EE208 Experiment-1

Dynamic response of transfer functions on MATLAB platform.

Group no. 9

Aryan Bansal – 2020EEB1162

Hiya Kwatra – 2020EEB1173

Ishaan Sethi – 2020EEB1174

Contents

Objective

We shall analyse and discuss the dynamic response of a given linear analog system in terms of different performance measures with MATLAB.

Project Statement

You are given the second order analog OLTF of an oven temperature system is provided, for which a given PD controller is to be considered in (a) Cascade and (b) Feedback.

OLTF: $\frac{K}{s^2+3s+10}$ and PD Controller: $80(s+5)$

Problem 1: In either configuration, the CL root loci are identical. However, the step response always differs. Compare the two responses in terms of standard performance measures to a step input, comment on the results, and explain the reason for the differences in the response.

Problem 2: If the OL gain of the oven temperature OLTF is prone to variation, investigate the effect of this on the performance measures, and explain the reason for the same.

Overview of the analysis

We shall take two approaches to problem 1 – first we shall look at the transfer functions directly, then we shall take our analysis to the time domain by way of partial fraction decomposition.

For problem 2, we shall try to analyse the transfer functions, then we shall investigate the effect on performance measures.

MATLAB Functions Used

The following functions were used:

tf, feedback, step, stepinfo, pzmap, grid, ones, zeros, stepplot, table, length, residue, linspace, logspace

MATLAB Code Setup

Let us first create the OLTF and the PD controller for K=1.

```
clear;
s = tf('s'); %creating an empty TF to be used as a variable.
G = 1/(s^2 + 3*s +10); %creating the OLTF with K=1.
C = 80*(s+5); %creating the PD Controller
```

Since MATLAB cannot handle variables in transfer functions, we shall create an array of K-values.

```
K = [0.01, 0.1, 0.5, 1, 2, 4, 10,];
```

Next, we shall create a model array for the OLTF.

```
system = tf(zeros(1,1,1,length(K)));
for i = 1:length(K)
    system(:,:,:,i) = G*K(i);
end
```

Now, we create the Cascaded and Feedback CLTF arrays.

```
cltf_cascade = feedback(C*system,1);
cltf_fdback = feedback(system,C);
```

We'll also create the CLTFs for K=1 separately for easy analysis.

```
cltf_cascade_1 = feedback(G*C,1);
cltf_fdback_1 = feedback(G,C);
```

We are now ready to perform the step response analyses.

Visualising the Step responses

The step response can be obtained by the step () function.

Step Response for OLTF

```
h = stepplot(G);
grid on
h.showCharacteristic('SteadyState')
h.showCharacteristic('SettlingTime')
h.showCharacteristic('PeakResponse')
h.showCharacteristic('RiseTime')
```

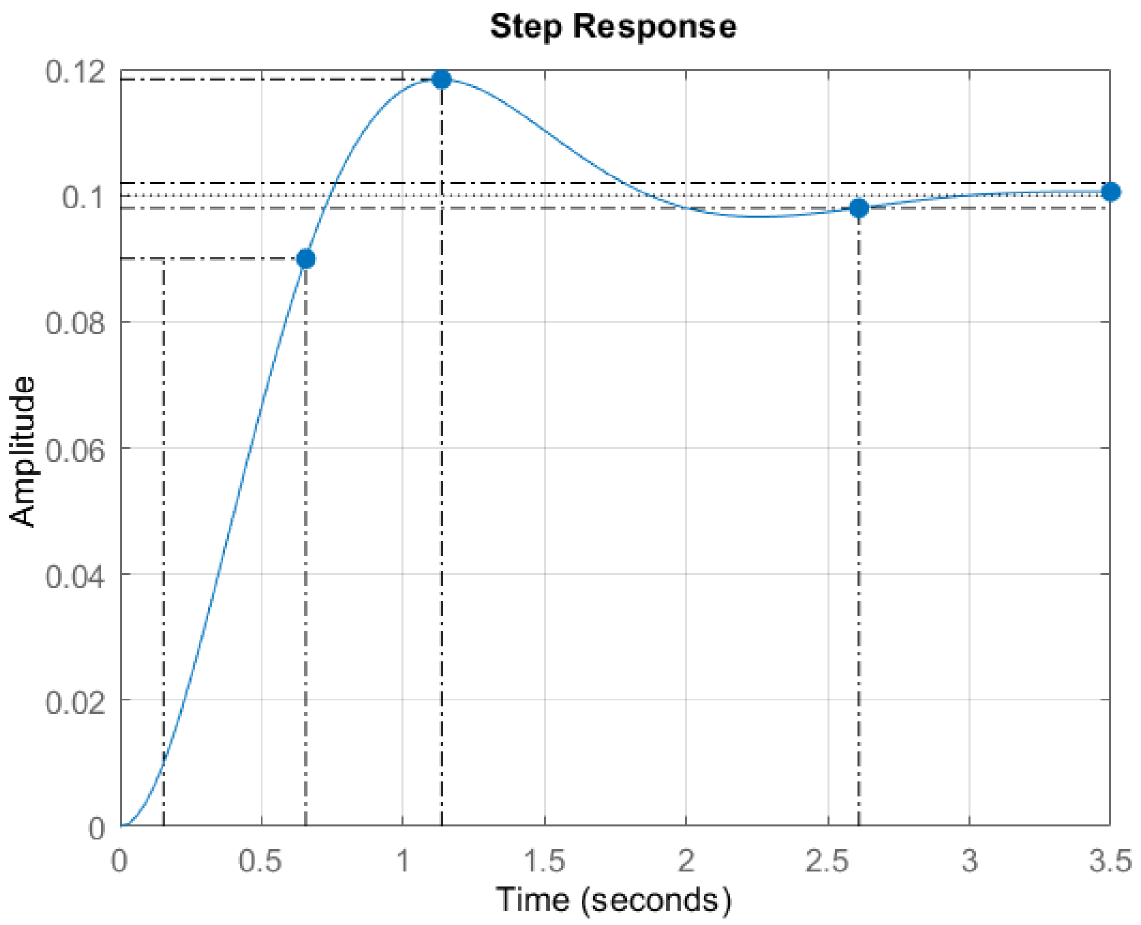

Figure 1 - Step Response of OLTF

Step Response for Cascade CLTF

```
h = stepplot(cltf_cascade_1);
grid on
h.showCharacteristic('SteadyState')
h.showCharacteristic('SettlingTime')
h.showCharacteristic('PeakResponse')
h.showCharacteristic('RiseTime')
```

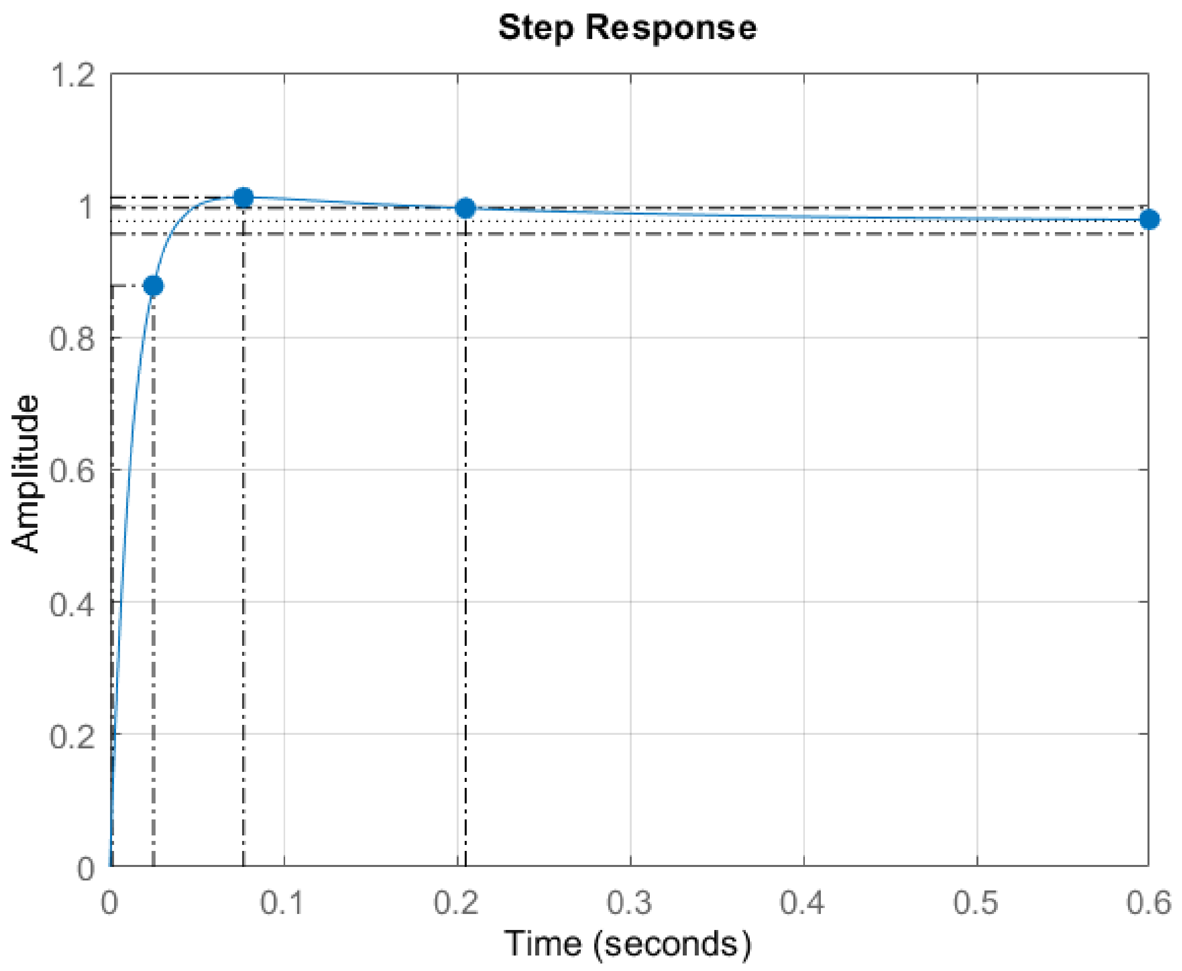

Figure 2 - Step Response of Cascade CLTF

Step Response for Feedback CLTF

```
h = stepplot(cltf_fdback_1);
grid on
h.showCharacteristic('SteadyState')
h.showCharacteristic('SettlingTime')
h.showCharacteristic('PeakResponse')
h.showCharacteristic('RiseTime')
```

