EENG307: PD Design Using Root Locus and SISOtool*

Elenya Grant, Kathryn Johnson, and Hisham Sager†

Lecture 19

Contents

1 PD Design

To design the controller, we need to see how the control will affect the closed loop system response. In this lecture, we'll consider both the physically-based and design configurations that are discussed in Lecture 13: Proportional-Derivative (PD) Control to Improve Transient Response. In that lecture, we were able to pick K_p and K_d so that the denominator matched a desired second order polynomial $s^2+2\zeta\omega_n s+\omega_n^2$. However, this only works if $G(s)$ is second order.

What do we do if $G(s)$ is not second order, or if we want a more flexible way to look at even second-order plant systems? In this case, we can use a method that is based on the root locus, which is the subject of this lecture.

Recall that the design and physically-based configurations from Lecture 13 have the same closed-loop denominators but different numerators. In each case, the PD controller incorporates a

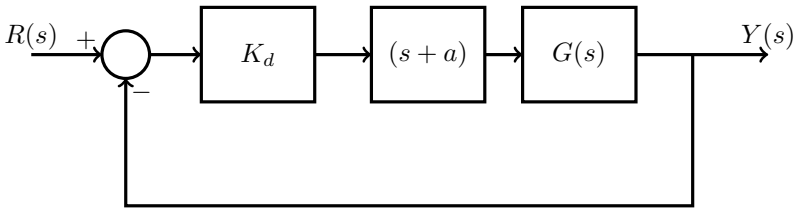
$$K_d\left(s+\frac{K_p}{K_d}\right)$$

term into the denominator from the PD controller. Thus for PD control design we can consider the following steps to achieve a suitable controller:

1. Define $a \equiv K_p/K_d$ (for convenience).
2. Given a set of step response specifications, find the allowable region for the dominant closed loop poles.
3. Choose the PD controller's zero location $s=-a$ so that the root locus goes through this desirable closed-loop pole region.
4. Choose K_d to achieve these specific poles.
5. Use $K_p = aK_d$ to calculate the proportional gain.
6. Verify and (if needed) tune the design.

†Developed and edited by Tyrone Vincent and Kathryn Johnson, Colorado School of Mines, with contributions from Salman Mohagheghi, Chris Coulston, Kevin Moore, CSM and Matt Kupilik, University of Alaska, Anchorage

Root Locus Design Problem



Since the closed loop "design configuration" system (shown in this block diagram) includes an undesired zero at $-\frac{K_p}{K_d}$, once we solve for K_p and K_d we need to either (1) check and tune the results further to accommodate the extra closed-loop zero in the design configuration, or (2) implement these gains using the physically-based system configuration to avoid the zero.

2 PD Design Process

We will work through the PD design steps using an example. Assume you have an open-loop plant system

$$G(s) = \frac{1}{(s+10)(s^2+3s+2)}$$

and are asked to design a feedback controller so that the closed-loop system meets the following step response specifications:

- % OS $\leq$ 10 %
- 1% Settling time $\leq$ 2.3s

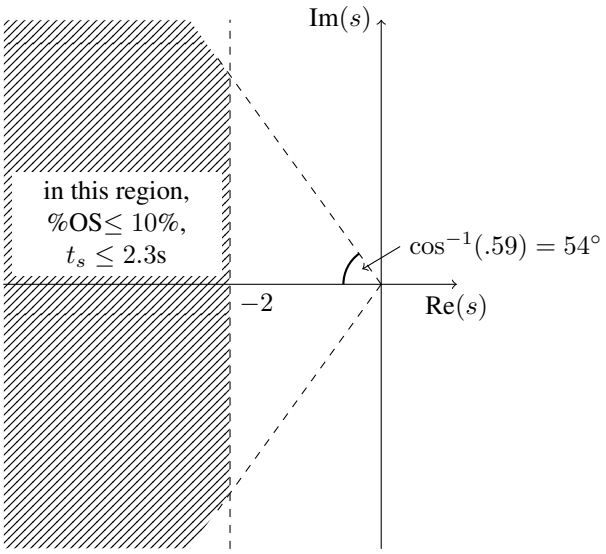
Step 1: Define $a \equiv K_p/K_d$.

