Bangladesh University of Engineering and Technology

Electrical and Electronic Engineering Department

EEE 318 : Control System I Laboratory

Experiment No. 4

a) Effect of input waveform, loop gain, and system type upon steady-state errors
b) Effect of open-loop poles and zeros upon the shape of the root locus

Part-A: Effect of input waveform, loop gain, and system type upon steady-state errors

Objective: To verify the effect of input waveform, loop gain, and system type upon steady-state errors

Minimum required software packages: MATLAB, Simulink, and the Control System Toolbox.

Theory:

Steady-state error is defined as the difference between the input (command) and the output of a system in the limit as time goes to infinity (i.e. when the response has reached steady state). The steady-state error will depend on the type of input (step, ramp, etc.) as well as the system type (0, I, or II).

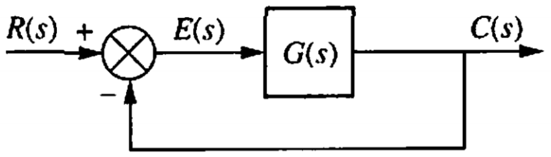


Figure 1: Representation of unity feedback system

Consider the feedback control system shown in Figure 1. Since the feedback, *H(s),* equals 1, the system has unity feedback. The implication is that *E(s)* is actually the error between the input, *R(s),* and the output, *C(s).* Thus, if we solve for *E(s),* we will have an expression for the error. We will then apply the final value theorem to evaluate the steady-state error.

From Figure 1,

$$E(s) = R(s) - C(s)$$

Now, replacing the value of $C(s)$ in the above equation by using $C(s) = G(s)E(s)$ and upon rearranging, we obtain

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

Now, if the closed system is stable, we can calculate the steady state error $e(\infty)$ as following:

$$e(\infty) = \lim_{s\to 0} \frac{sR(s)}{1+G(s)} \qquad (1)$$

Equation (1) allows us to calculate the steady-state error, $e(\infty)$, given the input, *R(s),* and the system, *G(s).* We now substitute several inputs for *R(s)* and then draw conclusions about the relationships that exist between the open-loop system, *G(s),* and the nature of the steady-state error , $e(\infty)$.

For a step input with u(t) or $R(s) = \frac{1}{s}$, steady-state error

$$e(\infty) = e_{step}(\infty) = \lim_{s\to 0} \frac{sR(s)}{1 + G(s)} = \lim_{s\to 0} \frac{s({}^{1}/_{s})}{1 + G(s)} = \frac{1}{1 + \lim\limits_{s\to 0} G(s)}$$

The term $\lim_{s\to 0} G(s)$ is the dc gain of the forward transfer function, since s the frequency variable is approaching zero. In order to have zero steady state error,

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

Hence, for a step input to a unity feedback system, the steady-state error will be zero if there is at least one pure integration in the forward path. If there are no integrations, then there will be a nonzero finite error.

Similarly, for a ramp input $tu(t)\ or\ R(s) = {}^{1}/_{s^2}$

$$e(\infty) = e_{ramp}(\infty) = \frac{1}{\lim\limits_{s\to 0} sG(s)}$$

For a parabolic input ${}^{1}/_{2}\, t^2 u(t)$ or $R(s) = {}^{1}/_{s^3}$

$$e(\infty) = e_{parabola}(\infty) = \frac{1}{\lim\limits_{s\to 0} s^2 G(s)}$$

The three terms in the denominator that are taken to the limit determine the steady-state error. We call these limits *static error constants.* Individually, their names are

Position constant $K_p\ where\ K_p = \lim_{s\to 0} G(s)$

Velocity constant $K_v\ where\ K_v = \lim_{s\to 0} sG(s)$

Acceleration constant $K_a\ where\ K_a = \lim_{s\to 0} s^2 G(s)$

These quantities, depending upon the form of *G(s),* can assume values of zero, finite constant, or infinity. Since the static error constant appears in the denominator of the steady-state error, the value of the steady-state error decreases as the static error constant increases.