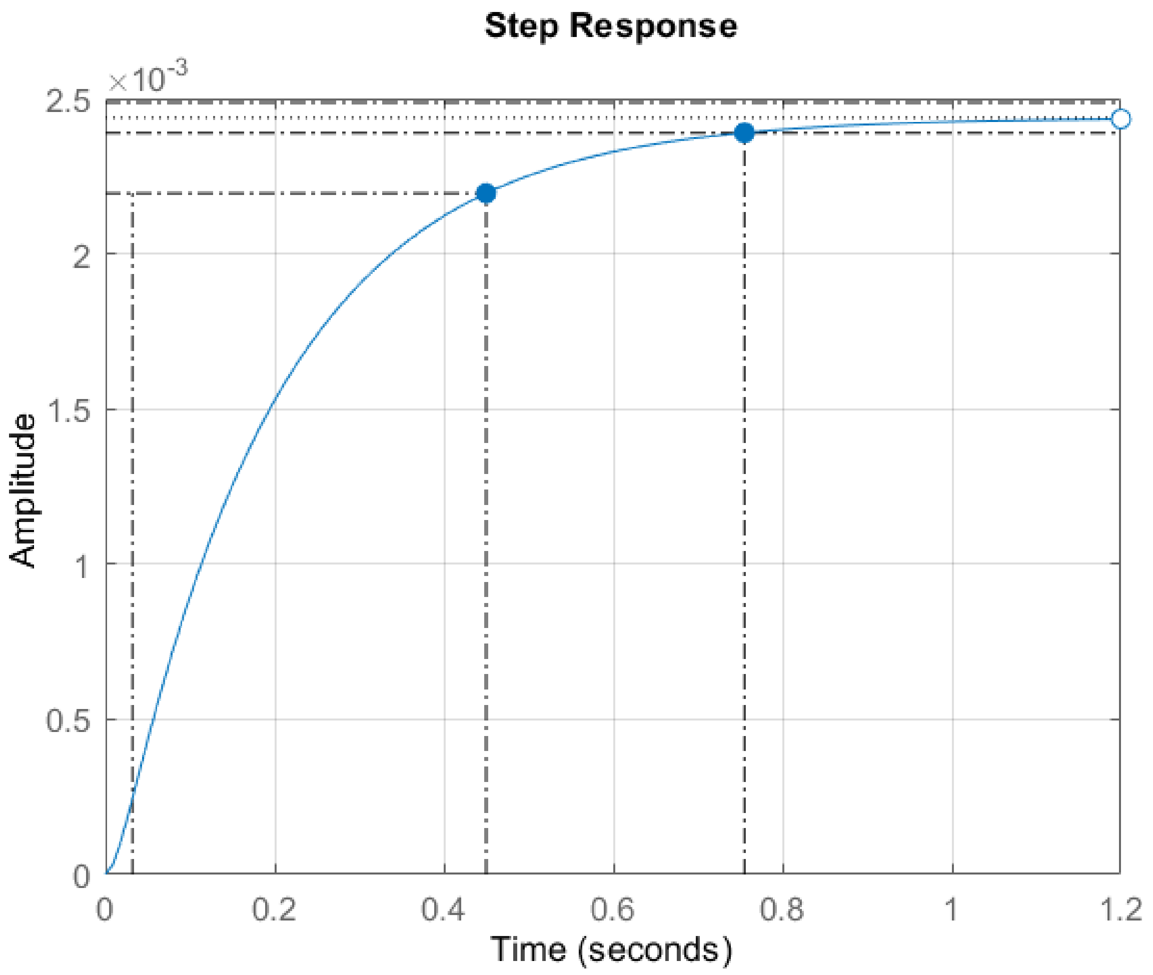

Figure 3 - Step Response of Feedback CLTF

Viewing all the responses together

```
step(G,'r',cltf_cascade_1,'b',cltf_fdback_1,'g',5), grid
```

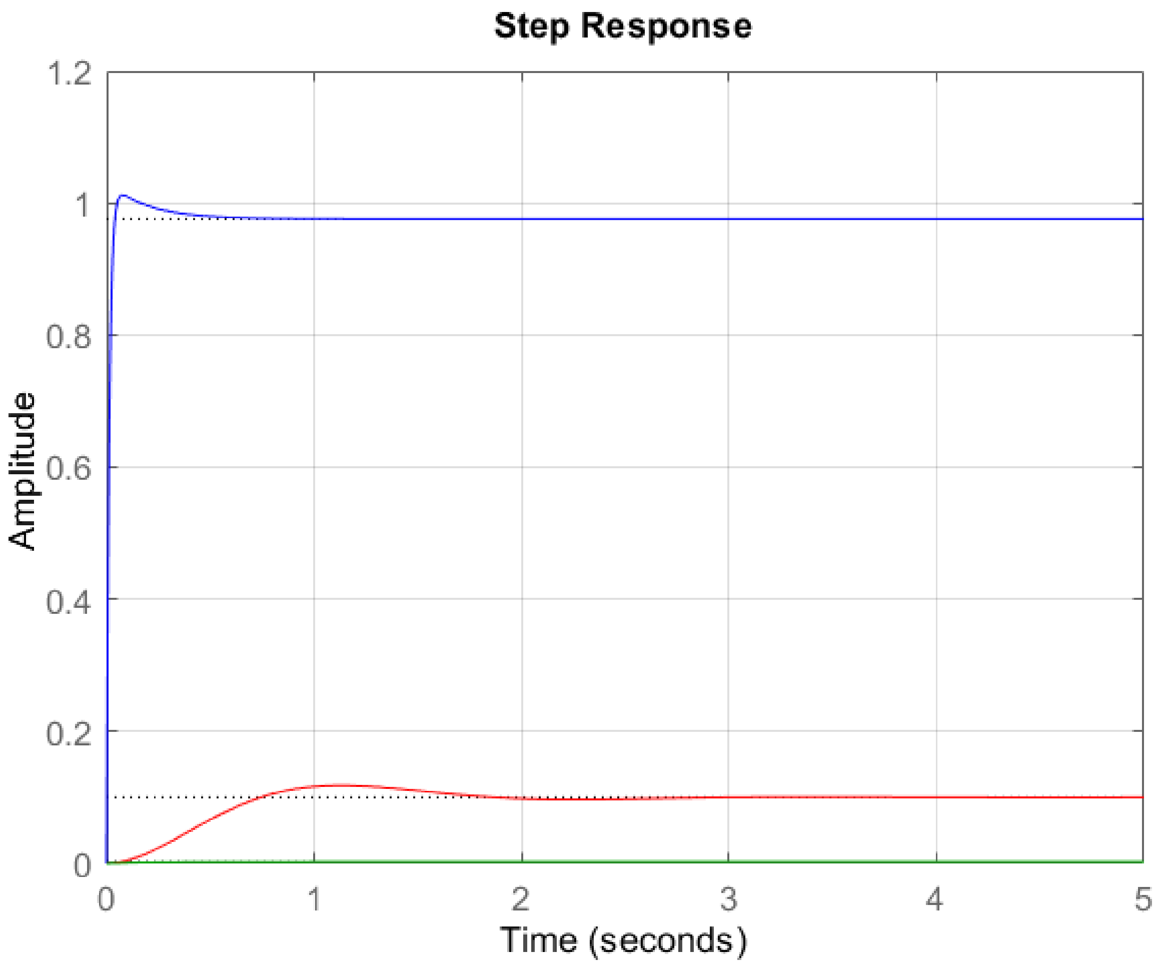

Figure 4 - All three step responses (un-normalized)

The response for cltf_fdback_1 is quite small, so it is seen only as a green line at the bottom. We shall normalize the step responses later, which will aid in both visualization and analysis.

Comparing Performance Measures

```
 Response = ["OLTF";"Cascade CLTF";"Feedback CLTF"];
 SteadyStateError = 1-[0.1;0.976;0.00244]; %Ess
 PeakTime = [1.14;0.077;0]; %Tpeak
 PeakOvershoot = [18.4;3.66;0]; %Mp
 SettlingTime = [2.61;0.205;0.755]; %Tsettling
 RiseTime = [0.503;0.0237;0.418]; %Trise
 Data = table(Response, SteadyStateError, PeakTime, PeakOvershoot, SettlingTime,
RiseTime)
 %Data obtained manually from the step response plots.
```

Data = 3×6 table

Response	SteadyStateError	PeakTime	PeakOvershoot	SettlingTime	RiseTime
"OLTF"	0.9	1.14	18.4	2.61	0.503
"Cascade CLTF"	0.024	0.077	3.66	0.205	0.0237
"Feedback CLTF"	0.99756	0	0	0.755	0.418

We can clearly see that the Cascade CLTF has the fastest response of the three - it rises and settles very swiftly. It has a large, quickly decaying overshoot, which does not oscillate. It also has the smallest steady-state error. The PD Controller seems to be working well for the step input.

The OLTF shows oscillatory dynamics. It rises slowly and has the greatest settling time. It has a large steady-state error, rising only to 0.1 units in the steady-state. We desire to speed these dynamics up.

The Feedback CLTF displays slow, lag type dynamics and rises to a very small steady state value, and thus has the largest steady state error. It appears that this configuration is not useful if our purpose is to closely follow the reference signal. Since it does not decay to zero, it is not even useful for disturbance rejection.

Explaining the Step Responses

Approach 1: Investigating the CLTFs

There is a great difference in the responses of the Cascade CLTF and the Feedback CLTF. To understand the reason for the difference, we must first have a look at the CLTFs themselves.

```
cltf_fdback_1
```

```
cltf_fdback_1 =

        1
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

This is the CLTF for the controller in feedback. It is a simple 2nd order TF, whose poles can be found as follows:

```
pole(cltf_fdback_1)
```

```
ans = 2×1
  -77.7250
   -5.2750
```

Thus, it has poles at -77.725 and -5.275.

Now we look at the CLTF for the controller in cascade.

```
cltf_cascade_1
```

```
cltf_cascade_1 =

     80 s + 400
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

The denominator is the same as before - thus the poles are the same - but this transfer function has a zero at s=-5. Also, it has a different DC gain, which we shall now normalize for ease of analysis.

