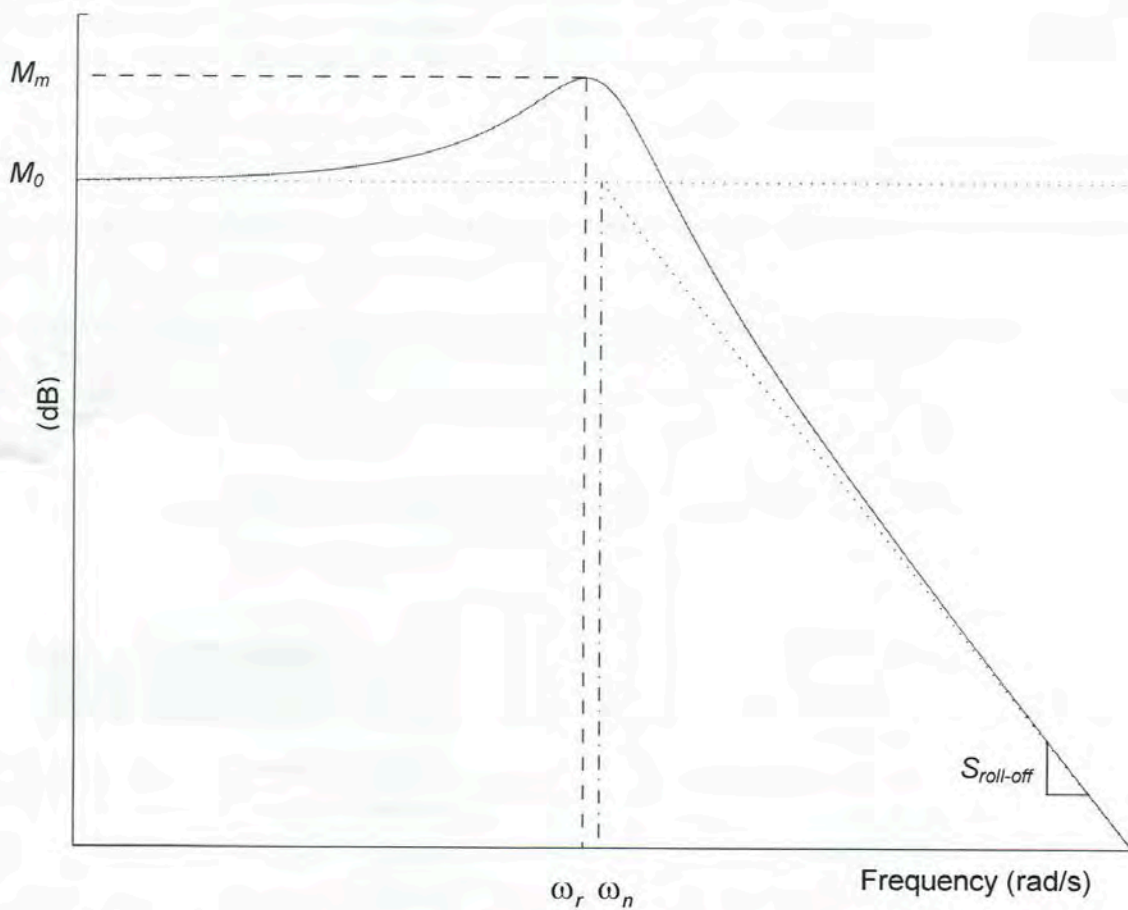


Figure 1.2 – Frequency Magnitude Response of a Second-Order System

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 1.7 gives a generic S-plane diagram for a pair of complex conjugate poles. The system quantities for a second order transfer function, annotated in Figure 1.7, are referenced by Table 1.1. Table 1.1 presents important control system quantities utilized in the evaluation of the closed loop performance. The quantities in Table 1.1 are also employed when estimating transfer functions.

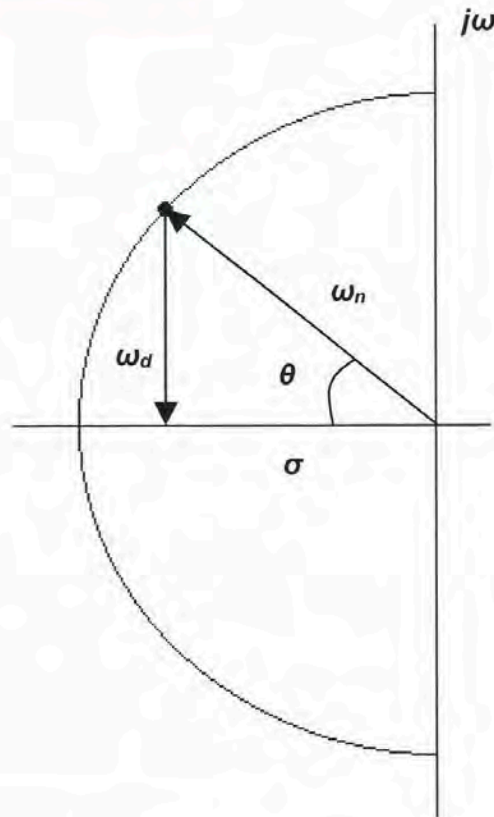


Figure 1.7 – S-plane diagram for a complex pole

Table 1.1 – Control System Performance Quantities

DC gain	K
Undamped natural frequency	ω_n
Damping ratio	$\zeta = \cos(\theta)$
Damping constant	$\sigma = \zeta\omega_n$
Frequency of oscillation	$\omega_d = \omega_n\sqrt{1-\zeta^2}$
System time constant	$\tau = 1/\sigma$
Time to peak overshoot	$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$
Steady-state output	$y_{ss} = \lim_{t \rightarrow \infty} y(t)$
Peak overshoot	$M_p = y_{ss} \left(1 + e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \right)$
Percent overshoot	$P.O. = \frac{(M_p - y_{ss})}{y_{ss}} \times 100\%$
$\varepsilon\%$ settling time	$T_s \cong -\frac{\ln(\varepsilon)}{\sigma}$
Rise time	$T_r \cong \frac{1 + 1.1\zeta + 1.4\zeta^2}{\omega_n}$
Resonance frequency	$\omega_r = \omega_n\sqrt{1-2\zeta^2}, \zeta \leq \frac{1}{\sqrt{2}}$
Magnitude of resonance	$M_r = \frac{1}{2\zeta\sqrt{1-\zeta^2}}, \zeta \leq \frac{1}{\sqrt{2}}$
(note: M_r is not in dB units here)	

4. Measured Results

For our post lab exercise we were given the following tasks stated below:

1.4 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:

- Submit three plots of captured data as explained in Section 1.3.2
- 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.
- Use the technique discussed in Section 1.2.1 to find an estimate of the transfer function of the Black Box.
- Produce the Bode plot of the estimated transfer function and mark (on your plot) the measured points.

Task I

Plots at $f = 0.1$, $f = 20$ and $f = 200$ are as shown below:

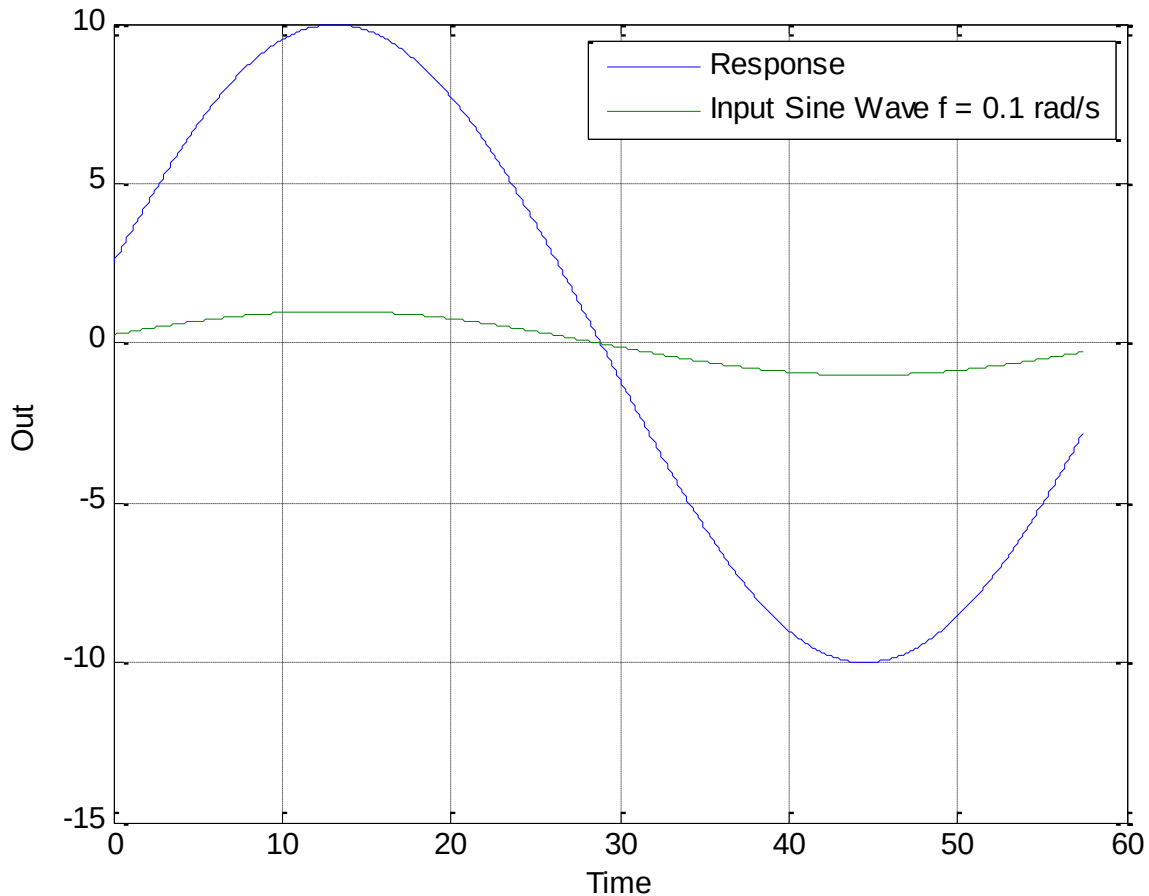


Fig 1. System magnitude response at $f = 0.1$ rad/s

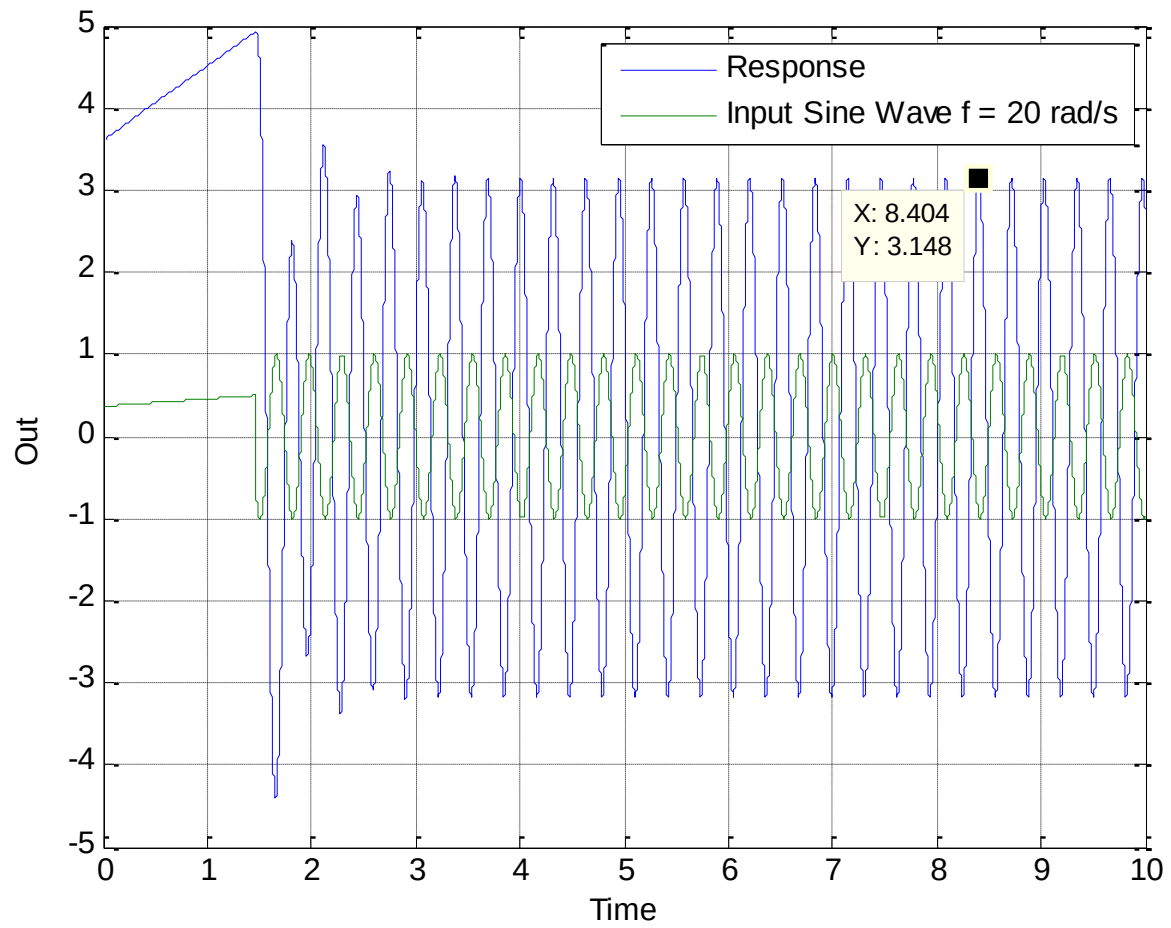


Fig 2. System magnitude response at $f = 20$ rad/s

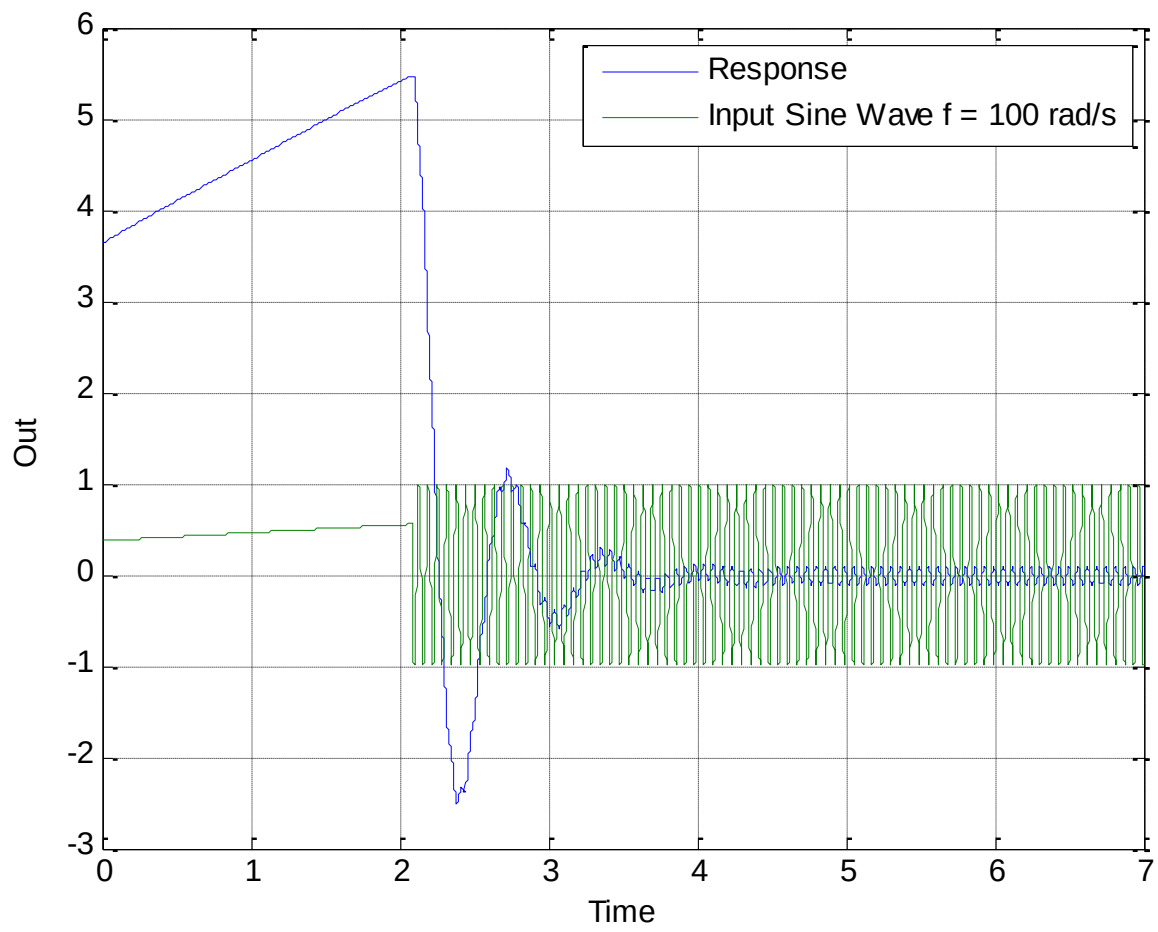


Fig 3. System magnitude response at $f = 100\text{rad/s}$

Task II

In table 1, we organized system magnitude response at selected frequency and calculated their gain ratios using the following formula:

$$Gain = Output \frac{Magnitude}{Input\ Magnitude}$$
$$Gain\ (dB) = 20Log_{10}(Gain)$$

Input Frequency/(rad/s)	Input Magnitude	Output Magnitude	Gain Ratio	Gain Ratio/dB
0.1	1	10.00	10.00	20.00
2	1	10.34	10.34	20.29
4	1	11.61	11.61	21.30
6	1	14.12	14.12	23.00
9	1	20.44	20.44	26.21
10	1	19.97	19.97	26.01
20	1	3.14	3.14	9.95
30	1	1.21	1.21	1.67
40	1	0.64	0.64	-3.85
50	1	0.40	0.40	-7.87
60	1	0.27	0.27	-11.28
70	1	0.19	0.19	-14.24
80	1	0.14	0.14	-16.89
90	1	0.11	0.11	-19.17
100	1	0.09	0.09	-21.31

Table 1. Measured gain ratio at selected frequencies

Plot the gain in dB vs. frequency in *MATLAB* gives bode plot of the system as shown in figure 4.

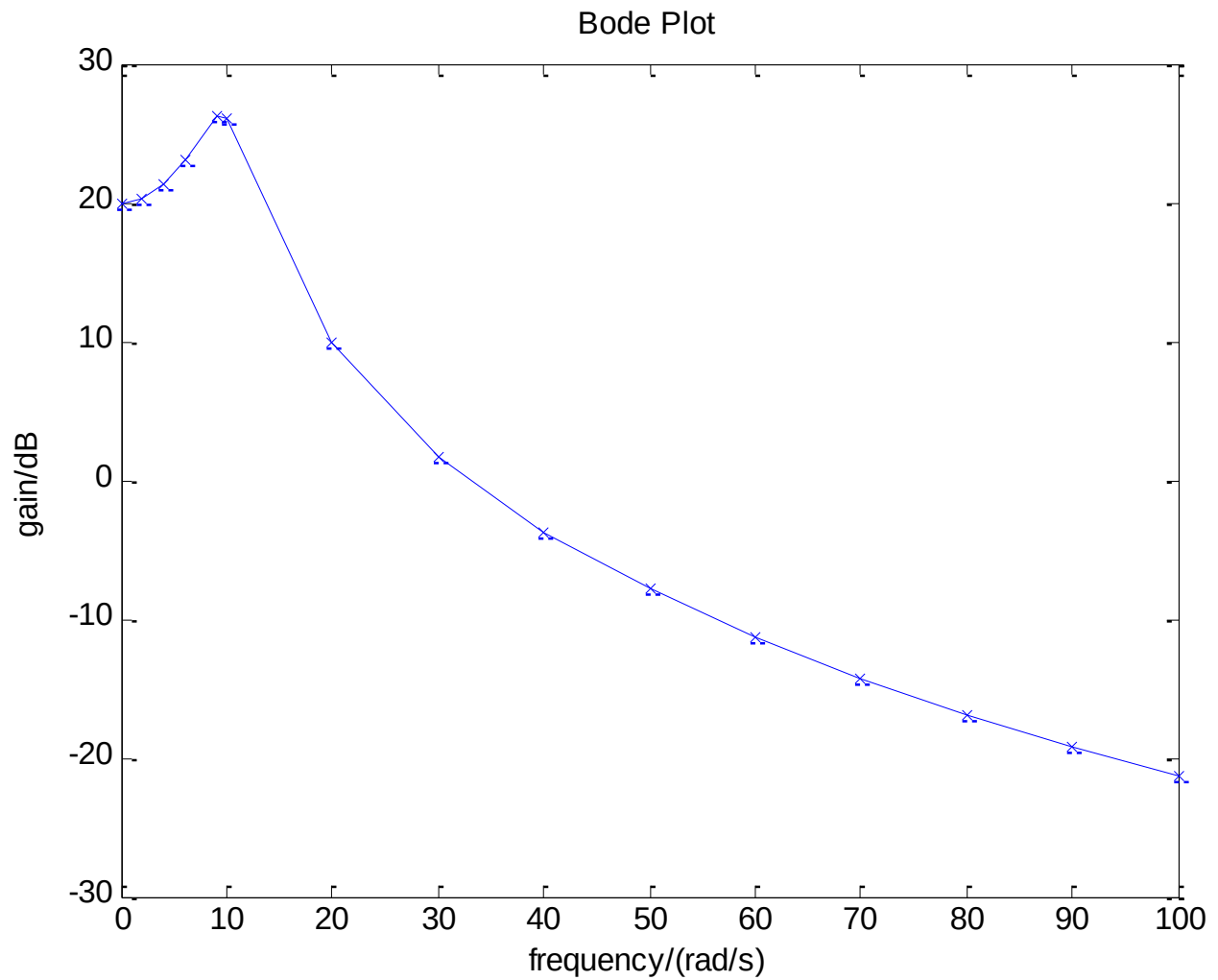


Fig 4. Experimental result of system's bode plot

Task III

In this part we estimate the plant's transfer function base on the experimental results as shown in table 1 and fig 4. Main goal is to find two pole model's damping ratio ξ , natural frequency ω_n and gain K. Please refer to *Theoretical Results* as a reference for the formulas used in this problem.

$$M_m = 26.21$$

$$M_0 = 20.00$$

Percentage Overshoot (PO):

$$PO = \frac{M_m - M_0}{M_0} \times 100 = \frac{26.21 - 20.00}{20.00} \times 100 = 31.05\%$$

$$PO = e^{-\xi\pi/\sqrt{1-\xi^2}} \times 100$$

$$-\xi\pi / \sqrt{1-\xi^2} = \ln(0.3105) = -1.170$$

$$\xi = 0.35$$

$$\omega_r = 9 \text{ rad/s}$$

$$\omega_r = \omega_n \sqrt{1-2\xi^2}$$

$$\omega_n = \frac{9}{\sqrt{1-2 \times 0.35^2}} = 10.36 \text{ rad/s}$$

$$K = 10^{\frac{M_0}{20}} = 10^{\frac{20}{20}} = 10$$

$$Gp = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{1073.30}{s^2 + 7.25s + 107.33}$$

Task IV

Fig 5 gives the bode plot of estimated system model found in Task III in *MATLAB* together with experimental data points on the same plot:

