Sample Instructional Activity: Lecture

Week 9

Agenda: Control Design Root Locus

- We will learn how to design a controller for a system using Root Locus.
- We will cover how to design each of the controllers introduced in previous modules to satisfy some control objectives
- You will attempt to design a cruise controller for a vehicle using MATLAB (this livescript)

Resources: Lecture hall with blackboard and projector. 2 hour class.

Last Week

- P, PD, PI and PID Control
- Lead, Lag and Lead-Lag Control

This Week

- How to use Root Locus to design these controllers

Learning Outcomes

- Distinguish between the typical use of each control and the effect on performance
- Formulate system performance specifications into Root Locus design criteria
- Employ Root Locus for control design
- Simulate controllers on MATLAB

The example of cruise control in cars will be used to demonstrate the taught concepts and learning outcomes.

The LiveScript is posted in the course website. *Access it to be able to follow the design activity in the end*.

PI Control

Goal: Increase system type to eliminate steady state error.

$$G_c = \frac{K(s+a)}{s} = K + Ka\frac{1}{s}$$

Implementation: Place the zero at $-a$ close to the origin

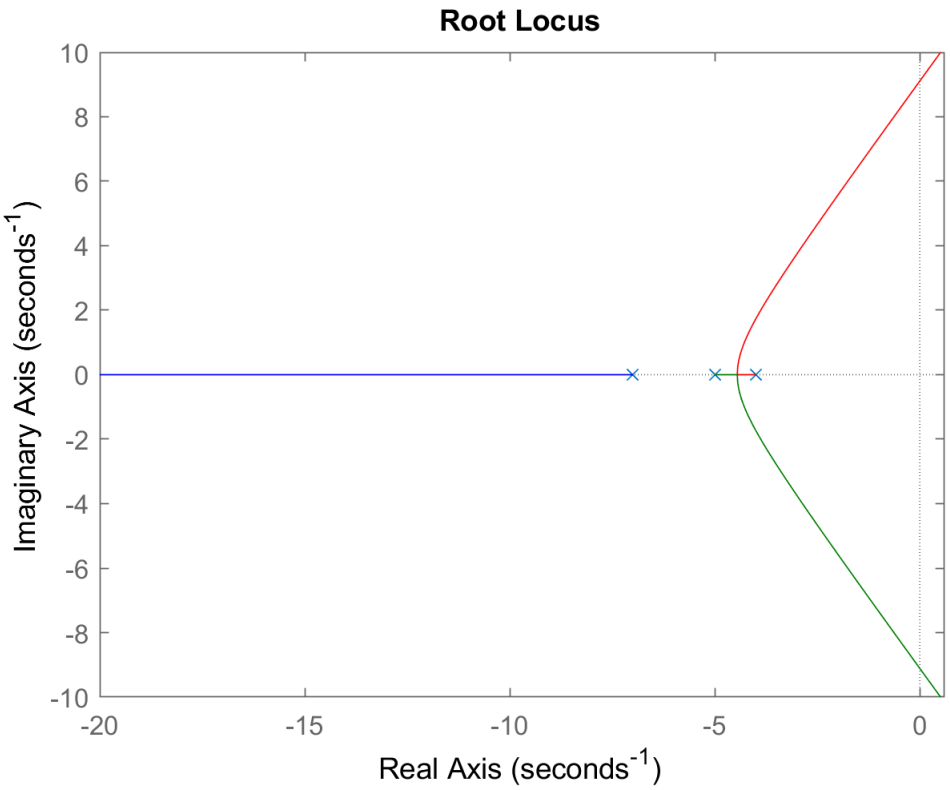
Explain steady state error elimination using Final Value Theorem on board

Example:

Uncontrolled System

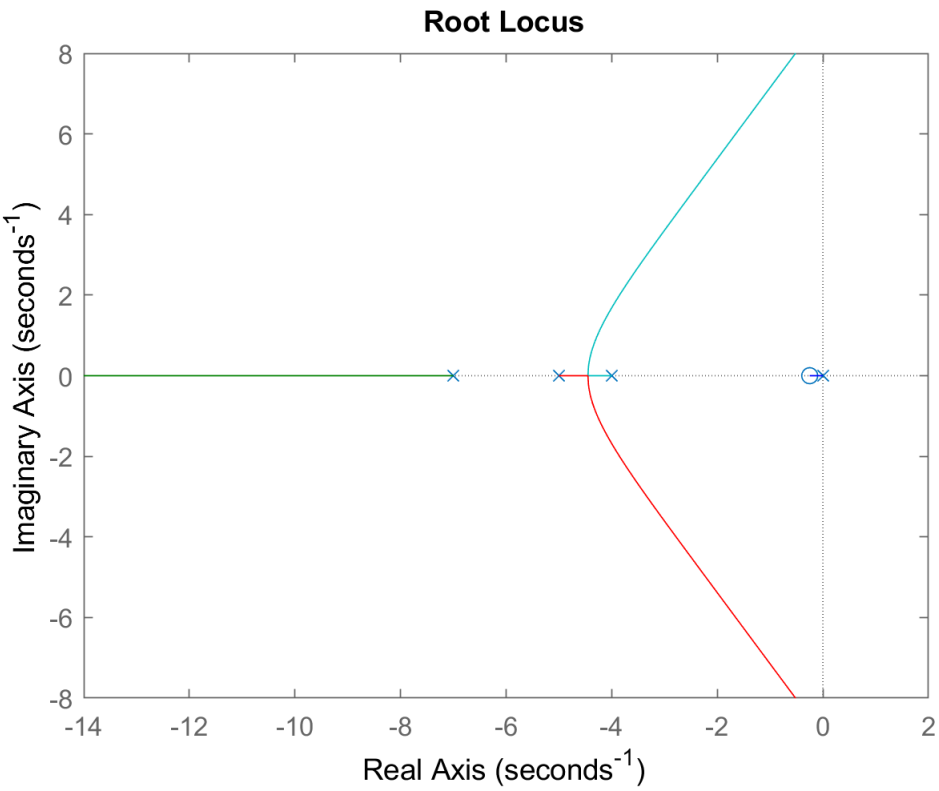
```
s = tf([1 0], 1);
```

```
Gp = 1/(s+4)/(s+5)/(s+7);
rlocus(Gp)
```



Controlled system

```
Gc = tf([1 0.1], [1 0]);
rlocus(Gc*Gp)
```



The Root Locus does not change significantly with the addition of the controller (i.e. performance does not change considerably). However, the increase in system type eliminates steady state error.