Let us create new variables. We shall put the transfer functions in standard form, and will also remove the gain of 80 in the cascade CLTF. This will allow us to easily compare the two and will be useful when plotting their responses.

```
H_f = tf( 410, [1,83,410])
```

```
H_f =

       410
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

```
H_c = tf([1/5,1],(1))*H_f
```

```
H_c =

     82 s + 410
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

Now we once again look at the step responses -

```
step(H_f,H_c), grid on
```

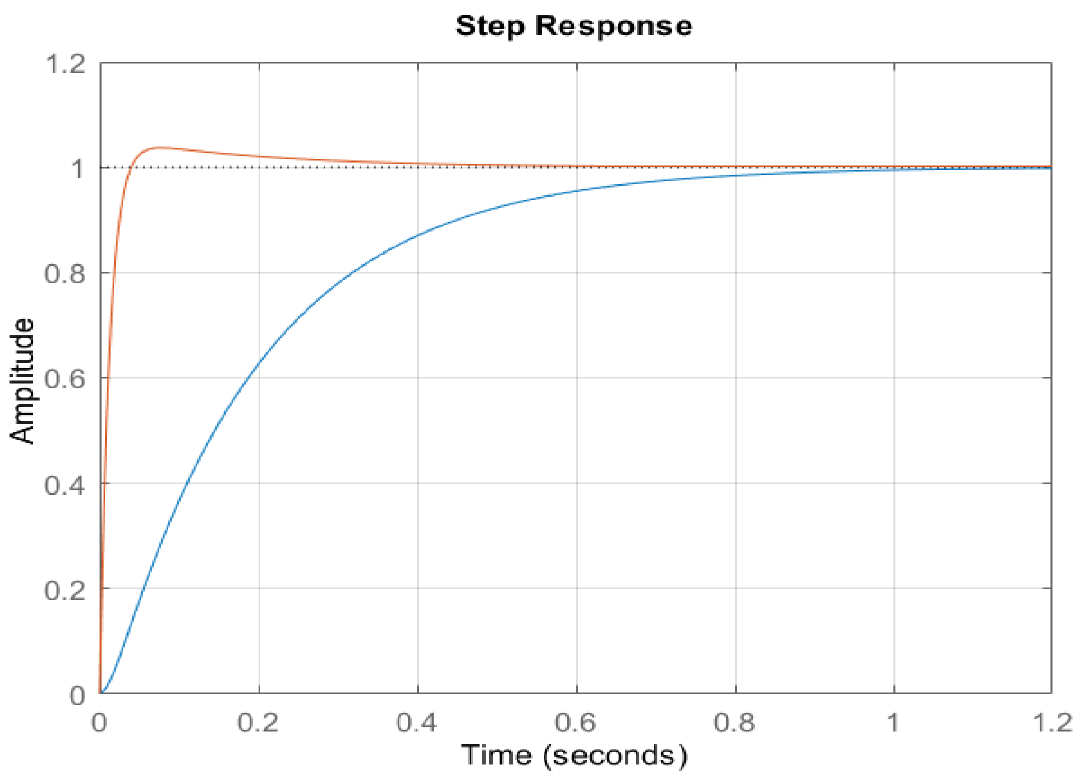

Figure 5 - Normalized step responses of Cascade and Feedback CLTFs

It is clear that the addition of zero “speeds” up the response. It rises and settles faster, and has a much smaller steady-state error.

From the transfer functions:

$$H_c = H_f - \frac{1}{(-5)} * (s \times H_f)$$

We know that s*H_f is simply the derivative of H_f in the time domain.

Hence our time response becomes:

$$\text{Cascade Step Response} = \text{Feedback Response} - \frac{1}{(\text{value of zero})} * \frac{\mathrm{d}}{\mathrm{d}t}\text{Feedback Response}$$

In other words, the effect of adding a zero to the CLTF, is to add a scaled version of the derivative of the original response, to the original step response.

It is this addition of the derivative response that makes the overall response "faster".

Let us plot this.

```
der_H_f = tf([1,0],1) * H_f; %derivative of H_f
step(H_f,der_H_f*0.2,H_f + der_H_f*0.2,H_c,'r--'), grid on
```

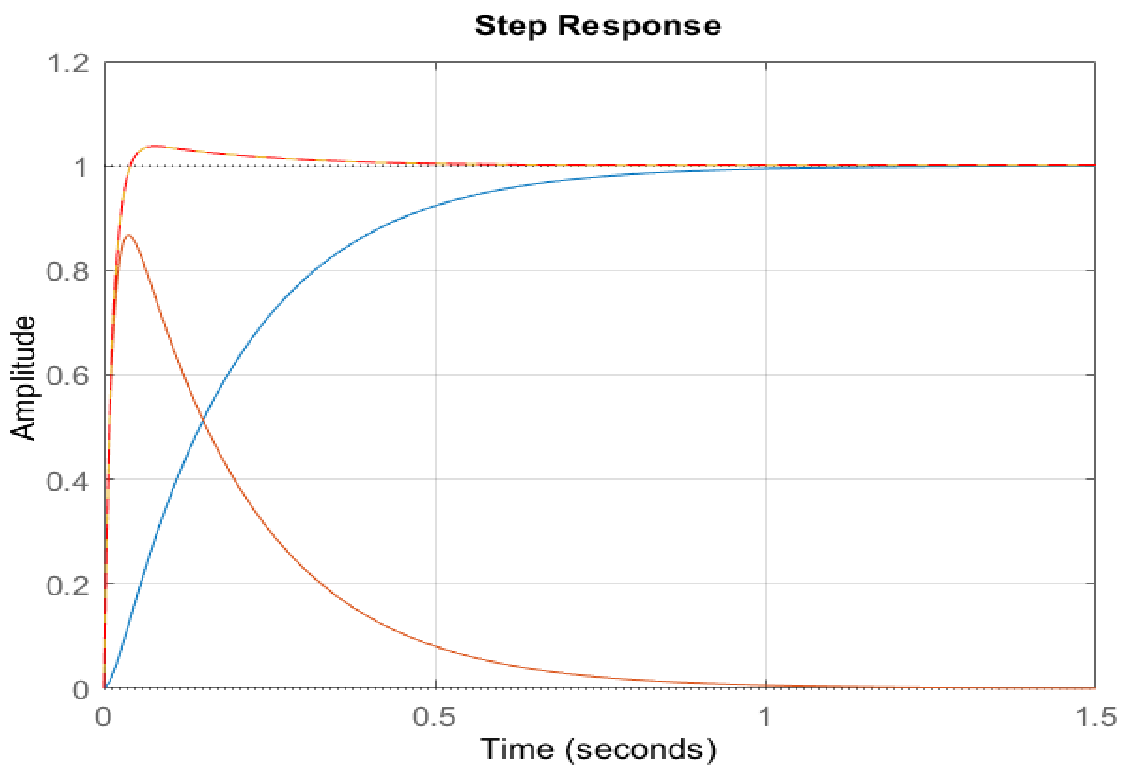

Figure 6 - Addition of a zero as adding the derivative

We see that H_c lines up perfectly with H_f + 0.2*der_H_f.

Thus, we have explained the reason for difference between the step responses.

Let us now look at the unnormalized transfer functions, still with the gain of 80 removed for the sake of visualization - although it makes no difference to the overall dynamics, as we shall see.

```
T_f = tf( 1, [1,83,410])
```

```
T_f =

       1
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

```
T_c = tf([1,5],(1))*T_f
```

```
T_c =

      s + 5
  ----------------
  s^2 + 83 s + 410

Continuous-time transfer function.
```

```
step(T_f,T_c)
```

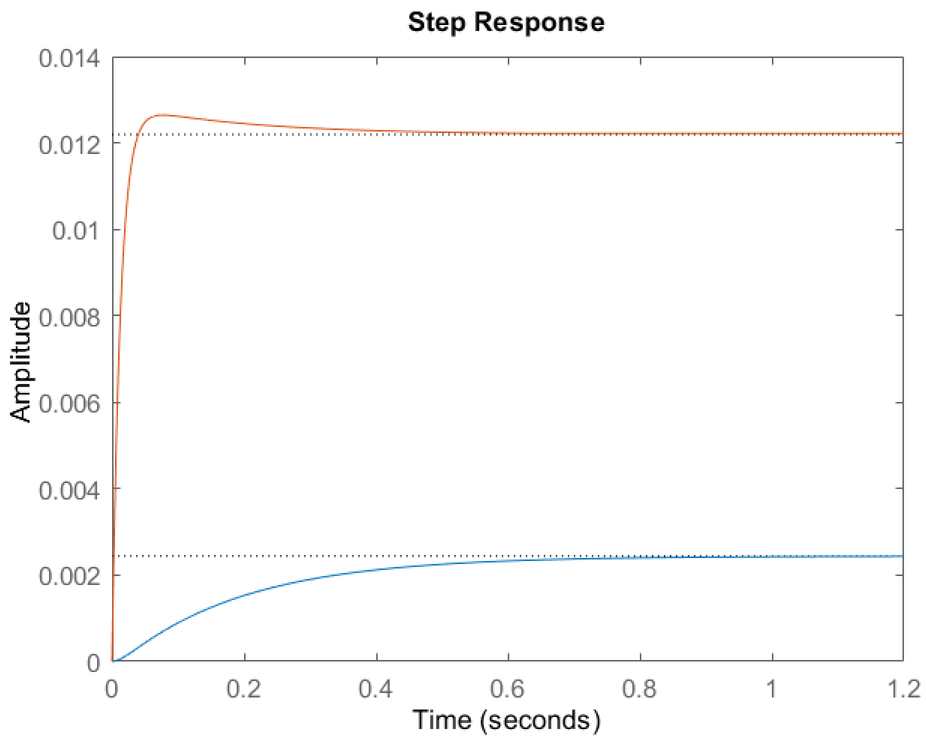

Figure 7 - Unnormalized step responses

```
stepinfo(H_c)
```

```
ans = struct with fields:
        RiseTime: 0.0237
    SettlingTime: 0.2051
     SettlingMin: 0.9054
     SettlingMax: 1.0366
       Overshoot: 3.6644
      Undershoot: 0
            Peak: 1.0366
        PeakTime: 0.0770
```

```
stepinfo(T_c)
```

```
ans = struct with fields:
        RiseTime: 0.0237
    SettlingTime: 0.2051
     SettlingMin: 0.0110
     SettlingMax: 0.0126
       Overshoot: 3.6644
      Undershoot: 0
            Peak: 0.0126
        PeakTime: 0.0770
```

Clearly the only effect is the increase in DC gain - i.e., effect on Peak Value. The overall dynamics remain the same.

Because of this increase in DC gain (the reason for which is obvious if we look at the unnormalized TF), our steady-state error is reduced.

Approach 1: Effect of position of the zero on the Step response

Let us now plot various step responses corresponding to different placements of the zero.

