EE 208

Control Engineering Lab

Experiment-9: Simulation and control design on Simulink as a CAD interface to MATLAB.

Group Number: 20

Vedansh: 2020EEB1218

Aniket Arya: 2020EEB1157

Chirag Rathi: 2020EEB1165

Professor: Dr. S. Roy

Date: 27/03/2022

OBJECTIVE: -

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

Given: -

The basic OLTF unit provided considering different multiplicities is:

$$G(s) = \frac{1}{(s+1.5)^k} \text{ where } k = 1\ldots, 10$$

Ziegler–Nichols method

Ziegler and Nichols proposed a heuristic Proportional-Integral-Derivative (PID) control structure to obtain most appropriate values so as to achieve required regulation for the following gain parameters:

a) K P - Controller's path gain
b) TI - Controller's integrator time constant
c) T D - Controller's derivative time constant

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

- **LAG TYPE RESPONSE:**

This method is applicable only if the open loop system response to a step input exhibits an S-shaped curve.

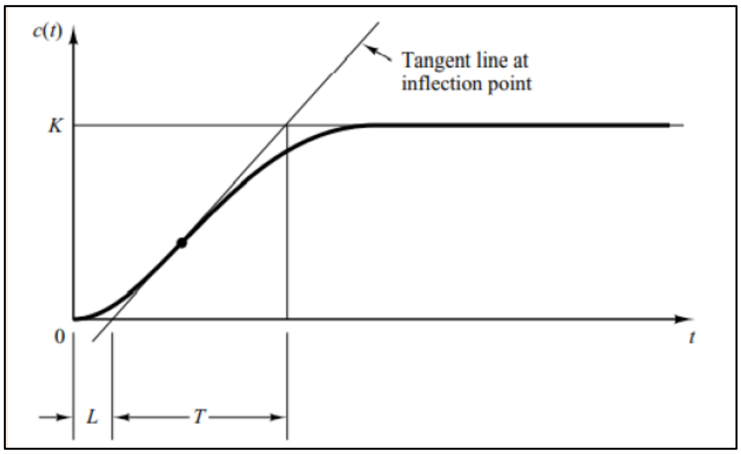
Here, we obtain the step response of the open loop system without any type of controller.
The output S-shaped curve can be characterized by the constants - delay time (L) and time constant (T_U).

Delay Time: - Measured as x-coordinate of the point where the tangent at the point of inflection meets the x-axis

Time constant (T_U): Measured as difference between x-coordinates of the point where the tangent at the point of inflection meets the x-axis and the point where it meets the steady state asymptote

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 $K = \frac{T}{L\ K_{ss}}$



The Z-N Rules for Lag Type response are-

Type of Controller	K_P	T_I	T_D
P	K	∞	0
PI	0.9 K	3.3 L	0
PID	1.2 K	2 L	0.5 L

PID Tuning using open loop experiment

Since the open loop poles are on the real axis –

- The step response of the OLTF is lag type for each multiplicity of the poles.
- Therefore will neglect the oscillatory method of Z-N tuning for every case.

The Open Loop Experiments were performed as explained above for various multiplicities.
The observed values and Z-N controllers are as follows:

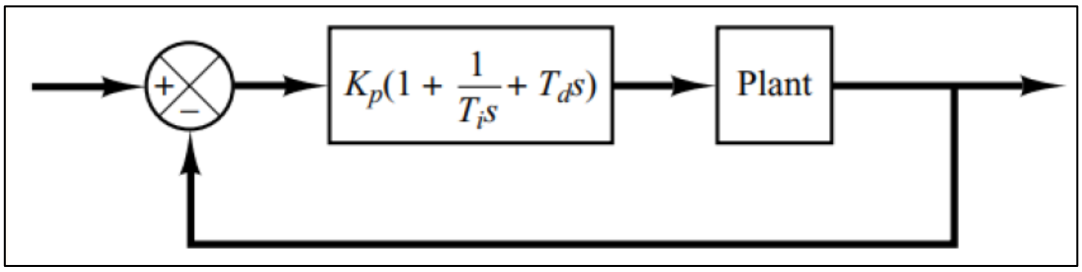
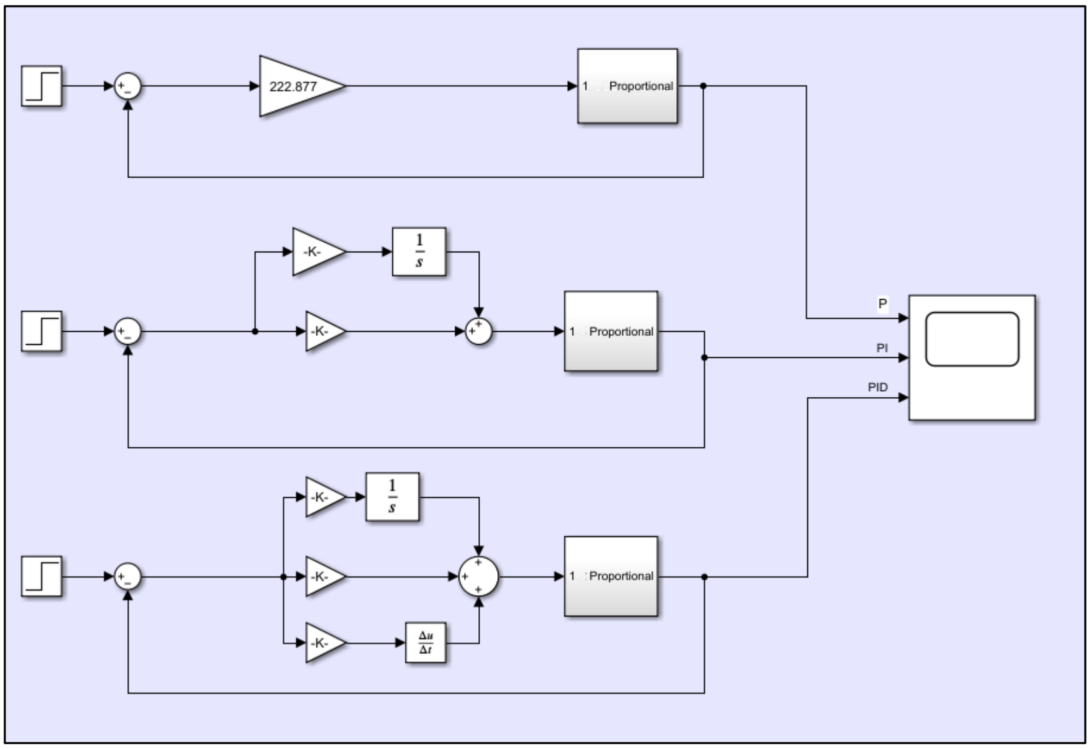
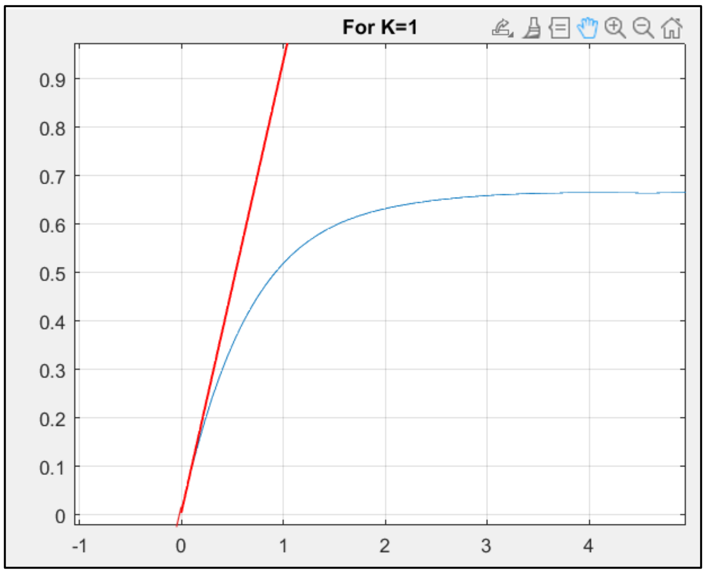


