CONTROL ENGINEERING LAB

WINTER SEMESTER, 2022

GROUP 18

AMAN SAINI (2020EEB1155)

ANIRUDH SHARMA (2020EEB1158)

ANSAF AHMAD (2020EEB1160)

OBJECTIVE:

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

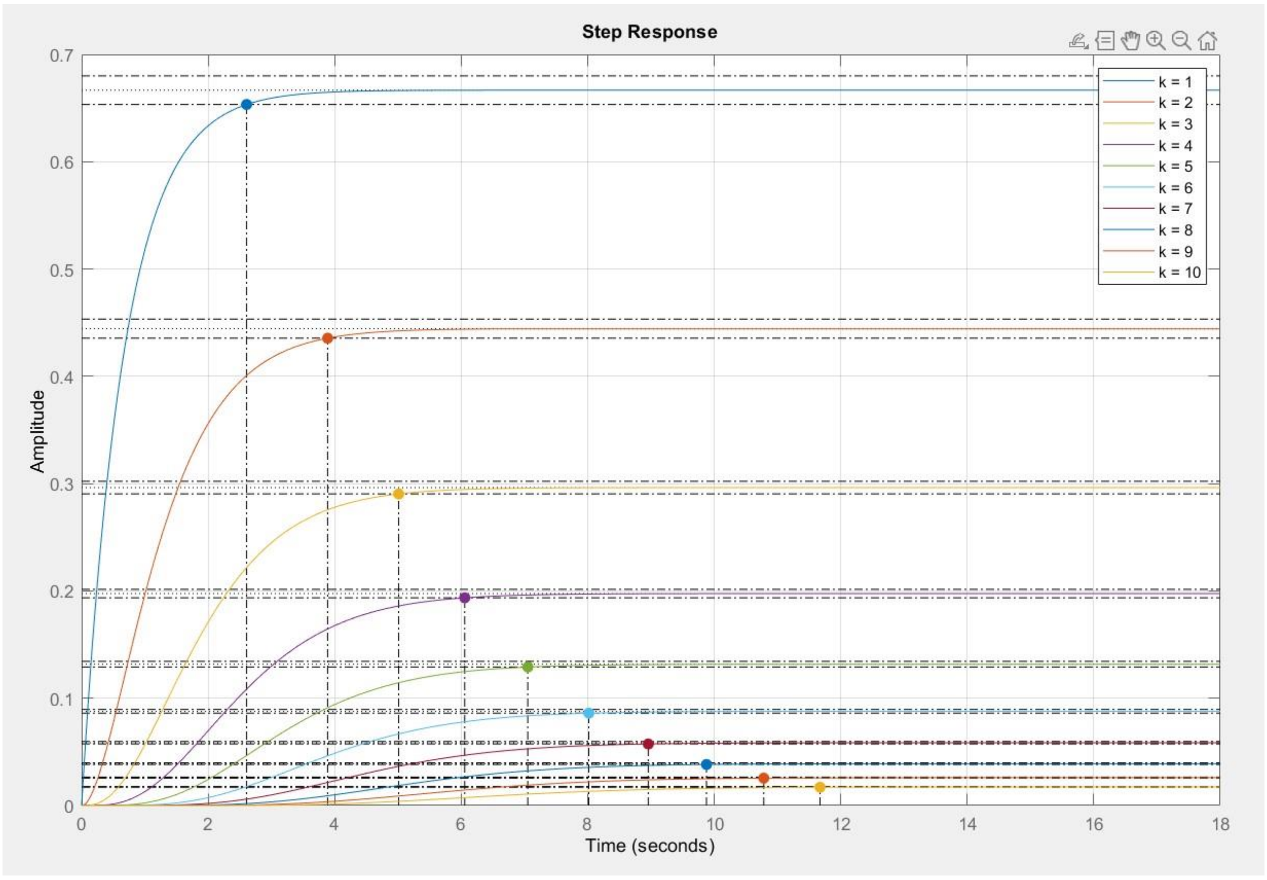
INTRODUCTION:

The following continuous time OLTF is given, on which the studies are required to be performed to design the P, PI, PID controller using Ziegler Nichols rules.

$$G(s) = \frac{1}{(s + 1.5)^k} \; ; k = 1\ldots10$$

Here k quantifies the multiplicity of the pole at s = -1.5. For each multiplicity, a Simulink Model is required to be simulated and a controller needs to be constructed using Ziegler Nichols Rules, and thereafter, the effectiveness of the controller is required to be quantified. It is clear from the OLTF, that the given system represents lag type dynamics, therefore the **process reaction curve method** would be required to obtain the tuned parameters for diff. type of controllers.

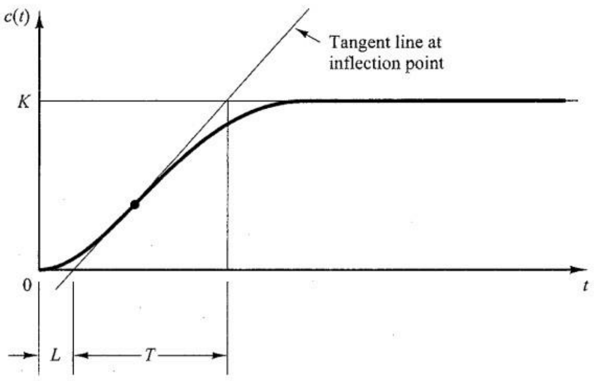
OPEN LOOP SYSTEM DYNAMICS: We study the response of the open loop system for unit step input as reference for different multiplicities and obtain the following plots:



As is clear from the step response, the steady state gain decreases (for a given k, steady-state-gain = $(2/3)^k$) and settling time increases progressively (due to introduction of terms {$\mathbf{t...t^k}$}in the response of the system) with increase in multiplicity. No overshoot or undershoot is observed for any multiplicity.

PROCESS REACTION CURVE:

As we are required to tune PID parameters using Zeigler-Nichol's method, therefore for the given lag type system, we use process reaction curve method and approximate the output of the open loop system using the tangent at point of inflection (max. slope) in unit step response.

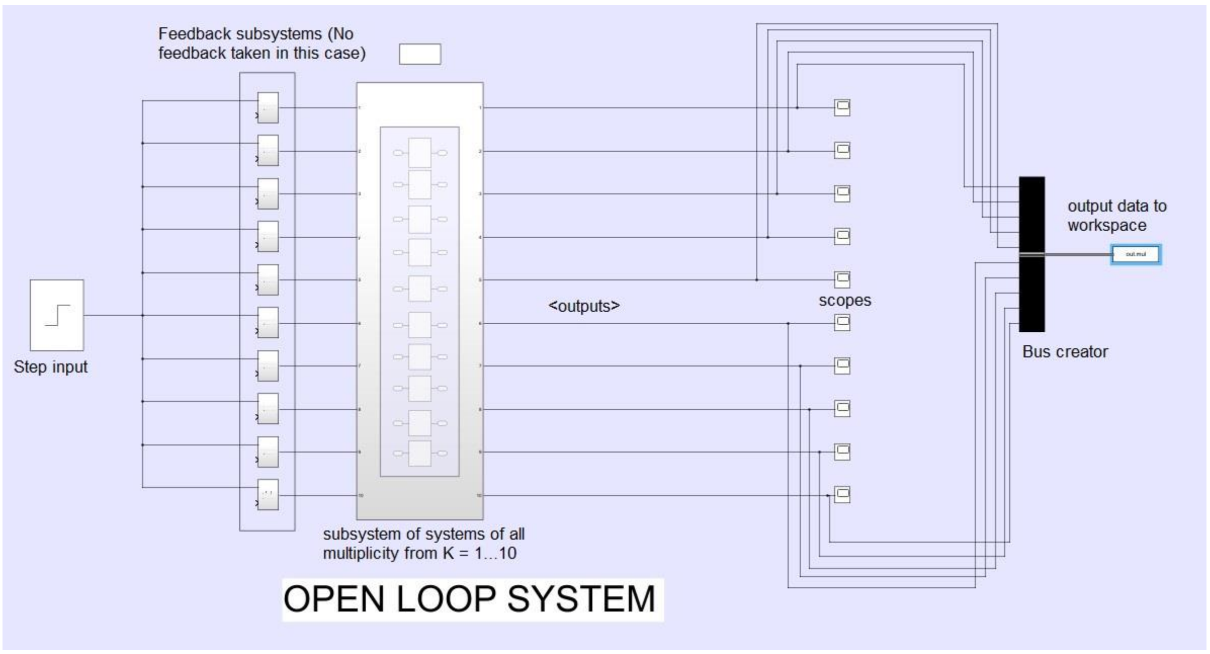


*Point of inflection, lag time **L**, time constant **T** in step response for lag type system*

The tuning parameters can then be approximated from **delay time L** and **time constant T** as follows:

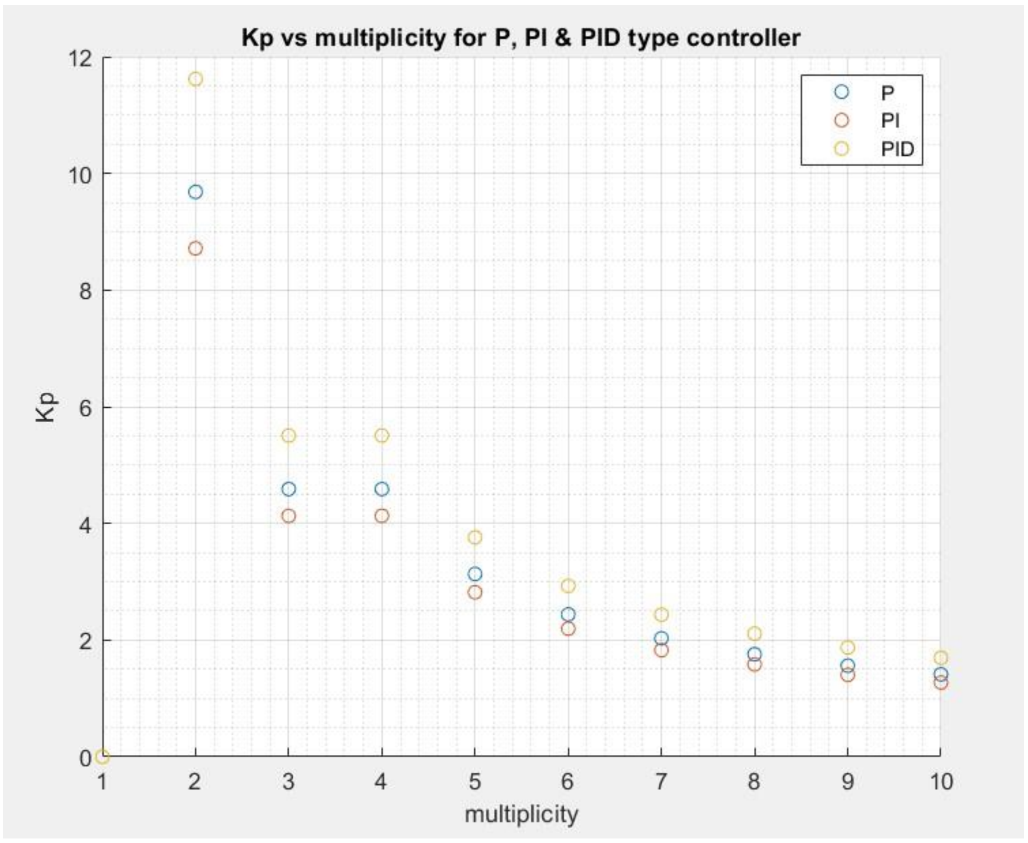
Type of controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	2L	0.5L

We begin by building the model of the given system in Simulink, and obtaining the open loop step response, then we export the output data from Simulink to MATLAB to analyze it and obtain the tuned parameters.

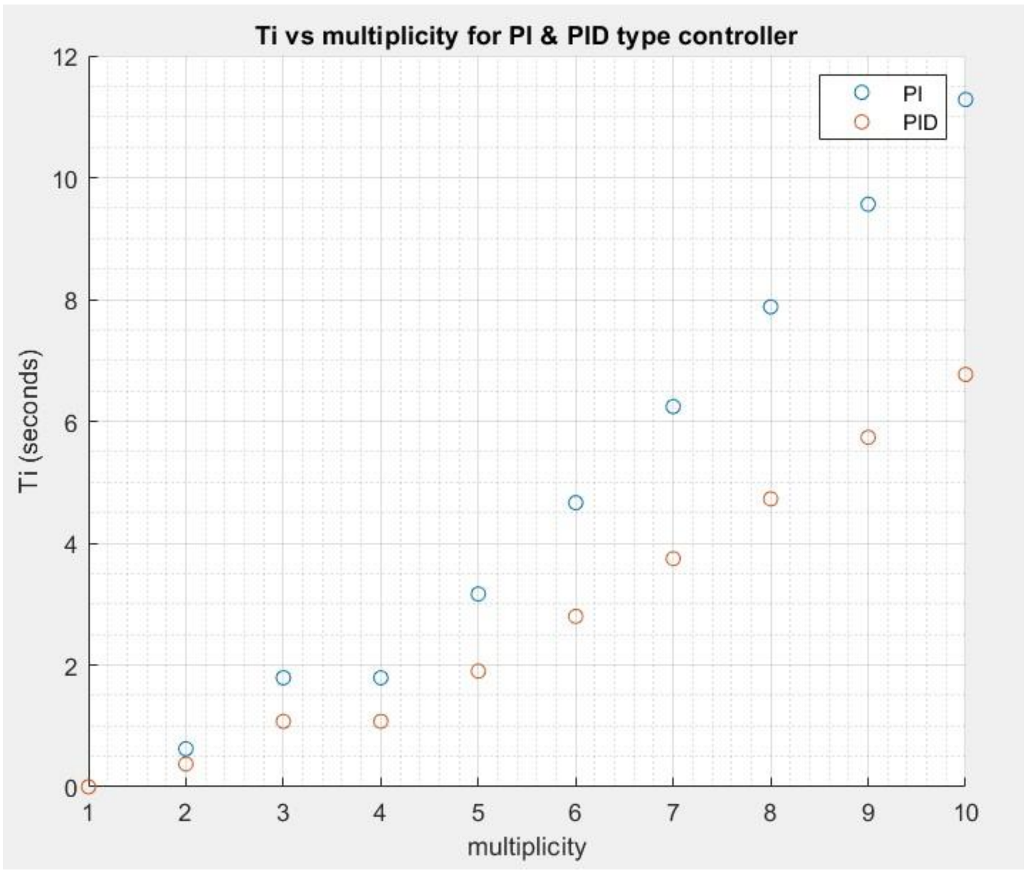

Feedback subsystems (No feedback taken in this case)
Step input
subsystem of systems of all multiplicity from K = 1...10
<outputs>
scopes
output data to workspace
Bus creator
OPEN LOOP SYSTEM

Simulink model for simulating open loop system and exporting the data to workspace

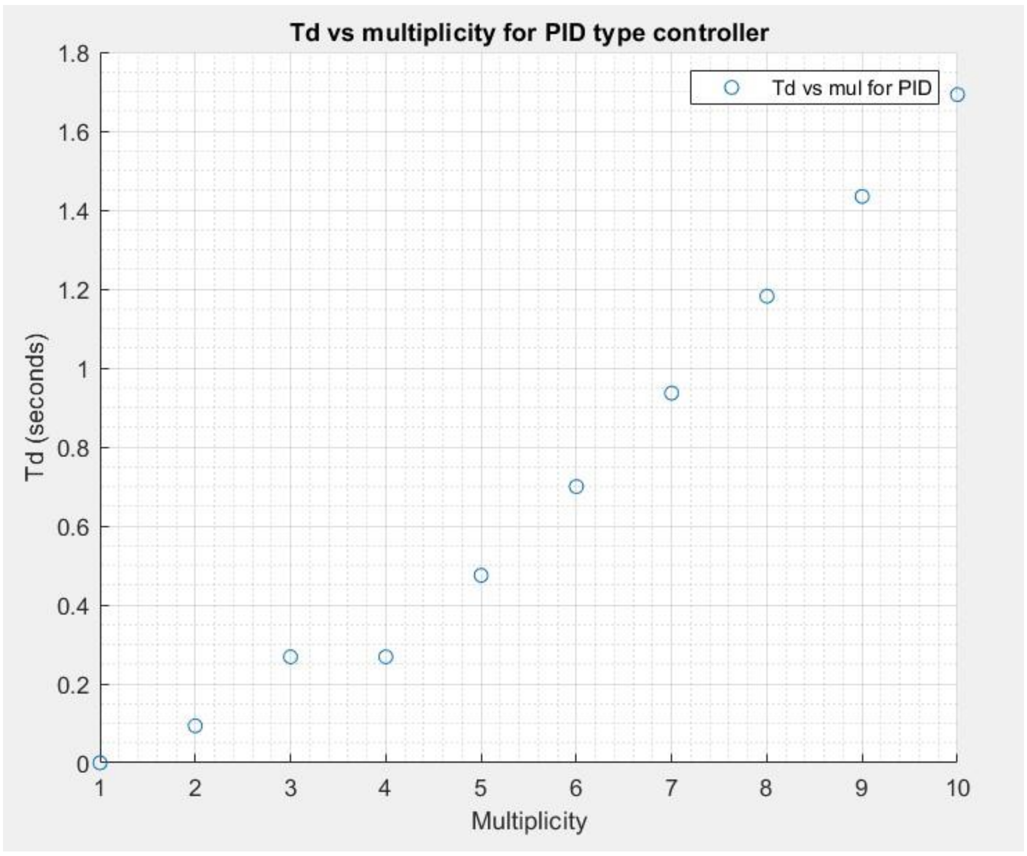
Using the data imported from Simulink to MATLAB, we obtain the parameters for PID controller. The dependencies for K_p, T_i, T_d on multiplicity and the type of control intended can be illustrated using the following plots.