```
Zeros = [1,4,5,10,50,70,100];
sys1 = tf(zeros(1,1,1,length(Zeros)));
for i = 1:length(Zeros)
    sys1(:,:,:,i) = tf([1/Zeros(i),1],(1))*H_f;
end
step(H_f, 'r', H_f*tf([1/5.275,1],(1)), 'r--', H_f*tf([1/77.725,1],1), 'r--',
sys1), grid on
```

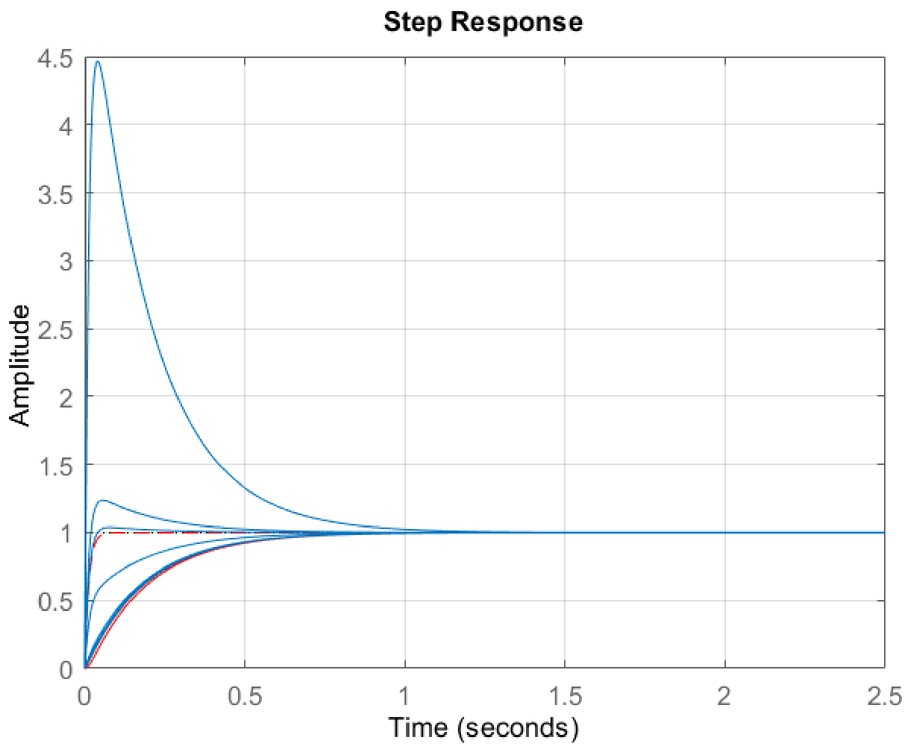

Figure 8 - Step Responses for various Zero Placements

There are many things to observe:

- The overshoot tends to infinity as we approach the origin, this is because in our scaling we have divided by the zero.

- For zeros smaller than the first pole, we have an early overshoot that decays quickly. In this region, the dynamics of the larger pole dominate since those of the smaller pole are, to an extent, dulled out by the effect of the zero.

- As we reach Pole 1, there is cancellation, and so we obtain a 1st order TF. The step response of this TF is that of exponential growth, with a small time constant befitting the large magnitude of Pole 2 (-77.725).

- Between Poles 1 and 2 - there are no overshoots now, and the dynamics are lag type, with characteristics in-between those at Pole 1 and Pole 2.

- Once again, when we approach Pole 2, cancellation occurs and we are once again left with the step response corresponding to the 1st order low-pass TF at frequency 5 rad/s. Since it was the high-magnitude pole which was cancelled this time, the dynamics are slow.

- Beyond Pole 2, we begin to approach the original, zero-less curve, which corresponds to a zero at infinity.

Let us also look at the denormalized responses to observe the steady state values.

```
sys2 = tf(zeros(1,1,length(Zeros)));
for i = 1:length(Zeros)
    sys2(:,:,:,i) = tf([1,Zeros(i)],(1))*H_f/410;
end
step(H_f/410, 'b', H_f*tf([1,5.275],(1))/410, 'r--', H_f*tf([1,77.725],1)/410,
'r--', sys2), grid on
```

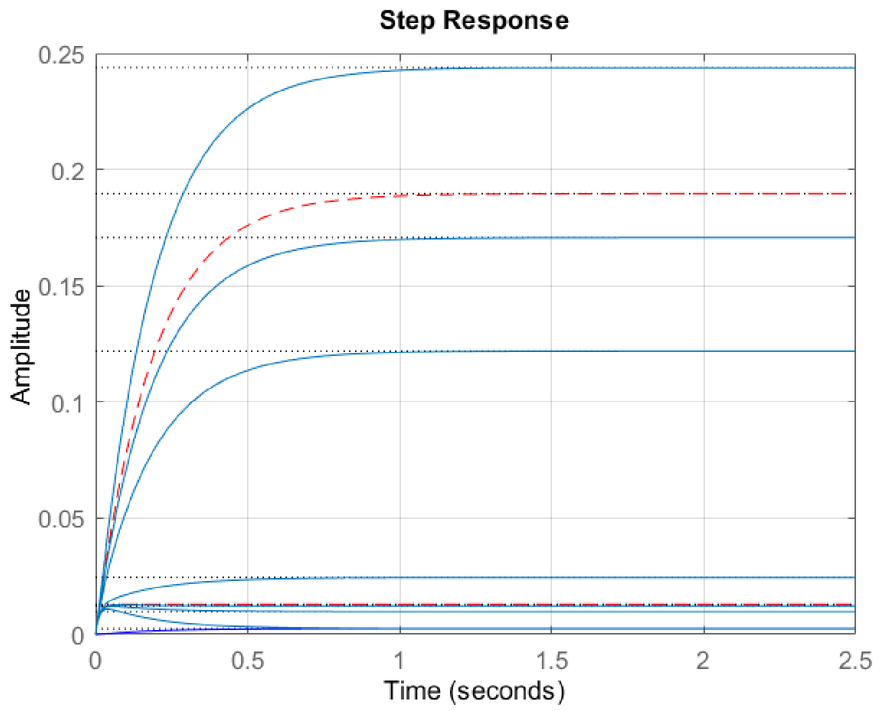

Figure 9 - Unnormalized step responses at various zero placements

The dynamics are the same as before, now we can see how the larger zeros cause large DC gains, and thus larger steady state values.

Approach 1: Effect of Zero Placement on Performance measures

```
Zeros = linspace(1,90,70);
sys1 = tf(zeros(1,1,1,length(Zeros)));
for i = 1:length(Zeros)
    sys1(:,:,:,i) = tf([1/Zeros(i),1],(1))*H_f;
end

x = zeros(1,length(Zeros));
for i=1:length(Zeros)
    x(:,i) = stepinfo(sys1(:,:,:,i)).RiseTime;
end
plot(x), grid on
```

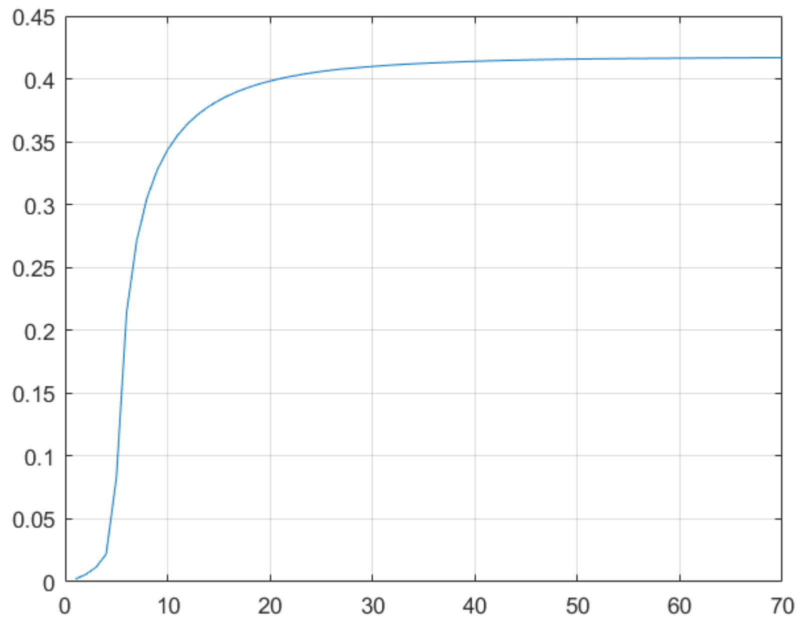

Figure 10 - Plot of Rise Time v/s Zero placement

Initially we have fast dynamics, that die out as we pass Pole 1. Our RiseTime quickly increases, then slows down beyond z=-10. We approach the RiseTime of H_f as we move our zero towards -infinity.

```
x2 = zeros(1, length(Zeros));
for i=1:length(Zeros)
    x2(:,i) = stepinfo(sys1(:,:,:,i)).Overshoot;
end
plot(Zeros,x2), grid on
```

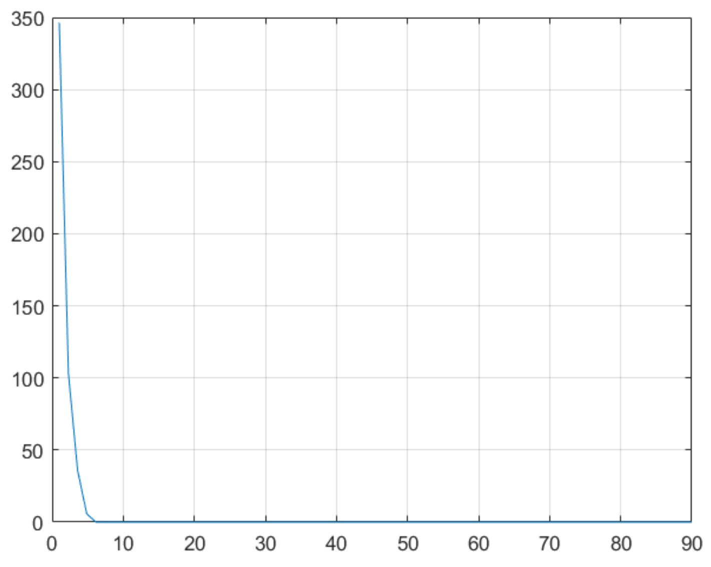

Figure 11 - Overshoot% vs Zero placement

The initial zeros cause a large overshoot, which suddenly dies out after we cross Pole 1.

After Pole 1 is crossed, our dynamics become lag type, and the effect of zeros begins to attenuate somewhat.

