

---

# EE208:CONTROL ENGINEERING LAB 09

**Simulation and control design on Simulink as a CAD  
interface to MATLAB**

**Course Instructor : Dr. Sanjay Roy**

**Date : 21<sup>th</sup> March 2022**



**Group Number - 14**

<b>Raj Kumar Deb</b>	<b>2020eeb1025</b>
<b>Tushar Sharma</b>	<b>2020eeb1325</b>
<b>Tiya Jain</b>	<b>2020eeb1213</b>

---

## TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>2</b>
<b>1. OBJECTIVE</b>	<b>3</b>
<b>2. GIVEN INFORMATION</b>	<b>3</b>
<b>3. INTRODUCTION : Ziegler-Nichols Rules</b>	<b>3</b>
<b>4. ZIEGLER - NICHOLS TUNING METHOD</b>	<b>5</b>
4.1. Simulink Diagrams	5
4.1.1 P controller for ( $k=1, 2, 3, \dots, 10$ )	5
4.1.2 PI controller for ( $k=1, 2, 3, \dots, 10$ )	6
4.1.3 PID controller for ( $k=1, 2, 3, \dots, 10$ )	6
4.2. Tuned constants for P, PI, and PID control	7
4.2.1 P controller for ( $k=1, 2, 3, \dots, 10$ )	7
4.2.2 PI controller for ( $k=1, 2, 3, \dots, 10$ )	7
4.2.3 PID controller for ( $k=1, 2, 3, \dots, 10$ )	7
<b>5. OBSERVATIONS &amp; THEIR ANALYSIS</b>	<b>8</b>
5.1 Inflection Points	8
5.2 The anomalous case of $k = 1$	8
5.3 Dynamic responses of CLTF curves and Performance Measures	9
5.2.1 P controller	9
5.2.2 PI controller	10
5.2.3 PID controller	10
5.2.4 Performance Comparison Of P, PI and PID Controllers :	10
5.3 Analysis of the effectiveness of the “ZN rules” for multiple poles systems	11
5.4 Overall effectiveness of Z-N Rules:	12
<b>6. CONCLUSION</b>	<b>13</b>
<b>7. MATLAB SCRIPT</b>	<b>13</b>

---

## 1. OBJECTIVE

Analyse the effectiveness of Ziegler-Nichols rules in presence of poles with high multiplicity.

(as a CAD interface to MATLAB)

## 2. GIVEN INFORMATION

The basic OLTF unit provided that is to be considered at different multiplicities :

$$G(s) = \frac{1}{(s + 1.5)^k} ; k = 1, 2, 3, \dots, 10$$

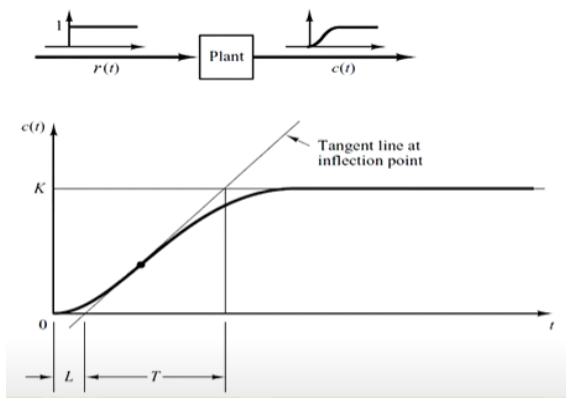
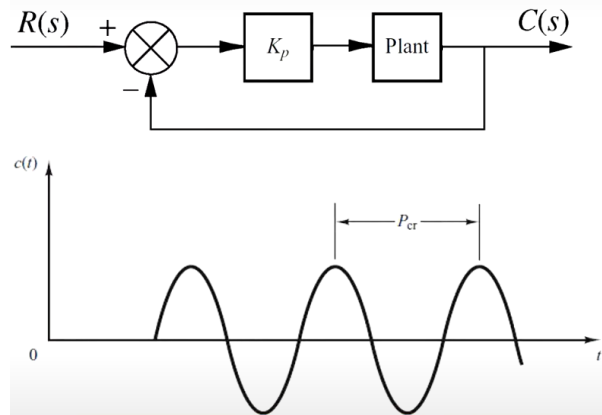
In the above equation,  $k$  ( *the order of multiplicity* ) assumes any integer value between one to ten .

## 3. INTRODUCTION : Ziegler-Nichols Rules

The intention of Ziegler-Nichols tuning rules was to achieve a fast closed loop step response without excessive oscillations and excellent disturbance rejection. Ziegler and Nicholas proposed some rules for determining the values of

- a)  $K_p$  - Controller's path gain
- b)  $T_I$  - Controller's integrator time constant
- c)  $T_D$  - Controller's derivative time constant .