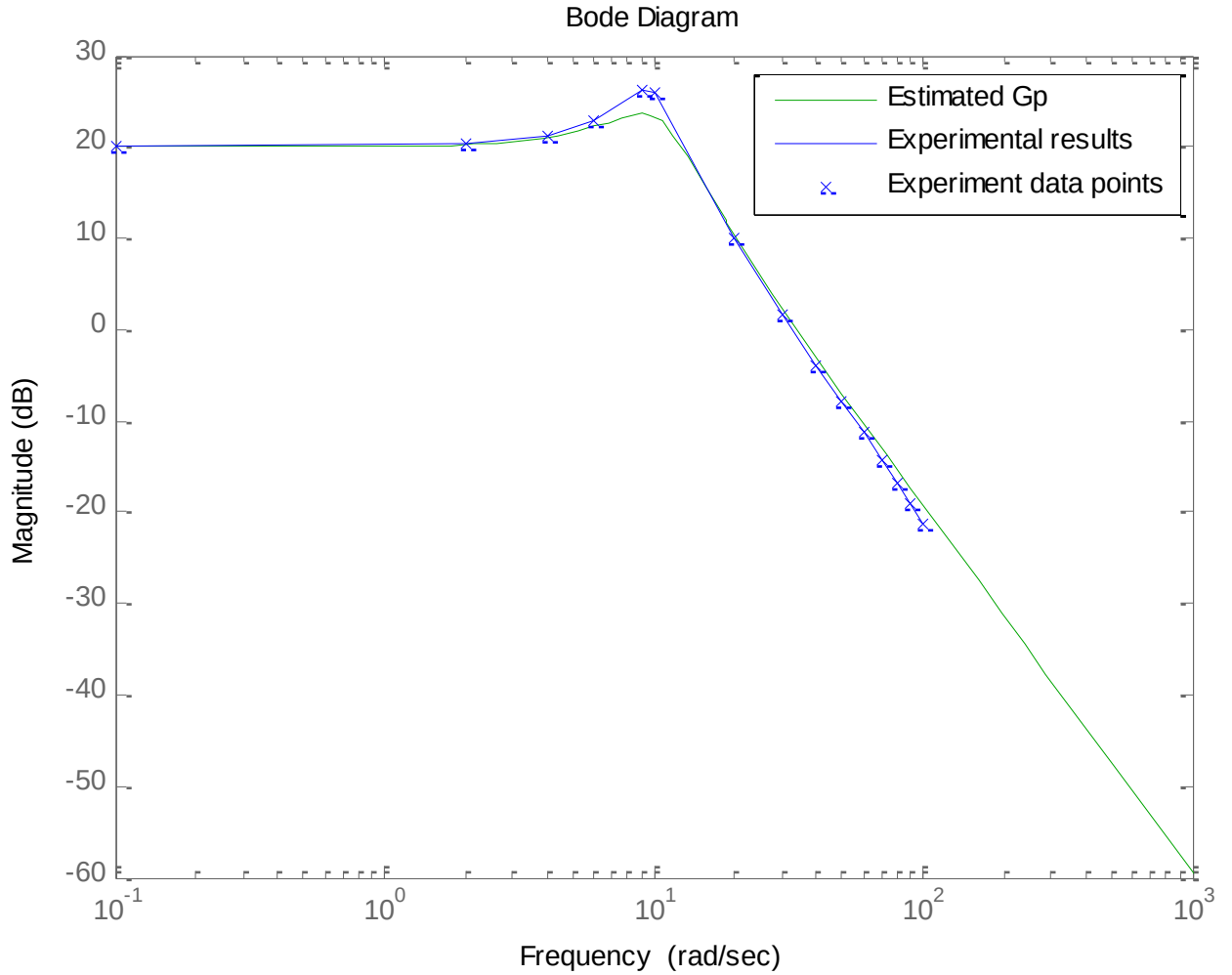


Fig 5. Bode plot of estimated system transfer function and experiment results

We found that the estimated system transfer function resembles the experimental results. By comparing ξ and ω_n value of the two curves in fig 5, we tuned our system transfer function to achieve best match of experiment data. Tuned transfer function is:

$$Gp_{-tuned} = \frac{1000}{s^2 + 5s + 1000}$$

Fig 6 shows the experimental results, estimated plant transfer function and tuned transfer function on the same plot:

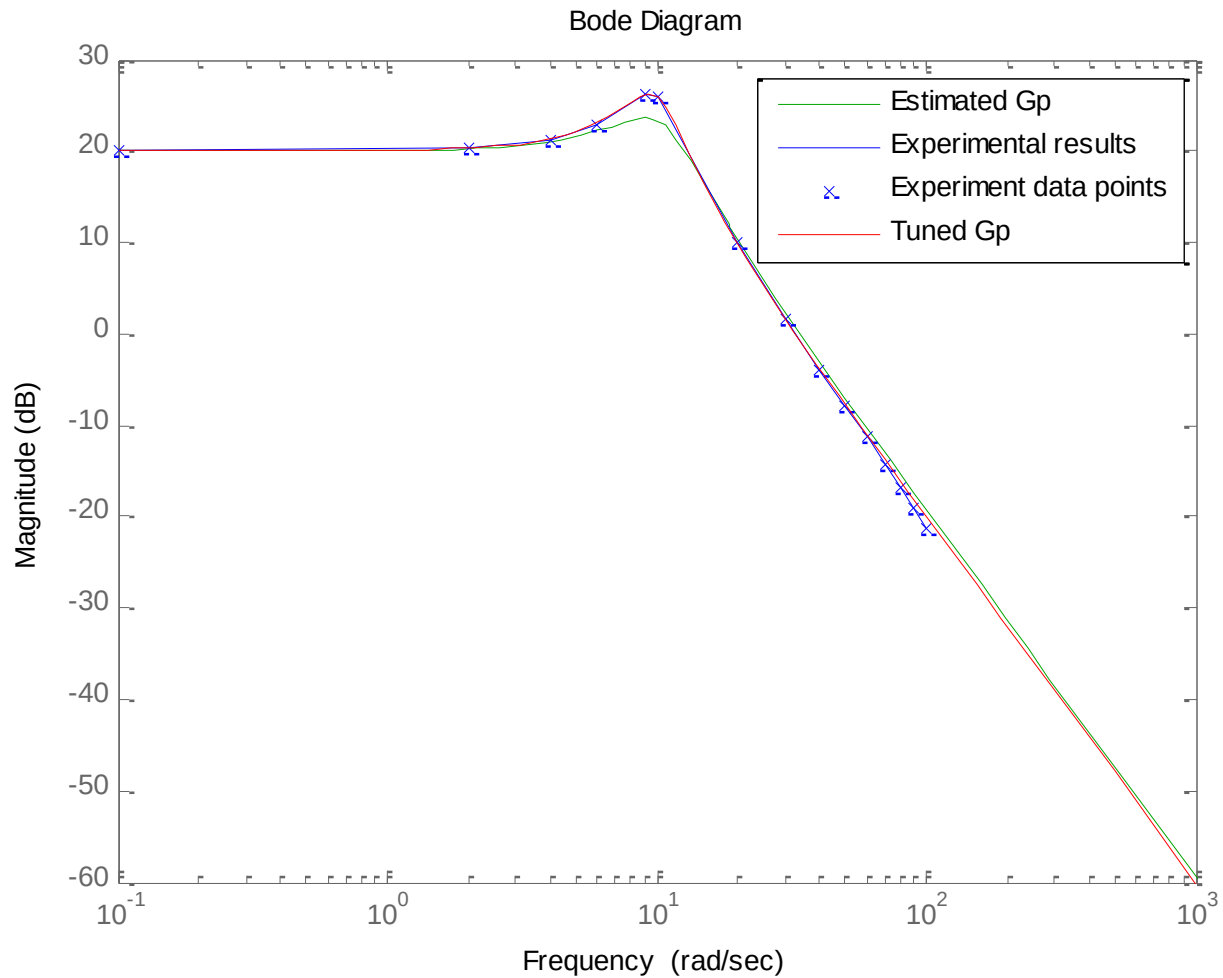


Fig 6. Estimated plant transfer function, experimental results and tuned estimated plant transfer function

Now we found a very close model for the plant.

5. Conclusions

This lab introduced the methods of data acquisition used for subsequent labs. We learned how to create a control system in MATLAB using Simulink and how to build the application for use with dSPACE. This lab introduced the dSPACE control desk and we learned how to create instruments and interfaces for use with the dSPACE data acquisition hardware, and the physical world.

The plant transfer function is estimated to be:

$$Gp_tuned = \frac{1000}{s^2 + 5s + 1000}$$

ECE 557 Lab 2:

Introduction to Digital Signal Processing

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 1/19/12

Abstract: This lab served as a brief introduction to digital signal processing and analysis methods. The effects of sampling an analog signal using an analog to digital (A/D) converter and then reconstructing the analog signal using a digital to analog (D/A) converter were analyzed.

1. Introduction

The purpose was a brief introduction to digital signal processing and analysis methods. The effects of sampling an analog signal using an analog to digital (A/D) converter and then reconstructing the analog signal using a digital to analog (D/A) converter were analyzed, with the effects of aliasing and quantization errors being demonstrated. The lab also reviews the basics of Fourier series analysis, and examines the sampling of finite and infinite bandwidth signals, while briefly addressing the concept of discrete-time equivalent approximation of a continuous-time filter.

2. Procedure

In this lab we explored a typical digital signal processing system shown in Figure 2.1, with a block diagram representation for an A/D converter shown in Figure 2.2, and a block diagram representation for a D/A converter shown in Figure 2.3.

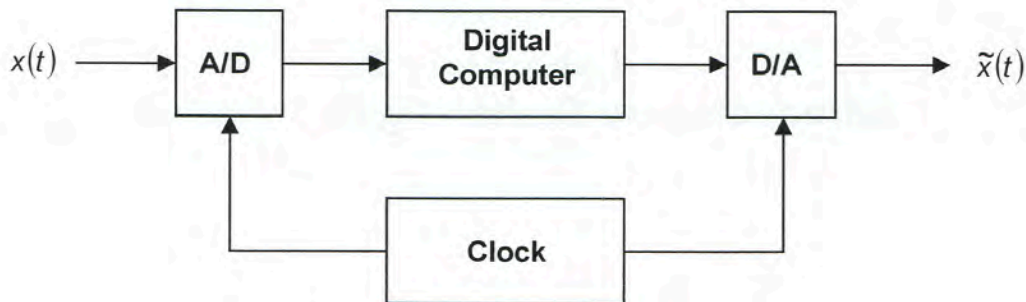


Figure 2.1 – Digital Signal Processing System

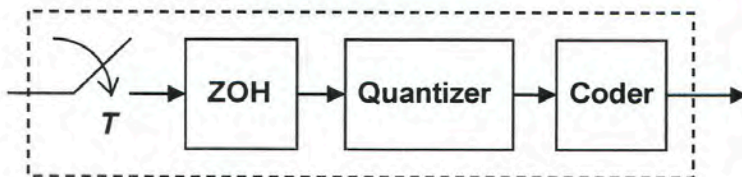


Figure 2.2 – A/D Converter

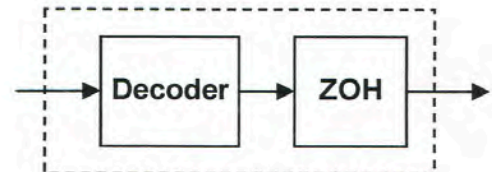


Figure 2.3 – D/A Converter

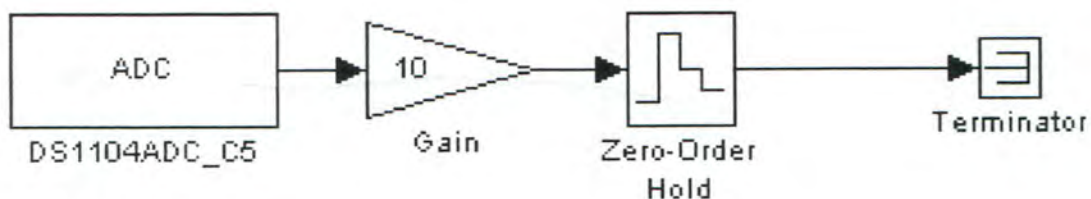
We used the procedure lined out in the Lab 2 section of the assigned lab manual, seen below:

2.4 Laboratory 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.
- 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.
- 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.

We constructed the given Simulink model in the lab manual, seen below:



We constructed the model in Simulink, set our simulation parameters, and connected to channel 5 of the A/D (12-bit) board. For the three different sampling frequencies we made sure to set the ZOH and simulation step size each time we change the sampling frequency.

Sampling Frequency (f_s)	Sampling Time (ZOH)	Simulation Step Size
4 Hz	1/4	1/40
12 Hz	1/12	1/120
1000 Hz	1/1000	1/10000

3. Results

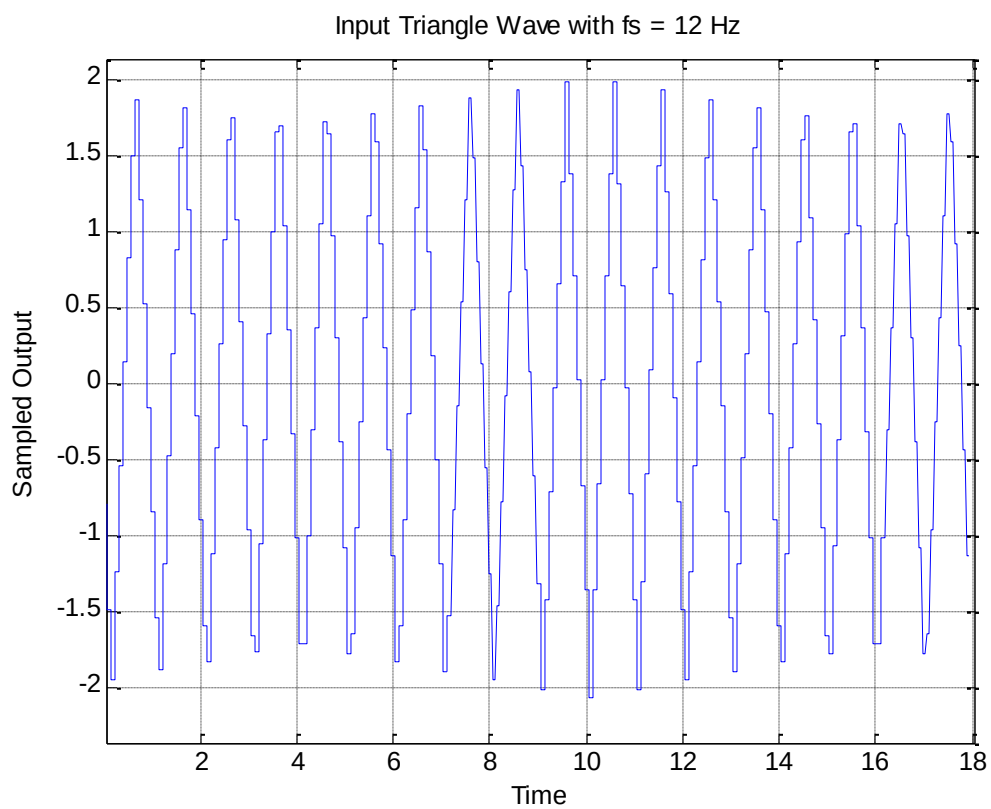
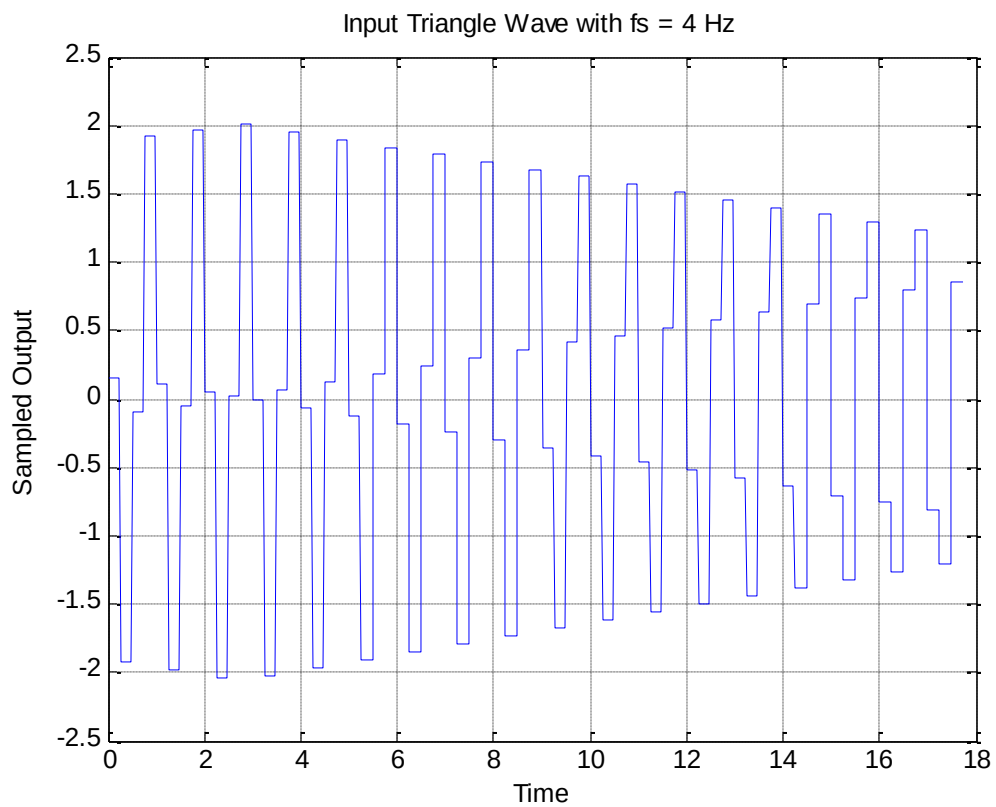
For our post lab exercise we were given the following tasks stated below:

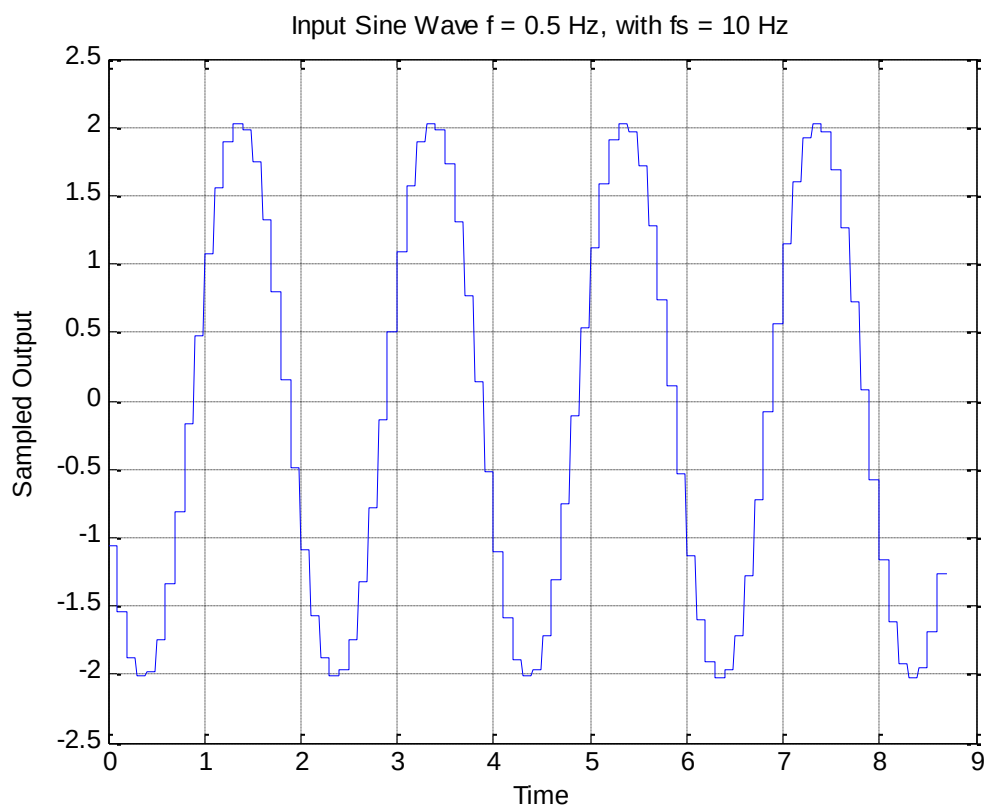
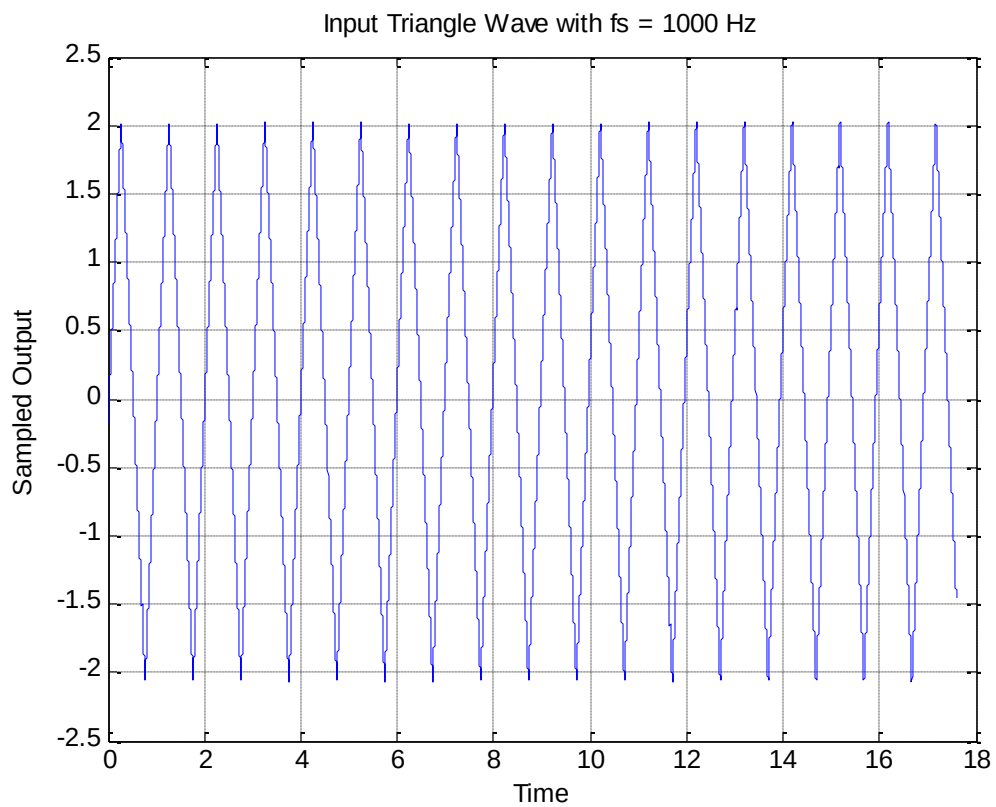
2.5 Post-Laboratory Exercises