```
x3 = zeros(1,length(Zeros));
for i=1:length(Zeros)
    x3(:,i) = stepinfo(sys1(:,:,:,i)).SettlingTime;
end
plot(Zeros,x3), grid on
```

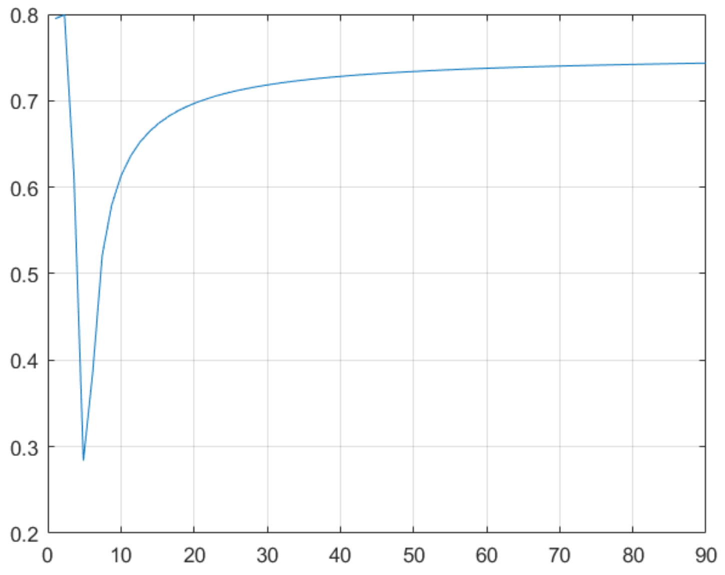

Figure 12 - Settling Time vs Zero placement

Settling time quickly decreases till we reach Pole 1 - only Pole 2's dynamics exist here, so this is the position of least settling time. Beyond Pole 1, settling time keeps increasing since the slow dynamics of Pole 1 aren't cancelled as easily by the zero since it moves away from it.

Approach 2: Partial Fraction Decomposition and Dominant Dynamics

We shall use the residue-pole-constant form of the transfer function to see how the zero directly effects the step response by way of its effect on the exponential terms, and thus see how the dominant dynamics are changed.

```
 [r1,p1,k1] = residue(1,[1,83,410,0]); %this command computes residues for the
PFD
 [r2,p2,k2] = residue([80,400],[1,83,410,0]);
```

First for the residues and poles of cltf_fdback_1*(1/s). 1/s is the TF for a step input.

```
 r1
```

```
r1 = 3×1
    0.0002
   -0.0026
    0.0024
```

```
 p1
```

```
p1 = 3×1
  -77.7250
   -5.2750
         0
```

The decomposed transfer function is thus $\frac{0.0002}{s+77.725}-\frac{0.0026}{s+5.275}+\frac{0.0024}{s}$.

In the time domain, this will be $0.0002\cdot e^{-77.725\cdot t}-0.0026\cdot e^{-5.275\cdot t}+0.0024$.

Now we look at the residues and poles of cltf_cascade_1*(1/s)

```
 r2
```

```
r2 = 3×1
   -1.0332
    0.0576
    0.9756
```

```
 p2
```

```
p2 = 3×1
  -77.7250
   -5.2750
         0
```

The TF is now $\frac{-1.0332}{s+77.725}+\frac{0.0576}{s+5.275}+\frac{0.9756}{s}$, which in the time domain will be $-1.0332\cdot e^{-77.725\cdot t}+0.0576\cdot e^{-5.275\cdot t}+0.9756$.

Let us plot the time domain equations to see how the relative contribution of each mode is changed.

```
t = linspace(0,1,1000);
y1 = r1(1)*exp(p1(1)*t) + r1(2)*exp(p1(2)*t) + r1(3)*exp(p1(3)*t);
y2 = r2(1)*exp(p2(1)*t) + r2(2)*exp(p2(2)*t) + r2(3)*exp(p2(3)*t);

plot(t,y1,t,r1(1)*exp(p1(1)*t),t,r1(2)*exp(p1(2)*t),'r--
',t,r1(3)*exp(p1(3)*t)),grid on
```

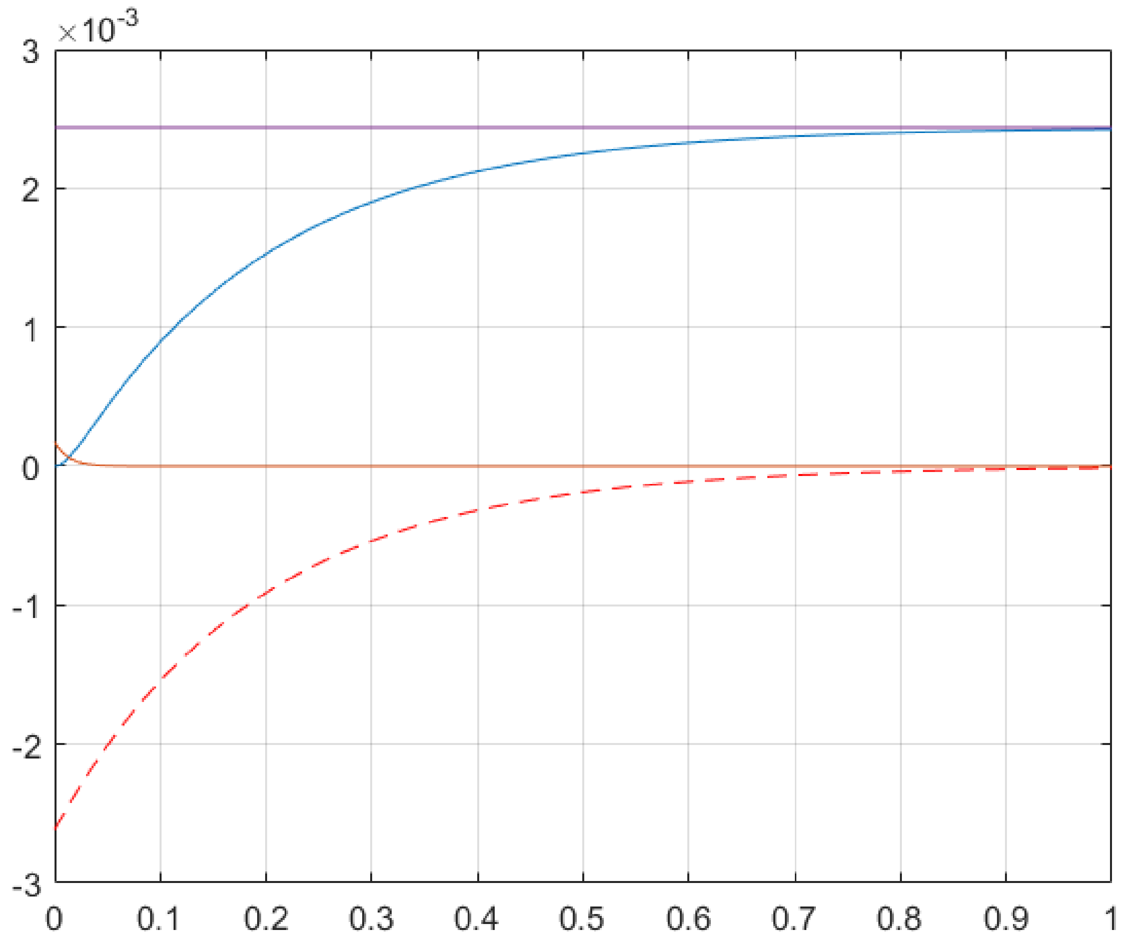

Figure 13 - Decomposing the step response of Feedback CLTF

Red = Dynamics of pole at -5.275
Orange = Dynamics of pole at -77.725
Purple = Dynamics of pole at 0, in other words, the forced response

The dynamics that are dominant are those of the pole near -5.

```
plot(t,y2,t,r2(1)*exp(p2(1)*t),t,r2(2)*exp(p2(2)*t),'r--
',t,r2(3)*exp(p2(3)*t)),grid on
```

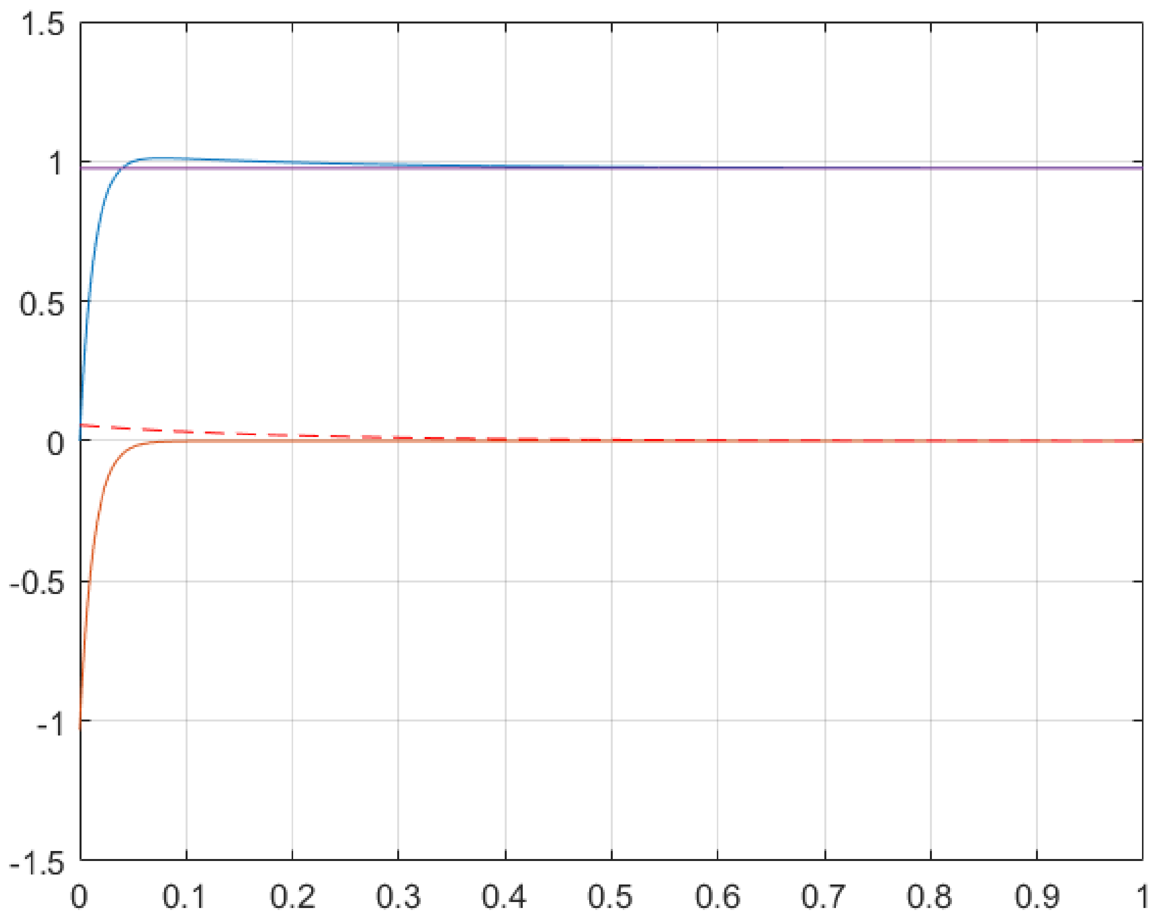

Figure 14 - Decomposing the dynamics of the Cascade CLTF

Red = Dynamics of pole at -5.275
Orange = Dynamics of pole at -77.725
Purple = Dynamics of pole at 0, in other words, the forced response

From these plots we can clearly see the contribution of each term. Purple is the forced response - a constant curve. Red is the response of the pole at -5.275, and yellow is the response of the pole at -77.725.

Because of the addition of the zero, the dominant response is that of Pole 2 (-77.725). The zero cancels the effects of Pole 1 to a large extent.

Summary

We identified the reason for the difference between step response to be the presence of a zero in the cascade CLTF. To analyse the precise effect of the zero, we followed two approaches to the problem.

First, we looked at the transfer functions directly (after some normalization for ease of visualization, which does not change our findings), and saw how the addition of a zero changed them. We can summarise the findings of the first approach as follows –

$$\boxed{\text{Cascade Step Response} = \text{Feedback Response} - \frac{1}{(\text{value of zero})} * \frac{d}{dt}\text{Feedback Response}}$$

We also saw how changing the position of the zero changes the overall response.

For the second approach, we simplified the transfer function by way of Partial Fraction Decomposition. In doing so we obtained the residues belonging to each pole. We then plotted the dynamics of each pole in the time domain, and observed how the zero changes the relative contribution of each eigenvalue depending on its location.

We can write the findings of Approach 2 as follows -

1. The PD Controller connected in Cascade adds a zero at s=-5 and an additional gain of 80 units.
2. The zero at -5 reduces the contribution of the pole at -5.275 by a large extent.
3. Hence the dominant dynamics are those of the large pole at -77.725.
4. This pole's dynamics are fast – they have a small time constant.
5. Thus, the dynamics of the Cascade CLTF are "sped up".
6. In comparison, we saw that the dominant dynamics for the Feedback CLTF are due to the pole at -5.275, which are relatively slower.

Both the approaches gave useful results, with Approach 2 showing us the exact effect of the zero on the time domain response.

The effect of variable OLTF gain "K"

Visualization

Let us now look at the effect of variable "K" on the dynamics.