Step 2: From the provided specifications, we find requirements on ζ and ω_n:

$$\begin{aligned}\zeta &\geq \frac{-\ln(0.1)}{\sqrt{\pi^2+\ln(0.1)^2}} \\ &\geq 0.59\end{aligned}$$

$$\begin{aligned}2.3 &\geq \frac{4.6}{\zeta\omega_n} \\ \zeta\omega_n &\geq 2\end{aligned}$$

This implies the following region of acceptable closed loop poles:



Steps 3-6 will be illustrated at points in Sections 2.1-2.2.

2.1 Using control system designer (sisotool)

The Mathworks has a tool called the "Control System Designer" that was formerly known as "sisotool" (which explains the origin of the command that calls the tool). This tool is very helpful for control design, including via root locus.

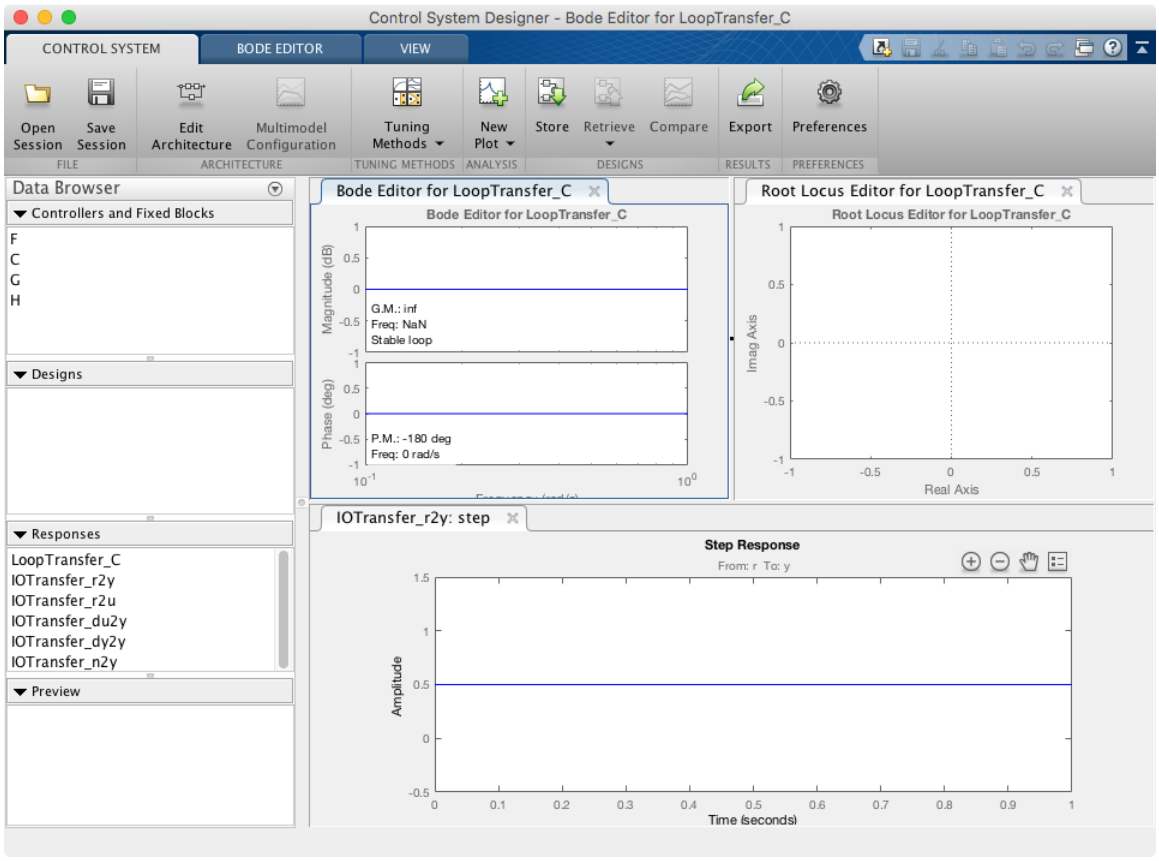
First, let's enter the system transfer function in the MATLAB Command Window

```
>> s=tf([1 0],1)
>> sys=1/((s+10)*(s^2+3*s+2))
```