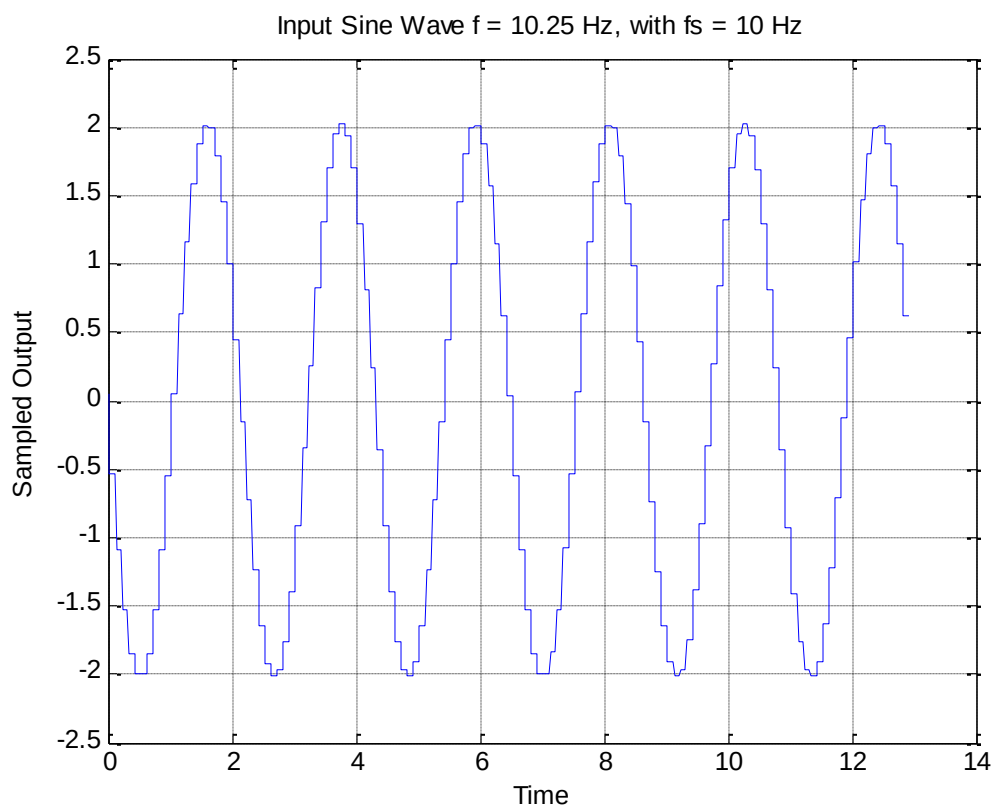
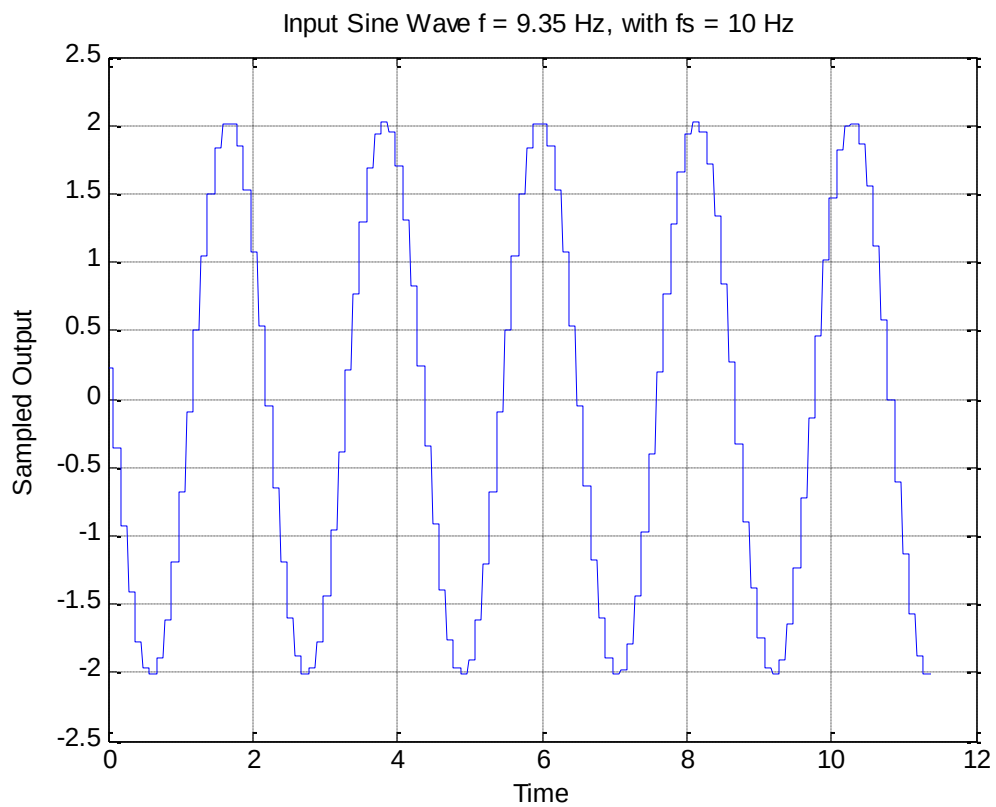
- Discuss the implications of aliasing and relate this concept with what you know about the frequency content of the triangular wave as determined in the Laboratory Preparation 1. Discuss your results and be sure to include a discussion on the accuracy of your results. Does the Nyquist Sampling Theorem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



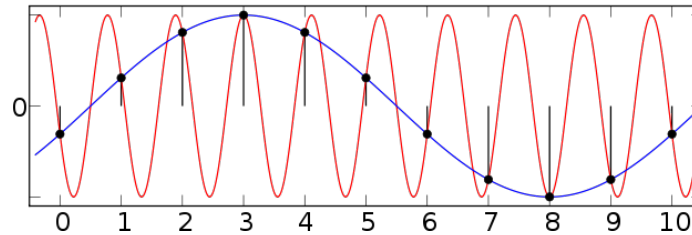




4. Discussion Questions

1.

The frequency of discretized analog waveform is dependent on the sampling frequency. In the figure below two sinusoidal waveforms with different frequencies are shown in blue and red.



The black dots represent a certain sampling rate. Both the red and blue waveforms have exactly the same discrete form at this sampling rate. This is called aliasing, which refers to the case where distinct frequencies become indistinguishable. The criteria for avoiding aliasing is Nyquist criteria, which says that

$$|\text{normalized frequency } f| = |\text{analog frequency } F / \text{sampling frequency } f_s| < 1/2$$

Based on Fourier theories, any signal can be decomposed into linear combination of its frequency contents. Aliasing will cause loss in frequency content in the original signal. With aliasing, discretized signal cannot be reconstructed back to the original waveform.

Fourier series of the triangular wave we input to the system is:

$$-2 \left(\frac{1}{2\pi f_0} \right)^2 \frac{1}{n^2} \cos(2\pi n f_0 t), \quad n \text{ is odd integer}$$

The sampling frequencies we used are 4Hz, 12Hz and 100Hz. Results of sampling under each sampling frequency are shown in our results section figures.

We can see at $f_s = 4$ Hz, the reconstructed waveform does not look like a triangular wave. At $f_s = 12$ Hz, a triangular waveform appears. At $f_s = 1000$ Hz, the reconstructed signal looks very much like a triangular wave.

Nyquist sampling theorem holds for triangular wave. Although triangular wave has infinite harmonics, the magnitude of higher harmonics is very small. This can be seen from the factor $1/n^2$ in the Fourier series. Therefore at $f_s = 12$ the triangular wave is still recognizable and $f_s = 1000$ is very close to the original signal.

It is clear that the sampled waveforms at $f_s = 4\text{Hz}$ and 12Hz have an envelope around. This is because the non-perfect synchronization of the input triangular wave frequency and the sampling frequency. e.g. the first sample may be slightly off the start of triangular waveform.

2.

We found that at 9.35Hz and 10.25Hz , the output frequency is both 0.5Hz . If both frequencies exist in a signal (assume no other frequencies are alias with these two), only the 9.35Hz will be preserved after reconstruction. They are not distinguishable.

3.

12bit ADC: $\text{SNR} = 6\text{dB} \times 12 = 72\text{dB}$, step size = $1/2^{12}$

16bit ADC: $\text{SNR} = 6\text{dB} \times 16 = 96\text{dB}$, step size = $1/2^{16}$

Based on the calculations, the higher bit ADC we use, the higher signal to noise ratio and smaller step size.

12 bit vs. 16 bit ratio of step size is 4:1 and SNR is $|3/4 \text{ dB}| = 20\log(3/4) = 2.5$. $2.5 < 4$. So choosing a higher number of bit provides more resolution but signal to noise ratio did not decrease with the same proportion. In another word, the result from a 16 bit ADC can exhibit more noise (relative to signal) and therefore more quantization error compare with a 12 bit ADC.

5. Conclusions

This lab introduced digital signal processing and analysis methods, as well as the effects of sampling an analog signal when using an analog to digital (A/D) converter and then reconstructing the analog signal using a digital to analog (D/A). The effects of aliasing and quantization errors were also demonstrated. The lab also reviewed the basics of Fourier series analysis, and examined the sampling of finite and infinite bandwidth signals, while briefly addressing the concept of discrete-time equivalent approximation of a continuous-time filter.

ECE 557 Lab 3:

Time Domain System Identification for a DC Servo

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 1/26/12

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

1. Introduction

In this lab we performed time-domain system identification for the DC servomotor that is the main actuator for the Quanser SRV-02ET equipment series. The time-domain system ID technique consisted of applying a step function input to the plant and measuring certain transient characteristics in order to estimate the transfer function. The design model in Figure 3.1 was created and evaluated for accuracy. A state-space model and the corresponding transfer function were evaluated using stated parameters, Laplace transform, and s-domain reasoning, as seen in the below figure.

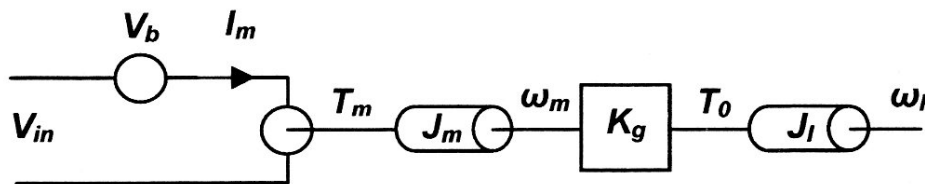


Figure 3.1 – Block Diagram of Position Servo

- θ is the angle of the output shaft
- ω_m is the angular velocity of the motor shaft
- ω_l is the angular velocity of the output shaft
- K_g is the gear ratio
- T_m is the motor torque
- T_o is the output torque after the gearbox
- J_m is the motor inertia
- J_l is the load inertia

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 $s=0$, $s=-41.58$

2. Procedure

The following connections were used to connect the DC servomotor, the DS1104 interface board, and the computer:

3.4.2 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.

The Simulink model in Figure 3.6 was designed in Matlab, but changed to the below figure for better performance and used in Matlab:

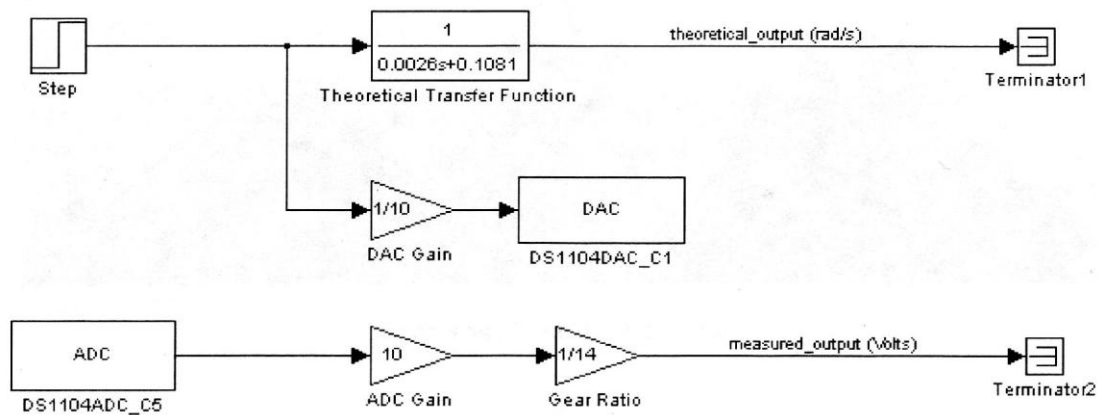
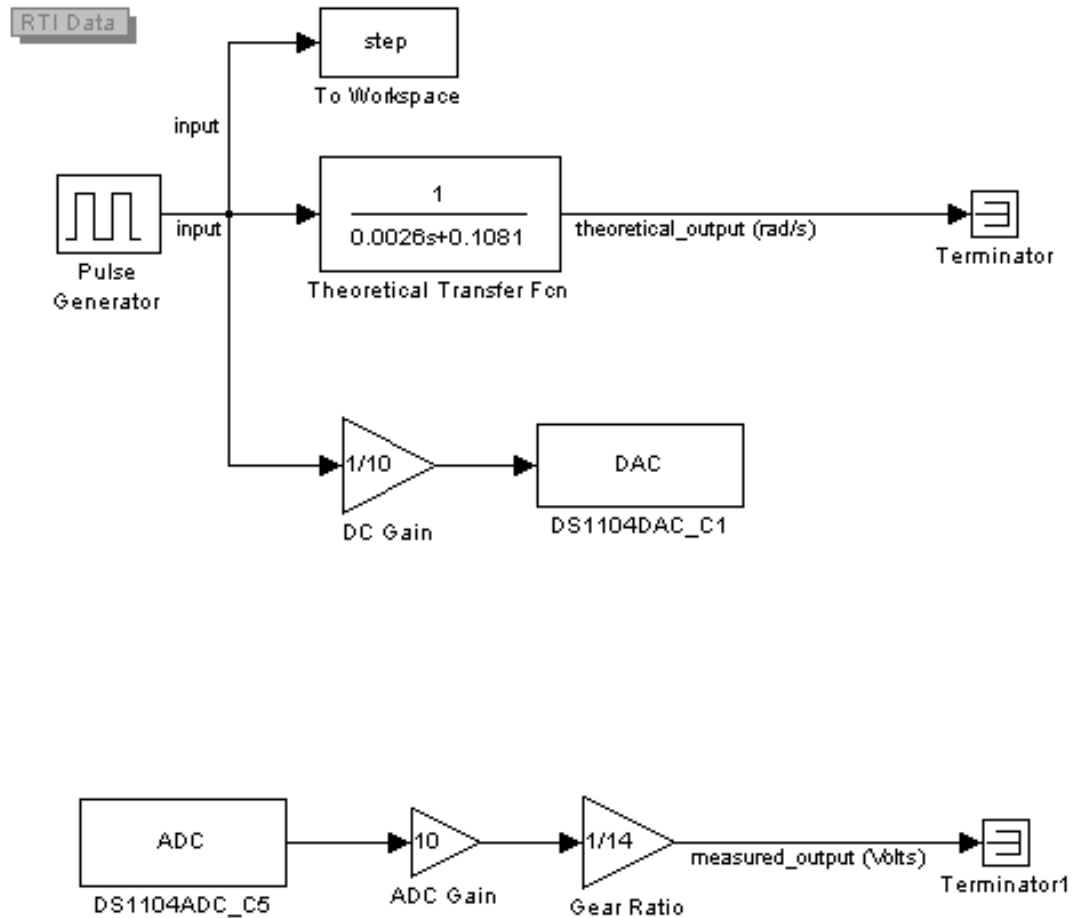


Figure 3.6 – Simulink Model



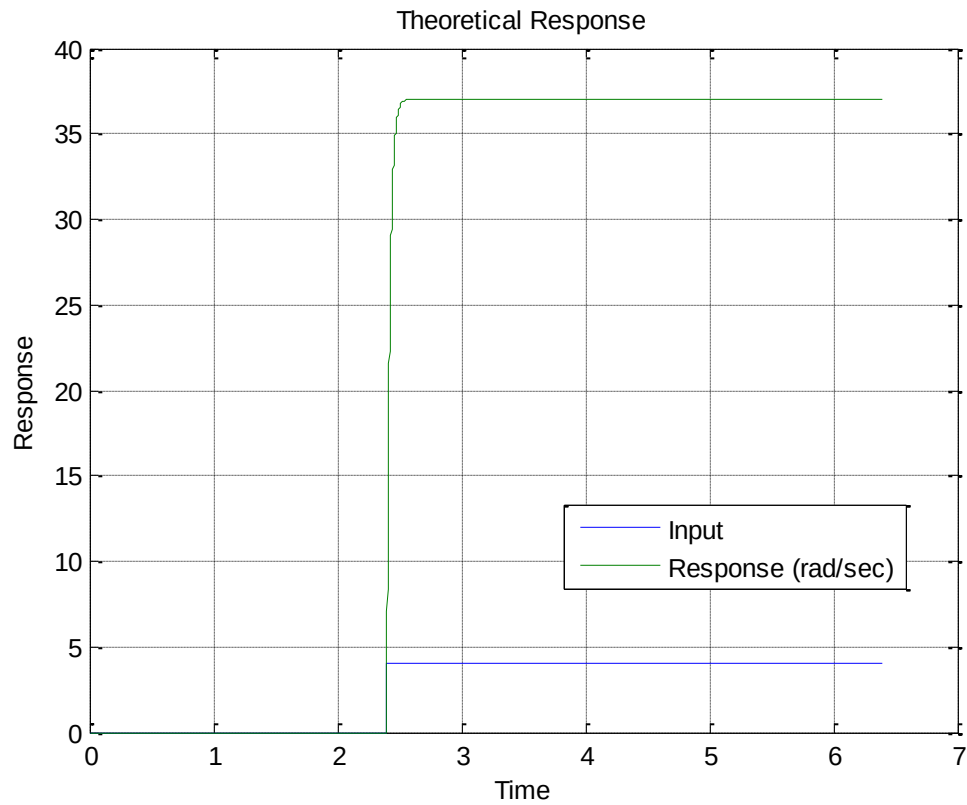
We replaced the step function with a pulse function, with a start time of $t = 4$ seconds and a final value of 4 Volts. The solver was set to 1 ms and simulation time to 8 seconds.

For the second part of the lab, the center gear of the motor was set to 0 degrees and an input time of 7 seconds. We then began reducing the times until the motor made only one revolution.

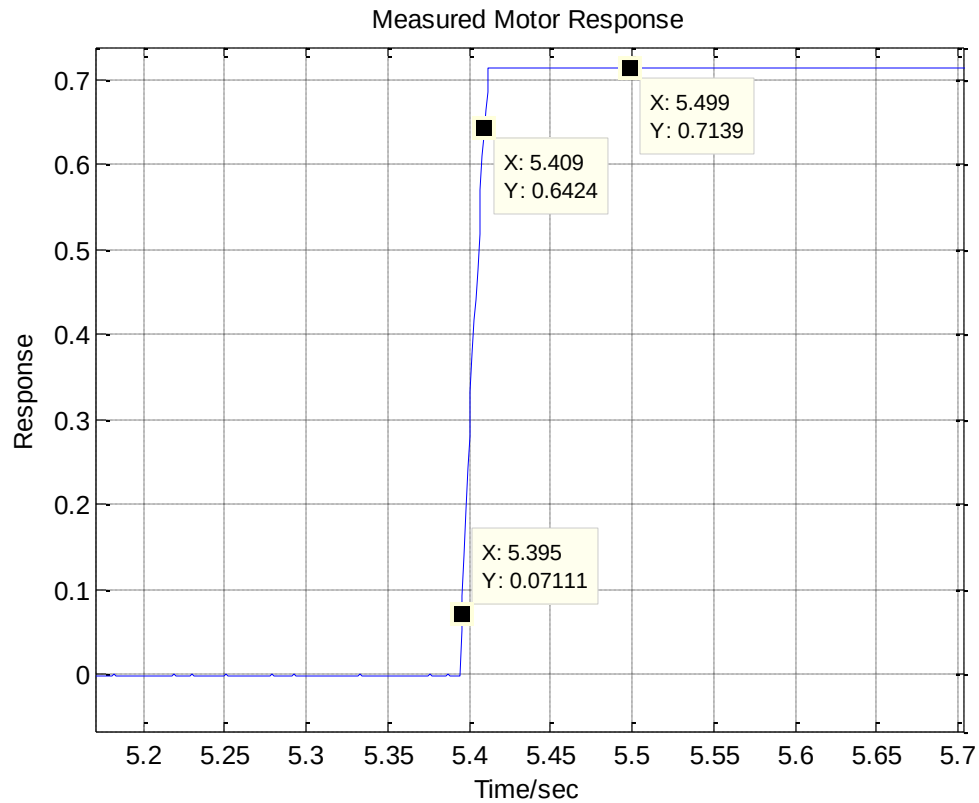
3. Results

For the first part of the lab, we theorized in Matlab the following resulting graph. The input to the system is a step function starts at $t = 4$ sec with *height 4 Volts*.

In the ControlDesk we obtained the response from both theoretical transfer function and the actual motor. Plots for both responses are as shown in the following two figures:

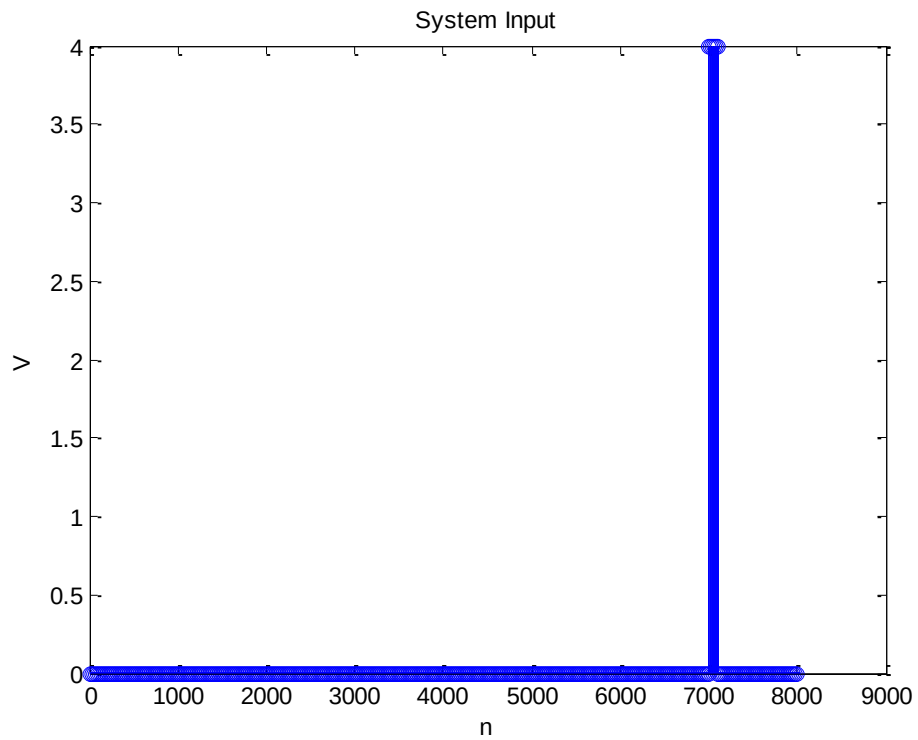


Plot 1. Theoretical response



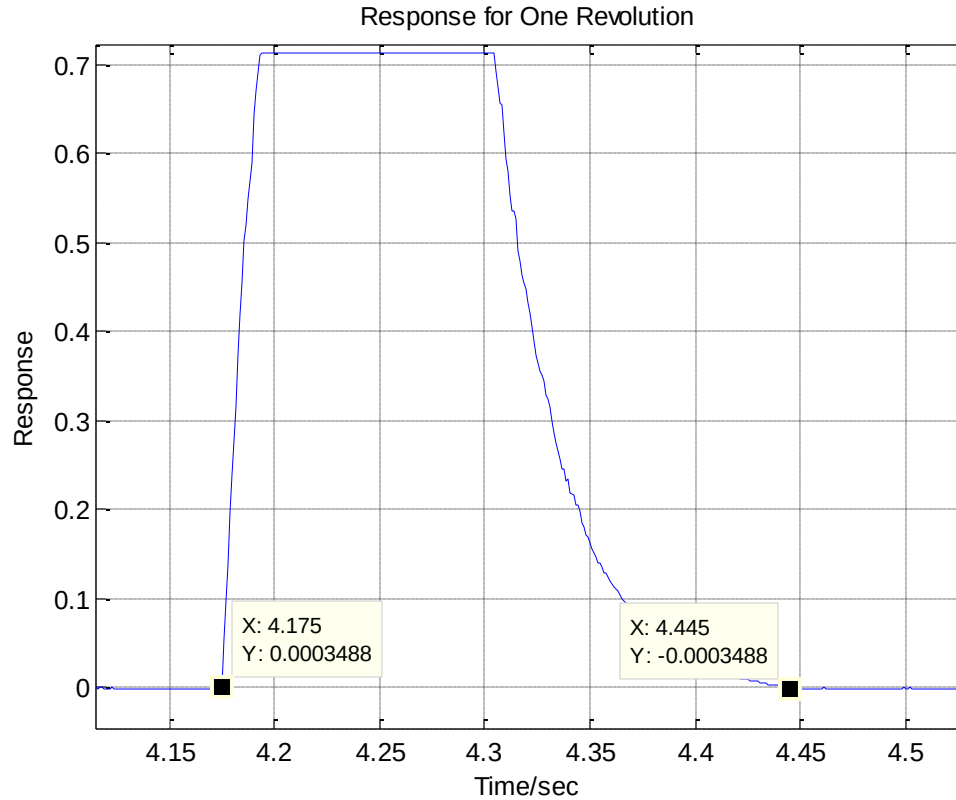
Plot 2. Theoretical response

Since the tachometer introduces its own gain, we want to know this extra gain in order to obtain the correct transfer function for the motor. For the second part of the lab, as we began to reduce our run time in order to achieve one rotation exactly, we theorized in Matlab the following resulting graph:



Plot 3. Input to the system resulting 1 rotation of motor

We measured in the ControlDesk the following graph:



Plot 4. Response from 1 rotation of motor

4. Discussion Questions

For our post lab exercise we were given the following tasks stated below:

3.5 Post-Laboratory Exercises

1. Using the acquired plot, compute the values of K and σ as described in Section 3.3.
Note: remember that the step input used was 4.
2. Compute the value of K_{tach} as follows:
 - Compute the average velocity during one revolution: $\omega_0 = \frac{2\pi}{t_{stop}}$.
 - Compute K_{tach} using the mean, v_{avg} computed earlier: $K_{tach} = \frac{v_{avg}}{\omega_0}$.
3. Compute the value of K_{motor} and write the motor transfer function. Compare this estimate with the model derived in the pre-lab.

