King Fahd University of Petroleum & Minerals

CISE 316
Control System Design
Laboratory Manual

Systems Engineering Department

Table of Contents

CISE 316

Control Systems Design

Lab Experiment 1: Introduction to Root Locus

Objective: The primary objective of this experiment is to learn root locus using MATLAB.

Introduction:

Application of the many classical and modern control system design and analysis tools is based on mathematical model. MATLAB can be used with systems given in the form of transfer function description. We are interested in how MATLAB can assist us in determining:

(i) the number of branches.
(ii) the starting and ending points of all the branches.
(iii) the intersections of the root loci with the imaginary axis and the corresponding value of *K*.
(iv) the system's oscillating frequency associated with the gain *K*.
(v) the breakaway and break-in points.
(vi) the value of *K* at breakaway point.

List of Equipment/Software

Following equipment/software is required:

- MATLAB

Category Soft-Experiment

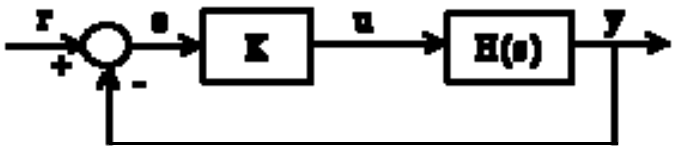
Deliverables

A complete lab report including the following:

- Your steps/calculations and answers
- Your hand sketch of the root locus.
- The root locus plotted by using MATLAB (On the figure window, click Edit → Copy Figure, and then paste to a word document).
- Make sure all necessary information is shown on the plot (refer to item (a) until (h) in Lab Assignment section) by clicking on the locus and moving your cursor accordingly. Note: You can adjust the font size and alignment by right-clicking on it.

Root Locus

The root locus of an (open-loop) transfer function H(s) is a plot of the locations (locus) of all possible closed loop poles with proportional gain *K* and unity feedback:



The closed-loop transfer function is:

$$\frac{Y(s)}{R(s)} = \frac{K\,H(s)}{1 + K\,H(s)}$$

and thus the poles of the closed loop system are values of *s* such that **1 + K H(s) = 0**.

If we write **H(s) = b(s)/a(s)**, then this equation has the form:

$$a(s) + K\,b(s) = 0$$

$$\frac{a(s)}{K} + b(s) = 0$$

Let n = order of a(s) and m = order of b(s) [the order of a polynomial is the highest power of s that appears in it].

We will consider all positive values of *K*. In the limit as $K \rightarrow 0$, the poles of the closed-loop system are a(s) = 0 or the poles of H(s). In the limit as $K \rightarrow$ infinity, the poles of the closed-loop system are b(s) = 0 or the zeros of H(s).

Independently from *K*, **the closed-loop system must always have n poles**, where n is the number of poles of H(s). **The root locus must have n branches**, each branch starts at a pole of H(s) and goes to a zero of H(s). If H(s) has more poles than zeros (as is often the case), m < n and we say that H(s) has **zeros at infinity**. In this case, the limit of H(s) as s → infinity is zero. The number of zeros at infinity is n-m, the number of poles minus the number of zeros, and is the number of branches of the root locus that go to infinity (asymptotes).

Since the root locus is actually the locations of all possible closed loop poles, from the root locus we can select a gain such that our closed-loop system will perform the way we want. If any of the selected poles are on the right half plane, the closed-loop system will be unstable. The poles that are closest to the imaginary axis have the greatest influence on the closed-loop response, so even though the system has three or four poles, it may still act like a second or even first order system depending on the location(s) of the dominant pole(s).

PLOTTING THE ROOT LOCUS OF A TRANSFER FUNCTION

Consider an open loop system which has a transfer function of

$$KH(s) = \frac{K}{s(s+3)(s^2+2s+2)}$$

How do we design a feed-back controller for the system by using the root locus method? Make a MATLAB file called rl.m. Enter the transfer function and the command to plot the root locus:

```
% Create the numerator polynomial.
num = 1;

% Create the denominator polynomial, using the conv function. Conv takes vectors and
expands them.
den = conv(conv([1 0],[1 3]),[1 2 2]);

% Create a new figure window to plot on. It will not overlap with the previous figure,
%provided that it is not labeled with the same number (i.e. figure(1), figure(2) etc).
figure(1)

% Plot the root locus with the command rlocus.
rlocus(num,den)
```

LAB ASSIGNMENT:

By using the same transfer function, draw manually the root locus plot for the system and determine:

(a) the number of branches.

(b) the starting and ending points of all the branches.

(c) the location of the centroid and angles of asymptotes.

(d) the range of K to keep the system stable.

(e) the intersections of the root loci with the imaginary axis and the corresponding value of K.

(f) the system's oscillating frequency associated with the gain K found above.

(g) the angle of departure and angle of arrival (if any).

(h) Given that the breakaway point is at $s = -2.3$, find the value of K at that point.

Verify your answers with that generated using MATLAB.

Click on any of the locus and move your cursor accordingly.

You will see, among the 6 items appeared on the plot, 3 of them are 'Gain', 'Pole' and 'Frequency (rad/sec)'.

'Gain' refers to the value of K.

'Pole' refers to the value of poles (the roots of your denominator).

'Frequency (rad/sec)' refers to ω.

CISE 316

Control Systems Design

Laboratory Experiment 2: Controller Design Using Root Locus

Objectives: At the conclusion of this laboratory experience, students should be able to:

- Compute the defect angle related to a pair of desired closed-loop pole locations.
- Use the defect angle to aid in the placement of compensator (controller) poles and zeros.
- Design P, PD, PI, and PID controllers to meet closed-loop performance specifications including transient performance and steady-error.

Overview

In this lab you will explore the use of the root locus controller design methodology. The root locus indicates the achievable closed-loop pole locations of a system as a parameter (usually the controller gain) varies from zero to infinity. For a given plant it may or may not be possible to implement a simple proportional controller (i.e., select a gain that specifies closed-loop pole locations along the root locus) to achieve the specified performance constraints. In fact, in most cases it will not be possible. When this occurs, it is the control engineers job to select a controller structure (a gain and numbers of poles and zeros of a controller transfer function) and the respective controller parameters (values for the gain and poles and zeros) to change the shape of the root locus so that for some values of the controller gain, the dominant second order closed-loop poles lie within the performance region. In this lab we are investigating several controller structures on individual plants and comparing the design process and performance. We will be using the MATLAB 'sisotool' toolbox to complete the root locus designs.

List of Equipment/Software

Following equipment/software is required:

- MATLAB

Category Soft-Experiment

Deliverables

A complete lab report including the following:

- Figures with plots of closed-loop step responses and control efforts.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs.

Background

For this lab, we will assume a unity feedback controller of the form shown in Figure 1, where $C(s)$ is the controller transfer function and $P(s)$ is the plant transfer function.

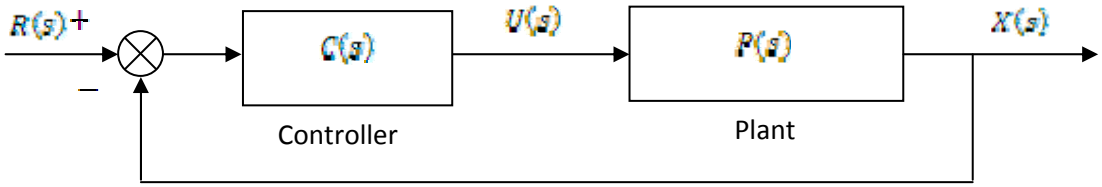


Figure 1 – Generic Unity Feedback Control System.

Recall that the first step to any root locus controller design is computing the defect angle, which can be interpreted as how far a set of desired second order closed-loop poles are from being on the root locus. The larger the magnitude of the defect angle, the "further" those desired closed-loop poles are from being on the root locus. Once you know the defect angle, you can make an informed decision for controller structure. With controller structure specified, the defect angle prescribes the relative locations of the poles and zeros.

To find the defect angle, first set $C(s) = K$, a simple proportional controller. Then use MATLAB (you can use 'sisotool' or the 'rlocus' command – use MATLAB help to learn more about 'rlocus') to plot the root locus using the given transfer function for the plant. Next, compute the desired closed-loop pole locations ($s = \sigma \pm j\omega$) and place them on the complex plane where you plotted the root locus. Finally, compute the defect angle using the definition in (1) below.

$$\varphi_d = \angle G(s)|_{s=\sigma\pm j\omega} \pm (2n+1)\cdot 180^\circ \qquad (1)$$

Select the n in $\pm(2n+1)\cdot 180^\circ$ and the 'plus' or 'minus' option to make the magnitude of φ_d as small as possible. Note that (1) can also be written as

$$\varphi_d = \sum_i \angle(s-z_i)|_{s=\sigma\pm j\omega} - \sum_j \angle(s-p_j)|_{s=\sigma\pm j\omega} \pm (2n+1)\cdot 180^\circ \qquad (2)$$

where the angles are defined as shown in Figure 2.

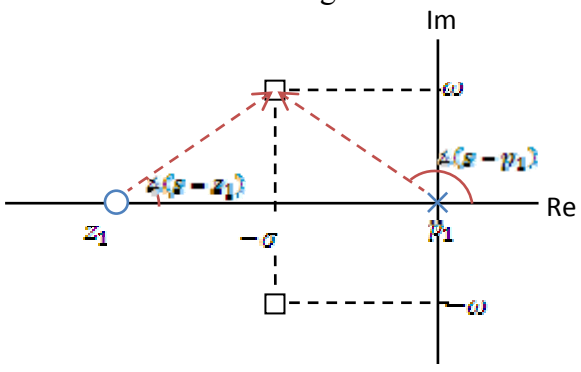


Figure 2 – Defining the Angles $\angle(s-z_i)$ and $\angle(s-p_j)$ for the Defect Angle Calculation.

Note, in controller design there are multiple possible solutions, some better than others. It is possible to have multiple designs that satisfy the given performance constraints, but practical implementation issues and cost could be prohibitive for some designs. As a general rule, it is a good idea to keep your controller as simple as possible while meeting the prescribed performance criteria. In this lab we will be investigating several controller structures on individual plants and comparing the design process and performance. The common controller structures we will be using in this lab are listed in Table 1 along with their respective transfer functions.

Table 1 – Common Controller Types

Controller Type	Controller Structure
Proportional (P)	$C(s) = k_p$
Integral (I)	$C(s) = \frac{k_i}{s}$
Proportional + Integral (PI)	$C(s) = k_p + \frac{k_i}{s} = \frac{k \cdot (s+z)}{s}$
Lag Controller	$C(s) = \frac{K_c \cdot (s+z)}{(s+p)}$ where $\|z\| > \|p\|$
Proportional + Derivative (PD)	$C(s) = k_p + k_d s = k \cdot (s+z)$
Lead Controller	$C(s) = \frac{k \cdot (s+z)}{(s+p)}$ where $\|p\| > \|z\|$
Proportional + Integral + Derivative (PID) (Real Zeros)	$C(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k \cdot (s+z_1)(s+z_2)}{s}$
Proportional + Integral + Derivative (PID)	$C(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k \cdot (s+z)(s+z^*)}{s}$

(Complex Conjugate Zeros)	

Note that the I, PI, and PID controllers will produce a position error (e_p) of zero as long as the plant does not contain a zero at the origin, which would cancel the controllers pole at the origin.

Introduction to MATLAB 'sisotool'

A. Getting Started

1. Enter the transfer function for the plant, $P(s)$, in your workspace (i.e., from the MATLAB command prompt).
2. Type 'sisotool' at the command prompt.
3. Click 'close' when the help window comes up.
4. Click on **View → Design Plots Configuration…** In the 'Plot Type' column to the far right of the window that opens, make sure all are set to 'None' except the first one, which should be set to 'Root Locus' to turn off the bode plot.

B. Loading the Transfer Function

1. Click on **File → Import**.
2. A window on the left will show you the transfer functions in your workspace, while the window on the right will let you choose the control system configuration.
3. We will usually be assigning $P(s)$ to block 'G' (the plant). Double-click the space next to 'G' and type your transfer function name and hit **Enter**. You must hit enter or nothing will happen.
4. Once you hit enter, you should be able to click the **OK** button at the bottom of the window. Then the window will close.
5. After you enter the transfer function, the root locus will be displayed. Double check to make sure that the open-loop poles and zeros of your plant are in the correct locations.

C. Generating the Step Response and the Control Effort Plot

1. Click on **Analysis → Response to Step Command**.

2. You will probably have both curves on one set of axes plot. To fix this click on the 'Control and Estimation Tools Manager' window to activate it. Next, click on the 'Analysis Plots' tab. Under 'Plot 2' select 'Step' for the plot type. Now, in the lower window, check the box next to 'closed-loop r to u' under 'plot 2' and uncheck the same box under 'plot 1'. You should now have the step response in the upper plot of the figure window and the control effort in the lower plot of the figure window.

3. You can now click on the pink boxes on the root locus (the current closed-loop poles for the given gain) and move them along the root locus. Essentially, you are exploring different controller gain values by doing this. Note how the step response changes as you move the closed-loop pole locations.

4. The values of the closed-loop poles will appear at the bottom of the root locus window as you click and hold the mouse on the pink boxes representing them. This only gives you the value of the closed-loop pole you are clicking on. If you need the other closed-loop pole locations, you will have to click on them on each of the other branches.

D. Entering the Compensator (Controller)

1. In the 'Control and Estimation Tools Manager' window, click on the **Compensator Editor** tab. To add compensator poles and zeros, right click in the 'Dynamics' box. You will be able to make changes to these values later. After you enter a pole/zero location in the box that appears to the right in the window, hit **Enter**.

2. Look at the form of $C(s)$ to be sure it is correct. Then look at the root locus window and see how it changed once the compensator was added.

3. You can again see how the step response changes with the compensator by c licking on the closed-loop poles (the pink squares) and dragging them along the root locus.

4. You can also change the location of the poles/zeros of the compensator by clicking on them and dragging them. Be careful not to inadvertently change the poles and zeros of the plant!

E. **Adding Design Constraints**

1. Right-click on the root locus plot. From the menu that pops-up, select **Design Requirements → New**… to add constraints or **Design Requirements → Edit**… to edit existing constraints.

2. At this point you can choose from settling time, percent overshoot, damping ratio, and natural frequency constraints.

F. **Printing/Saving the Figures**

To save a figure 'sisotool' created during your session, click **File → Print to Figure**. This opens a figure window and puts the current figure there.

G. Odds and Ends

1. You may want to adjust the axes. To do this, right-click on the root locus plot and select **Properties** from the list. Click on the **Limits** tab and set the desired axis limits.
2. You may also want to turn the grid on. Right-click on the root locus plot and select **Grid** from the pop-up list.
3. It is convenient to use the zero/pole/gain format for the compensators. To do this, from the root locus window select **Edit → SISO Tool Preferences → Options** and click on **zero/pole/gain**.

In-Lab – Part A

Use the plant given in (3) for this section of the lab.

$$P(s) = \frac{30}{s^2+11s+30} = \frac{30}{(s+5)(s+6)} \tag{3}$$

This is a second order system with two real poles located at -5 and -6. Our goal is to speed up the closed-loop system response so that the two-percent settling time is less than 1 second, produce a position error of 0.1 or less, and keep percent overshoot less than 10%. To keep things reasonable, keep the gain, k, less than 10 for all designs.

1. Compute the Defect Angle

Compute the defect angle for the closed-loop poles lying at the intersection of the transient performance constraints. Use this as a general guide as you seek to place you controller poles and zeros to achieve the performance specifications.

2. Entering the Constraints

Enter the percent overshoot and settling time constraints in 'sisotool'. Remember that these constraints are based on a second order system step response and for higher order systems are predicated by the assumption of second order dominance of the closed-loop system poles. Therefore, these design constraints are guidelines and you may have to refine your design to stay further inside these constraints to meet the performance specifications.

3. Proportional (P) Control

Determine the root locus for this system with proportional control. (When you enter the plant transfer function in 'sisotool', this is the default root locus plot. At this point the controller is specified as $C(s) = 1$.

Look at the step response as the gain increases. You should notice a few things:

- as k increases, the imaginary part of the closed-loop poles increases, and therefore the percent overshoot increases
- as k increases, the position error decreases
- since the real part of the pole does not change once k is greater than about 0.008, the settling time remains constant at about 0.8 seconds.
- there is no value of k for which the system is unstable

Do as well as you can to meet both constraints (you will not be able to do very well) then save the step response and control effort plots and your controller gain to turn in with the lab worksheet.

4. Integral (I) Control

a) Add a real pole at zero to implement the integral controller. You can do this from the root locus plot or the 'Control and Estimation Tools Manager' as described earlier. Note that once you place a compensator pole (or zero) you can click and drag it to a new location. However, for an integral controller the pole is always at zero, so leave it there for now.
b) You should note that there are two root locus branches that head towards the imaginary axis (toward instability), which is generally not desirable.
c) Find the value of k that makes the system marginally stable (the critical gain).
d) Try to find a value for k that gives a response with settling time less than or equal to 2 seconds and has little overshoot. Save the corresponding step response and control effort plots and the controller gain to turn in with the lab worksheet.
e) Is the position error zero? Could you find a k value for which you could meet the 1 second settling time constraint?

5. Proportional + Derivative (PD) Control

a) Edit your compensator by removing the integrator (the pole at zero) and add a real zero. Note that in this PD design that you can select where you place this real zero along the real axis. Take a moment to explore what happens to the root locus, the step response, and the control effort as you move the zero.
b) Now move the zero between -1 and -4. Find a configuration with a position error less than 0.5. Save the step response and control effort figure and the controller that produced it.

c) Next move the zero between -7 and -9. What happens to the root locus? Are we likely to get a faster response of the closed-loop system with this design than the previous one? Specify a controller with a zero in this range that produces a settling time of 0.1 seconds or less. (Don't forget that $k < 10$.) Save the step response and control effort figure and the controller that produced it.

6. Proportional + Integral (PI) Control

a) Edit your compensator by adding a real pole at zero (adding the integral element). Note that in this PI design that you can select where you place this real zero along the real axis, but that the real pole must remain at zero.
b) Place the zero between 0 and -5. Find a controller that produces a settling time of less than 0.8 seconds, a percent overshoot of less than 2%, and a position error of zero. Save the step response and control effort figure and the controller that produced it. Are all your poles inside the design region?
c) Now set the real zero to the left of -6. This type of configuration is not likely to get a faster response than with just a P controller. Why?