Lag Control

Goal: Reduce steady state error. (Approximate PI control)

$$G_c = \frac{K(s+z)}{(s+p)}$$

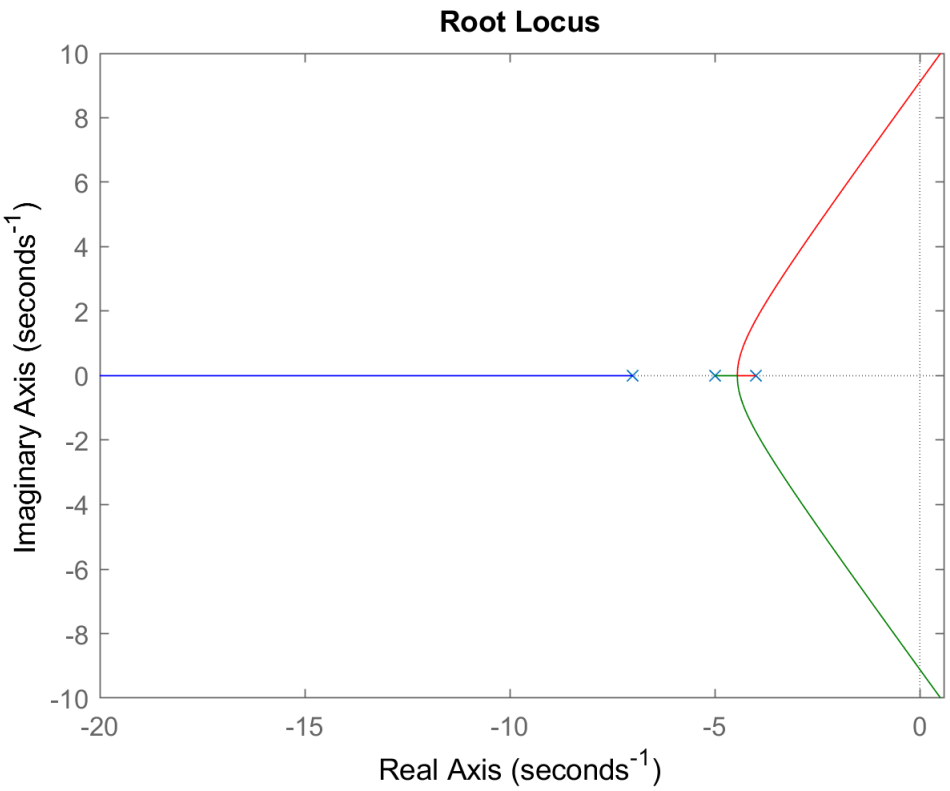
Implementation: Place the zero and pole close to the origin such that $-z < -p < 0$

Explain reduction in steady state error using Final value theorm on board

Example:

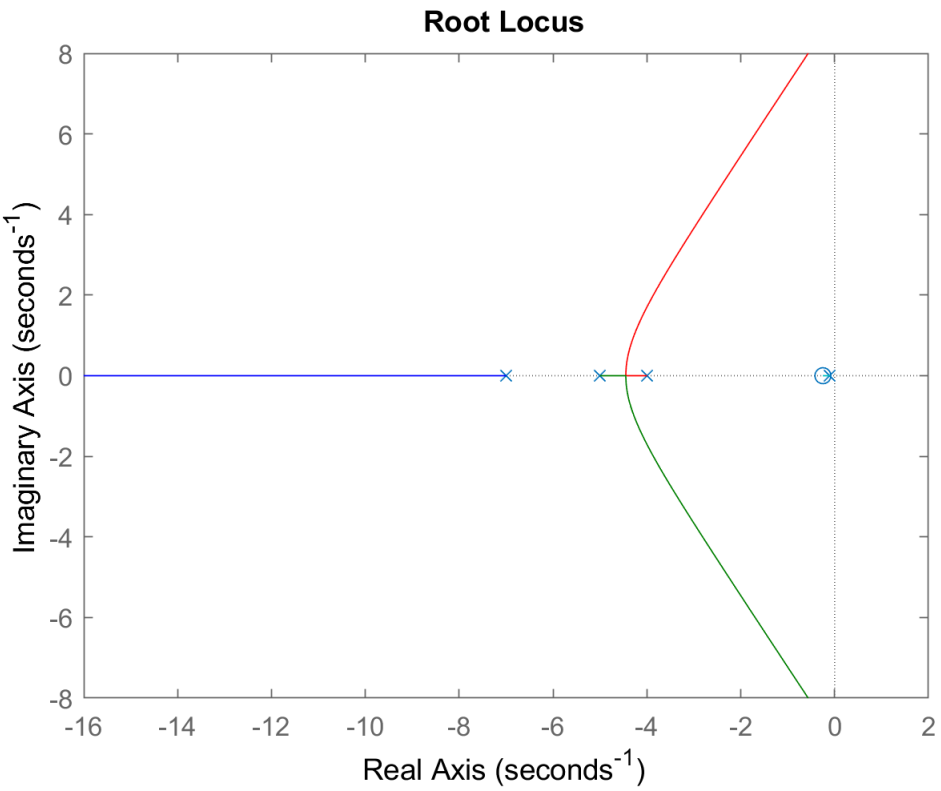
Uncontrolled System

```
s = tf([1 0], 1);
Gp = 1/(s+4)/(s+5)/(s+7);
rlocus(Gp)
```



Controlled system

```
Gc = tf([1 0.1], [1 0.05]);
rlocus(Gc*Gp)
```



The Root Locus does not change significantly with the addition of the controller (i.e. performance does not change considerably).