Base on the general first-order transfer function model:

$$G(s) = \frac{K\sigma}{s + \sigma} \quad \text{where} \quad \sigma = \frac{\ln 9}{t_r}$$

$$t_r = 5.409 - 5.395 = 0.014 \text{ sec} \quad - \text{ rising time}$$

K – system gain

We found:

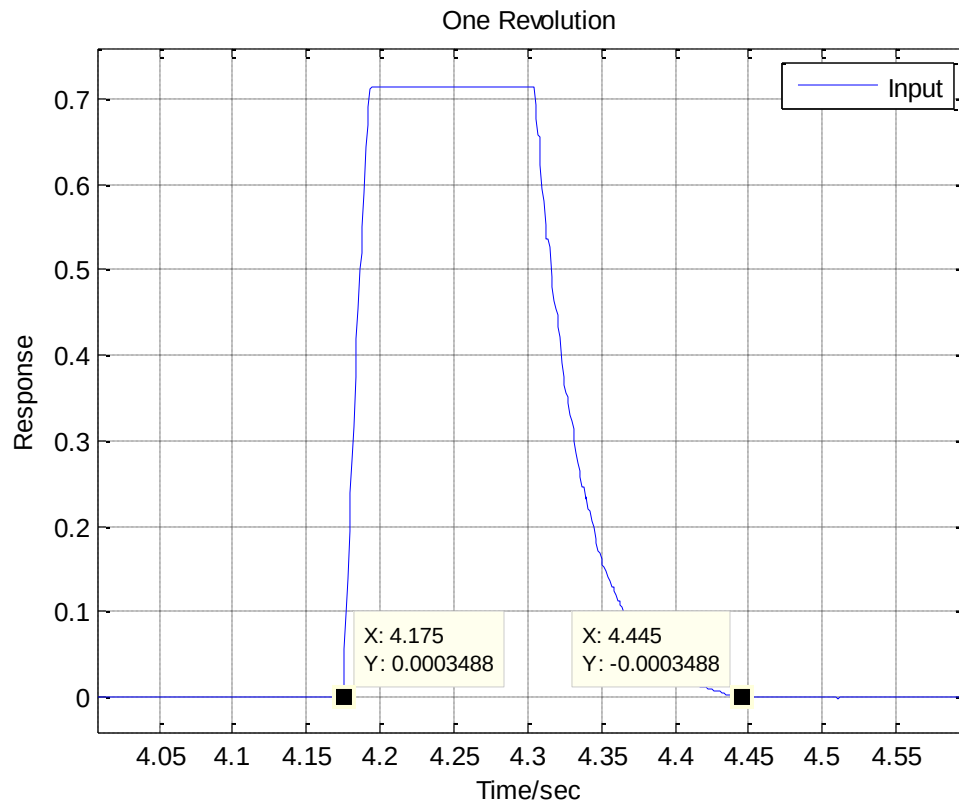
$$\sigma = \frac{\ln 9}{0.014} = 156.94$$

From plot 4, we found that $t_{stop} = 4.445 - 4.175 = 0.26 \text{ sec}$. By using the following MATLAB

command:

```
y2 = getfield(v.Y(2), 'Data');
y3 = find(y2 > 0.01)
mean(y2(y3))
```

we found $V_{ave} = 0.4379 \text{ V}$



$$\omega_0 = \frac{2\pi}{t_{stop}} = \frac{2\pi}{0.26} = 23.17 \text{ rad}$$

$$K_{tach} = \frac{V_{ave}}{\omega_0} = \frac{0.4379}{23.17} = 0.019$$

$$G(s) = \frac{K\sigma}{s + \sigma} = \frac{K_{input} K_{motor} K_{tach} \sigma}{s + \sigma}$$

$$K_{motor} = \frac{K}{K_{input} K_{tach}} = \frac{156.94}{4 \times 0.019} = 2065.06$$

From above results, the measured plant transfer function is

$$G(s) = \frac{2065.06 \times 156.94}{s + 156.94} = \frac{3.24 \times 10^5}{s + 156.94}$$

Theoretical plant transfer function found in prelab is:

$$G(s) = \frac{369.5}{s + 39.93}$$

The theoretical result and measured results deviate quite a lot.

5. Conclusions

In this lab we performed time-domain system identification for the DC servomotor and used time-domain system ID technique consisted of applying a pulse function input to the plant then measuring certain transient characteristics in order to estimate the transfer function. A design model was created and evaluated for accuracy. A state-space model and the corresponding transfer function were evaluated using stated parameters, Laplace transform, and s-domain reasoning. The value of K_{tach} (in V/rad/s) was determined.

By analyzing measured data plots we found that the actual plant transfer function is more like $G(s) = \frac{3.24 \times 10^5}{s + 156.94}$, which is quite different from the theoretical plant transfer

function. We are expecting to see big differences between measured and theoretical results in the coming labs.

ECE 557 Lab 4:

Gain Compensation and Feedback for a DC Servo

Members: Larissa Borecz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 2/1/12

Abstract: The purpose of this lab was to employ cascade gain compensation in a unity feedback configuration to adjust the damping of a closed loop system. The choice of the compensating gains was determined via computer-aided design using root locus techniques.

1. Introduction

The objective of this lab was to employ cascade gain compensation in a unity feedback configuration to adjust the damping of a closed loop system. We employed both the root locus and frequency response methods for designing gain compensation in a unity feedback configuration. The choice of the compensating gains was determined via computer-aided design using root locus techniques.

2. Procedure

For our experiment we used a closed loop unity feedback system like shown in Figure 4.2 with our given input transfer function, from input voltage to angular position of the servomotor.

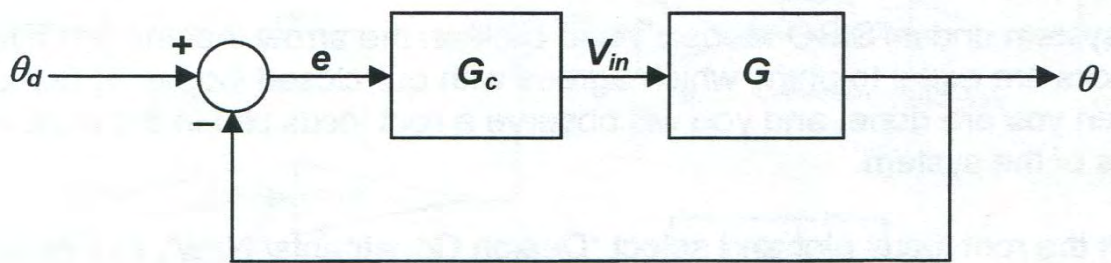


Figure 4.2 – Closed loop system

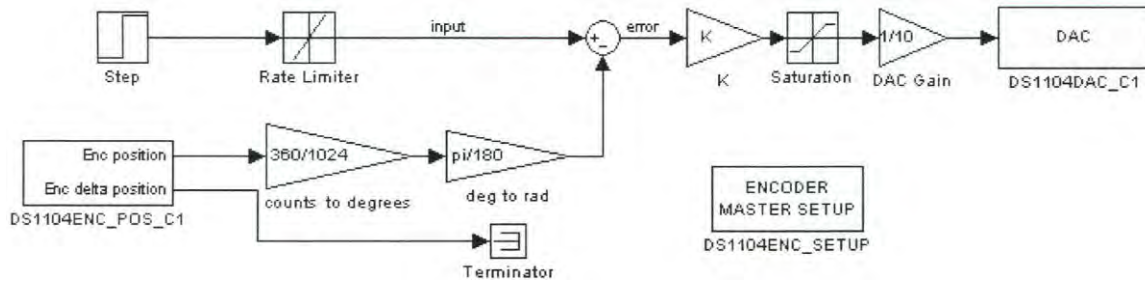
The following connections were used to connect the DC servomotor, the DS1104 interface board, and the computer:

4.4.1 Connections

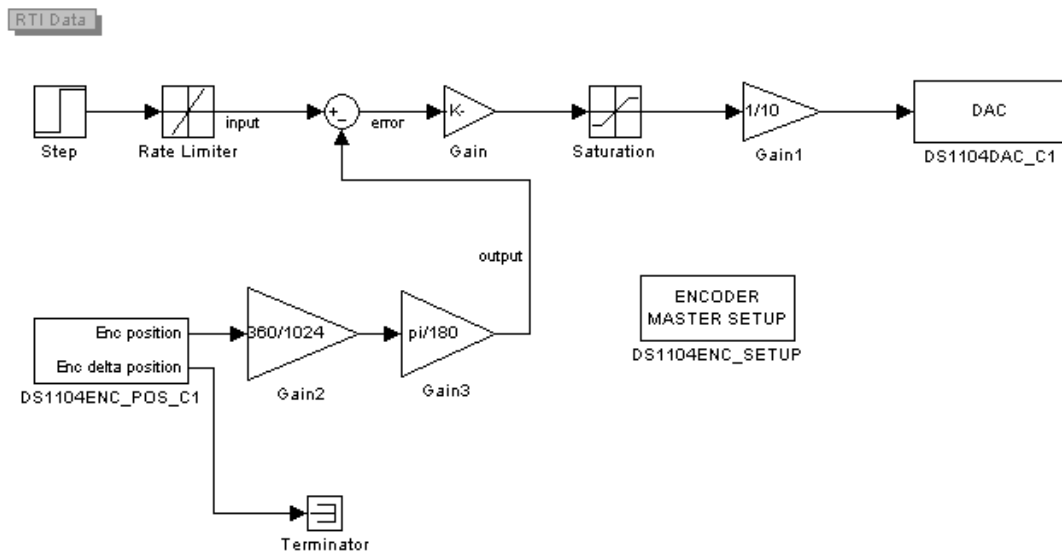
Since we are performing position control on the servomotor, we will need to use the encoder for position sensing. Also, remember that we are using the motor in low gear ratio, with an inertia added on the top. Use the following procedure:

- 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.

We constructed the given Simulink diagram, as seen in the below figure:



Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 30seconds, unit input step input start at $t = 10$, slew rates ± 1000 , saturation limits to $\pm 5V$, and revolutions of 360/1024.



The following ControlDesk setup was used to run and determine results:

4.4.3 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

3. Results

Generally, damped oscillators use the second-order differential equation:

$$\frac{d^2x}{dt^2} + 2\zeta\omega_o \frac{dx}{dt} + \omega_o^2 x = 0$$

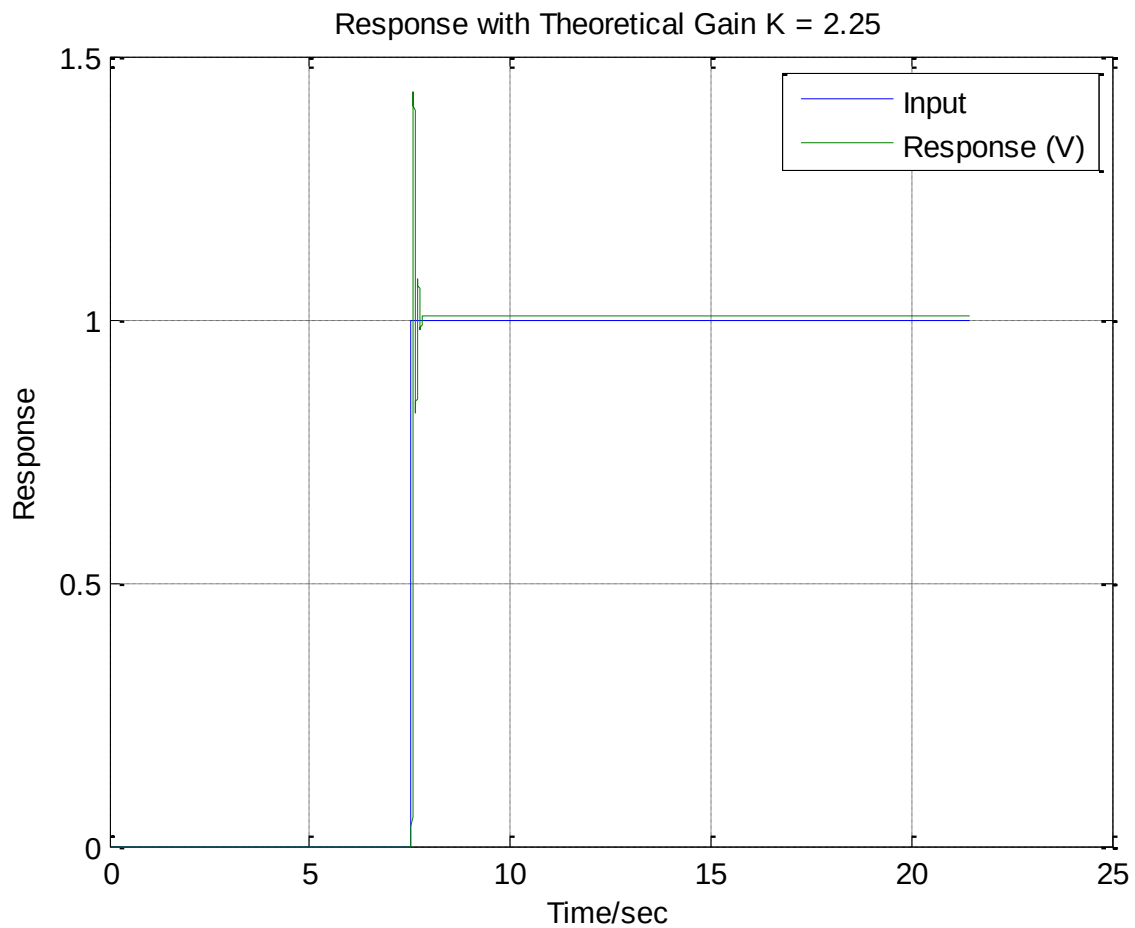
The value of the damping ratio ζ determines the behavior of the system. A damped harmonic oscillator can be:

- 1) Overdamped ($\zeta > 1$): The system returns exponentially decays to equilibrium without oscillating. Larger values of the damping ratio ζ return to equilibrium more slowly.
- 2) Critically damped ($\zeta = 1$): The system returns to equilibrium as quickly as possible without oscillating. This is often desired for the damping of systems such as doors.
- 3) Underdamped ($0 < \zeta < 1$): The system oscillates (at reduced frequency compared to the undamped case) with the amplitude gradually decreasing to zero.
- 4) Undamped ($\zeta = 0$): The system oscillates at its natural resonant frequency (ω_o).

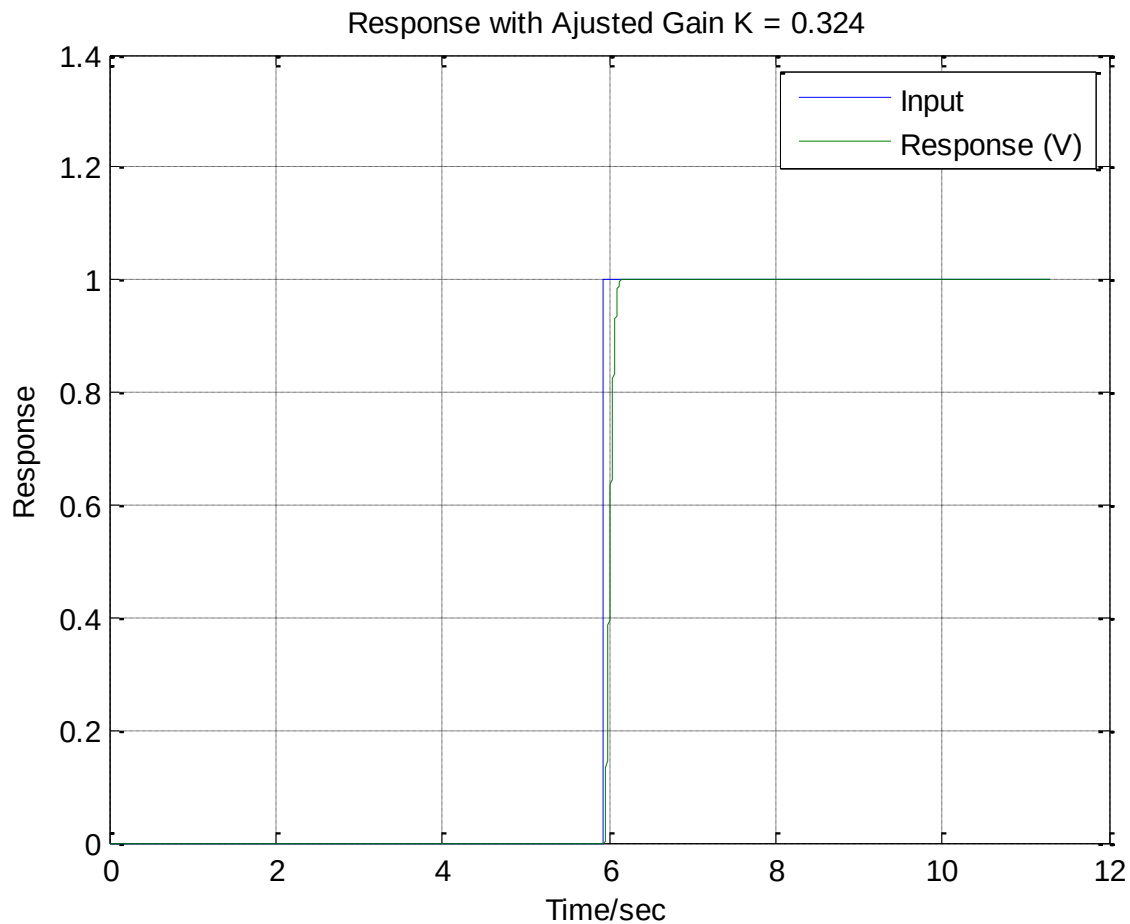
For our experiment we attempted to achieve a critically damped signal or an under-damped signal with less than 5% overshoot. If a system is critically damped the system

converges to zero as fast as possible without oscillating and has a double root which is real. When a system is over-damped there are two different real roots.

We begin our experiment we used the gain input K that we had calculated in the pre-lab of $K = 2.25$, which is shown in the below figure. It can be seen that this K is too large, and produces too much overshoot to be within our needed 5%.



Through experimentation with different K values, we achieved the critically damped system, seen the figure below. Our final value was $K = .324$:



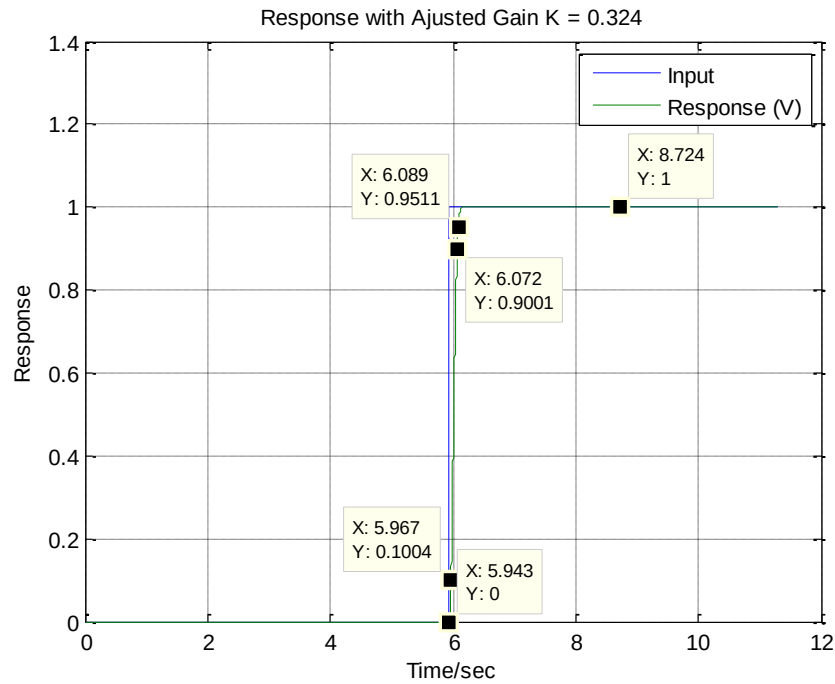
4. Discussion Questions

For our post lab exercise we were given the following tasks stated below:

4.5 Post-Laboratory Exercises

- From the saved data, evaluate the steady state error, rise time, settling time, percent overshoot.
- Compare those values with the results you obtained in the pre-laboratory assignment.
- What is the value of the gain you finally tuned in dSPACE? How did you tune this gain?
- Explain why the value of the gain is different than the one found in the pre-laboratory.

