**LAB # 10**

**OBJECTIVE:** Generate a Matlab Code to study the effect of each of the parameters of the PID controller by using the step response of closed loop system.

**THEORY:**

**PID Theory**

1. **Proportional Response**

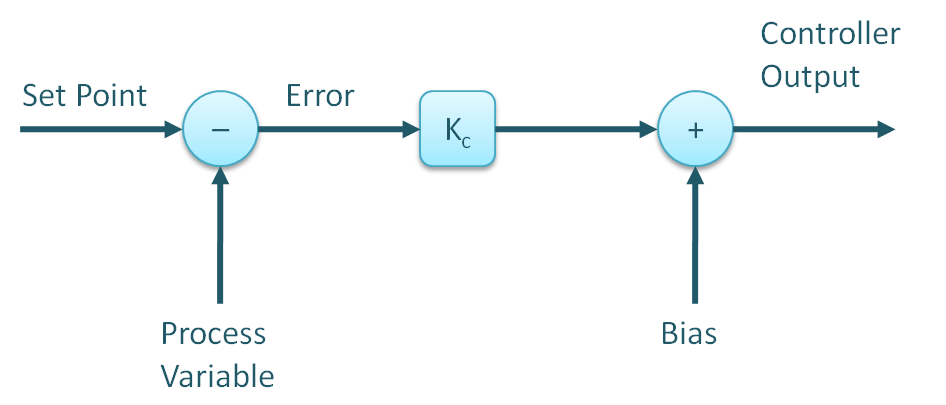
The proportional control mode is in most cases the main driving force in a controller. It changes the controller output in proportion to the error. If the error gets bigger, the control action gets bigger. This makes a lot of sense, since more control action is needed to correct large errors.

[Proportional Action](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\p_action.png)The adjustable setting for proportional control is called the Controller Gain (Kc).

The proportional component depends only on the difference between the set point and the process variable. This difference is referred to as the Error term. The proportional gain (Kc) determines the ratio of output response to the error signal. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If Kc is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

Adjusting the controller gain setting actually influences the integral and derivative control modes too. That is why this parameter is called controller gain and not proportional gain.

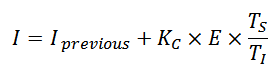
[Proportional Controller](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\p_controller.png)



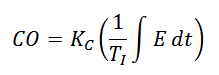
**Figure1. A proportional-only controller algorithm.**

1. **Integral Response**

The integral component sums the error term over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and set point. A phenomenon called integral windup results when integral action saturates a controller without the controller driving the error signal toward zero.  
If the error is large, the integral mode will increment/decrement the controller output fast, if the error is small, the changes will be slower. For a given error, the speed of the integral action is set by the controller’s integral time setting (TI). A large value of TI (long integral time) results in a slow integral action, and a small value of TI (short integral time) results in a fast integral action. If the integral time is set too long, the controller will be sluggish, if it is set too short, the control loop will oscillate and become unstable.

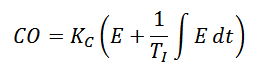
[](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\i_action.png)

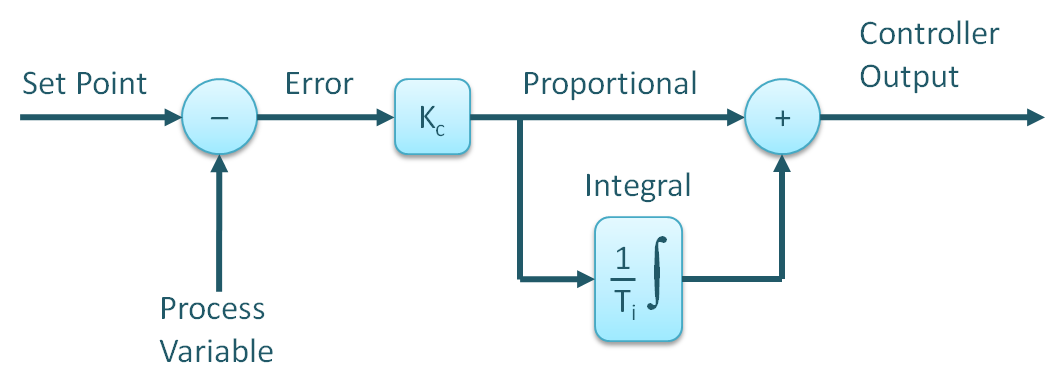
TS is the control algorithm’s execution interval, sometimes called sampling time or scan time.