7. Proportional + Integral + Derivative (PID) Control

a) Let's start by making a PID controller with complex conjugate zeros. Edit the previous compensator by deleting the real zero and adding complex conjugate zeros at $-7 \pm j7$ and look at the root locus plot.
b) Find a gain value of k on this root locus so that the step response has less than 10% percent overshoot and a settling time less than 0.5 seconds. Save the step response and control effort figure and the controller that produced it. Are all your poles and zeros within the design region? Would you call this a second order dominant system?
c) Keeping the real part of the zeros at -7, reduce the imaginary part of the zeros as much as possible while keeping the same basic shape of the root locus. At some point, as you reduce the imaginary part, the root locus will take a very different shape. Find a value of k on this root locus so that the step response has a percent overshoot of less than 2%, settling time less than 0.02 seconds, and a position error of less than 0.01. (Remember to keep $k < 10$.) Save the step response and control effort figure and the controller that produced it.
d) Now let's make a PID controller with real zeros at -7 and -8. Determine the root locus for this system. Find a value of k on this root locus so that percent overshoot is less than 2% and the settling time is less than 0.02 seconds. (Remember to keep $k < 10$.) Save the step response and control effort figure and the controller that produced it.

In-Lab – Part B

Use the plant given in (4) for this section of the lab.

$$P(s) = \frac{8.96}{0.00147s^2 + 0.01455s + 1} \tag{4}$$

This is a model obtained from one of the mass-spring-damper systems in the controls lab.

Performance Constraints

- $e_p \leq 0.1$
- $t_{s,2\%} \leq 0.5$ seconds
- $\%OS \leq 10\%$

Controller Parameter Constraints

- $k_p \leq 1$
- $k_d \leq 0.03$
- $k_i \leq 10$

Meet these design constraints by implementing the following controller structures

As you choose the placement of the controller poles and zeros, use the defect angle to help guide you. Consider the angle added to the system by each pole and zero as you place it on the root locus. Remember that your compensator needs to add $-\varphi_d$ to the system in order to make the root locus pass through the desired closed-loop pole location.

- I controller (hard to meet settling time, probably need $T_{s,2\%} \approx 1$ sec)
- PD controller (try to get $T_{s,2\%} \leq 0.1$ sec)
- PI controller (hard to meet settling time, probably need $t_{s,2\%} \approx 1.5$ sec)
- PID controller with real zeros
- PID controller with complex conjugate zeros

For each one of these controller designs, you need to include your plot of the step response and control effort and your controller parameters.

In-Lab – Part C

Use the plant given in (5) for this section of the lab.

$$P(s) = \frac{9.29}{0.00087s^2+0.00118s+1} \tag{5}$$

This is a model obtained from another mass-spring-damper system in the controls lab.

Performance Constraints

- $e_p \leq 0.2$
- $T_{s,2\%} \leq 1$ seconds
- $\%OS \leq 20\%$

Controller Parameter Constraints

- $k_p \leq 1$
- $k_d \leq 0.03$
- $k_i \leq 10$

Meet these design constraints by implementing the following controller structures

As you choose the placement of the controller poles and zeros, use the defect angle to help guide you. Consider the angle added to the system by each pole and zero as you place it on the root locus. Remember that your compensator needs to add $-\varphi_d$ to the system in order to make the root locus pass through the desired closed-loop pole location.

- I controller (hard to meet settling time, do the best you can)
- PD controller (try to get $t_{s,2\%} \leq 0.1$ sec)
- PI controller (hard to meet settling time, probably need $t_{s,2\%} \approx 10$ sec)
- PID controller with real zeros
- PID controller with complex conjugate zeros

For each one of these controller designs, you need to include your plot of the step response and control effort and your controller parameters.

CISE 316

Control Systems Design

Laboratory Experiment 3: Digital Controller Design for DC Motor Position Control Using Root Locus Method

Objective: The objective of this exercise is the analysis and design of a Digital Control System for DC motor Position control. The procedure is the same as in continuous except that careful attention is needed to the requirement that the roots should be within the unit circle.

Introduction:

In this experiment, the controller will be designed by a Root Locus method. A digital DC motor model can obtain from conversion of analog DC motor model, as we will describe. The open-loop transfer function for DC motor's position is as shown.

$$\frac{\theta(s)}{V(s)} = \frac{K}{s((Js+b)(Ls+R)+K^2)}$$

where:

- electric resistance (R) = 4 ohm
- electric inductance (L) = 2.75E-6 H
- electromotive force constant (K=Ke=Kt) = 0.0274 Nm/Amp
- moment of inertia of the rotor (J) = 3.2284E-6 kg*m^2/s^2
- damping ratio of the mechanical system (b) = 3.5077E-6 Nms
- input (V): Source Voltage
- output (sigma dot): Rotating speed
- The rotor and shaft are assumed to be rigid

The **design requirements** are:

- $e_p \leq 0$
- $T_{s,2\%} \leq 0.04$ seconds
- $\%OS \leq 16\%$

List of Equipment/Software

Following equipment/software is required:

- MATLAB

Category Soft-Experiment

Deliverables

A complete lab report including the following:

- Figures with plots of closed-loop step responses and control efforts.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs.

Continuous to Discrete Conversion

The first step in the design of a discrete-time system is to convert a continuous transfer function to a discrete transfer function. MATLAB can be used to convert the above transfer function $\frac{\theta(s)}{V(s)}$ to discrete transfer function by using the **c2d** command. The **c2d** command requires three arguments: system, a sampling time (T) and a type of hold circuit. In this example we will use the zero-order hold (**zoh**).

From the design requirement, let the sampling time, **T** equal to 0.001 seconds, which is 1/100 of the required time constant or 1/40 of the required settling time.

Create a new m-file and add the following MATLAB code:

```
R = 4;
L = 2.75E-6;
K = 0.0274;
J = 3.2284E-6;
b = 3.5077E-6;

num = K;
den = [(J*L) (J*R)+(L*b) (R*b)+(K^2) 0];
motor = tf(num,den)

Ts = 0.001;
motor_d = c2d(motor, Ts, 'zoh')
[numd,dend] = tfdata(motor_d,'v')
```

MATLAB should return the following:

Transfer function:
0.001039 z^2 + 0.001021 z + 9.454e-10
---
z^3 - 1.942 z^2 + 0.9425 z

Sampling time: 0.001

```
numd =

 0    0.0010    0.0010    0.0000


dend =

1.0000   -1.9425    0.9425         0
```

Both the numerator and the denominator of the discrete transfer function have one extra root at z = 0. Also, we can get rid of the leading zero coefficient in the numerator. To do this, add the following code to cancel out extra pole and zero to avoid numerical problems in MATLAB.

```
numd = numd(2:3);
dend = dend(1:3);
motor_d = tf(numd,dend,Ts)
```

Therefore, the discrete-time transfer function from the motor position output to the voltage input is:

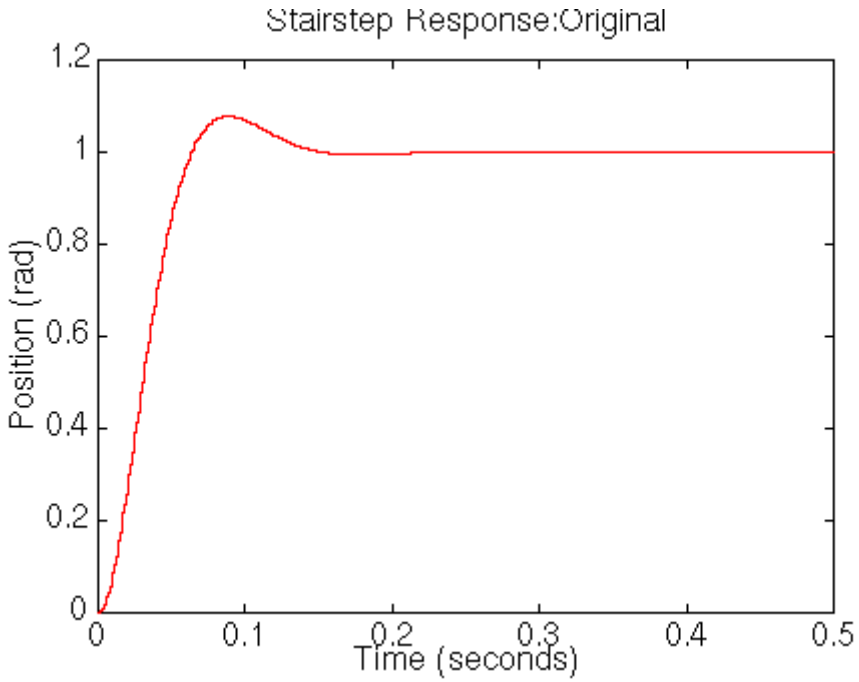
$$\frac{\theta(z)}{V(z)} = \frac{0.001z + 0.001}{z^2 - 1.9425z + 0.9425}$$

We would like to see what the closed-loop response of the system looks like when no controller is added. First, we have to close the loop of the transfer function by using the "**feedback**" command. After closing the loop, let's see how the closed-loop stair/step response perform by using the "**step**" and "**stairs**" commands. The "**step**" command will provide the vector of discrete step signals and "**stairs**" command will connect these discrete samples.

Add the following MATLAB code at the end of previous m-file and rerun it.

```
sys_cl = feedback(motor_d,1);
[x1,t] = step(sys_cl,.5);
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Step Response:Original')
```

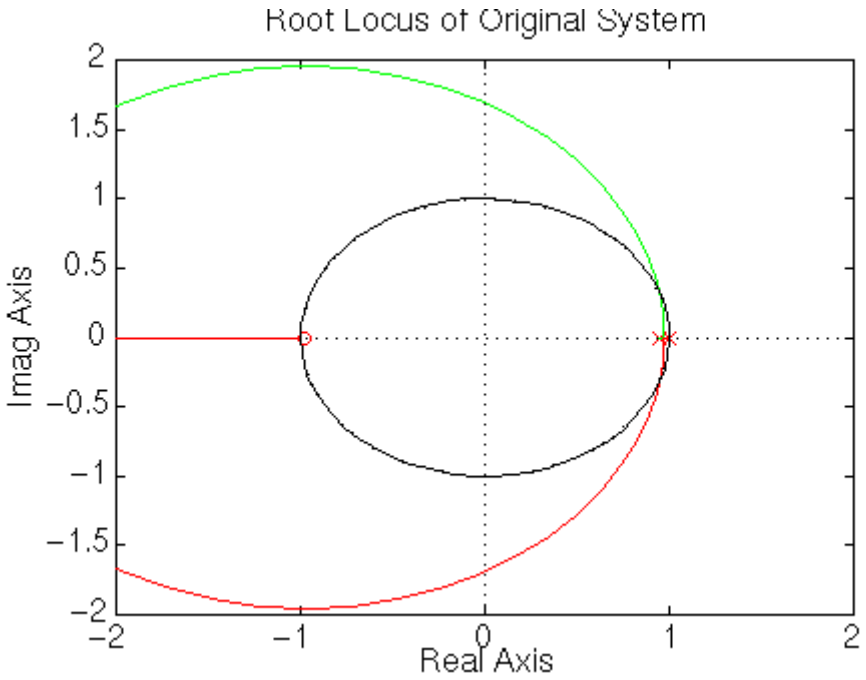
You should see the following plot:



Root Locus Design

The main idea of root locus design is to obtain the closed-loop response from the open-loop root locus plot. By adding zeroes and poles to the original system, the root locus can be modified, to a new closed-loop response. First let's see the root-locus for the system itself. In your m-file, add the following commands and rerun it. You should get the root-locus plot as shown below.

```
rlocus(motor_d)
title('Root Locus of Original System')
zgrid(0,0)
axis([-2,2,-2,2])
```



To get the zero steady-state error from the closed-loop response, we have to add an integral control. Recall that the integral control in the continuous- time is 1/s. If we use the backward difference approximation for mapping from the s-plane to the z-plane as described by s = 1/(z-1), one pole will be added at 1 on the root locus plot. After adding the extra pole at 1, the root locus will have three poles near 1. Therefore the root locus will move out to the right, and the closed-loop response will be more unstable. Thus, we must add one zero near 1, inside the unit circle, to cancel with one pole and pull the root locus in. We will add a zero at z = 0.95. In general, we must at least add as many poles as zeroes for the controller to be causal. Now add the following MATLAB commands in to your m-file.

```
numi = [1 -0.95];
deni = [1 -1];
icontr = tf(numi,deni,Ts);
```

Recall, the "**zgrid**" command can be used to find the desired region (satisfying the design requirements) on the discrete root locus plot. The "**zgrid**" command requires two arguments: the natural frequency (Wn) and the damping ratio (zeta). From the design requirement, the settling time is less than 0.04 seconds and the percent overshoot is less than 16%. We know the formulas for finding the damping ratio and natural frequency as shown:

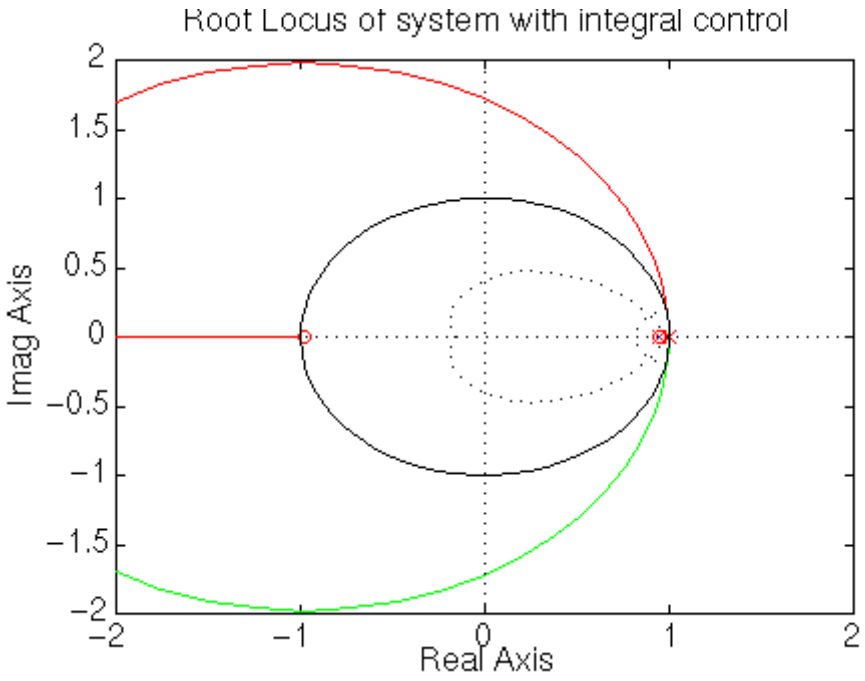
$$\xi = \frac{-\ln(OS)}{\sqrt{\pi^2 + (\ln(OS)^2)}}$$

$$\omega_n = \frac{4}{T_{S,2\%}}$$

The required damping ratio is 0.5 and the natural frequency is 200 rad/sec, but the **"zgrid"** command requires a non-dimensional natural frequency. Therefore Wn = 200*Ts = 0.2 rad/sample. Add the following MATLAB code into the end of your m-file and rerun it.

```
rlocus(icontr*motor_d);
zgrid(0.5,0.2)
title('Root Locus of system with integral control')
axis([-2,2,-2,2])
```

You should get the following plot:

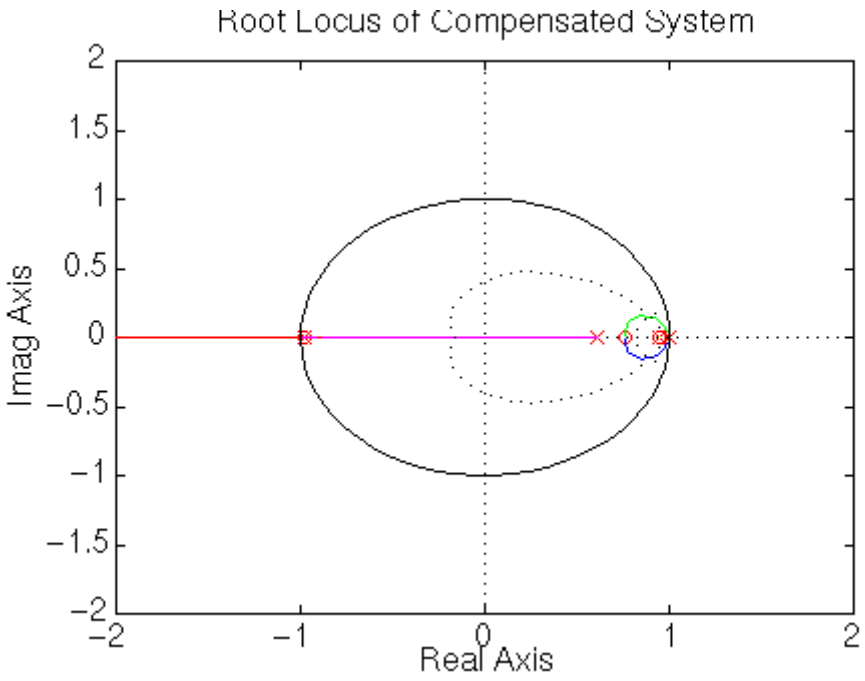


From the above root locus plot, we can see that system is unstable at all gains because the root locus is outside the unit circle. Moreover, the root locus should be in the region where the damping ratio line and natural frequency cross each other to satisfy the design requirements. Thus we have to pull the root locus in further by first canceling the zero at approximately -0.98, since this zero will add overshoot to the step response. Then we have to add one more pole and two zeroes near the desired poles. After going through some trial and error, one more pole is added at 0.61 and two zeroes are added at 0.76. Add the following commands to your m-file and rerun it in the MATLAB window.

```
numc = conv([1 -0.76],[1 -0.76]);
denc = conv([1 0.9831],[1 -0.61]);
contr = tf(numc,denc,Ts);
```

```
rlocus(icontr*contr*motor_d);
zgrid(0.5,0.2)
title('Root Locus of Compensated System')
axis([-2,2,-2,2])
```

