Christopher Lum
lum@uw.edu

Lecture 07d
Using Root Locus to Meet Performance Requirements

Lecture is on YouTube

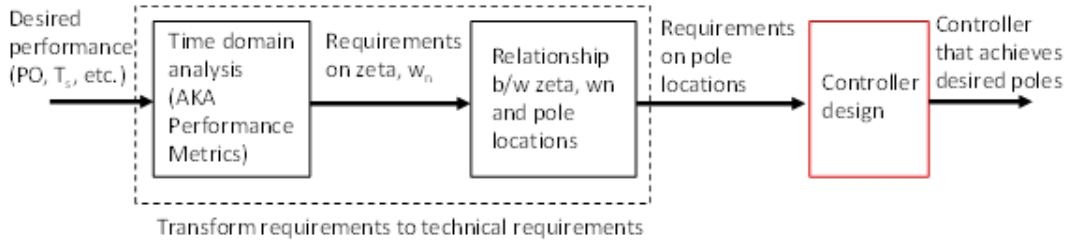
YouTube video entitled 'Using Root Locus to Meet Performance Requirements' covering this is located at https://youtu.be/rNYHww84juM.

Outline

-Combining with Root Locus to Design a Controller

Combining with Root Locus to Design a Controller

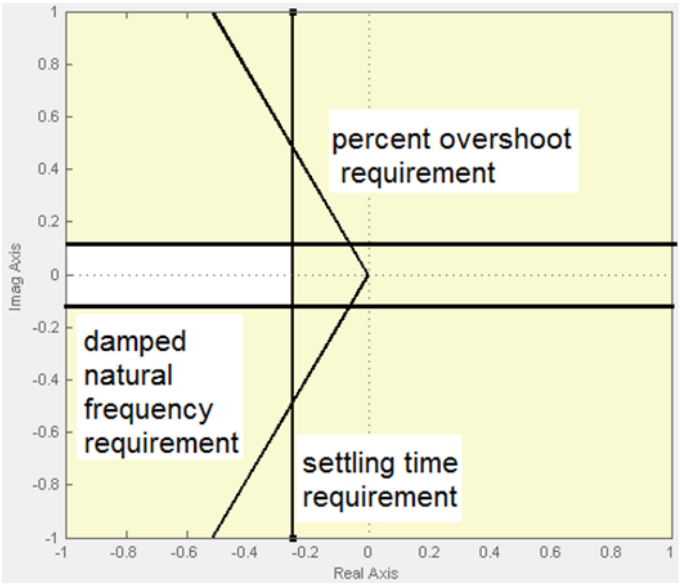
We can now turn our attention to the red box in the workflow diagram.



In the previous lecture, we investigated an example with the following performance requirements

PO < 20 × %	⇒	$\theta_{\text{required}} = 62.9^o$
T_s < 10 s w/ $\delta = 0.1$	⇒	$\sigma_{\text{required}} = 0.23$
oscillation frequency < 0.1 rad/s	⇒	$\omega_d = 0.1$

