Root Locus Controller Design Using the Matlab 'sisotool' Toolbox

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 engineer's 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.

Objectives

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

- To successfully design P, I, PD, PI, and PID controllers to meet closed-loop performance specifications including transient performance and steady-error.

Deliverables

A completed worksheet including the following:

- Figures with plots of closed-loop step responses.
- Controller parameters, gain, pole(s), and zero(s), for each of the controller designs.
- Answer all the questions on the worksheet.

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.

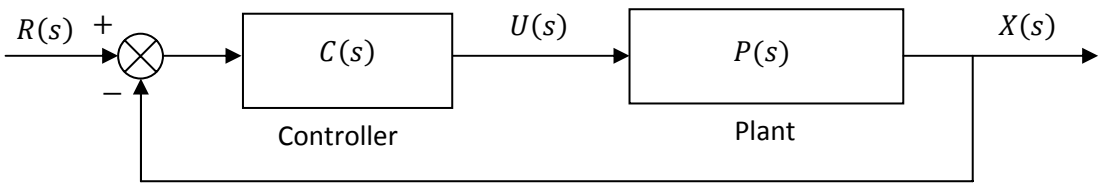

Figure 1 – Generic Unity Feedback Control System.

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 $\lvert z\rvert > \lvert p\rvert$
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 $\lvert p\rvert > \lvert z\rvert$
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) (Complex Conjugate Zeros)	$C(s) = k_p + \frac{k_i}{s} + k_d s = \frac{k \cdot (s+z)(s+z^*)}{s}$

Note that the I, PI, and PID controller's 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 → Open Loop Bode** to turn off the bode plot. (Whatever is checked in this list will be displayed in the window.)

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

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

2. You will probably two curves on your step response plot. To fix this click on **Analysis → Other Loop Responses…** Make sure only r to y is checked, and then click OK.

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. Click **Compensators → Edit → C**. Click on **Add Real Zero** or **Add Real Pole** to enter controller zeros or poles. You will be able to make changes to these values later. After you are done, click **OK** to exit this window.

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. Click Edit → **Root Locus → Design Constraints** then either **New** to add new constraints or **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, click **Edit → Root Locus → Properties**, click on **Limits**, and set the desired axis limits.

2. You may also want to turn the grid on. Click **Edit → Root Locus → Grid**.

3. It is convenient to use the zero/pole/gain format for the compensators. To do this, click on **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. 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.

2. 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.

3. 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?

4. 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.

5. 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?

6. 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 1

second, 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 1 second. (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

- $e_{ss} \leq 0.1$ for unit step inputs
- $t_{s,2\%} \leq 0.5$ seconds
- $\%\text{OS} \leq 10\%$

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

- **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, your controller parameters, and the steady state error.

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

- $e_{ss} \leq 0.2$ for unit step inputs
- $t_{s,2\%} \leq 1$ seconds
- $\%\text{OS} \leq 20\%$

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

- **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, your controller parameters, and the steady state error.