PD Control

Goal: Modify system performance by enabling a poles location that was previously unattainable.

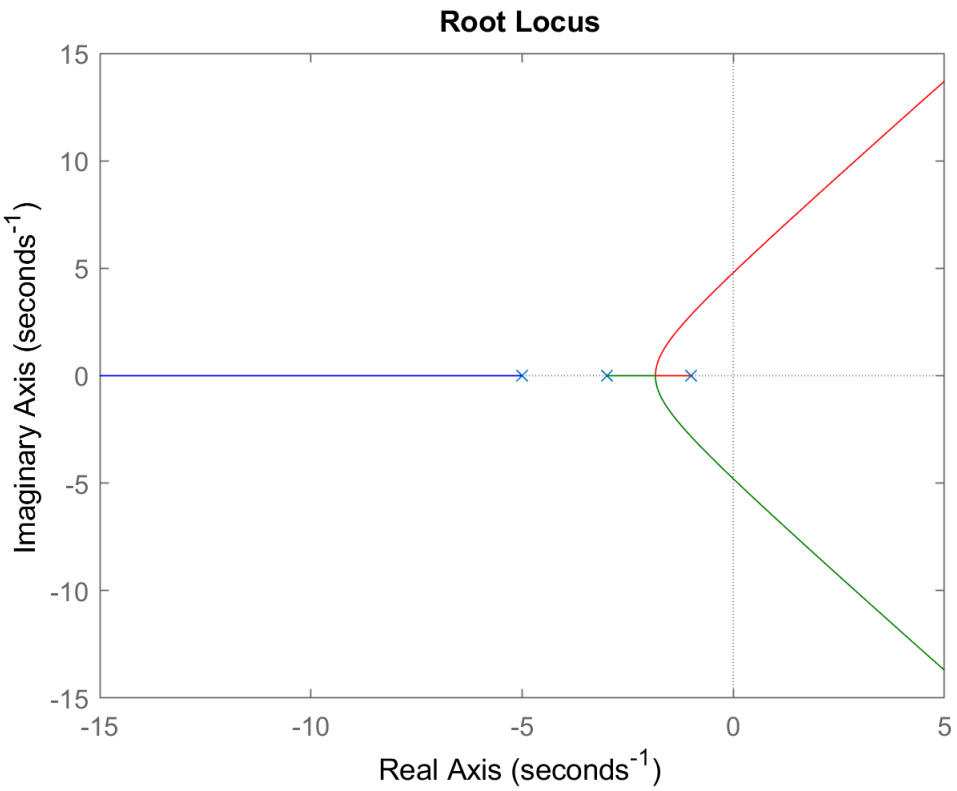
$G_c = K(s + z)$

Implementation: *Details on board*

Example:

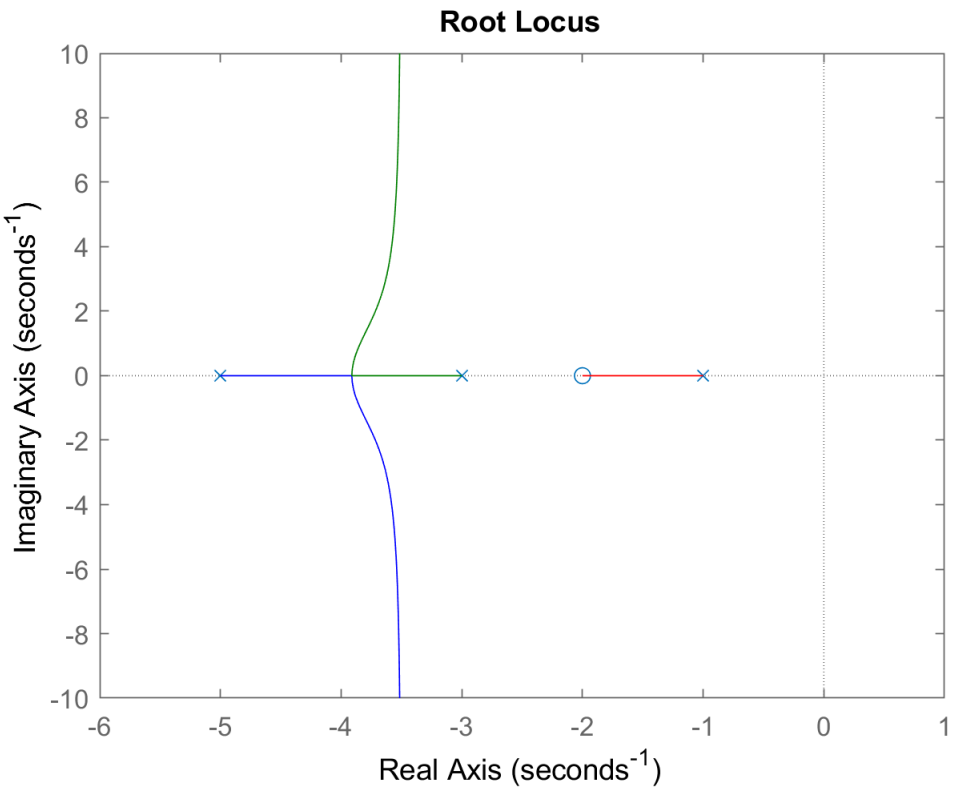
Uncontrolled System

```
s = tf([1 0], 1);
Gp = 1/(s+1)/(s+3)/(s+5);
rlocus(Gp)
```



Controlled system

```
Gc = tf([1 2], 1);
rlocus(Gc*Gp)
```



The Root Locus changes with the addition of the controller (i.e. performance changes).