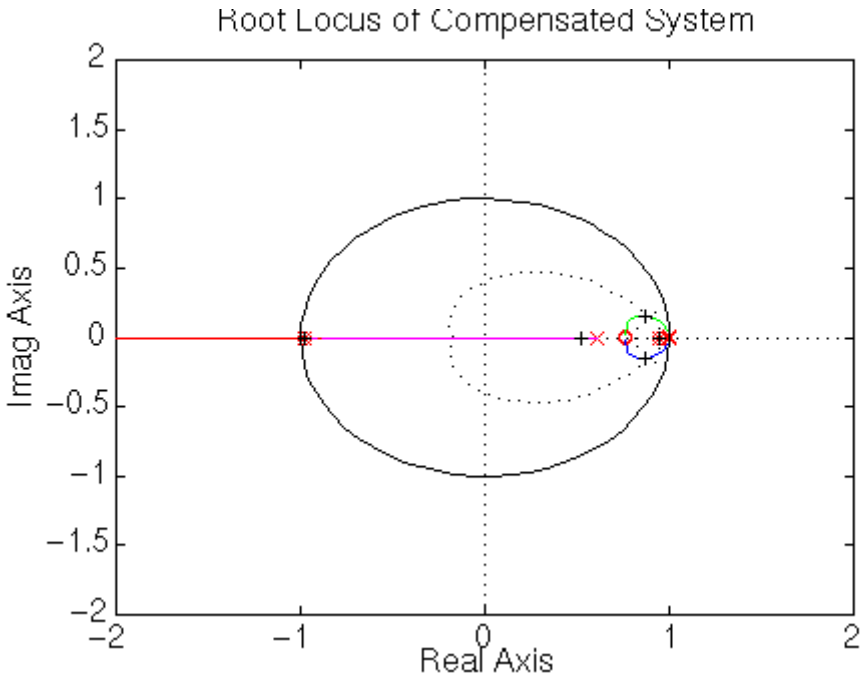
The system will have a pole at 0.61 instead of -0.98. You should get the following root locus plot:



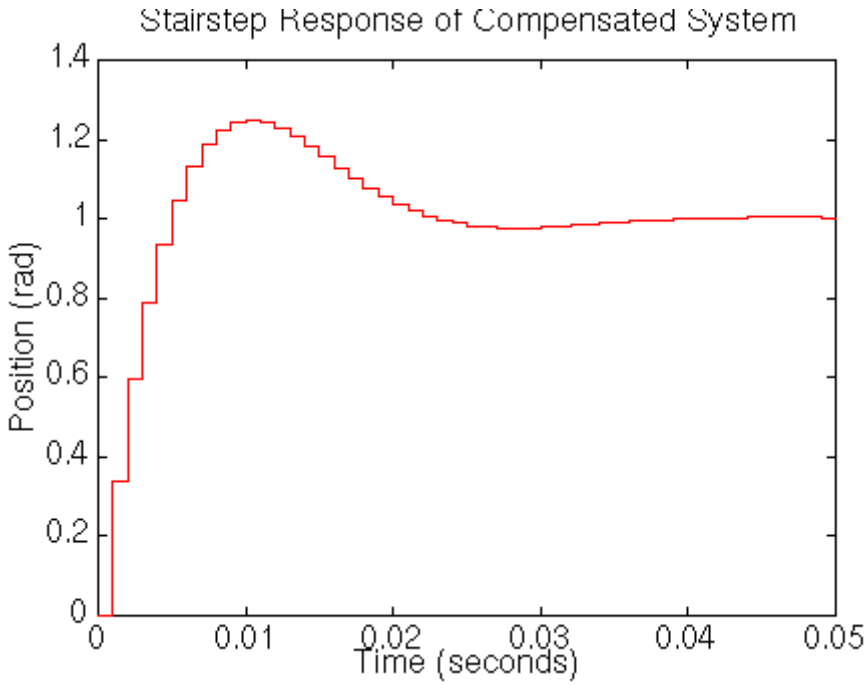
From the above root locus plot, we see that the root locus is in the desired region. Let's find a gain, K, on the root locus plot by using the **"rlocfind"** command and obtain the stairstep response with the selected gain. Enter the following commands at the end of your m-file and rerun it.

```
K = rlocfind(icontr*contr*motor_d)
sys_cl = feedback(K*icontr*contr*motor_d,1);

[x2,t] = step(sys_cl,.05);
stairs(t,x2)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Stairstep Response of Compensated System')
```

In the MATLAB window, you should see the command asking you to select a point on the root-locus plot. You should click on the plot as the following:



The selected gain should be around 330, and it will plot the closed-loop compensated response as follows.

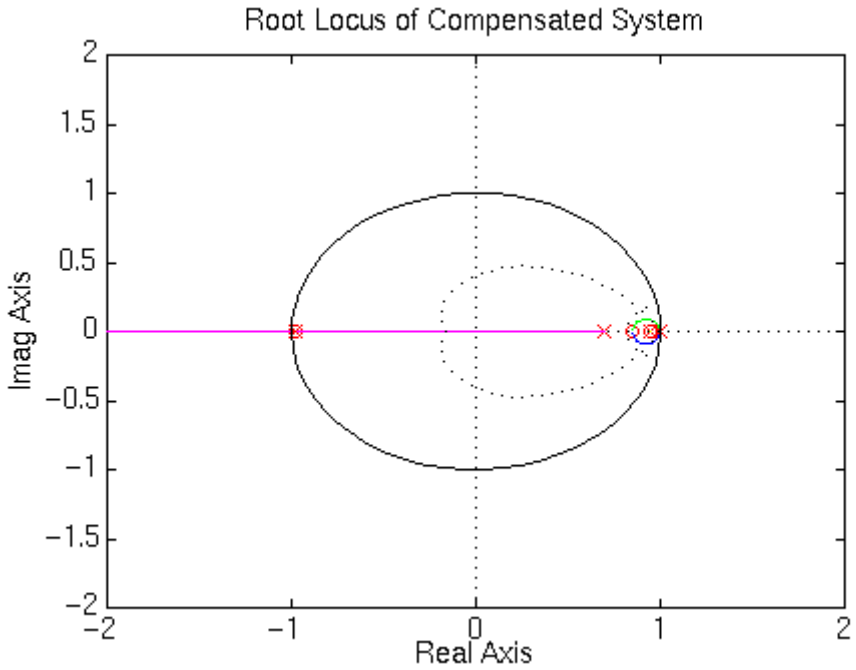


From the above closed-loop response, the settling time is about 0.05 seconds which is satisfy the requirement, but the percent overshoot is 22% which is too large due to the zeroes. If we select the gain to be larger, the requirements will be satisfied. On the other hand, the problem will be unrealistic and a huge actuator is needed, you can try this

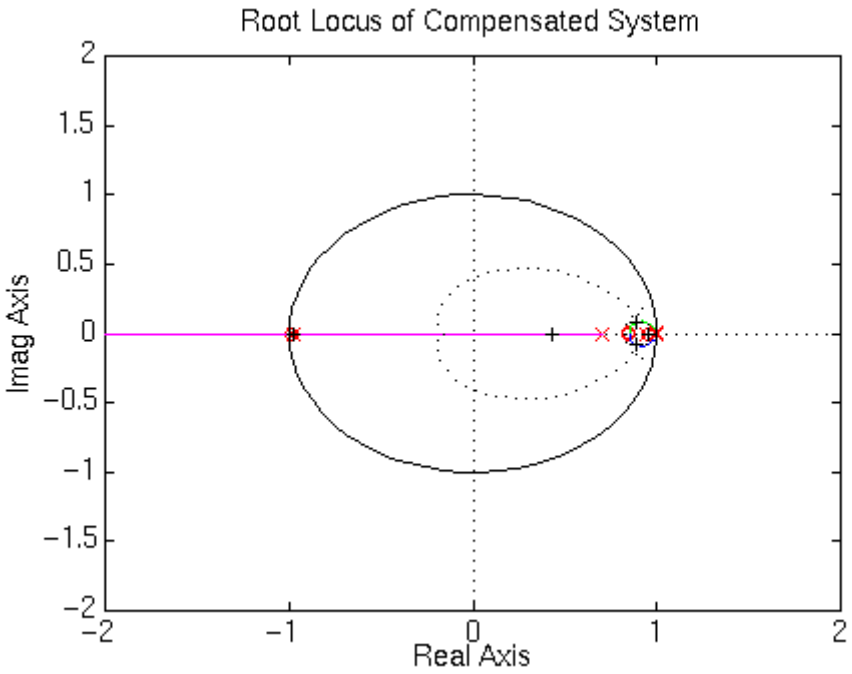
yourself by picking a larger gain on the previous root locus plot which will yield an unrealistically sudden step response. So we have to move both one pole and two zeroes a little further to the right to pull root locus in a little bit more. The new pole will be put at 0.7 and two zeroes will be at 0.85. Go back to your m-file and change only **"numc"** and **"denc"** as shown below and rerun it in MATLAB window.

```
numc = conv([1 -0.85],[1 -0.85]);
denc = conv([1 0.9831],[1 -0.7]);
```

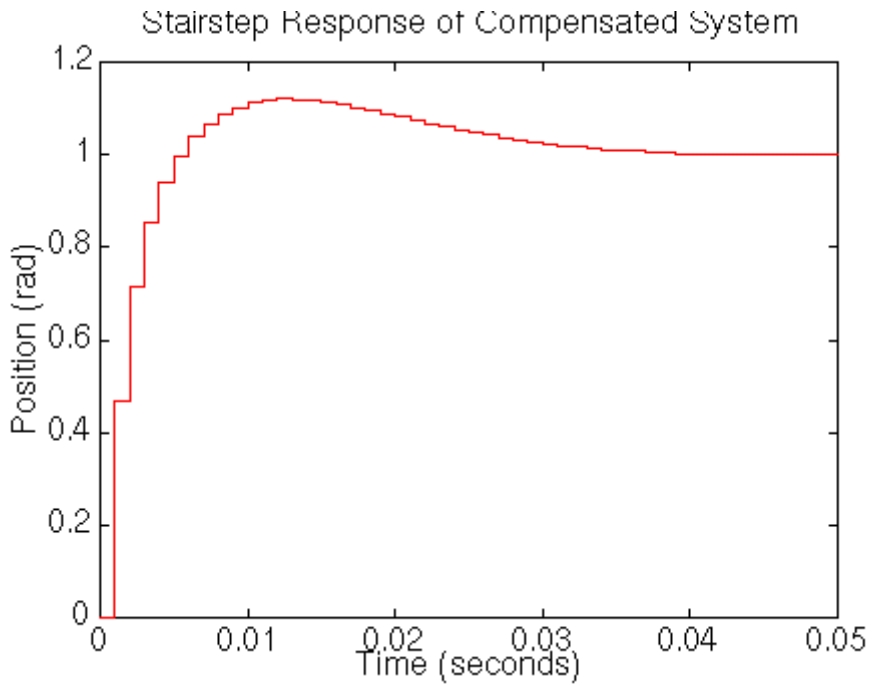
Then you should see the following root locus plot.



On the new root locus, you should click on the plot to select a new gain as the following:



The selected gain should be around 450, and then it will plot the closed-loop compensated response as follows:

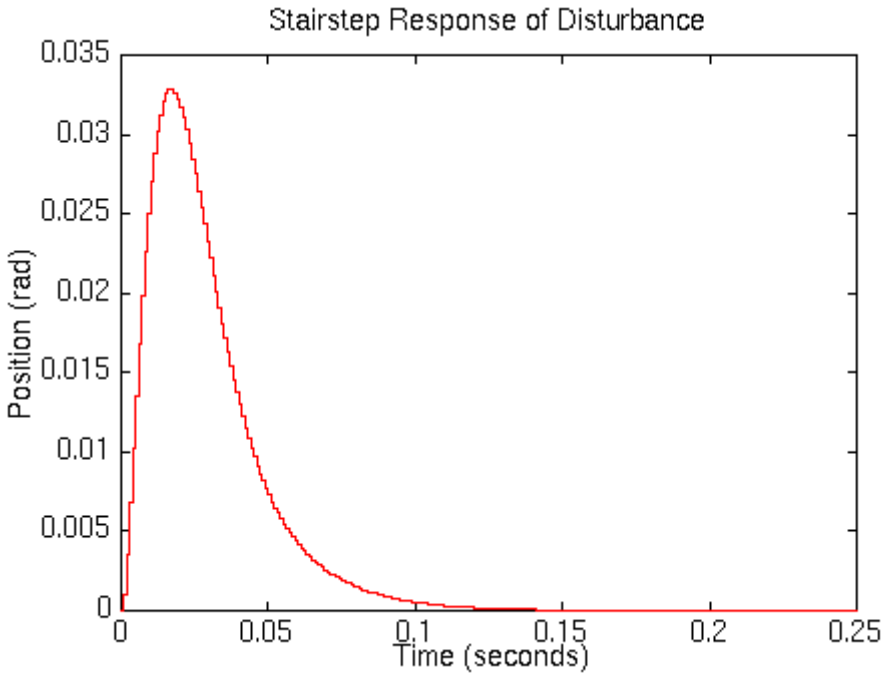


Now we see that the settling time and percent overshoot meet the design requirements of the system. The settling time is 0.04 seconds and the percent overshoot is about 10%.

Now let's take a look at a disturbance response of the closed-loop system. We need to cancel the selected gain, the integral transfer function and controller's transfer function from the closed-loop transfer function. So add the following code into your m-file and rerun it.

```
dist_cl=feedback(motor_d,K*icontr*contr);

[x4,t] = step(dist_cl,.25);
stairs(t,x4)
xlabel('Time (seconds)')
ylabel('Position (rad)')
title('Stairstep Response of Compensated System')
```

MATLAB should return the following plot:



We can see that a response to the disturbance is small (3.3% of the disturbance) and settles within 2% of the disturbance after 0.04 seconds and eventually reaches zero.

CISE 316

Control Systems Design

Lab Experiment 4: Position Control of DC Motor

Objectives: This is the experimental validation of experiment 3. Main objective of this experiment is to utilize the analysis and design knowledge of Root Locus design for the experimental set up of DC Motor's Position Control. Students will obtain a model for a DC motor Position control and then design & implement a suitable digital controller on the DC Motor.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW
- DC Servo System (feedback equipment)
    a. OU150A Op Amp Unit
    b. AU150B Attenuator Unit
    c. PA150C Pre-Amplifier Unit
    d. SA150D Servo Amplifier
    e. PS150E Power Supply
    f. DCM150F DC Motor
    g. IP150H Input Potentiometer
    h. OP150K Output Potentiometer
    i. GT150X Reduction Gear Tacho
    j. DC Voltmeter

Category Software-Hardware Experiment

Deliverables

A complete lab report including the following:

- Completed sections according to the instructions given in each section below
- Figures with plots of closed-loop step responses and control efforts.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs.
- Plots of Software Codes

Design of Experiment:

Introduce the method to be used for the model identification for speed control of DC motor. How to conduct the experiment and which input/output data is to be collected from the experimental set up. Which plots are to be plotted and what are the settings like sampling time etc.

Equipment:

Instructions for the students:
List down the equipment to be used in this experiment.

Hardware Connections:

Instructions for the students:
Mention the Connection diagram of the DAQ interfacing device with the Input/Output of the DC motor system. Mention the connections for signal generator, Multi-meter, Oscilloscope or other equipment as required and used.

Procedure:

Instructions for the students:

Introduce the overall procedure and strategy to conduct the experiment. Steps should be mentioned properly. Sequence of procedure can be followed as:

- Identify what signals/data are required for the analysis and design of Position control of DC motor.
- Introduce the overall hardware setup and connections.
- Write the LabVIEW program to collect, archive and display the data.
- Do analysis of the archive data in LabVIEW or MATLAB. MATLAB is preferred.
- From the analysis, and with the help of Design theory of the related method, Design a Digital Controller.
- Program the Digital Controller in LabVIEW and implement it on the experimental setup.
- During the procedure followed, students should take care of the tables and graphs to be shown properly as mentioned below:
  - Table of Data Collected: Show the table of data collected in excel.

- Analysis: Show the working on the crude data collected. Any mathematical manipulation should be shown in tabulated or graphical form e.g., logarithm of the data values or related graphs.
- Results: Show the conclusive milestones like arriving at the transfer function after analysis.
- Controller Design:
    - Mention the required specifications for the controller design.
    - Stepwise description of controller design and calculation and related plots should be mentioned properly.
    - Mention the Controller parameters after design steps
    - Implement and show the results.
- Conclusion and Comments:
    - Conclude the overall method, analysis and design of the controller and system.
    - Key aspects of the analysis and design should be highlighted.
    - Comment about the overall understanding and learning outcome.

CISE 316

Control Systems Design

Lab Experiment 5: Introduction to Bode Plots, Phase and Gain Margin

Objective: In this lab, concepts of Bode Plots, Phase Margin and Gain Margin will addressed using MATLAB.

Introduction:
The frequency response method may be less intuitive than other methods you have studied previously. However, it has certain advantages, especially in real-life situations such as modeling transfer functions from physical data.

The frequency response of a system can be viewed two different ways: via the Bode plot or via the Nyquist diagram. Both methods display the same information; the difference lies in the way the information is presented. We will explore both methods during this lab exercise.

The frequency response is a representation of the system's response to sinusoidal inputs at varying frequencies. The output of a linear system to a sinusoidal input is a sinusoid of the same frequency but with a different magnitude and phase. The **frequency response** is defined as the magnitude and phase differences between the input and output sinusoids. In this lab, we will see how we can use the open-loop frequency response of a system to predict its behavior in closed-loop.

To plot the frequency response, we create a vector of frequencies (varying between zero or "DC" and infinity i.e., a higher value) and compute the value of the plant transfer function at those frequencies. If G(s) is the open loop transfer function of a system and ω is the frequency vector, we then plot $G(j\omega)$ vs. ω. Since $G(j\omega)$ is a complex number, we can plot both its magnitude and phase (the Bode plot) or its position in the complex plane (the Nyquist plot).

List of Equipment/Software

Following equipment/software is required:

- MATLAB

Category Soft Experiment

Deliverables

A complete lab report including the following:

- Figures with plots of all the frequency responses along with the MATLAB codes.