We saw that this led to the following permissible locations of poles

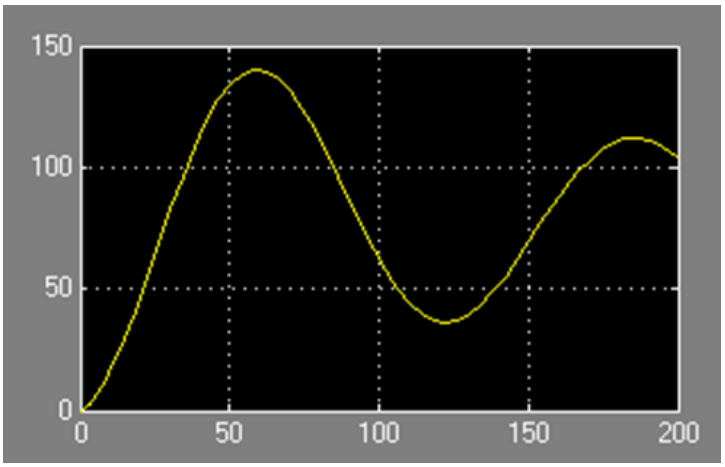


We now consider the plant model of

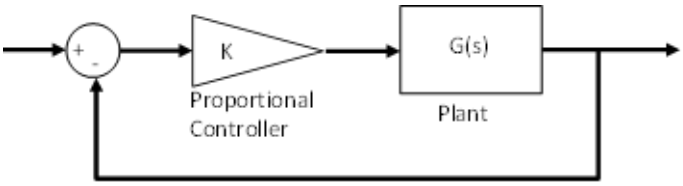
$$G(s) = \frac{s+0.2}{s^2+0.01\,s+0.0025}$$

```
G[s_] = (s + 0.2)/(s^2 + 0.01 s + 0.0025);
```

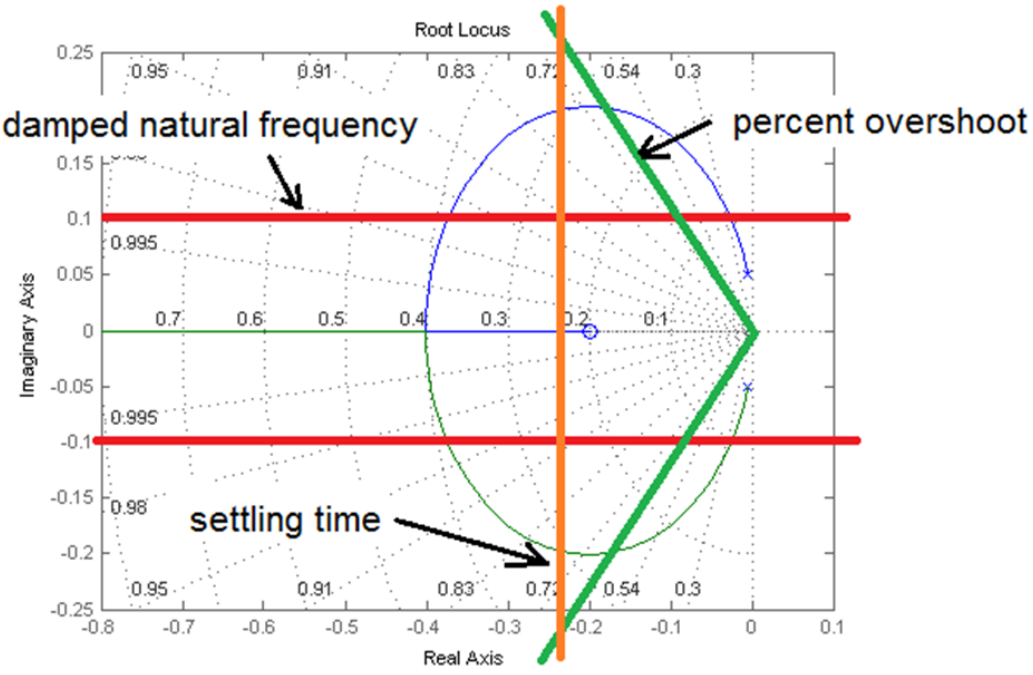
We can look at the open loop response of this system to a step function



As can be seen, this clearly does not meet the design specifications (percent overshoot looks much more than 20% and settling time is much too long). We can propose a simple proportional controller as shown below.



We can design the proportional gain, K, using the root locus method. The root locus for the closed loop system is shown below. The percent overshoot requirement is shown in green, the settling time requirement is shown in orange, and the damped natural frequency is shown in red.