Lead Control

Goal: Increase system type to eliminate steady state error.

$$G_c = \frac{K(s+z)}{(s+p)}$$

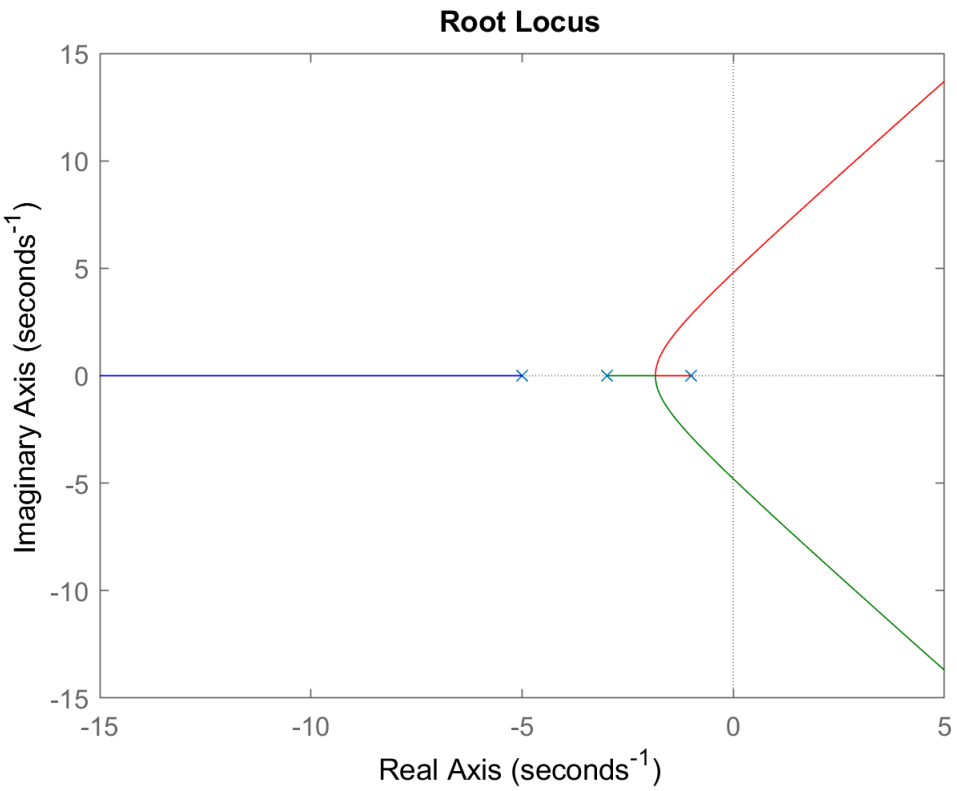
Implementation: Place $-p << -z < 0$

Details on board

Example:

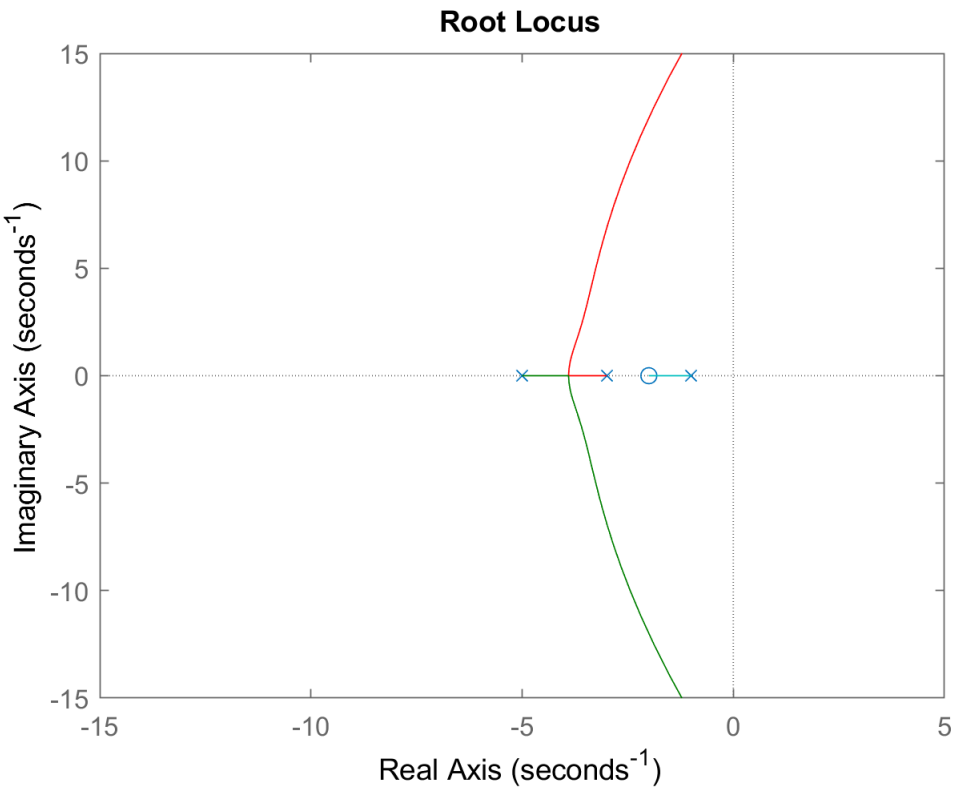
Uncontrolled System

```
s = tf([1 0], 1);
Gp = 1/(s+1)/(s+3)/(s+5);
rlocus(Gp)
```



Controlled system

```
Gc = tf([1 2], [1 50]);
rlocus(Gc*Gp)
axis([-15 5 -15 15]);
```



The Root Locus does not change significantly with the addition of the controller (i.e. performance does not change considerably). However, the increase in system type eliminates steady state error.