Fig- General PID controller



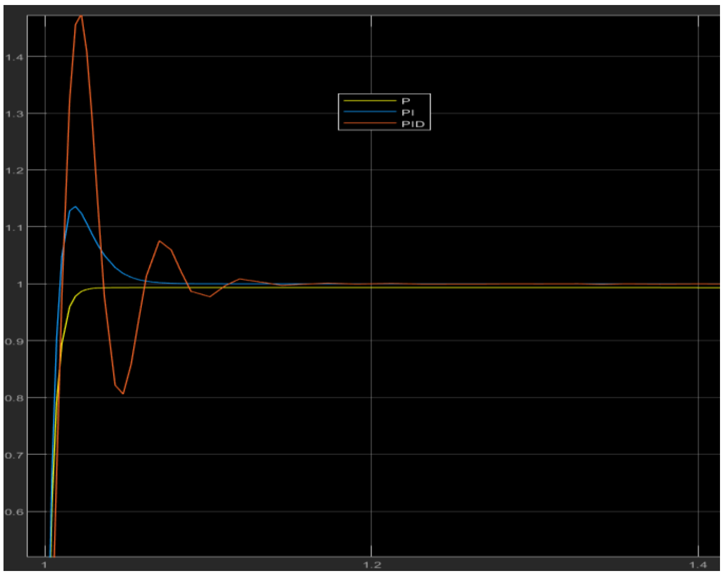
Simulink Model For P, PI, PID implication

K=1

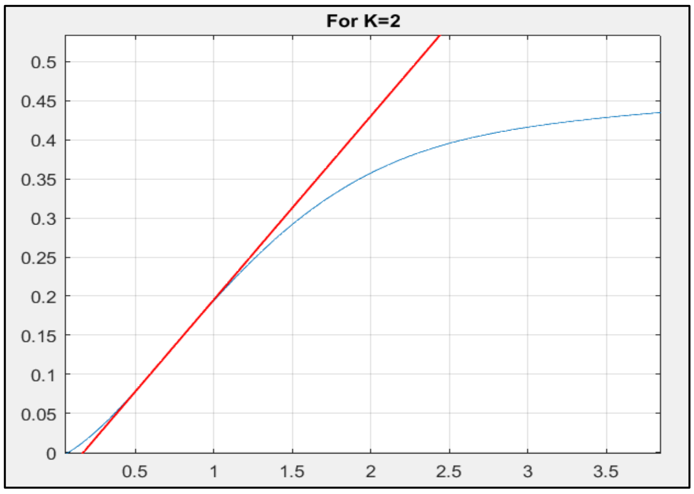


L = *-0.0048*   *T = 0.7127*

It can be observed that the value of lag time is negative while calculating through Zeigler Nicholas Method and hence the system for k=1, cannot be realised by this method. In order to design this particular multiplicity(k=1), we used MATLAB tunning method to tune this.

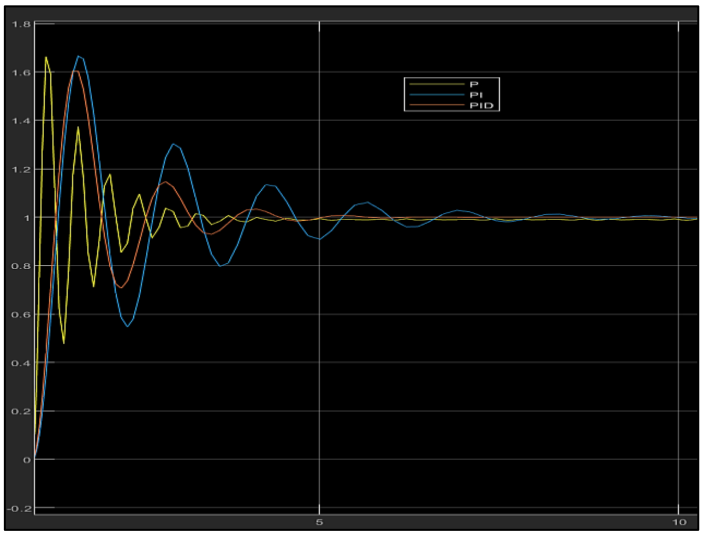


- **K = 2**

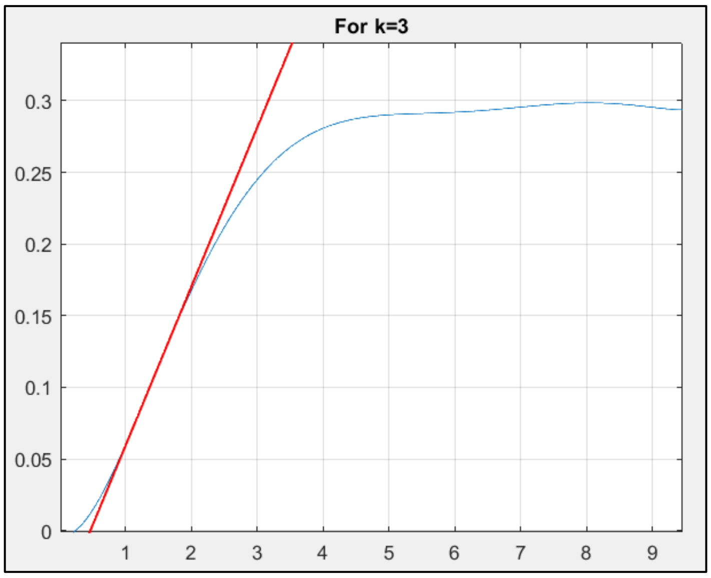


L = 0.1671 *T = 1.9040*

Type of Controller	K_P	T_I	T_D
P	25.47	∞	0
PI	22.9296	0.5513	0
PID	30.5727	0.3342	0.0835