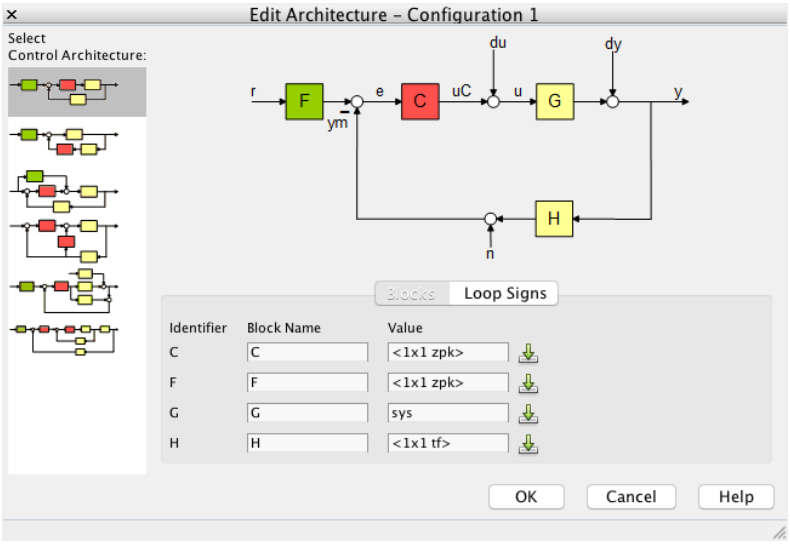
The design tool is initialized by typing `sisotool` at the command prompt.

```
» sisotool
```

The following window – or something similar, depending on your version of MATLAB – will open.

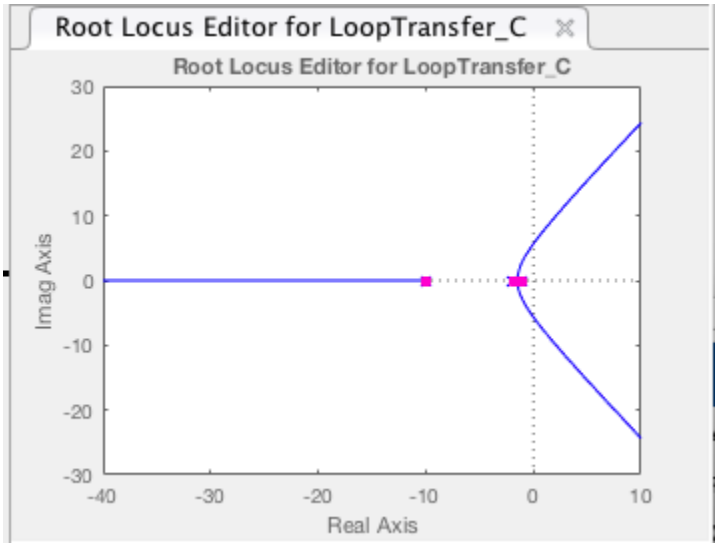


Click on "Edit Architecture". It opens up the following window



This shows the architecture of the control system that we will be working with. The yellow blocks represent fixed blocks (system and sensor dynamics) that can be set, and the green and red blocks are feedforward and feedback control blocks that can be designed.

Enter `sys` in the Value column for the G block to set this block to the transfer function you entered at the command line. Click OK, and the main designer window will change. Note that in the upper right corner, the root locus when the open loop system is `sys` is shown.