Cruise Control

1. System Model

```
clc
m = 1000; % mass of vehicle is 1000 kg
b = 50; % coefficient of air drag is 50 Ns/m
G = tf(1, [m b])
```

```
G =

      1
  -----------
  1000 s + 50

Continuous-time transfer function.
```

2. Control Performance Specifications

Rise time < 5 sec

Overshoot < 10%

Steady state error < 2%

This imposes the following constraints on our system:

$$T_r = \frac{1.8}{\omega_n} < 5$$

ω_n must be larger than:

```
wn = 1.8/5
```

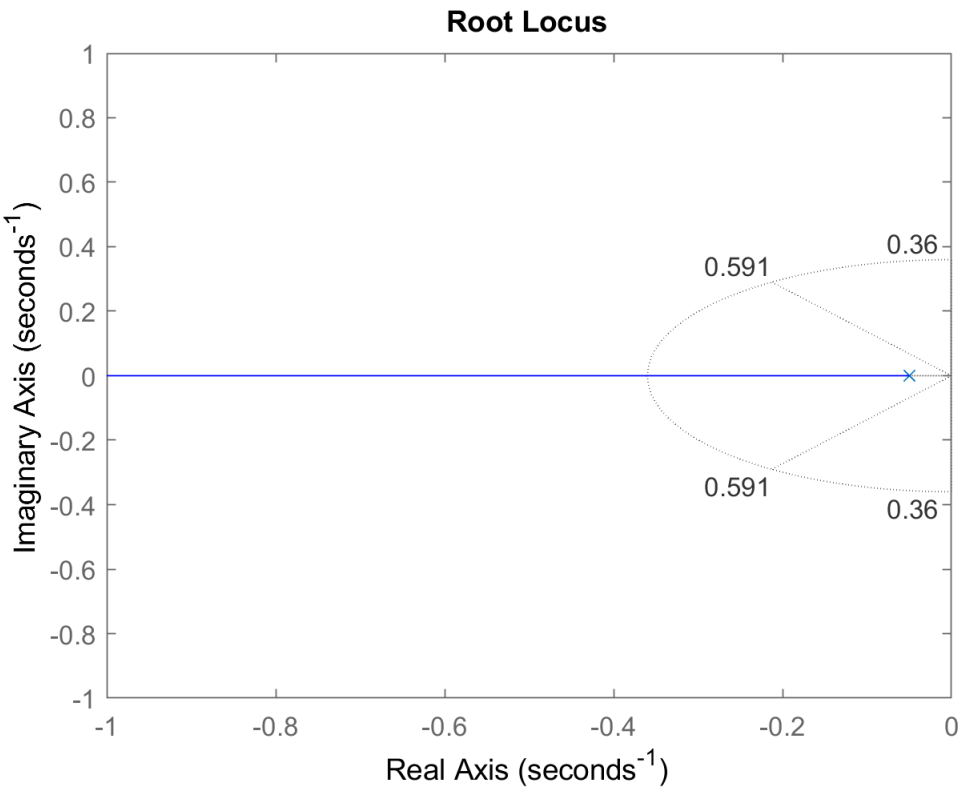
```
wn = 0.3600
```

Damping ratio must be larger than:

```
damping_ratio = abs(log(0.1))/(sqrt(pi^2 + log(0.1)^2))
```

```
damping_ratio = 0.5912
```

```
rlocus(G)
axis([-1 0 -1 1]);
sgrid(damping_ratio, wn)
```

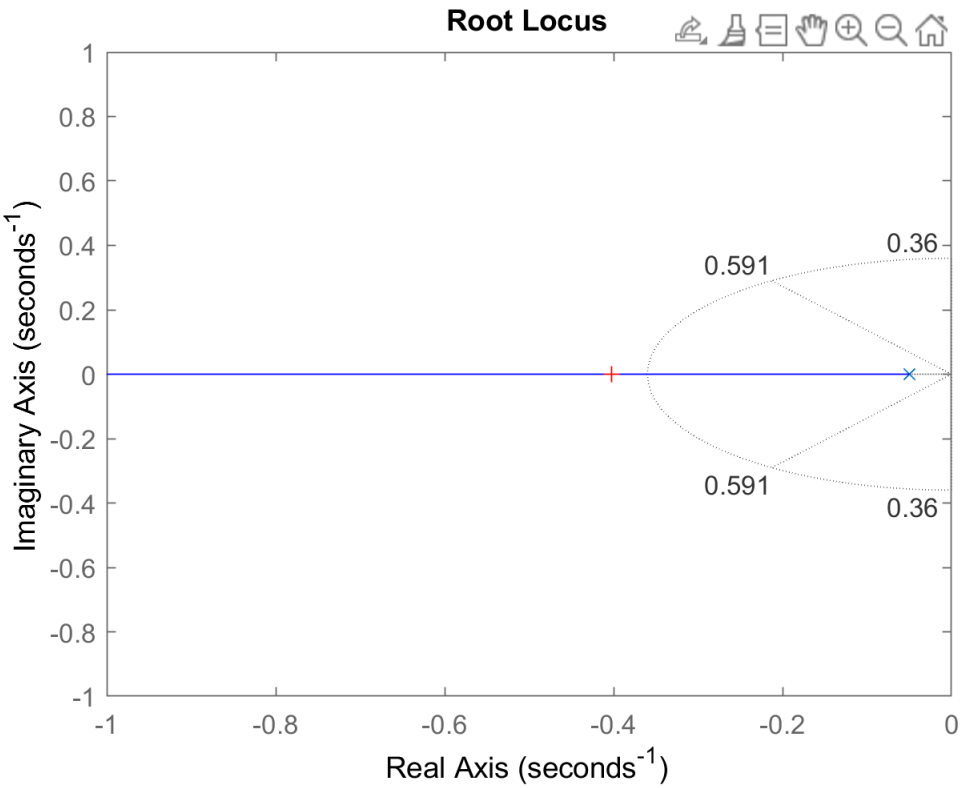


3. Proportional Controller

Looking at the root locus, it seems we can design a proportional controller to meet the requirements.

```
[Kp,poles]=rlocfind(G)
```

```
Select a point in the graphics window
```