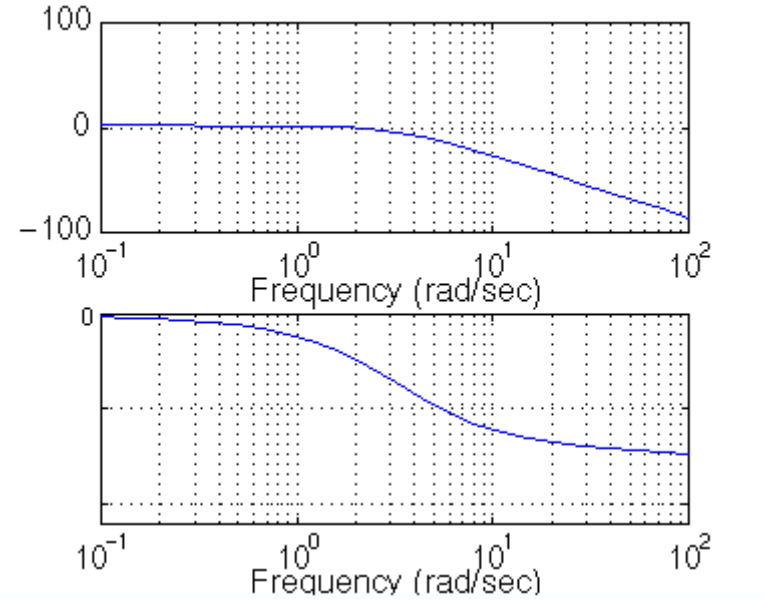
Bode Plots

As noted above, a Bode plot is the representation of the magnitude and phase of $G(j\omega)$ (where the frequency vector ω contains only positive frequencies). To see the Bode plot of a transfer function, you can use the MATLAB "**bode**" command. For example,

```
num = 50;
den = [1 9 30 40];
sys = tf(num,den);
bode(sys)
```

This displays the Bode plots for the transfer function:

$$\frac{50}{s^3 + 9s^2 + 30s + 40}$$

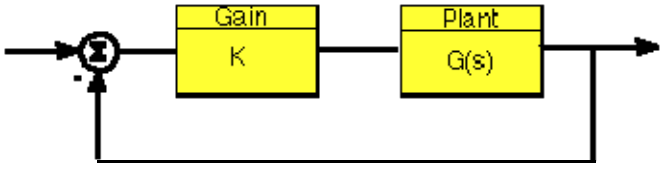


Please note the axes of the figure. The frequency is on a logarithmic scale, the phase is given in degrees, and the magnitude is given as the gain in decibels.

Note: a decibel is defined as 20*log10 (|G(j*w|)

Gain and Phase Margin

Let's say that we have the following system:



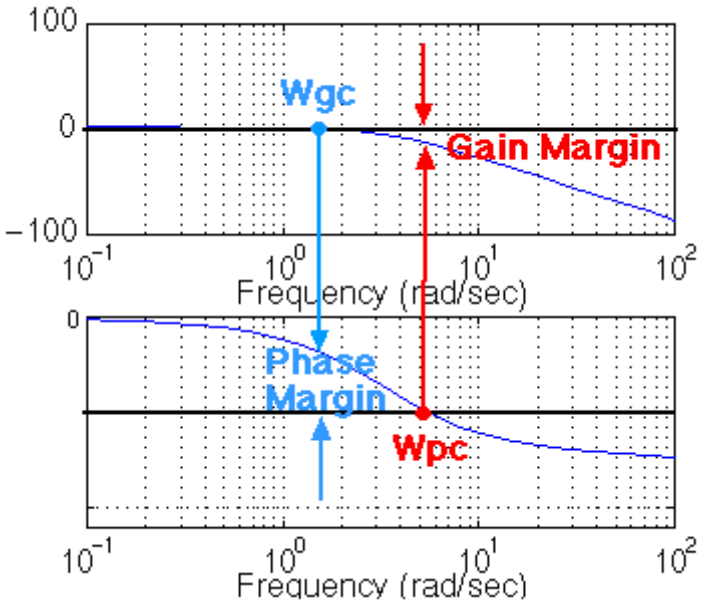
where K is a variable (constant) gain and G(s) is the plant under consideration. The **gain margin** is defined as the change in open loop gain required to make the system unstable. Systems with greater gain margins can withstand greater changes in system parameters before becoming unstable in closed loop.

Note: Unity gain in magnitude is equal to a gain of zero in dB.

The **phase margin** is defined as the change in open loop phase shift required to make a closed loop system unstable.

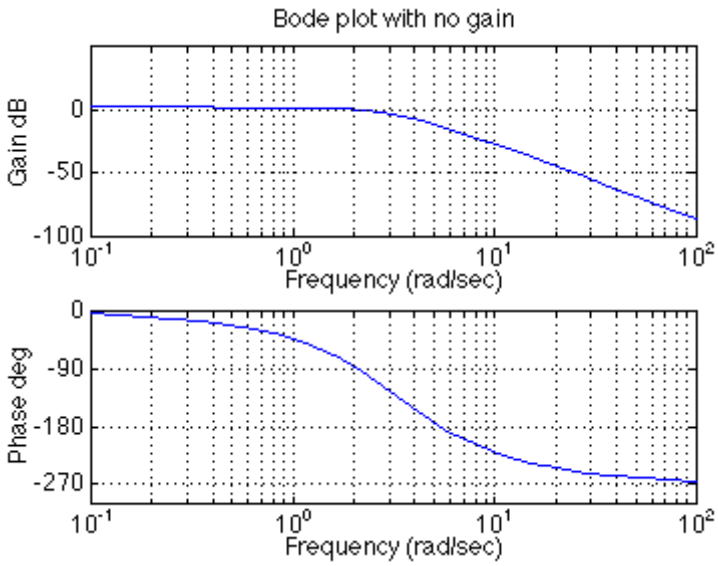
The phase margin also measures the system's tolerance to time delay. If there is a time delay greater than $180/\omega_{PC}$ in the loop (where ω_{PC} is the frequency where the phase shift is 180 deg), the system will become unstable in closed loop. The time delay can be thought of as an extra block in the forward path of the block diagram that adds phase to the system but has no effect on the gain. That is, a time delay can be represented as a block with magnitude of 1 and phase ω*time_delay (in radians/second).

The phase margin is the difference in phase between the phase curve and -180 deg at the point corresponding to the frequency that gives a gain of 0dB (the gain cross over frequency, ω_{gc}). Likewise, the gain margin is the difference between the magnitude curve and 0dB at the point corresponding to the frequency that gives a phase of -180 deg (the phase cross over frequency, ω_{PC}).

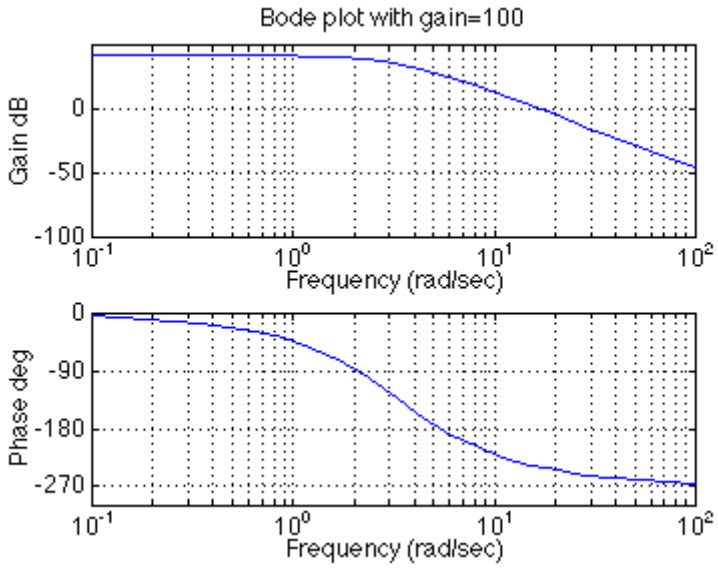


When changing the gains, the phase margin plots do not require to update all Bode curves in order to find the new phase margin. Multiplying the system by a gain shifts the magnitude plot up or down. Finding the phase margin is a simple matter of finding the new cross-over

frequency. For example, suppose you entered the command **bode(sys).** You will get the following bode plot:



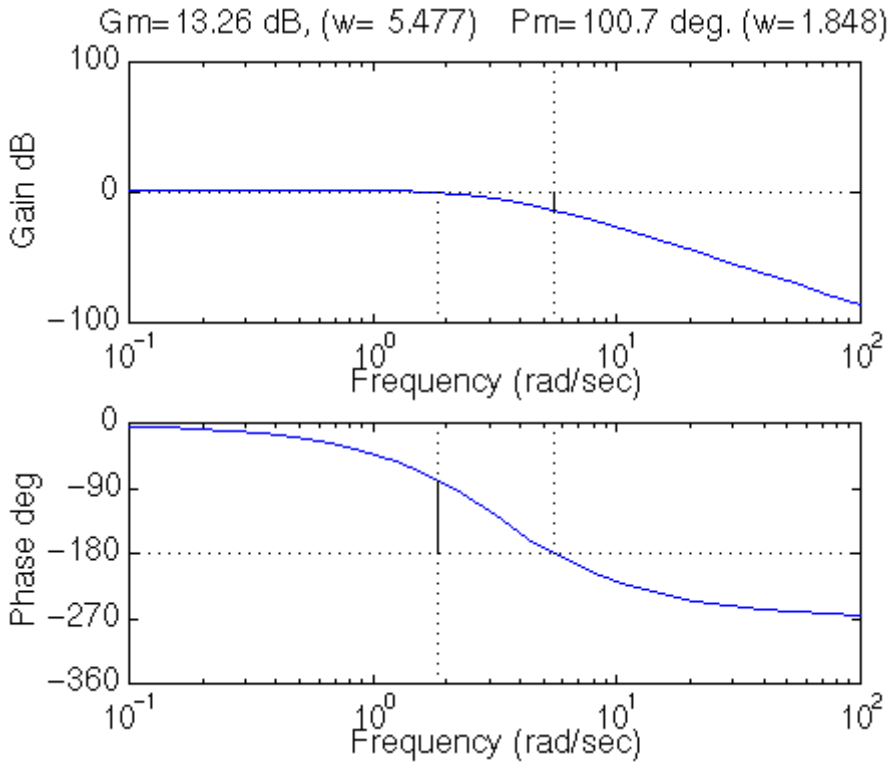
You should see that the phase margin is about 100 degrees. Now suppose you added a gain of 100, by entering the command **bode(100*sys).** You should get the following plot (note we changed the axis so the scale would be the same as the plot above, your bode plot may not be exactly the same shape, depending on the scale used):



As you can see the phase plot is exactly the same as before, and the magnitude plot is shifted up by 40dB (gain of 100). The phase margin is now about -60 degrees. This same result could be achieved if the y-axis of the magnitude plot was shifted down 40dB. Try this, look at the first Bode plot, find where the curve crosses the -40dB line, and read off the phase margin. It should be about -60 degrees, the same as the second Bode plot.

We can find the gain and phase margins for a system directly, by using MATLAB. Just use the margin command. This command returns the gain and phase margins, the gain and phase cross over frequencies, and a graphical representation of these on the Bode plot. MATLAB can determine the phase margin using **margin(sys)**

```
>>margin(sys)
```



Bandwidth Frequency

The bandwidth frequency is defined as the frequency at which the **closed-loop** magnitude response is equal to -3 dB. However, when we design via frequency response, we are interested in predicting the closed-loop behavior from the open-loop response. Therefore, we will use a second-order system approximation and say that the bandwidth frequency equals the frequency at which the **open-loop** magnitude response is between -6 and - 7.5dB, assuming the open loop phase response is between -135 deg and -225 deg. For a complete derivation of this approximation, consult your textbook.

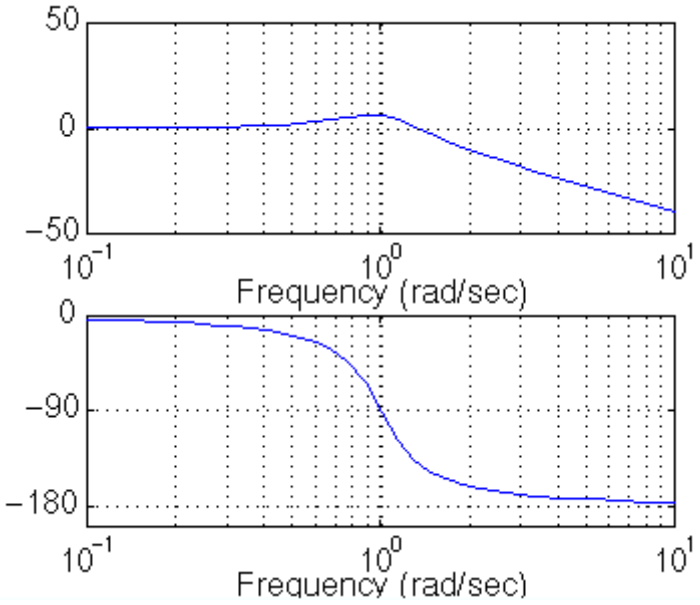
In order to illustrate the importance of the bandwidth frequency, we will show how the output changes with different input frequencies. We will find that sinusoidal inputs with frequency less than Wbw (the bandwidth frequency) are tracked "reasonably well" by the system. Sinusoidal inputs with frequency greater than Wbw are attenuated (in magnitude) by a factor of 0.707 or greater (and are also shifted in phase).

Let's say that we have the following closed-loop transfer function representing a system:

$$\frac{1}{s^2 + 0.5\,s + 1}$$

First of all, let's find the bandwidth frequency by looking at the Bode plot:

```
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
bode (sys)
```

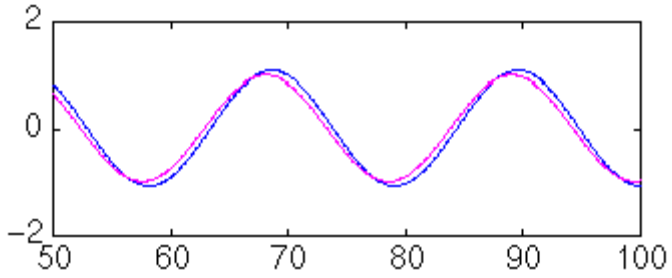


Since this is the closed-loop transfer function, our bandwidth frequency will be the frequency corresponding to a gain of -3 dB. looking at the plot, we find that it is approximately 1.4 rad/s. We can also read off the plot that for an input frequency of 0.3 radians, the output sinusoid should have a magnitude about one and the phase should be shifted by perhaps a few degrees (behind the input). For an input frequency of 3 rad/sec, the output magnitude should be about -20dB (or 1/10 as large as the input) and the phase should be nearly -180 (almost exactly out-of-phase). We can use the **"lsim"** command to simulate the response of the system to sinusoidal inputs.

First, consider a sinusoidal input with a **frequency lower than Wbw**. We must also keep in mind that we want to view the steady state response. Therefore, we will modify the axes in order to see the steady state response clearly (ignoring the transient response).

```
w = 0.3;
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
t = 0:0.1:100;
u = sin(w*t);
[y,t] = lsim(sys,u,t);
```

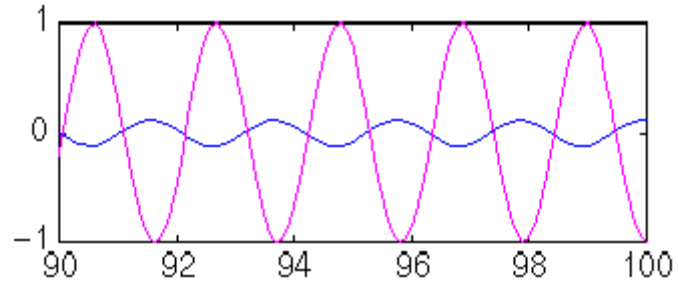
```
plot(t,y,t,u)
axis([50,100,-2,2])
```



Note that the output (blue) tracks the input (purple) fairly well; it is perhaps a few degrees behind the input as expected.

However, if we set the frequency of the input **higher than the bandwidth frequency** for the system, we get a very distorted response (with respect to the input):

```
w = 3;
num = 1;
den = [1 0.5 1];
sys = tf(num,den);
t = 0:0.1:100;
u = sin(w*t);
[y,t] = lsim(sys,u,t);
plot(t,y,t,u)
axis([90, 100, -1, 1])
```



Again, note that the magnitude is about 1/10 that of the input, as predicted, and that it is almost exactly out of phase (180 degrees behind) the input. Feel free to experiment and view the response for several different frequencies w, and see if they match the Bode plot.

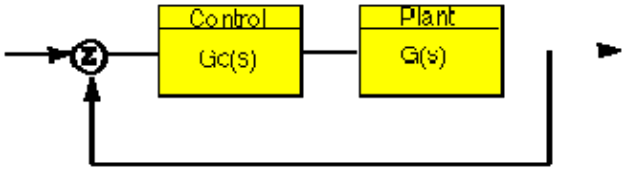
Closed-loop performance

In order to predict closed-loop performance from open-loop frequency response, we need to have several concepts clear:

- The system must be stable in open loop if we are going to design via Bode plots.

- If the **gain cross over frequency** is less than the **phase cross over frequency** (i.e. Wgc < Wpc), then the closed-loop system will be stable.
- For second-order systems, the closed-loop damping ratio is approximately equal to the phase margin divided by 100 if the phase margin is between 0 and 60 deg. We can use this concept with caution if the phase margin is greater than 60 deg.
- For second-order systems, a relationship between damping ratio, bandwidth frequency and settling time exists.
- A very rough estimate that you can use is that the bandwidth is approximately equal to the natural frequency.

Let's use these concepts to design a controller for the following system:



Where Gc(s) is the controller and G(s) is:

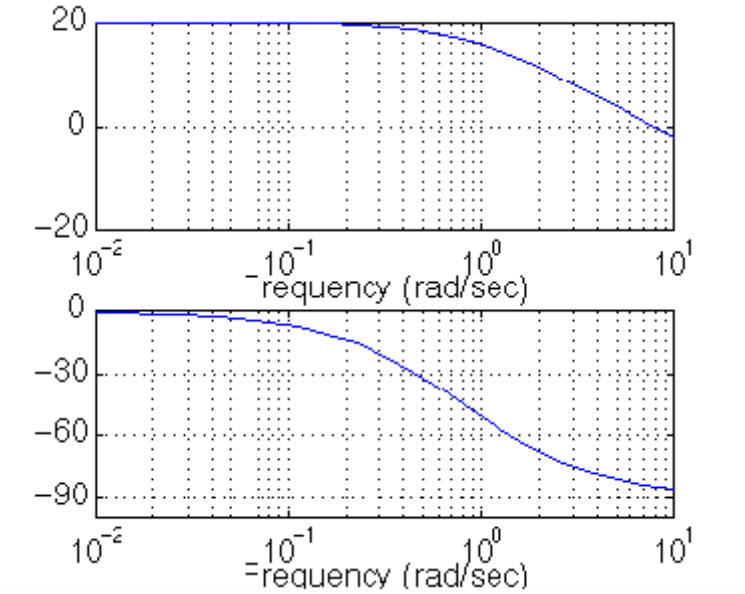
$$\frac{10}{1.25s + 1}$$

The design must meet the following specifications:

- Zero steady state error.
- Maximum overshoot must be less than 40%.
- Settling time must be less than 2 secs.