```
step(system), grid on
```

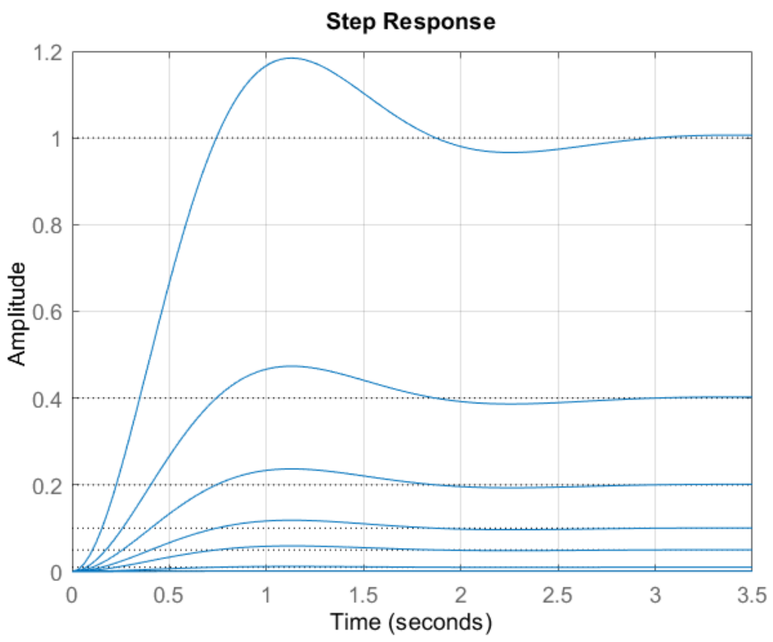

Figure 15 - Varying K, effect on OLTF response

For the OLTF, variable K only changes DC gain and steady-state error, and has no effect on the dynamics as such.

```
step(cltf_fdback,cltf_fdback_1,'r'), grid on
```

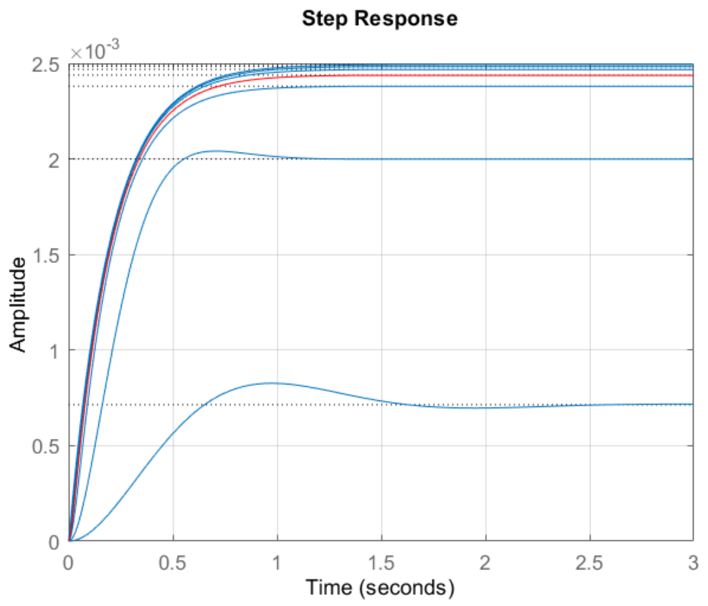

Figure 16 - Varying K, effect on feedback CLTF response

```
step(cltf_cascade,cltf_cascade_1,'r'), grid on
```

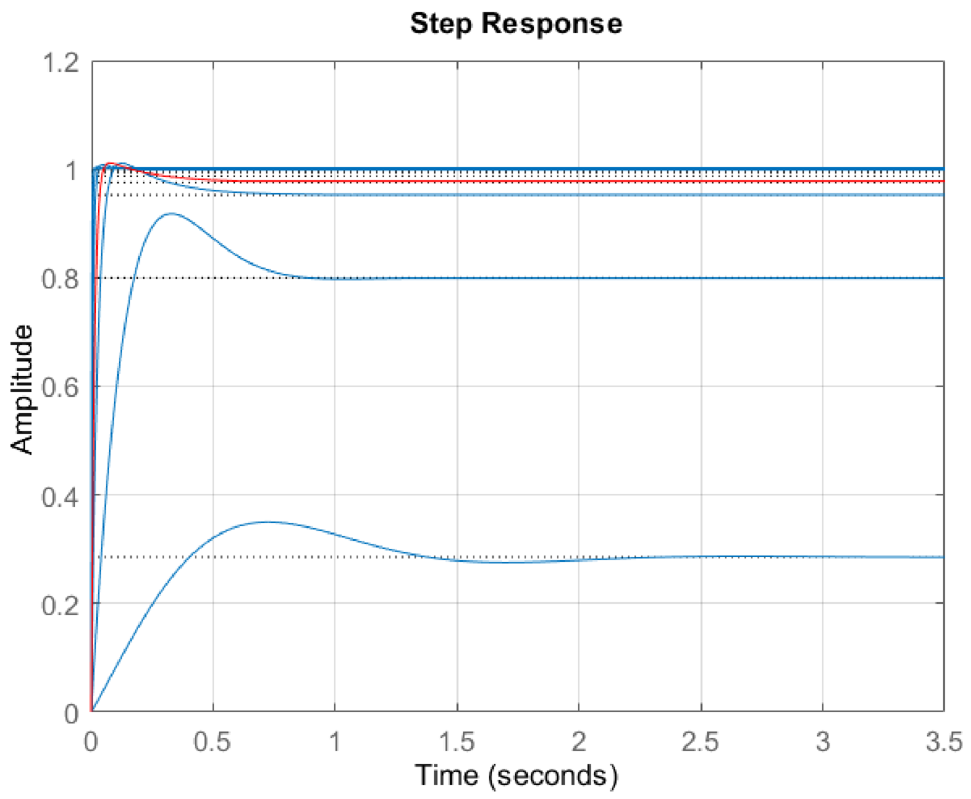

Figure 17 - Varying K, effect on cascade CLTF response

Clearly, a variable K can greatly change the dynamics of the CLTF.

What happens when we “vibrate" K around a certain value, say K=1?

```
K2 = linspace(0.8,1.2,20);
system2 = tf(zeros(1,1,1,length(K2)));
for i = 1:length(K2)
    system2(:,:,:,i) = G*K2(i);
end
cltf_cascade2 = feedback(C*system2,1);

step(cltf_cascade2,cltf_cascade_1,'r'), grid on
```

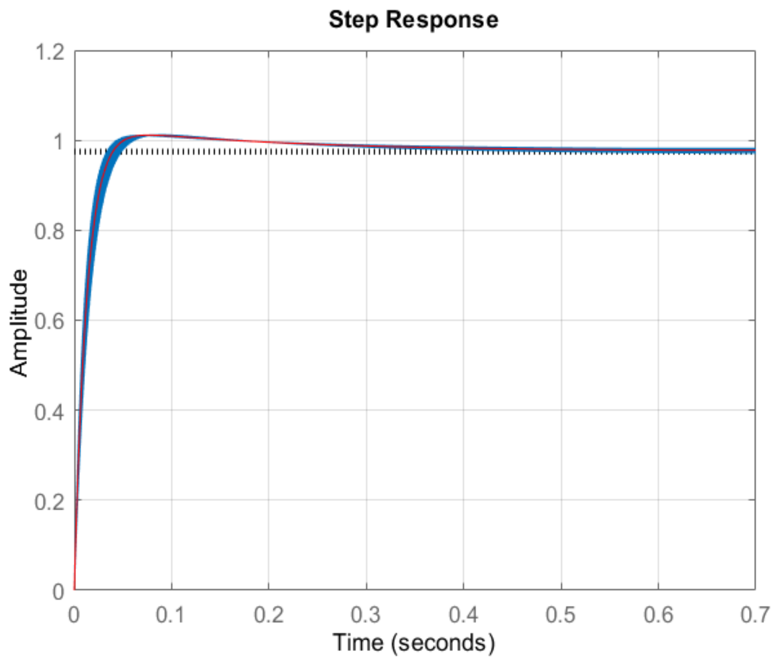

Figure 18 - perturbing K around K=1

It looks like the dynamics don't change greatly around K=1. Let us look around K=0.01.