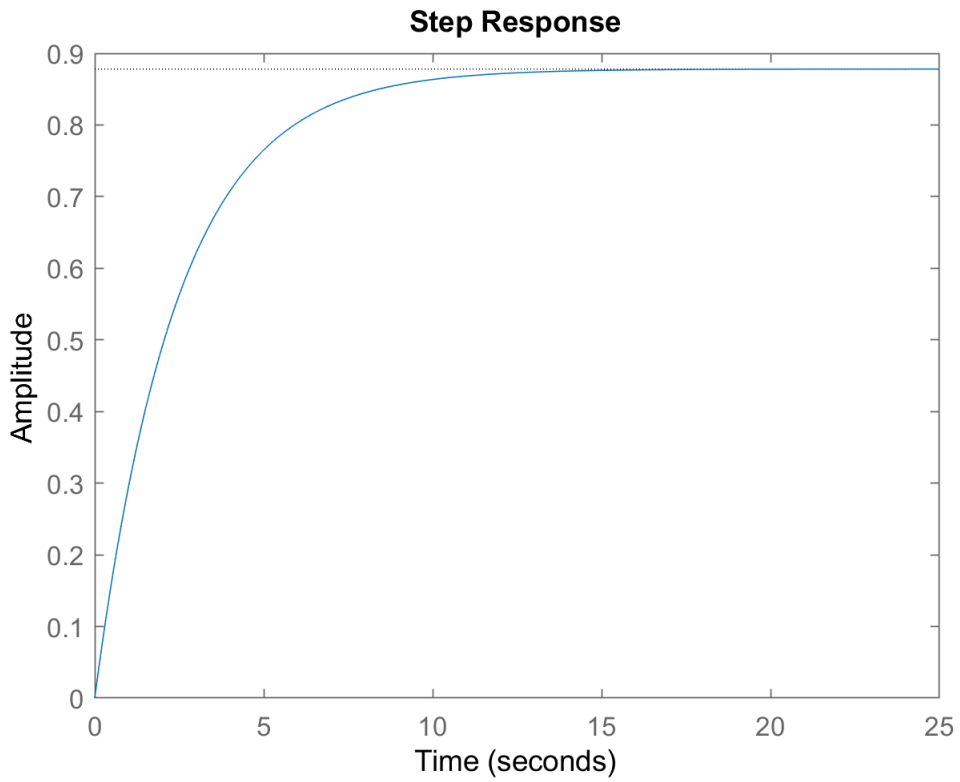



```
selected_point = -0.4024 - 0.0025i
Kp = 352.3784
poles = -0.4024
```

```
cls = feedback(Kp*G,1)
```

```
cls =

      360
  ------------
  1000 s + 410

Continuous-time transfer function.
```

```
step(cls)
```




```
stepinfo(cls)
```

```
ans = struct with fields:
        RiseTime: 5.3592
    SettlingTime: 9.5427
     SettlingMin: 0.7942
     SettlingMax: 0.8780
       Overshoot: 0
      Undershoot: 0
            Peak: 0.8780
        PeakTime: 25.7245
```

We must pick a further point (to the left) on the root locus to meet the rise time (due to appr.) and steady state error.

Kp must be greater than 2450 to satisfy the steady state requirement

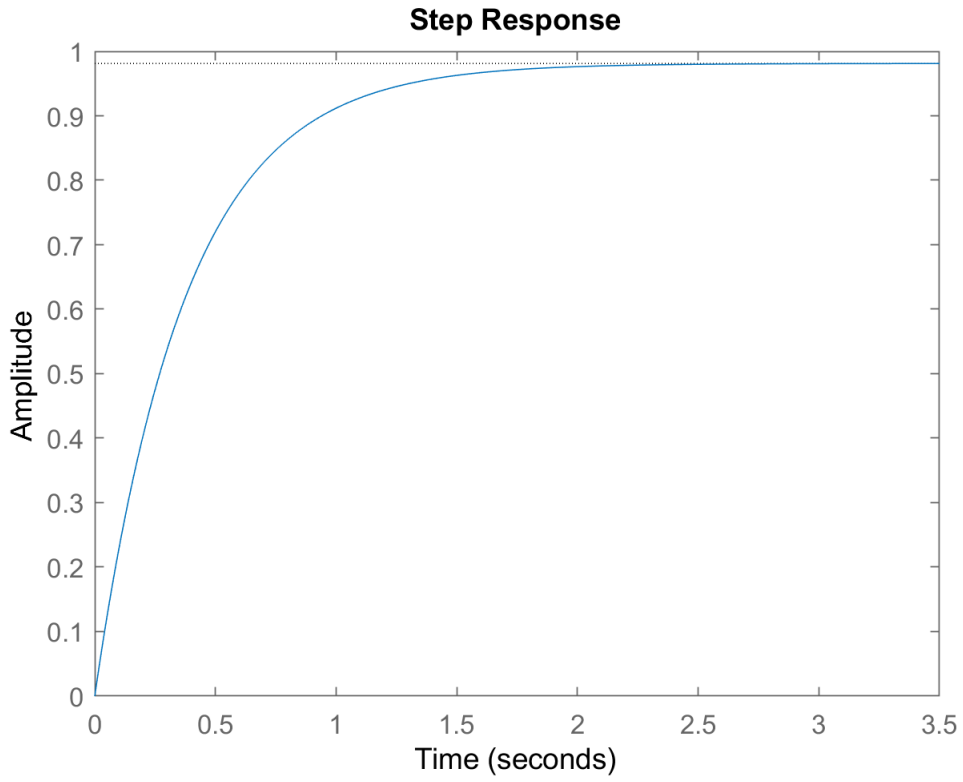
```
Kp = 2600
```

```
Kp = 2600
```

```
cls = feedback(Kp*G,1)
```

```
cls =

      2600
  -------------
  1000 s + 2650

Continuous-time transfer function.
```

```
step(cls)
```