There are two ways of solving this problem: one is graphical and the other is numerical. Within MATLAB, the graphical approach is best, so that is the approach we will use. First, let's look at the Bode plot. Create an m-file with the following code:

```
num = 10;
den = [1.25,1];
sys = tf(num,den);
bode(sys)
```

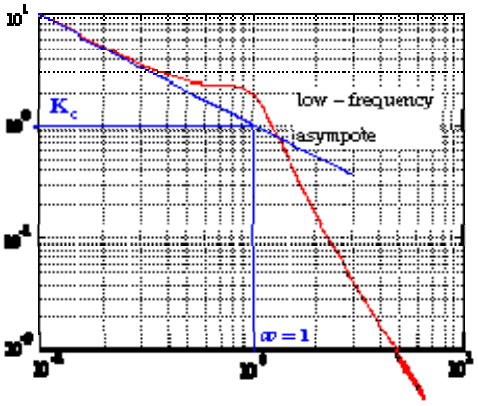


There are several characteristics of the system that can be read directly from this Bode plot. First of all, we can see that the bandwidth frequency is around 10 rad/sec. Since the bandwidth frequency is roughly the same as the natural frequency (for a first order system of this type), the rise time is 1.8/BW=1.8/10=1.8 seconds. This is a rough estimate, so we will say the rise time is about 2 seconds.

The phase margin for this system is approximately 95 degrees. The relation damping ratio = pm/100 only holds for PM < 60. Since the system is first-order, there should be no overshoot.

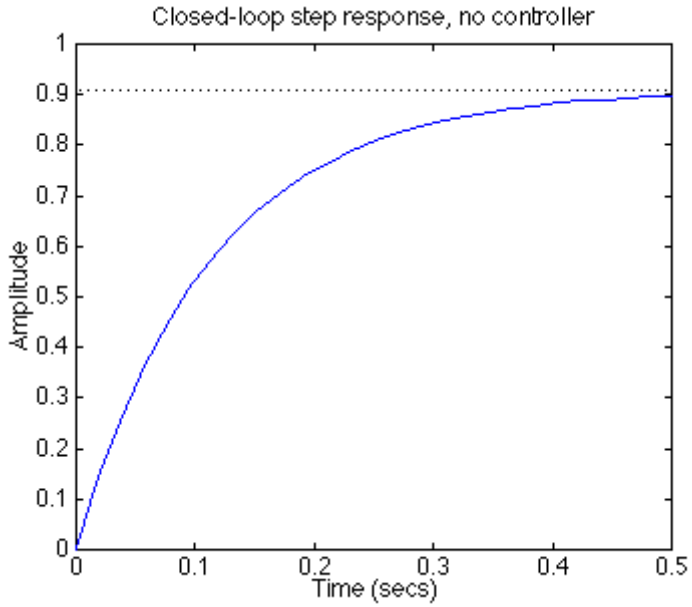
The last major point of interest is **steady-state error**. The steady-state error can be read directly off the Bode plot as well. The constant (Kp, Kv, or Ka) is found from the intersection of the low frequency asymptote with the w=1 line. Just extend the low frequency line to the w=1 line. The magnitude at this point is the constant. Since the Bode plot of this system is a horizontal line at low frequencies (slope = 0), we know this system is of type zero. Therefore, the intersection is easy to find. The gain is 20dB (magnitude 10). What this means is that the constant for the error function it 10.

The steady-state error is 1/(1+Kp)=1/(1+10)=0.091. If our system was type one instead of type zero, the constant for the steady-state error would be found in a manner similar to the following



Let's check our predictions by looking at a step response plot. This can be done by adding the following two lines of code into the MATLAB command window.

```
sys_cl = feedback(sys,1);
step(sys_cl)
```



As you can see, our predictions were very good. The system has a rise time of about 2 seconds, has no overshoot, and has a steady-state error of about 9%. Now we need to choose a controller that will allow us to meet the design criteria. We choose a PI controller because it will yield zero steady state error for a step input. Also, the PI controller has a zero, which we can place. This gives us additional design flexibility to help us meet our criteria. Recall that a PI controller is given by:

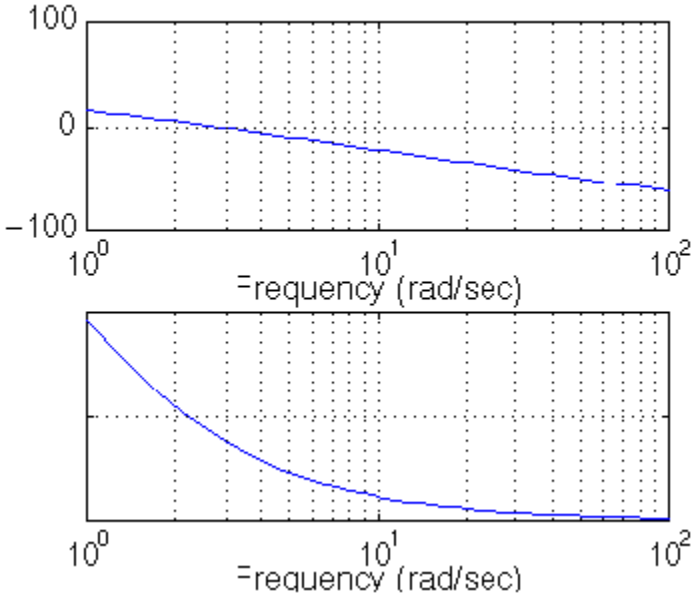
$$G_c(s) = \frac{K*(s+a)}{s}$$

The first thing we need to find is the damping ratio corresponding to a percent overshoot of 40%. Plugging in this value into the equation relating overshoot and damping ratio (or consulting a plot of this relation), we find that the damping ratio corresponding to this overshoot is approximately 0.28. Therefore, our phase margin should be at least 30 degrees. From our **Ts*Wbw vs damping ratio plot**, we find that Ts*Wbw ~ 21. We must have a bandwidth frequency greater than or equal to 12 if we want our settling time to be less than 1.75 seconds which meets the design specs.

Now that we know our desired phase margin and bandwidth frequency, we can start our design. Remember that we are looking at the open-loop Bode plots. Therefore, our bandwidth frequency will be the frequency corresponding to a gain of approximately -7 dB.

Let's see how the integrator portion of the PI or affects our response. Change your m-file to look like the following (this adds an integral term but no proportional term):

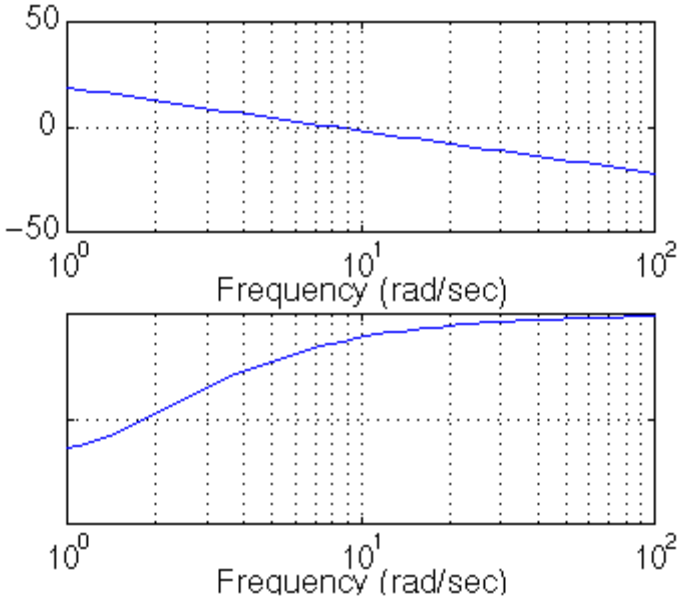
```
num = 10;
den = [1.25 1];
plant = tf(num,den);
numPI = 1;
denPI = [1 0];
contr = tf(numPI,denPI);
bode(contr * plant, logspace(0,2))
```



Our phase margin and bandwidth frequency are too small. We will add gain and phase with a zero. Let's place the zero at 1 for now and see what happens. Change your m-file to look like the following:

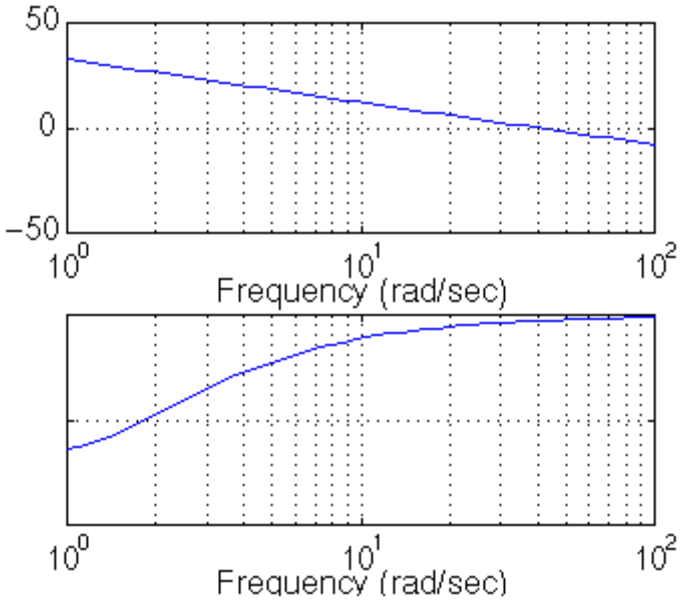
```
num = 10;
den = [1.25 1];
plant = tf(num,den);
numPI = [1 1];
```

```
denPI = [1 0];
contr = tf(numPI,denPI);
bode(contr * plant, logspace(0,2))
```



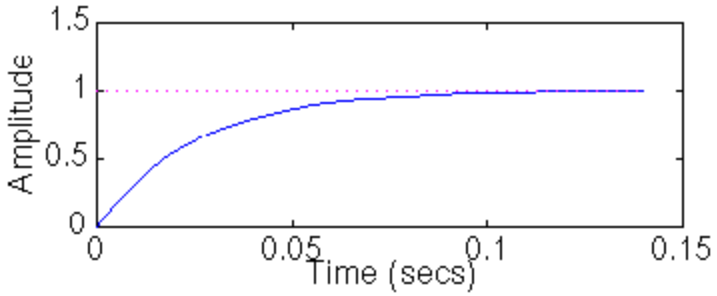
It turns out that the zero at 1 with a unit gain gives us a satisfactory answer. Our phase margin is greater than 60 degrees (even less overshoot than expected) and our bandwidth frequency is approximately 11 rad/s, which will give us a satisfactory response. Although satisfactory, the response is not quite as good as we would like. Therefore, let's try to get a higher bandwidth frequency without changing the phase margin too much. Let's try to increase the gain to 5 and see what happens. This will make the gain shift and the phase will remain the same.

```
num = 10;
den = [1.25 1];
plant = tf(num,den);
numPI = 5*[1 1];
denPI = [1 0];
contr = tf(numPI,denPI);
bode(contr * plant, logspace(0,2))
```



That looks really good. Let's look at our step response and verify our results. Add the following two lines to your m-file:

```
sys_cl = feedback(contr * plant,1);
step(sys_cl)
```



As you can see, our response is better than we had hoped for. However, we are not always quite as lucky and usually have to play around with the gain and the position of the poles and/or zeros in order to achieve our design requirements.

CISE 316

Control Systems Design

Lab Experiment 6: Control System Design Using Bode Plot – Lead, Lag and Lead-Lag Controllers

Objective: The objective is to learn frequency response design method based on Root Locus and Bode Plot to design controllers. Ball and Beam is used as the example system.

Introduction:

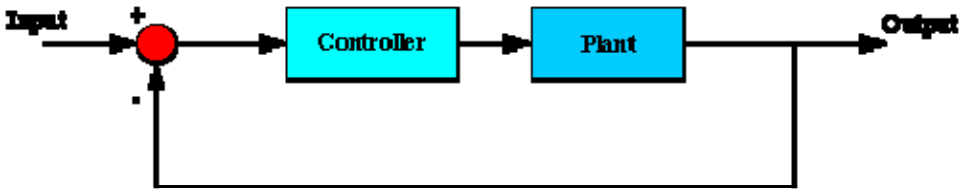
The open-loop transfer function of the plant for the **ball and beam** example is given below:

$$\frac{R(s)}{\theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\frac{1}{s^2}$$

The design criteria for this problem are:

- Settling time less than 3 seconds
- Overshoot less than 5%

A schematic of the closed loop system with a controller is given below:



List of Equipment/Software

Following equipment/software is required:

- MATLAB

Category Soft Experiment

Deliverables

A complete lab report including the following:

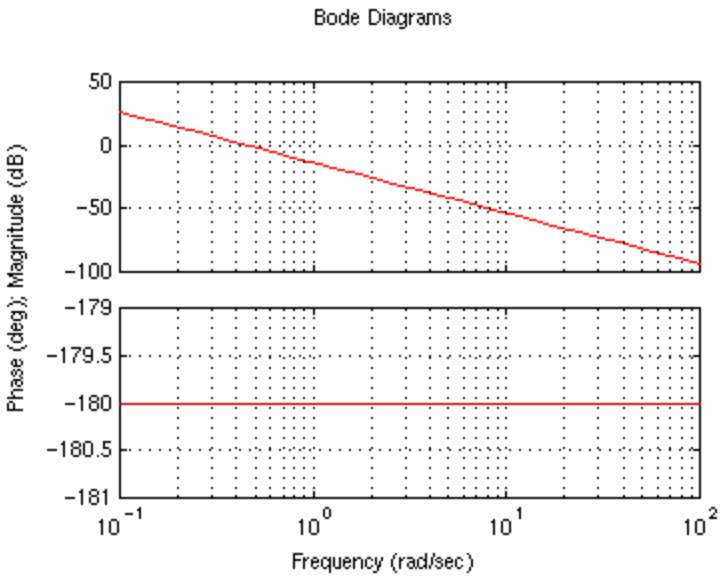
- Figures with plots of open loop and closed-loop step responses for lead and lag compensators.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs along with their specific responses and compare.
- Report properly with MATLAB codes and respective plots based on root locus and bode methods.

Open-loop Bode Plot

The main idea of frequency based design is to use the Bode plot of the open-loop transfer function to estimate the closed-loop response. Adding a controller to the system changes the open-loop Bode plot, therefore changing the closed-loop response. Let's first draw the bode plot for the original open-loop transfer function. Create an m-file with the following code and then run it in the MATLAB command window:

```
m = 0.111;
R = 0.015;
g = -9.8;
L = 1.0;
d = 0.03;
J = 9.99e-6;

K = (m*g*d)/(L*(J/R^2+m));   %simplifies input

num = [-K];
den = [1 0 0];
ball=tf(num,den);

bode(ball)
```

You should get the following Bode plot:



From this plot we see that the phase margin is zero. Since the phase margin is defined as the change in open-loop phase shift necessary to make a closed-loop system unstable this means that our zero phase margin indicates our system is unstable. We want to increase the phase margin and we can use a lead compensator controller to do this.

Phase-Lead Controller

A first order phase-lead compensator has the form given below:

$$G(s)=K\left(\frac{1+Ts}{1+aTs}\right)$$

The **phase-lead compensator** will add positive phase to our system over the frequency range 1/aT and 1/T, which are called the corner frequencies. The maximum added phase for one lead compensator is 90 degrees. For our controller design we need a percent overshoot of less than 5 %, which corresponds to a zeta of 0.7. Generally zeta * 100 will give you the minimum phase margin needed to obtain your desired overshoot. Therefore we require a phase margin greater than 70 degrees.

To obtain "T" and "a", the following steps can be used.

1. Determine the positive phase needed:

We need at least 70 degrees from our controller.

2. Determine the frequency where the phase should be added (center frequency):

In our case this is difficult to determine because the phase vs. frequency graph in the bode plot is a flat line. However, we have a relation between bandwidth frequency (wbw)

and settling time which tells us that wbw is approximately 1.92 rad/s. Therefore we want a center frequency just before this. For now we will choose 1.

3. Determine the constant "a" from the equation below, this determines the required space between the zero and the pole for the maximum phase added.

$$a = \frac{1-\sin\phi}{1+\sin\phi}$$

where phi refers to the desired phase margin. For 70 degrees, a = 0.0311.