The two methods of the Ziegler-Nichols tuning are described below:

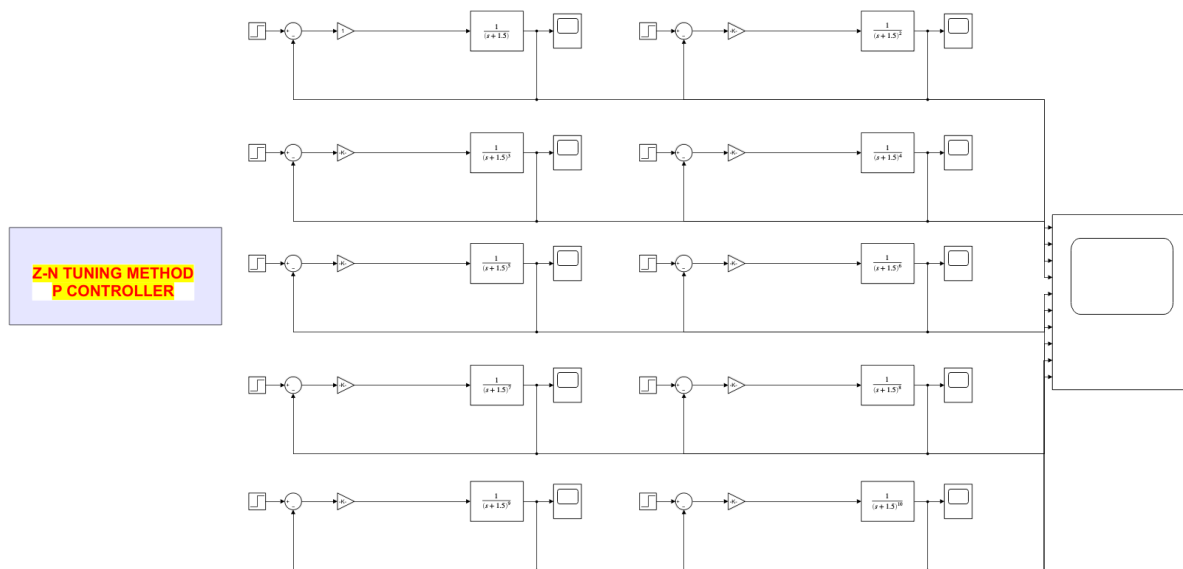
First Method : LAG TYPE RESPONSE	Second method : OSCILLATORY TYPE RESPONSE																																
Based on open-loop concepts relying on reaction curves	Based on closed-loop concepts requiring the computation of the critical gain and critical period																																
<div></div>	<div></div>																																
Obtain experimentally the response of the plant to a unit-step input.	Set $T_i \rightarrow \infty$ and $T_d = 0$ .																																
The process gain is defined as $K_{ss} = \frac{Y_{ss}}{U_{ss}}$ where Y SS is the steady state value of the output response. Thus, gain parameter $K = \frac{T}{L \cdot K_{ss}}$	If the output does not exhibit sustained oscillations for whatever value Kp may take, then this method does not apply.																																
<table><tr><th>Type</th><th><math>K_p</math></th><th><math>T_I</math></th><th><math>T_D</math></th></tr><tr><td>P</td><td><math>K</math></td><td><math>\infty</math></td><td>0</td></tr><tr><td>PI</td><td><math>0.9K</math></td><td><math>\frac{L}{0.3}</math></td><td>0</td></tr><tr><td>PID</td><td><math>1.2K</math></td><td><math>2L</math></td><td><math>0.5L</math></td></tr></table>	Type	$K_p$	$T_I$	$T_D$	P	$K$	$\infty$	0	PI	$0.9K$	$\frac{L}{0.3}$	0	PID	$1.2K$	$2L$	$0.5L$	<table><tr><th>Type</th><th><math>K_p</math></th><th><math>T_I</math></th><th><math>T_D</math></th></tr><tr><td>P</td><td><math>0.5 K</math></td><td><math>\infty</math></td><td>0</td></tr><tr><td>PI</td><td><math>0.45K</math></td><td><math>\frac{T_u}{1.2}</math></td><td>0</td></tr><tr><td>PID</td><td><math>0.6K</math></td><td><math>\frac{T_u}{2}</math></td><td><math>\frac{T_u}{8}</math></td></tr></table>	Type	$K_p$	$T_I$	$T_D$	P	$0.5 K$	$\infty$	0	PI	$0.45K$	$\frac{T_u}{1.2}$	0	PID	$0.6K$	$\frac{T_u}{2}$	$\frac{T_u}{8}$
Type	$K_p$	$T_I$	$T_D$																														
P	$K$	$\infty$	0																														
PI	$0.9K$	$\frac{L}{0.3}$	0																														
PID	$1.2K$	$2L$	$0.5L$																														
Type	$K_p$	$T_I$	$T_D$																														
P	$0.5 K$	$\infty$	0																														
PI	$0.45K$	$\frac{T_u}{1.2}$	0																														
PID	$0.6K$	$\frac{T_u}{2}$	$\frac{T_u}{8}$																														