System type:

Let us continue to focus on a unity negative feedback system. The values of the static error constants, again, depend upon the form of *G(s),* especially the number of pure integrations in the forward path. Since steady-state errors are dependent upon the number of integrations in the forward path, we give a name to this system attribute.

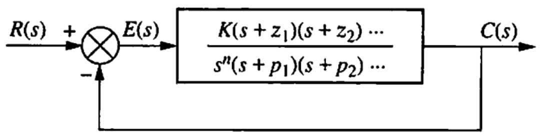


Figure 2: Feedback control system for defining system type

Given the system in Figure 2, we define system type to be the value of n in the denominator or, equivalently, the number of pure integrations in the forward path. Therefore, a system with n = 0 is a Type 0 system. If n = 1 or n = 2, the corresponding system is a Type 1 or Type 2 system, respectively.

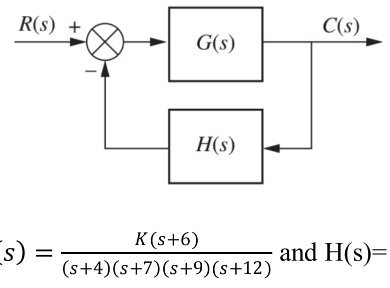
Table 1 ties together the concepts of steady-state error, static error constants, and system type. The table shows the static error constants and the steady-state errors as functions of input waveform and system type.

Table 1: Relationships between input, system type, static error constants, and steady-state errors

		Type 0		Type 1		Type 2	
Input	Steady-state error formula	Static error constant	Error	Static error constant	Error	Static error constant	Error
Step, $u(t)$	$\frac{1}{1+K_p}$	$K_p =$ Constant	$\frac{1}{1+K_p}$	$K_p = \infty$	0	$K_p = \infty$	0
Ramp, $tu(t)$	$\frac{1}{K_v}$	$K_v = 0$	∞	$K_v =$ Constant	$\frac{1}{K_v}$	$K_v = \infty$	0
Parabola, $\frac{1}{2}t^2u(t)$	$\frac{1}{K_a}$	$K_a = 0$	∞	$K_a = 0$	∞	$K_a =$ Constant	$\frac{1}{K_a}$

Prelab:

1. Read the theory discussed in the previous section.
2. What system types will yield zero steady-state error for step inputs?
3. What system types will yield zero steady-state error for ramp inputs?
4. What system types will yield infinite steady-state error for ramp inputs?
5. What system types will yield zero steady-state error for parabolic inputs?
6. What system types will yield infinite steady-state error for parabolic inputs?
7. For the negative feedback system of the following figure, where



$$G(s) = \frac{K(s+6)}{(s+4)(s+7)(s+9)(s+12)}$$ and H(s)=1,

Calculate the steady-state error in terms of K, for the following inputs:
$5u(t), 5tu(t)$ *and* $5t^2u(t)$

8. Repeat Prelab 7 for $G(s) = \frac{K(s+6)(s+8)}{(s+4)(s+7)(s+9)(s+12)}$ and $H(s) = 1$
9. Repeat Prelab 7 for $G(s) = \frac{K\,(s+1)(s+6)(s+8)}{s^2(s+4)(s+7)(s+9)(s+12)}$ and $H(s) = 1$.

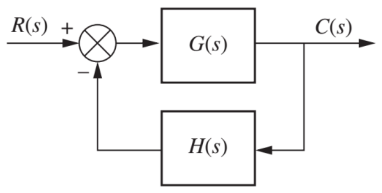
Lab Work:

1. Using Simulink and from your experience of performing experiment no 3 of EEE 402, set up the negative feedback system of Prelab 7. Plot on one graph the error signal of the system for an input of $5u(t)$ and K ¼ 50; 500; 1000, and 5000. Repeat for inputs of $5tu(t)$ *and* $5t^2u(t)$.
2. Using Simulink, set up the negative feedback system of Prelab 8. Plot on one graph the error signal of the system for an input of $5u(t)$ and K ¼ 50; 500; 1000, and 5000. Repeat for inputs of $5tu(t)$ *and* $5t^2u(t)$
3. Using Simulink, set up the negative feedback system of Prelab 9. Plot on one graph the error signal of the system for an input of $5u(t)$ and K ¼ 200; 400; 800, and 1000. Repeat for inputs of $5tu(t)$ *and* $5t^2u(t)$