K_p vs multiplicity of the transfer function for P, PI, PID controllers

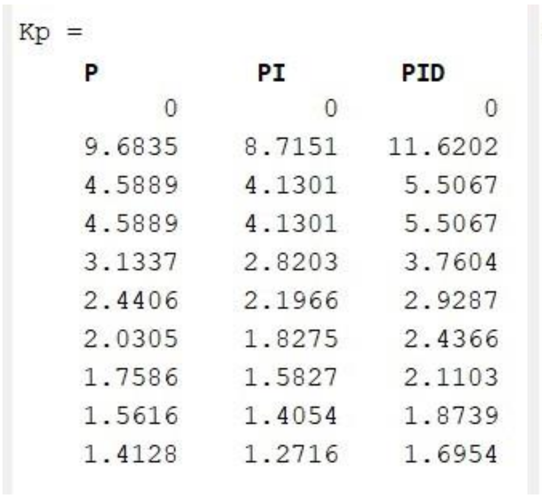


Ti vs multiplicity for PI, PID controller

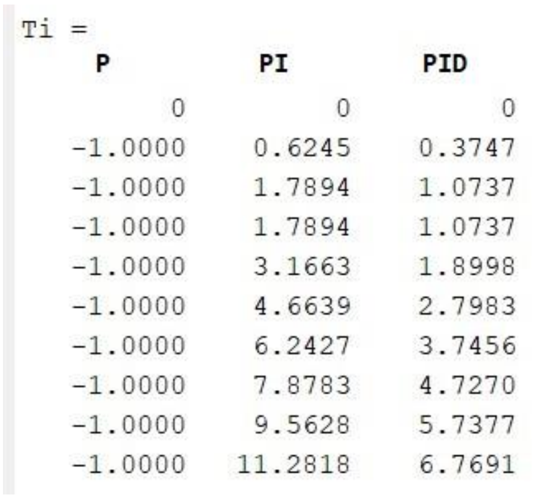


Td vs multiplicity for PID controller

A general trend observed from the above plots is that the value of Kp decreases and Ti, Td increases with increase in multiplicity for every type of controller. It is also important to note that since for multiplicity 1, the system becomes a first order system, which leads to no lag time (and no S-type response), the value of L becomes 0, so the conventional formulas to calculate PID parameters also does not apply. Therefore, from now on, we study the control for system with multiplicity from 2 to 10 and look at the case of Multiplicity = 1 after that.

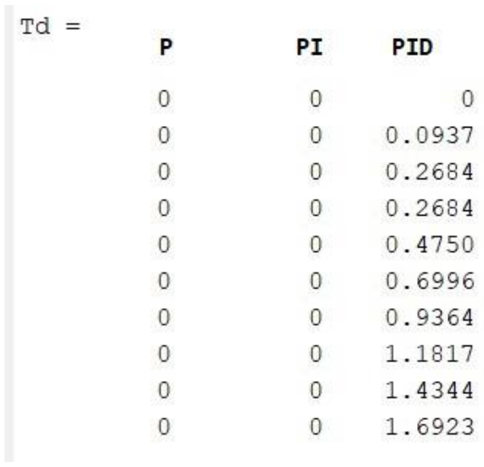


```
Kp =
      P           PI        PID
           0           0           0
      9.6835      8.7151     11.6202
      4.5889      4.1301      5.5067
      4.5889      4.1301      5.5067
      3.1337      2.8203      3.7604
      2.4406      2.1966      2.9287
      2.0305      1.8275      2.4366
      1.7586      1.5827      2.1103
      1.5616      1.4054      1.8739
      1.4128      1.2716      1.6954
```



```
Ti =
      P           PI          PID
           0           0           0
     -1.0000      0.6245      0.3747
     -1.0000      1.7894      1.0737
     -1.0000      1.7894      1.0737
     -1.0000      3.1663      1.8998
     -1.0000      4.6639      2.7983
     -1.0000      6.2427      3.7456
     -1.0000      7.8783      4.7270
     -1.0000      9.5628      5.7377
     -1.0000     11.2818      6.7691
```

Here Ti for P controller is inf (in other words Integrator is not connected in P type controller, -1 is just for that representation (that integrator is not connected))



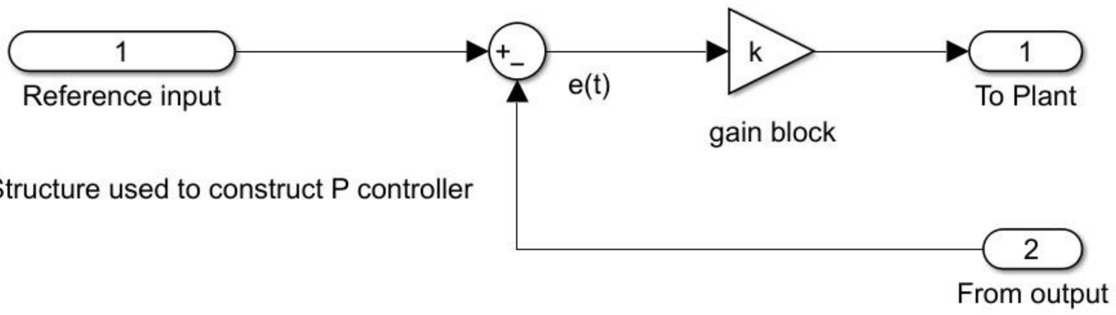
```
Td =
            P          PI      PID

            0           0         0
            0           0    0.0937
            0           0    0.2684
            0           0    0.2684
            0           0    0.4750
            0           0    0.6996
            0           0    0.9364
            0           0    1.1817
            0           0    1.4344
            0           0    1.6923
```

the Kp, Ti, Td for ***multiplicity 1*** *are 0 because they have not been calculated using the conventional formulas as it* ***results in singularities for gains of proportional block***

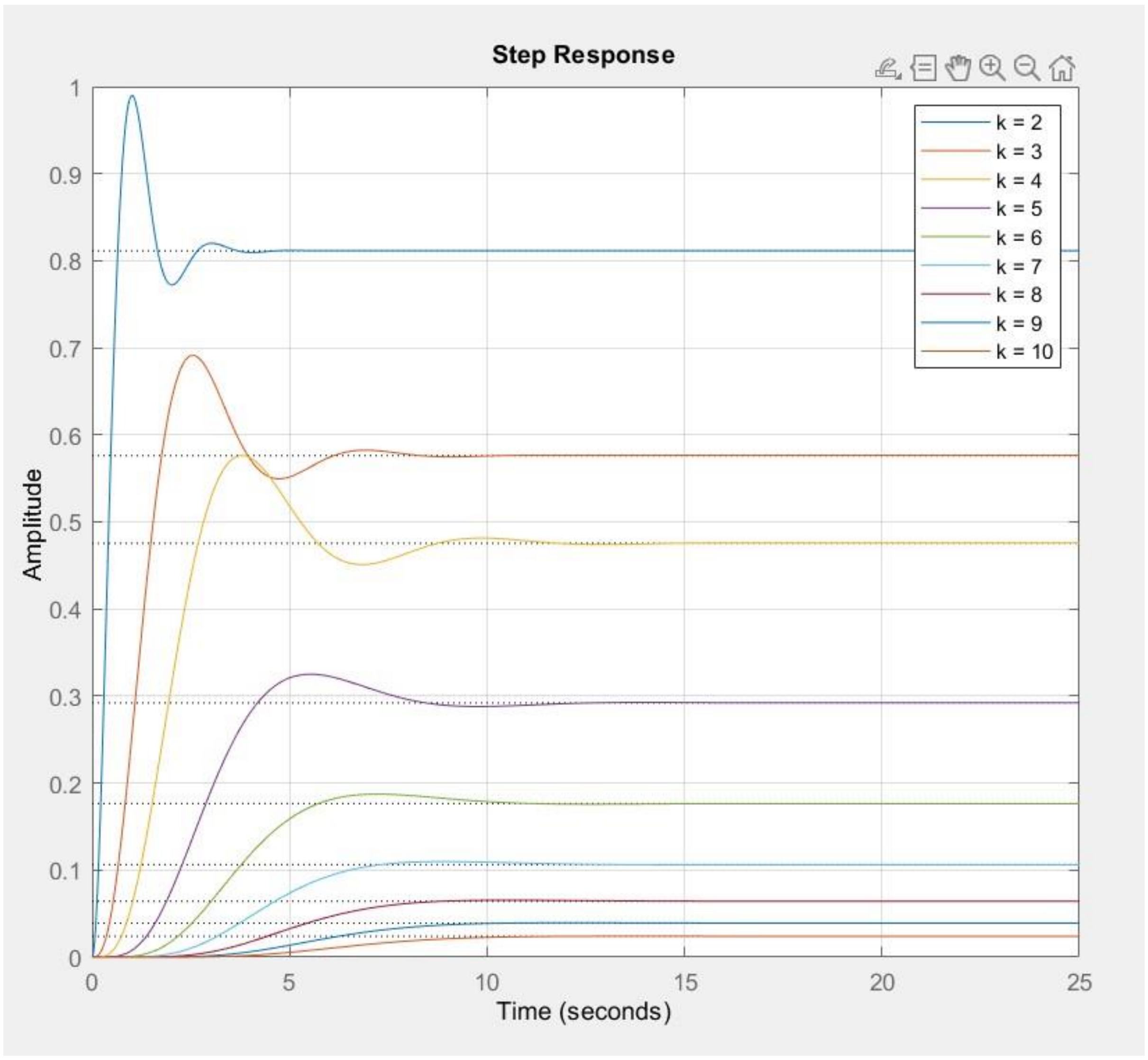
Now that the value of parameters has been obtained it is easy to construct the PID controller in Simulink and study the step response characteristics using MATLAB.