---

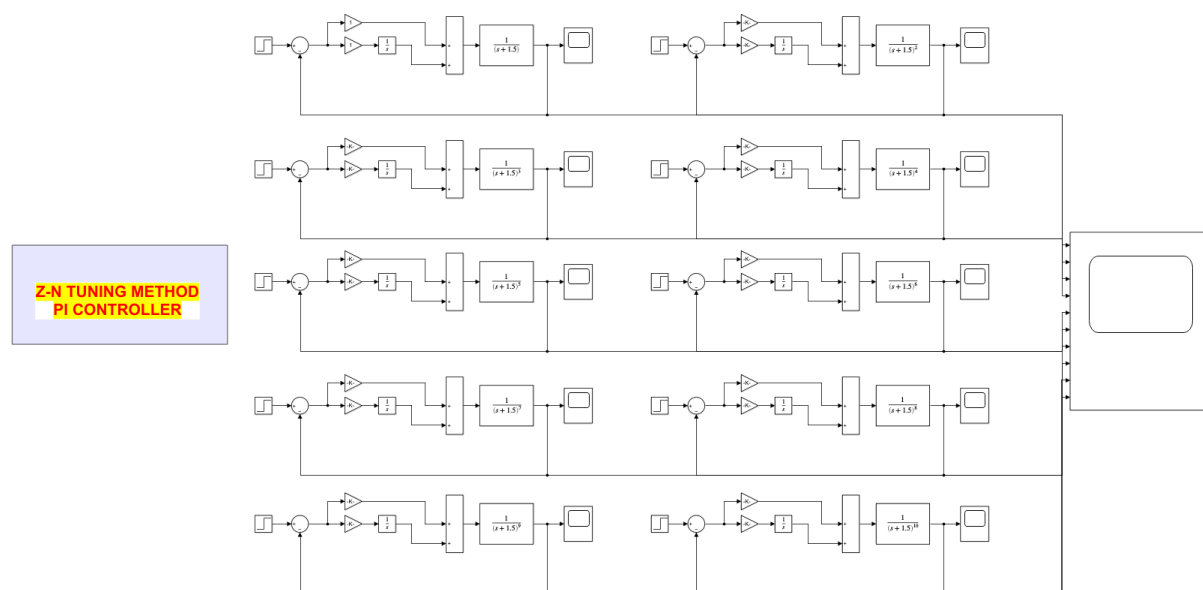
## 4. ZIEGLER - NICHOLS TUNING METHOD

### 4.1. Simulink Diagrams

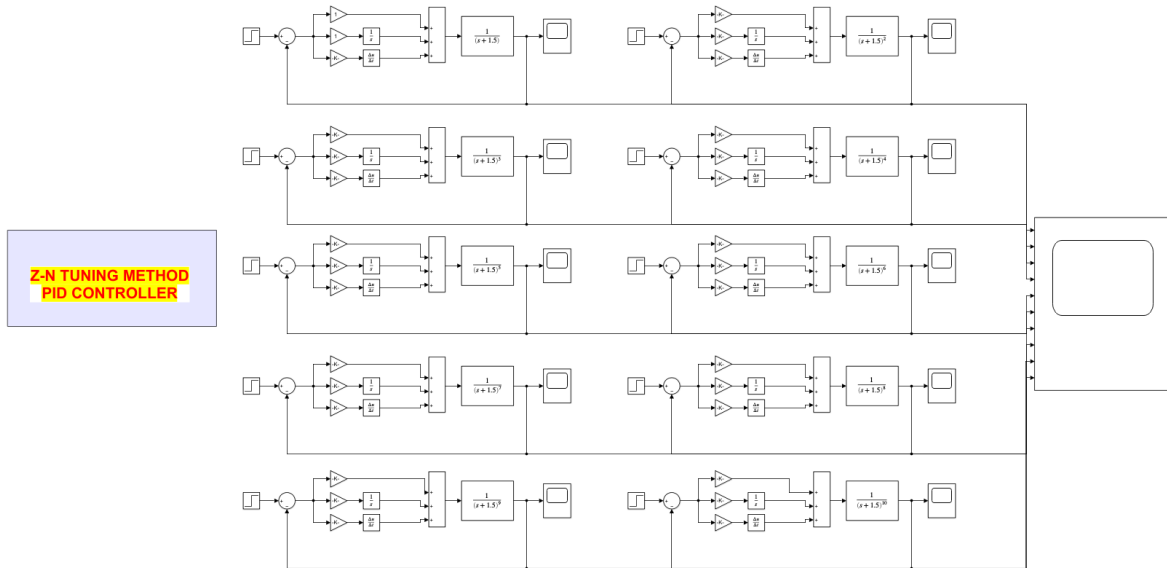
#### 4.1.1 P controller for ( $k = 1, 2, 3, \dots, 10$ )



#### 4.1.2 PI controller for ( $k = 1, 2, 3, \dots, 10$ )



### 4.1.3 PID controller for ( $k = 1, 2, 3, \dots, 10$ )



## 4.2. Tuned constants for P, PI, and PID control

### 4.2.1 P controller for ( $k = 1, 2, 3, \dots, 10$ )

Constants	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
$K_P$	-	9.6295	4.5867	3.1305	2.4336	2.0236	1.7503	1.5563	1.4097	1.2927
$K_I$	0	0	0	0	0	0	0	0	0	0
$K_D$	0	0	0	0	0	0	0	0	0	0

### 4.2.2 PI controller for ( $k = 1, 2, 3, \dots, 10$ )

Constants	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
$K_P$	Inf	8.6666	4.1280	2.8175	2.1903	1.8213	1.5753	1.4007	1.2688	1.1635
$K_I$	Inf	13.844	2.3063	0.8895	0.4693	0.2915	0.1998	0.1464	0.1123	0.0892
$K_D$	0	0	0	0	0	0	0	0	0	0