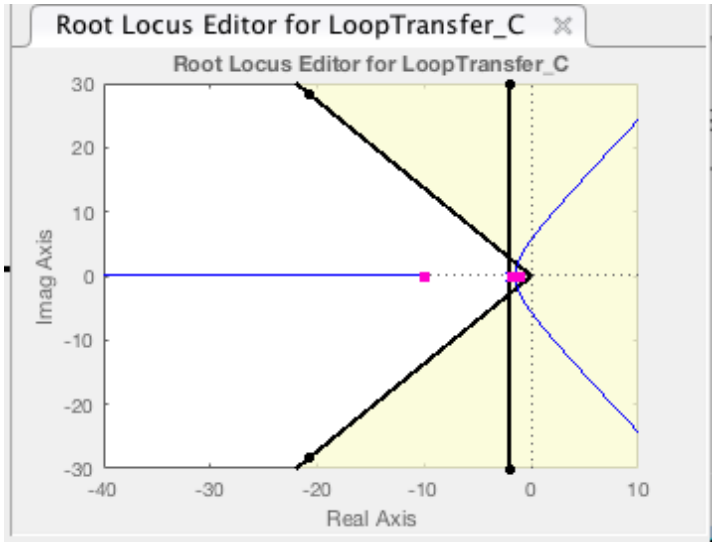


This is a root locus plot for the current loop gain $H(s)G(s)C(s) = \frac{1}{(s+10)(s^2+3s+2)}$. The blue lines are the three loci that make up the root locus plot (all of which are the same color in this tool), and the blue 'x's indicate the open loop poles. Since currently $C(s) = 1$, the closed loop poles are at the locations associated with $K_d = 1$, and no $s + a$ term. These closed-loop poles are indicated by the little pink squares.

Now, we will put our closed loop pole performance region on this plot. Right click on the root locus plot, and select "Design Requirements -> New". Note that settling time is one of the choices. Annoyingly, in this case Matlab defaults to 2% settling time, so we need to convert our 1% specification accordingly. Since the 2% settling time specification is $3.9/\omega_n$, while the 1% settling time specification is $4.6/\omega_n$, we should enter

$$t_s^{scaled} = t_s \times \frac{3.9}{4.6}.$$

So, choose settling time from the menu, and enter $2.3 \times \frac{3.9}{4.6} = 1.95$ seconds. Right click and select "Design Requirements -> New" again to enter a percent overshoot specification of less than 10%. The SISO Design window will now look like the following

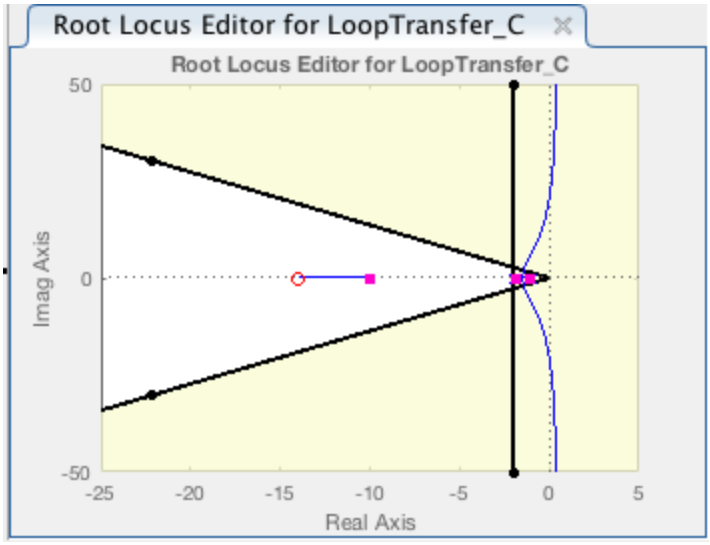


The white portion is the acceptable region, while the yellow shading indicates that closed loop poles in this region are not acceptable. Note that the root locus on the right does not enter the white region. These would be the dominant poles for the closed loop system, so proportional control alone will not be sufficient to meet the specifications. To

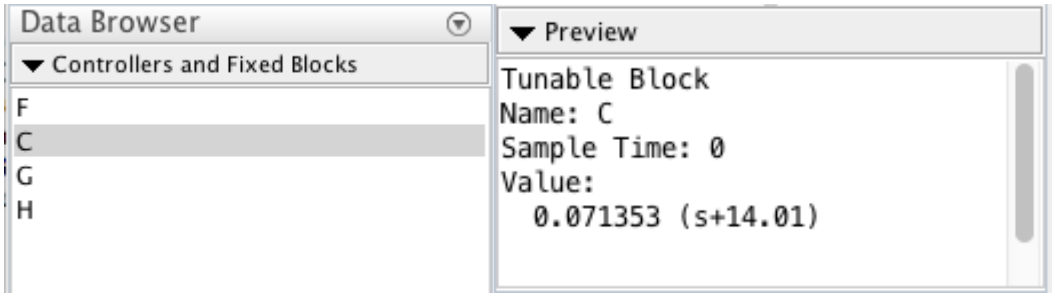
implement PD control, we will want to use a controller of the form

$$C(s) = K(s + a).$$

Step 3: We will start by adding the term $(s + a)$, which corresponds to a zero. On the upper left of the SISO Design window, there is a red circle. If you hover the cursor over it, a help box will say "add real zero". Click on the red circle, and then click on the location s=-14+j0 on the root locus plot. The root locus plot will now have changed to the following:

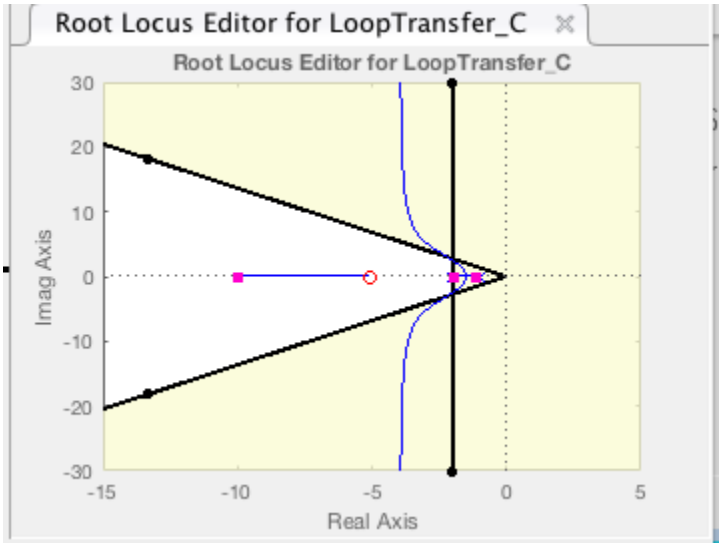


We can check that the control $C(s)$ has been changed by going back to the main window, and click on C in the Data Browser window. Below, the preview window gives the current value for $C(s)$.

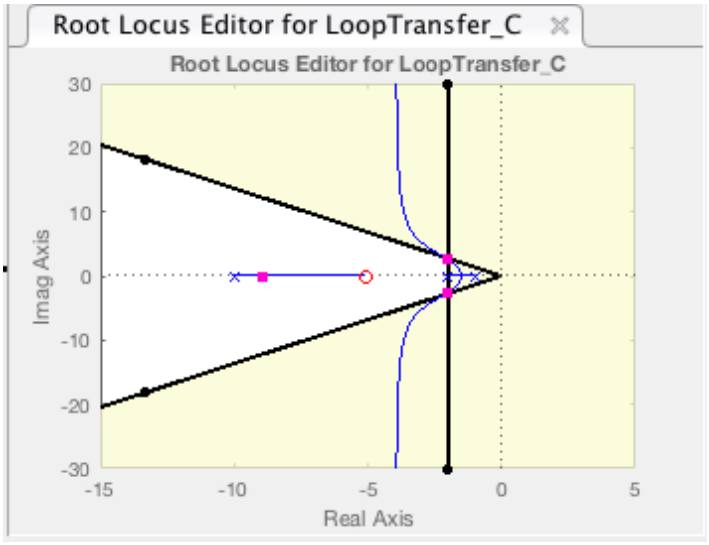


Note that it says that $C(s) = 0.071353(s+14.1)$. You can verify that this has a zero at -14.1, which is apparently where we actually clicked on the graph.

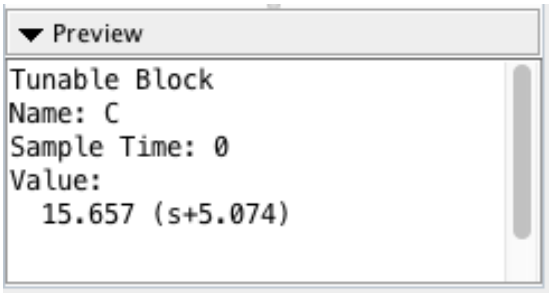
Step 4: Now go back to the SISO Design window. Hover the cursor over the zero that we inserted. You should see the cursor change to a hand symbol. Now click to "grab" the zero, and drag the cursor slowly to the right. Note that the root locus plot is updated to reflect the change in this zero location. Keep moving to the right until the root locus for the dominant poles just enters the acceptable region, and let go. The result should look something like the following



Now the root locus will go through the acceptable region, but we still need to choose a gain to get us there. Remember that as the gain increases the closed loop poles move along the root loci to either an open loop zero or infinity. Using the cursor, you can grab one of the pink squares and move it. Notice that this moves all the pink squares in tandem. Move the closed loop poles until the dominant poles lie inside the acceptable pole region. The result should look like the following:



To see what the resulting control $C(s)$ is, go back to the preview.