PROPORTIONAL CONTROLLER:

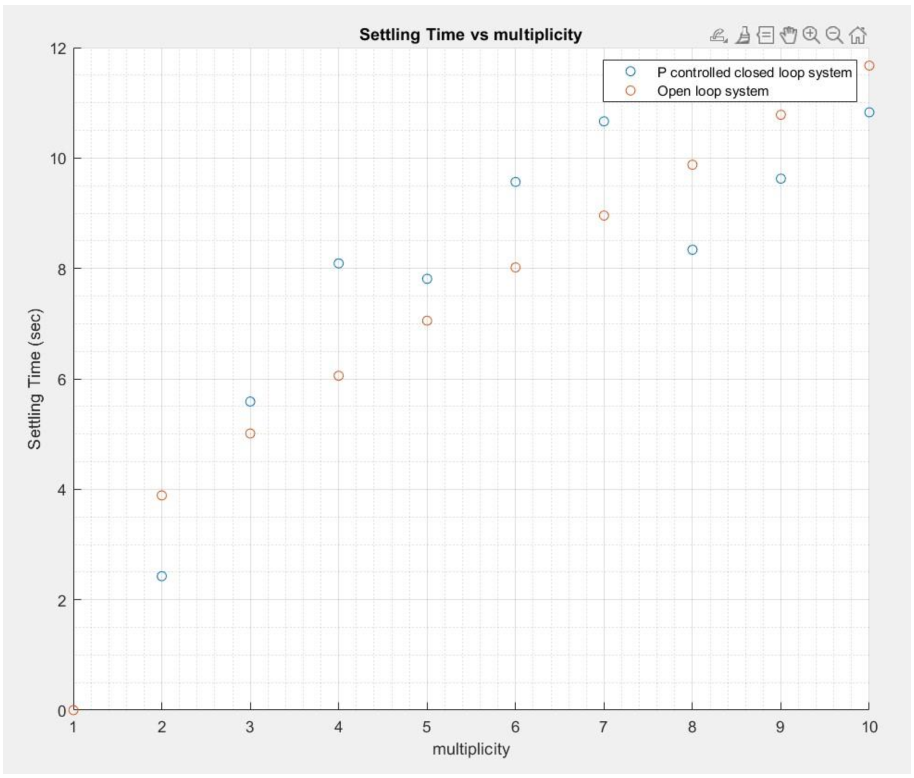
Using a single gain block operating on the output and providing input to the plant we study the step response of the closed loop system, here gain K of proportional block = T/L.



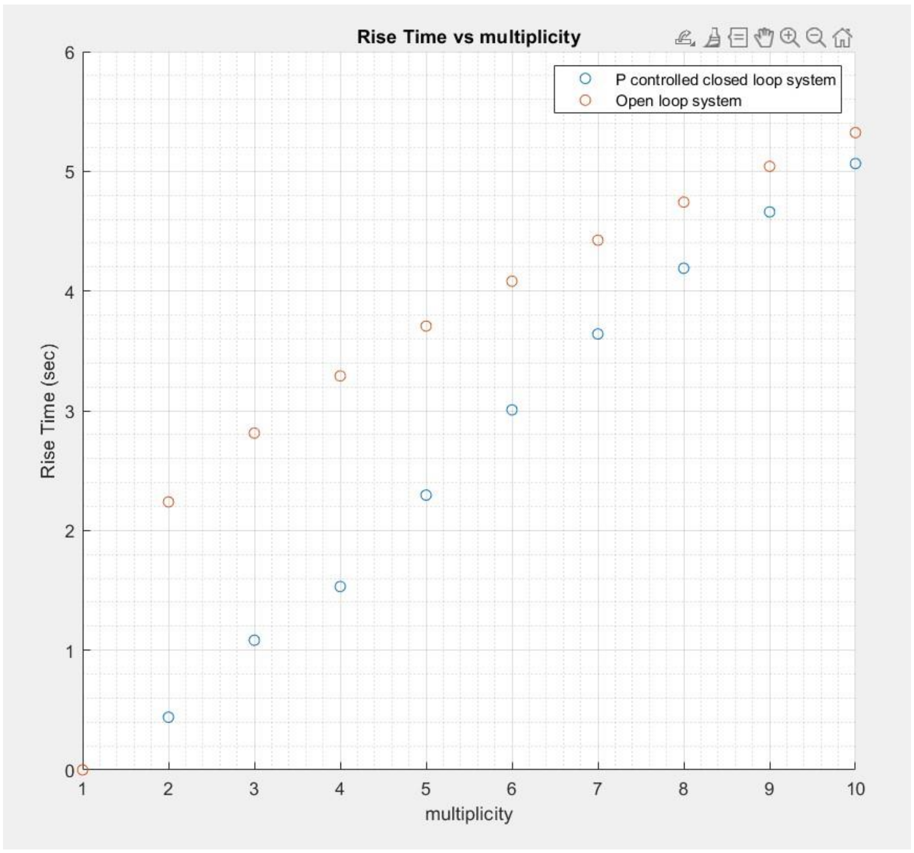
The following step response characteristics are obtained for multiplicities from 2 to 10.



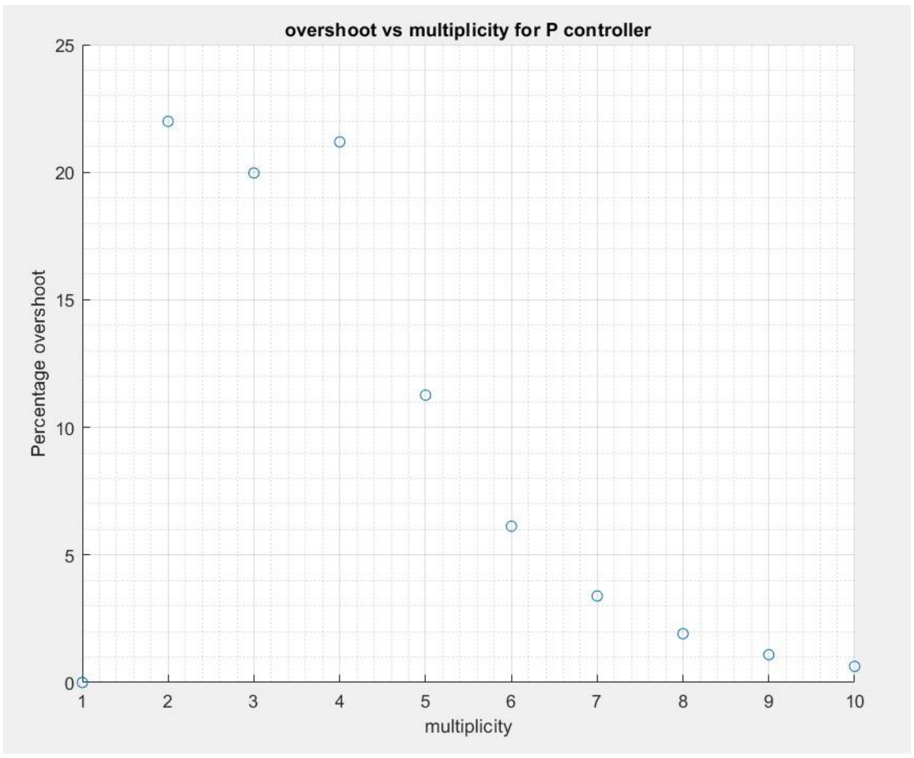
The steady state gain does not match the reference input (i.e., 1), also we observe a peak overshoot in some cases unlike open loop system response. Although we are able to reduce the steady state error using the Ziegler Nichols method for P controller, there is no significant improvement in terms of settling time, in some cases, the settling time is even greater than the original open loop system. The following plots shows the variations in step response characteristics for diff. multiplicities.