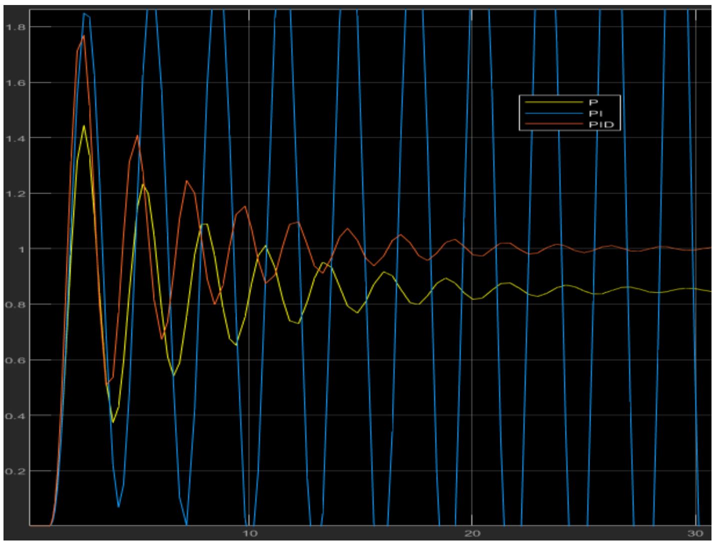


- **<u>K = 3</u>**

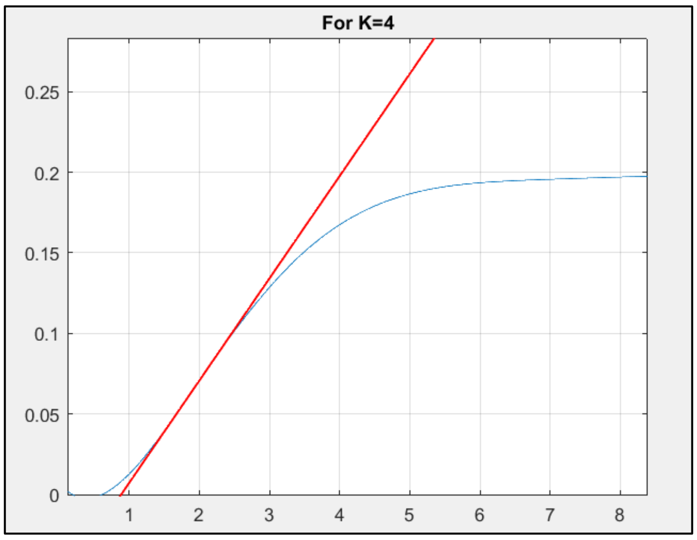


L = *0.4677* *T* = *2.7223*

Type of Controller	K_P	T_I	T_D
P	19.2361	∞	0
PI	17.3125	1.5433	0
PID	23.0833	0.9353	0.2338

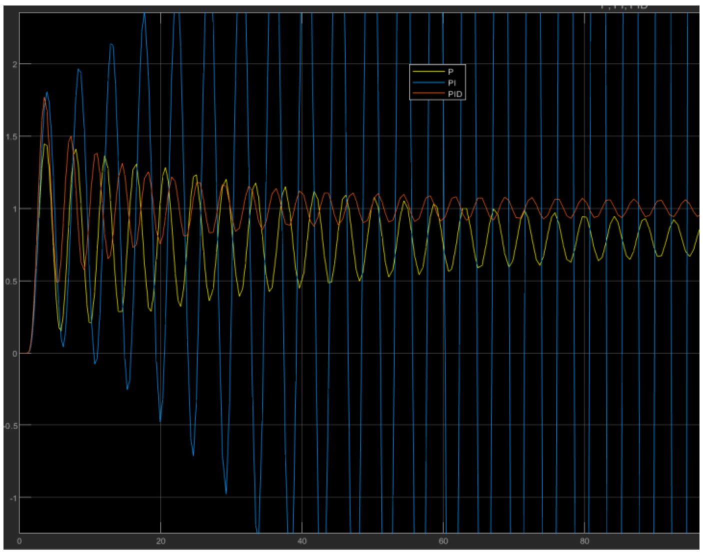


- **<u>K = 4</u>**

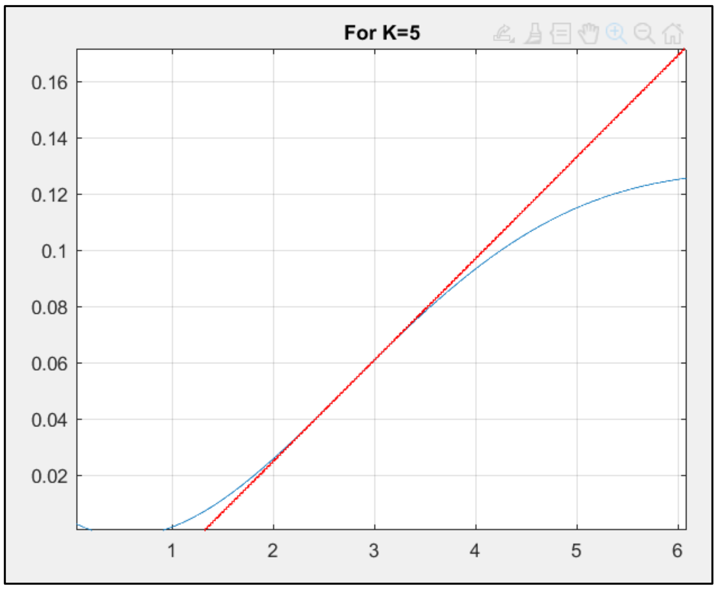


L = 0.8830 *T = 3.1070*

Type of Controller	K_P	T_I	T_D
P	19.277	∞	0
PI	16.0702	2.9138	0
PID	21.4270	1.7660	0.4415