The designed controller is thus

$$C(s) = 15.657(s + 5.074) = 79.44 + 15.657s$$

Step 5: In other words, this controller corresponds to a proportional gain of $K_p = 79.44$ and a derivative gain of $K_d = 15.66$.

2.2 Verifying the Design

Step 6: We can verify if this meets the control specification by plotting the closed loop step response at the MATLAB command line or using Simulink. A useful command in MATLAB is `feedback`, which will simplify a feedback loop for us. If we type `help feedback` at the MATLAB command line, we get the following information:

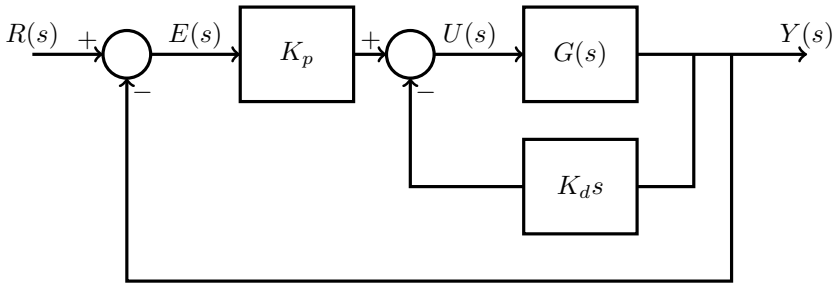
```
feedback  Feedback connection of two input/output systems.

M = feedback(M1,M2) computes a closed-loop model M for the feedback loop:

u --->O---->[ M1 ]----+---> y
|                |           y = M * u
+-----[ M2 ]<---+
...
```

(the help file also continues after this with more information) Note that the two arguments of the `feedback` command are the transfer functions in the forward and reverse path, respectively.

To check our design, we need to enter in the transfer functions for our implementation feedback loop, which is repeated here with loops separated for emphasis.

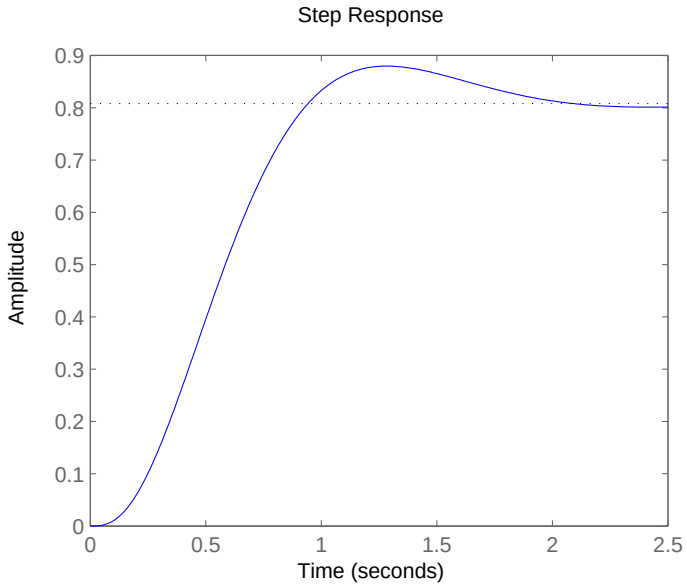


The following commands will find the closed loop transfer function $T(s) = \frac{Y(s)}{R(s)}$ for our design. Note that we assume that `s` has already been defined as the Laplace variable s and `G` has already been entered as done earlier in this section. We are also using slightly different gains from Section 2.1

```
>>  Kd = 16.01
>>  Kp = 84.25
>>  Inner = feedback(G,Kd*s) % find transfer function for inner loop
>>  T = feedback(Inner*Kp,1) % find closed loop transfer function
```

We can now find the closed loop step response

```
>> step(T)
```



and verify the step response specifications.

```
stepinfo(T,'SettlingTimeThreshold',.01)

ans =

RiseTime: 0.586472325105971
SettlingTime: 1.954623565535615
SettlingMin: 0.728640511065229
SettlingMax: 0.879716710297188
Overshoot: 8.855153766744039
Undershoot: 0
Peak: 0.879716710297188
PeakTime: 1.278713133961483
```