4. Determine "T" and "aT" from the following equations:

$$T = \frac{1}{w\sqrt{a}}$$

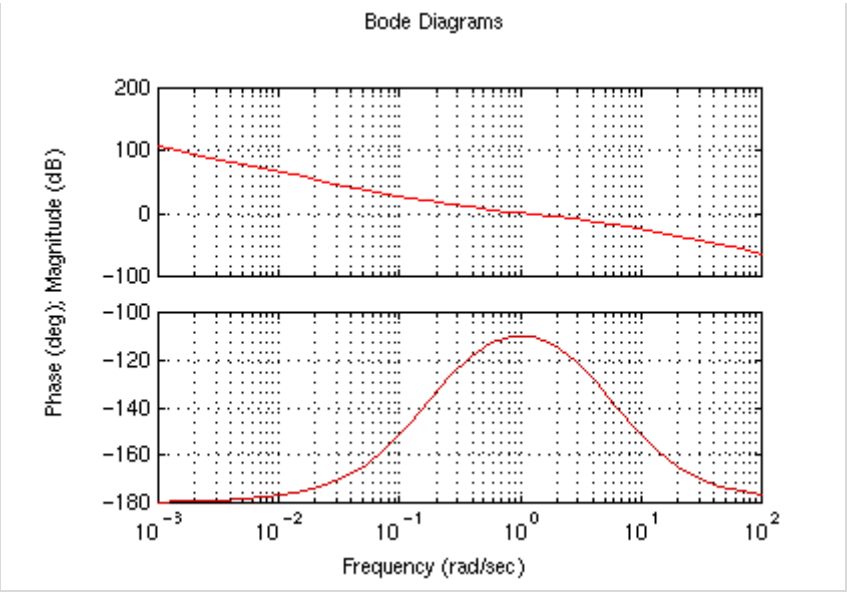
$$a\,T = \frac{\sqrt{a}}{w}$$

For 70 degrees and center frequency (w) = 1, aT = 0.176 and T = 5.67

Now, we can add our lead controller to the system and view the bode plot. Remove the bode command from your m-file and add the following:

```
phi=70*pi/180;
a=(1-sin(phi))/(1+sin(phi));
w=1;
T=1/(w*sqrt(a));
k = 1;
numlead = k*[T 1];
denlead = [a*T 1];
contr = tf(numlead,denlead);

bode(contr*ball)
```

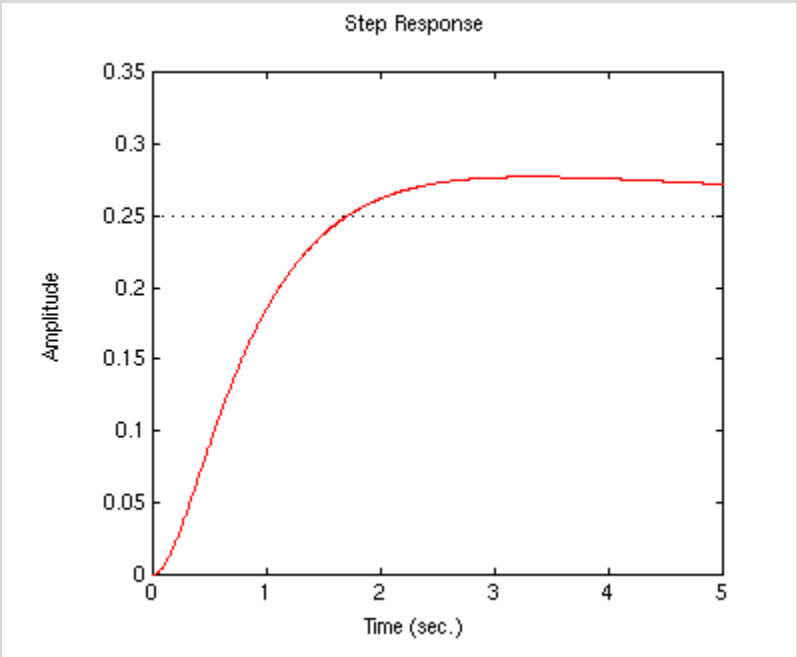
You should get the following bode plot:



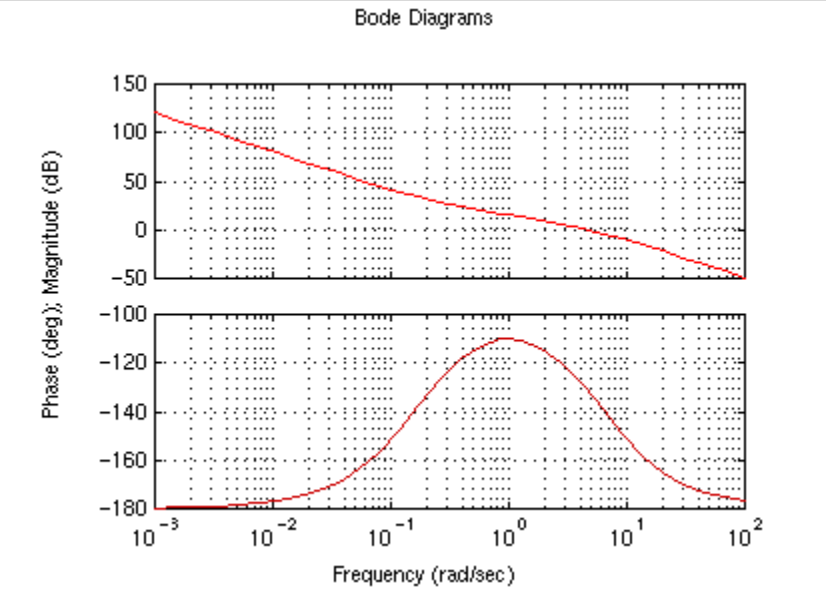
You can see that our phase margin is now 70 degrees. Let's check the closed-loop response to a step input of 0.25m. Add the following to your m-file:

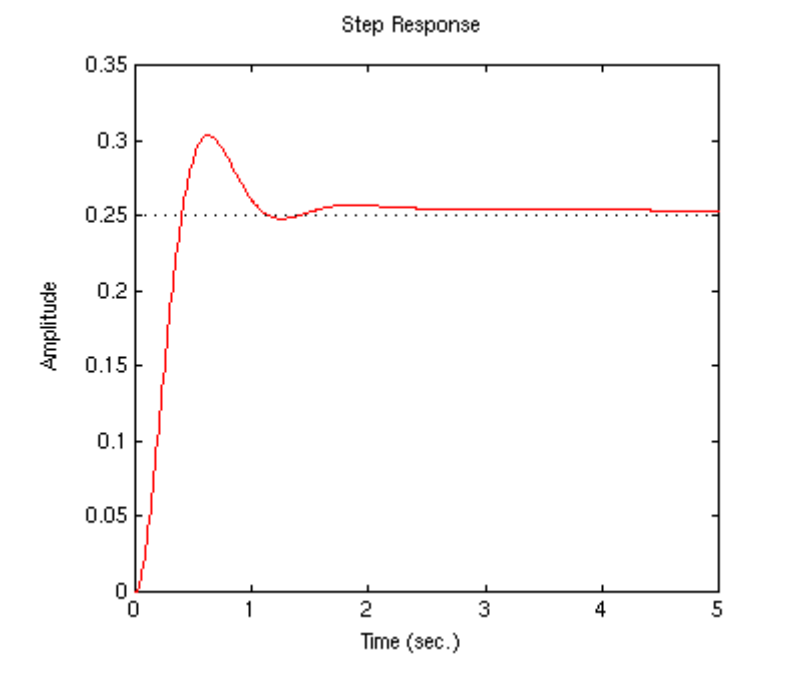
```
sys_cl = feedback(contr*ball,1);
t = 0:0.01:5;
step(0.25*sys_cl,t)
```

You should get the following plot:



Although the system is now stable and the overshoot is only slightly over 5%, the settling time is not satisfactory. Increasing the gain will increase the crossover frequency and make the response faster. With k = 5, your response should look like:





The response is faster, however, the overshoot is much too high. Increasing the gain further will just make the overshoot worse.

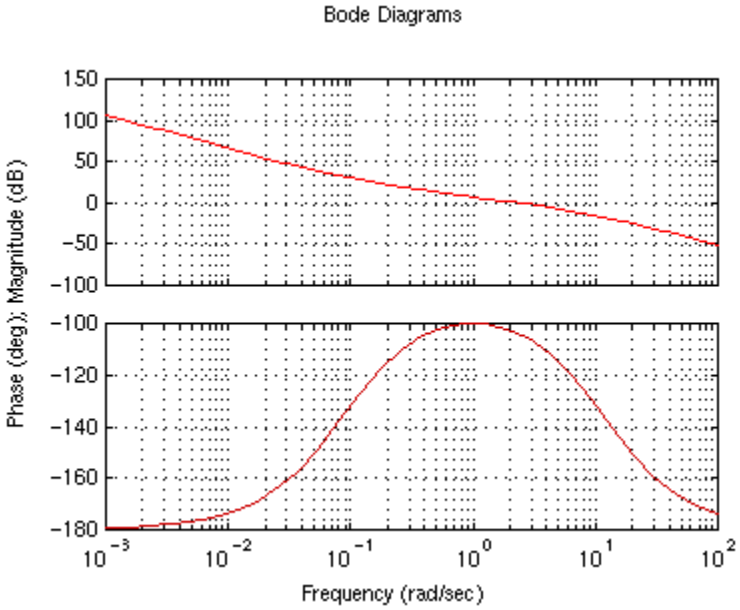
Adding More Phase

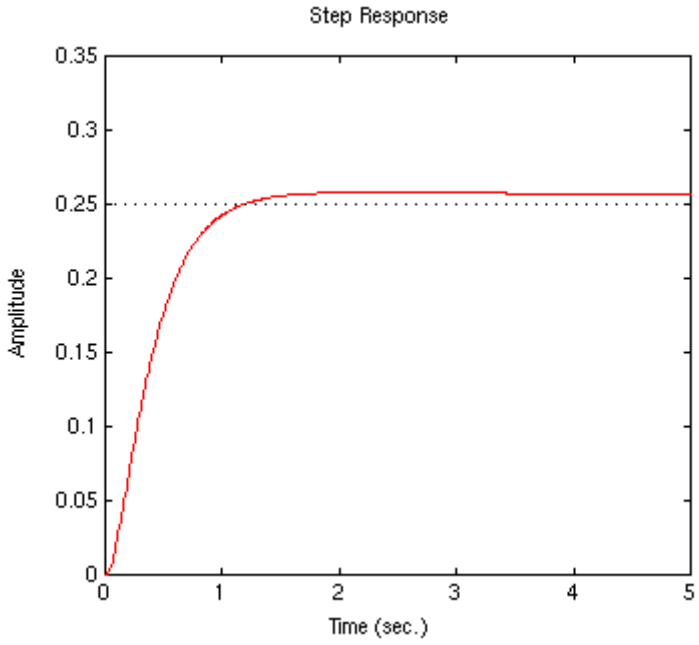
We can increase our phase-lead compensator to decrease the overshoot. In order to make the iterative process easier use the following program.

```
function[ ] = phaseball()

%define TF
m = 0.111;
R = 0.015;
g = -9.8;
L = 1.0;
d = 0.03;
J = 9.99e-6;

K = (m*g*d)/(L*(J/R^2+m));   %simplifies input

num = [-K];
den = [1 0 0];
ball = tf(num,den);

%ask user for controller information
pm = input('Phase Margin?.......');
w  = input('Center Frequency?...');
k  = input('Gain?.............');

%view compensated system bode plot
pmr = pm*pi/180;
a = (1 - sin(pmr))/(1+sin(pmr));
T = sqrt(a)/w;
aT = 1/(w*sqrt(a));

numlead = k*[aT 1];
denlead = [T 1];
contr = tf(numlead,denlead);

figure
bode(contr*ball)

%view step response
sys_cl = feedback(contr*ball,1);
t = 0:0.01:5;
figure
step(0.25*sys_cl,t)
```

With this m-file you can choose the phase margin, center frequency, and gain. Run your m-file with the following values and you should see the plots below on your screen.

Phase Margin?.....80
Center Frequency?...1
Gain?.....1

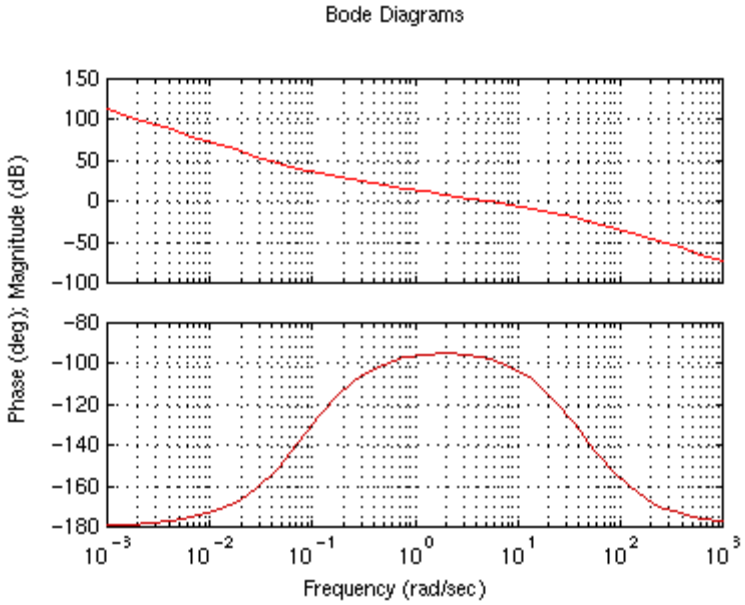


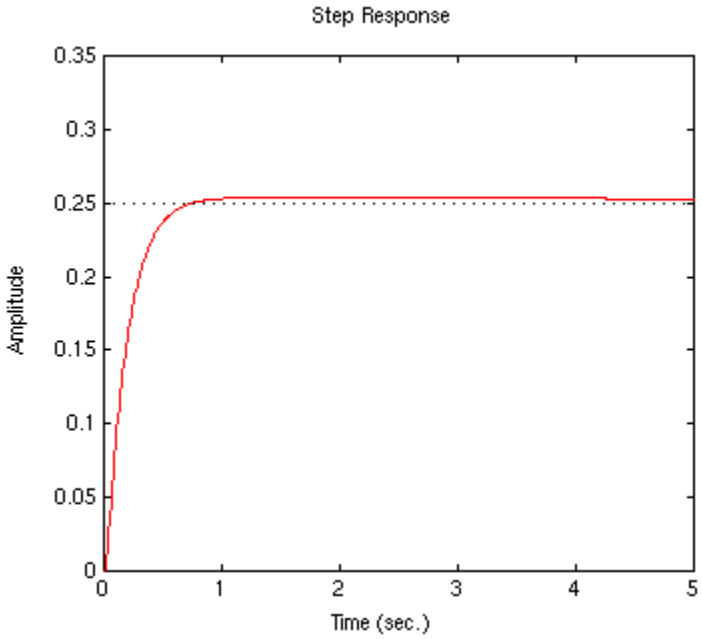


The overshoot is fine but the settling time is just a bit long. Try different numbers and see what happens.

Using the following values the design criteria was met.

Phase Margin?.....85
Center Frequency?...1.9
Gain?.....2





Note: A design problem does not necessarily have a unique answer. Using this method (or any other) may result in many different compensators. For practice you may want to go back and change the added phase, gain, or center frequency.

Designing Lead and Lag Compensators

Lead and lag compensators are used quite extensively in control. A lead compensator can increase the stability or speed of response of a system; a lag compensator can reduce (but not eliminate) the steady state error. Depending on the effect desired, one or more lead and lag compensators may be used in various combinations.

Lead, lag, and lead/lag compensators are usually designed for a system in transfer function form.

Lead or phase-lead compensator using root locus

A first-order lead compensator can be designed using the root locus. A lead compensator in root locus form is given by

$$G(s) = K_C \frac{(s - z_0)}{(s - p_0)}$$

where the magnitude of zo is less than the magnitude of po. A phase-lead compensator tends to shift the root locus toward the left half plane. This results in an improvement in the system's stability and an increase in the response speed.

How is this accomplished? If you recall finding the asymptotes of the root locus that lead to the zeros at infinity, the equation to determine the intersection of the asymptotes along the real axis is:

$$\alpha = \frac{\sum(poles) - \sum(zeros)}{(\#\, poles) - (\#\, zeros)}$$

When a lead compensator is added to a system, the value of this intersection will be a larger negative number than it was before. The net number of zeros and poles will be the same (one zero and one pole are added), but the added pole is a larger negative number than the added zero. Thus, the result of a lead compensator is that the asymptotes' intersection is moved further into the left half plane, and the entire root locus will be shifted to the left. This can increase the region of stability as well as the response speed.

In MATLAB a phase lead compensator in root locus form is implemented by using the transfer function in the form

```
numlead = kc*[1 z];
denlead = [1 p];
lead = tf(numlead,denlead);
```

and we can interconnect it with a plant as follows:

```
sys = lead*plant;
```

Lead or phase-lead compensator using frequency response

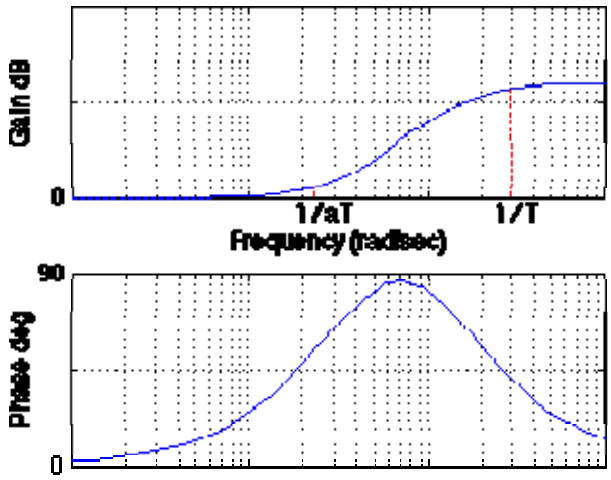
A first-order phase-lead compensator can be designed using the frequency response. A lead compensator in frequency response form is given by

$$G(s)=\frac{1+aTs}{1+Ts} \qquad (a > 1)$$

Note that this is equivalent to the root locus form

$$G(s)= K_C \frac{(s-z_0)}{(s-p_0)}$$

with p = 1/T, z = 1/aT, and Kc = a. In frequency response design, the phase-lead compensator adds positive phase to the system over the frequency range 1/aT to 1/T. A bode plot of a phase-lead compensator looks like the following



The two corner frequencies are at 1/aT and 1/T; note the positive phase that is added to the system between these two frequencies. Depending on the value of a, the maximum added phase can be up to 90 degrees; if you need more than 90 degrees of phase, two lead compensators can be used. The maximum amount of phase is added at the center frequency, which is located at

$$\omega_n = \frac{1}{T\sqrt{a}}$$

The equation which determines the maximum phase is

$$\sin\phi = \frac{a-1}{a+1}$$

Additional positive phase increases the phase margin and thus increases the stability of the system. This type of compensator is designed by determining **a** from the amount of phase needed to satisfy the phase margin requirements, and determining **T** to place the added phase at the new gain-crossover frequency.

Another effect of the lead compensator can be seen in the magnitude plot. The lead compensator increases the gain of the system at high frequencies (the amount of this gain is equal to a). This

can increase the crossover frequency, which will help to decrease the rise time and settling time of the system.
In MATLAB, a phase lead compensator in frequency response form is implemented by using the transfer function in the form

```
numlead = kc*[1 z];
denlead = [1 p];
lead = tf(numlead,denlead);
```

and we can interconnect it with a plant as follows:

```
sys = lead*plant;
```

Lag or Phase-Lag Compensator using Root Locus

A first-order lag compensator can be designed using the root locus. A lag compensator in root locus form is given by

$$G(s)=\frac{(s-z_0)}{(s-p_0)}$$

where the magnitude of zo is greater than the magnitude of po. A phase-lag compensator tends to shift the root locus to the right, which is undesirable. For this reason, the pole and zero of a lag compensator must be placed close together (usually near the origin) so they do not appreciably change the transient response or stability characteristics of the system.

How does the lag controller shift the root locus to the right? If you recall finding the asymptotes of the root locus that lead to the zeros at infinity, the equation to determine the intersection of the asymptotes along the real axis is:

$$\alpha=\frac{\sum(poles)-\sum(zeros)}{(\#\,poles)-(\#\,zeros)}$$

When a lag compensator is added to a system, the value of this intersection will be a smaller negative number than it was before. The net number of zeros and poles will be the same (one zero and one pole are added), but the added pole is a smaller negative number than the added zero. Thus, the result of a lag compensator is that the asymptotes' intersection is moved closer to the right half plane, and the entire root locus will be shifted to the right.