With $K = 0.324$:



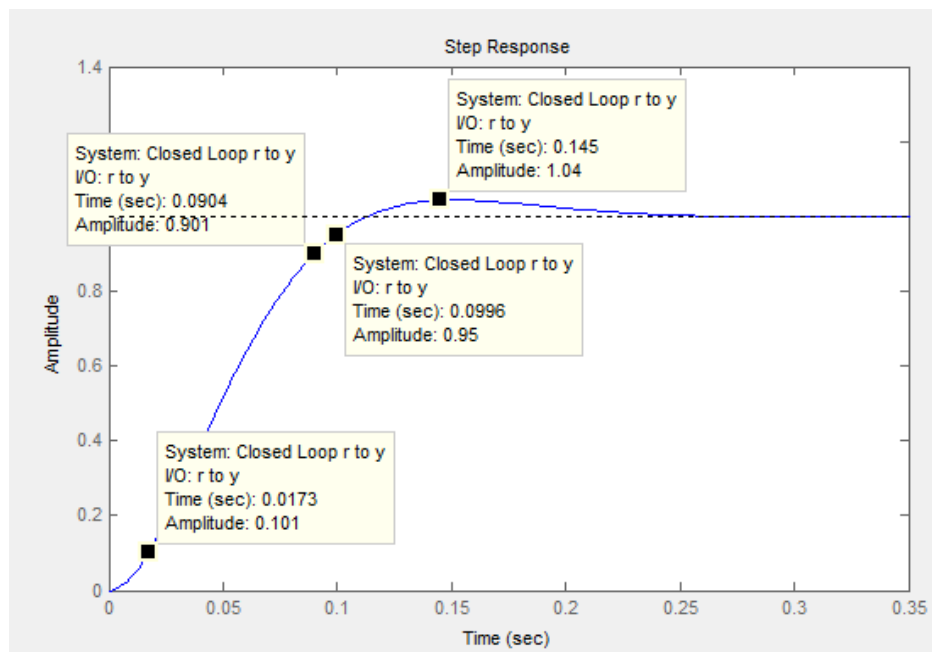
Steady state error = 0

Rise time = $6.072 - 5.967 = 0.105$ sec

Setting time (5%) = $6.089 - 5.943 = 0.146$ sec

Percentage overshoot = 0

The response obtained from theoretical model (prelab) is as shown below:



Steady state error = 0

Rise time = $0.0904 - 0.0173 = 0.0731$ sec

Settling time = 0.0904 sec

Percentage overshoot = 4%

Summary of the actual response and theoretical response is shown in the table below:

	Theory	Actual
Steady state error (V)	0	0
Rise time (sec)	0.0731	0.105
Settling time (sec)	0.0904	0.146
Percentage overshoot (%)	4	0

Only the actual steady state error matches with theory. Actual rise time, settling time and percentage overshoot are very different from theory.

The gain we found in Dspace is 0.324. We first started looking for gain value from $K = 2.25$ (theoretical). From the response we found that the percentage overshoot is almost 40%. Then we increased the gain till 5V, which is the limit, and no reduced percentage overshoot was observed. After that we start tuning down from 2.25 and found percentage overshoot decreased accordingly. The best result was obtained at 0.324, which we observe no steady state error and 0 percentages overshoot. Any tuning round this value will result in a steady state error.

The gain is dramatically different from the one found in prelab. Recall that from lab 3 we also obtained a transfer function which is very different from the theoretical transfer function. Therefore obtaining a different gain is reasonable.

5. Conclusions

For this experiment, the goal was to employ cascade gain compensation in a unity feedback configuration to adjust the damping of a closed loop system. We used a closed

loop unity feedback system, and utilized both the root locus and frequency response methods for designing gain compensation in a unity feedback configuration. The choice of the compensating gains was determined via computer-aided design using root locus techniques. We attempted to achieve less than 10% overshoot, and through experimentation we achieved a critically damped system with a final value of $K = .324$. This value is quite different from theoretical gain ($K = 2.25$). This is reasonable because the actual transfer function is quite different from theoretical transfer function.

ECE 557 Lab 5:

Lag Compensation for Speed Control of a DC Servo

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 2/8/12

Abstract: Utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lag compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor.

1. Introduction

The objective of this lab was to utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lag compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor. We employed bode design method and root locus design method in creating a lag compensator in a unity feedback configuration. The choice of the poles and values was determined via computer-aided design tools.

2. Procedure

For our experiment we used a general unity feedback configuration like shown in Figure 5.1 with our calculated compensator transfer function. The general compensation network $G_c(s)$ is cascaded with the plant $G(s)$ in order to provide a desired closed response.

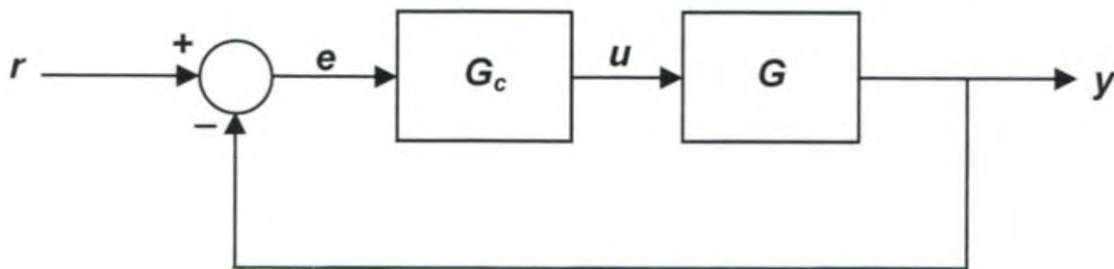
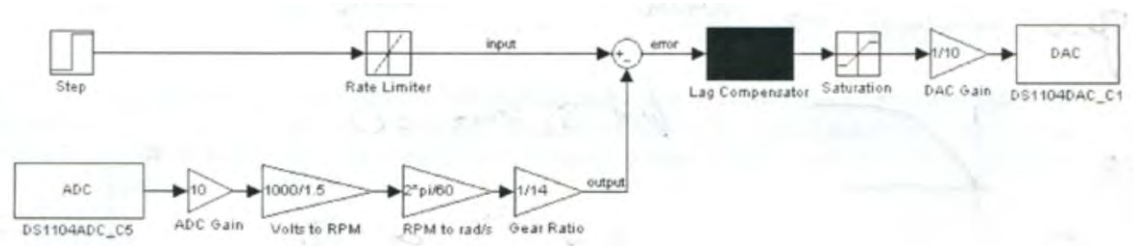


Figure 5.1: Cascade lag compensation closed loop system

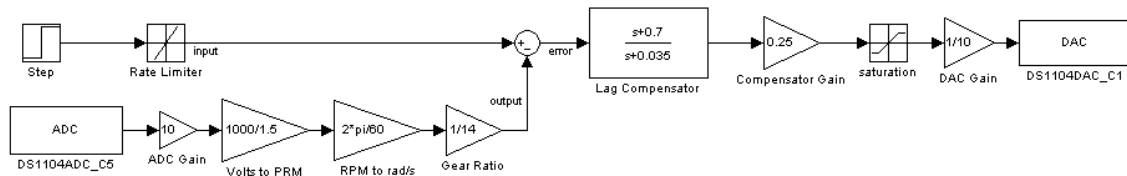
The following connections were used to connect the DC servomotor, the DS1104 interface board, and the computer:

- 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).

We constructed the given Simulink diagram, as seen in the below figure:



Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 15s, unit input step input start at $t = 4$, and have a final value of 4. The slew rate was set to ± 1000 , and saturation limits to $\pm 5V$. The lag compensator block was the transfer function of the compensator designed in the pre-lab, with both the bode design and root locus design methods implemented.

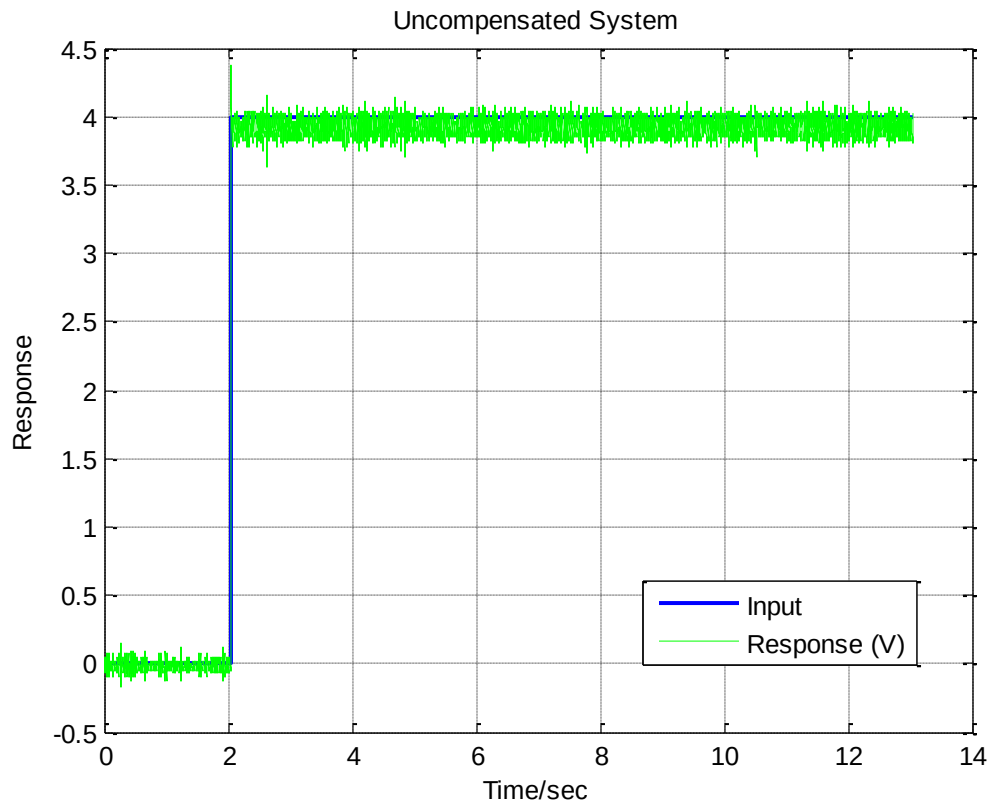


3. Results

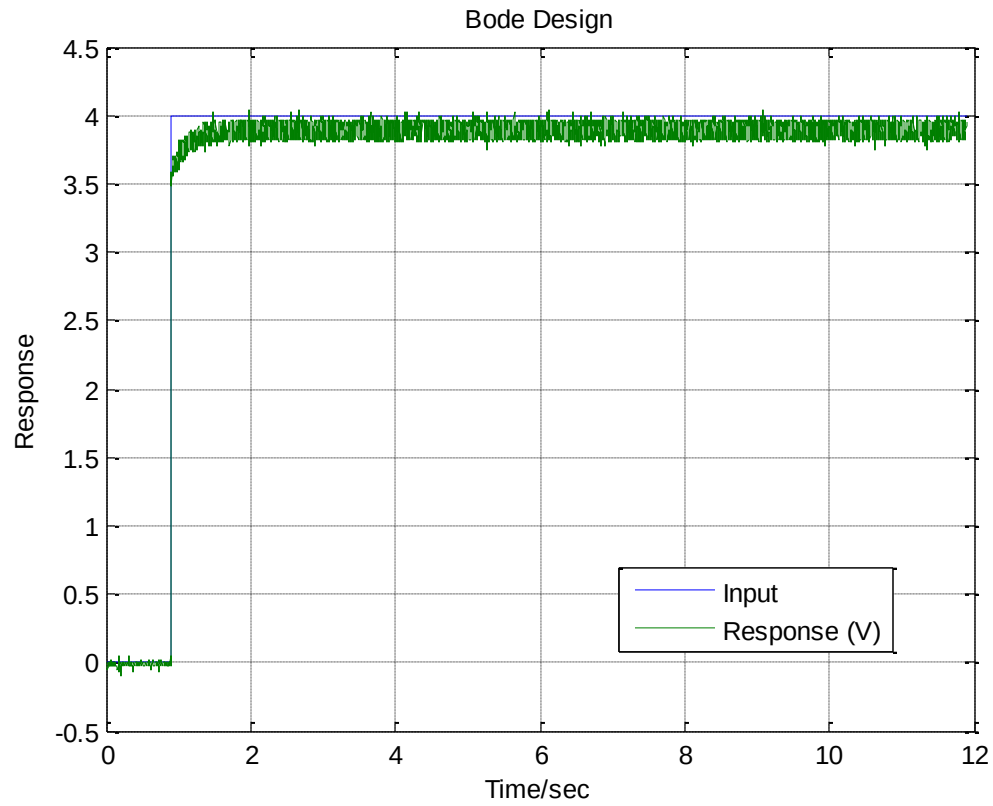
A lead-lag compensator is a component in a control system that improves an undesirable frequency response in a feedback and control system. It is a fundamental building block in classical control theory. A lag compensator introduces a pole-zero pair into the open loop transfer function with the zero usually being greater than the pole. The lag compensator provides phase lag at low frequencies, which reduces the steady state error. The effects of a lag compensator on the overall closed system performance are: decrease in bandwidth; increase in error constant and decrease in steady-state error; improvement in the gain or phase margin; a more sluggish system.

For our experiment we attempted to achieve a damped signal using the lag compensator that we designed in the pre-lab using the bode-method and root locus method.

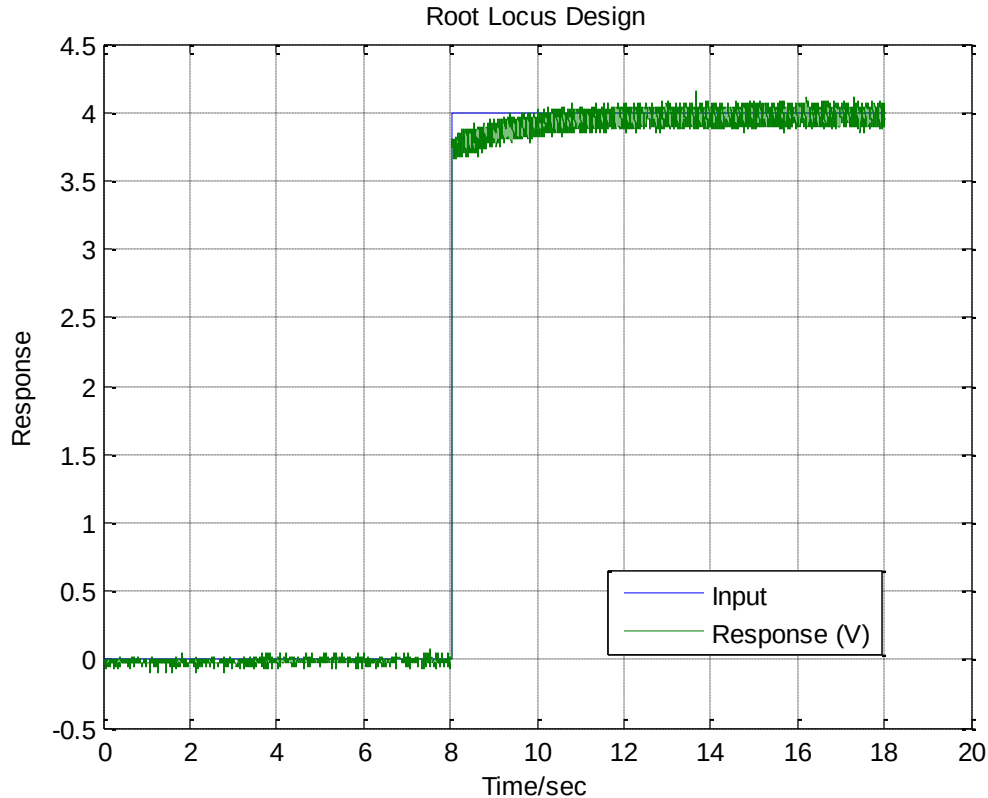
We begin our experiment by simulating the uncompensated system, seen in the figure below.



We then simulated the lag compensator designed using the bode method with the plant, which is shown in the below figure.



We lastly simulated the lag compensator designed using the root-locus method with the plant, which is shown in the below figure.



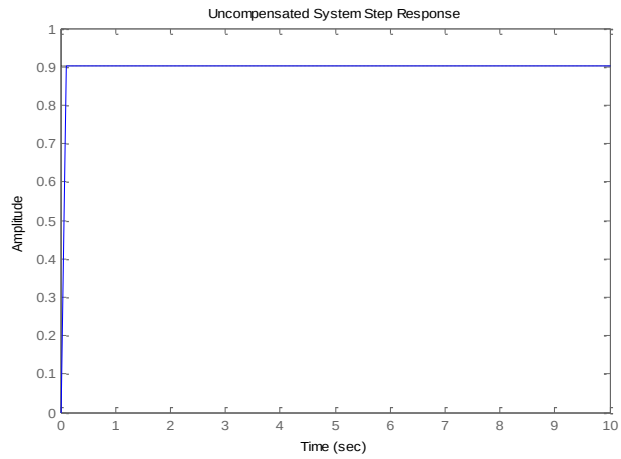
4. Discussion Questions

For our post lab exercise we were given the following tasks stated below:

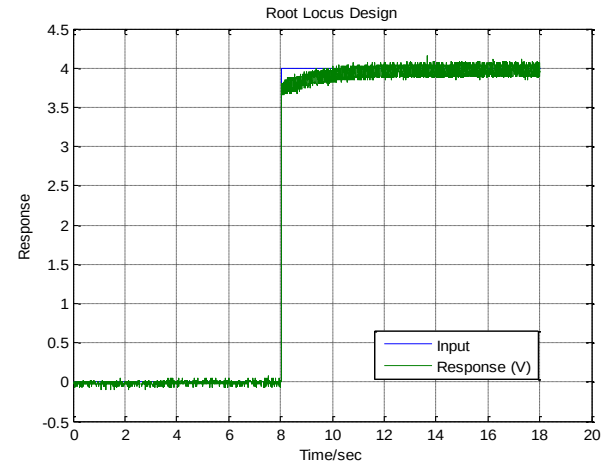
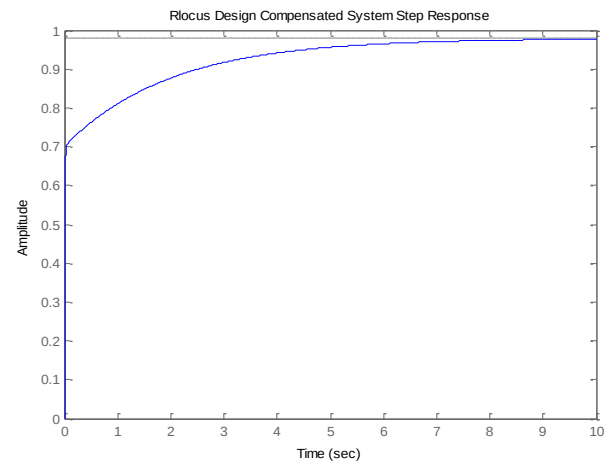
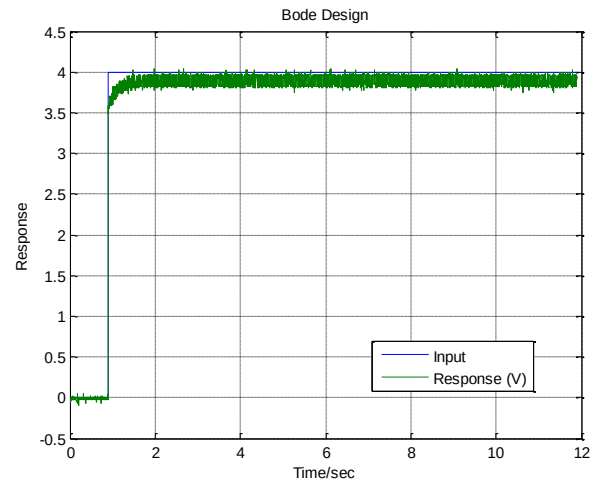
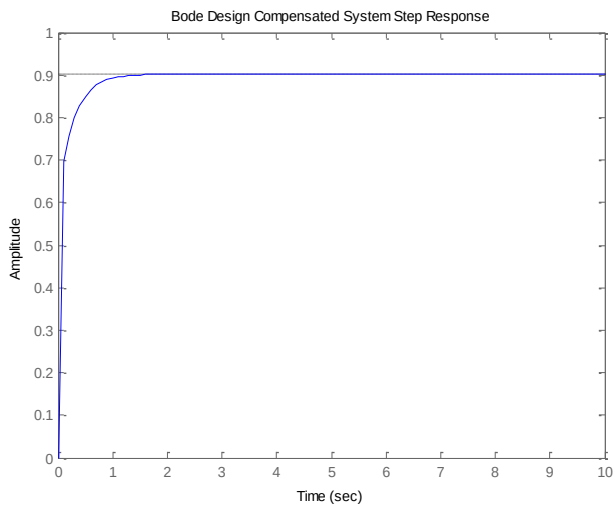
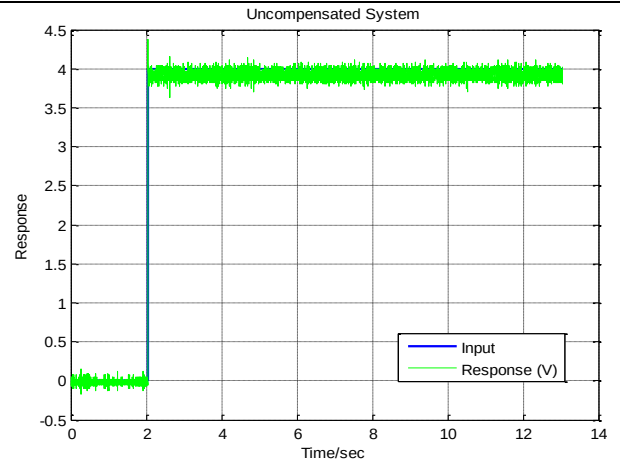
5.5 Post-Laboratory Exercises

- Plot the acquired data of both the Bode design method and the root locus method.
 - Compare those plots with the plots you obtained in the pre-laboratory assignment.
 - 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 4.2.3.
- 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.
- 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.

Theoretical



Measured



1. Theoretical and measured step responses of all three systems (uncompensated, bode designed and root locus designed) agree with each other to certain extent. The measured responses tend to have less steady state error compare with the theoretical responses. The major discrepancy is shown in the root locus design: the measured response reaches steady state in less than 3 seconds while the theoretical response takes about 10 seconds.
2. The bode design compensator we used was $G_c = \frac{0.167(s+4.96)}{s+0.83}$ and theoretical plant transfer function was $G_p = \frac{1}{0.0026s+0.1081}$.

Theoretically the steady state error is:

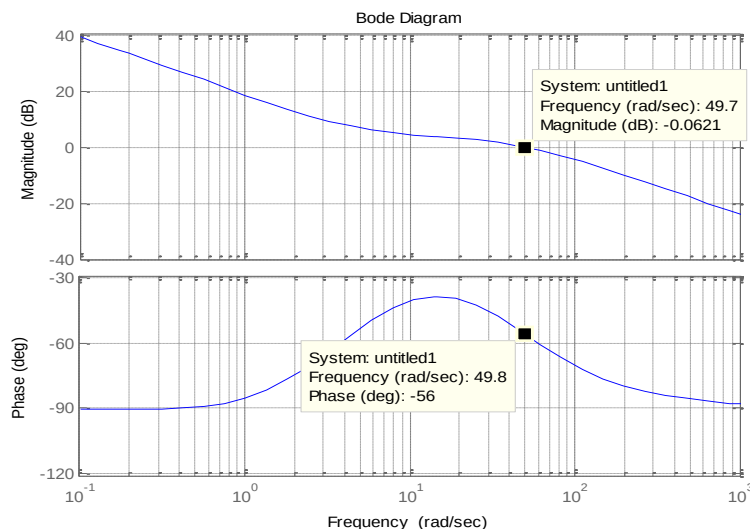
$$K_p = \lim_{s \rightarrow 0} G = \lim_{s \rightarrow 0} G_c G_p = \lim_{s \rightarrow 0} \frac{0.167(s+4.96)}{s+0.83} \frac{1}{0.0026s+0.1081} = \frac{0.167 \times 4.96}{0.83 \times 0.1081} = 9.23$$

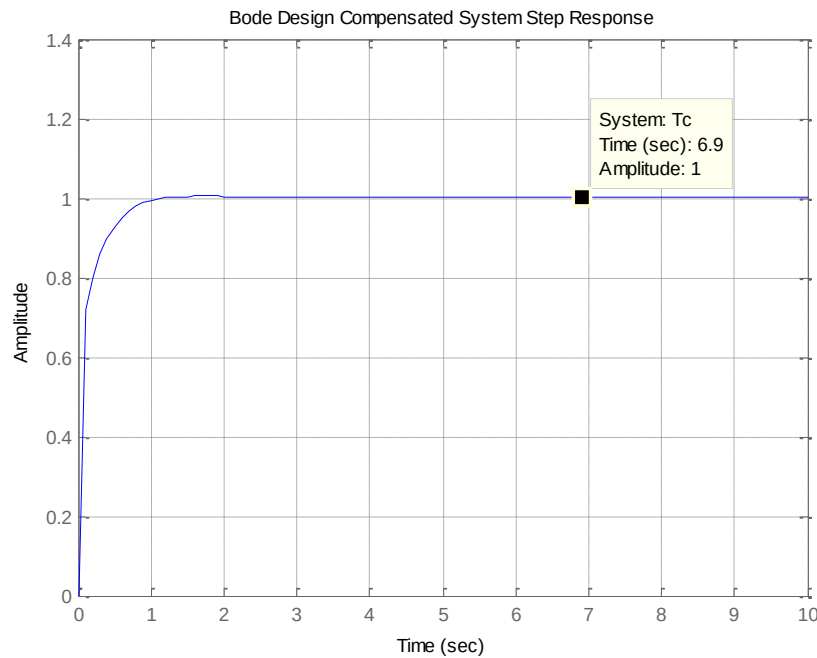
$$ess = \frac{1}{1+K_p} = 9.78\%$$

In order to have $ess = 0$, we need to let $K_p \rightarrow \infty$. One way to do this is to increase system type by 1 by adding a pole at origin. In order not to disturb the Bode plot too much to preserve phase margin, we add another zero near the origin. So a plausible design is like:

$$G_c = \frac{0.167(s+4.96)(s+1)}{s(s+0.83)}$$

Theoretical bode plot and response for this new Bode design is shown below:





The resulting response shows little overshoot with steady state error 0. Phase margin of new design is $180-56=124$ degree.