Variation of settling time vs multiplicity



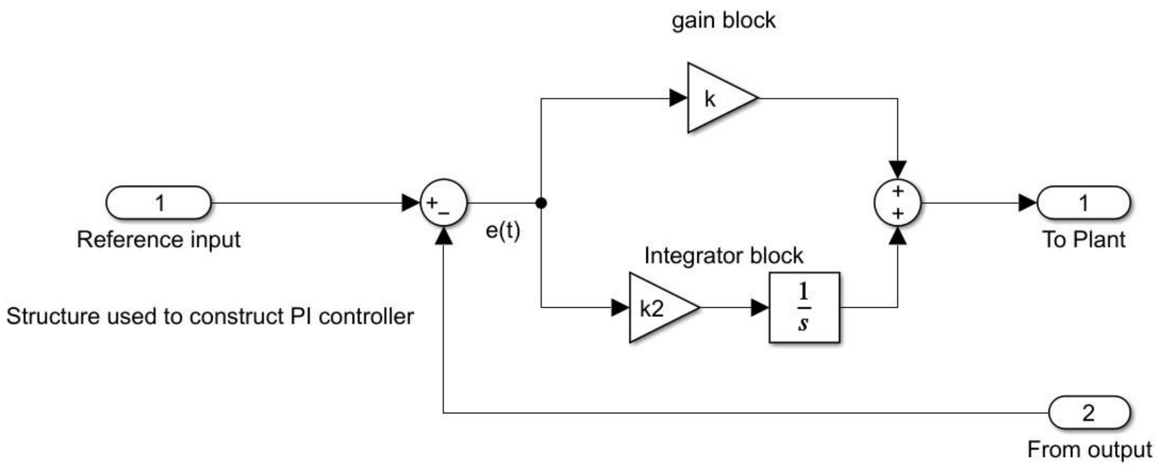
Variation of rise time with multiplicity



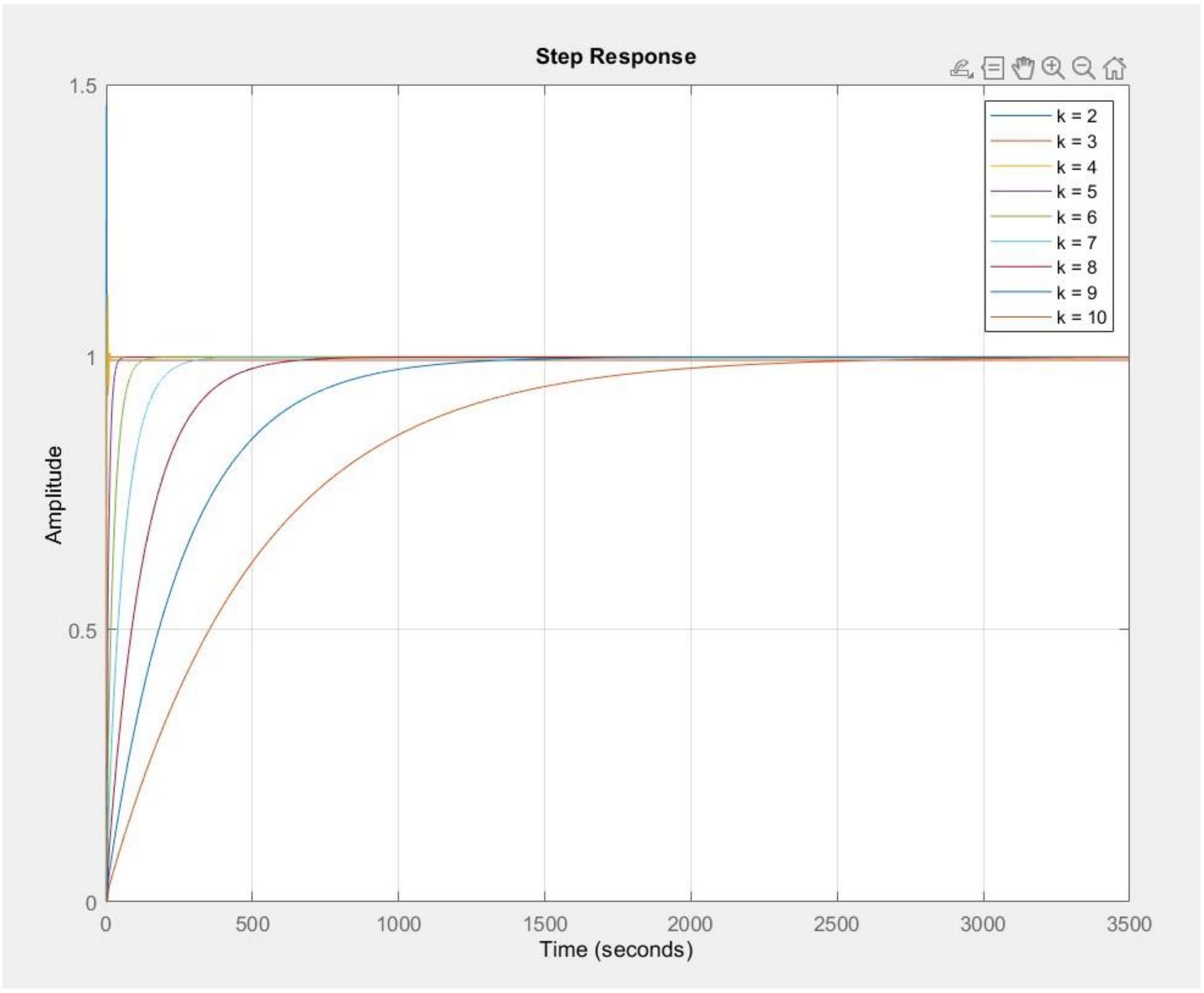
Overshoot vs multiplicity for P controller

As clear from the above graphs that the settling time & rise time increases and overshoot decreases with increase in multiplicity of the plant.

PROPORTIONAL INTEGRAL CONTROLLER:

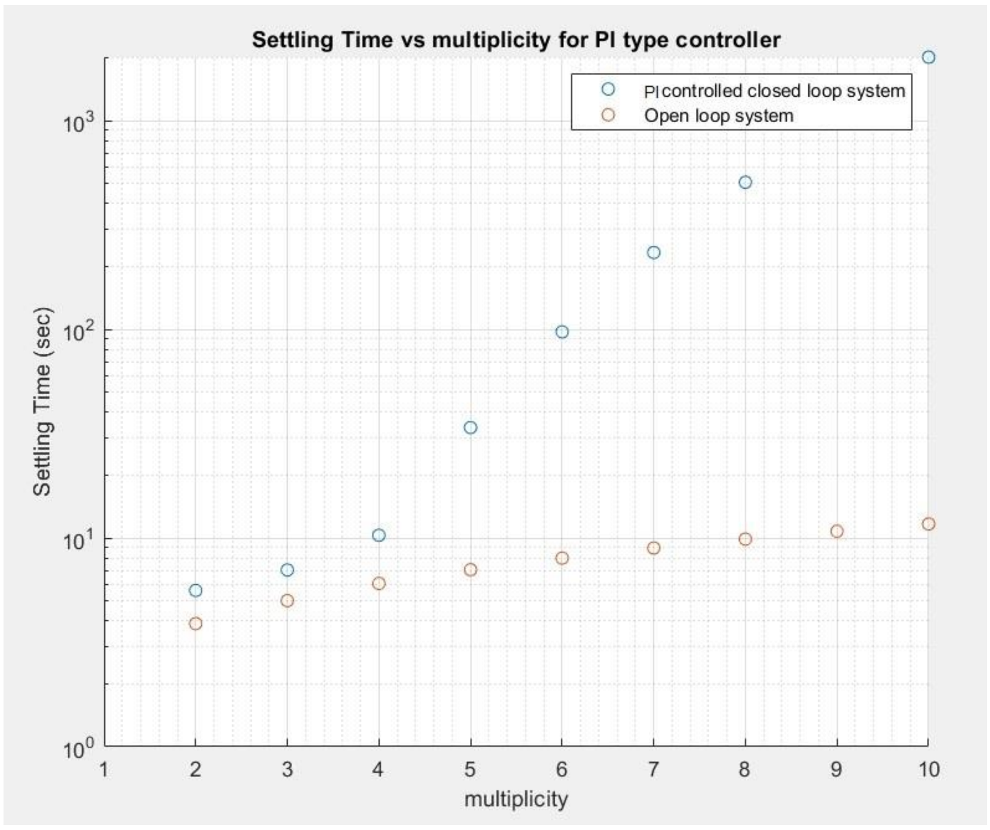


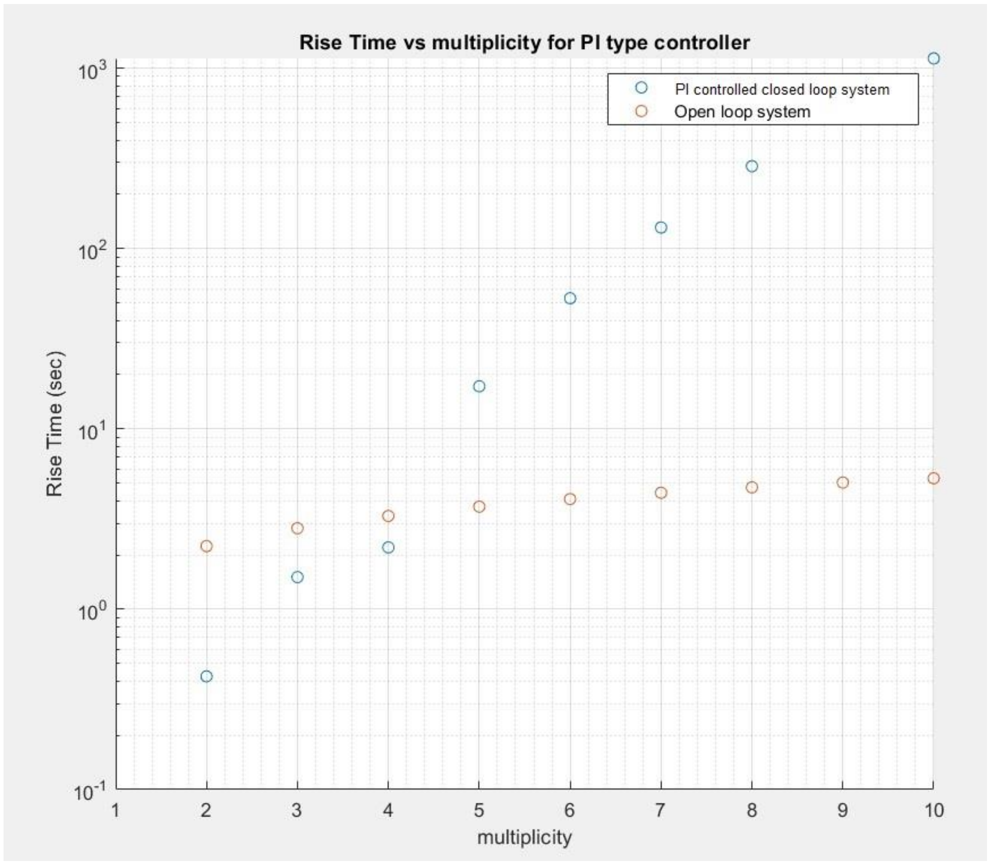
In addition to the gain block, we now use an integrator too with gain K2 **where K2 = Kp / Ti**, the main objective of the integrator is to accumulate the error, which results in steady state error = 0. However, introducing an integrator block makes the system response slower than usual.



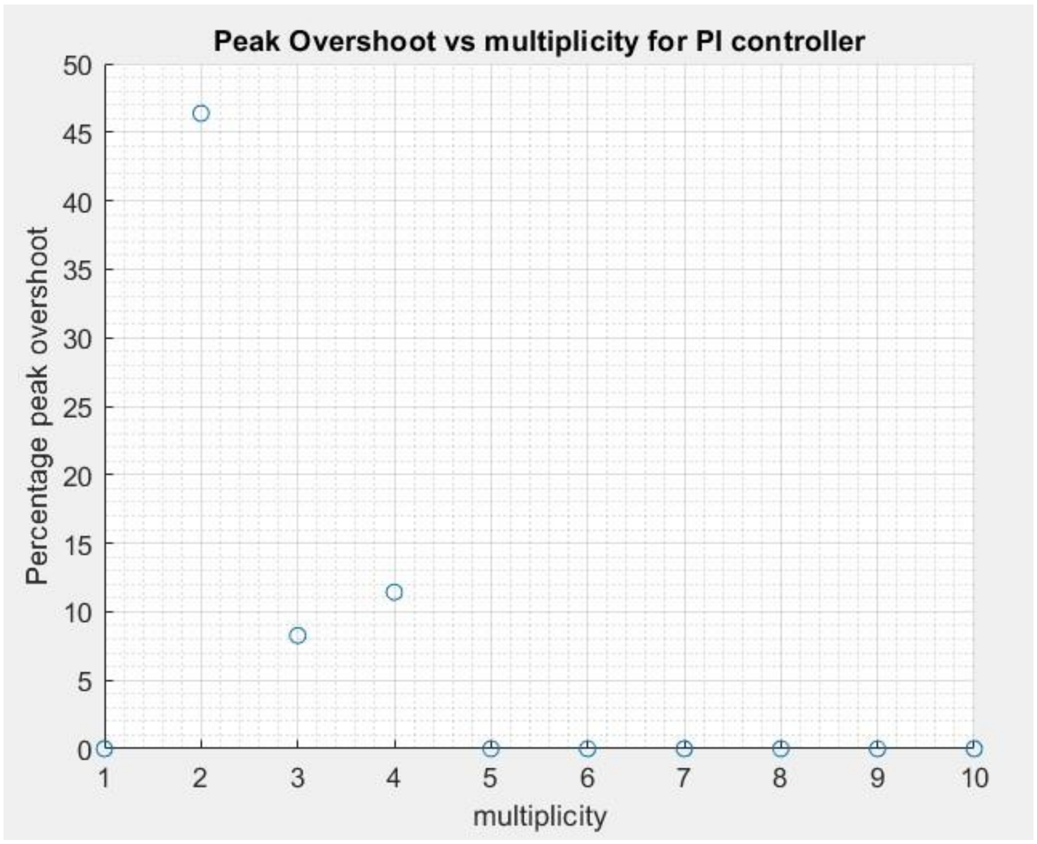
Step response of the system for different multiplicities (settling time increases considerably with increase in multiplicity approaching the order of 10^3)

Various step response characteristics can be summarized using the plots below:



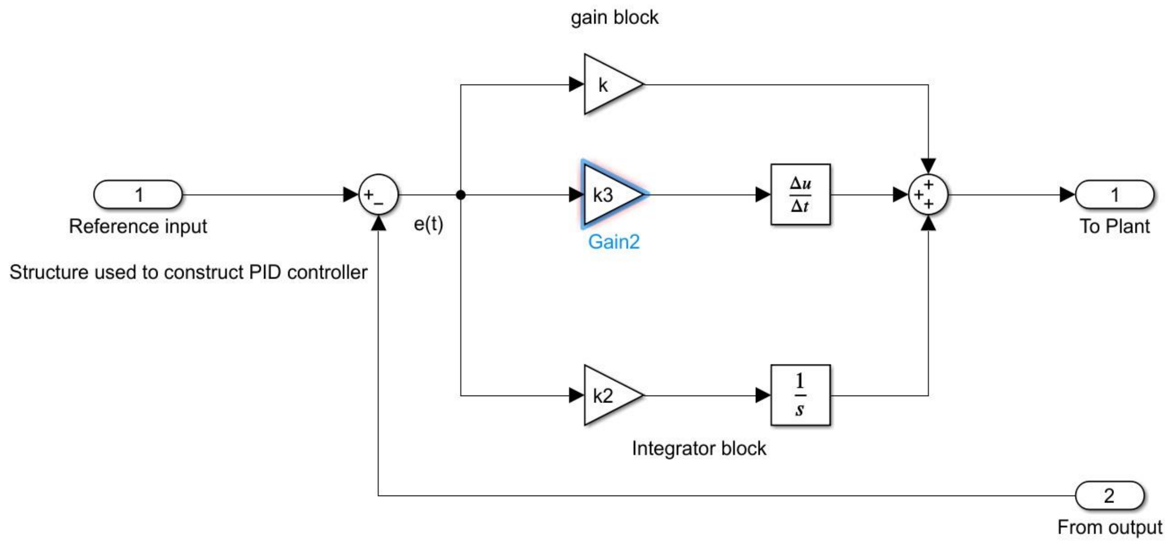


The settling time and Rise Time increases with increase in multiplicity, moreover no improvement in transient time is seen for the PI controller as the settling time shoots to the order of 10^3 seconds.