Report:

1. Use your plots from Labwork 1 and compare the expected steady-state errors to those calculated in the Prelab. Explain the reasons for any discrepancies.
2. Use your plots from Labwork 2 and compare the expected steady-state errors to those calculated in the Prelab. Explain the reasons for any discrepancies.
3. Use your plots from Labwork 3 and compare the expected steady-state errors to those calculated in the Prelab. Explain the reasons for any discrepancies.

Additional:



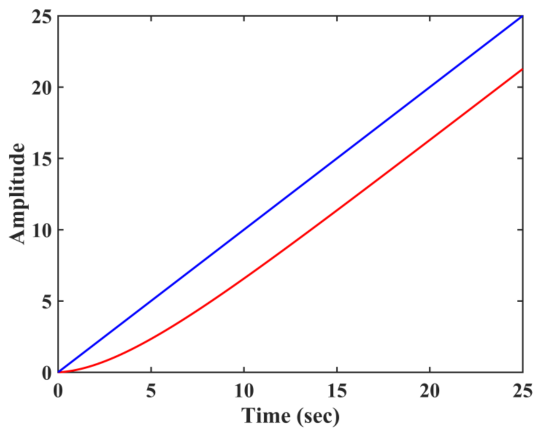
For this example, let us

$G(s) = \frac{K(s+3)(s+5)}{s(s+7)(s+8)}$ and $H(s) = 1$;

Since this system is type 1, there will be no steady-state error for a step input and there will be infinite error for a parabolic input. The only input that will yield a finite steady-state error in this system is a ramp input. We can examine the ramp input response for a gain K=1 as following:

```
clc;
clear all;
close all;
s = tf('s');
G = ((s+3)*(s+5))/(s*(s+7)*(s+8));
ts = feedback(G,1);
t = 0:0.1:25;
u = t;    % ramp
[y,t,x] = lsim(ts,u,t);
plot(t,y,'r',t,u,'b')
xlabel('Time (sec)')
ylabel('Amplitude')
```

Output:



The steady-state error for this system is quite large, since we can see that at time 20 seconds the output is 16.29 as compared to an input of 20 (steady-state error is approximately equal to 4).

Now,

- **Design the system such that the closed-loop system has a steady-state error of 0.1 in response to a ramp reference**.

***Feedback from Experiment 3**:* Explore the Matlab documentation for the functionality of stepinfo().

❖ Can you design your own functions for example namely 'rampinfo()' and 'parabinfo()' that would perform the similar task for the ramp and parabolic inputs just like stepinfo() does it for the step input?

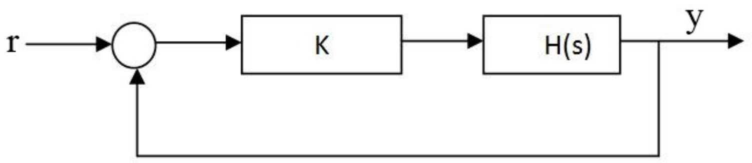
Part B: Effect of open-loop poles and zeros upon the shape of the root locus

Objective: To verify the effect of open-loop poles and zeros upon the shape of the root locus.

Minimum required software packages: MATLAB

Theory:

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 transfer function of the system is –

$$\frac{Y(s)}{R(s)} = \frac{KH(s)}{1 + KH(s)}$$

and thus the poles of the closed loop system are values of s such that 1 + KH(s) = 0

We can write H(s) as $\frac{\mathrm{b(s)}}{\mathrm{a(s)}}$; then the above equation has the form:

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

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→0, the poles of the closed loop system are a(s) = 0 or the poles of H(s). In the limit as k→∞, the poles of the closed-loop system are b(s) = 0 or the zeros of H(s).