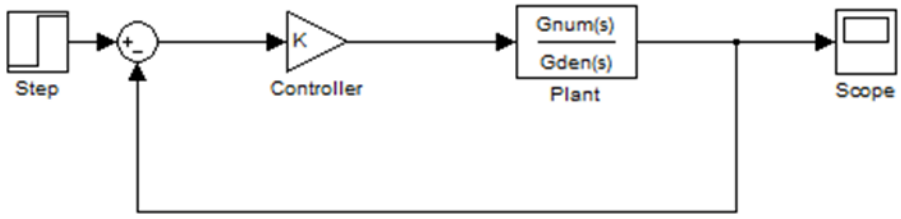
From this, we see that the damped natural frequency requirement is the hardest to meet but we should be able to achieve all 3 constraints simultaneously if we choose

$K \approx 0.74$

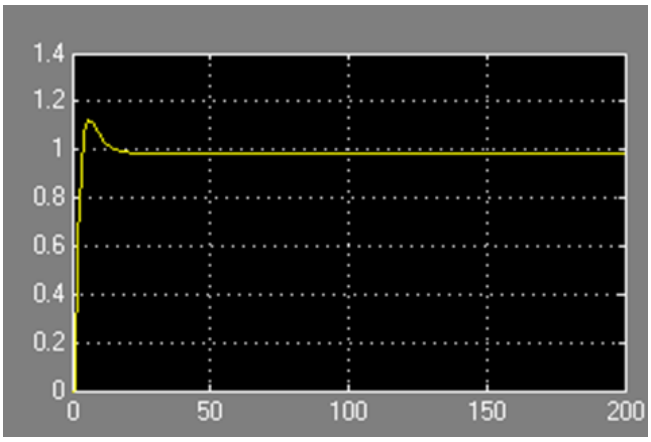
```
T[s_] = K G[s] / (1 + K G[s]) // Simplify;
Δ[s] = Denominator[T[s]];
Solve[(Δ[s] /. {K → 0.74}) == 0, s]
```

```
{{s → -0.375 - 0.099373 i}, {s → -0.375 + 0.099373 i}}
```

We can verify the behavior of the closed loop system by constructing a Simulink model of the system



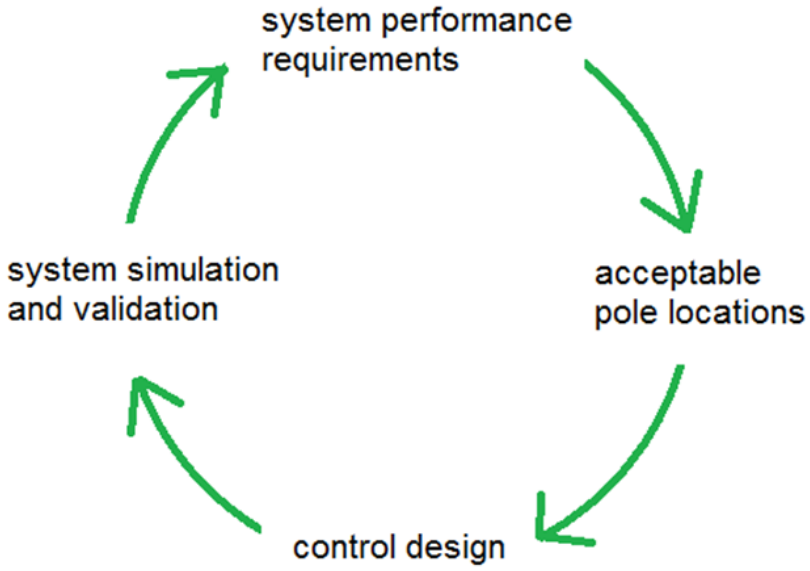
And the response of the system to a unit step is shown below



So we finally are able to link performance requirements to pole locations, which we can then link to controller design using root locus methods.

So a workflow for designing a controller to meet performance requirements might take the format of:

1. System performance requirements dictate acceptable pole locations.
2. Acceptable pole location drives controller design (i.e. using root locus methods).
3. Control design tested in simulation and/or in hardware.
4. Testing results validate original performance requirements.



Note that the above technique of manually drawing the acceptable pole locations is a little tedious. In a future lecture, we will discuss a tool called the Control System Designer in Matlab that will make this much easier.

Furthermore, the controller we investigated here was exceedingly simple (a simple proportional controller). In a future lecture, we will discuss using the aforementioned Control System Designer to

design more complicated controllers like a full PID controller.