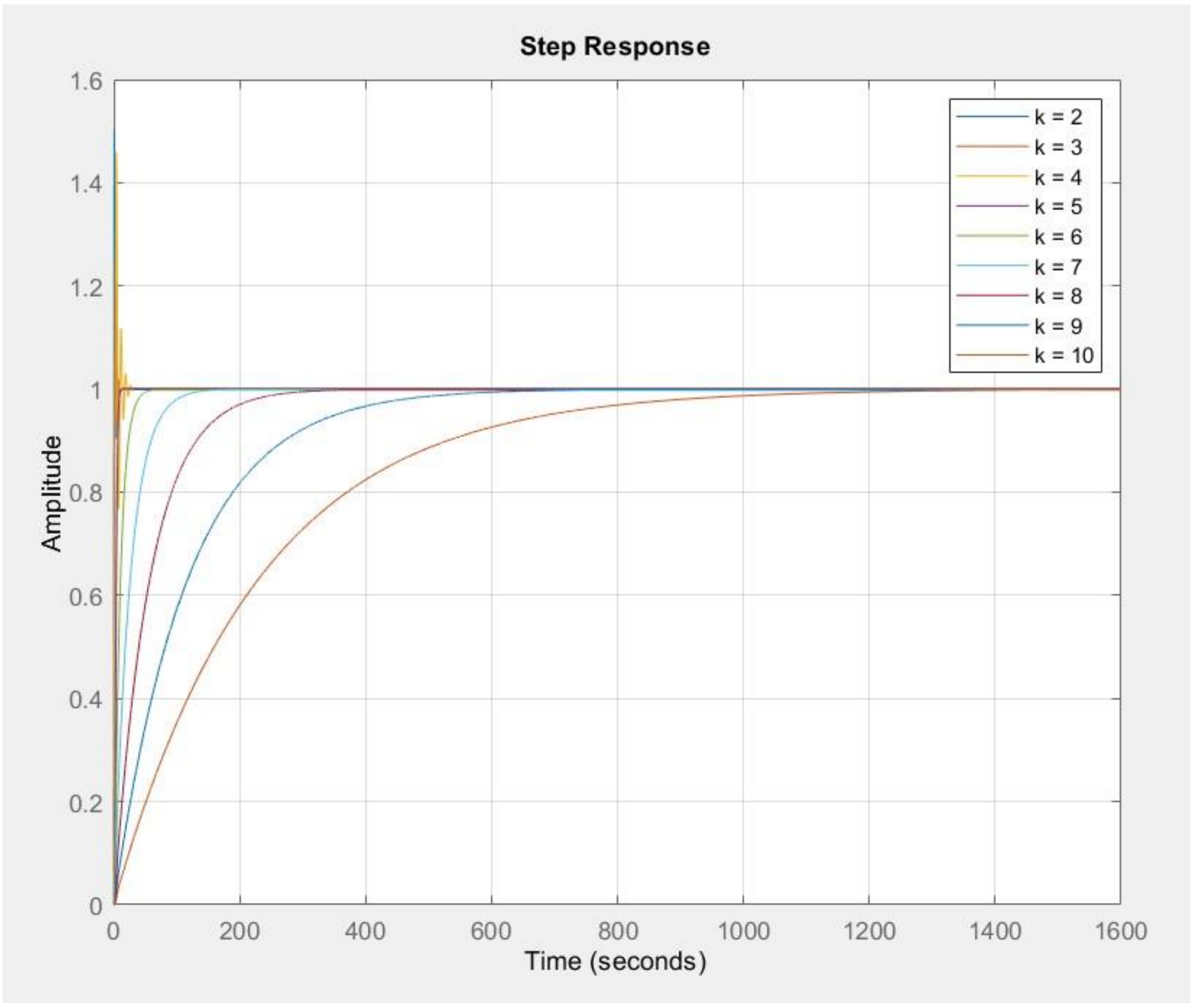


Peak overshoot of up to 47.5% is detected for multiplicity of 2 but it becomes zero for multiplicity > 4 (system starts showing lag type dynamics)

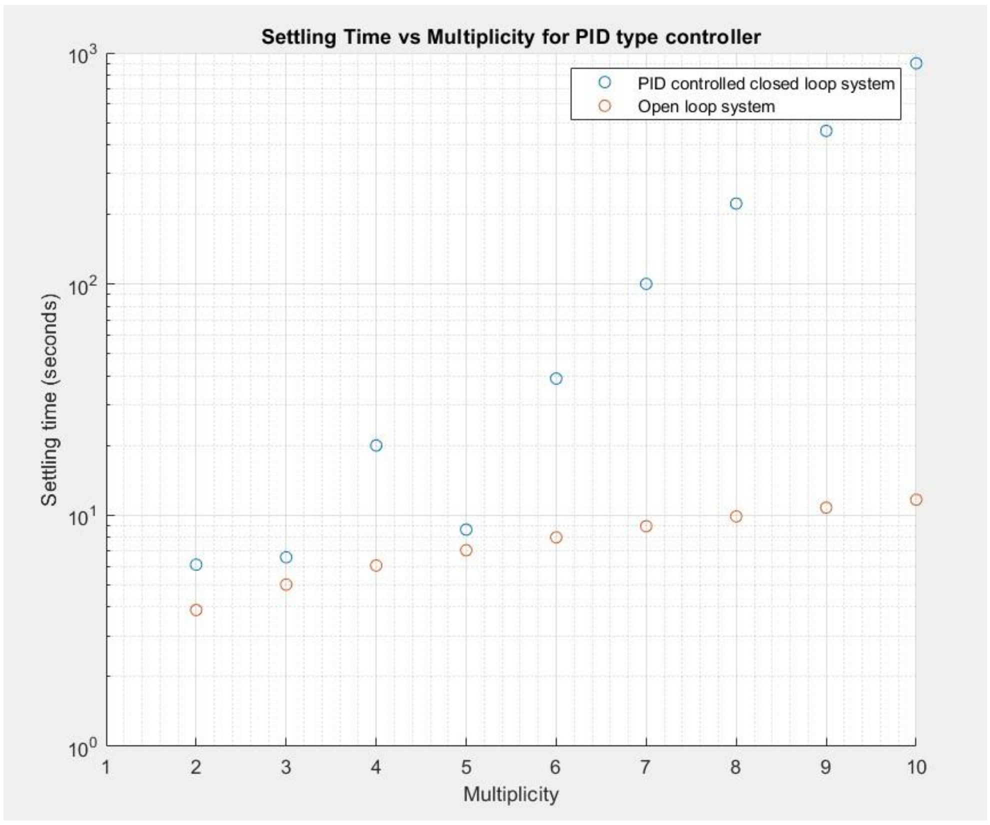
PROPORTIONAL INTEGRAL DERIVATIVE CONTROLLER:



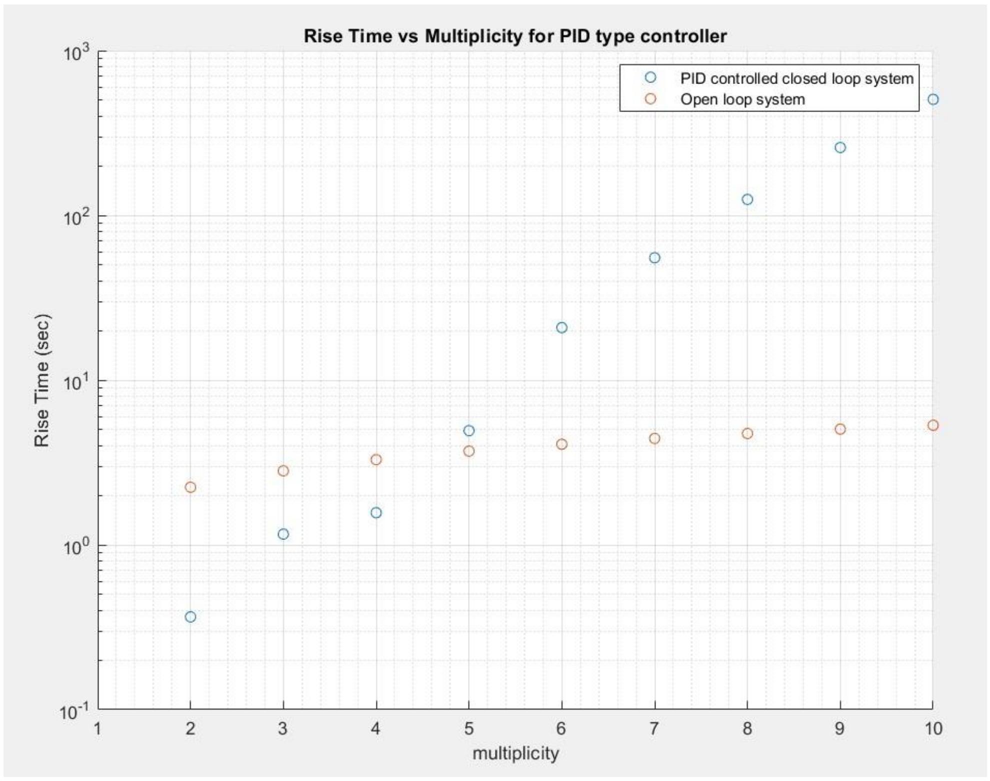
In addition to integrator block, we use a derivative block with gain K3 where **K3 = Td * Kp.** The main use of the derivative block is used to predict the system behaviour, improve settling time and overall stability. The step response of the system can be seen in the following plots.



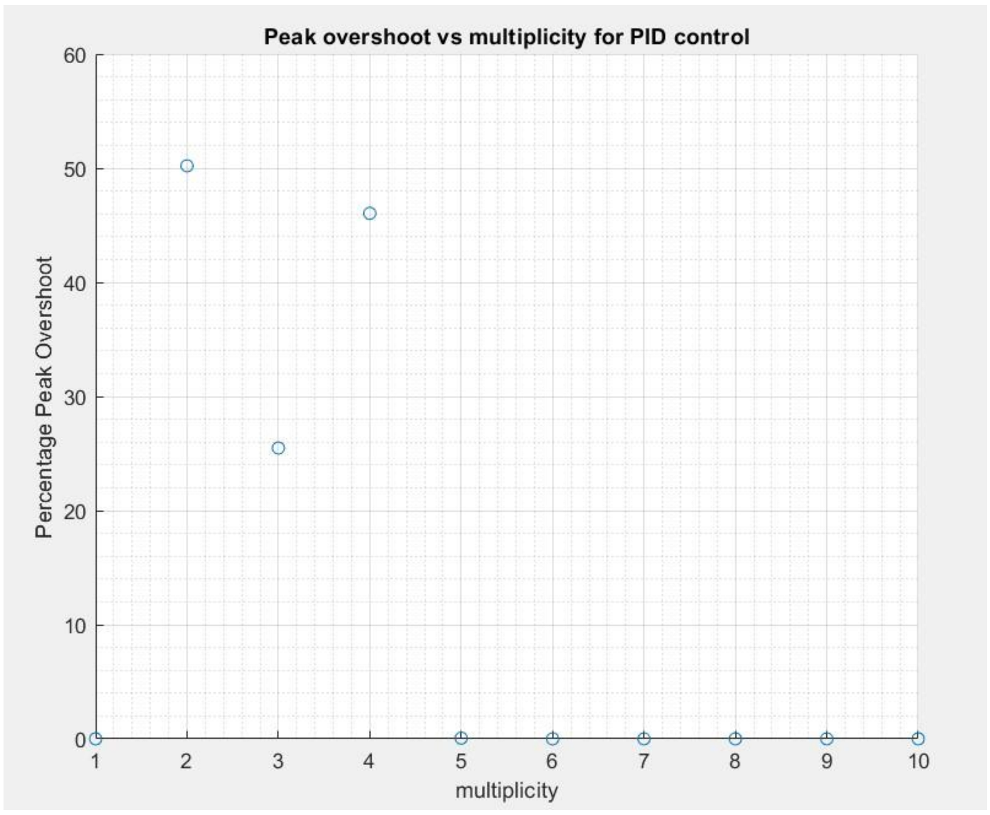
It is easy to see that introducing a derivative block reduces the settling time as compared to the case of PI controller, however it still exceeds the open loop system settling time (by high margin for high multiplicities).