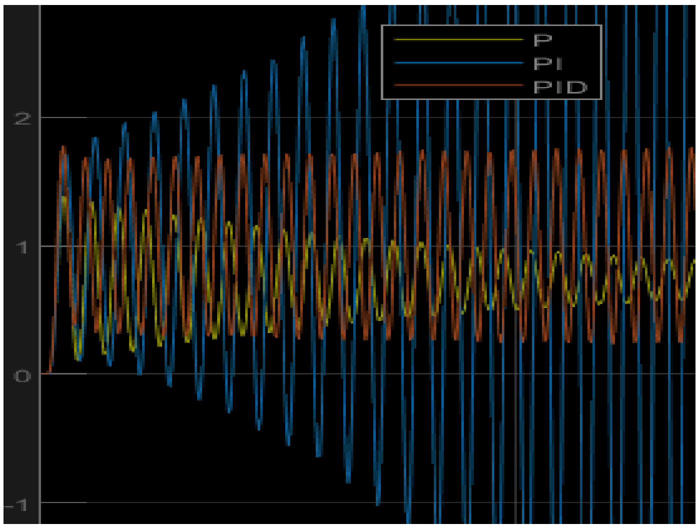


- **K = 5**



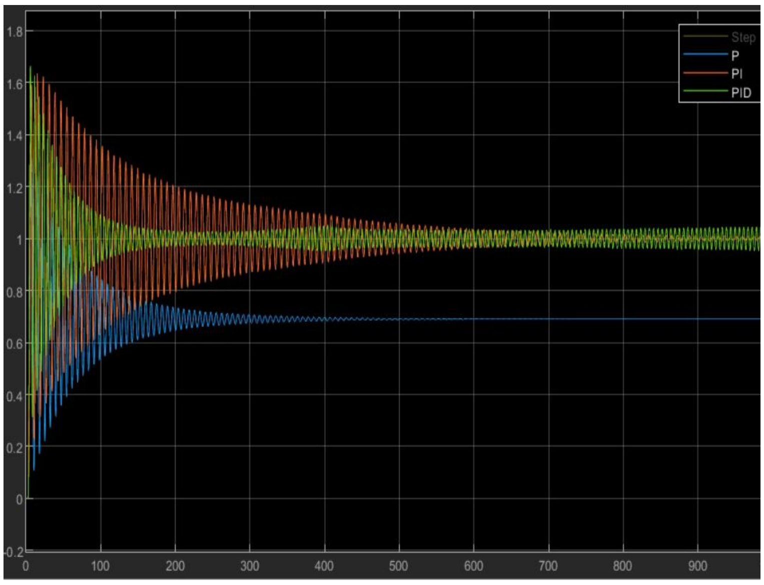
$L = 1.3106 \qquad T = 3.5840$

Type of Controller	K_P	T_I	T_D
P	21.15	∞	0
PI	19.0419	4.3249	0
PID	25.3891	2.6211	0.6553

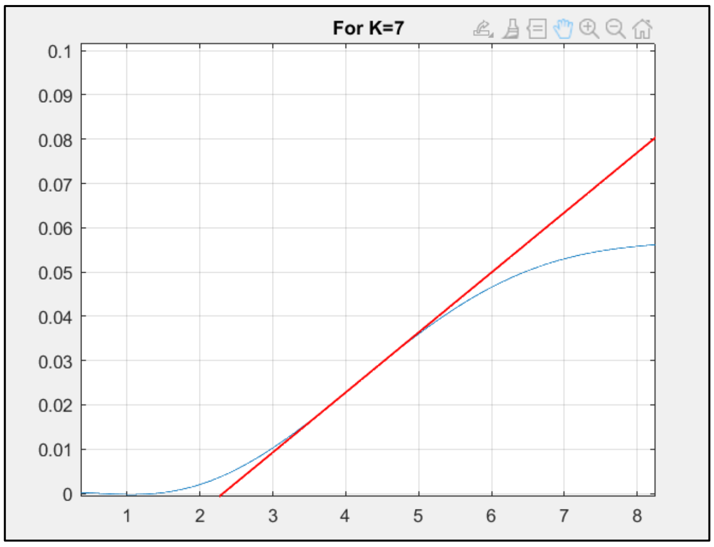


- **K = 6**
  L = 1.8031 T = 3.9304

Type of Controller	K_P	T_I	T_D
P	25.38	∞	0
PI	22.8505	5.9504	0
PID	30.4673	3.6063	0.9016

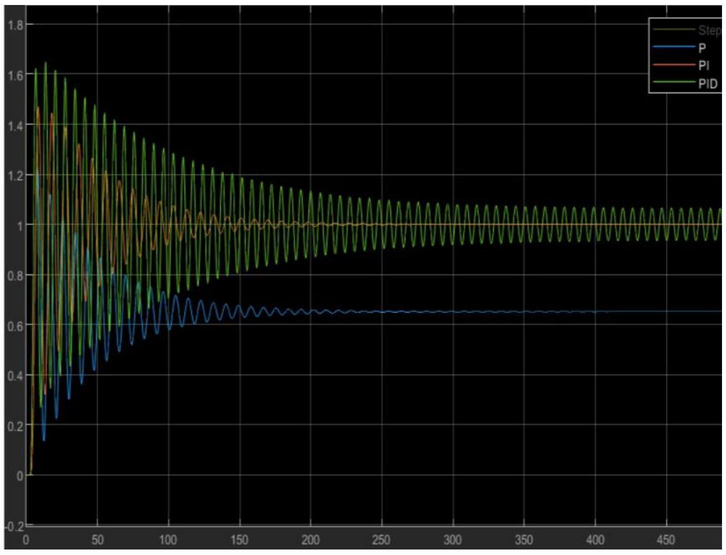


- **<u>K = 7</u>**

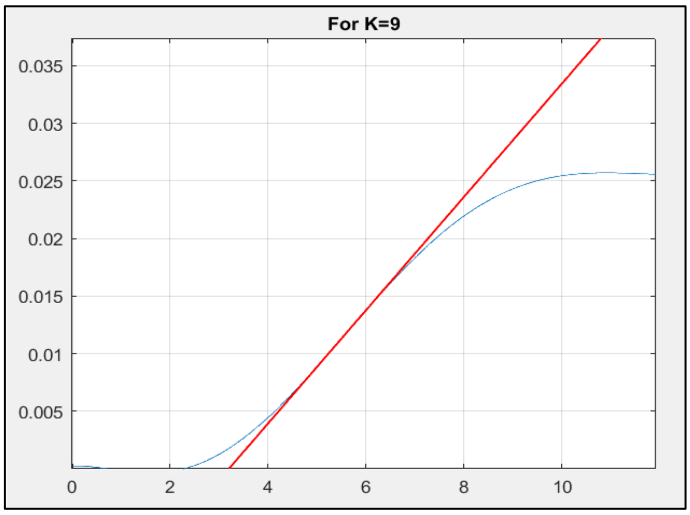


L =*2.3141* *T* = *4.2353*

Type of Controller	K_P	T_I	T_D
P	31.911	∞	0
PI	28.7255	7.6365	0
PID	38.3006	4.6282	1.1570