No matter what we pick k to be, **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→∞, 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).

For further detail, you can go through the sections 8.1-8.4 of "CONTROL SYSTEMS ENGINEERING" by Norman S. Nise.

Drawing the root locus

The main idea of root locus design is to find the closed-loop response from the root locus plot. Then by adding zeros and/or poles to the original plant, the closed-loop response will be modified.

For this purpose, let us consider a DC motor system with the following transfer function:

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

Let us consider the following values for the physical parameters of a DC motor:

1. Moment of inertia of the rotor (J) = 3.2284E-6 kg.m^2/s^2
2. Damping ratio of the mechanical system (b) = 3.5077E-6 Nms
3. Electromotive force constant ($K = K_e = K_t$) = 0.0274 Nm/Amp
4. Electric resistance (R) = 4 Ω
5. Electric inductance (L) = 2.75E-6 H
6. Input (V) = Source Voltage
7. Output (θ) = position of shaft

The rotor and shaft are assumed rigid.

Now, consider the transfer function of a DC Motor relating input voltage to rotor position as given above.

Put the transfer function into Matlab by defining the numerator and denominator as vectors: Create a new m-file and enter the following commands:

```
J=3.2284E-6;
b=3.5077E-6;
K=0.0274;
R=4;
L=2.75E-6;
num=K;
den=[(J*L)  ((J*R)+(L*b))  ((b*R)+K^2)  0];
step(num,den,0:0.001:0.2)
```

Add the following commands at the end of your m-file and run the code.

```
sys= tf (num,den);
rlocus(sys);
sgrid(.5,0);
```

[sgrid is a function in the Matlab tool box. The variables in the sgrid command are the zeta term (0.5 corresponds to a overshoot of 16%), and the wn term (no rise time criteria) respectively.]

If you look at the axis scales on your plot, one open-loop pole is very far to the left (further than $-1x10^6$). This pole does not affect the closed loop dynamics unless very large gains are used, where the system becomes unstable. We will ignore it by changing the axes to zoom in to just the lower-gain portion of the root locus. We also need to remove steady-state error due to a disturbance.

```
axis([-400 100 -200 200])
```

Integral Control

Now, let us try using integral control to remove steady-state error due to a disturbance. This adds a 1/s term to the forward loop or in other words a pole at the origin. Insert the following lines to define the controller (numcf and dencf) in your m-file.

```
numcf=[1];
dencf=[1 0];
controller=tf(numcf,dencf);
I_sys=controller*sys;
rlocus(I_sys);
```

Run the m-file and obtain the root locus of the system. Comment on the stability of the system from the root locus plot.

Proportional plus Integral Control

Now, let's modify the integral controller to a PI controller. Using PI instead of I control adds a zero to the open-loop system. We'll place this zero at s=-20. Change the lines defining the controller (numcf and dencf) in your m-file to the following.

```
numcf=[1 20];
dencf=[1 0];
```

Re-run your m-file. Obtain the step response of the system for disturbance input. Does PI controller improve system performance for disturbance input?

Proportional plus Integral plus Derivative Control

In order to pull the root locus further to the left, to make it faster, we need to place a second open-loop zero, resulting in a PID controller. You can place the two PID zeros at s= -60 and s= -70. Change the lines defining the controller (numcf and dencf) in your m file and obtain the root locus plot for gain **0 : 0.001 : 1.**

Now, you can see that two of the closed-loop poles loop around well within both the settling time and percent overshoot requirements. The third closed loop pole moves from the open-loop pole at s=-59.2 to the open loop zero at s=-60. This closed-loop pole nearly cancels with the zero (which remains in the closed loop transfer function) because it is so close. Therefore, we can ignore its effect. However, the other open loop zero also remains in the closed-loop, and will affect the response, slowing it down, and adding overshoot. Therefore, we have to be conservative in picking where on the root locus we want the closed-loop poles to exist.

