Design a PID controller s.t.

$T_p = 0.3\ s$ $\qquad G(s) = \dfrac{1}{s(s+2)(s+5)}$

$T_s = 2s$

① Evaluate OL system performance

TF = zpk ([ ], [-5, -2, 0], 1)

rlocus (TF)

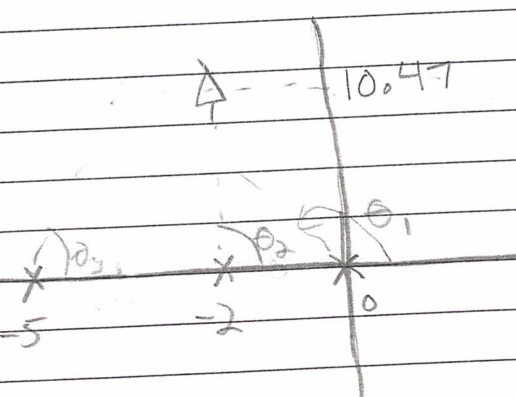
② Find desired poles $\quad -\sigma \pm \omega_d j$

$T_p = \dfrac{\pi}{\omega_d} \Rightarrow 0.3 = \dfrac{\pi}{\omega_d} \qquad \omega_d = 10.47$

$T_s = \dfrac{4}{\sigma} \Rightarrow 1 = \dfrac{4}{\sigma} \qquad \sigma = -2$

$s_{1,2} = -2 \pm 10.47 \Rightarrow$ cannot be achived on current RL

③ Determine where to put a zero s.t. these poles are on RL



$\theta_1 = 90° + \tan^{-1}\left(\dfrac{2}{10.47}\right)$

$= 100.81$

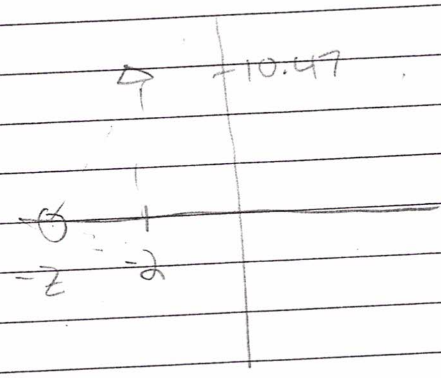
$\theta_2 = 90°$

$\theta_3 = \tan^{-1}\left(\dfrac{10.47}{5-2}\right) = 74.01$

$\Sigma\theta_z - \Sigma\theta_p = (2k-1)180$

$\sum \theta_z - \sum \theta_p = (2k+1)(180)$

$\theta_z = -84.82$



$\tan^{-1}\left(\dfrac{10.47}{z-2}\right) = 84.82$

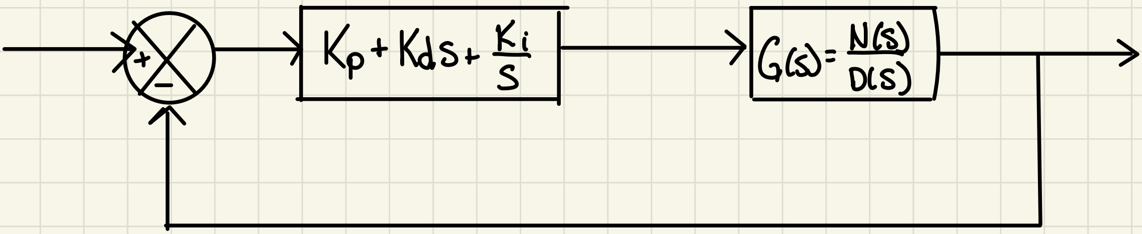
$\dfrac{10.47}{z-2} = 11.03$

$z - 2 = .949$

$z = 2.95$

RL_peaktime_settlingtime.m

Let's revisit our PID controllers with this in mind.



We can rewrite our controller as

$$G_c(s) = \frac{K_d s^2 + K_p s + K_i}{s}$$

In other words, it adds a pole at the origin and 2 zeros to the transfer function

If you have a P only $\Rightarrow$ scales $G(s)$, reduces rise time & SSE

If you have a PD only $\Rightarrow K_p + K_D s$ $\Rightarrow$ it adds a zero to the OLTF