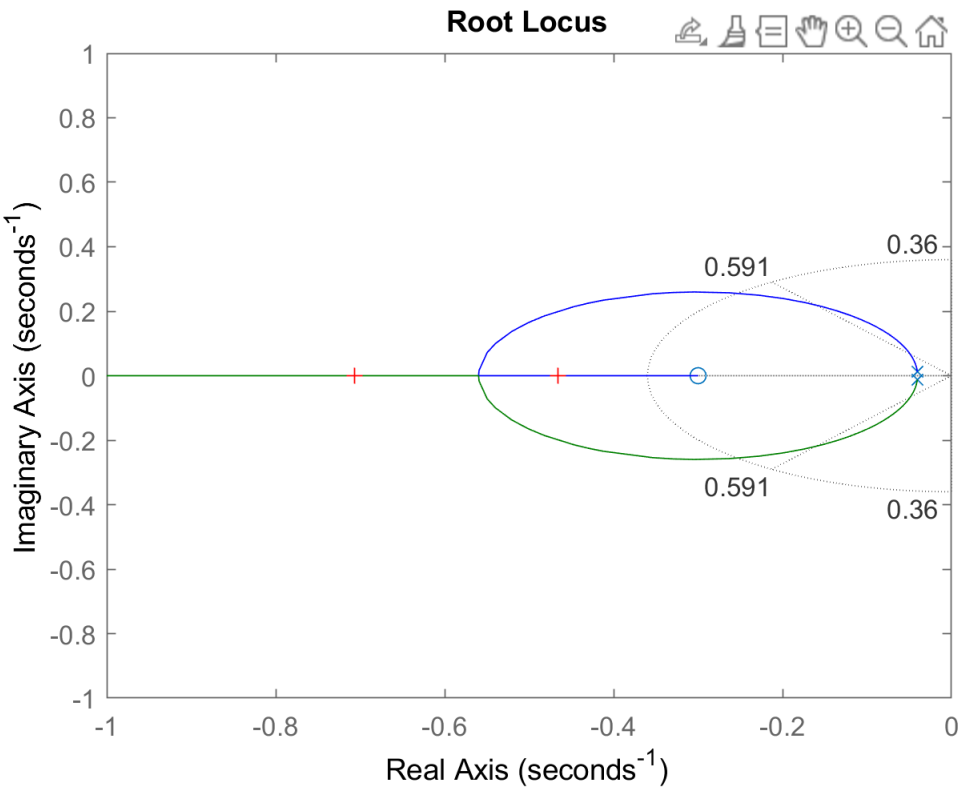

```
stepinfo(cls)
```

```
ans = struct with fields:
        RiseTime: 0.8291
    SettlingTime: 1.4763
     SettlingMin: 0.8874
     SettlingMax: 0.9811
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9811
        PeakTime: 3.9796
```

4. Passive Controller

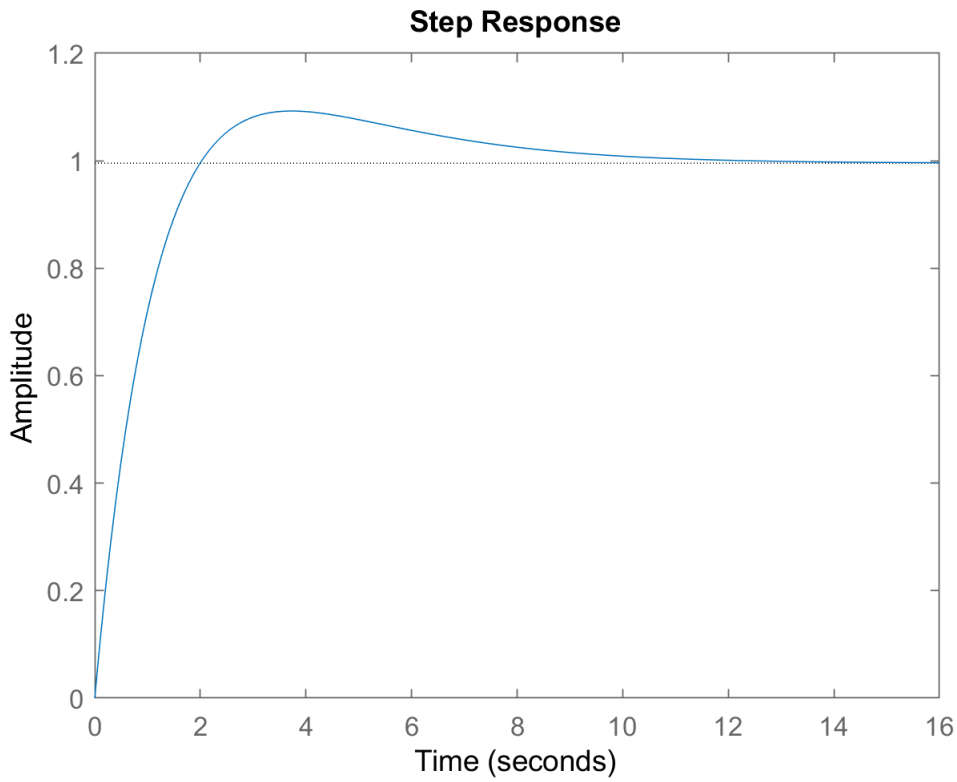
```
zc = 0.3; pc = 0.03;
Gc = tf([1 zc], [1 pc]);
cls = feedback(G*Gc,1);
rlocus(cls);
axis([-1 0 -1 1]);
sgrid(damping_ratio, wn);

[Kp,poles]=rlocfind(cls)
```

```
Select a point in the graphics window
```




```
selected_point = -0.4659 - 0.0050i
Kp = 1.0915e+03
poles = 2×1
   -0.7064
   -0.4661
```

```
cls = feedback(Kp*G*Gc,1);
step(cls)
```




```
stepinfo(cls)
```