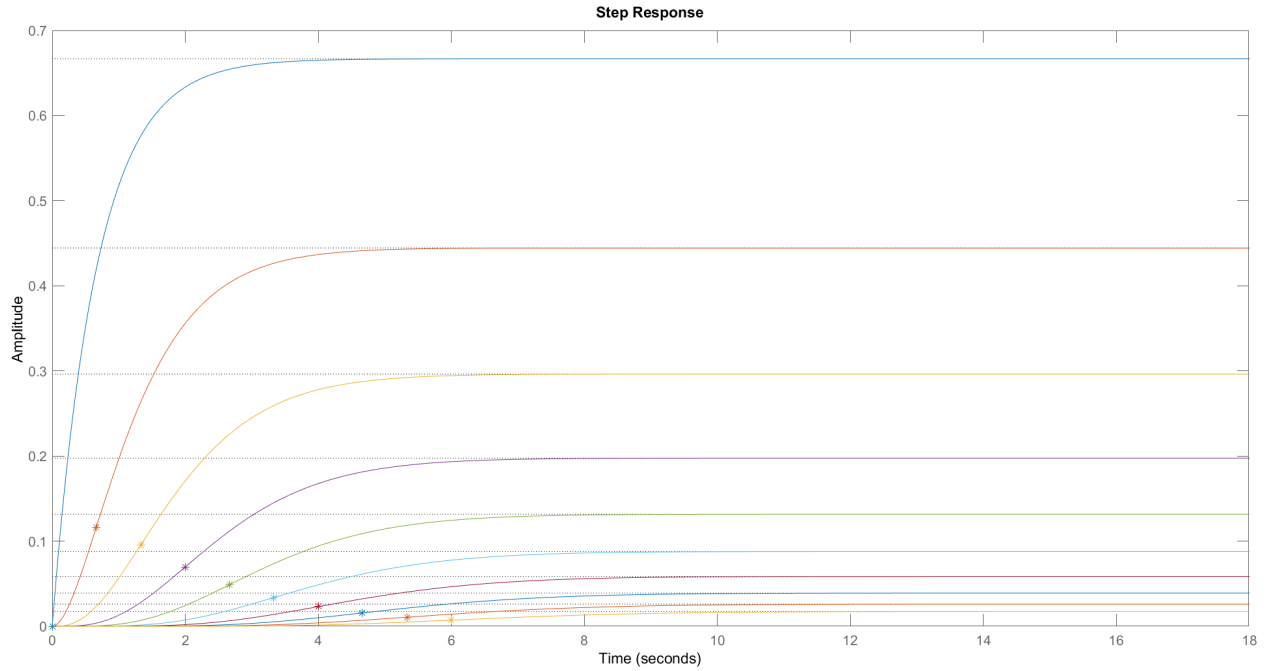
### 4.2.3 PID controller for ( $k = 1, 2, 3, \dots, 10$ )

Constants	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
$K_P$	Inf	11.5554	5.5040	3.7567	2.9204	2.4284	2.1004	1.8676	1.6917	1.5513
$K_I$	Inf	30.7642	5.1251	1.9766	1.0429	0.6478	0.4438	0.3253	0.2497	0.1982
$K_D$	0.4092	1.0850	1.4777	1.7849	2.0444	2.2756	2.4847	2.6811	2.8652	3.0346

---

## 5. OBSERVATIONS & THEIR ANALYSIS

### 5.1 Inflection Points



These are the initial step responses of transfer functions from  $k = 1$  to  $k = 10$  (As  $k$  increases, the steady state value reduces gradually) . Further, we find the values and of their inflection points and plot the same with various multiplicities on the above graph (except at  $k = 1$ ). These responses for the original system are found to be an '*S*' – *shaped curve*. This is a trait of lag systems.

So, we apply the lag Z-N rules for the lag type systems.

### 5.2 The anomalous case of $k = 1$

We observed that any inflection point could not be found in case of multiplicity  $k = 1$ .

As it is obvious from the given transfer function  $\frac{1}{(s+1.5)}$  that for unit step input, the output will be  $\frac{1}{s(s+1.5)}$  whose laplace inverse will give  $y(t) = \frac{2}{3} \{1 - e^{-1.5t}\}$ . On differentiating it wrt time, we get  $y'(t) = e^{-1.5t}$  and differentiating again, we deduce  $y''(t) = -1.5e^{-1.5t}$ .

Now, we know that at  $y''(t) = 0$  at inflection point but for transfer function with multiplicity  $k = 1$  we obtain  $y''(t) = -1.5e^{-1.5t}$  which does not equate to '0' for any finite possible value of time  $t$ .

Thus we conclude that there exists no inflection point for  $G(s) = \frac{1}{(s+1.5)}$ .