It was previously stated that that lag controller should only minimally change the transient response because of its negative effect. If the phase-lag compensator is not supposed to change the transient response noticeably, what is it good for? The answer is that a phase-lag compensator can improve the system's steady-state response. It works in the following manner.

At high frequencies, the lag controller will have unity gain. At low frequencies, the gain will be z0/p0 which is greater than 1. This factor z0/p0 will multiply the position, velocity, or acceleration constant (Kp, Kv, or Ka), and the **steady-state error** will thus decrease by the factor z0/p0.

In MATLAB, a phase lead compensator in root locus form is implemented by using the transfer function in the form

```
numlag = [1 z];
denlag = [1 p];
lag = tf(numlag,denlag);
```

and we can interconnect it with a plant as follows:

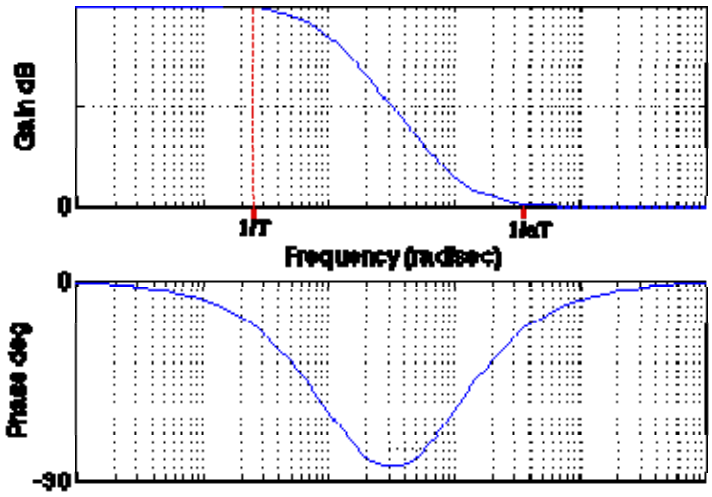
```
sys = lag*plant;
```

Lag or Phase-Lag Compensator using Frequency Response

A first-order phase-lag compensator can be designed using the frequency response. A lag compensator in frequency response form is given by

$$G(s)=\frac{1}{a}\left(\frac{1+aTs}{1+Ts}\right) \qquad (a < 1)$$

The phase-lag compensator looks similar to a phase-lead compensator, except that **a** is now less than 1. The main difference is that the lag compensator adds negative phase to the system over the specified frequency range, while a lead compensator adds positive phase over the specified frequency. A bode plot of a phase-lag compensator looks like the following



The two corner frequencies are at 1/T and 1/aT. The main effect of the lag compensator is shown in the magnitude plot. The lag compensator adds gain at low frequencies; the magnitude of this gain is equal to **a**. The effect of this gain is to cause the **steady-state error** of the closed-loop system to be decreased by a factor of **a**. Because the gain of the lag compensator is unity at middle and high frequencies, the transient response and stability are not impacted too much.

The side effect of the lag compensator is the negative phase that is added to the system between the two corner frequencies. Depending on the value **a**, up to -90 degrees of phase can be added. Care must be taken that the phase margin of the system with lag compensation is still satisfactory.

In MATLAB, a phase-lag compensator in frequency response form is implemented by using the transfer function in the form

```
numlag = [a*T 1];
denlag = a*[T 1];
lag = tf(numlag,denlag);
```

and we can interconnect it with a plant as follows:

```
sys = lag*plant;
```

Instructions for Lead-lag Compensator using either Root Locus or Frequency Response

A lead-lag compensator combines the effects of a lead compensator with those of a lag compensator. The result is a system with improved transient response, stability and steady-state error. To implement a lead-lag compensator, first design the lead compensator to achieve the desired transient response and stability, and then add on a lag compensator to improve the steady-state response.

CISE 316

Control Systems Design

Lab Experiment 7: Control System Design of DC Motor Position Control Using Lead Method

Objective: This lab is designed to design and implement lead compensators for Position control of DC motor available in the laboratory. Aim is to use both root locus and bode plot to design the lead compensator.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW
- DC Servo System (feedback equipment)
    a. OU150A Op Amp Unit
    b. AU150B Attenuator Unit
    c. PA150C Pre-Amplifier Unit
    d. SA150D Servo Amplifier
    e. PS150E Power Supply
    f. DCM150F DC Motor
    g. IP150H Input Potentiometer
    h. OP150K Output Potentiometer
    i. GT150X Reduction Gear Tacho
    j. DC Voltmeter

Category Software Hardware Experiment

Deliverables

A complete lab report including the following:

- Design work based on root locus method
- Design work and calculations based on bode plot method.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs along with their specific responses and compare.
- Report properly with MATLAB codes and respective plots based on root locus and bode methods.

Part A - Lead Compensation Techniques Based on the Root-Locus Approach.

The root-locus approach to design is very powerful when the specifications are given in terms of time-domain quantities, such as the damping ratio and undamped natural frequency of the desired dominant closed-loop poles, maximum overshoot, rise time, and setting time.

Consider a design problem in which the original system either is unstable for all values of gain or is stable but has undesirable transient-response characteristics. In such a case, the reshaping of the root locus is necessary in the broad neighborhood of

the $j\omega$ axis and the origin in order that the dominant closed-loop poles be at desired locations in the complex plane. This problem may be solved by inserting an appropriate lead compensator in cascade with the system transfer function.

The procedures for designing a lead compensator for the system shown in figure 7-6 by the root-locus method may be stated as follows:

From the performance specifications, determine the desired location for the dominant closed-loop poles.

By drawing the root-locus plot of the uncompensated system (original system), ascertain whether or not the gain adjustment alone can yield the desired closed-loop poles. If not, calculate the angle deficiency ϕ. This angle must be contributed by the lead compensator if the new root locus is to pass through the desired locations for the dominant closed-loop poles.

Assume the lead compensator $G_c(s)$ to be

$$G_c(s) = K_c \alpha \frac{T_1 + 1}{\alpha T s + 1} = K_c \frac{s + \frac{1}{T}}{s + \frac{1}{\alpha T}}, \quad (0 < \alpha < 1)$$

Where α and T are determined from the angle deficiency. k_c is determined from the requirement of the open-loop gain.

If static error constants are not specified, determine the location of the pole and zero of the lead compensator so that the lead compensator will contribute the necessary angle $\emptyset$. If no other requirements are imposed on the system, try to make the value of α as large as possible. A large value of α generally results in a larger value of k_v, which is desirable. (If a particular static error constant is specified, it is generally simpler to use the frequency-response approach.)

Determine the open-loop gain of the compensated system from the magnitude condition.

Once a compensator has been designed, check to see whether all performance specifications have been met. If the compensated system does not meet the performance specification, then repeat the design procedure by adjusting the compensator pole and zero until all such specifications are

met. If a large static error constant is required, cascade a lag network or later the lead compensator to a lag-lead compensator.

Note that if the selected dominant closed-loop poles are not really dominant, it will be necessary to modify the location of the pair of such selected dominant closed-loop poles. (The closed-loop poles other than dominant once modify the response obtained from the dominant closed-loop poles alone. The amount of modification depends on the location of these remaining closed-loop poles.) Also, the closed-loop zeros affect the response if they are located near the origin.

Part B - Lead Compensation Techniques Based on the Frequency-Response Approach

The primary function of the lead compensator is to reshape the frequency-response curve to provide sufficient phase-lead angle to offset the excessive phase lag associated with the components of the fixed system.

Specifications should be given.

Consider the system shown in figure 9-5. Assume that the performance specifications are given in terms of phase margin, gain margin, static velocity error constants, and so on. The procedure for designing a lead compensator by the frequency-response approach may be stated as follows:

Assume the following lead compensator:

$$G_c(s) = K_c \alpha \frac{T_1 + 1}{\alpha T s + 1} = K_c \frac{s + \frac{1}{T}}{s + \frac{1}{\alpha T}}, \quad (0 < \alpha < 1)$$

Define

$$k_c \alpha = k$$

Then

$$G_c(s) = K \frac{T_1 + 1}{\alpha T_1 + 1}$$

The open-loop transfer function of the compensated system is

$$G_c(s) G(s) = K \frac{T_s + 1}{\alpha T_s + 1} G(s) = \frac{T_s + 1}{\alpha T_s + 1} KG(s) = \frac{T_s + 1}{\alpha T_s + 1} G_1(s)$$

Where

$$G_1(s) = KG(s)$$

Determine gain K to satisfy the requirement on the given static error constant.

Using the gain K thus determined, draw a broad diagram of $G_1(j\omega)$, the gain-adjusted but uncompensated system. Evaluate the phase margin.
Determine the necessary phase-lead angle to be added to the system. Add an additional 5^o to 12^o to the phase lead angle required, because the addition of the lead compensator shifts the gain crossover frequency to the right and decreases the phase margin.

Determine the attenuation factor α by use of Equation (9-1). Determine the frequency where the magnitude of the uncompensated system $G_1(j\omega)$ is equal to -20 log $\left(1/\sqrt{\alpha}\right)$. Select this frequency as the new gain crossover frequency. This frequency corresponds to $\omega_m = 1/\left(\sqrt{\alpha}T\right)$, and the maximum phase shift ϕ_m occurs at this frequency.

Determine the corner frequencies of the lead compensator as follows:

Zero of lead compensator: $\omega = \frac{1}{T}$

Pole of lead compensator: $\omega = \frac{1}{\alpha T}$

Using the value of K determine in step 1 and that of α determined, in step 4, calculate constant K_c from

$$K_c = \frac{K}{\alpha}$$

Check the gain margin to be sure it is satisfactory. If not, repeat the design process by modifying the pole-zero location of the compensator until a satisfactory result is obtained.

CISE 316

Control Systems Design

Lab Experiment 8: Control System Design of DC Motor Position Control Using Lag Controller

Objective: This lab is designed to design and implement lag compensators for Position control of DC motor available in the laboratory. Aim is to use both root locus and bode plot to design the lag compensator.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW
- DC Servo System (feedback equipment)
    a. OU150A Op Amp Unit
    b. AU150B Attenuator Unit
    c. PA150C Pre-Amplifier Unit
    d. SA150D Servo Amplifier
    e. PS150E Power Supply
    f. DCM150F DC Motor
    g. IP150H Input Potentiometer
    h. OP150K Output Potentiometer
    i. GT150X Reduction Gear Tacho
    j. DC Voltmeter

Category Software Hardware Experiment

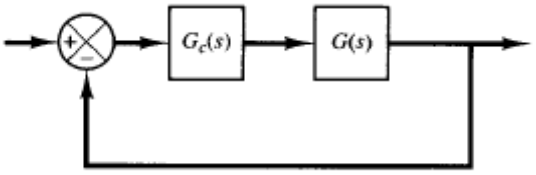
Deliverables

A complete lab report including the following:

- Design work based on root locus method
- Design work and calculations based on bode plot method.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs along with their specific responses and compare.
- Report properly with MATLAB codes and respective plots based on root locus and bode methods.

Part A - Design Procedures for Lag Compensation by the Root Locus Method

Procedure: The procedure for designing lag compensators for the system shown in following figure by root locus method is described below.



1. Draw the root locus plot for the uncompensated system whose open-loop transfer function is G(s). Based on the transient-response specifications, locate the dominant closed-loop poles on the root locus.

2. Assume the transfer function of the lag compensator to be

$$G_c = \hat{K}_c\,\beta\frac{Ts+1}{\beta Ts+1} = \hat{K}_c\,\frac{s+\dfrac{1}{T}}{s+\dfrac{1}{\beta T}}$$

3. Then the open-loop transfer function of the compensated system becomes $G_c(s)G(s)$..

4. Evaluate the particular static error constant specified in the problem.

5. Determine the amount of increase in the static error constant necessary to satisfy the specifications.

6. Determine the pole and zero of the lag compensator that produce the necessary increase in the particular static error constant without appreciably altering the original root loci.

   Note that the ratio of the value of gain required in the specifications and the gain found in the uncompensated system is the required ratio between the distance of the zero from the origin and that of the pole form the origin.

7. Draw a new root-locus plot for the compensated system. Locate the desired dominant closed-poles on the root locus.

8. If the angle contribution of the lag network is very small, that is, a few degrees, then the original and new root loci are almost identical. Otherwise, there will be a slight discrepancy between them. Then locate, on the new root locus, the desired dominant closed-loop poles based on the transient-response specifications.

9. Adjust gain $\hat{K}_c$ of the compensator from the magnitude condition so that the dominant closed-loop poles lie at the desired location. ($\hat{K}_c$ will be approximately 1).

Part B - Lag Compensation Techniques Based on the Frequency-Response Approach.

The primary function of a lag compensator is to provide attenuation in the high-frequency range to give a system sufficient phase margin. The phase-lag characteristic is of no consequence in lag compensation.

Procedure:

By the frequency-response approach may be stated as fellows:

1. Assume the following lag compensator:

$$G_c(s) = K_c\beta = \frac{T_s + 1}{\beta T_s + 1} = K_c \frac{s + \frac{1}{T}}{s + \frac{1}{\beta T}} \quad (\beta > 1)$$

Define

$$K_c\beta = K$$

2. Then

$$G_c(s) = K \frac{T_s + 1}{\beta T_s + 1}$$

3. The open-loop transfer function of the compensated system is

$$G_c(s)G(s) = K = \frac{T_s + 1}{\beta T_s + 1} G(s) = \frac{T_s + 1}{\beta T_s + 1} KG(s) = \frac{T_s + 1}{\beta T_s + 1} G_1(s)$$

Where

$$G_1(s) = KG(s)$$

4. Determine gain K to satisfy the requirement on the given static velocity error constant. If the gain-adjusted but uncompensated system $G_1(j\omega) = KG(j\omega)$ does not satisfy the specifications on the phase and gain margins, then find the frequency point where the phase angle of the open-loop transfer function is equal to -180^o plus the required phase margin. The required phase margin is the specified phase margin plus 5^o to 12^o. (The

addition of 5^o to 12^o compensates for the phase log of the lag compensator.) Choose this frequency as the new gain crossover frequency.

5. To prevent detrimental effects of phase lag due to the lag compensator, the pole and zero of the lag compensator must be located substantially lower than the new gain crossover frequency. Therefore, choose the corner frequency $\omega = 1/T$ (corresponding to the zero of the lag compensator) 1 octave to 1 decade below the new gain crossover frequency. (If the time constants of the lag compensator do not become too large, the corner frequency $\omega = 1/T$ may be chosen 1 decade below the new gain crossover frequency.)

   Notice that we choose the compensator pole and zero sufficiently small. Thus the phase lag occurs at the low-frequency region so that it will not affect the phase margin.

6. Determine the attenuation necessary to bring the magnitude curve down to 0 dB at the new gain crossover frequency. Noting that this attenuation is -20 $\log\beta$, determine the value of β. Then the other corner frequency (corresponding to the pole of the lag compensator) is determined from $\omega = 1/\beta T$

7. Using the value of K determined in step 1 and that of β determined in step 4, calculate constant K_c from

$$K_c = \frac{K}{\beta}$$

CISE 316

Control Systems Design

Lab Experiment 9: Control System Design of DC Motor Position Control Using Lead-Lag Controller

Objective: This lab is designed to design and implement lag-lead compensators for Position control of DC motor available in the laboratory. Aim is to use both root locus and bode plot to design the lag-lead compensator.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW
- DC Servo System (feedback equipment)
    a. OU150A Op Amp Unit
    b. AU150B Attenuator Unit
    c. PA150C Pre-Amplifier Unit
    d. SA150D Servo Amplifier
    e. PS150E Power Supply
    f. DCM150F DC Motor
    g. IP150H Input Potentiometer
    h. OP150K Output Potentiometer
    i. GT150X Reduction Gear Tacho
    j. DC Voltmeter

Category Software Hardware Experiment

Deliverables

A complete lab report including the following:

- Design work based on root locus method
- Design work and calculations based on bode plot method.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs along with their specific responses and compare.
- Report properly with MATLAB codes and respective plots based on root locus and bode methods.

Specifications:

Part A - Lag-Lead Compensation Techniques Based on the Root Locus Approach

Procedure: Consider following lag-lead compensator structure for the same system under discussion:

$$G_c(s) = K_c \frac{\beta}{\gamma} \frac{(T_1 s+1)(T_2 s+1)}{(\frac{T_1}{\gamma}s+1)(\beta T_2 s+1)} = K_c \left(\frac{s+\frac{1}{T_1}}{s+\frac{\gamma}{T_1}} \right) \left(\frac{s+\frac{1}{T_2}}{s+\frac{1}{\beta T_2}} \right)$$

Where $\beta > 1$, γ > 1 and K_c is the lead portion of the lag-lead compensator.

There can be different cases due to the different values of β and γ, during the designing of a lag-lead compensator.

Case 1: γ = β

In this case, the design process is a combination of the design of the lead compensator and that of the lag compensator. The design procedure for the lag-lead compensator follows:

From the given performance specifications, determine the desired location for the dominant closed-loop poles.

1. Using the uncompensated open-loop transfer function $G(s)$, determine the angle deficiency ϕ if the dominant closed-loop poles to be at the desired location. The phase-lead portion of the lag-lead compensator must contribute this angle ϕ.
2. Assuming that we later choose T_2 sufficiently large so that the magnitude of the lag portion

$$\left| \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} \right|$$

is approximately unity, where $s = s_1$ is one of the dominant closed-loop poles, choose the values of T_1 and γ from the requirement that

$$\angle(s_1 + \frac{1}{T_1}) / \angle(s_1 + \frac{\gamma}{T_1}) = \phi$$

3. The choice of T_1 and γ is not unique. Since, infinitely many sets of T_1 and γ are possible.

4. Then determine the value of K_c from the magnitude condition:

$$\left| K_c \frac{s_1 + \frac{1}{T_1}}{s_1 + \frac{\gamma}{T_1}} G(s_1) \right| = 1$$

5. If the static velocity error constant K_v is specified, determine the value of β to satisfy the requirement for K_v. The static velocity error constant K_v is given by

$$\begin{aligned} K_v &= \lim_{s \to 0} s G_c(s) G(s) \\ &= \lim_{s \to 0} s K_c \left(\frac{s + \frac{1}{T_1}}{s + \frac{\gamma}{T_1}} \right) \left(\frac{s + \frac{1}{T_2}}{s + \frac{1}{\beta T_2}} \right) G(s) \\ &= \lim_{s \to 0} s K_c \frac{\beta}{\gamma} G(s) \end{aligned}$$

where K_c and γ are already determined in step 3. Hence, given the value of K_v, the value of β can be determined from this last equation. Then, using the value of β thus determined, choose the value of T_2 such that

$$\left| \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} \right| = 1$$

$$-5^o < \left\langle \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} < 0^o \right. \qquad \text{(Note: larger < is the angle sign)}$$

Case 2: $\gamma = \beta$.