$\Rightarrow$ adds damping to CLTF

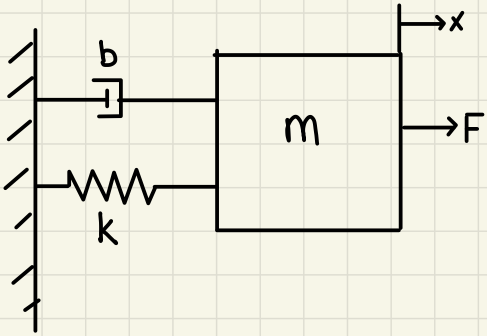
If you have a PI only $\Rightarrow \frac{K_p s + K_i}{s}$ $\Rightarrow$ adds a zero and pole at origin

$\Rightarrow$ improves stabity and reduces or eliminates SSE, but transient response degraded

PID removes SSE and decreases settling time while maintaining a reasonable transient response

Let's use our new root locus skills to help us understand this. We'll use our familiar simple harmonic oscillator as the plant, since we are already well versed in its behavior

SMD_PID_RL.m

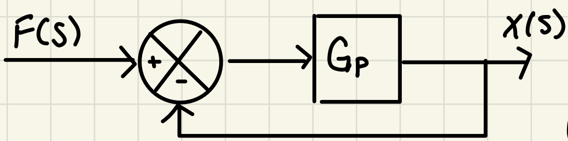


$$G_p(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2+bs+k}$$

Let's use $m=1$, $b=10$, $k=20$.

$\Rightarrow$ The poles are at -7.23 and -2.76, indicating an overdamped response to a step.
(Part 1 in Matlab file)

Now let's simply close the loop



$$OLTF = G_P$$

$$CLTF = \frac{G_P}{1+G_P}$$

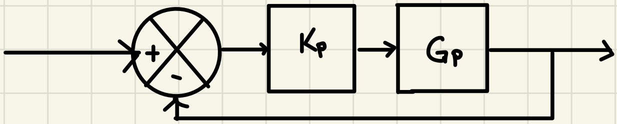
(part 2 in .m)

The poles of the CLTF are at $-7,-3$ and the step response is slightly improved.

This is like adding $K_P = 1$

* We consider both the transient response and the SSE when we think about performance

So let's look at adding K_p, and use the root locus to choose a K_p



$$OLTF = \frac{1}{ms^2 + bs + k} (K_p)$$

$$CLTF = \frac{K_p}{ms^2 + bs + k + K_p}$$

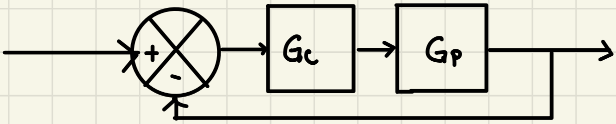
In the OLTF, we will keep the K_p seperate, since this is what the root locus will vary

(Part 3)

now, in our Matlab, let's choose a few values of K_p and see how the step response changes. We can click on the points in the root locus and see what values they correspond to.

$K_p = 5, 6, 10, 50, 500$

So let's look at adding K_p and K_d



$$G_c = K_p + K_d s = K_d\left(K_p/K_d + s\right) = K_d(z+s)$$

let's call this z, since it represents the new zero

$$CLTF = \frac{K_d s + K_p}{s^2 + (b + K_d)s + (k + K_p)}$$

$$OLTF = (z+s)G_p$$

part 4)

Caution: The OS% values shown in rlocus tool are not predictive of sys behavior - it tells what the system would do in the absence of other poles & zeros

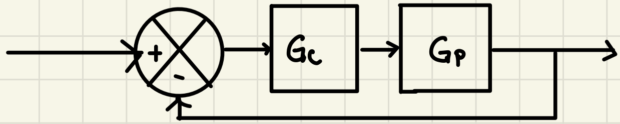
* Note that SSE is still an issue here!

* look at changing both K_d and z

⇒ we set K_d w/ RL, and K_p is then known from the ratio

⇒ RL only varies one constant parameter

Now let's do PI only



z

$$G_c = K_p + \frac{K_I}{s} = \frac{K_p s + K_I}{s} = K_p\left(\frac{s + K_I/K_p}{s}\right)$$

$$CLTF = \frac{K_p s + K_I}{s^3 + 10s^2 + (120 + K_p)s + K_I}$$

$$OLTF = \left(\frac{s + K_I/K_p}{s}\right) G_c$$

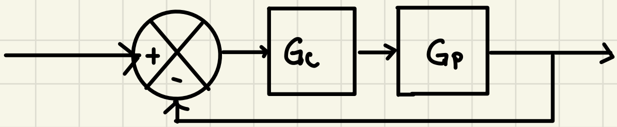
Part 5

look at $z = .5, 5, 10$ across a range of K_p

* Observe transient response degredation
* Elimination of SSE

⇒ We are varying K_p and letting K_I be calculated from z

And finally, let's look at all 3



$$G_c = K_p + sK_d + \frac{K_i}{s} = \frac{K_d s^2 + K_p s + K_i}{s}$$

$$= K_d \frac{\left(s^2 + \overset{a}{\frac{K_p}{K_d}} s + \overset{b}{\frac{K_i}{K_d}}\right)}{s} \qquad a = \frac{K_p}{K_d} \quad b = \frac{K_i}{K_d}$$