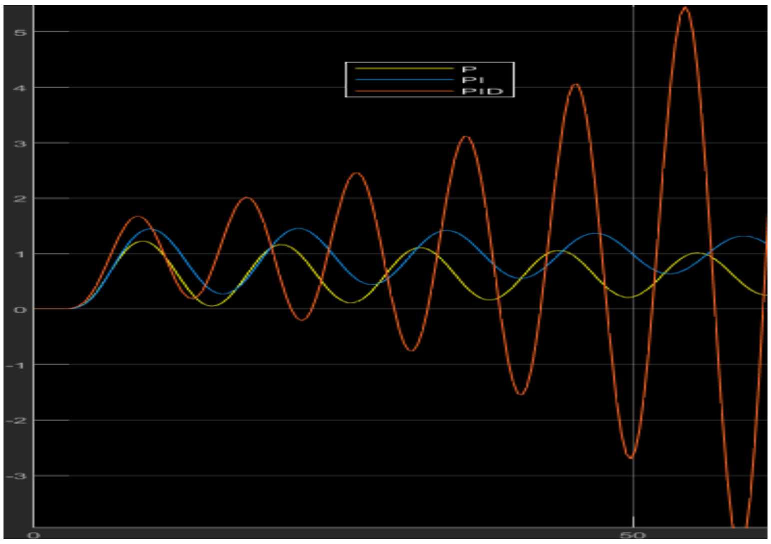


- **K = 9**



L = 3.1981   *T = 5.0906*

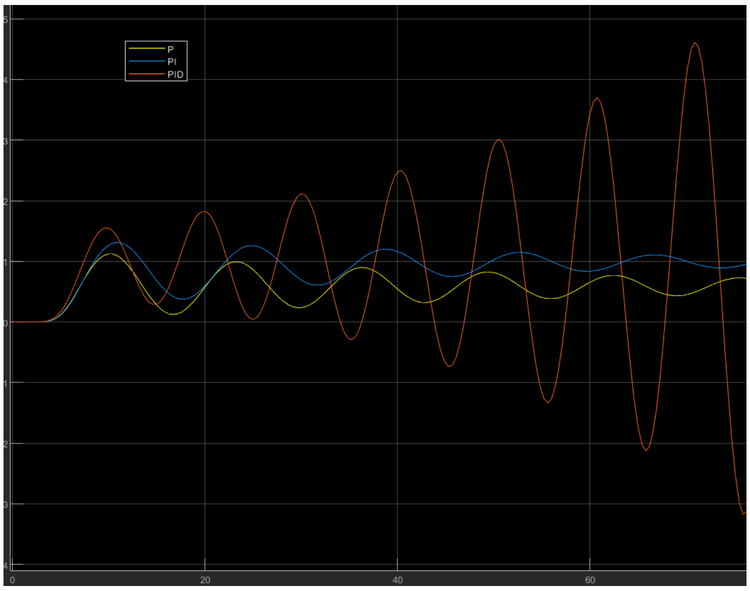
Type of Controller	K_P	T_I	T_D
P	63.63	∞	0
PI	57.2707	10.5539	0
PID	30.5727	0.3342	0.0835



- **<u>K=10</u>**

  L = 3.7734   *T = 5.2987*

Type of Controller	K_P	T_I	T_D
P	82.8344	∞	0
PI	74.5510	12.4522	0
PID	99.4013	7.5468	1.8867