If $\gamma = \beta$ is required in the following structure of the lag-lead compensator, the following procedure should be followed:

$$G_c(s) = K_c \frac{\beta}{\gamma} \frac{(T_1 s+1)(T_2 s+1)}{(\frac{T_1}{\gamma}s+1)(\beta T_2 s+1)} = K_c \left(\frac{s+\frac{1}{T_1}}{s+\frac{\gamma}{T_1}} \right) \left(\frac{s+\frac{1}{T_2}}{s+\frac{1}{\beta T_2}} \right)$$

From the given performance specifications, determine the desired location for the dominant closed-loop poles.

1. The lag-lead compensator given by above equation is modified to

$$G_c(s) = K_c \frac{(T_1 s+1)(T_2 s+1)}{(\frac{T_1}{\beta}s+1)(\beta T_2 s+1)} = K_c \left(\frac{s+\frac{1}{T_1}}{s+\frac{\beta}{T_1}} \right) \left(\frac{s+\frac{1}{T_2}}{s+\frac{1}{\beta T_2}} \right)$$

Where $\beta > 1$. The open-loop transfer function of the compensated system is $G_c(s)G(s)$. If the static velocity error constant K_v is specified, determine the value of constant K_c from the following equation:

$$K_v = \lim_{s \to 0} sG_c(s)G(s)$$

$$K_v = \lim_{s \to 0} sK_c G(s)$$

2. To have the dominant closed-loop poles at the desired location, calculate the angle contribution ϕ needed from the phase lead portion of the lag-lead compensator.

3. For the lag-lead compensator, we later choose T_2 sufficiency large so that

$$\left| \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} \right|$$

is approximately unity, where $s = s_1$ is one of the dominant closed loop poles. Determine the values of T_1 and β from the magnitude and angle conditions:

$$\left| K_c \left(\frac{s_1 + \frac{1}{T_1}}{s_1 + \frac{\beta}{T_1}} \right) G(s_1) \right| = 1$$

$$\left\langle \frac{s_1 + \frac{1}{T_1}}{s_1 + \frac{\beta}{T_1}} = \phi \right.$$

4. Using the value of β just determined, choose T_2 so that

$$\left| \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} \right| = 1$$

$$-5^o < \left\langle \frac{s_1 + \frac{1}{T_2}}{s_1 + \frac{1}{\beta T_2}} < 0^o \right.$$

5. The value of βT_2, the largest time constant of the lag-lead compensator, should not be too large to be physically realized.

Part B - Lag-Lead Compensation Based on the Frequency-Response Approach.

The design of a lag-lead compensator by the frequency-response approach is based on the combination of the design techniques discuss under lead compensation and lag compensation.

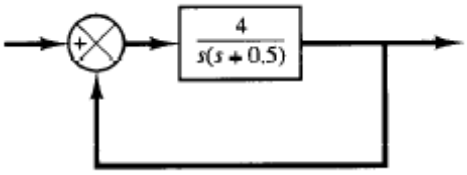
Let us assume that the lag-lead compensator is of the following form:

$$G_c(s) = K_c \frac{(T_1 s + 1)(T_2 s + 1)}{\left(\frac{T_1}{\beta} s + 1 \right)(\beta T_2 s + 1)} = K_c \frac{\left(s + \frac{1}{T_1} \right)\left(s + \frac{1}{T_2} \right)}{\left(s + \frac{\beta}{T_1} \right)\left(s + \frac{1}{\beta T_2} \right)}$$

Where $\beta > 1$. The phase lead portion of the lag-lead compensator (the portion involving T_1) alters the frequency-response curve by adding phase-lead angle and increasing the phase margin at the gain crossover frequency. The phase-lag portion (the portion involving T_2) provides

attenuation near and above the gain crossover frequency and there by allows an increase of gain at the low-frequency range to improve the steady-state performance.

Soft Exercise: Consider the control system shown in following figure.



The feedforward transfer function is $G(s) = \dfrac{4}{s(s+0.5)}$

Suppose that the lag-lead compensator of following structure is to be used:

$$G_c(s) = K_c \frac{\left(s+\frac{1}{T_1}\right)\left(s+\frac{1}{T_2}\right)}{\left(s+\frac{\beta}{T_1}\right)\left(s+\frac{1}{\beta T_2}\right)} \quad (\beta > 1)$$

Design a lag-lead compensator $G_c(s)$ to achieve following specifications:

- Damping ratio of dominant closed loop poles = 0.5
- Undamped natural frequency = 5 rad/sec
- Static velocity error constant = 80 /sec

Exercise:

(a) Follow the procedure for root locus design to design and implement lag-lead compensator for the position control of DC motor system available in the laboratory.

(b) Follow the procedure for bode plot method to design and implement lag-lead compensator for the position control of DC motor system available in the laboratory.

CISE 316

Control Systems Design

Lab Experiment 10: Pole Placement by State Feedback

Overview

We will design better controllers if we have more measurements of the system behavior. In the case where we can measure all of the system states, we can place the closed-loop poles anywhere we choose within the constraint of actuator saturation. In this lab, you will use the model of a pendulum system to design a state feedback controller to keep the pendulum upright (the inverted pendulum).

Objectives: At the conclusion of this laboratory experience, students should be able to:

- Use Ackermann's formula to design a state feedback controller given desired pole locations.
- **Extra Credit/Term Project:** Implement a state feedback controller on the Feedback hardware of Inverted Pendulum using the MATLAB Simulink interface / LabVIEW interface.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW
- Inverted Pendulum (feedback equipment)

Category Software Hardware Experiment

Deliverables

A completed lab report including the following:

- Individual plots of the outputs of each of your successful controller designs

Background

Applying the first principles, the state space description of the linear cart inverted pendulum system is nearly identical to the crane mode model. In fact, the inverted model can be found from the crane model by simply changing the sign on a few terms. Here we show only the state space

model, since you will be using state space for design. The schematic of the translational cart-pendulum system is included in Figure 1.

$$\dot{x} = Ax + B\,f \tag{1}$$

$$y = Cx \tag{2}$$

The state space system matrices for the translational inverted cart-pendulum system are given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-J_A b_1}{D} & \dfrac{m_p^2 l_{eg}^2 g}{D} & \dfrac{-m_p l_{eg} b_2}{D} \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-m_p l_{eg} b_1}{D} & \dfrac{(m_p + m_c) m_p l_{eg} g}{D} & \dfrac{-(m_p + m_c) b_2}{D} \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 & \dfrac{J_A}{D} & 0 & \dfrac{m_p l_{eg}}{D} \end{bmatrix}^{\tau} \tag{4}$$

$$C = \begin{bmatrix} K_1 & 0 & 0 & 0 \\ 0 & 0 & K_2 & 0 \end{bmatrix} \tag{5}$$

$$\dot{x} = \begin{bmatrix} x_c & \dot{x} & \theta & \dot{\theta} \end{bmatrix} \tag{6}$$

$$D = J_A (m_p + m_c) - m_p^2 l_{eg}^2 \tag{7}$$

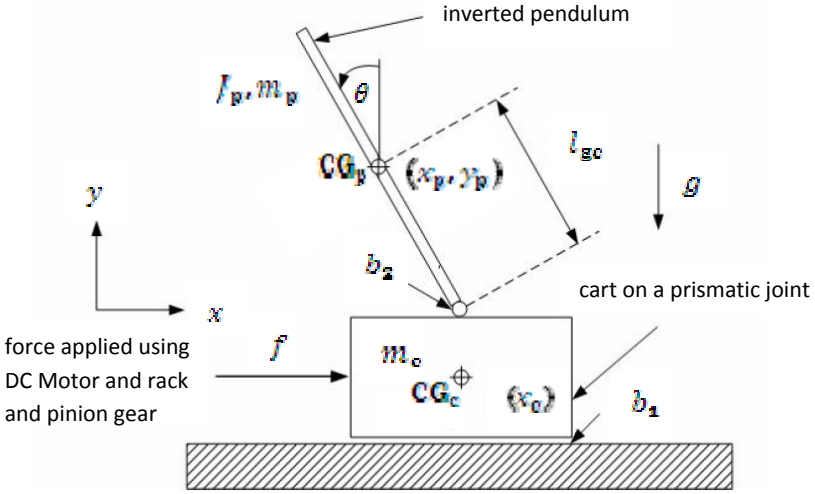


Figure 1 – Inverted Cart-Pendulum System Schematic

Assume we have the following known parameters of the system.

$m_p = 0.1565\ Kg$

$l_{eg} = 0.216\ m$

$g = 9.8^{m/\sec^2}$

$r = 0.125 m$

We will implement state feedback control in the form shown in Figure 3. The pre-filter gain will be zero for this experiment.

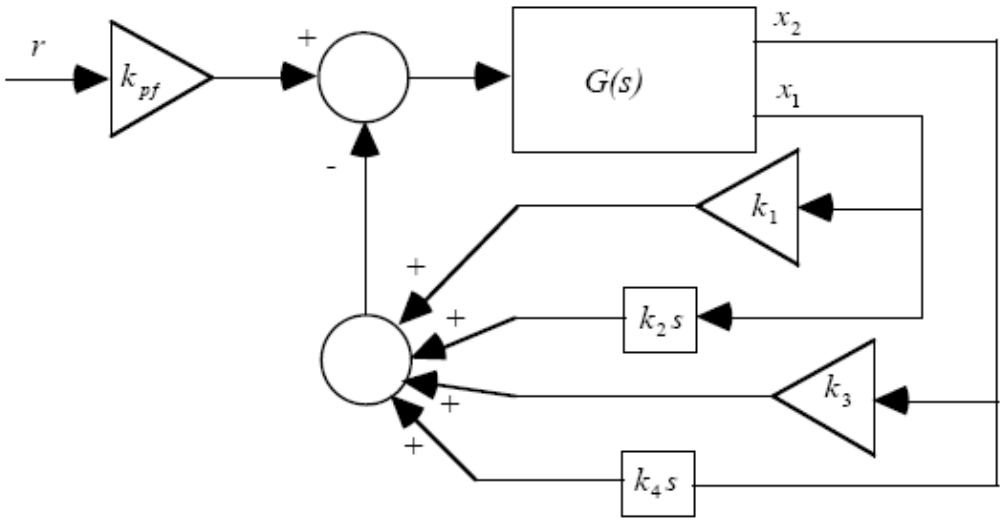


Figure 3 – Closed-Loop Block Diagram.

For each of the sets of desired poles given in Table 1, construct the state feedback controller using Ackermann's formula,

$$K = [0 \quad 0 \quad \cdots\cdots \quad 0 \quad 1] \bullet [B \quad AB \quad \cdots\cdots \quad A^{n-1}B] \cdot \varphi(A)$$

$$= [k_1 \quad k_2 \quad k_3 \quad k_4] \qquad (13)$$

where $\varphi(A)$ is the desired characteristic polynomial evaluated at s = A or

$$\varphi(A) = (s - p_1)(s - p_2)(s - p_3)(s - p_4)$$

$$= s^4 + a_1 s^2 + a_2 s^2 + a_2 s^1 + a_4 \qquad (14)$$

Table 1 – Desired Poles (Eigenvalues) for State Feedback Controller Design.

Translation System:			
Set 1	**Set 2**	**Set 3**	**Set 4**
$p_{1,2} = -2.3544 \pm j2.2492$ $p_3 = -54.2314$ $p_4 = -2.4591$	$p_{1,2} = -2.3679 \pm j2.2994$ $p_3 = -47.6143$ $p_4 = -2.4286$	$p_{1,2} = -2.3741 \pm j2.3284$ $p_3 = -66.7402$ $p_4 = -2.4139$	$p_{1,2} = -2.3779 \pm j2.3491$ $p_3 = -95.9380$ $p_4 = -2.4067$

In-Lab

1. Configure the inverted pendulum system.

a) You should have the pendulum mounted on the cart with no spring, no damper attached. If you need help configuring your station ask your instructor.

b) Position the cart and pendulum at zero (vertical hanging position).

c) Type your gains into the block 'Gain7' on translational systems. Keep the negative sign in front of the gain array, i.e. you computed K , this block should be $-K$. A screen capture of the input window is given in Figure 4.

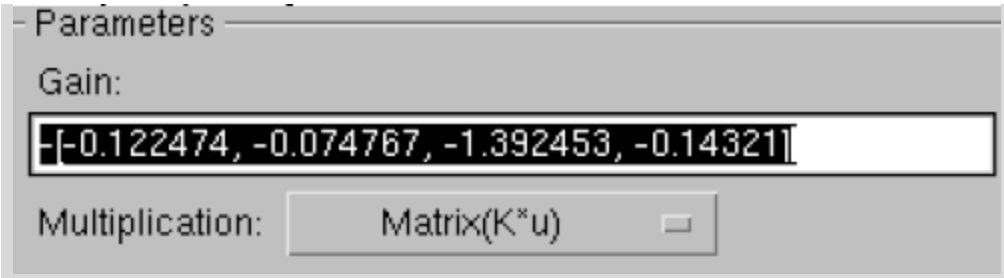


Figure 4 – Matrix Gain Adjustment with Typical Values.

d) In the model 'inverted_pendulum_cl.mdl' connect to target. Screen captures of these two block diagrams are included below in Figures 5. Now lift the pendulum to the inverted pendulum. Have your lab partner start the simulation (play button), and release the pendulum.

e) See if the pendulum balances. Notice how many seconds it is able to balance the pendulum. Note, all the gain sets you computed will probably work but hopefully at least two or three will work.

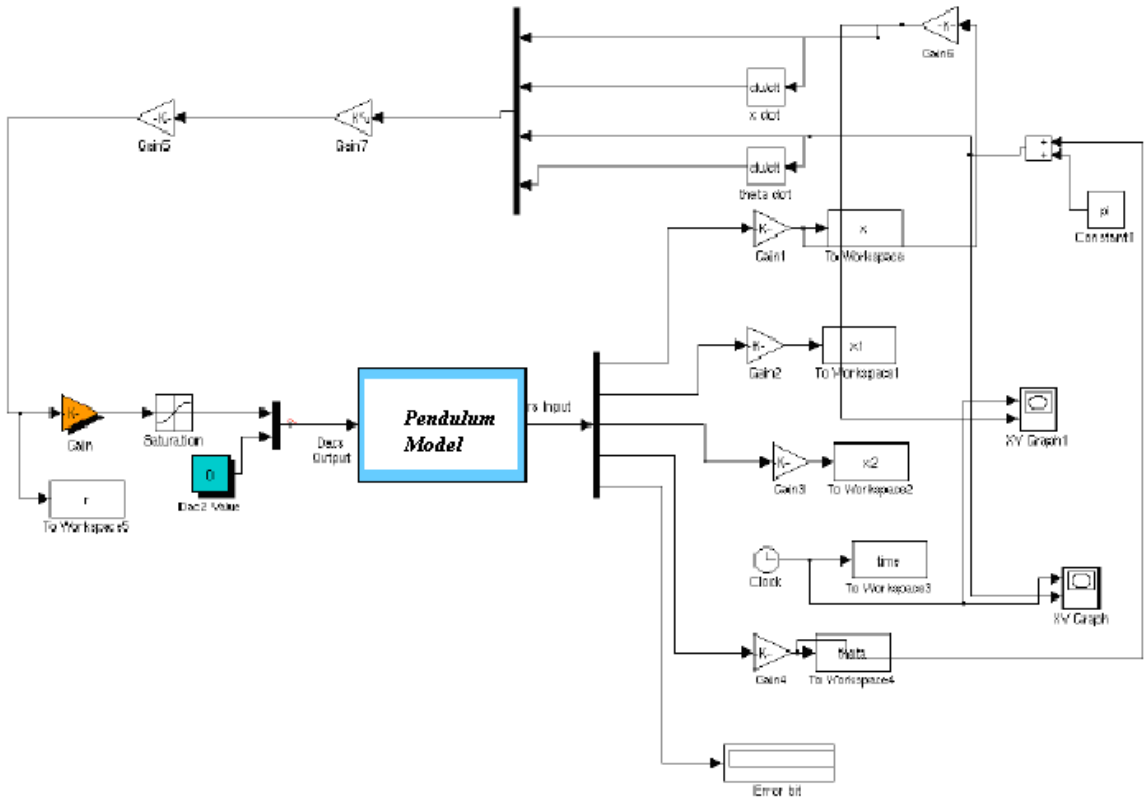


Figure 5 – Translational Inverted Pendulum Closed-Loop Model (inverted_pendulum_cl.mdl).

CISE 316

Control Systems Design

Project I: Two Tank System Control

Objective: We would like to control the two-tank system using a PID type of controller. The selection of the PID control gains should allow the system dynamic to have the following specifications:

1- Overshoot less than 5%

2- Drop due to disturbance less than 10%

3- Steady state error zero.

Project Duration: 2 weeks

Deliverables

A basic project design report including the following:

- Project Overview Diagram
- Project Plan
- List of Equipment and Items
    a. Hardware
    b. Software packages/tools
- Two tank system model simulation in MATLAB.
- Design of the controller in MATLAB.
- Model validation using step response.
- Implementation of the controller obtained in MATLAB on the real system.
- Comparison between real performance and simulated performance.
- Tuning of the controller gains. Students should record the procedure they used to tune the gains.
- The engineering design steps should be followed.

Grading: as per the rubric available in WebCt.

CISE 316

Control Systems Design

Project II: Magnetic levitation

Objective: We would like to control the Magnetic levitation system using a PID or a Lead-Lag type of controller. The selection of the controller gains should allow the system dynamic to have the following specifications:

1. Stable position:
    a. Phase Margin at least 35^o
    b. Gain Margin at least 10 db.
2. Robust control to disturbances
3. Control of the ball position within Magnetic field range with an error < 1%.

Project Duration: 2 weeks

Deliverables: A basic project design report including the following:

- Project Overview Diagram
- Project Plan
- List of Equipment and Items
    a. Hardware
    b. Software
- Magnetic Levitation model simulation in MATLAB.
- Identify a model for the magnetic levitation using experimental data.
- Model validation using step response.
- Design of the controller in MATLAB.
- Implementation of the controller obtained in MATLAB on the real system.
- Comparison between real performance and simulated performance.
- Tuning of the controller gains. Students should record the procedure they used to tune the gains.
- The engineering design steps should be followed.

Grading: as per the rubric available in WebCt.