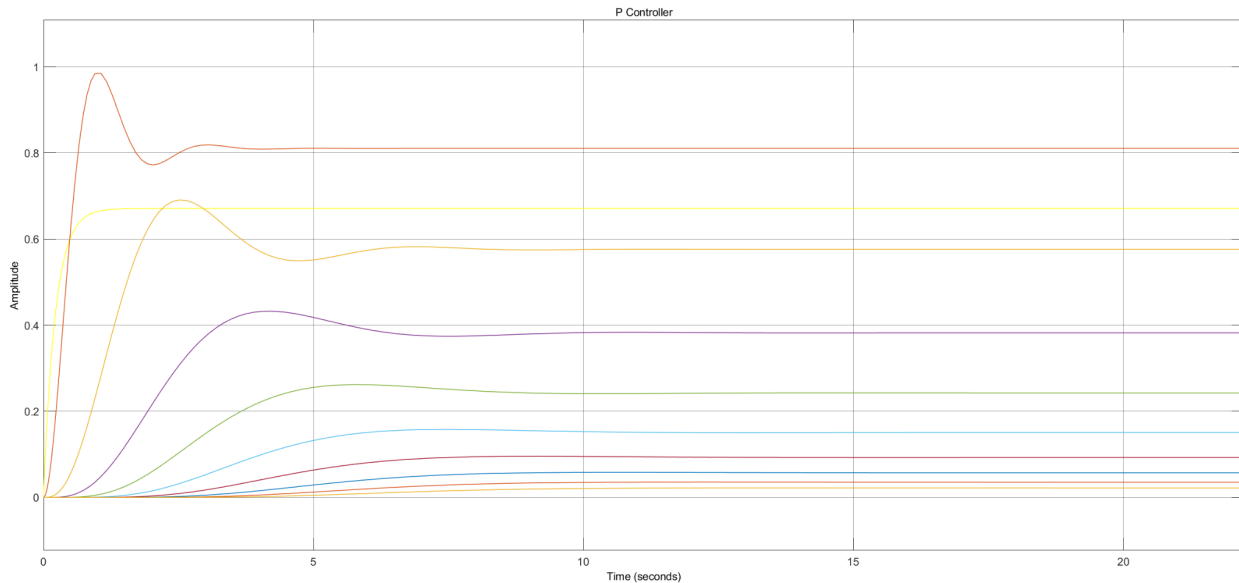
Further  $y'(t) = e^{-1.5t}$  is a monotonic decreasing function which means the slope of the curve (output response) always keeps decreasing and hence the required tangent for finding the Z-N parameters is drawn at the starting point itself (where  $t = 0$ ), but this lends us with the value of parameter  $L = 0$ , which inturn signifies that the gain parameter should attain an infinite value, which is not realisable at all !!

So, we found the values via `pidtune(sys,'P')` tune methods of matlab. And the values were as follows:

Types of controller →	P controller	PI controller	PID controller
$K_p$	3.06	1.56	2.15
$K_i$	0	7.04	5.84
$K_d$	0	0	0