Adding 2 zeros and a pole at 0 to the OLTF

start $a = 10, \; b = 8$

$a = 1, \; b = 20$

⋮

for varying K

⇒ response has 0 SSE

⇒ relative size of a & b has large impact on the transient response

⇒ PIDs require <u>tuning</u> to get desired response

⇒ RL is a tool that can help with this

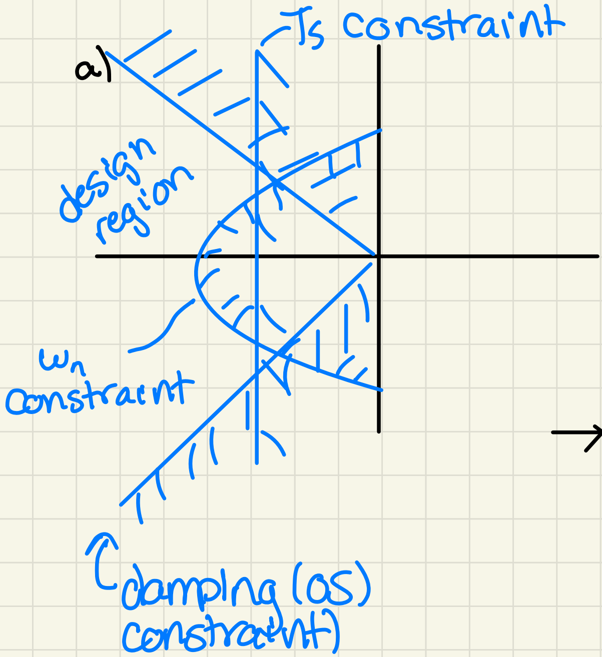
⇒ The integrator is key to eliminating SSE

A summary of what we observed in our root locus PID design example:

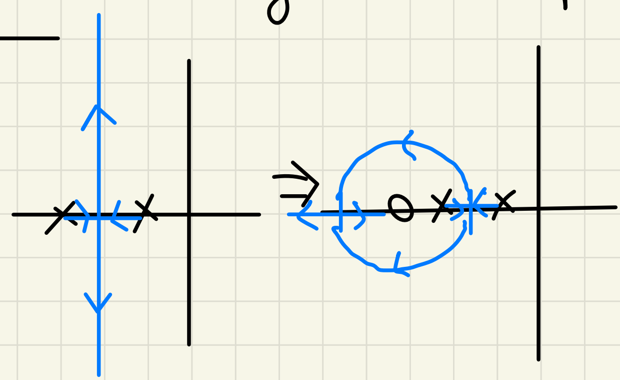
1) Our intuitive estimates of the response were more accurate in cases where the system behaved as second-order (ish!)
   - ⇒ our rules about the s-plane apply to second order systems only
   - ⇒ but, we could still see general performance trends

2) We could target areas to choose our K (or even to move our locus) to get behavior we wanted.

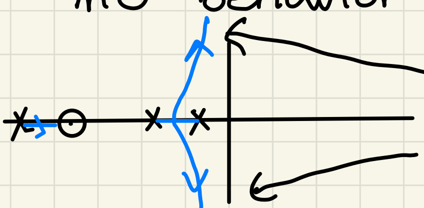
a)



b) adding poles or zeros by designing our controller changed our shape



3) Poles closer to the real axis often dominated the behavior ⇒ dominant poles



for ex. we had to change our arch here b/c Ts was always high

4) Since we could only vary one K in the RL, we used the zero locations (which are located from the ratios K_P/K_D, K_I/K_P, K_I/K_D, etc) to get our locus into the area of the design space we wanted, then we could select a K value on that locus to get desired behavior

5) The addition of an integrator was necessary to eliminate SSE in response to a step input

Putting all of this together suggets a general process for using RL for controller design

1) Determine the desired region of the s-space to meet performance criteria

2) Sketch the root locus for the system and determine how it needs to change to go through the target area

3) Design a controller that adds poles/zeros as needed to move locus to target area

4) Find the gain K on that locus to achive desired behavior

5) Iterate as needed

Now, let's consider how different types of inputs might impact our design choices