Settling time for PID based controller vs open loop system



Rise time for PID based control vs open loop system



Overshoot for PID controller, the overshoot shoots up to 50% for multiplicity = 2 and it becomes 0 for multiplicity > 4

PID CONTROL FOR MULTIPLICITY = 1:

The system for multiplicity = 1 is a single order system, and therefore the response of the system in not a S type response, which means that it does not show any lag before responding to an input, i.e., point of inflection lies at t = 0. Also, we cannot use the method of sustained oscillations, as the system is a lag type system, therefore tune the controller using PID tuner.

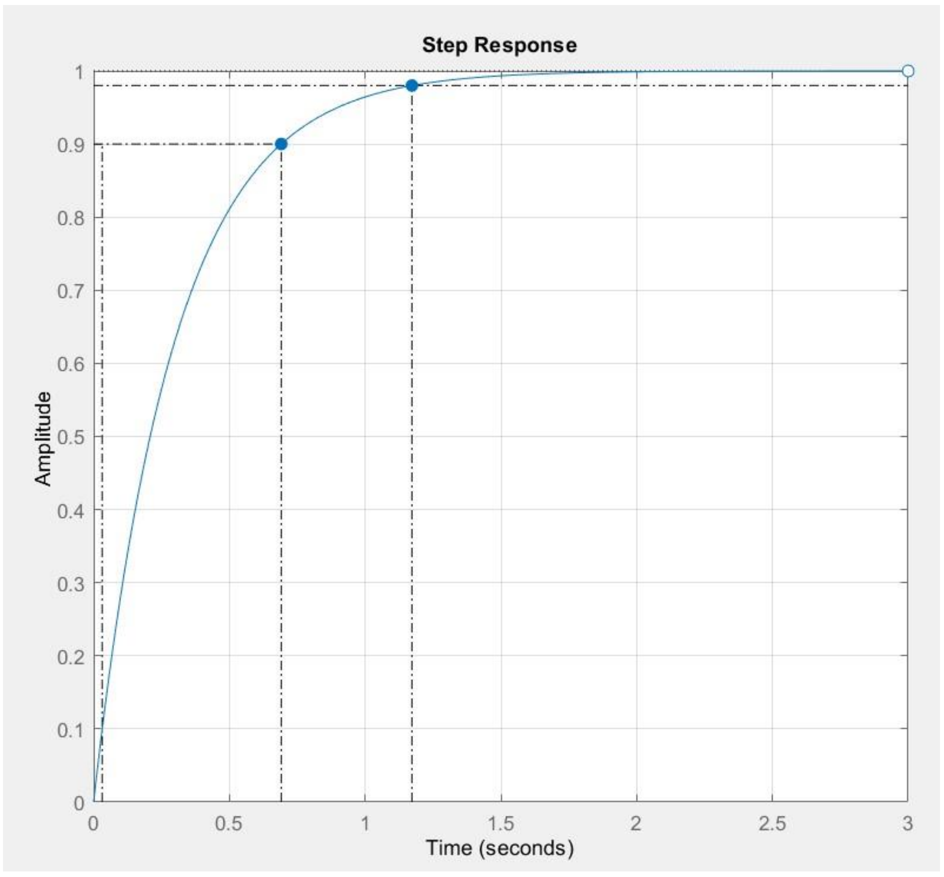
The values for Kp, Ki, Kd obtained are as follows:

Kp = 3.33, Ki = 5, Kd = 0

The following are the step response characteristics obtained for the system for Multiplicity = 1

```
>> stepinfo(feedback(controller * H, 1))

ans =

  struct with fields:

         RiseTime: 0.6592
    TransientTime: 1.1724
     SettlingTime: 1.1724
      SettlingMin: 0.9004
      SettlingMax: 1.0000
        Overshoot: 0
       Undershoot: 0
             Peak: 1.0000
         PeakTime: 3.3079
```

The step response obtained for the system is as follows:



CONCLUSIONS:

We studied the effectiveness of Ziegler-Nichols Tuning methods and observed that the P controller, the reduces the settling time of the system, however it is not appropriate to reduce the steady state error. An Integrator is required for that purpose, however introducing it increases the settling time for higher multiplicity by a large factor. Further introducing the derivative block reduces the settling time. The Tuning method works for multiplicity greater that 1, however it does not do the job for multiplicity = 1 due to absence of lag, here hit and trial or Simulink based PID tuning results in a relatively good response.

MATLAB SCRIPT USED:

```
Kp = zeros(10, 3);
Ti = zeros(10, 3);
Td = zeros(10, 3);


%% DATA RETRIEVAL FROM SIMULINK MODEL
% PROCESS REACTION CURVE METHOD FOR LAG TYPE SYSTEMS
for i=2:10
    fprintf('%s %i\n','Multiplicity',i);
    si=out.mul.signals.values(:,i);
    st=out.mul.time;
    d=diff(si)./diff(st); %d is the slope of the signal
    [C,I]=max(d); %find maximum of the slope C
    y1=out.mul.signals.values(I,i);
    x1=out.mul.time(I);
    x=0:(x1+30);
    y=(C*(x-x1))+y1;
    %legend('signal','tangent')
    T=((si(end-1)-y1)/C)+x1;
    L=-(y1/C)+x1;
    T = T - L;
    Kp(i, (1:3)) = [(T)/L;0.9*T/L;1.2*T/L]; % prooprtional gain for P, Pi, Pid
    Ti(i, (1:3)) = [-1;L/0.3;2*L]; % time constant for I
    Td(i, (1:3)) = [0;0;0.5*L]; % time constant for D
    %hold on
    %plot(st,si)
    %plot(x,y);
    %hold on
end


%% Dependencies on Multiplicity
mul = (1:10); % array for multiplicity
scatter(mul,Kp(:, 1)); grid on; grid minor; hold on; % P
scatter(mul,Kp(:, 2)); % PI
scatter(mul,Kp(:, 3)); % PID
legend('P','PI','PID'); hold off;
figure;
scatter(mul,Ti(:, 2));hold on; % PI;
scatter(mul,Ti(:, 3)); %PID;
legend('PI', 'PID'); grid on; grid minor;
hold off;
figure;
scatter(mul, Td(:, 3)); grid on; grid minor;
legend('Td vs mul for PID')


%% P CONTROL
s = tf('s'); H = 1/(s+1.5);
settling_Time = zeros(1, 10);
oshoot = zeros(1, 10);
figure;
for i = 2:10
    controller = Kp(i, 1);
    cltf = ((controller)*(H^i))/(1 + controller*(H^i));
    step(cltf); hold on;
    A = stepinfo(cltf);
    settling_Time(i) = A.SettlingTime;
    oshoot(i) = A.Overshoot;
end
grid on; grid minor; hold off;
figure;
scatter(mul, settling_Time); grid on; grid minor;
figure;
scatter(mul, oshoot); grid on; grid minor;
```

```
%% PI control
figure;
for i = 2:10
    controller = Kp(i, 2) + Kp(i, 2)/(s*Ti(i, 2));
    cltf = ((controller)*(H^i))/(1 + controller*(H^i));
    step(cltf); hold on;
    A = stepinfo(cltf);
    settling_Time(i) = A.SettlingTime;
    oshoot(i) = A.Overshoot;
end
grid on; grid minor; hold off;
figure;
scatter(mul, settling_Time); grid on; grid minor;
figure;
scatter(mul, oshoot); grid on; grid minor;


%% PID CONTROL
figure;
for i = 2:10
    controller = Kp(i, 3)*(1 + 1/(s*Ti(i, 3)) + s*Td(i, 3));
    cltf = ((controller)*(H^i))/(1 + controller*(H^i));
    step(cltf); hold on;
    A = stepinfo(cltf);
    settling_Time(i) = A.SettlingTime;
    oshoot(i) = A.Overshoot;
end
grid on; grid minor; hold off;


figure;
scatter(mul, settling_Time); grid on; grid minor;
figure;
scatter(mul, oshoot); grid on; grid minor;
```