3. The design we adopted is a lag compensator: $G_c = 0.25 \frac{s+0.7}{s+0.035}$. The compensator introduce a pole very near to zero $p = -0.035$. The time response therefore is added a component $\exp(-0.035t)$, which is a very slow component. Therefore the system response is slow in spite of the gain ($G=5$) we adopted.

5. Conclusions

For this experiment, the goal was to utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lag compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor. We employed bode design method and root locus design method in creating a lag compensator in a unity feedback configuration. The choice of the poles and values was determined via computer-aided design tools. We attempted to

achieve a critically damped signal or an over-damped signal with our designed lag compensator, and we able to confirm both of our calculated transfer functions.

ECE 557 Lab 6:
Lead Compensation for Speed Control of a DC Servo

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 2/16/12

Abstract: Utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lead compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor.

1. Introduction

The objective of this lab was to utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lead compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor. We employed bode design method and root locus design method in creating a lead compensator in a unity feedback configuration. The choice of the poles and values was determined via computer-aided design tools.

2. Procedure

For our experiment we used a general unity feedback configuration like shown in Figure 6.1 with our calculated compensator transfer function. The general compensation network $G_c(s)$ is cascaded with the plant $G(s)$ in order to provide a desired closed response.

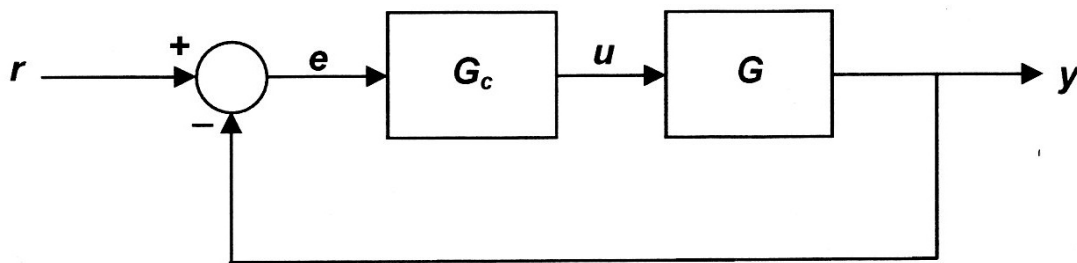


Figure 6.1 – Cascade Lead Compensation Closed-Loop System

The following connections were used to connect the DC servomotor, the DS1104 interface board, and the computer:

6.4.1 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:

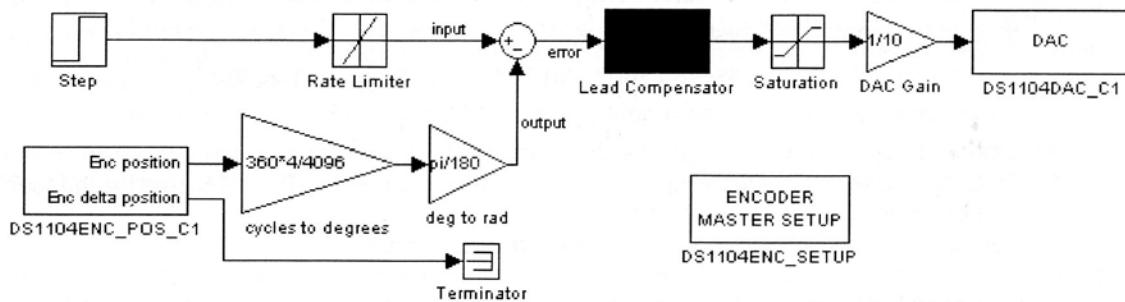
- Connect the UPM "To Load" port to the motor input.
- Connect the UPM "From D/A" port to analog output 0.
- Connect the encoder port of the motor (see Fig. 3.3) to the ENC1 on the interface board.

The following setup was used to run the ControlDesk and obtain results.

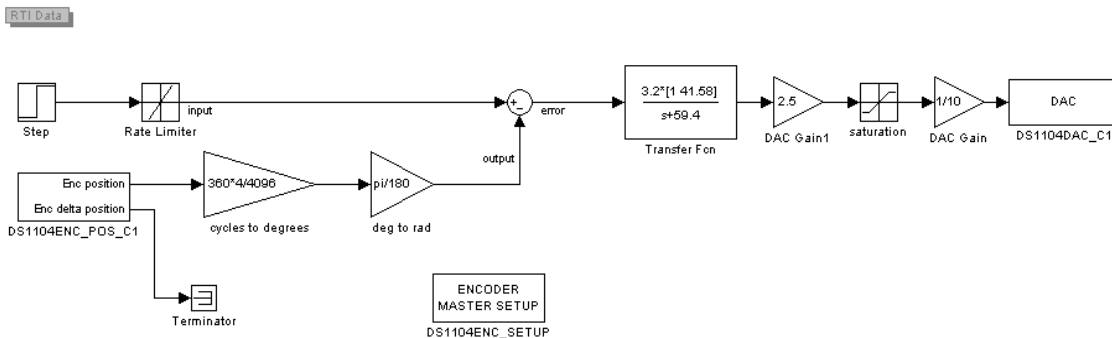
6.4.3 ControlDesk and Results

- 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

We constructed the given Simulink diagram, as seen in the below figure:



Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 30s, unit input step input start at $t = 10$, and have a final value of 10. The slew rate was set to ± 1000 , and saturation limits to $\pm 5V$. The lead compensator block was the transfer function of the compensator designed in the pre-lab, with both the bode design and root locus design methods implemented.



The transfer function block is for designed system compensator. In this experiment we tested two different compensators designed using Bode method and root locus method respectively.

Bode design compensator is

$$G_c(s) = 1.85 \frac{s + 4.97}{s + 9.17}$$

Root locus design compensator is

$$G_c(s) = 3.2 \frac{s + 41.58}{s + 59.4}$$

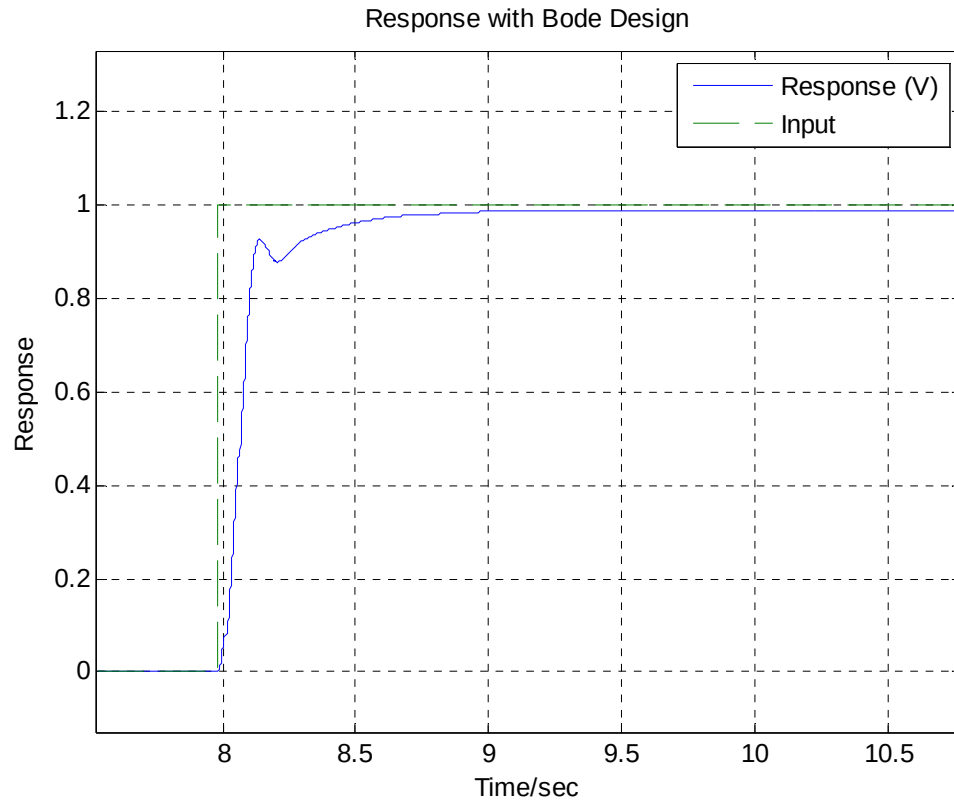
For the root locus design we can also cancel the pole at the origin. However, as mentioned in prelab, cancelling the pole at origin makes the compensator pole and zero very close to each other, which imposes a stricter requirement for the circuit components in order to function properly.

3. Results

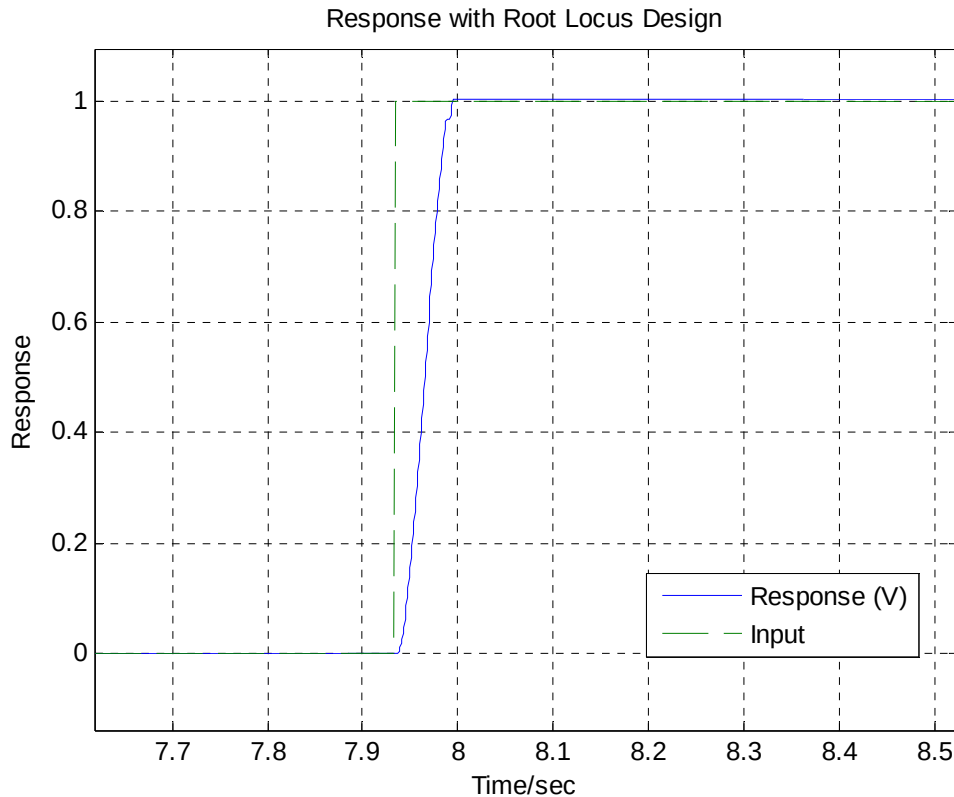
A lead-lag compensator is a component in a control system that improves an undesirable frequency response in a feedback and control system. It is a fundamental building block in classical control theory. A lead compensator introduces a pole-zero pair into the open loop transfer function with the zero usually being greater than the pole. The lead compensator provides phase lead at high frequencies, which enhances the responsiveness and stability of the system. The effects of a lag compensator on the overall closed system performance are: increase in bandwidth; decrease in error constant; improvement in the gain or phase margin; a faster system.

For our experiment we attempted to achieve a damped signal using the lead compensator that we designed in the pre-lab using the bode-method and root locus method.

We simulated the lead compensator designed using the bode method with the plant, which is shown in the below figure.



We lastly simulated the lead compensator designed using the root-locus method with the plant, which is shown in the below figure.

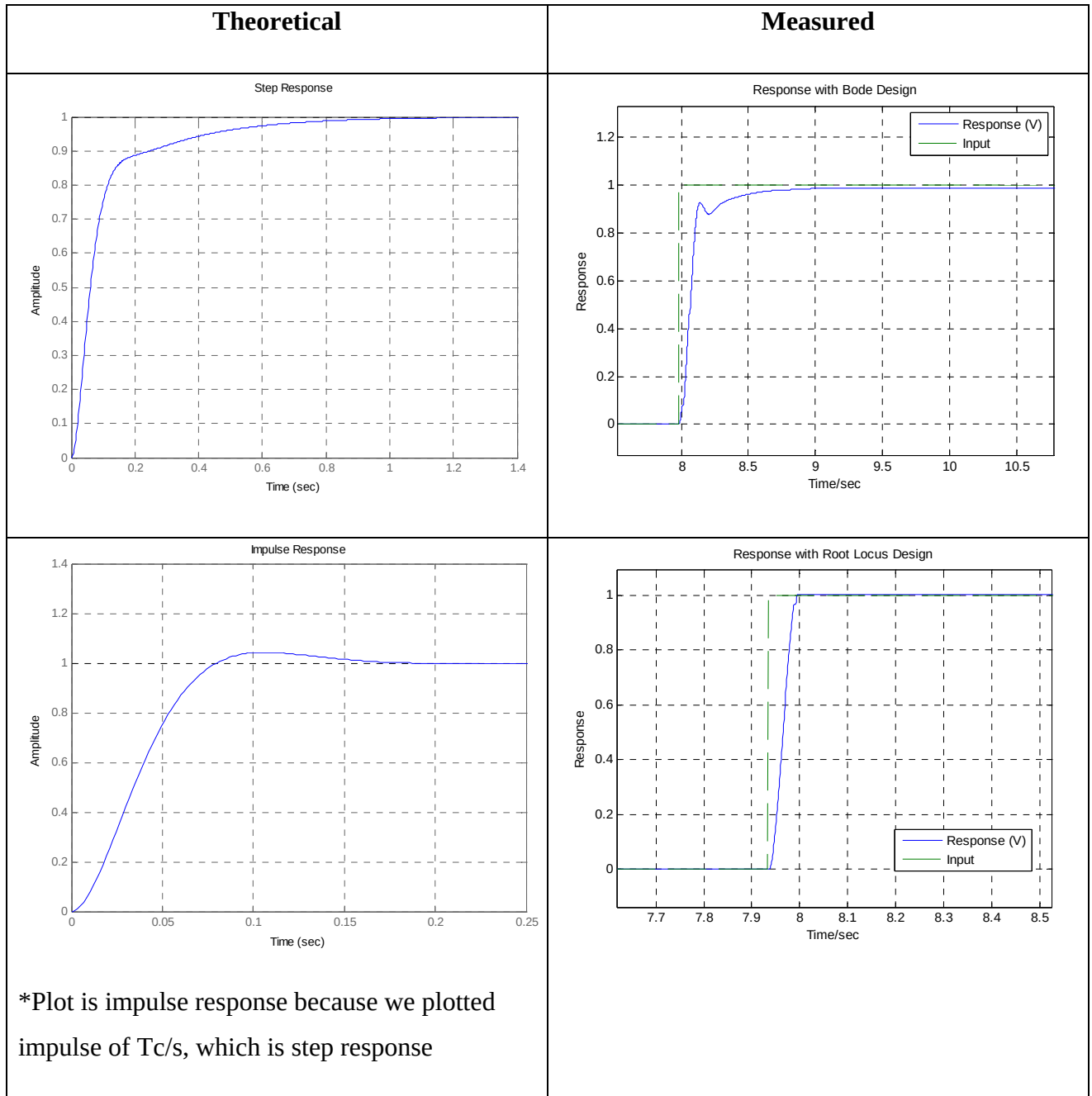


4. Discussion Questions

For our post lab exercise we were given the following tasks stated below:

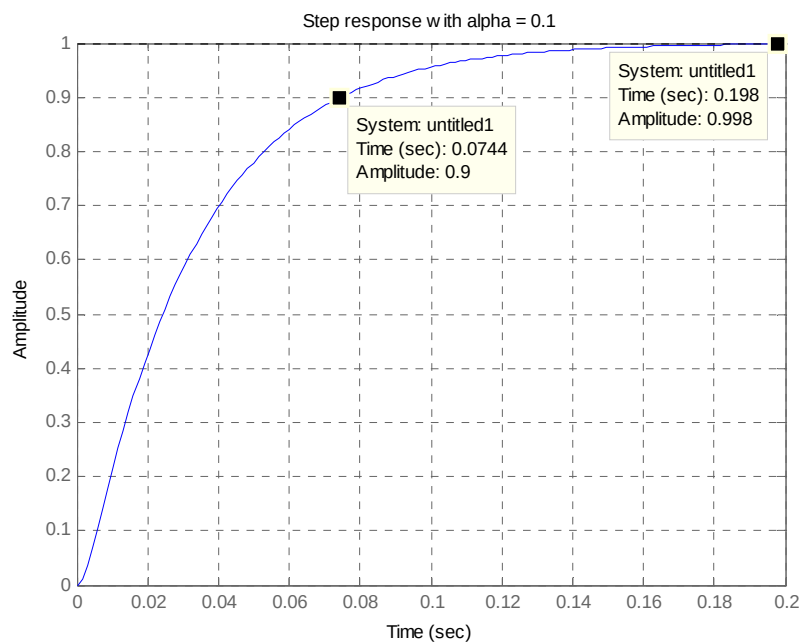
6.5 Post-Laboratory Exercises

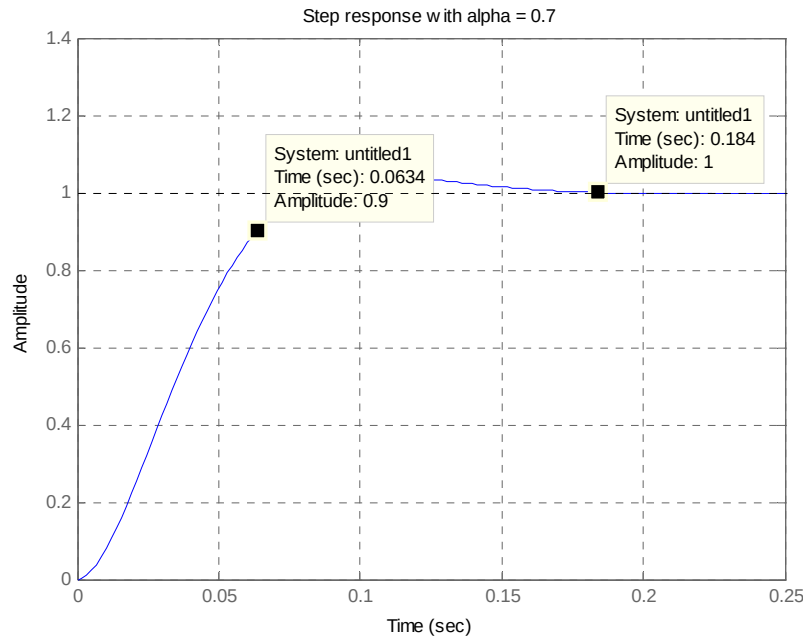
- Plot the acquired data of both the Bode design method and the root locus method. Show steady state error, rise time, settling time and overshoot on each plot.
- Compare those values with those of the plots you obtained in the pre-laboratory assignment.
- In the root locus design procedure, we picked $\alpha = 0.7$; 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.



Compare the theoretical and measured result, bode design shows almost exactly the same behavior. The measured response of root locus showing similar rise time behavior but much less percentage overshoots. In general, theoretical results and measured results match.

In the root locus design we selected $\alpha = \text{zero/pole} = 0.7$. If we select $\alpha = 0.1$, this means that the pole is further away from the zero also further from the origin. We expect to see that the system will response slower because the new pole location contributes a larger time constant. Also at a fixed gain the complex pole locations will be closer to the real axis because the root locus is stretched further. We run the root locus design MATLAB code again (which has a fixed zero cancels pole at -41.58 and compensator gain 3.2) with new $\alpha = 0.1$ and other quantity unchanged. Both results are shown below:





Compare the two results we can see with $\alpha = 0.1$ response is about 0.01 second slower time to reach 90% of final value and about 0.2 seconds slower to reach steady state. In addition with a smaller α , the response is less oscillatory.

5. Conclusions

For this experiment, the goal was to utilize computer aided design tools with techniques from Bode and root locus methods, to design and subsequently implement lead compensation in a unity feedback configuration in order to perform speed control on the SRV-20 DC servomotor. We employed bode design method and root locus design method in creating a lead compensator in a unity feedback configuration. The choice of the poles and values was determined via computer-aided design tools. We attempted to achieve a damped with our designed lead compensator, and we able to confirm both of our calculated transfer functions.

ECE 557 Lab 7:

Tuning a PID controller

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 2/16/12

Abstract: The objective of this laboratory was to investigate the proportional-integral-derivative (PID) type of control law, and model-based tuning rules for designing a PID controller to perform position control of the SRV-02 DC motor with modeled and un-modeled dynamics.

1. Introduction

The objective of this laboratory was to investigate the proportional-integral-derivative (PID) type of control law, and model-based tuning rules for designing a PID controller to perform position control of the SRV-02 DC motor with modeled and un-modeled dynamics. We used the PID design method in creating a compensator in a unity feedback configuration. The choice of the poles and values was determined first by hand, then tweaked via computer-aided design tools.

2. Procedure

For our experiment we used a general unity feedback configuration in cascade configuration like shown in Figure 7.1 with our tweaked compensator transfer function. The general compensation network $G_c(s)$ is cascaded with the plant $G(s)$ in order to provide a desired closed response.