3 Lecture Highlights

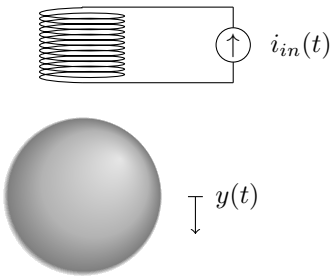
The primary takeaways from this article include

1. Root locus plots can be used to design PID controllers by showing whether, and for what values of a gain K, the dominant closed loop poles lie within the allowable region. The allowable region is determined from the rise time, settling time, and percent overshoot specifications for the closed-loop step response.
2. Matlab's `sisotool` is handy for root-locus based design because it allows easy manipulation of the controller gains and the location(s) of any controller poles and zeros. It can show the closed loop step response plot in real time to enable rapid tuning of these gains until specifications are met.

4 Quiz Yourself

4.1 Questions

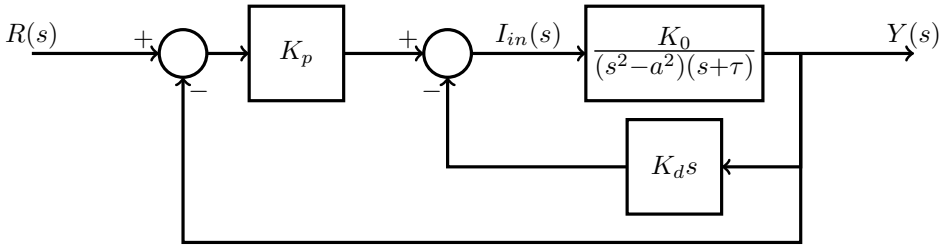
1. An electro-magnet can be used to balance a ball made out of a ferromagnetic material. However, for a fixed current, the force on the object becomes stronger when the ball moves closer, and weaker when the ball moves farther away, and thus the system is unstable in open loop.



The transfer function for a magnetic levitation system has the form

$$\frac{Y(s)}{I_{in}(s)} = \frac{K_0}{(s^2 - a^2)(s + \tau)}$$

You wish to use a PD control system to stabilize the mag-lev system and achieve a 1% settling time less than 1.3 seconds and %OS less than 10%.



Suppose the parameters are $K_0 = 10$, $a = 1$ and $\tau = 10$.

(a) Sketch the root locus for $K_d = 0$, and indicate the allowable closed loop pole regions.

(b) Using the technique from class, choose K_d and K_p to place the closed loop poles on or inside the allowable region.

(c) Find the closed loop transfer function $Y(s)/R(s)$. Using `Matlab` find the step response to verify that your design specifications have been met.

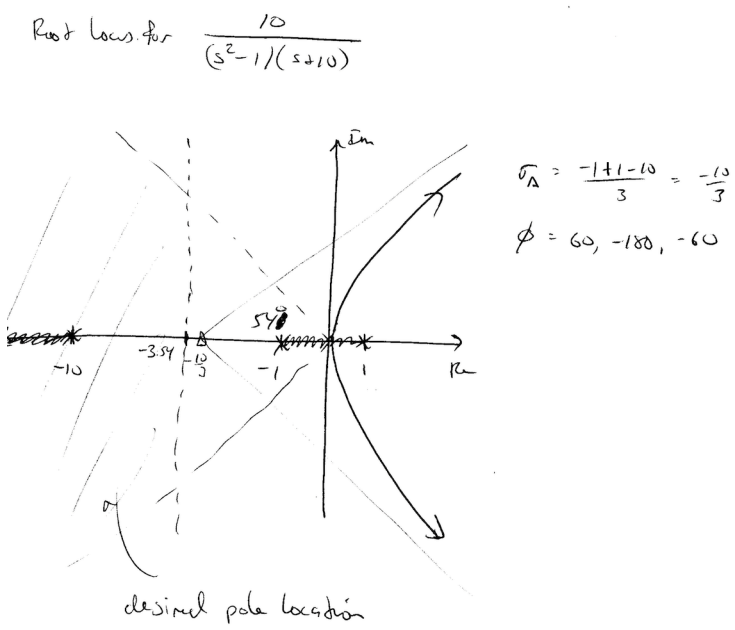
4.2 Solutions

1.