### 5.3 Dynamic responses of CLTF curves and Performance Measures

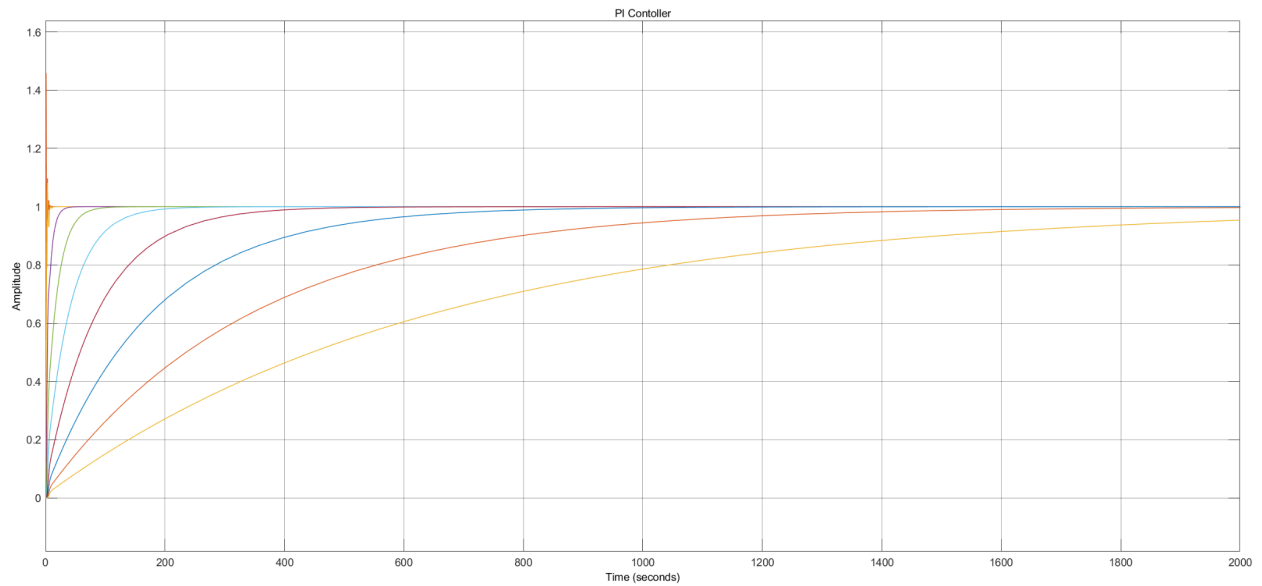
#### 5.2.1 P controller



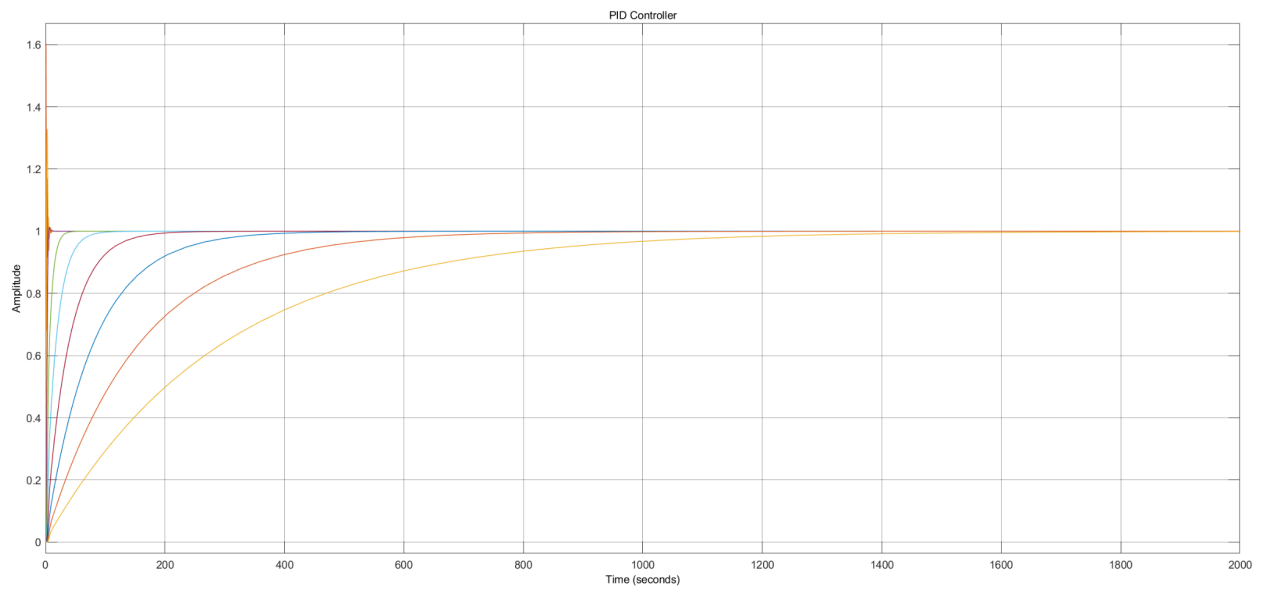


---

### 5.2.2 PI controller



### 5.2.3 PID controller



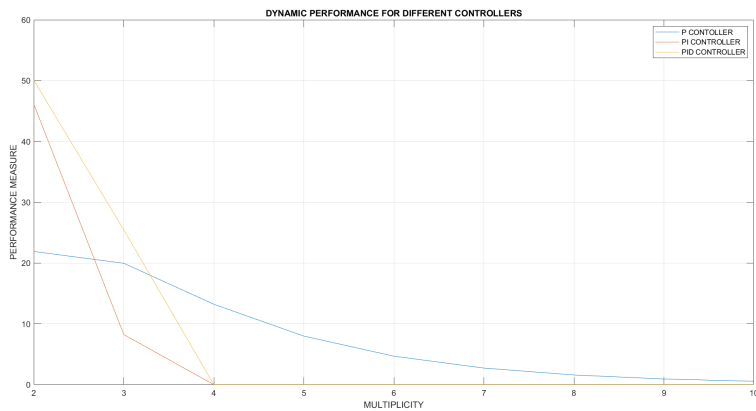
### 5.2.4 Performance Comparison Of P, PI and PID Controllers :

It should be noted that when multiplicity increases, the speed of decrease in rise time is quite fast for the PI controller as compared to the rest. The settling time decrement rate however is fastest for the P controller, followed by the PID and then the PI controller. The increase in overshoot is

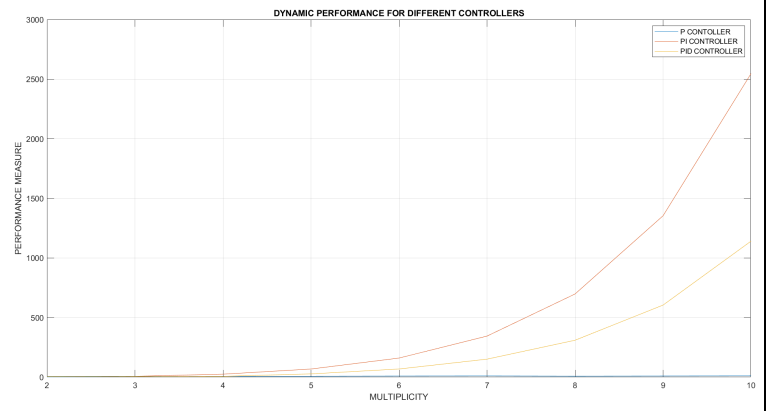
slight for the P controller but is drastic for others, and the steady state error is quite low for the PI and PID controllers as compared to the P controller.