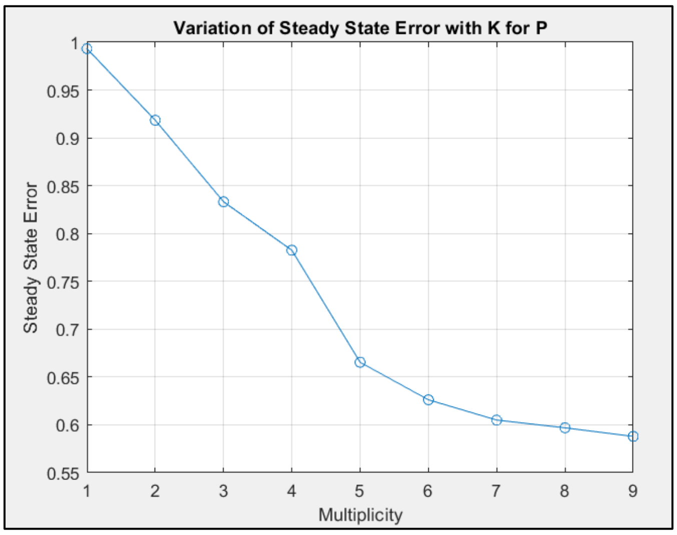


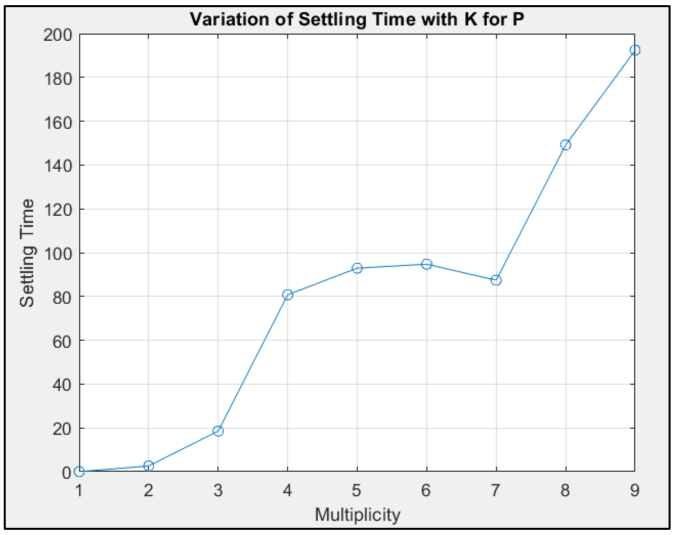
<u>Observations and analysis</u>

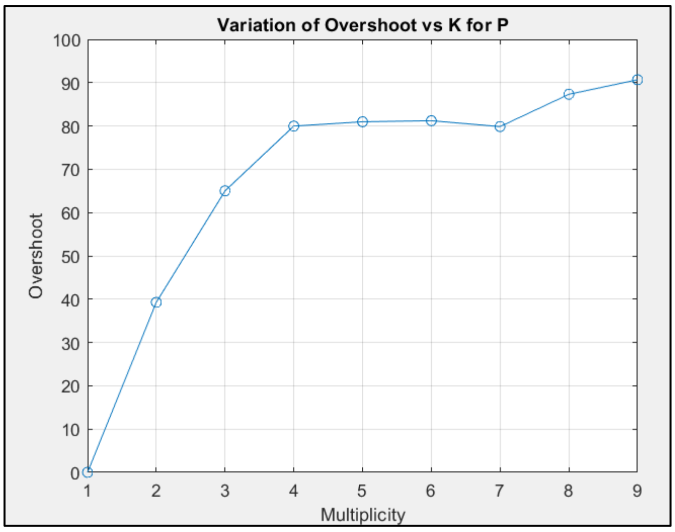
- **P Controller**

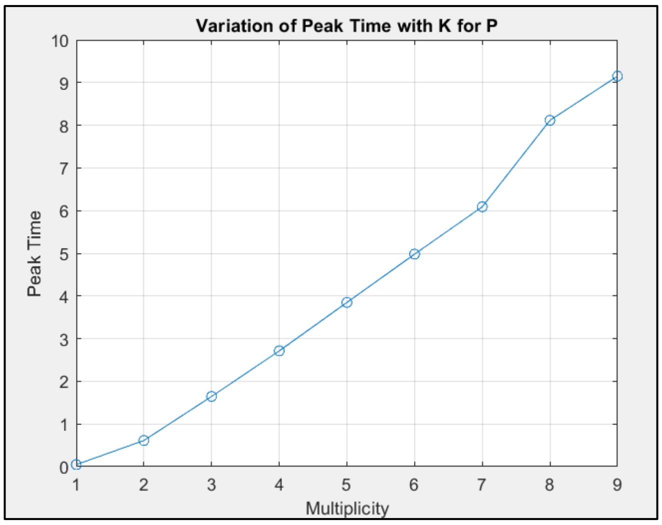
The most basic idea is to use a simple proportional controller which **can minimise the steady state error**

n	Steady state	Settling time	Overshoot	Peak Time
1	0.9932	0.0194	0	0.047
2	0.9184	2.6057	39.2747	0.6140
3	0.8332	18.5141	65.0156	1.6451
4	0.7825	80.7721	79.9741	2.7130
5	0.6654	92.9157	80.9738	3.8484
6	0.6262	94.7342	81.2153	4.9805
7	0.6050	87.3949	79.8568	6.0898
9	0.5969	149.1991	87.3070	8.1154









In case of Proportional Control (P), with increase in multiplicity k

- Steady state **error decreases**, but **never reaches at zero value**.
- Settling time **decreases**
- Overshoot **increases**
- Peak Time **increases**

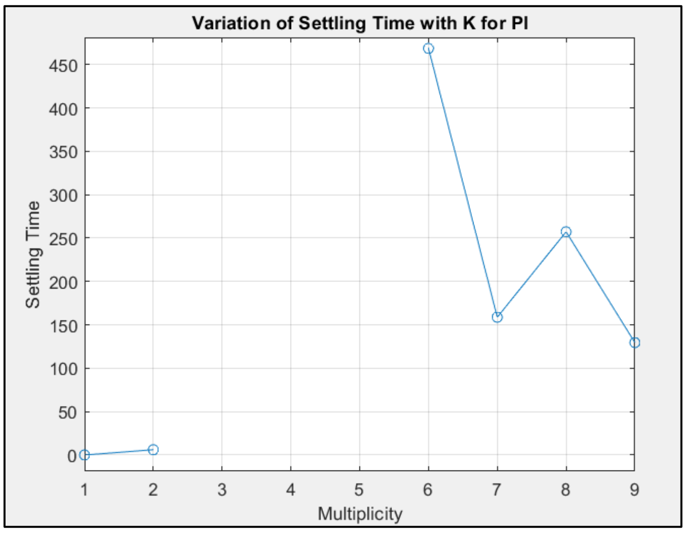
Since the P controller is designed using Z-N tuning method, it can be concluded

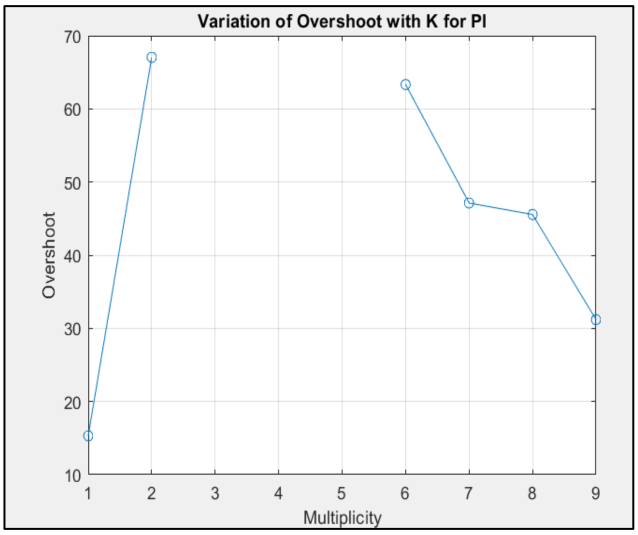
- Z-N tuning method becomes less effective with increase in multiplicity (k) as the performance parameters such as Steady state error, Overshoot, peak time increases.

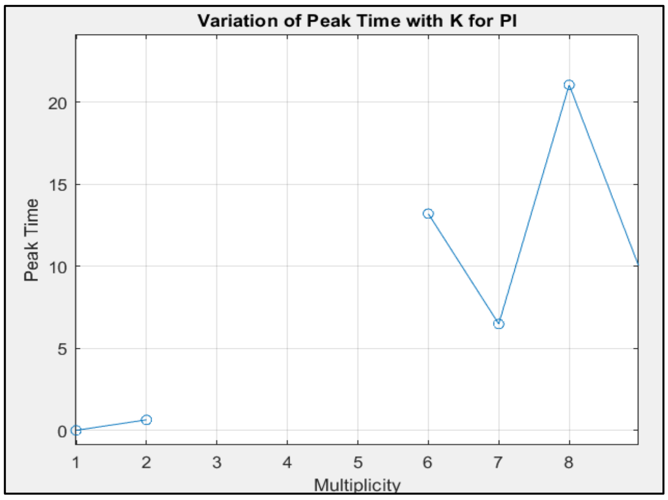
- **PI Controller**

Integral unit in parallel with the P controller **forces the steady state error to become zero at cost of the speed of the system.**

n	Steady state	Settling time	Overshoot	Peak Time
1	1	0.0461	15.3008	0.0187
2	1	6.1105	67.0431	0.6547
3	1	-	-	-
4	1	-	-	-
5	1	-	-	-
6	1	468.6011	63.3266	13.2044
7	1	158.8521	47.1450	6.4912
9	1	257.2804	45.5533	21.0474
10	1	129.5196	31.1959	9.8652







Note- discontinuity observed in the above plotted trends is due to instability of the system after cascading with the conventionally derived PI controller

For PI Controller, with increase in multiplicity (k)

- Steady state error comes out zero for every stable case.
- Trends for Settling Time, Overshoot and Peak time are non-monotonic in nature

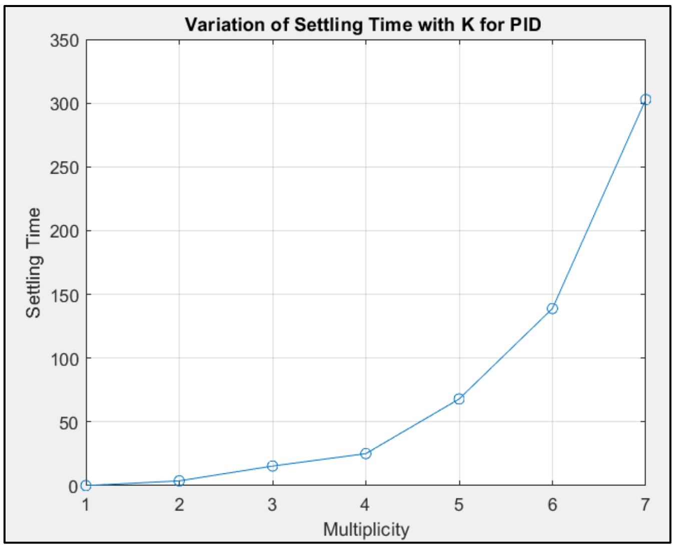
It can be concluded

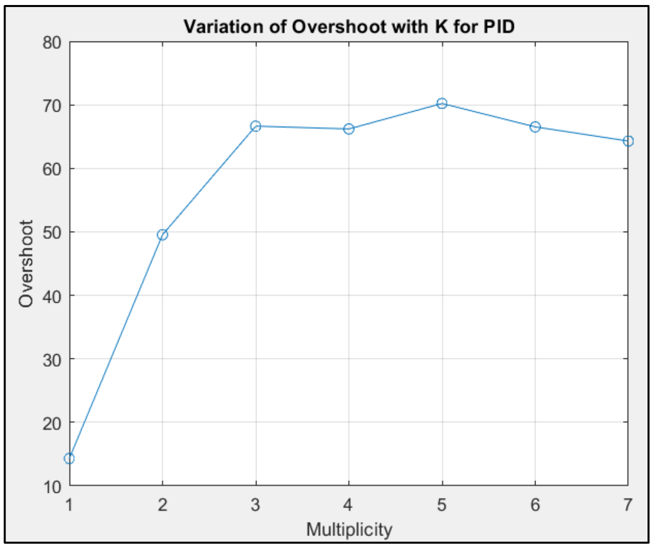
- For lower multiplicity, PI is more optimised as compared to the P controller with almost **zero steady state error** in each case.
- Still, Peak time and settling Time are more optimised for P controller, which shows that **adding Integration factor slows down the system**.
- But as the multiplicity increases, system starts showing arbitrary behaviour
- Overall system comes out **unstable for a range of multiplicity (3 – 5)**, due to relocation of a pair of conjugate poles in right hand plane.

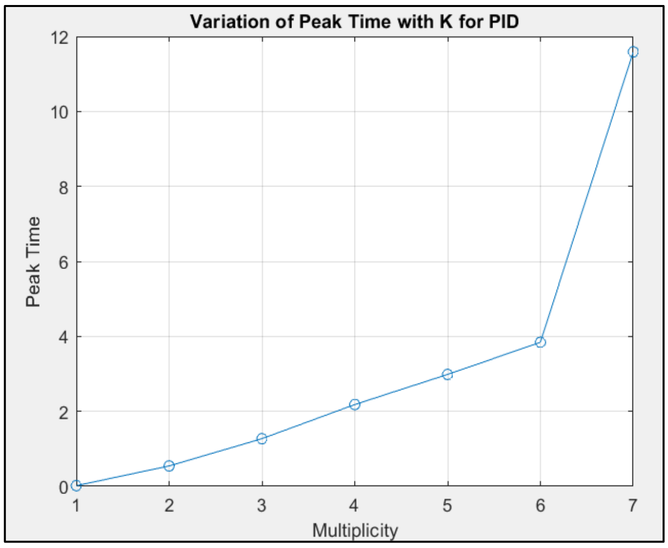
- **PID Controller**

Integrator causes a **phase lag** in the system and makes **the response slow, to compensate for this lag a differential controller** is used in parallel with a PI controller.

n	Steady state	Settling time	Overshoot	Peak Time
1	1	0.0349	14.3144	0.0174
2	1	3.7828	49.5401	0.5437
3	1	15.3902	66.6410	1.2710
4	1	25.1296	66.1967	2.1813
5	1	67.9783	70.2105	2.9852
6	1	138.9293	66.5191	3.8421
7	1	302.8322	64.2888	11.5891
9	1	-	-	-
10	1	-	-	-





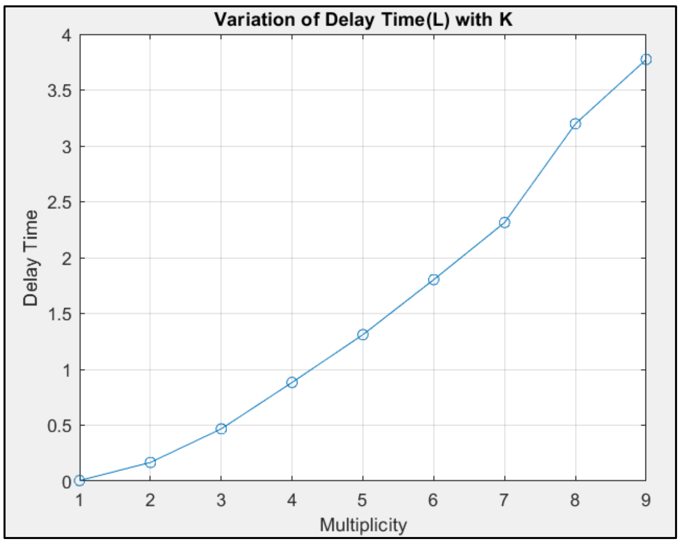


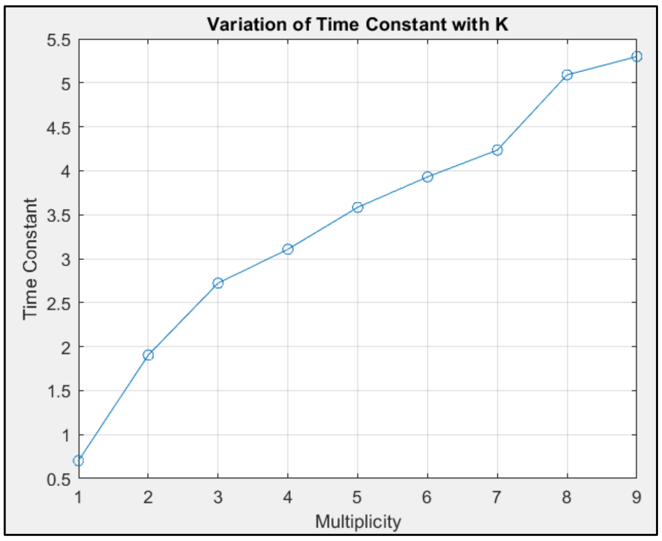
For PID Controller, with increase in multiplicity (k)

- Steady State error remains zero.
- Settling Time **increases**, but **remains less as compared to PI** controller for a specific (k).
- Overshoot **increases**.
- Peak Time **increases but remains less as compared to PI** controller for a specific (k).