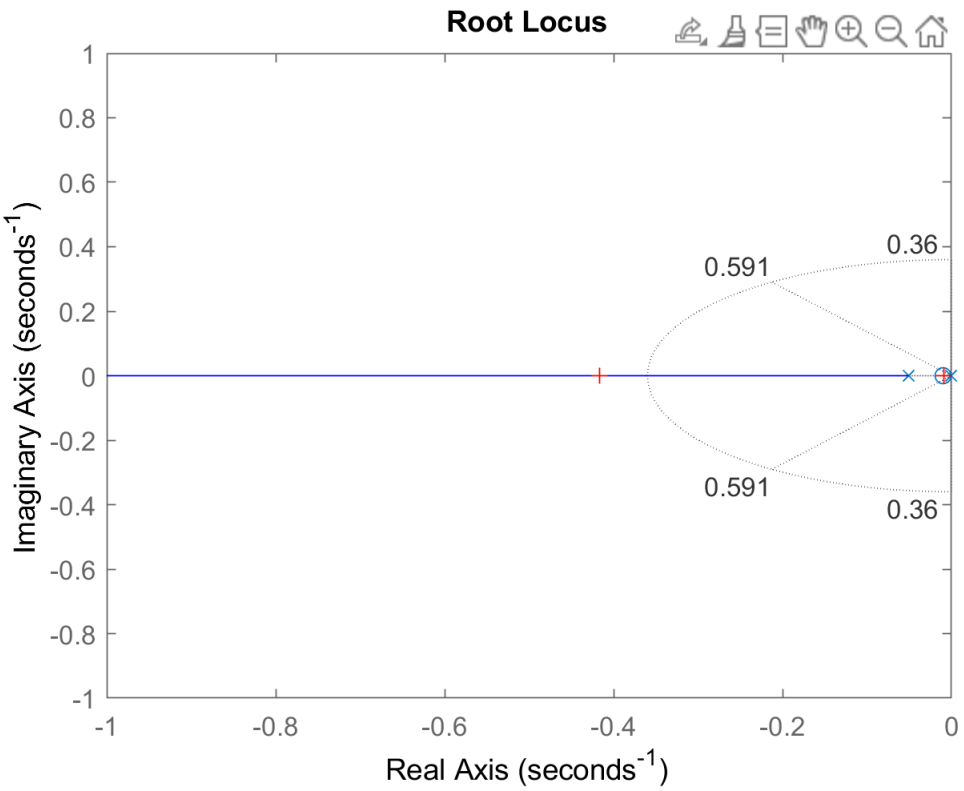
```
ans = struct with fields:
        RiseTime: 1.4137
    SettlingTime: 9.0014
     SettlingMin: 0.9138
     SettlingMax: 1.0927
       Overshoot: 9.7657
      Undershoot: 0
            Peak: 1.0927
        PeakTime: 3.7488
```

5. Active Controller

Why is a passive controller better in this case study?

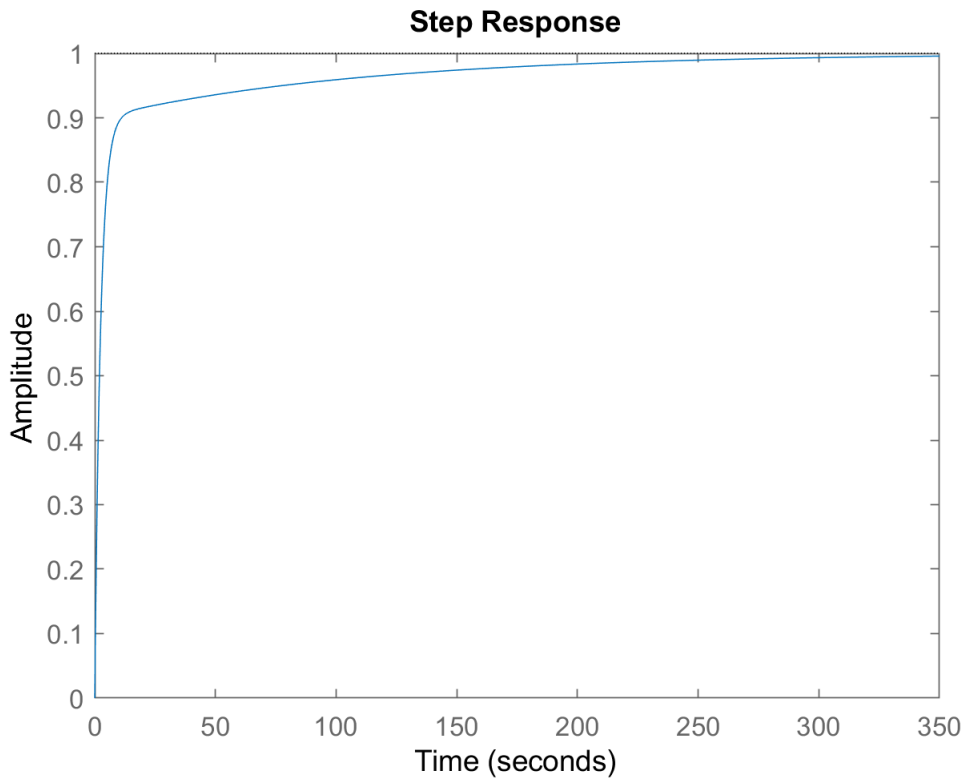
```
zc = 0.01; pc = 0;
Gc = tf([1 zc], [1 pc]);
cls = feedback(G*Gc,1);
rlocus(cls);
axis([-1 0 -1 1]);
sgrid(damping_ratio, wn);

[Kp,poles]=rlocfind(cls)
```

```
Select a point in the graphics window
```




```
selected_point = -0.4166 - 0.0025i
Kp = 374.6122
poles = 2×1
   -0.4166
   -0.0090
```

```
cls = feedback(Kp*G*Gc,1);
step(cls)
```




```
stepinfo(cls)
```

```
ans = struct with fields:
        RiseTime: 10.8450
    SettlingTime: 179.4433
     SettlingMin: 0.9008
     SettlingMax: 0.9959
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9959
        PeakTime: 354.3646
```