International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 5 Issue 5, May-2016

Comparison of Common PID Tuning Methods for Higher Order Plants

Ms. Sami Jan Lolu[1], Mr. Zahoor Ahmad Ganie[2],
Mr. Rayees Ahmad Lone[3], Gayas ud din Lolu[4]

[1,2,3]Assistant Professor Department of Electrical Engineering, Islamic University of Science and Technology, Awantipora, Pulwama, Jammu & Kashmir, India
[4]B.Tech CSE student, Islamic University of Science & Technology, Awantipora

ABSTRACT

Controlling the process is the main issue that rises in the process industry. It is very important to keep the process working probably and safely in the industry, for environmental issues and for the quality of the product being processed. PID control is a control strategy that has been successfully used over many years. Simplicity, robustness, a wide range of applicability and near-optimal performance are some of the reasons that have made PID control so popular in the academic and industry sectors. Recently, it has been noticed that PID controllers are often poorly tuned and some efforts have been made to systematically resolve this matter. In the paper a brief summary of PID theory is given, and then some of the most used PID tuning methods are discussed using five different higher order plants.

Keywords: PID Controller; Tuning Methods; Performance indices.

I: INTRODUCTION

Controlling the process is the main issue that rises in the process industry. It is very important to keep the process working probably and safely in the industry, for environmental issues and for the quality of the product being processed. The control problems leak out into the most areas of human operations. It includes machine control and business control operations as well. In the areas of technical relevance the concept of PID controller (Proportional-Integral-Derivative) is widely known. The tuning process of PID controller can be solved by experimental way (the expert is needed) or determined as optimization task. An optimal PID control design includes more goals of optimal regulation process which are often contradictory. Optimal setting of PID controller is generally a task of nonlinear mathematical optimization which is furthermore done on top of dynamic system.

The Proportional- Integral- Derivative (PID) controller is widely used in the process industries. The main reason is their simple structure, which can be easily understood and implemented in practice. Finding design methods that lead to the optimal operation of PID controllers is therefore of significant interest. A PID controller produces an output signal consisting of three terms – one proportional to error signal, another one proportional to integral of error signal and third one proportional to derivative of error signal. The combination of proportional control action, integral control action and derivative control action is called PID control action. The combined action has the advantage of each of the three individual control actions. The proportional controller stabilizes the gain but produces a steady-state error. The integral controller reduces or eliminates the steady-state error. The derivative controller reduces the rate of change error. The main advantages

of PID controllers are higher of stability, no offset and reduced overshoot. PID controllers are commonly used in process control industries. Fig 1 shows the block diagram of PID control system.[1]

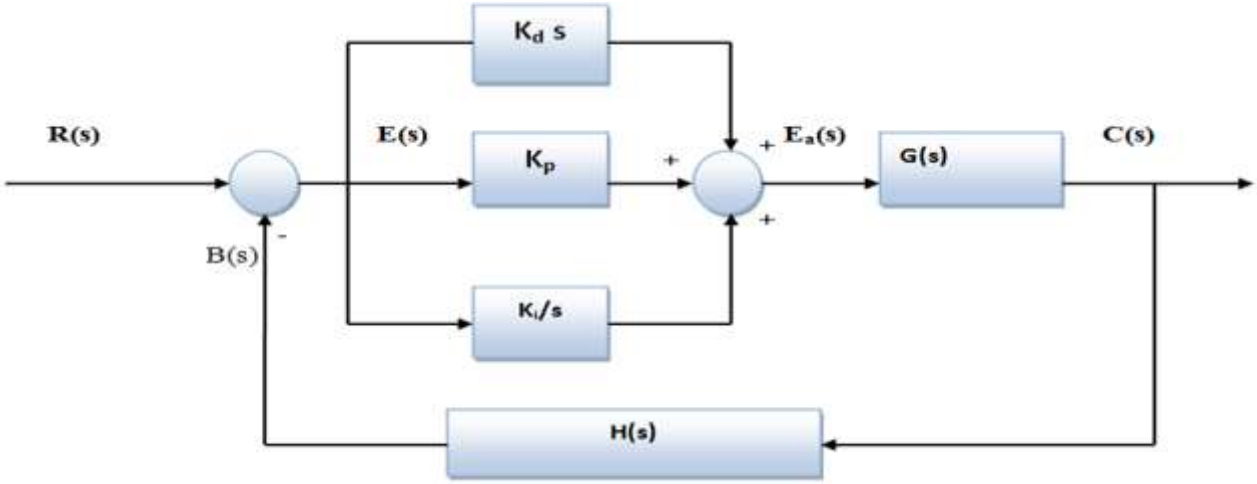


Fig.1: Block diagram of a proportional (PID) control system.

The actuating signal or output signal from a PID controller in time domain is given by

$$E_a(t) = K_p * e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt} \quad (1.1)$$

Taking Laplace transform, we get

$$E_a(s) = \left(K_p + \frac{K_i}{s} + K_d\, s\right) E(s) \quad (1.2)$$

II: CONTROLLER DESIGN

The theory of control deals with methods which lead to change of behavior of controlled dynamic system. The desired output of a system is called the reference or set point. When one or more outputs of the system need to follow a certain reference over time then a controller modifies the inputs of system to obtain the desired value on the output of the system, as shown in fig 2.ThePID controller has three separate constant parameters: Proportional (P)Integral (I) and Derivative (D). It can be said the P depends on present error, I on accumulation of past errors and D is prediction of future errors based on rate of change. The PID controller calculates an error value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the control error by adjusting the process controller outputs. After corrective action from the controller the system should reach point of stability as the result. Stability means the set point is being held on the output without oscillating around it.

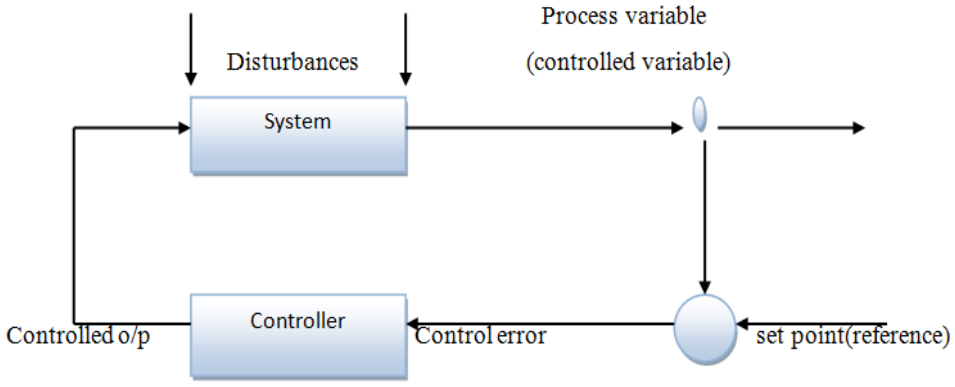


Fig.2: The general concept of the negative feedback loop to control the dynamic behavior of the system with description of the major parts.

The controller in control system is given by fig 2 is the classical one degree-of-freedom (1–DOF) controller. Basic block diagram of standard PID controller is based on parallel circuit, fig.3. The proportional, integral and derivative terms are summed to calculate the output of the PID controller. Defining u(t) as the controller output, the general ideal form of the PID algorithm is:

$$u(t) = K_p * e(t) + \frac{1}{T_i}\int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt} \quad (1.3)$$

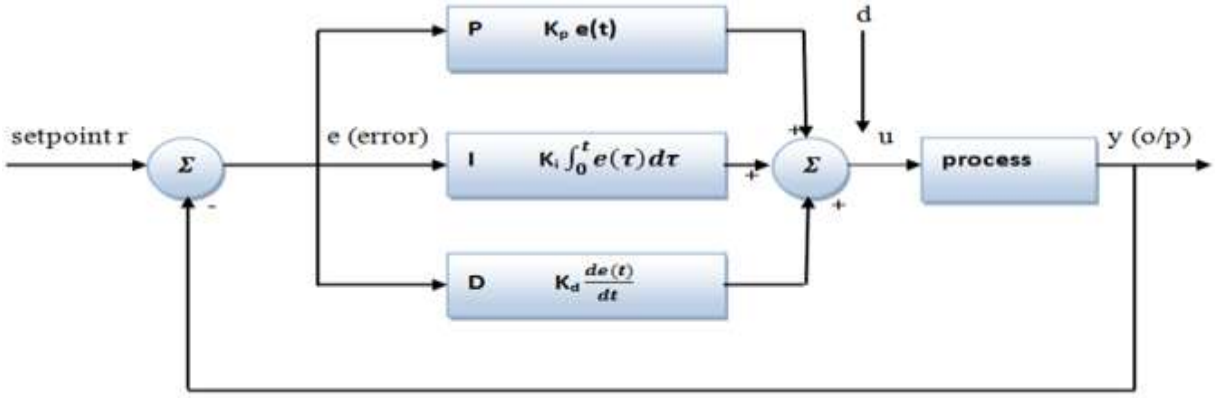


Fig. 3: The block diagram of PID controller.

Where K_p is the single gain, $K_i = \frac{K_p}{T_i}$, $K_d = K_p * T_d$, T_i is the integral time constant and $T_d$is the derivative time constant. In many cases are used the variants of PID controller given in standard form by equation (1.4).
Mutual conversion of controller's constants is obvious.

$$u(t) = K_p [\, e(t)] + K_i \int_0^t e(\tau)dt + K_d \frac{d\, e(t)}{dt} \quad (1.4)$$

Here the parameters have a clear physical meaning. In particular, the inner summation produces a new single error value which is compensated for future and past errors. The addition of proportional and derivative components effectively predicts the error value at T_d seconds in the future, assuming that the loop control remains unchanged. The integral component adjusts the error value to compensate for sum of all past errors, with the intention of completely eliminating them in T_i seconds. The resulting compensated single error value is scaled by the single gainK_p. [1].

Using Laplace transformation the transfer function of PID controller looks like

$$G_c(s) = P + I + D = K_p + \frac{Ki}{s} + K_d\, s \quad (1.5)$$

$$G_c(s) = K_p\, (1 + \frac{1}{T_i\, S} + T_d s\,) \quad (1.6)$$

Time Response Specifications:

The various time response specifications include:

1. **Delay time T_d**: It is the time required for the response to reach 50% of the final value in the first attempt.
2. **Rise time T_r**: It is the time required for the response to rise from 10% to 90% of the final value for over damped systems and 0 to 100% of final value for underdamped systems. The rise time is reciprocal of the slope of the response at the instant; the response is equal to 50% of the final value.
3. **Peak time T_p**: It is the time required for the response to reach it's peak value. It is also defined as the time at which response undergoes the first overshoot which is always peak overshoot.
4. **Peak overshoot M_p**: It is the largest error between reference input and output during the transient period. It can also be defined as the amount by which output overshoots its reference steady state value during the first overshoot.
5. **Settling time T_s:** This is defined as the time required for the response to decrease and stay within specified percentage of its final value.

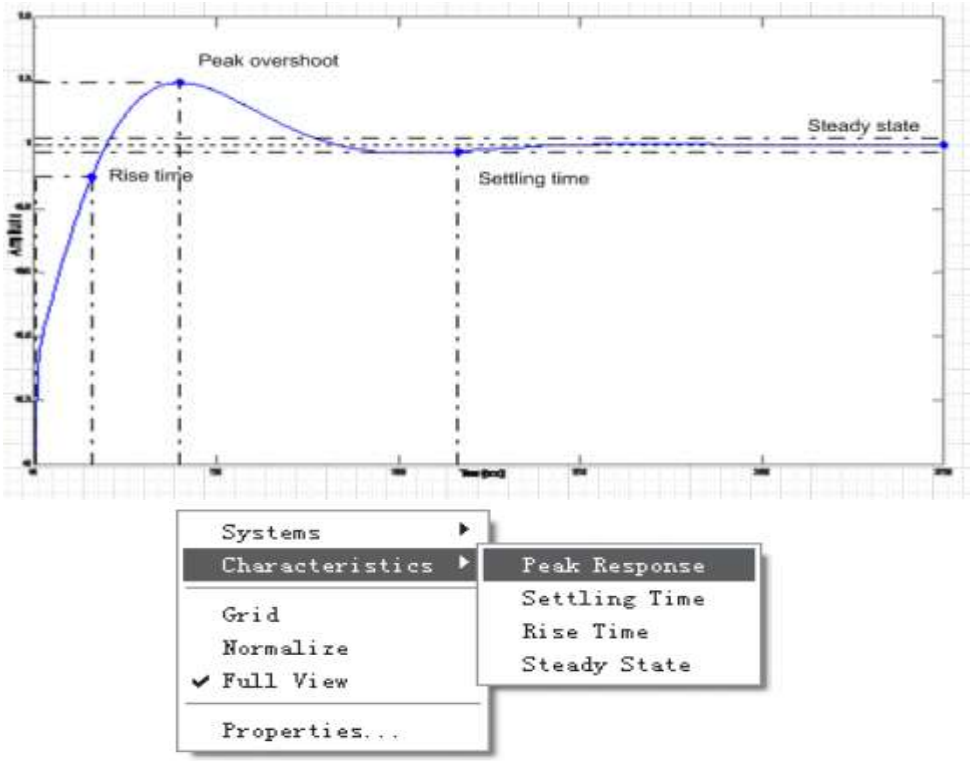


Fig.4 Popup menu with specifications

Tuning Methods of PID Controller:

The different PID parameter tuning methods include [1]:

i. **The Ziegler-Nichols step response method:** The Ziegler-Nichols step response method is an experimental tuning method for determining values of the proportional gain K_P, integral time T_i and derivative time T_d based on the transient response characteristics of a given plants. The first step in this method is to calculate two parameters T (time constant) and L (delay time) that characterize the plant. These two parameters (T, L) can be determined graphically from a measurement of the step response of the plant as illustrated in fig 5. First, the point on the step response curve with the maximum slope is determined and the tangent is drawn with the time axis. Once T and L are determined, the PID controller parameters are then given in terms of T and L by the following formulas

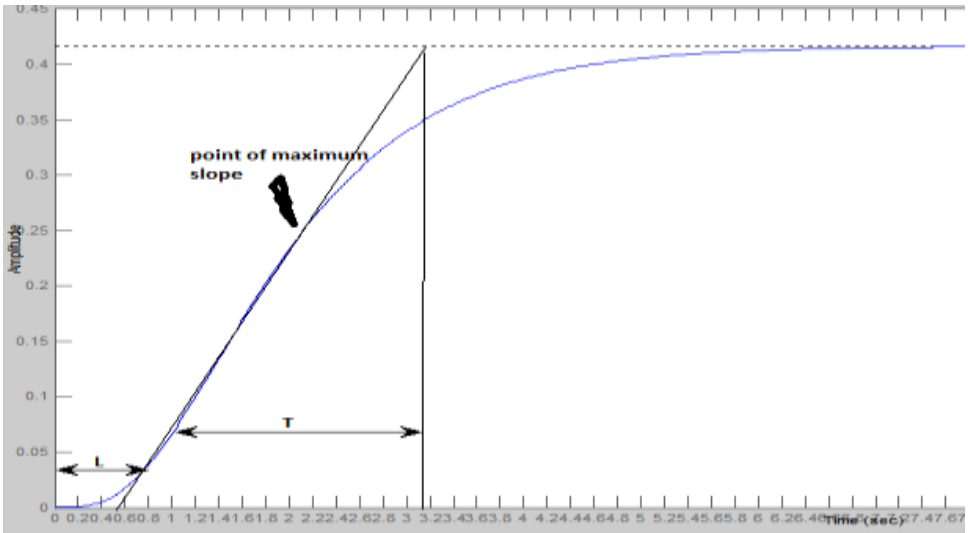


Fig.5: Graphical determination of parameters T and L.

$$K_p = 1.2\frac{T}{L}$$
$$K_i = 2L$$
$$K_d = 0.5L$$

The transfer function C(s)/U(s) may then be approximated by a first-order system with a transport lag as follows:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1}$$

The PID controller tuned by this method gives

$$G_c(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)$$
$$= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right)$$
$$= 0.6T\frac{\left(s+\frac{1}{L}\right)^2}{s}$$

The PID controller has a pole at the origin and double zeros at s = -1/L.

ii. The Ziegler-Nichols frequency-response method:

The Ziegler-Nichols frequency-response method is a closed-loop tuning method. In this method, the two parameters to be calculated are the critical gain K_{cr} and the corresponding period P_{cr} which can be calculated experimentally in the following way:

Set the $T_i = \infty$ and T_d = zero and hence the controller become in the proportional mode only. Close loop system is shown in fig 6. The proportional gain K_p is then increased slowly until a periodic oscillation in the output is observed. This critical value of K_p is called the critical gain K_{cr}. The resulting period of oscillation is referred to as the ultimate period $P_{cr.}$

Based on K_{cr} and P_{cr}, the Ziegler-Nichols frequency response method gives the following simple formulas for setting PID controller parameters according to table [1.1]

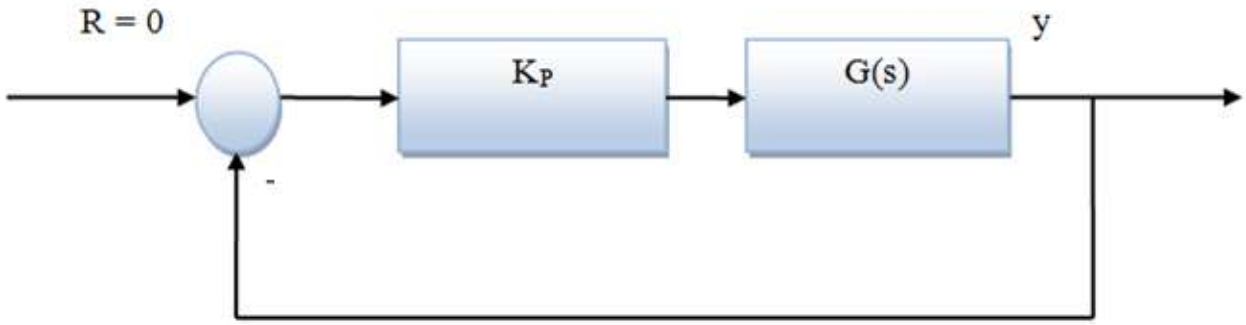


Fig .6: The closed-loop system with the proportional

TABLE.1.1: PID controller parameters

Type of controller	**K_p**	**T_i**	**T_d**
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

The PID controller tuned by this method gives

$$G_c(s) = K_p\left(1 + \frac{1}{T_i S} + T_d s\right)$$
$$= 0.1K_{cr}\left(1 + \frac{1}{0.5P_{cr}\, s} + 0.125P_{cr}\, s\right)$$
$$= 0.075K_{cr}P_{cr}\frac{\left(s+\frac{4}{P_{cr}}\right)^2}{s}$$

The PID controller has a pole at the origin and double zeros at s = - $4/P_{cr}$.[2]

iii. Computational Approach for obtaining Optimal sets of parameter values:

To obtain optimal set of parameter values to satisfy the transient response specification it is desired to obtain a combination of K and a such that the closed-loop system is underdamped and the maximum overshoot in the unit-step response is less than 10%, but more than 5%, to avoid an overdamped or a close-to-overdamped response.[2]

Solving by MATLAB, let us assume the region to search for K and a bounded by

$$2 \leq K \leq 5$$

and

$$0.5 \leq a \leq 1.5$$

Here, the PID Controller is given by

$$G_c(s) = K\frac{(s+a)^2}{s}$$

MATLAB Program

```
t=0:0.01:8;
for K=5:-0.2:2;%Starts the outer loop to vary the K values
for a=1.5:-0.2:0.5;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 t];
den2=[p q r s];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.1 & m>1.05;
plot(t,y);
grid;
title('Unit-Step Response')
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.1 & m>1.05;
break;%Breaks the outer loop
end
end
```

```
sol =

u
v
w
```

SIMULATION RESULTS:

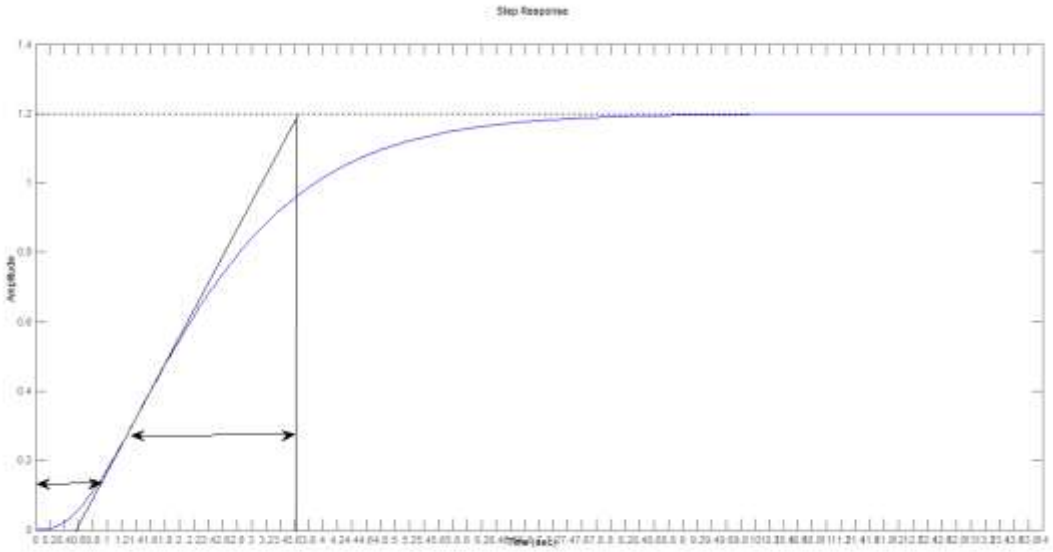
For our test we have taken five higher order plants on which above tuning rules are performed and the simulation results are obtained as under:

1) Plant –I:

$$G_p(s) = \frac{1.2}{0.36s^3 + 1.86s^2 + 2.5s + 1}$$

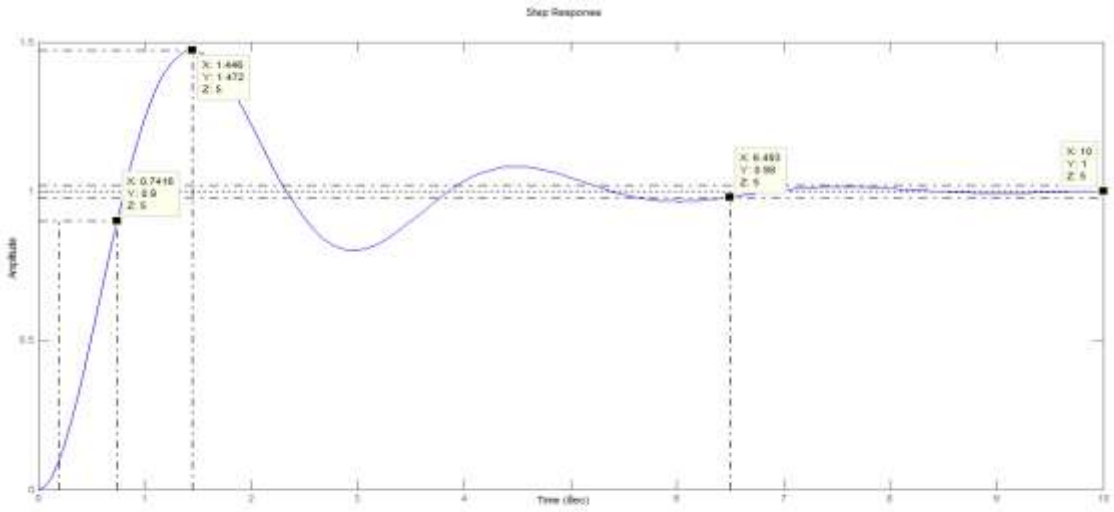
First Method:

Parameter	Value
K_p	6
T_i	1.2
T_d	0.3



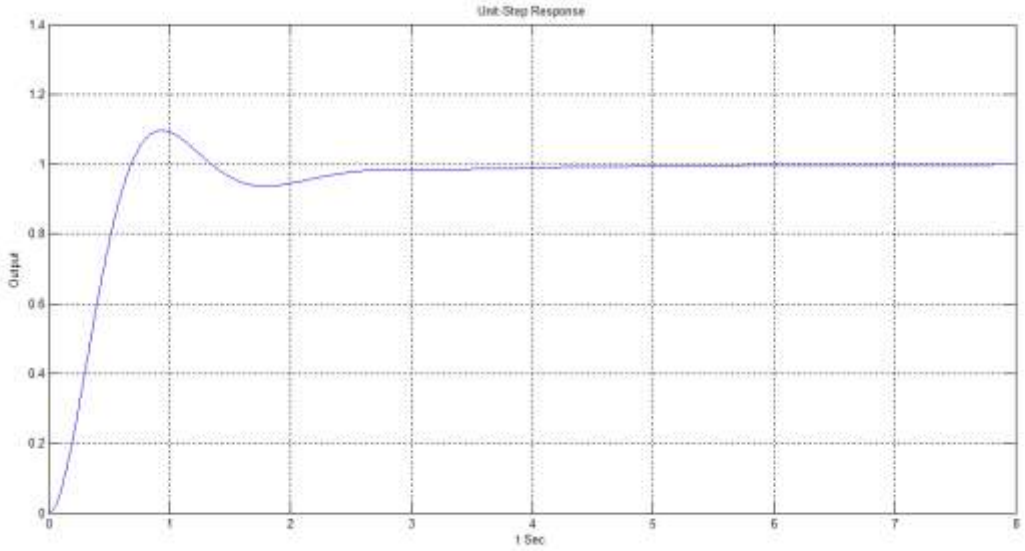
Second Method:

Parameter			Rising time	Peak Amplitude	Overshoot (%)	Time at which Overshoot occurs	Settling time	Steady state
K_p	T_i	T_d						
5.958	1.19	0.2975	0.547	1.47	47.2	1.45	6.49	1



Third Method:

K	a	m
4.2	0.7	1.0962

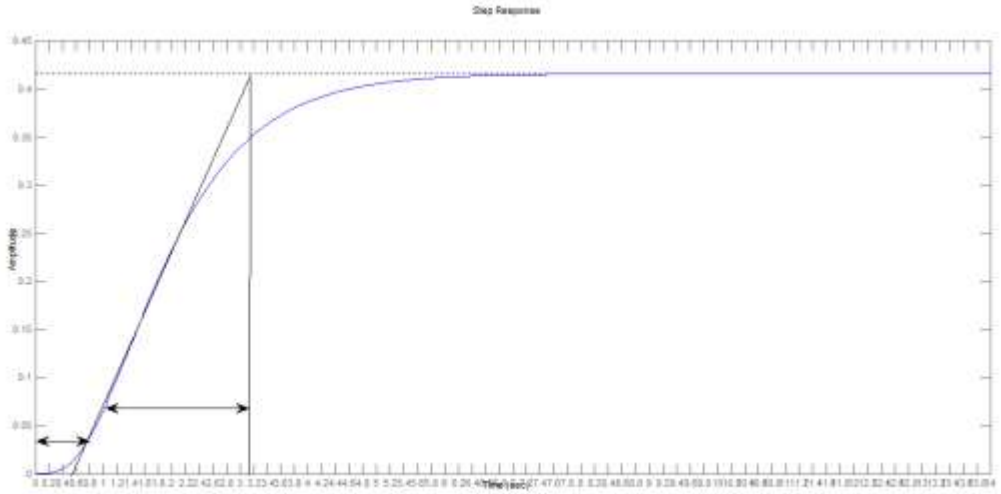


2) Plant – II:

$G_p(s) = \frac{10}{(s+1)(s+2)(s+3)(s+4)}$

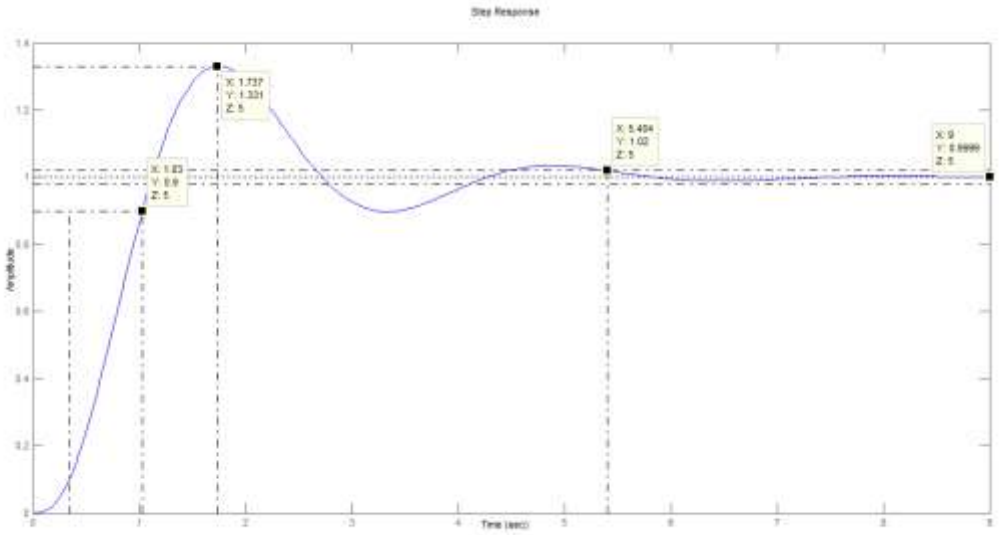
First Method:

Parameter	Value
K_p	5.67
T_i	1.1
T_d	0.275



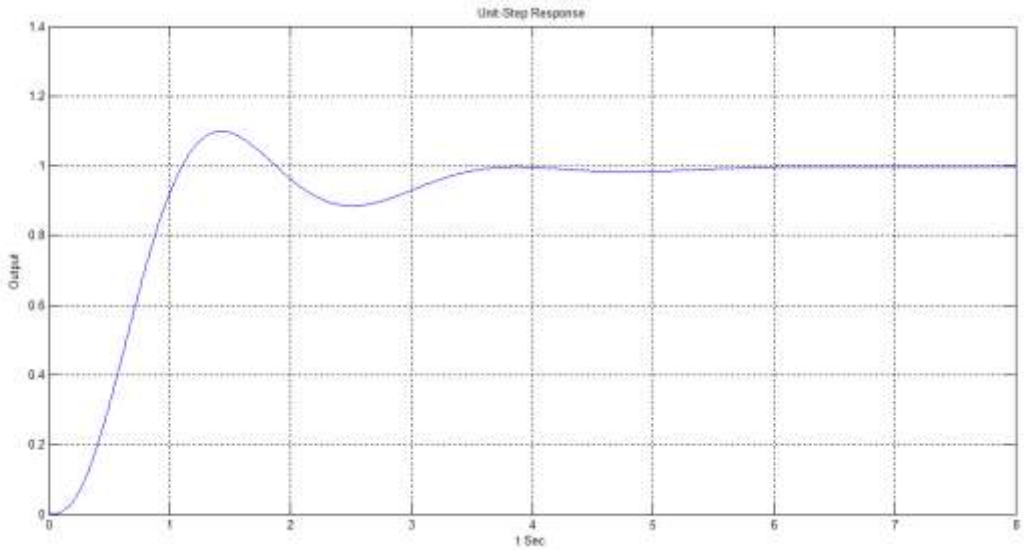
Second Method:

Parameter			Rising time	Peak Amplitude	Overshoot (%)	Time at which Overshoot occurs	Settling time	Steady state
K_p	T_i	T_d						
7.56	1.405	0.3512	0.687	1.33	33.1	1.74	5.4	1



Third Method:

K	a	m
4	0.9	1.0985

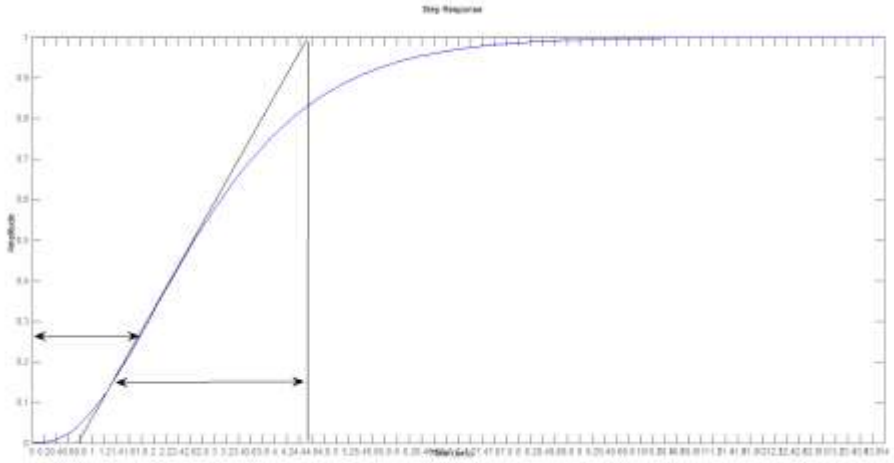


3) Plant – III:

$G_p(s) = \frac{1}{(s+1)^3}$

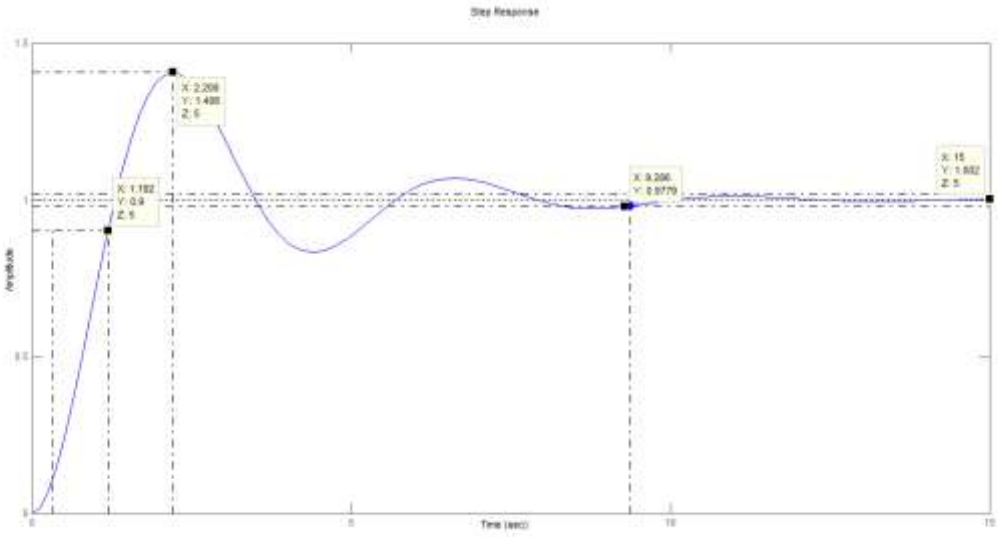
First Method:

Parameter	Value
K_p	6
T_i	1.5
T_d	0.375



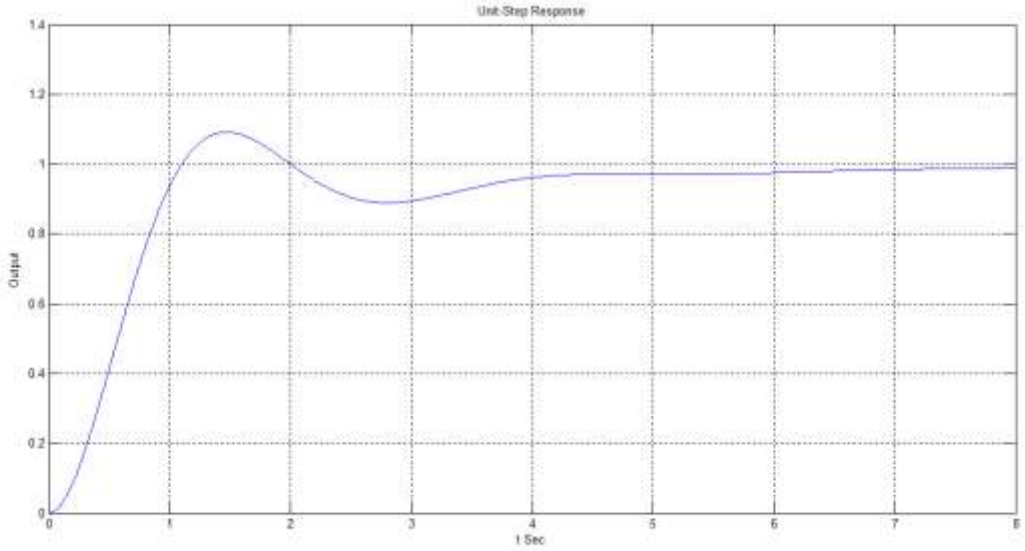
Second Method:

Parameter			Rising time	Peak Amplitude	Overshoot (%)	Time at which Overshoot occurs	Settling time	Steady state
K_p	T_i	T_d						
4.8	1.814	0.453	0.878	1.41	40.6	2.21	9.38	1



Third Method:

K	a	m
5	0.5	1.0915

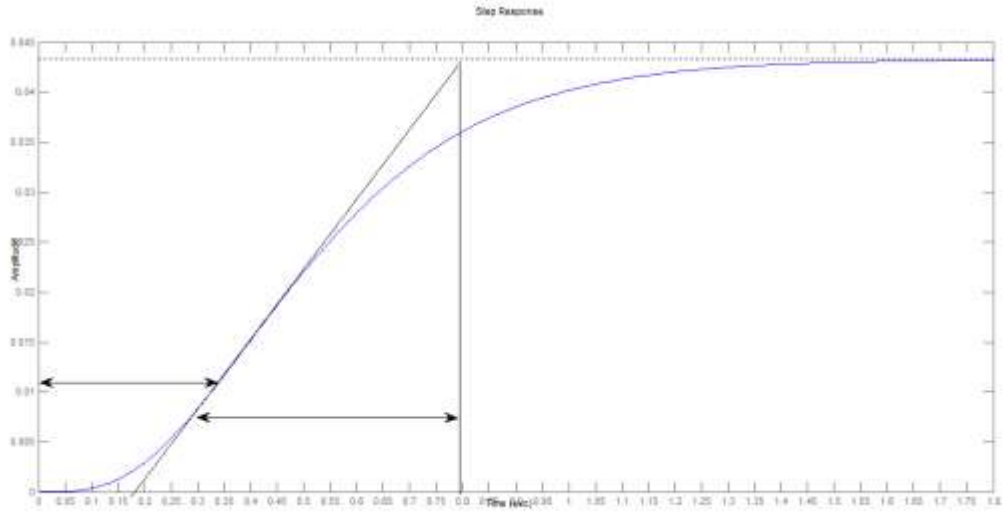


4) Plant – IV:

$$G_p(s) = \frac{150}{s^4 + 32s^3 + 374s^2 + 1888s + 3465}$$

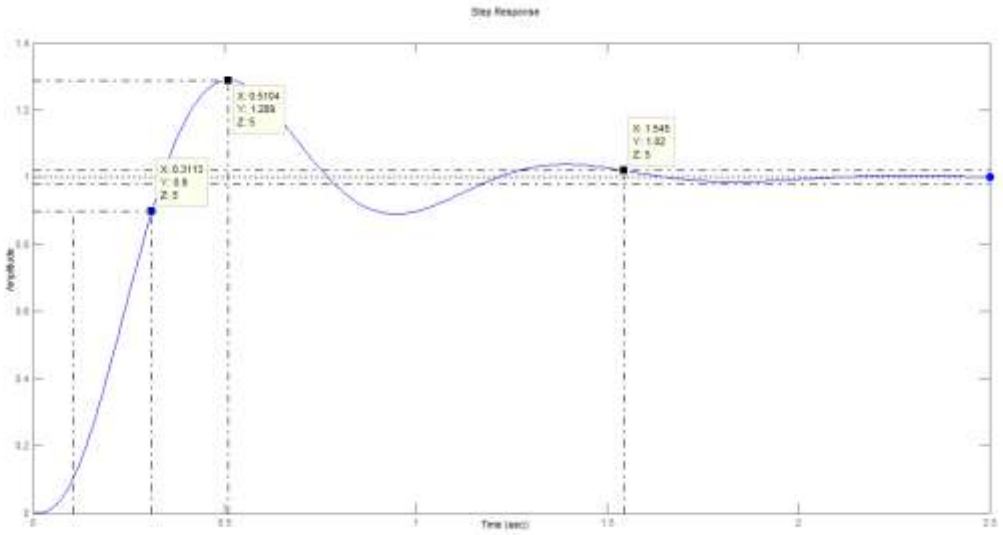
First Method:

Parameter	Value
K_p	4.07
T_i	0.36
T_d	0.09



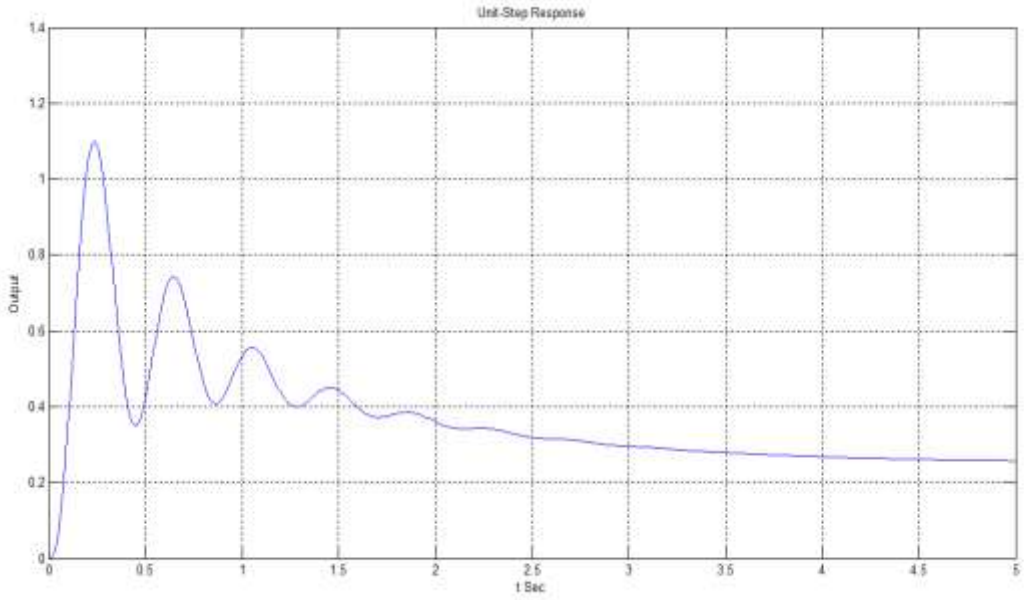
Second Method:

Parameter			Rising time	Peak Amplitude	Overshoot (%)	Time at which Overshoot occurs	Settling time	Steady state
K_p	T_i	T_d						
60.48	0.409	0.102	0.207	1.29	28.9	0.51	1.55	1



Third Method:

K	a	m
32	0.1	1.0973

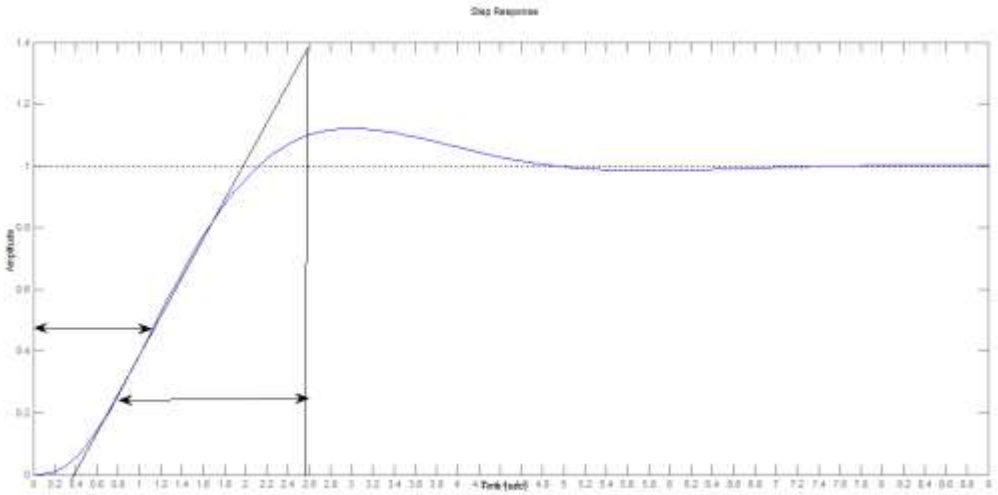


5) Plant – V:

$$G_p(s) = \frac{10}{s^3 + 7s^2 + 10s + 10}$$

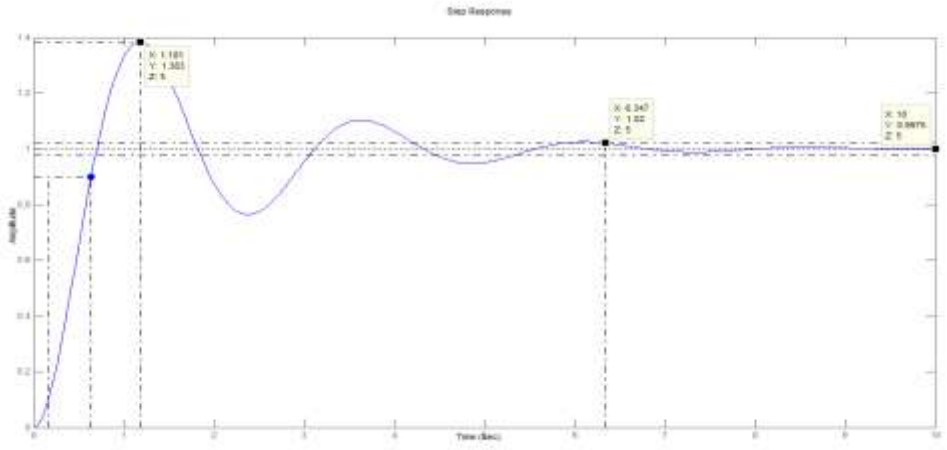
First Method:

Parameter	Value
K_p	6.45
T_i	0.8
T_d	0.2



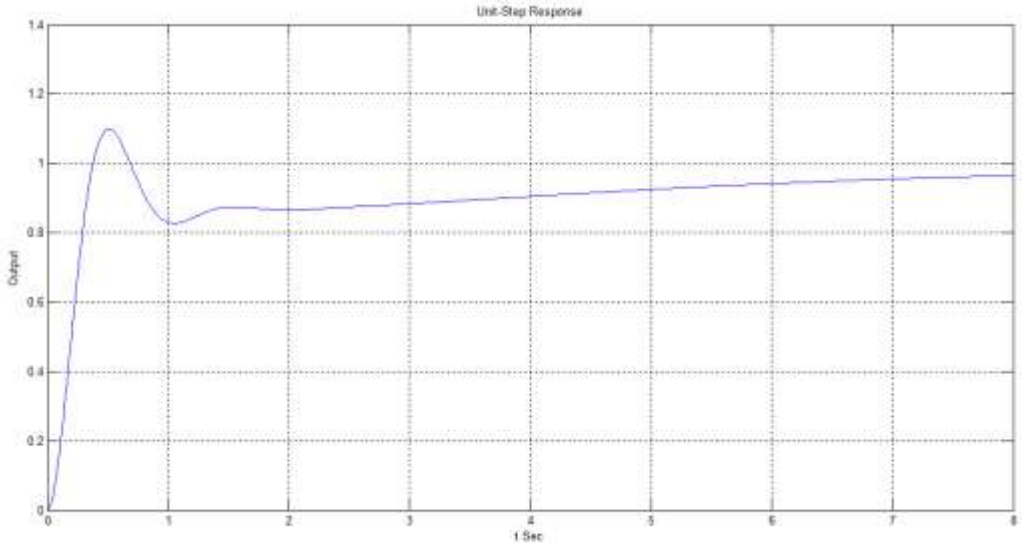
Second Method:

Parameter			Rising time	Peak Amplitude	Overshoot (%)	Time at which Overshoot occurs	Settling time	Steady state
K_p	T_i	T_d						
3.6	0.9935	0.25	0.474	1.38	38.3	1.18	6.35	1



Third Method:

K	a	m
4	0.5	1.0991



III: CONCLUSION

This paper has presented an overview of PID control and three tuning methods are compared here on five different higher order plants. However, it must also be pointed out that PID control may not be sufficient for some cases, for example, processes with more than one oscillatory mode or processes with large time delays or with complex disturbance behavior. It is concluded here that PID control is still of great interest, and is a promising control strategy that deserves further research and investigation. Both industry and academia have a lot to gain from this.

APPENDIX

MATLAB Codes of five plants:

1. Plant-1:

PID first method code:

```
numgp=[0 0 0 1.2];
dengp=[0.36 1.86 2.5 1];
G=tf(numgp,dengp)
Transfer function:
          1.2
-------------------------------
0.36 s^3 + 1.86 s^2 + 2.5 s + 1
k=dcgain(G)
k =

   1.2000
```

PID second method code:

```
numgff=[0 0 2.124 7.14 6];
dengff=[0.36 1.86 2.5 1 0];
gff=tf(numgff,dengff)
```

```
Transfer function:
    2.124 s^2 + 7.14 s + 6
--------------------------------
0.36 s^4 + 1.86 s^3 + 2.5 s^2 + s

T=feedback(gff,1)

Transfer function:
         2.124 s^2 + 7.14 s + 6
-----------------------------------------------
0.36 s^4 + 1.86 s^3 + 4.624 s^2 + 8.14 s + 6

step(T)
```

PID third method code:

```
t=0:0.01:8;
for K=5:-0.2:2;%Starts the outer loop to vary the K values
for a=1.5:-0.2:0.5;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 1.2];
den2=[0.36 1.86 2.5 1];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.1 & m>1.05;
plot(t,y);
grid;
title('Unit-Step Response')
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.1 & m>1.05;
break;%Breaks the outer loop
end
end

sol =

   4.2000
   0.7000
   1.0962
```

2. Plant-2:

PID first method code:

```
s=tf('s');
G=10/(s+1)/(s+2)/(s+3)/(s+4);
step(G);
k=dcgain(G)
```

```
k =

    0.4167
```

PID second method code:

```
numgff=[0 0 0 26.6 75.6 53.81];
dengff=[1 10 35 50 24 0];
gff=tf(numgff,dengff)

Transfer function:
      26.6 s^2 + 75.6 s + 53.81
-------------------------------------
s^5 + 10 s^4 + 35 s^3 + 50 s^2 + 24 s

T=feedback(gff,1)


Transfer function:
          26.6 s^2 + 75.6 s + 53.81
-----------------------------------------------------
s^5 + 10 s^4 + 35 s^3 + 76.6 s^2 + 99.6 s + 53.81

step(T)
```

PID third method code:

```
t=0:0.01:8;
for K=5:-0.2:2;%Starts the outer loop to vary the K values
for a=1.5:-0.2:0.5;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 0 10];
den2=[1 10 35 50 24];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.1 & m>1.05;
plot(t,y);
grid;
title('Unit-Step Response')
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.1 & m>1.05;
break;%Breaks the outer loop
end
end

sol =

    4.0000
```

0.9000
1.0985

3. Plant-3:

PID first method code:

```
numg=[0 0 0 1];
deng=[1 3 3 1];
G=tf(numg,deng)
Transfer function:
         1
---------------------
s^3 + 3 s^2 + 3 s + 1
k=dcgain(G)

k =

     1
step(G)
```

PID second method code:

```
numgff=[0 0 2.175 4.8 2.6466];
dengff=[1 3 3 1 0];
gff=tf(numgff,dengff)

Transfer function:
2.175 s^2 + 4.8 s + 2.647
-------------------------
s^4 + 3 s^3 + 3 s^2 + s

 T=feedback(gff,1)

Transfer function:
      2.175 s^2 + 4.8 s + 2.647
---------------------------------------
s^4 + 3 s^3 + 5.175 s^2 + 5.8 s + 2.647

step(T)
```

PID third method code:

```
t=0:0.01:8;
for K=5:-0.2:2;%Starts the outer loop to vary the K values
for a=1.5:-0.2:0.5;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 1];
den2=[1 3 3 1];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.1 & m>1.05;
plot(t,y);
grid;
title('Unit-Step Response')
```

```
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.1 & m>1.05;
break;%Breaks the outer loop
end
end
```

```
sol =

    5.0000
    0.5000
    1.0915
```

4. Plant-4:

PID first method code:

```
s=tf('s');
G=150/(s+5)/(s+7)/(s+9)/(s+11)
Transfer function:
              150
-------------------------------------
s^4 + 32 s^3 + 374 s^2 + 1888 s + 3465
step(G);
k=dcgain(G)
```

```
k =

    0.0433
```

PID second method code:

```
numgff=[0 0 0 927.6 9070.5 22168.5];
dengff=[1 32 374 1888 3465 0];
gff=tf(numgff,dengff)
```

```
Transfer function:
     927.6 s^2 + 9071 s + 2.217e004
-----------------------------------------
s^5 + 32 s^4 + 374 s^3 + 1888 s^2 + 3465 s
```

```
T=feedback(gff,1)
```

```
Transfer function:
            927.6 s^2 + 9071 s + 2.217e004
-------------------------------------------------------------
s^5 + 32 s^4 + 374 s^3 + 2816 s^2 + 1.254e004 s + 2.217e004
```

```
step(T)
```

PID third method code:

```
t=0:0.01:5;
for K=50:-1:2;%Starts the outer loop to vary the K values
for a=2:-0.05:0.05;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
```

```
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 0 150];
den2=[1 32 374 1888 3465];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.10 & m>1.02;
plot(t,y);
grid;
title('Unit-Step Response')
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.10 & m>1.02;
break;%Breaks the outer loop
end
end

sol =

   32.0000
0.1000
1.0973
```

5. Plant-5:

PID first method code:

```
numgp=[0 0 0 10];
dengp=[1 7 10 10];
G=tf(numgp,dengp)

Transfer function:
         10
-----------------------
s^3 + 7 s^2 + 10 s + 10
step(G)
k=dcgain(G)

k =

    1
```

PID second method code:

```
numgff=[0 0 8.94 35.94 36.12];
dengff=[1 7 10 10 0];
gff=tf(numgff,dengff)

Transfer function:
```

```
8.94 s^2 + 35.94 s + 36.12
--------------------------
s^4 + 7 s^3 + 10 s^2 + 10 s

T=feedback(gff,1)

Transfer function:
      8.94 s^2 + 35.94 s + 36.12
---------------------------------------
s^4 + 7 s^3 + 18.94 s^2 + 45.94 s + 36.12

step(T)
```

PID third method code:

```
t=0:0.01:8;
for K=5:-0.2:2;%Starts the outer loop to vary the K values
for a=1.5:-0.2:0.5;%Starts the inner loop to vary the a values
num1=K*[1 2*a a^2];
den1=[0 1 0];
tf1=tf(num1,den1);
num2=[0 0 0 10];
den2=[1 7 10 10];
tf2=tf(num2,den2);
tf3=tf1*tf2;
sys=feedback(tf3,1);
y=step(sys,t);
m=max(y);
if m<1.1 & m>1.05;
plot(t,y);
grid;
title('Unit-Step Response')
xlabel('t Sec')
ylabel('Output')
sol=[K;a;m]
break;%Breaks the inner loop
end
end
if m<1.1 & m>1.05;
break;%Breaks the outer loop
end
end

sol =

   4.2000
   0.5000
   1.0991
```

REFERENCES

[1]. Lolu. S.J, Bhatti. S.J, Sharma. R.K, “Approaches for tuning of PID Controller”, International Journal of Enhanced Research in Science Technology & Engineering, Vol-2, Issue 3, March 2013.
[2]. Ogata. K, “Modern Control Engineering”, Prentice-Hall of India,2003.
[3]. Stefani. R.T, Shahian.B, Savant, Hostetter, “Design of feedback control systems, Oxford University Press,2002.

[4]. Ziegler J.G and Nichols N.B., "Optimum Settings for Automatic Controllers", New York, N.Y., December 1-5, 1941, of The American Society Of Mechanical Engineers.
[5]. Skogestad Sigurd, "Simple analytic rules for model reduction and PID controller tuning", Journal of Process Control 13 (2003), 291–309.
[6]. Preeti, Dr. Beniwal Narendra Singh. "Comparison of Conventional and Fuzzy P/PI/PID Controller for Higher Order Non Linear Plant with High Dead Time," International Journal and Scientific Research Publications, vol.2, issue 8, August 2012, ISSN-2250-3153.
[7]. Abe N. and Nobuyama E., "Control of Delay Systems from Fundamentals to Frontiers #1: Introduction of Time Delay Systems-Transfer Function Approach," Journal of the Society of Instrument and Control engineers, Vol. 44, No. 11, pp. 799–804, 2005,(in Japanese).
[8]. Astrom K.J, H Agglund T., "The future of PID control", Control Engineering Practice 9 (2001) 1163–1175.
[9]. Bakshi U.A. and Bakshi V.U., Handbook of Principles of Control Systems, Technical publications Pune, 2009.
[10]. Cominos P. and Munro N., "PID controllers: recent tuning methods and design to specification" IEE, 2002.