### 5.3 Analysis of the effectiveness of the “ZN rules” for multiple poles systems

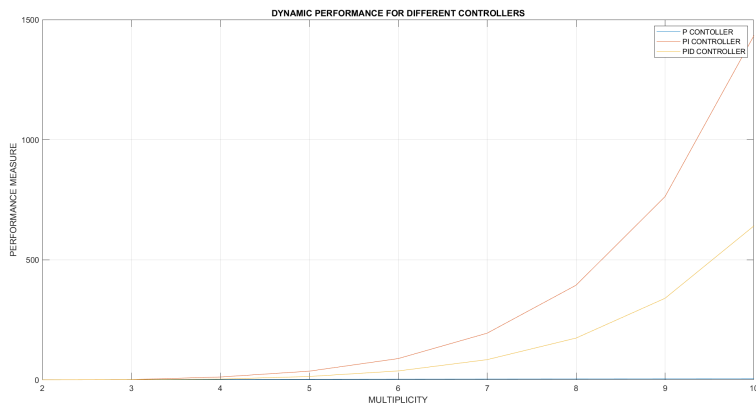
#### RISE TIME



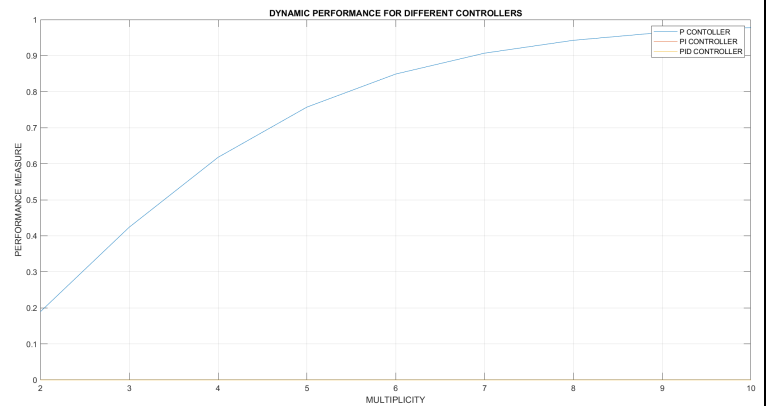
#### SETTLING TIME



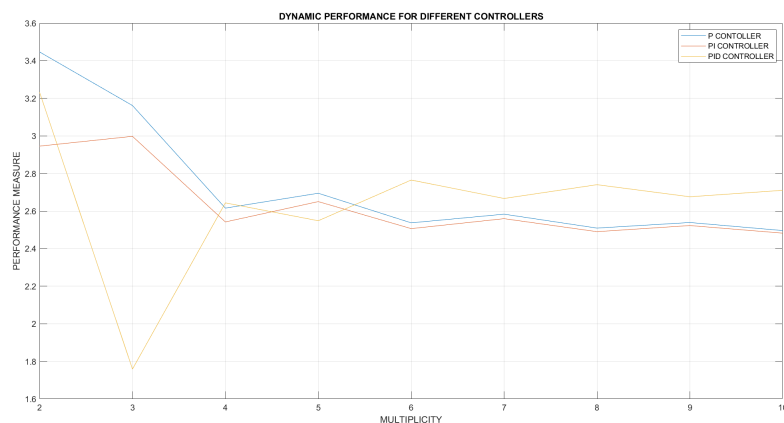
#### OVERSHOOT



#### STEADY STATE ERROR



#### STABILITY USING EIGENVALUES





Variation as multiplicity ↑	P Controller	PI Controller	PID Controller
Rise Time	Decreases Gradually	Decreases Faster	Decreases
Settling Time	Settles very fast	Settles very slowly	Settles slowly
Overshoot	Increases Slightly	Increases Drastically	Increases Drastically
Steady State Error	Error increases	Very minute error	Very minute error

#### 5.4 Overall effectiveness of Z-N Rules:

1. P Controller:

- Not capable of stabilising higher order systems
- Works decent for first order systems
- To be used where the steady state value is not a factor
- Magnitude of eigenvalues is in between PI and PID. Oscillatory behaviour is reduced

2. PI Controller:

- Eliminates steady state error.
- Should not be used where the response needs to be achieved quickly (where low settling time is desired).
- It has a negative effect on speed of the response. PI controllers cannot determine the effect of forthcoming response.
- Magnitude of eigenvalues is lowest. Oscillatory behaviour is reduced.

3. PID Controller:

- Steady state error is minimised and it takes a long time to achieve the steady state.
- Stability is increased.
- A decrease in time constant increases the speed of settling time as compared to the case of PI controller
- Magnitude of eigenvalues is highest. Oscillatory behaviour is reduced.

- 
- ❖ Proportional controller will help in minimising the rise time.
  - ❖ With the help of an additional integrator the steady state error is reduced to almost zero. It will also increase the time to achieve steady state.
  - ❖ Using the derivative controller we will be able to maximise the eigenvalues of the closed loop system and hence increasing the overall stability. It will also reduce the overshoot and improve the transient response of the system.

### 5.5 Limitations of Z-N Rules:

- The required parameters for the Z-N test cannot be found for some abrupt situations like in this experiment only, for  $k = 1$  we could not find a desirable inflection point and hence the tangent could be drawn only at  $t = 0$  (except the steady state asymptote) , which led to the value of  $L = 0$  and hence non realisable  $k$  and we could not use the Z-N method here.
- The steady state response is achieved after a long time, so the usage of these rules where we have to make a dead-time dominant process is not satisfactory.