```
K3 = linspace(0.008,0.012,20);
system3 = tf(zeros(1,1,1,length(K3)));
for i = 1:length(K3)
    system3(:,:,:,i) = G*K3(i);
end

cltf_cascade3 = feedback(C*system3,1);
step(cltf_cascade3), grid on
```

Around K=0.01, the effect of changing K is much more prominent, although the response still remains oscillatory.

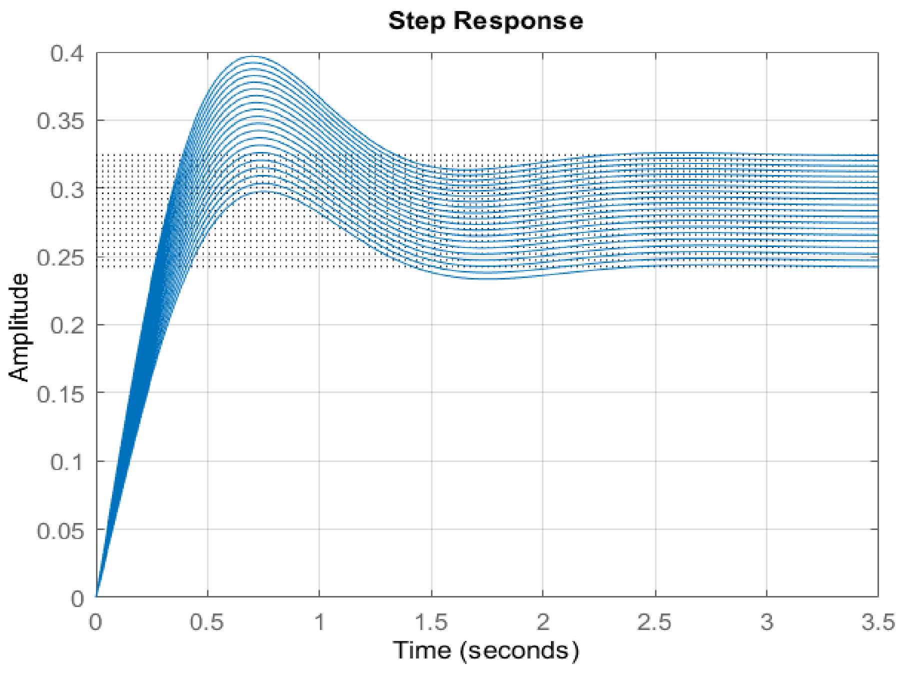

Figure 19 - changing K around K=0.01

Poles of the CLTF

To understand *how* K affects the dynamics, we shall look at the transfer functions.

Let us look at cltf_cascade's pole-zero map, so we can observe at the zero too -

```
pzmap(cltf_cascade)
```

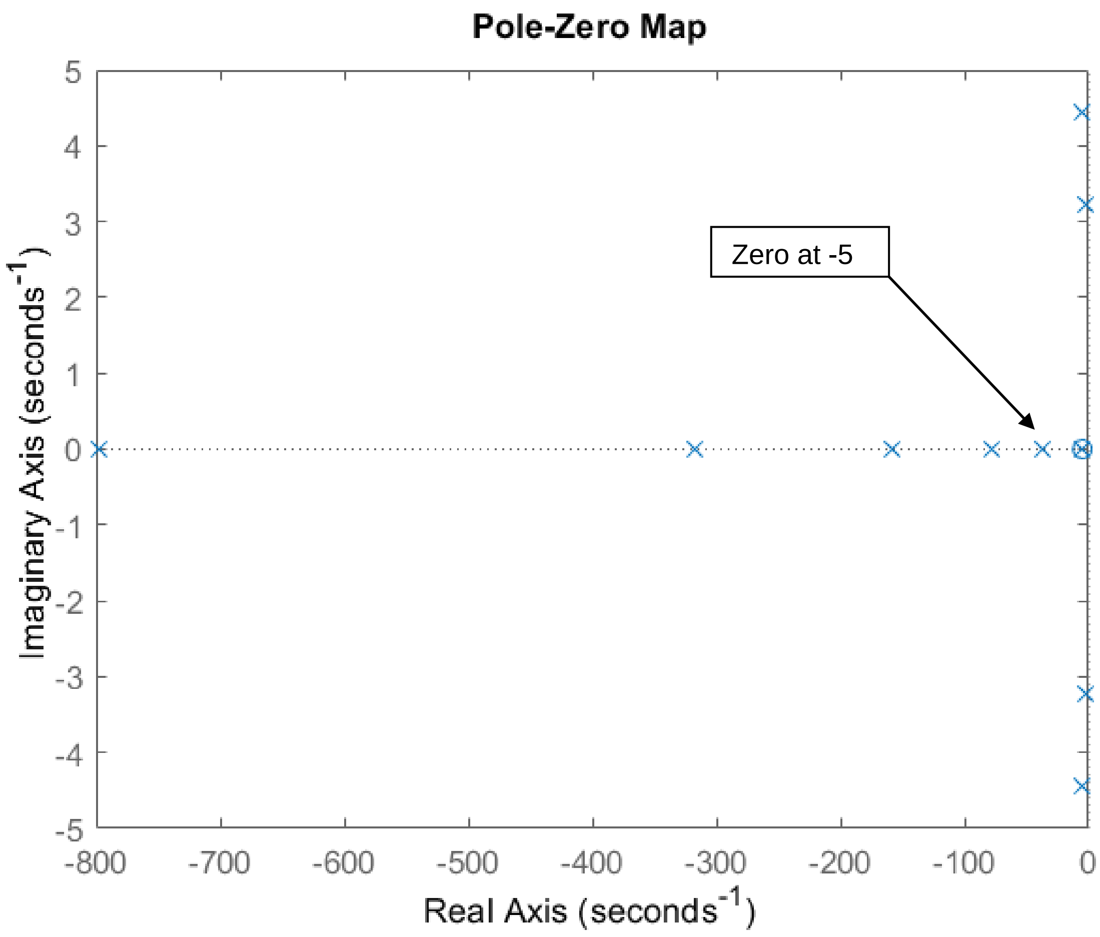

Figure 20 - pole zero map for cascade CLTFs

We can see that the zero remains unaffected by variable K, while the poles are changed - this is what causes the change in dynamics. This is, in effect, what is called the "Root Locus" - that explains how K changes the poles of the system.

We can also see that the poles are going towards the zero at -5, and towards infinity.

Hence the dynamics of one pole are dulled, allowing the larger pole to dominate.

As K goes to infinity, one pole goes towards infinity, and the other is cancelled by the zero at -5.

Thus, varying K changes the Closed Loop Poles, greatly affecting the dynamics of the system.

Effect of variable K on Cascade Controller Performance

Let us see how the performance measures are affected for the cascade CLTF.

```
K4 = logspace(-2,2,15);
system4 = tf(zeros(1,1,1,length(K4)));
for i = 1:length(K4)
    system4(:,:,:,i) = G*K4(i);
end
cltf_cascade4 = feedback(C*system4,1);
step(cltf_cascade4)
```

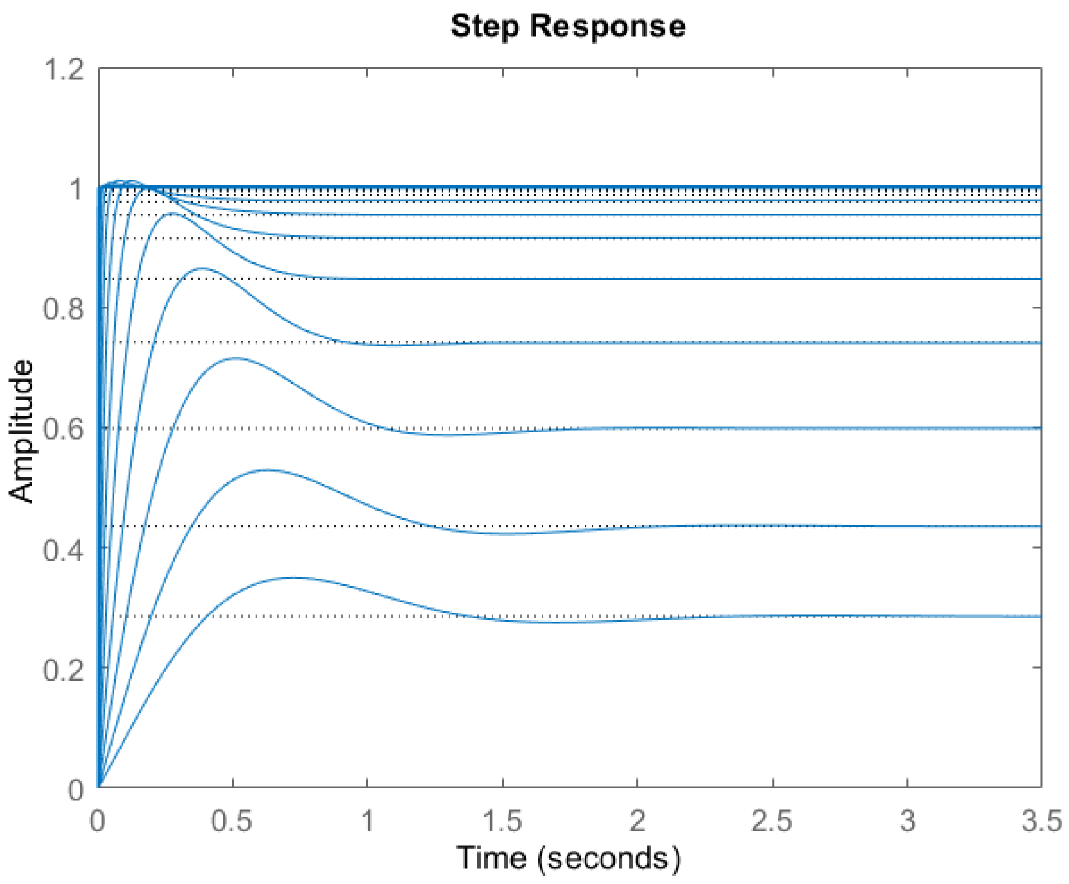

Figure 21 - step response of cascade CLTF for variable K

Steady-state error decreases as we increase K. This is because we are also multiplying the CLTF by K in the numerator.