Finding the gain using the *rlocfind* command

We need the settling time and the overshoot to be as small as possible, particularly because of the effect of the extra zero. Large damping corresponds to points on the root locus near the real axis. A fast response corresponds to points on the root locus far to the left of the imaginary axis. To find the gain corresponding to a point on the root locus, we can use the **rlocfind** command. We can find the gain and plot the step response using this gain all at once. To do this, enter the following commands at the end of your m-file and re-run it.

```
[k poles]=rlocfind(PID_sys);
feedbk_sys=feedback(k*PID_sys,1);
figure
step(feedbk_sys);
```

Go to the plot and select a point on the root locus on left side of the loop, close to the real axis as shown below with the small black + marks. This will ensure that the response will be as fast as possible with as little overshoot as possible. These pole locations would indicate that the response would have almost no overshoot, but you must remember that the zero will add some overshoot.

Help:

```
%Complete MATLAB code for PID compensated system:

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

```
num=K;
den=[(J*L)  ((J*R)+(L*b))   ((b*R)+K^2)   0];
sys= tf (num,den); % transfer function of the motor.
numcf=conv([1 60],[1 70]);
dencf=[1 0];
controller=tf(numcf,dencf);      % transfer function of the PID controller.
PID_sys=controller*sys;
rlocus(PID_sys,0: .001:1);       %plot the root locus of the system
sgrid(.5,0);
axis([-400 100 -200 200])
[k poles]=rlocfind(PID_sys);     % locate a suitable point on root locus  and thus obtain the required gain %for the system

feedbk_sys=feedback(k*PID_sys,1);
figure
step(feedbk_sys,0: .001: .1);      % step response of the compensated system.
Figure
dis_sys=feedbk_sys/(k*controller);  % transfer function for disturbance input
step(dis_sys,0: .001: .1);       % step response for disturbance input
```

Prelab

1. Sketch two possibilities for the root locus of a unity negative-feedback system with the open-loop pole/zero configuration shown in the following Figure:

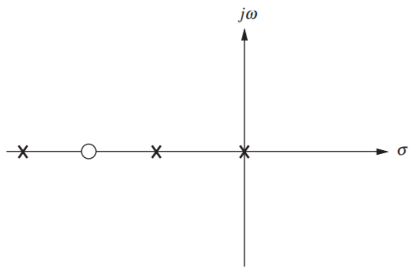


Figure for Prelab 1

2. If the open-loop system of Prelab 1 is

$$G(s) = \frac{K(s+1.5)}{s(s+0.5)(s+10)}$$

Then, estimate the percent overshoot at the following values of gain, K: 20, 50, 85, 200, and 700.

Additional Lab Task:

1. Using MATLAB set up a negative unity-feedback system with

$$G(s) = \frac{K(s+6)}{s(s+0.5)(s+10)}$$

to produce a root locus. Find the root locus for the zero at 6. Move the zero to the following locations and find out a root locus at each location: -2, -1.5, -1.37, and -1.2.

2. Using MATLAB set up a negative unity-feedback system with

$$G(s) = \frac{K(s+1.5)}{s(s+0.5)(s+10)}$$

to produce a root locus. Find the step response of the system. Using the values of K specified in Prelab 2, record the percent overshoot and settling time and find the root loci and step response for each value of K.

Report Task:

In the task we want to position a dc motor very precisely, thus the steady-state error of the motor position should be zero. We will also want the steady-state error due to a disturbance, to be zero as well. The other performance requirement is that the motor reaches its final position very quickly. In this case, we want it to have a settling time of 20ms. We also want to have an overshoot smaller than 5%.

Therefore, for a unit step input, the motor speed output should have:

- Settling time less than 20 milliseconds
- Overshoot less than 5%
- No steady-state error
- No steady-state error due to a disturbance

Prepared By:

Asir Intisar Khan
Lecturer, Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology

Supervised By:

Dr. Pran Kanai Saha
Professor, Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology

Dr. Mohammad Ariful Haque
Professor, Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology

Dr. Md. Zahurul Islam
Associate professor, Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology