Group Members Name	Urwa Maryam, Umar Hayyat, Mateen Abbas
Reg. #	2019-EE-352, 2019-EE-360, 2019-EE-384
Marks	

Lab # 07

PID Controller Tunning

Objective:

The objective of this lab is to get understanding of a PID controller and to design a controller according to the required specifications.

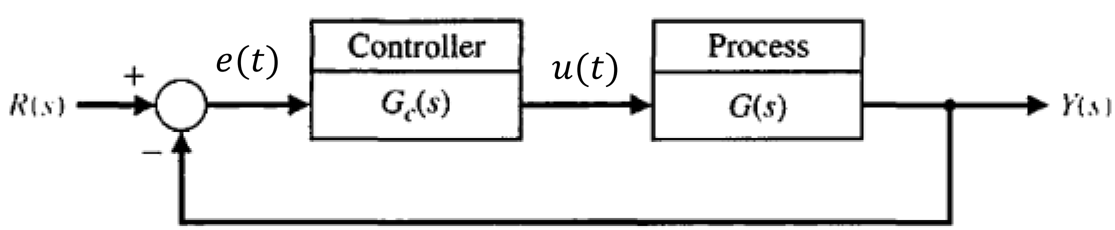
In this session we will discuss the effect of each of the PID parameters on the dynamics of a closed-loop system and will demonstrate how to use a PID controller to improve a system's performance.

Introduction:

Lab Briefing

The PID controller is widely employed because it is very understandable, easily realizable and because it is quite effective. PID controller contains a proportional, an integral, and a derivative term represented by K_P, K_I, and K_D respectively. It is quite sophisticated in that it captures the history of the system (through integration) and anticipates the future behavior of the system (through differentiation).

We will start by considering the following unity-feedback system:



The output of a PID controller, which is equal to the control input to the plant, is calculated in the time domain from the feedback error as follows:

$$u(t) = K_p e(t) + K_I \int e(t)\,dt + K_D \frac{de(t)}{dt}.$$

- **Proportional Term**

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant. The proportional term is given by:

$$P_{out} = K_p e(t)$$

- **Integral Term**

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain $\boldsymbol{Ki}$ and added to the controller output.

$$I_{out} = K_i \int e(\tau) d\tau$$

The integral term accelerates the movement of the process towards set-point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set-point value.

- **Derivative Term**

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d. Derivative action predicts system behavior and thus improves settling time and stability of the system. The derivative term is given by:

$$D_{out} = K_d \frac{de(t)}{dt}$$

PID controller has a transfer function in s-domain:

$$G_c(s) = K_p + \frac{K_I}{s} + K_D s.$$

If we set $K_D = 0$, we will have proportional plus integral (PI) controller

$$G_c(s) = K_p + \frac{K_I}{s}.$$

If we set $K_I = 0$, we will have proportional plus derivative (PD) controller

$$G_c(s) = K_p + K_D s.$$

PID controller transfer function is

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$
$$= \frac{K_D(s^2 + as + b)}{s} = \frac{K_D(s + z_1)(s + z_2)}{s},$$

where $a = K_P/K_D$ and $b = K_I/K_D$.

Therefore, a **PID** controller introduces a transfer function with one pole at the origin and two zeros that can be located anywhere in the s-plane. The closed-loop transfer function is,

$$T(s) = \frac{G(s)G_c(s)}{1 + G(s)G_c(s)}$$

Table 7.6 Effect of Increasing the PID Gains K_p, K_D, and K_I on the Step Response

PID Gain	Percent Overshoot	Settling Time	Steady-State Error
Increasing K_P	Increases	Minimal impact	Decreases
Increasing K_I	Increases	Increases	Zero steady-state error
Increasing K_D	Decreases	Decreases	No impact

To implement the PID controller, three parameters must be determined, the proportional gain, denoted by K_P, integral gain, denoted by K_I and derivative gain denoted by K_D. There are many methods available to determine acceptable values of the PID gains.

Two common methods are Manual PID tuning and Ziegler-Nichols tuning.

Manual PID tuning

In this method, the PID control gains are obtained by trial-and-error with minimal analytic analysis using step responses obtained via simulation, or in some cases, actual testing on the system and deciding on the gains based on observations and experience.

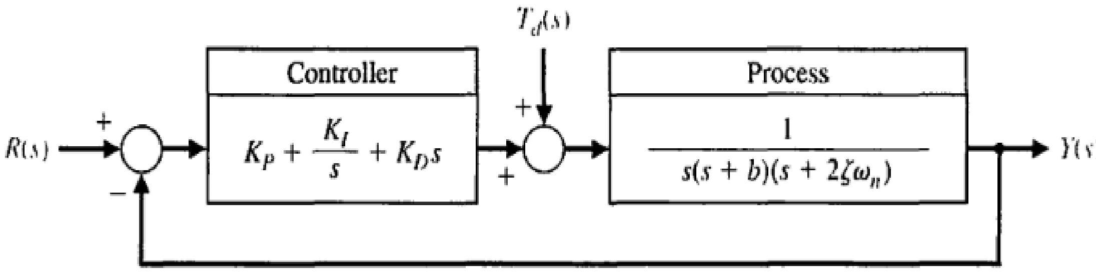
First set $K_I = 0$ and $K_D = 0$. This is followed by slowly increasing the gain Kp until the output of the closed-loop system oscillates just on the edge of instability.

Once the value of K_P is found that brings the closed-loop system to the edge of stability, you reduce the value of gain K_P to achieve what is known as the **quarter amplitude decay.** That is, the amplitude of the closed-loop response is reduced approximately to one-fourth of the maximum value in one oscillatory period. A rule-of-thumb is to start by reducing the proportional gain K_P by one-half.

The next step of the design process is to increase K_I and K_D manually to achieve a desired step response.

Example

Consider a close loop transfer function having a unity feedback



$$G(s) = \frac{1}{s(s+b)(s+2\zeta\omega_n)},$$

Where b = 10, ζ = 0.707, and ω_n = 4

Implement a PID controller for overshoot less than 12% and settling time less than 1sec and rise time less than 0.24sec.

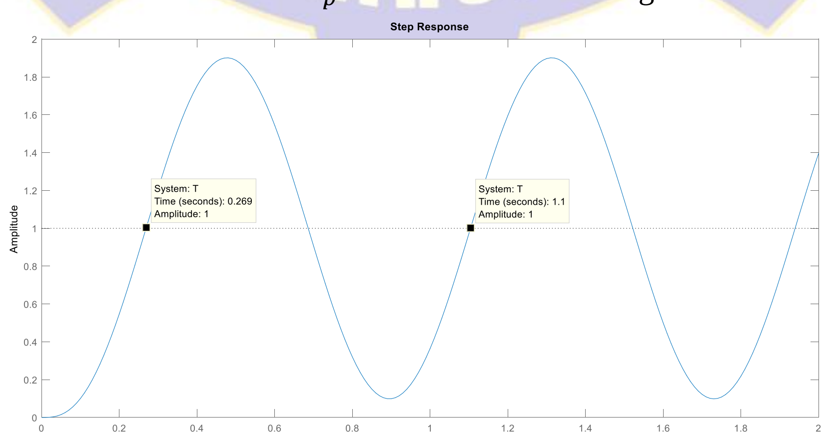
To begin process first set K_I = 0 and K_D = 0. Start increasing the value of K_p until neutral stability (marginal stability) is achieved. This is such a value of gain at which the step response of a feedback system is marginally stable. If you further increase the value of gain, the system response will become unstable. It implies that the step response should be oscillatory in such a way that it should neither decay nor grow.

In MATLAB, write the commands

```
s = tf('s');
zeta = 0.707;
wn = 4;
G = 1/(s*(s+10)*(s+2*zeta*wn));
Kp = 885.5;
C = pid(Kp);
T = feedback(C*G,1)
t = 0:0.01:2;
step(T,t)
```

where C is the controller and G is the system transfer function.

Now plot the step response of the feedback system in MATLAB. You can observe the sustained oscillations so we can call this value of K_p as the critical/ultimate gain.

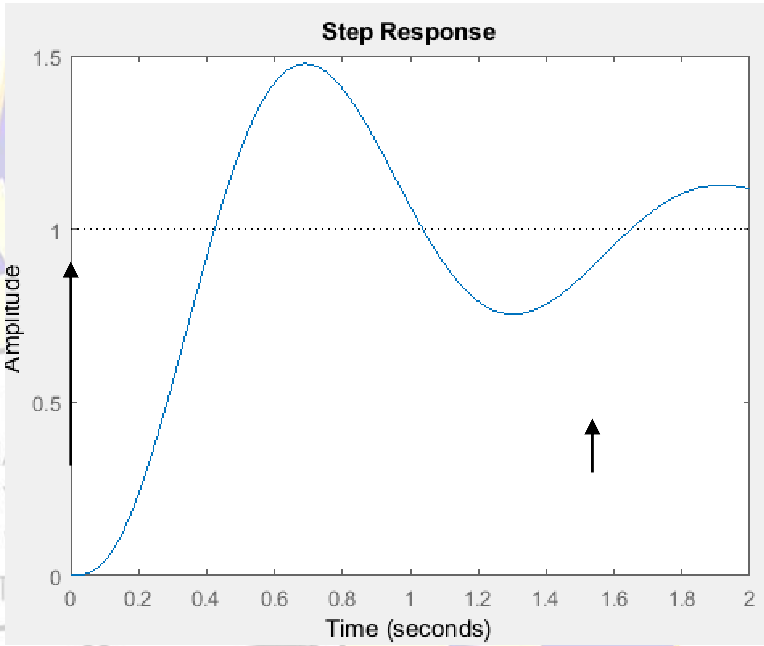


The critical period of oscillation is $T_u = 1.1 - 0.27 = 0.83\ s$. (You can calculate the time period by using data cursors).

Reduce K_P = 885.5 by half, as a first step to achieving a step response with approximately a quarter amplitude decay. Re-define the value of K_P by slowly reducing the value from K_P = 442.75 to K_P = 370.

Plot the step response of the feedback system with K_P = 370. You can observe quarter amplitude decay here.

```
Kp = 370;
C = pid(Kp);
T = feedback(C*G,1)
t = 0:0.01:2;
step(T,t)
```



Right click on the graph and click the characteristics to observe the steady state value, rise time, settling time, peak response to see the overshoot.

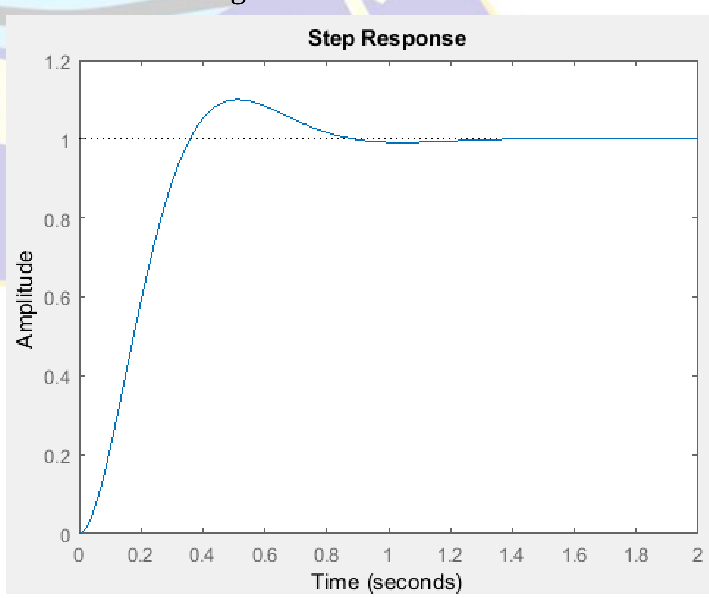
System requirements: overshoot less than 12% and settling time less than 1sec and rise time less than 0.24sec.

Does the above response meet all the system requirements? Comment on it.

No, the above response doesn't meet the requirement. The %OS is 47.7, T_S is 3.33 and T_r is 0.254.

Derivative controller tends to reduce both the overshoot and the settling time. Use Kd to decrease percent overshoot. Set K_P = 370 and K_D= 60. Use the following command for the controller.

```
Kp = 370;
Kd = 60;
C = pid(Kp,0,Kd)
T = feedback(C*G,1)
t = 0:0.01:2;
step(T,t)
```

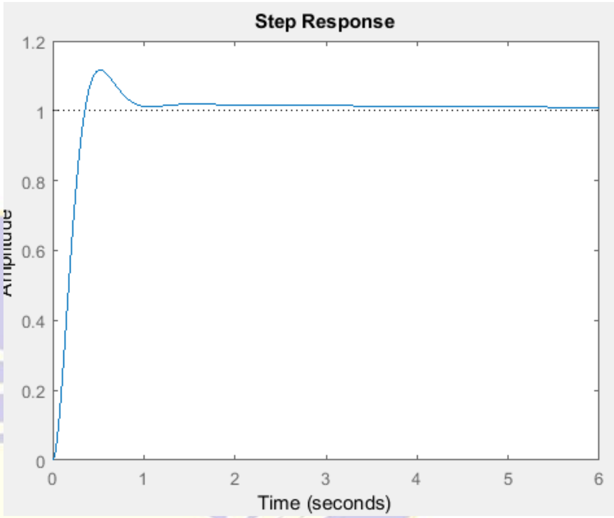


Now the step response of the feedback system looks like as follows.

The above plot shows that the addition of the derivative term reduced both the overshoot and the settling time.

Addition of an integral controller tends to decrease the rise time, but on the other hand it increases both the overshoot and the settling time, and also tends to reduce the steady-state error. Select $Kp = 370$, $Kd = 60$, and $Ki = 50$. Use the following command for the controller.

```
Kp = 370;
Kd = 60;
Ki = 50;
C = pid(Kp,Ki,Kd)
T = feedback(C*G,1)
step(T)
```



We have designed a closed-loop system with PID controller having desired specifications given in the statement. (Right click on the graph and click the characteristics to observe the steady state value, rise time, settling time, peak response to see the overshoot.)

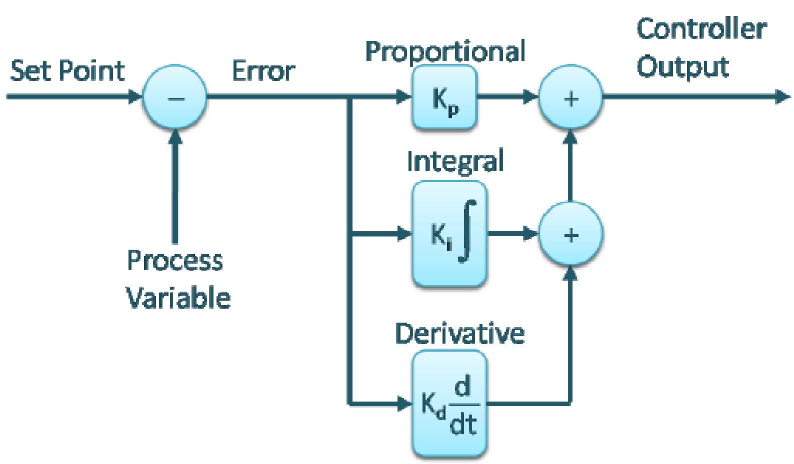
System requirements: overshoot less than 12% and settling time less than 1sec and rise time less than 0.24sec.

Does the above response meet all the system requirements? Comment on it.

Yes, the above response meets all the system requirements. The %OS is 11.7, T_s is 0.903 and T_r is 0.238.

Ziegler-Nichols PID tuning

The goal of Ziegler Nichols PID tuning method is to find the gains K_p, K_I and K_d appropriately. This method helps in finding the starting values for the 3 parameters which can then be changed according to the required design specifications.



The controller output $u(t)$ is given as

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

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

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

where $K_I = K_p / T_i$ and $K_d = K_p T_d$

Finding parameters

1. Start with a smaller value of K_p and set $K_d = K_I = 0$.
2. Increase the value of K_p until neutral stability (marginal stability) is achieved.
3. Record critical/ultimate gain $K_p = K_u$ at marginal stability.

The best method to find a precise value of critical/ultimate gain K_u is to use the Routh-Hurwitz criterion. (If you use Routh-Hurwitz criterion, then skip step 1 & 2).

4. Store the value of a critical gain in a variable *Ku* in MATLAB.
5. Now define the plant transfer function G.
6. Define the controller variable as
   C = pid(Ku)
7. Use feedback command to define the overall transfer function T.
   T = feedback(C*G,1)
8. Plot the step response.
   t = 0:0.01:10;
   step(T,t)

The step response should be oscillatory in such a way that step response should neither decay nor grow (marginal stable).

9. Find critical period of oscillation T_u in seconds (Refer to page 5 about find the critical period). Define and store the variable Tu in MATLAB. *(You may use data cursors to find the period).*
10. Now you have already defined the values of *Ku* and *Tu*. Write the formulas to define the variables K_p, T_i and T_d in MATLAB from the given table below for the desired control type.

Control Type	K_P	T_i	T_d
PID (classic)	$0.6\ K_U$	$T_U/2$	$T_U/8$
P	$0.5\ K_U$	-	-
PI	$0.45\ K_U$	$T_U/1.2$	-
PD	$0.8\ K_U$	-	$T_U/8$
Pessen Integration	$0.7\ K_U$	$2T_U/5$	$3\ T_U/20$
Some Overshoot	$K_U/3$	$T_U/2$	$T_U/3$
No Overshoot	$0.2\ K_U$	$T_U/2$	$T_U/3$

Define the variables, $K_I = K_p/T_i$ and $K_d = K_p T_d$ in the MATLAB.
Use the command to implement the controller and plot the step response.

```
C = pid(Kp,Ki,Kd)
T = feedback(C*G,1)
step(T)
```

11. Right click on the step response and click the characteristics to observe the settling time, peak response to see the percent overshoot.

Task 1: Consider the process/plant transfer function as

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

Design a controller using Ziegler Nichols tuning method. Find corresponding gains meeting the following design specifications,

1) Design the controller parameters using PID (classic) controller and plot the step response. Give the percentage overshoot and the settling time.
2) From the above table, which control type can be used to reduce the percentage overshoot to less than 5%? Plot the step response to verify it.

By Routh-Hurwitz criterion:

S4	1	35	10Ku+24
S3	10	50	0
S2	30	10Ku+24	0
S1	1260-100Ku / 30	0	0
S0	10Ku+24	0	0

(1260-100Ku)/30>0 1260-100Ku>0 -100Ku>-1260	10Ku+24 >0 10Ku>-24 Ku>-2.4
Ku<12.6	

Range of Ku **12.6> Ku>-2.4**

So, **-2.4<Ku<12.6** FOR marginally stable we use Ku=12.6

Part 1: PID Controller

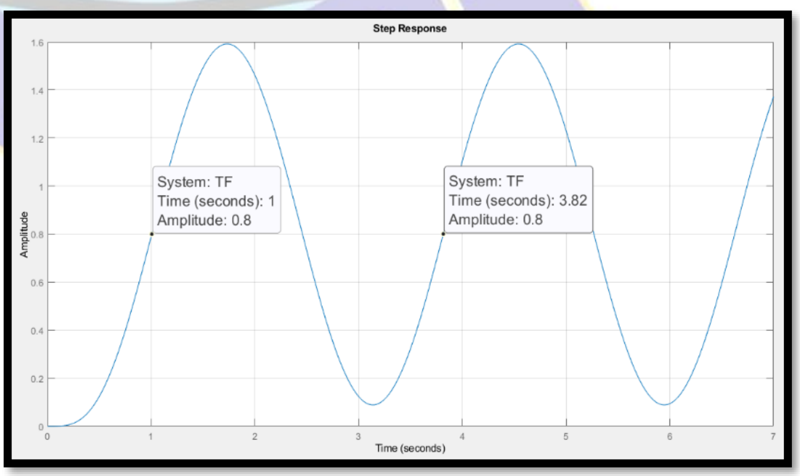
$$G(s) = \frac{10}{(s^2+3s+2)(s^2+7s+12)}$$

$$G(s) = \frac{10}{s^4+10s^3+35s^2+50s+24}$$

Finding Critical period (Tu)

Code:

```
ku=12.6;   %proportional Gain
C=pid(ku);
G=tf([10],[1 10 35 50 24]);
TF=feedback(G*C,1);
t = 0:0.001:10;
step(TF,t);
grid on;
```



Calculation of Tu:

Tu = 3.82 - 1 = 2.82 s

Defining k_p, k_I, k_d

proportional Gain :

Kp=Ku(0.6) kp=7.56

integral constant :

KI=Kp/Ti ∴Ti=Tu/2 = 2.82/2 = **1.41s**

ki= 7.56/1.41= **5.362**

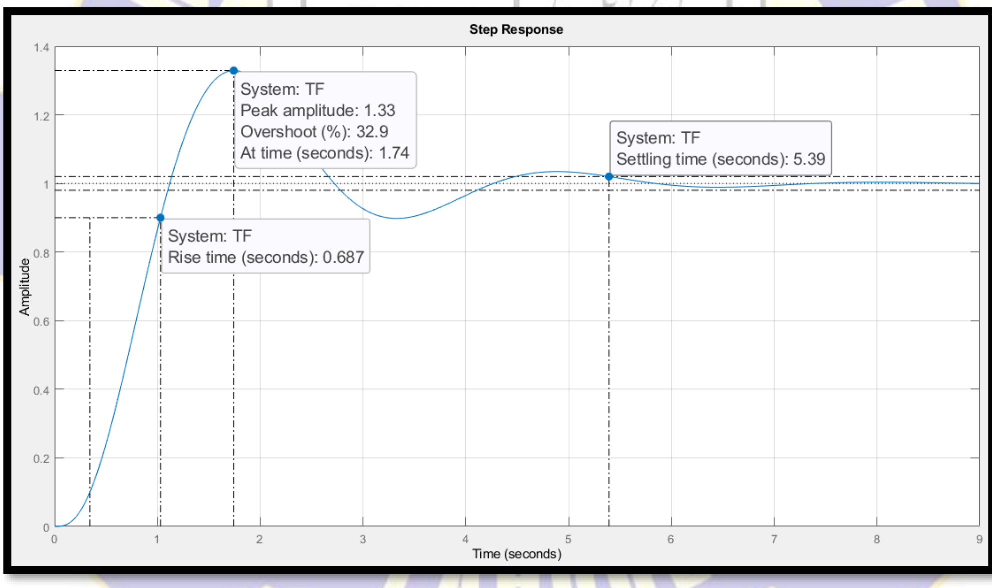
derivative const:

Kd=KpTd ∴Td=Tu/8 = 2.82/8 = **0.35s**

kd= 7.56(0.35125) = **2.6649**

Code:

```
clc; clear all; close all;
kp=7.56;            %proportional Gain kp = 0.6ku (put ku value find previously)
ki=5.362;           %integral constant ki = kp/Ti  where Ti = Tu/2 (Tu = 2.82)
kd=2.6649;          %derivative constant kd = kp*Td  where Td = Tu/8
C=pid(kp,ki,kd);
G=tf([10],[1 10 35 50 24]);
TF=feedback(G*C,1);
step(TF);
grid on;
```



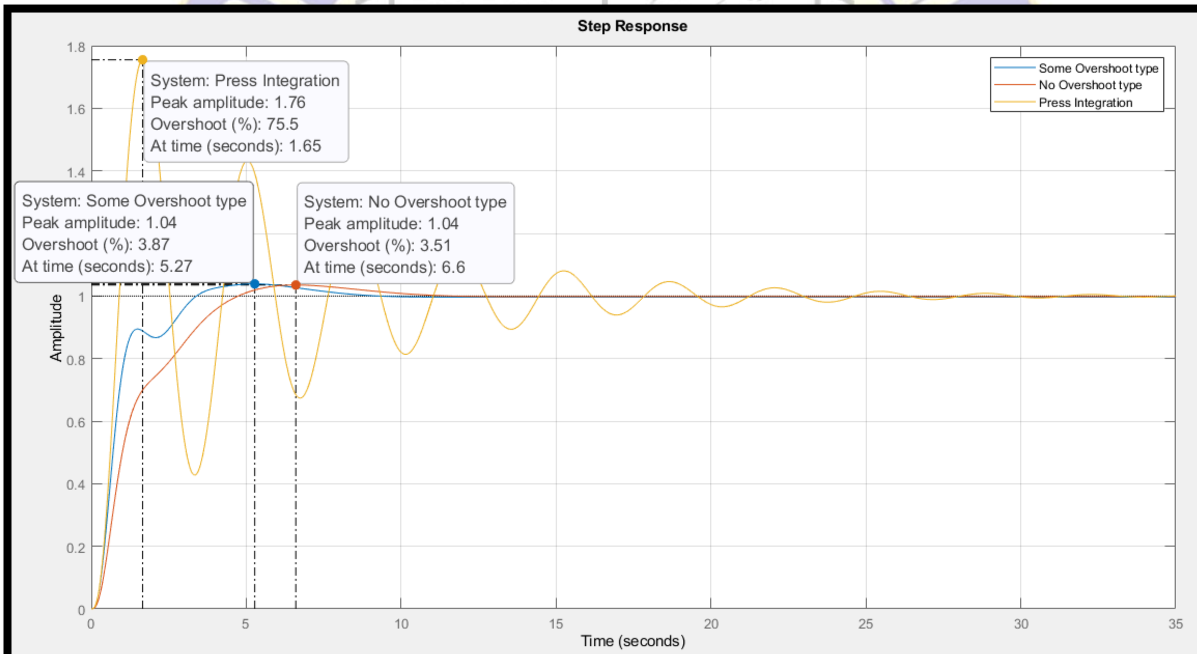
Part 2: %OS < 5%

Code: (critical period will be the same as in PID controller)

```
clc; clear all; close all;
%using the some overshoot controller type
kp=4.2;     %proportional Gain    kp = ku/3  (ku = 12.6)
ki=2.9787;       %integral constant     ki = kp/Ti where Ti = Tu/2   (Tu = 2.82)
kd=3.948;          %derivative constant kd = kp*Td where Td = Tu/3
C=pid(kp,ki,kd);
G=tf([10],[1 10 35 50 24]);
F=feedback(G*C,1);
step(F);
grid on;
hold on;
```

```
%using the No overshoot controller type
kp=2.52;         %proportional Gain    kp = 0.2*ku  (ku = 12.6)
ki=1.787;        %integral constant     ki = kp/Ti where Ti = Tu/2   (Tu = 2.82)
kd=2.3688;           %derivative constant kd = kp*Td where Td = Tu/3
C=pid(kp,ki,kd);
F=feedback(G*C,1);
step(F);
grid on;
hold on;

%using the Press integraton controller type
kp=8.82;         %proportional Gain    kp = 0.7*ku  (ku = 12.6)
ki=15.368;        %integral constant     ki = kp/Ti where Ti = 2*Tu/5   (Tu =
2.82)
kd=3.73086;           %derivative constant kd = kp*Td where Td = 3*Tu/20
C=pid(kp,ki,kd);
F=feedback(G*C,1);
step(F);
grid on;
legend('Some Overshoot type','No Overshoot type','Press Integration')
```



From the step response plot above of various controller type, No Overshoot and Some Overshoot type controller can be used to reduce the %OS less than 5.

Task 2: Consider the process/plant transfer function as

$$G(s) = \frac{1}{(s+1)(s+2)(s+3)}$$

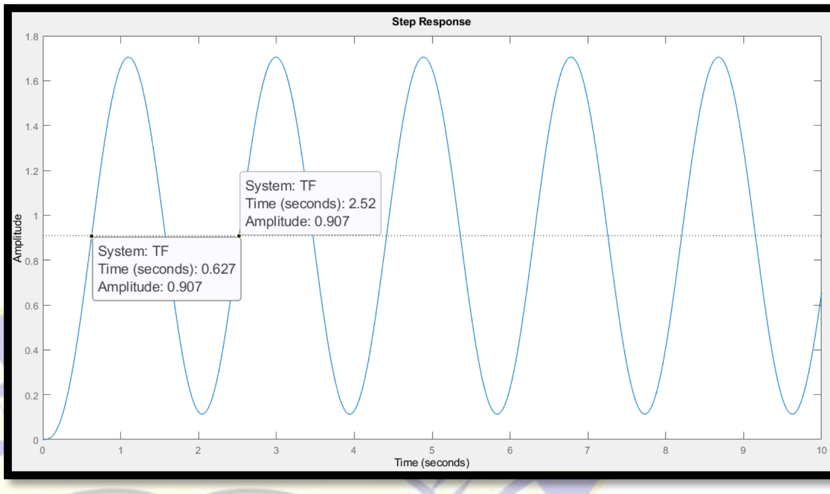
Design a controller using Ziegler Nichols tuning method. Plot the step response using

1) PI controller. Specify the percentage overshoot and the settling time.
2) PID controller [%OS < 15 & Ts < 5s (Both specifications should be met)]

Part 1: PI controller

$$G(s) = \frac{1}{(s^2 + 3s + 2)(s + 3)}$$

$$G(s) = \frac{1}{s^3 + 6s^2 + 11s + 6}$$



Finding Critical period (Tu)

Code:

```
ku=60;   %proportional Gain
C=pid(ku);
G=tf([1],[1 6 11 6]);
TF=feedback(G*C,1);
t = 0:0.001:10;
step(TF,t);
```

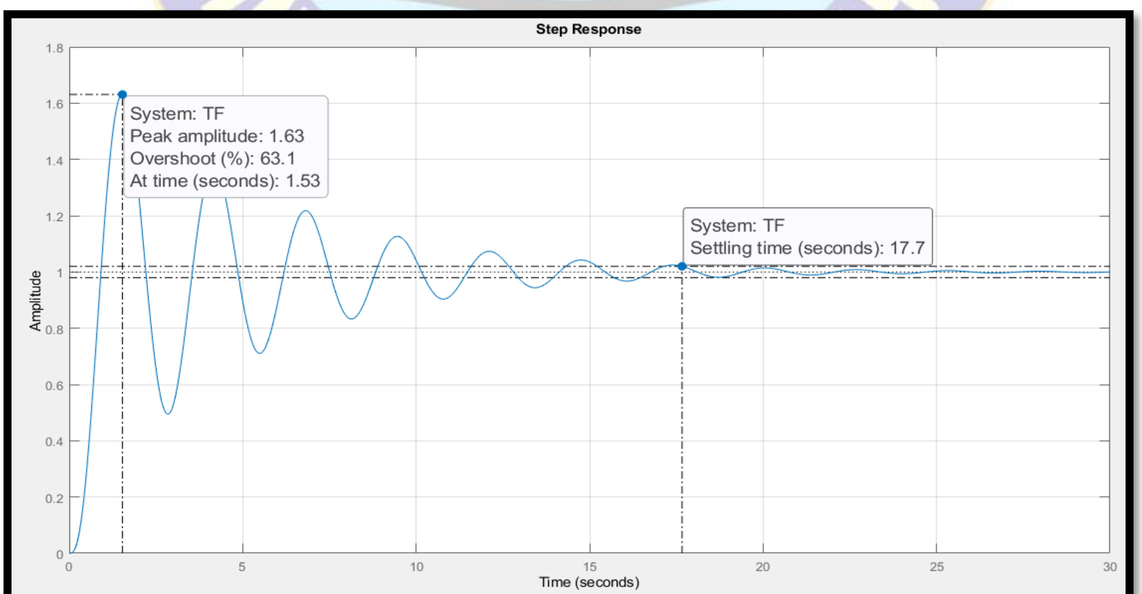
Calculation of Tu:

Tu = 2.52 – 0.627 = 1.893 s

Defining k_p, k_I, k_d

Code:

```
clc; clear all; close all;
ku=60;
Tu=1.893;
%using PI controller
kp=0.45*ku;            %proportional Gain kp = 0.45ku (ku = 60)
ki=kp/(Tu/1.2);        %integral constant ki = kp/Ti  where Ti = Tu/1.2 (Tu = 1.897)
kd=0;
C=pid(kp,ki,kd);
G=tf([1],[1 6 11 6]);
TF=feedback(G*C,1);
step(TF);
grid on;
```

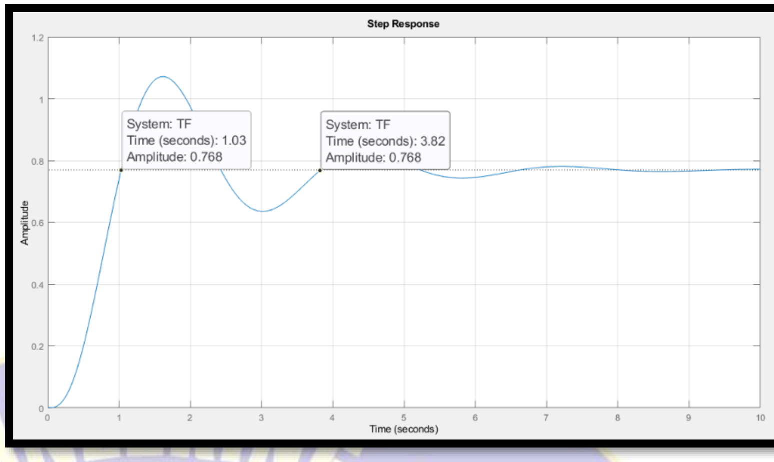


Part 2: PID controller

Finding Critical period (Tu)

Code:

```
ku=20;       %proportional Gain
C=pid(ku);
G=tf([1],[1 6 11 6]);
TF=feedback(G*C,1);
t = 0: 0.001:10;
stepinfo(TF)
step(TF,t);
grid on;
```



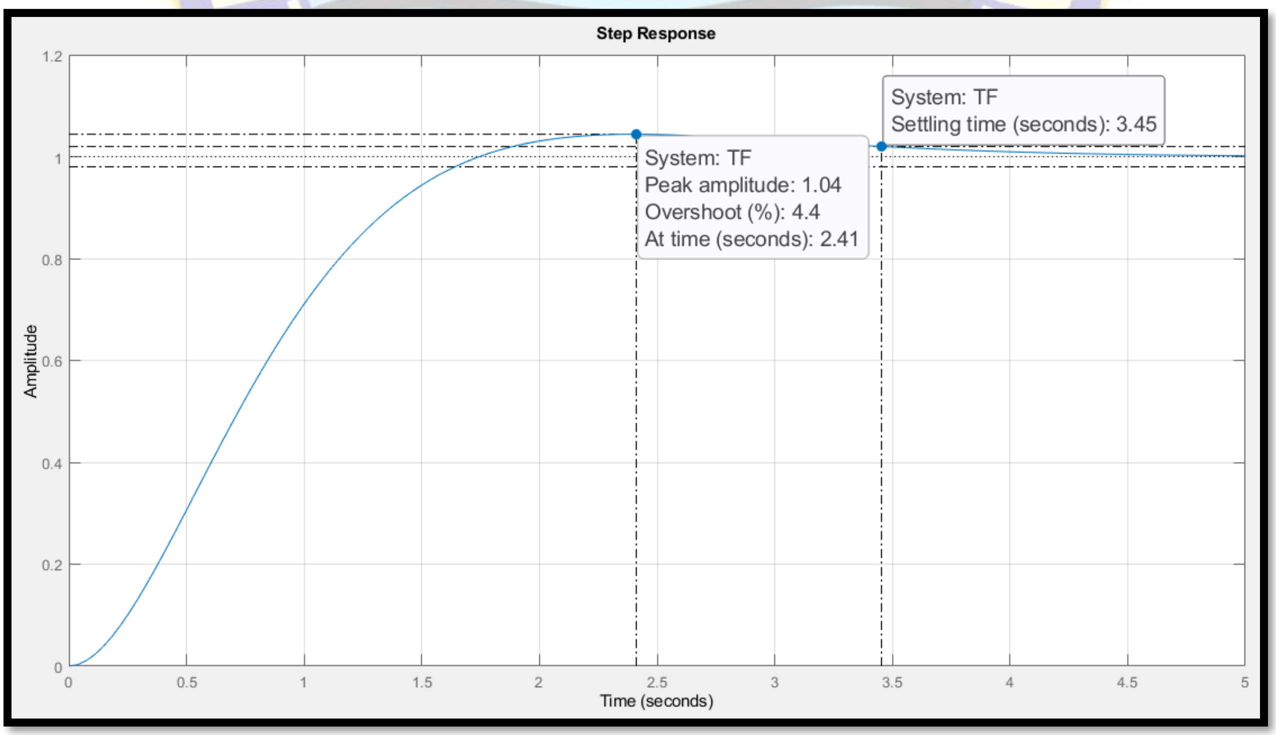
Calculation of Tu:

Tu = 3.82 – 1.03 = 2.79 s

Defining k_p, k_I, k_d

Code:

```
clc; clear all; close all;
ku=20;
Tu=2.79;
kp=0.6*ku;              %proportional Gain kp = 0.6ku (ku = 60)
ki=kp/(Tu/2);             %integral constant ki = kp/Ti  where Ti = Tu/2 (Tu = 1.897)
kd=kp*(Tu/8);           %derivative constant kd = kp*Td  where Td = Tu/8
C=pid(kp,ki,kd);
G=tf([1],[1 6 11 6]);
TF=feedback(G*C,1);
step(TF);
grid on;
```



Conclusion:

In this lab, we have learnt about the System Stability and use of Routh Hurwitz Criterion which proposes a necessary and a sufficient criterion to check stability of any system. The use of PID controller to make desired response of system. PID controller contains a proportional, an integral, and a derivative term represented by K_p, K_I, and K_d respectively. It is quite sophisticated in that it captures the history of the system (through integration) and anticipates the future behavior of the system (through differentiation). The use of **Ziegler Nichols PID tuning method** is to find the gains Kp, KI and Kd appropriately.