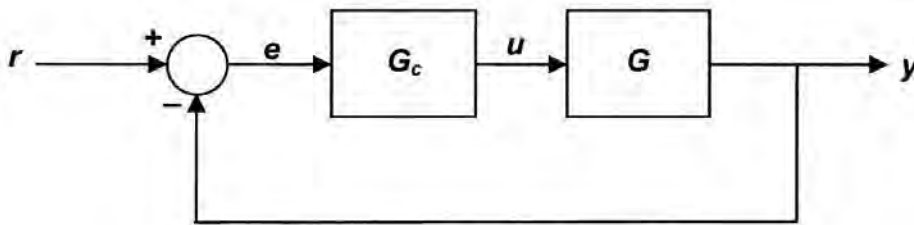
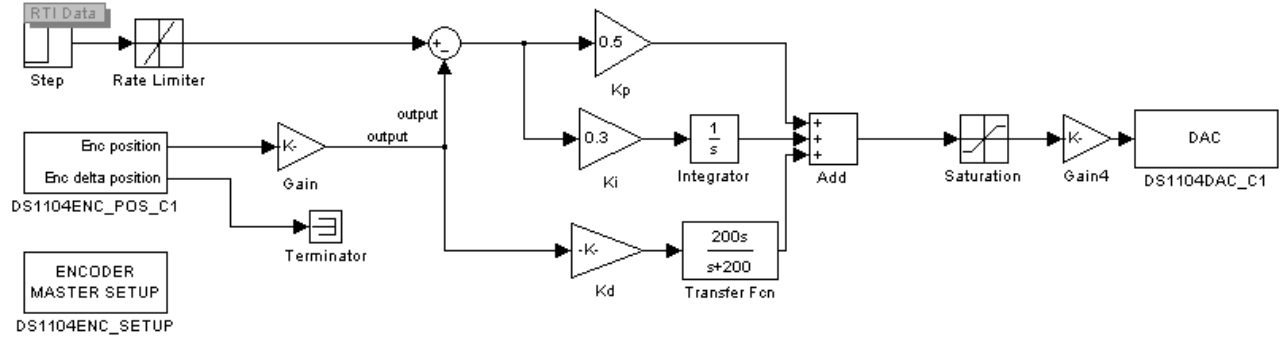


Figure 7.1 – Cascade PID Compensation Closed-Loop System

We used the connections stated in the lab manual procedure section 7.4.1 and 7.4.2 to connect the DC servomotor, the DS1104 interface board, and the computer. We used the setup and procedure stated in the lab manual procedure section 7.4.3 to run the ControlDesk and obtain results.

Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 30s, unit input step input start at $t = 10$, and have a final value of 10. The slew rate was set to ± 1000 , and saturation limits to $\pm 5V$. The compensator block K 's values designed in the pre-lab was then used to start the lab, then we tuned the values during the lab to get our desired response.



3. Results

A compensator is a component in a control system that improves an undesirable frequency response in a feedback and control system. It is a fundamental building block in classical control theory. The PID controller attempts to minimize the error by adjusting the process control inputs. The PID controller uses three separate constant parameters: the proportional (P), the integral (I), and derivative (D) values. These values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. The weighted sum of these three actions is used to adjust the response. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set-point and the degree of system oscillation. The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is: $u(t) = K_p * e(t) + K_i * \int_0^t e(\tau) d\tau + K_d \left(\frac{d}{dt} \right) * e(t)$, where: K_p = proportional gain (a tuning parameter), K_i = integral gain (a tuning parameter), K_d = derivative gain (a tuning parameter), e = Error = $SP - PV$, t = time or instantaneous time (the present).

In order to solve for the compensator, we used the PID pole zero cancellation method using the equations below:

$$\frac{V_{in}(s)}{E(s)} = K * \left(\frac{s^2 + 2 * \zeta * \omega_n * s + \omega_n^2}{s} \right), \quad \frac{U(s)}{E(s)} = G_c(s) = K_p + \frac{K_i}{s} + K_d * s$$

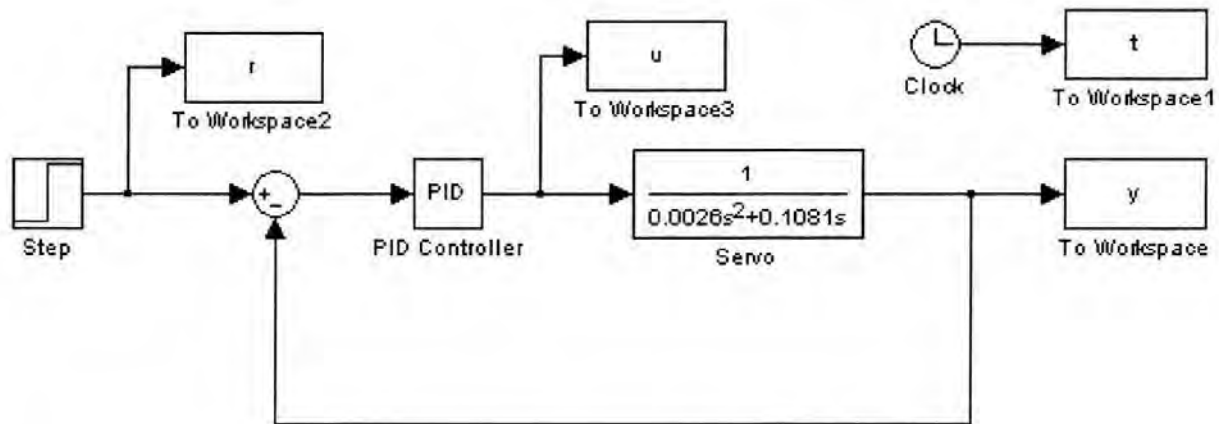
As solved for in the pre-lab, the PID controller equation can be written in this form:

$$G_c(s) = K_d \left(\frac{s^2 + \left(\frac{K_p}{K_d}\right)s + \left(\frac{K_i}{K_d}\right)}{s} \right), \quad \text{Using: } \frac{K_i}{K_d} = \omega_n^2, \quad \frac{K_p}{K_d} = 2 * \zeta * \omega_n$$

We were given $\zeta = 1/\sqrt{2}$ and $\omega_n = 50$ rad/sec to begin with to design our PID controller in the pre-lab. This gave us starting values of:

$$\frac{K_i}{K_d} = \omega_n^2 = 50^2 = 2500, \quad \frac{K_p}{K_d} = 2 * \zeta * \omega_n = 2 * .7071 * 50 = 70.710678 \approx 70$$

We used the basic Simulink diagram, seen below, to simulate our theoretical PID controller and tune it to an initial value u of +/- 5V and less than 2% overshoot.



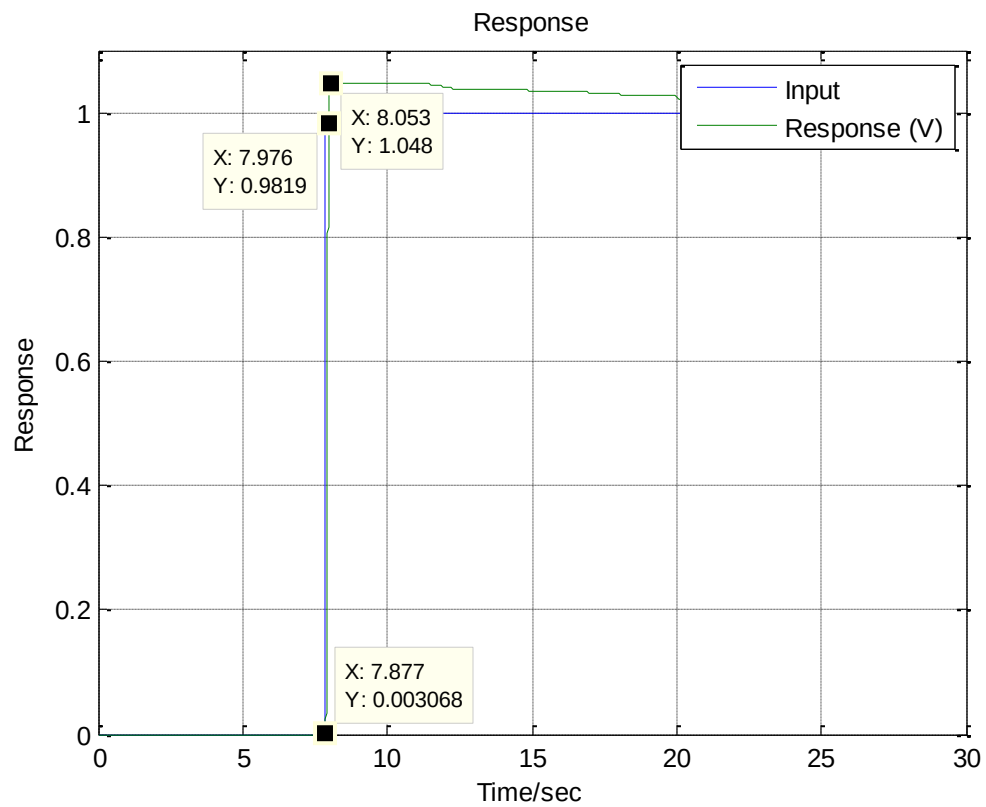
For our experiment we attempted to achieve a compensated signal using the compensator design that we tuned in the pre-lab. We achieved less than 2% overshoot, less than .5 sec settling time, and an initial value u of less than 5V. We simulated the compensator designed during pre-lab and tuned it to achieve the wanted response. During lab we then performed several iterations to tune the actual motor, the results seen below:

Iteration #1) Prelab values

$K_p = 0.8$; $K_i = 0.1$; $K_d = 0.01$

Percentage overshoot: 4.8%

Setting time: 0.099 sec

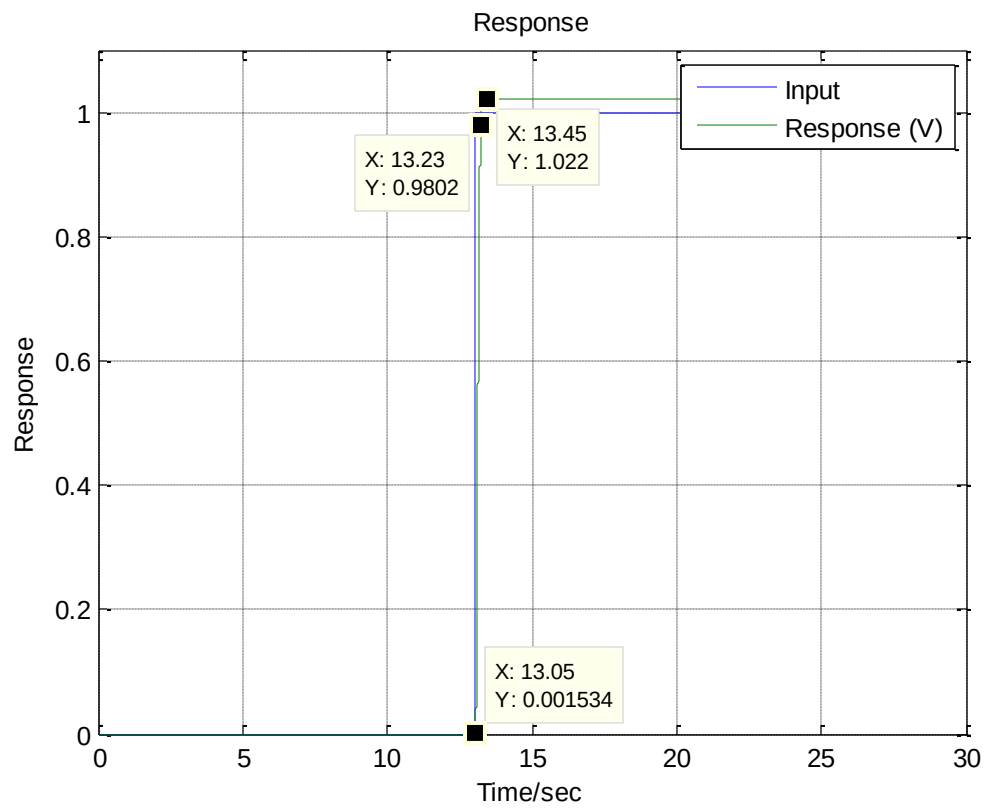


Iteration #2)

$K_p = 0.5$; $K_i = 0.3$; $K_d = 0.01$

Percentage overshoot: 2.2%

Setting time: 0.18 sec

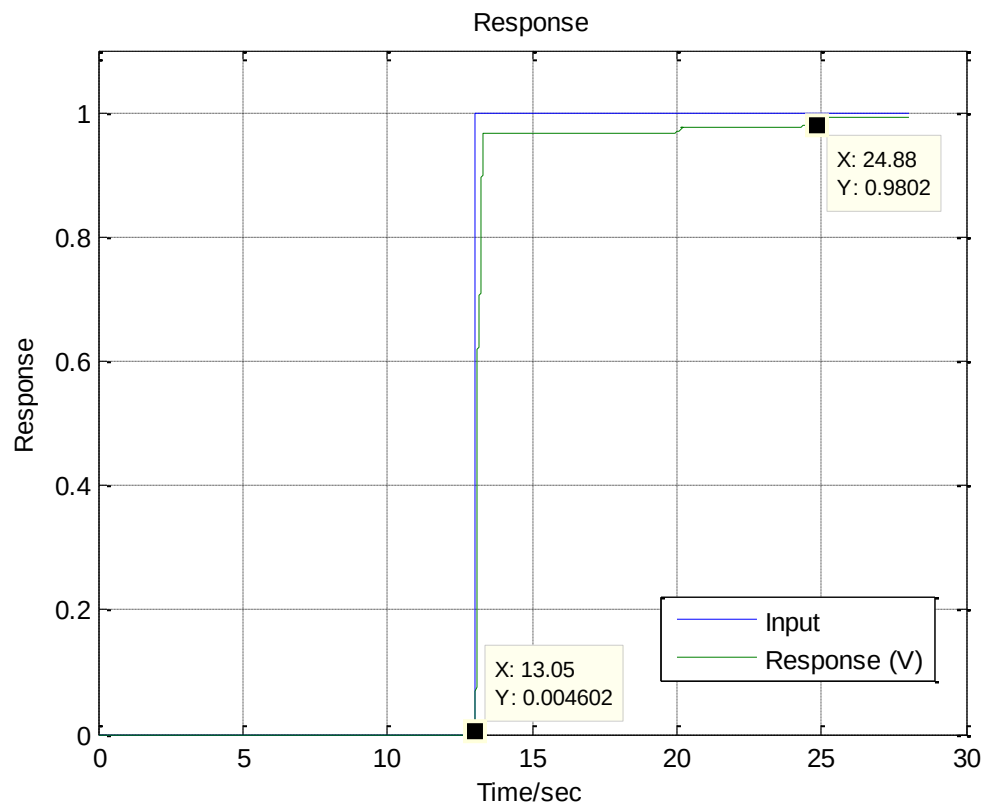


Iteration #3)

$K_p = 0.5$; $K_i = 0.1$; $K_d = 0.01$

Percentage overshoot: 0

Setting time: 1.83 sec

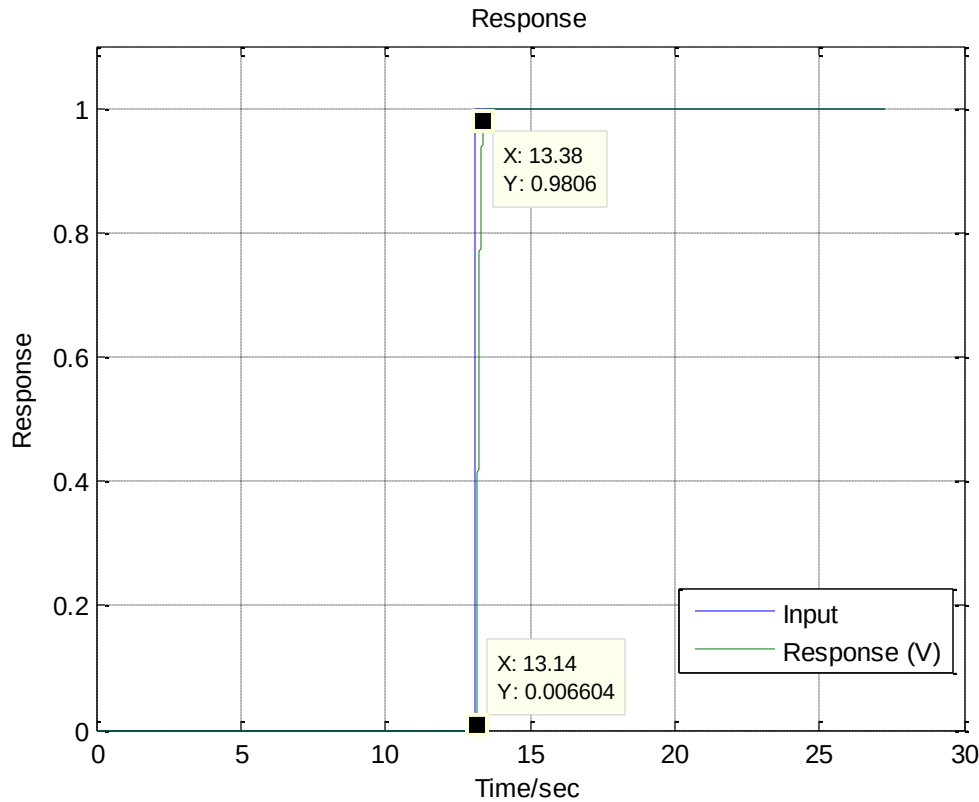


Iteration: Final value)

$K_p = 0.35$; $K_i = 0.3$; $K_d = 0.01$

Percentage overshoot: 0

Setting time: 0.24 sec



Our final tuned PID controller values were: $K_p = .35$, $K_i = .3$, $K_d = .001$

Which gives us a transfer function result of:

$$\frac{U(s)}{E(s)} = G_c(s) = K_p + \frac{K_i}{s} + K_d * s = K_d * \left(\frac{s^2 + \left(\frac{K_p}{K_d}\right) * s + \left(\frac{K_i}{K_d}\right)}{s} \right)$$
$$\rightarrow G_c(s) = .35 + \frac{.3}{s} + .001 * s = .001 * \left(\frac{s^2 + \left(\frac{.35}{.001}\right) * s + \left(\frac{.3}{.001}\right)}{s} \right)$$
$$\rightarrow G_c(s) = .35 + \frac{.3}{s} + .001 * s = .001 * \left(\frac{s^2 + 350 * s + 300}{s} \right)$$

4. Discussion Questions

Explain why it is practically impossible to implement a PID without a derivative filter?

It is necessary to have filtering to take care of the derivative part of PID controller. The error signal go into the derivative part would have sharp transitions coming from either the input step or the output noise fed back, either of them would result in huge control input from PID controller. This in practice will draw a lot of control effort and damage the plant. Step input will be limited by a rate limiter and the signal noise will therefore be constrained by a derivative filter (low pass).

5. Conclusions

For this experiment, the goal was to investigate the proportional-integral-derivative (PID) type of control law, and model-based tuning rules for designing a PID controller to perform position control of the SRV-02 DC motor with modeled and un-modeled dynamics. We used the PID design method in creating a compensator in a unity feedback configuration. The choice of the poles and values was determined first buy hand, then tweaked via computer-aided design tools. During the lab we further tuned the PID controller to fit with our DC motor, and obtained the values $K_p = .35$, $K_i = .3$, $K_d = .001$, and a transfer function equation of: $G_c(s) = .35 +$

$$\frac{.3}{s} + .001 * s = .001 * \left(\frac{s^2 + 350*s + 300}{s} \right)$$

ECE 557:

Lab 8 – Position Control for a Flexible Joint

Lab 9 – Position Control for a Flexible Link

Members: Larissa Borcz, Xiaoxi Zhao

Bench: #2

Instructor: Anusha Mannava

Lab Performed: 3/1/12

Abstract: The first objective of this laboratory was 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. The second objective of this laboratory was to position the tip of a rotary flexible link as quickly as possible with minimal vibrations using a PID controller.

1. Introduction

We accomplished two laboratory experiments during our lab time. The objective of the first laboratory was 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. We developed the model of the joint system and then tuned the PID gains to come up with the best position tracking while minimizing endpoint oscillation. The objective of the second laboratory was to position the tip of a rotary flexible link as quickly as possible with minimal vibrations using a PID controller. We developed the model of the (motor+link) system and then tuned the PID gains to come up with the best position tracking while minimizing endpoint oscillation.

2. Procedure

2.1 – Lab 8

For the first experiment, Lab 8, we modeled a flexible joint robotic arm using the default spring configuration, seen in Figure 8.2 along with the corresponding values.

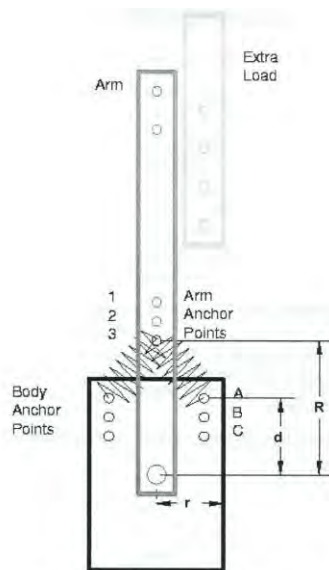


Figure 8.2 – Default Spring Configuration

θ is the motor shaft position, measured using an encoder.

α 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_{stiff} is a linear approximation of the joint stiffness. For the default connection (spring type 2 and anchors A-3, see Figure 8.2) the value of K_{stiff} is 1.6108 Nm/rad .

J_{hub} is the total inertia of the motor, with value 0.0021 Kg.m^2 .

J_{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 2.6Ω .

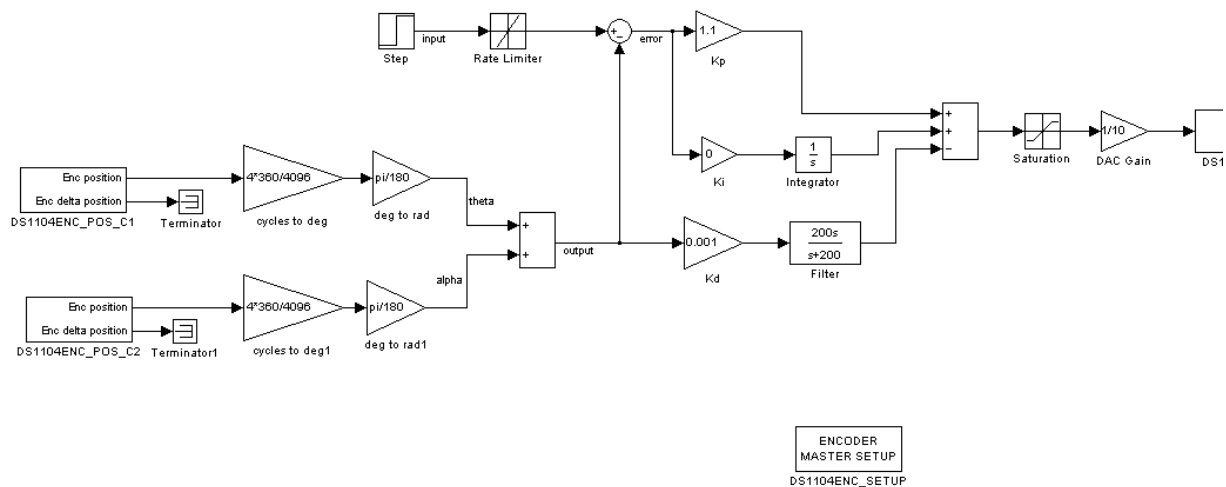
K_m is one of the motor torque constants, with value $0.00767 \text{ V/rad/sec}$.

K_g is the gear ratio of the motor. In the high gear ratio configuration, $K_g = 70$.

We created the Vin to (theta+alpha) and Vin to alpha output transfer functions in prelab, tuned PID K values in Matlab to compensate for the transfer functions, then implemented the found PID values in our ControlDesk setup during lab.

During the lab, we used the connections stated in the lab manual procedure section 8.4.1 to connect the DC servomotor, the joint, the DS1104 interface board, and the computer. We used the setup and procedures stated in the lab manual procedure section 8.4.2 to setup Simulink, run the ControlDesk and obtain results.

Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 30s, unit input step input start at $t = 10$, and have a final value of 10. The slew rate was set to ± 1000 , and saturation limits to $\pm 5V$. We then tuned the values of our K_p , K_i , and K_d during the lab to get our desired response.

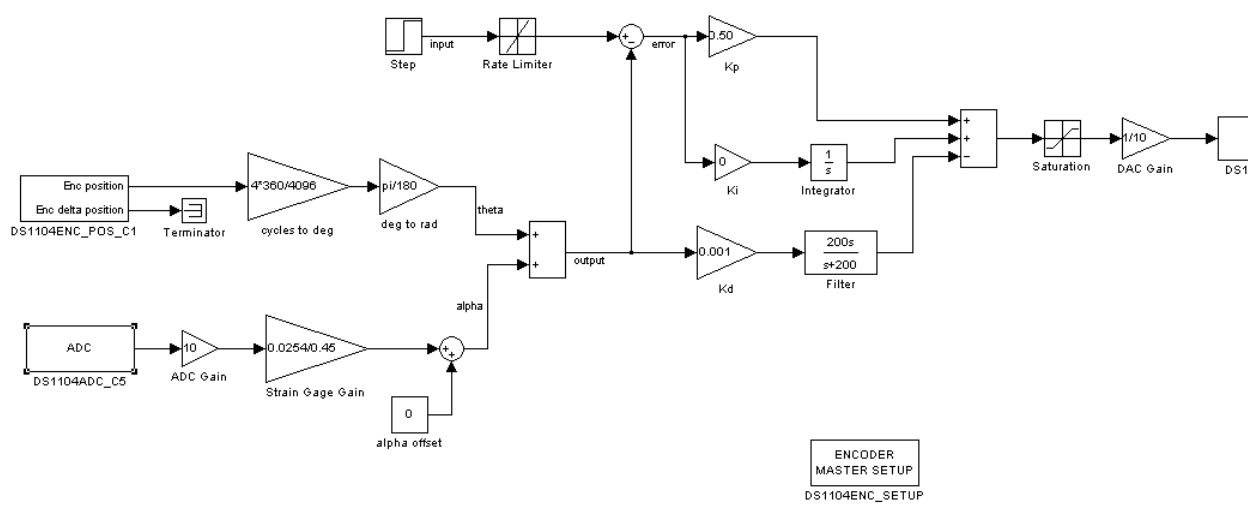


2.2 – Lab 9

We created the V_{in} to $(\theta + \alpha)$ and V_{in} to α output transfer functions in prelab, tuned PID K values in Matlab to compensate for the transfer functions, then implemented the found PID values in our ControlDesk setup during lab.

During the lab, we used the connections stated in the lab manual procedure section 9.4.1 to connect the DC servomotor, the flexible link, the DS1104 interface board, and the computer. We used the setup and procedures stated in the lab manual procedure section 9.4.2 to setup Simulink, run the ControlDesk and obtain results.

Our design, seen in the figure below, used a fixed step size of 1ms, simulation time of 30s, unit input step input start at $t = 10$, and have a final value of 10. The slew rate was set to ± 1000 , and saturation limits to $\pm 5V$. We then tuned the values of our K_p , K_i , and K_d during the lab to get our desired response.



3. Results

A compensator is a component in a control system that improves an undesirable frequency response in a feedback and control system. It is a fundamental building block in classical control theory. The PID controller attempts to minimize the error by adjusting the process control inputs. The PID controller uses three separate constant parameters: the proportional (P), the integral (I), and derivative (D) values. We adjusted our PID values throughout both experiments in order to achieve the desired signal.

For our experiment we attempted to design a compensated signal using the output transfer functions we designed in the pre-lab. Our goal for Lab 8 was to achieve a signal with an initial value u of less than 5V, a steady state error of less than .5%, a peak overshoot of less than 2%, and a settling time of less than .6 seconds. Our goal for Lab 9 was to achieve a signal with less than 5% overshoot, a steady state error of less than 1%, less than .6 sec settling time, and an initial value u of less than 5V.

3.1 – Lab 8

For this lab we first computed in prelab the design matrices used to model the system. By filling in the values given in Section 8.2

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_{stiff}}{J_{hub}} & \frac{-K_m^2 K_g^2}{R_m J_{hub}} & 0 \\ 0 & \frac{-K_{stiff}(J_{load} + J_{hub})}{J_{hub} J_{load}} & \frac{K_m^2 K_g^2}{R_m J_{hub}} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \frac{K_m K_g}{R_m J_{hub}} \\ -\frac{K_m K_g}{R_m J_{hub}} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

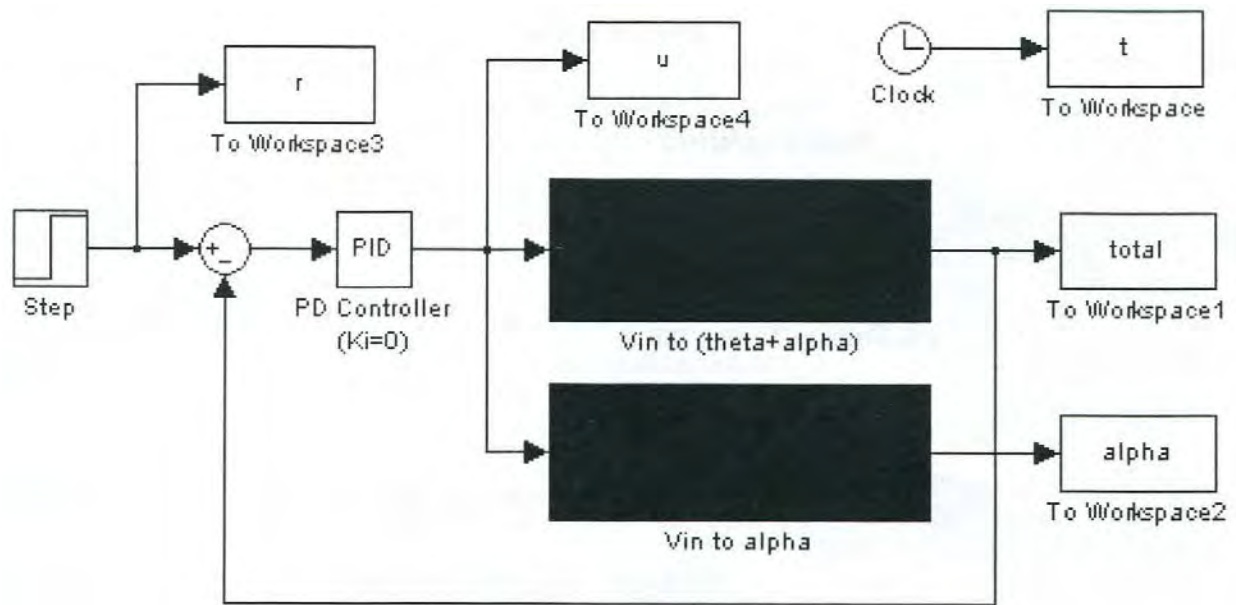
This gave the below final design model matrices of:

$$\begin{aligned} A &= \begin{bmatrix} 1.0e+003 & 0 & 0.0010 & 0 \\ 0 & 0 & 0 & 0.0010 \\ 0 & 0.7670 & -0.0528 & 0 \\ 0 & -1.0401 & 0.0528 & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 0 \\ 98.3333 \\ -98.3333 \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \\ D &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Then using Matlab, we solved for the output transfer functions, resulting in:

$$\begin{aligned} y_1(\theta) &= \frac{98s^2 + 26847}{s^4 + 53s^3 + 1040s^2 + 14414s} \\ y_2(\alpha) &= \frac{-98s^2}{s^4 + 53s^3 + 1040s^2 + 14414s} \\ y_3(\theta + \alpha) &= \frac{26847}{s^4 + 53s^3 + 1040s^2 + 14414s} \end{aligned}$$

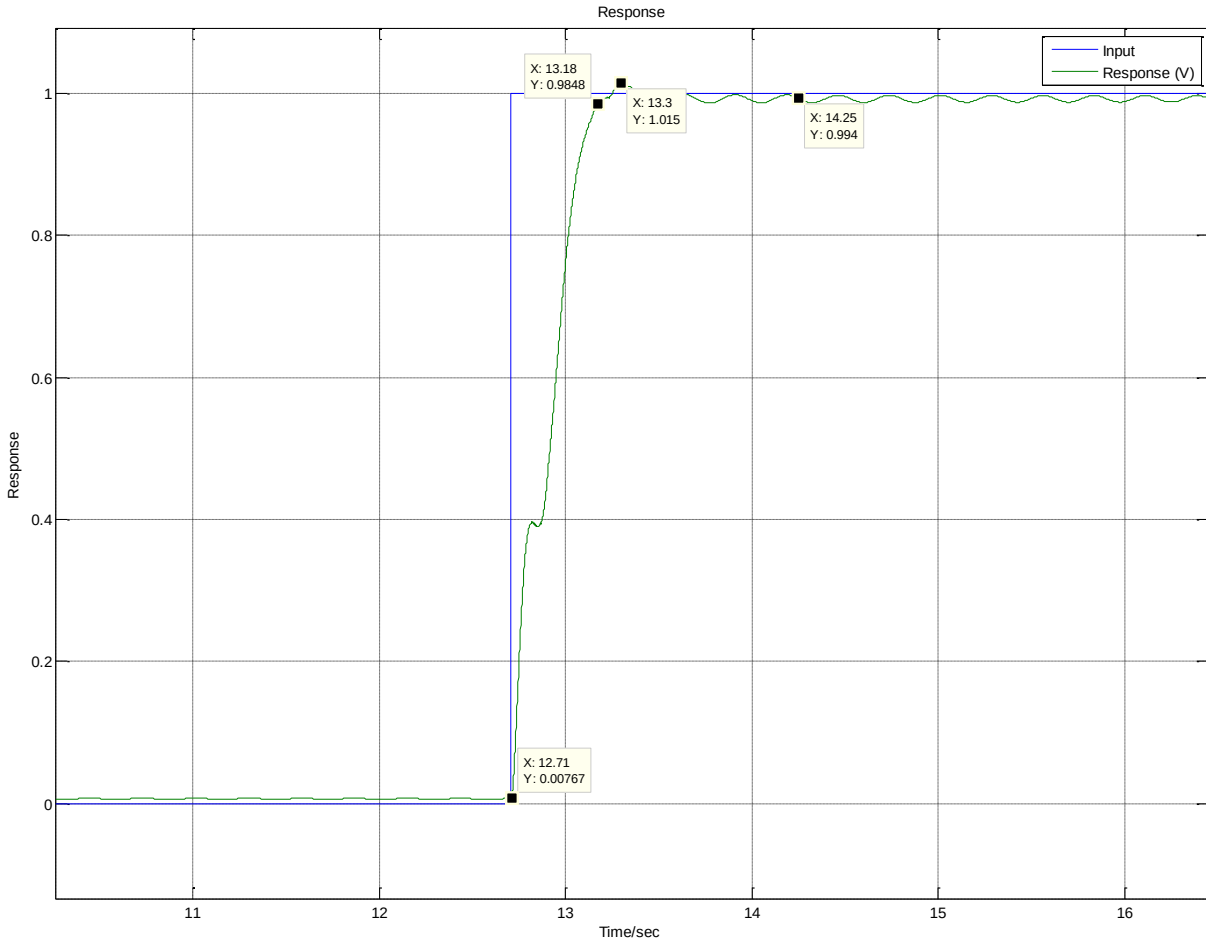
Inserting the corresponding y output transfer function equations into the below Simulink figure, we tuned our PID values during prelab.



To begin our lab simulation, we started out using the K values we solved for in the prelab:

$$K_p = 2.8, K_i = 0, K_d = .001$$

We then simulated the PID compensator for several iterations, and tuned the K values to achieve the wanted response, which is shown in the below figure.



Our tuned PID controller values were: $K_p = 1.1$, $K_i = 0$, $K_d = .001$

With an Overshoot: 1.5%, Steady state error: 0.6%, and Settling time (0.2%): 0.47 sec

These values give us essentially a PD controller, and a transfer function result of:

$$\frac{U(s)}{E(s)} = G_c(s) = K_p + \frac{K_i}{s} + K_d * s = K_d * \left(\frac{s^2 + \left(\frac{K_p}{K_d}\right) * s + \left(\frac{K_i}{K_d}\right)}{s} \right)$$

$$\rightarrow G_c(s) = 1.1 + .001 * s = .001 * \left(\frac{s^2 + \left(\frac{1.1}{.001}\right) * s}{s} \right)$$

$$\rightarrow G_c(s) = 1.1 + .001 * s = .001 * \left(\frac{s^2 + 1100s}{s} \right)$$

3.1 – Lab 9

For this lab we first used the stated model variables from the lab manual, which gave the below final design model matrices of:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 2165 & -53.6 & 0 \\ 0 & -2797 & 53.60 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 99 \\ -99 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

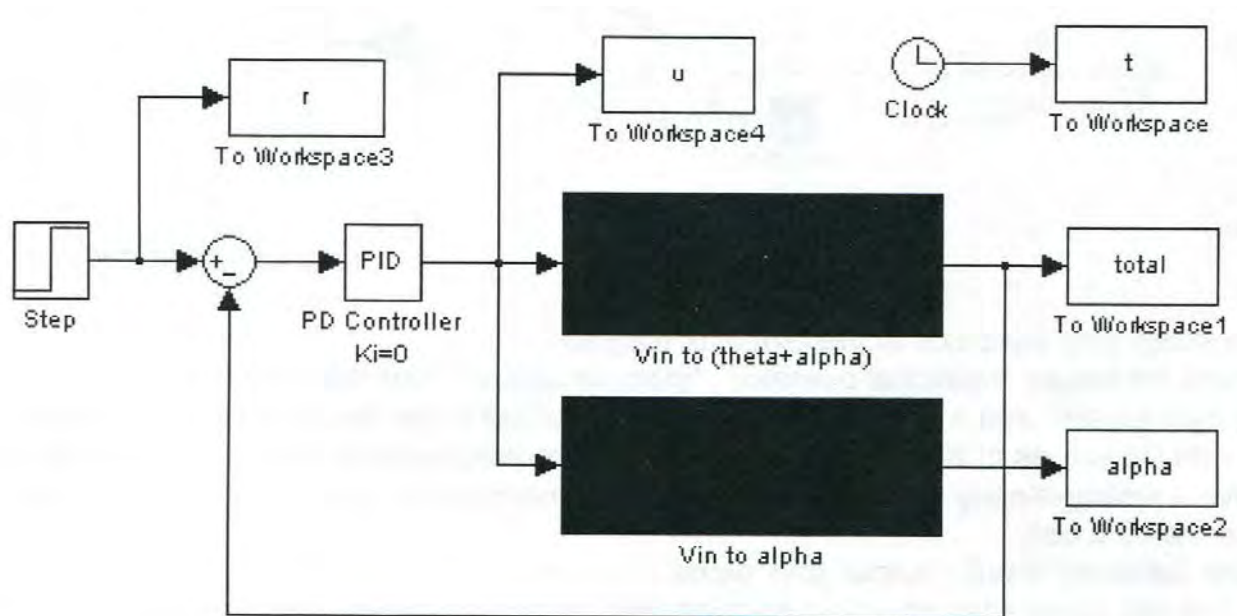
Then using Matlab, we solved for the output transfer functions, resulting in:

$$y_1(\theta) = \frac{99s^2 + 62568}{s^4 + 54s^3 + 2797s^2 + 33875s}$$

$$y_2(\alpha) = \frac{-99s^2}{s^4 + 54s^3 + 2797s^2 + 33875s}$$

$$y_3(\theta + \alpha) = \frac{62568}{s^4 + 54s^3 + 2797s^2 + 33875s}$$

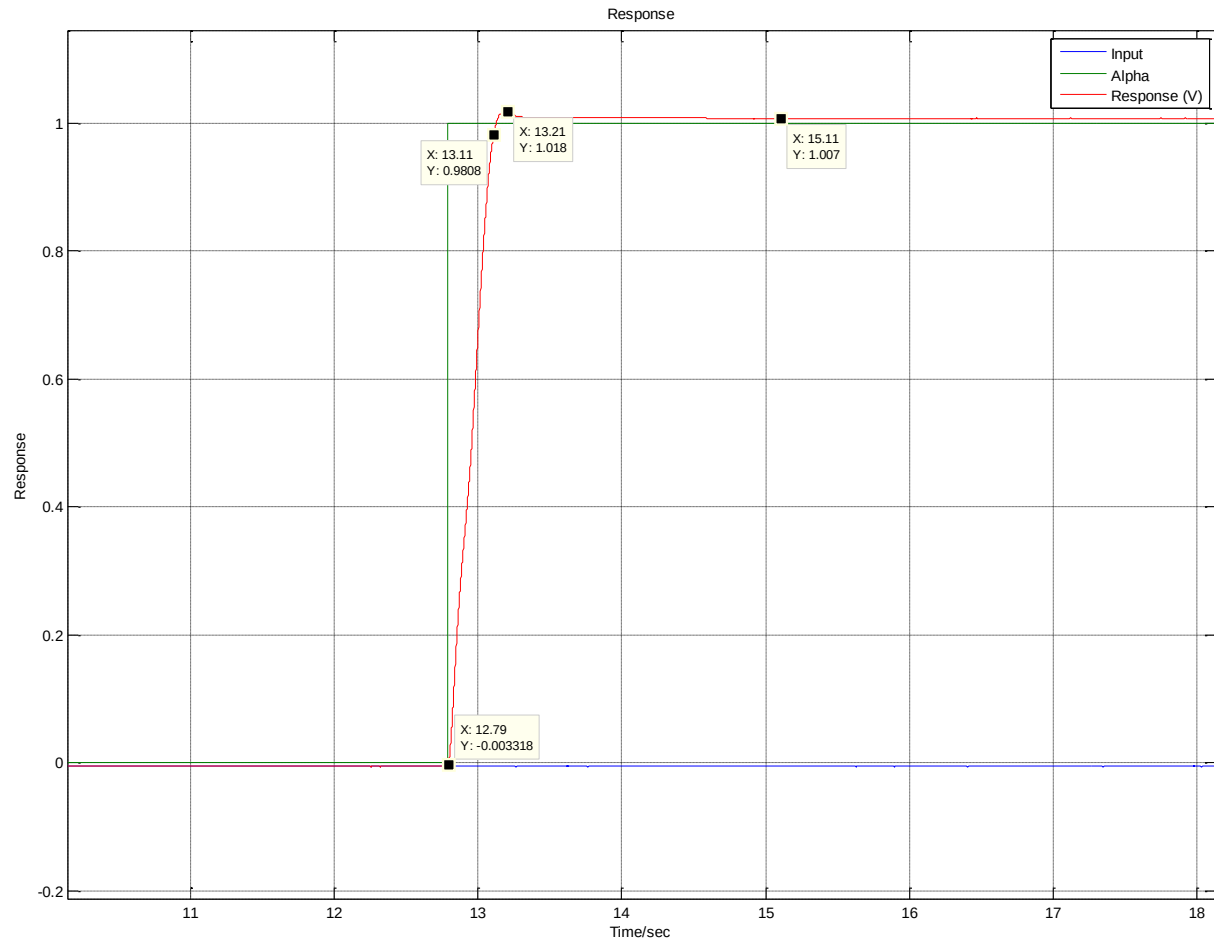
Inserting the corresponding y output transfer function equations into the below Simulink figure, we tuned the PID values during prelab.



To begin our simulation, we started out using the K values we solved for in the prelab:

$$K_p = 2.2, K_i = 0, K_d = .0001$$

We simulated the PID compensator for several iterations, and tuned it to achieve the wanted response, which is shown in the below figure.



Our tuned PID controller values were: $K_p = 1.3$, $K_i = 0$, $K_d = 0.01$

With an Overshoot: 1.8%, Steady state error: 0.7%, and Settling time: 0.32 sec

These values give us essentially a PD controller, and a transfer function result of:

$$\frac{U(s)}{E(s)} = G_c(s) = K_p + \frac{K_i}{s} + K_d * s = K_d * \left(\frac{s^2 + \left(\frac{K_p}{K_d}\right) * s + \left(\frac{K_i}{K_d}\right)}{s} \right)$$

$$\rightarrow G_c(s) = 1.3 + .01 * s = .01 * \left(\frac{s^2 + \left(\frac{1.3}{.01}\right) * s}{s} \right)$$

$$\rightarrow G_c(s) = 1.3 + .01 * s = .01 * \left(\frac{s^2 + 130s}{s} \right)$$

4. Discussion Questions

For our post lab exercises we were given the following tasks stated below:

8.5 Post-Laboratory Exercises

- Plot the acquired data and show the gain values, the steady-state error and the percent overshoot on the plot.
- Why was PD control alone (without the integral term) enough to obtain a satisfactory response in the preliminary design?
- Based on your answer to the previous question, why would you need to add integral control during implementation?
- 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.

9.5 Post-Laboratory Exercises

- Plot the acquired data and show the gain values, the steady-state error, the percent overshoot and the settling time on the plot.
- Explain how we obtained the strain gauge gain. (The length of the link is $L = 0.45 \text{ m}$).
- Explain why we have a minus sign for the K_d path in the final summing junction.
- Explain why the response is not as smooth as that of the flexible joint in the previous Laboratory.
- Draw comparison and contrast between the results of Laboratory #8 and Laboratory #9.

Lab 8

Why was PD control alone enough to obtain a satisfactory response in the preliminary design?
The system is completely controllable which means by choosing value of K we are able to place poles anywhere we need within finite time.

Test:

$$\dot{x} = Ax + Bu$$

$[A \ B \ A^2B \ A^3B]$ is of full rank $\Rightarrow$ controllable

A different way to approach the control problem

I draw out the block diagram directly base on state space model

Lab 9

Strain gauge gain

From trigonometry, displacement = angle in rad * r

Therefore 1V => 1 inch displacement = 1 inch/0.45m (0.45m is r)

Explain why we have a minus sign for the Kd path in the final summing junction

Step input has sharp edge and therefore its derivative could be too large to implement. We assume step remains constant and then the derivative equation becomes:

$$K_d * \text{error dot} = K_d(r \text{ dot} - y \text{ dot}) = -K_d(y \text{ dot})$$

where dot refers to derivative.

Explain why the response is not as smooth as that of the flexible joint in the previous lab?

Because the link is less rigid.

5. Conclusions

Our two experiments were: 1) Lab 8: 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, 2) Lab 9: Position the tip of a rotary flexible link as quickly as possible with minimal vibrations using a PID controller. For objective one, we developed the model of the joint system and then tuned the PID gains to come up with the best position tracking while minimizing endpoint oscillation. For objective two, we developed the model of the (motor+link) system and then tuned the PID gains to come up with the best position tracking while minimizing endpoint oscillation. We used the PID design method in compensating our response. The choice of the poles and values was determined first by hand, then tweaked via computer-aided design tools. Our results showed that instead of a PID compensator, we used a PD compensator.

We obtained the final Lab objective values:

- 1) $K_p = 1.1$, $K_i = 0$, $K_d = .001$, with an Overshoot: 1.5%, Steady state error: 0.6%, and Settling time (0.2%): 0.47 sec, and final compensator transfer function of:

$$G_c(s) = 1.1 + .001 * s = .001 * \left(\frac{s^2 + 1100s}{s} \right)$$

- 2) $K_p = 1.3$, $K_i = 0$, $K_d = 0.01$, with an Overshoot: 1.8%, Steady state error: 0.7%, and Settling time: 0.32 sec, and final compensator transfer function of:

$$G_c(s) = 1.3 + .01 * s = .01 * \left(\frac{s^2 + 130s}{s} \right)$$