```
x4 = zeros(1,length(K4));
for i=1:length(K4)
    x4(:,i) = stepinfo(cltf_cascade4(:,:,:,i)).RiseTime;
end
plot(K4,x4), grid on
```

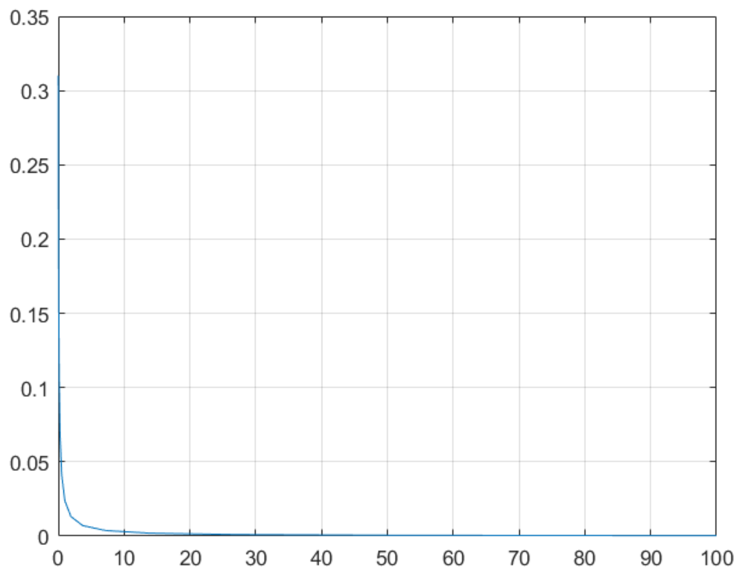

Figure 22 - changing Rise Time

```
x4 = zeros(1,length(K4));
for i=1:length(K4)
    x4(:,i) = stepinfo(cltf_cascade4(:,:,:,i)).Overshoot;
end
plot(K4,x4), grid on
```

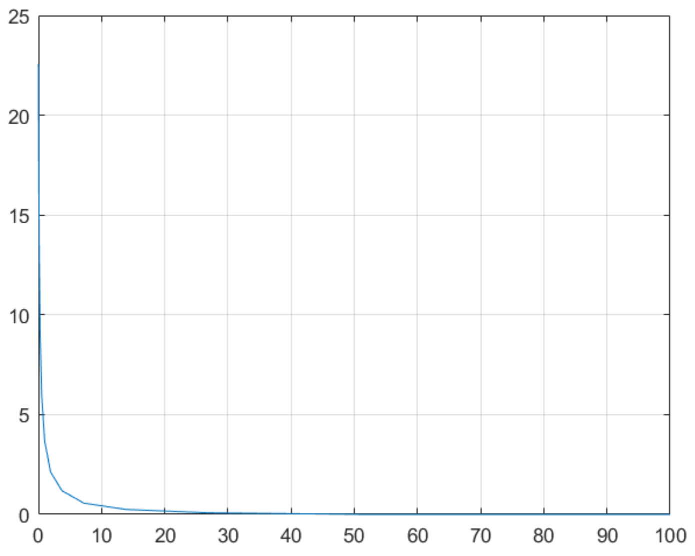

Figure 23 - Changing Peak Overshoot

```
x4 = zeros(1,length(K4));
for i=1:length(K4)
    x4(:,i) = stepinfo(cltf_cascade4(:,:,:,i)).PeakTime;
end
plot(K4,x4), grid on
```

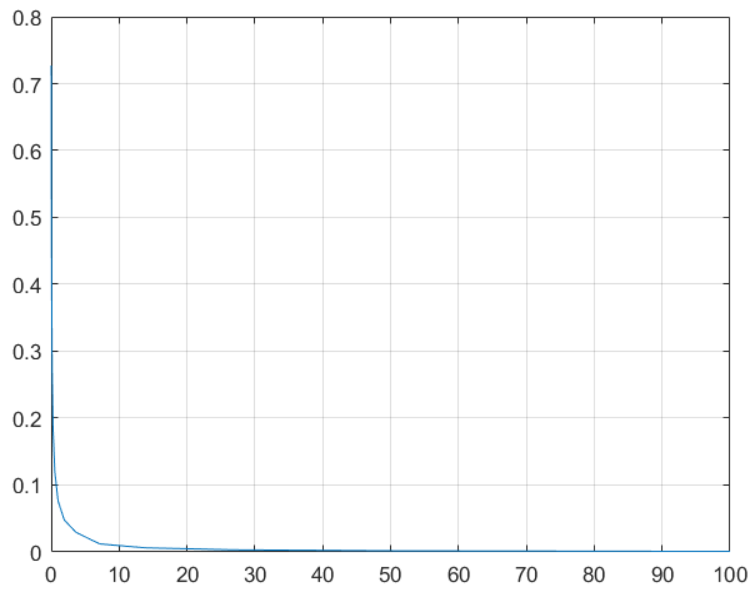

Figure 24 - Changing Peak Time

```
x4 = zeros(1,length(K4));
for i=1:length(K4)
    x4(:,i) = stepinfo(cltf_cascade4(:,:,:,i)).SettlingTime;
end
plot(K4,x4), grid on
```

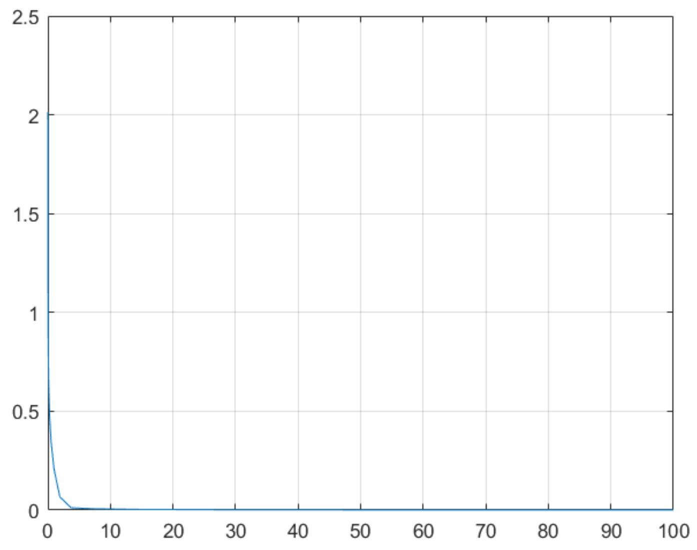

Figure 25 - Changing Settling Time

We see that all four performance measures quickly decrease as we increase K - this can be explained by observing that, as we increase K, one pole moves towards the zero, and the other towards infinity. The dynamics due to the far-away pole begin to dominate, as the pole near the zero has its dynamics dulled by the zero. So, we observe "faster" dynamics as expected from a far-away pole.

Comparing Performance between Feedback and Cascaded CLTF

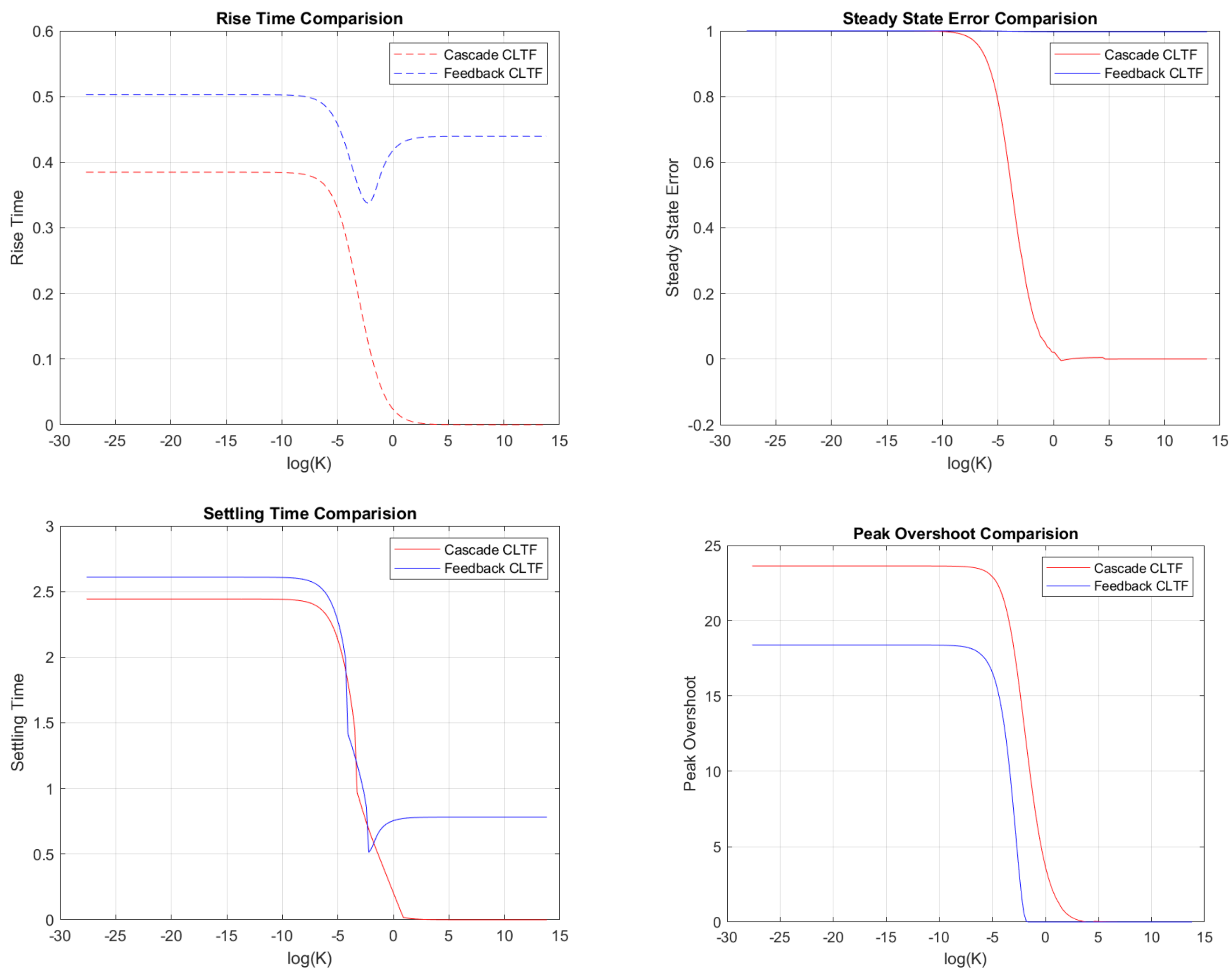

Figure 26 – Performance Comparision

From observing these graphs, we can clearly see that the Cascade CLTF performs far better than the Feedback CLTF over a large range of OLTF gains.

Summary

We visualized the effect of varying OLTF gain K on the step responses of the given transfer functions. We showed that this was because of K changing the CL poles of the system, giving rise to different dynamics. We investigated the effect of changing K on the performance measures of the Cascade CLTF. We have also compared the performance of the Cascade CLTF and the Feedback CLTF over a varying K, and come to the conclusion that the Cascade CLTF performs better.

Conclusions

In this experiment we observed the difference in step responses of a PD controller connected in cascade vs in feedback.

We explained the reason for this difference by two approaches, which have been summarized earlier in the report.

We also investigated the effect of varying OLTF gain on the Closed Loop Step Responses, and showed the reason for the same.

Finally, we compared the performance of the Cascade and Feedback CLTFs over a wide range of OLTF gains.

Based on the findings in our report, we conclude that the PD Controller for the Oven Temperature Control System performs much better for step inputs when it is connected in Cascade with the system, as we have demonstrated with the performance measurement graphs in the report.

References

- Automatic Control Systems, 9th Ed (Golnaraghi and Kuo) – 5.10, 5.11
- Feedback Control of Dynamic Systems, 8th Ed (Franklin, Powell, Emami-Naeni) – 3.4, 3.5
- MATLAB Documentation on the Mathworks website for the relevant MATLAB Functions

Matlab live scipt available at :

https://drive.google.com/file/d/1AEmkIhkZPi5-OZy3Tf10ArhvmUwYdsnI/view?usp=sharing