## 6. CONCLUSION

- The P, PI and PID controllers of different multiplicities ( $k = 1$  to  $k = 10$ ) using the Ziegler-Nichols rules were designed using simulink.
- The dynamic step responses for the controlled *CLTF* systems were plotted and analysed. The trends in performance measures were also taken into account.
- The effectiveness of the Ziegler-Nichols rules was also discussed in context to different multiplicities of the system poles.
- The limitations of the Ziegler-Nichols test were also summarised.

## 7. MATLAB SCRIPT

```

1  clc;
2
3  L_vals = zeros(1,10);
4  T_vals = zeros(1,10);
5  K_P = zeros(3,10);
6  K_PI = zeros(3,10);
7  K_PID = zeros(3,10);
8
9  for n = 1:10
10
11     syms s;
12     tf1 = 1/((s+1.5)^n);
13
14     Y_s = tf1/s;
15     yt = ilaplace(Y_s);
16     y2t = diff(diff(yt));
17     inf_pt = solve(y2t == 0);
18
19     s = tf('s');
20     tf1 = 1/((s+1.5)^n);
21     [y,time] = step(tf1);
22     pidentune(tf1,'PID');
23
24     l = time(find(y>=0.5*y(end),1));
25     t = time(find(y>= (1-exp(-1))*y(end),1));
26     D = diff(y)./diff(time);
27     inflex = find(diff(D)./diff(time(1:end-1))<0,1);
28     A = D(inflex)*time(inflex)-y(inflex);
29     tangent = D(inflex)*time - A;
30     step(tf1)
31     hold on
32     plot(time(inflex),y(inflex),'*')
33     hold on
34     L_vals(n) = (-y(inflex)/D(inflex))+time(inflex);
35     T_vals(n) = time(inflex)+((y(end-1)-y(inflex))/D(inflex))-L_vals(n);
36     K_P(:,n) = [T_vals(n)/L_vals(n) 0 0];
37     K_PI(:,n) = [(0.9*T_vals(n))/L_vals(n) (0.27*T_vals(n))/(L_vals(n)*L_vals(n)) 0];

```

```

37     K_PI(:,n) = [(0.9*T_vals(n))/L_vals(n) (0.27*T_vals(n))/(L_vals(n)*L_vals(n)) 0];
38     K_PID(:,n) = [(1.2*T_vals(n))/L_vals(n) (0.6*T_vals(n))/(L_vals(n)*L_vals(n)) 0.6*T_vals(n)];
39
40 end
41 hold off
42
43 OVERSHOOT = zeros(3,10);
44 RISE_TIME = zeros(3,10);
45 SETT_TIME = zeros(3,10);
46 SS_ERRORS = zeros(3,10);
47 EIG_VALUE = zeros(3,10);
48
49 for n = 2:10
50     s = tf('s');
51     sys = 1/((s+1.5)^n);
52     step(feedback(sys*pidentune(sys,'PID'),1))
53
54     C = pid(K_P(1,n),K_P(2,n),K_P(3,n),0);
55     G_CL = feedback(sys*C,1);
56     OVERSHOOT(1,n) = stepinfo(G_CL).RiseTime;
57     RISE_TIME(1,n) = stepinfo(G_CL).Overshoot;
58     SETT_TIME(1,n) = stepinfo(G_CL).SettlingTime;
59     SS_ERRORS(1,n) = 1-dcgain(G_CL);
60     EIG_VALUE(1,n) = max(abs(eig(G_CL)));
61
62     C = pid(K_PI(1,n),K_PI(2,n),K_PI(3,n),0);
63     G_CL = feedback(sys*C,1);
64     OVERSHOOT(2,n) = stepinfo(G_CL).RiseTime;
65     RISE_TIME(2,n) = stepinfo(G_CL).Overshoot;
66     SETT_TIME(2,n) = stepinfo(G_CL).SettlingTime;
67     SS_ERRORS(2,n) = 1-dcgain(G_CL);
68     EIG_VALUE(2,n) = max(abs(eig(G_CL)));
69
70     C = pid(K_PID(1,n),K_PID(2,n),K_PID(3,n),0);
71     G_CL = feedback(sys*C,1);
72     OVERSHOOT(3,n) = stepinfo(G_CL).RiseTime;
73     RISE_TIME(3,n) = stepinfo(G_CL).Overshoot;

```

---

```
73     RISE_TIME(3,n) = stepinfo(G_CL).Overshoot;
74     SETT_TIME(3,n) = stepinfo(G_CL).SettlingTime;
75     SS_ERRORS(3,n) = 1-dcgain(G_CL);
76     EIG_VALUE(3,n) = max(abs(eig(G_CL)));
77
78     hold on
79 end
80
81 hold off
82
83 plot(linspace(2,10,9),EIG_VALUE(1,2:10))
84 hold on
85 plot(linspace(2,10,9),EIG_VALUE(2,2:10))
86 hold on
87 plot(linspace(2,10,9),EIG_VALUE(3,2:10))
88 xlabel('MULTIPLICITY')
89 ylabel('PERFORMANCE MEASURE')
90 title('DYNAMIC PERFORMANCE FOR DIFFERENT CONTROLLERS')
91 legend('P CONTROLLER','PI CONTROLLER','PID CONTROLLER')
92 grid
```