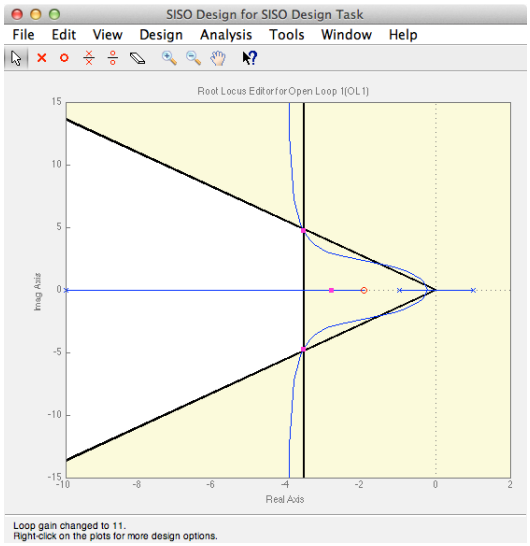
(a)

1% settling time ≤ 1.3 s $\Rightarrow \frac{4.6}{\sigma} \leq 1.3 \Rightarrow \sigma \geq 3.54$

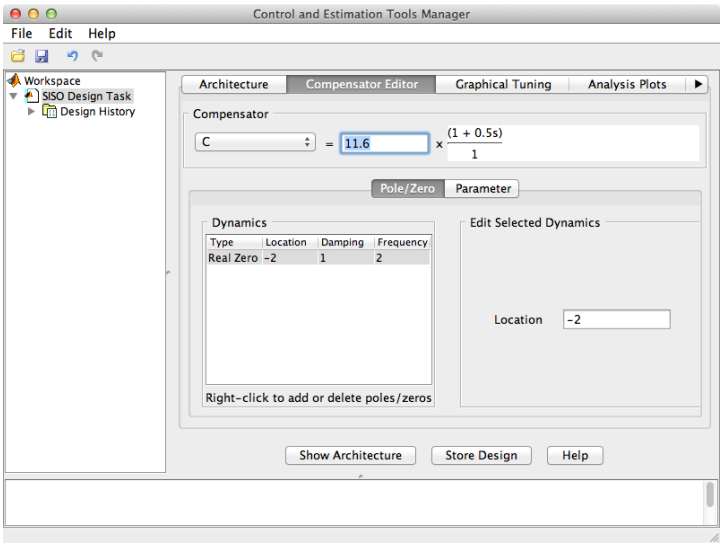
10% OS $\Rightarrow \zeta \geq \frac{-\ln(.1)}{\sqrt{\ln(.1)^2 + \pi^2}} = .59$ $\cos^{-1}(.59) = 54^\circ$



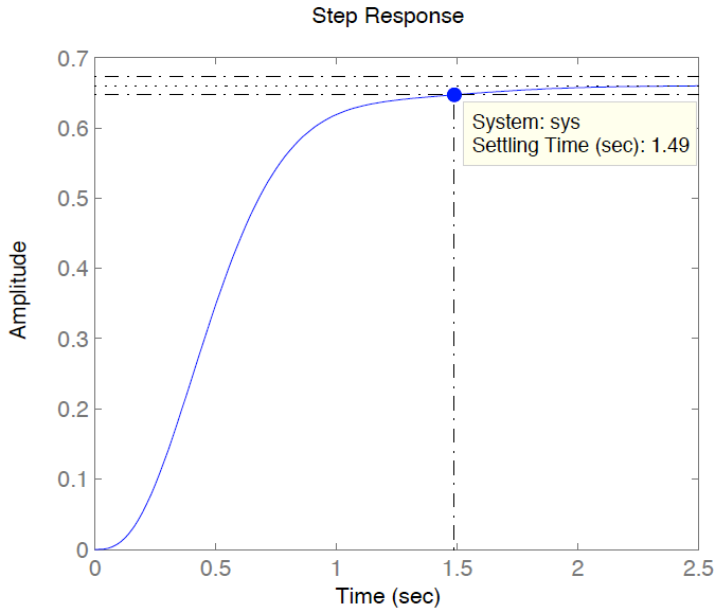
(b) Using SISOtool, we enter in the constraints and choose a zero to move the root locus inside the constraint region



The resulting controller is shown in the Control and Estimation Tools Manager



(c)



note that 80% spec is clearly met, but settling time is 1.5 s, rather that 1.3.

This problem is that our design placed a third closed loop pole at -2.93 which is not far enough to the left.