It can be concluded that

- With **increasing multiplicity, the rise time is increasing**. This indicates that the controller is **becoming less effective**.
- **Settling Time increases sharply** such that the system becomes **unstable for multiplicities equal or higher than 9**, making the PID controller ineffective for higher multiplicities.
- Increase in Peak time also corresponds to the ineffectiveness of the controller for higher multiplicities.





Effectiveness of Z-N tuning method: -

From the above analysis-

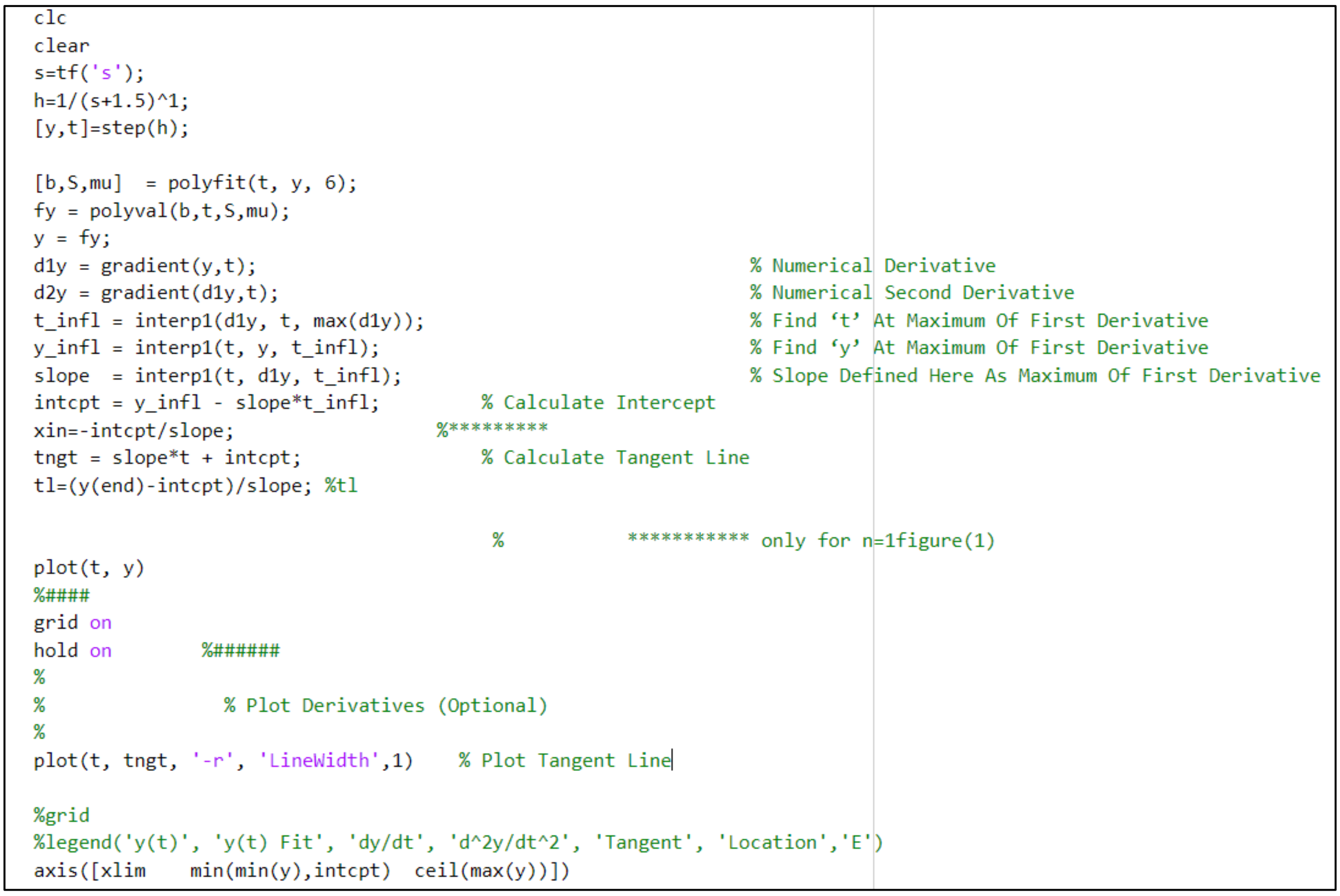
The variation of Time constant and Delay Time can be observed to **increase monotonically** with the **increase in multiplicity**, though the variation of Kp wouldn't display such behaviour as it varies directly with Time constant and inversely with Delay Time and steady state value.

- ❖ For system with lower multiplicities, Z-N tuned controllers are quite effective.
- ❖ For systems with higher multiplicities, performance of the system degraded and the system eventually becomes unstable.

Conclusion

The system was analysed for various multiplicity and the respective P, PI, PID controllers were designed for each multiplicity using the Zeigler Nicholas Method. The response of the designed cascaded systems was studied and the trends were analysed for various multiplicities of the given transfer function.

MATLAB Script



```
clc
clear
s=tf('s');
h=1/(s+1.5)^1;
[y,t]=step(h);

[b,S,mu]  = polyfit(t, y, 6);
fy = polyval(b,t,S,mu);
y = fy;
d1y = gradient(y,t);                                         % Numerical Derivative
d2y = gradient(d1y,t);                                       % Numerical Second Derivative
t_infl = interp1(d1y, t, max(d1y));                          % Find ‘t’ At Maximum Of First Derivative
y_infl = interp1(t, y, t_infl);                              % Find ‘y’ At Maximum Of First Derivative
slope  = interp1(t, d1y, t_infl);                            % Slope Defined Here As Maximum Of First Derivative
intcpt = y_infl - slope*t_infl;        % Calculate Intercept
xin=-intcpt/slope;                 %*********
tngt = slope*t + intcpt;               % Calculate Tangent Line
tl=(y(end)-intcpt)/slope; %tl

                                        %           *********** only for n=1figure(1)
plot(t, y)
%####
grid on
hold on       %######
%
%                % Plot Derivatives (Optional)
%
plot(t, tngt, '-r', 'LineWidth',1)    % Plot Tangent Line

%grid
%legend('y(t)', 'y(t) Fit', 'dy/dt', 'd^2y/dt^2', 'Tangent', 'Location','E')
axis([xlim    min(min(y),intcpt)  ceil(max(y))])
```

Script for Calculating inflection point and drawing tangent to get the precise values of the Lag time and Delay time.

THANKYOU