[](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\i_controller.png)

1. **Proportional + Integral Controller**

Commonly called the PI controller, its controller output is made up of the sum of the proportional and integral control actions (Figure 8).

[](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\pi_controller.png)

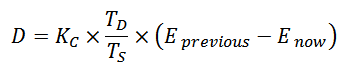


**Figure2. The PI controller algorithm.**

1. **Derivative Response**

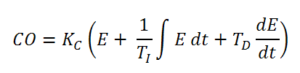
The third control mode in a PID controller is derivative. Derivative control is rarely used in controlling processes, but it is used often in motion control. For process control, it is not absolutely required, is very sensitive to measurement noise and it makes trial-and-error tuning more difficult. Nevertheless, using the derivative control mode of a controller can make a control loop respond a little faster than with PI control alone.

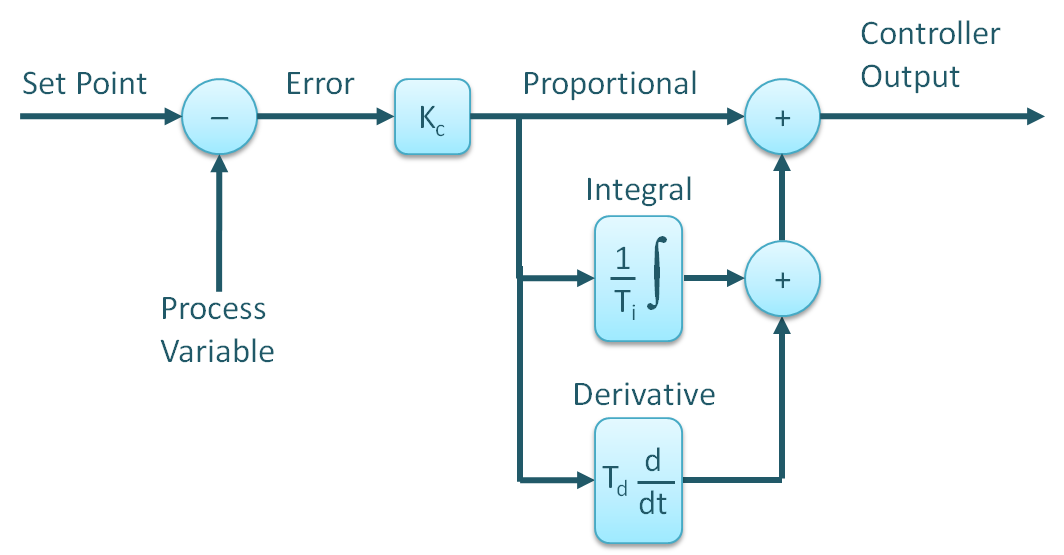
The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. The derivative control mode produces an output based on the rate of change of the error. Derivative mode is sometimes called Rate. The derivative mode produces more control action if the error changes at a faster rate. If there is no change in the error, the derivative action is zero. The derivative mode has an adjustable setting called Derivative Time (TD). Increasing the derivative time (Td) parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (Td), because the Derivative Response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable.

[](file:///C:\Users\Tayyab\Desktop\PID\PID%20Controllers%20Explained%20_%20Control%20Notes_files\d_action.png)

1. **Proportional + Integral + Derivative Controller**

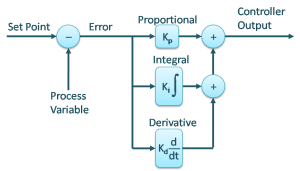
Commonly called the PID controller, its controller output is made up of the sum of the proportional, integral, and derivative control actions (Figure 11). There are other configurations too.

[](http://blog.opticontrols.com/wp-content/uploads/2011/03/pid_controller-1.png)



**Figure3. The Standard (Noninteractive) PID controller algorithm.**

1. **Parallel Algorithm**

[](http://blog.opticontrols.com/wp-content/uploads/2010/03/parallel.png)

**Figure4. Parallel Controller Algorithm**

[C:\Users\Tayyab\Desktop\PID\PID Controller Algorithms _ Control Notes_files\Parallel-Equation-300x54.jpg](http://blog.opticontrols.com/wp-content/uploads/2010/03/Parallel-Equation.jpg)

Some academic textbooks discuss the parallel form of PID controller, but it is also used in some DCSs and PLCs.  This algorithm is simple to understand, but not intuitive to tune.  The reason is that it has no controller gain (affecting all three control modes), it has a proportional gain instead (affecting only the proportional mode).  Adjusting the proportional gain should be supplemented by adjusting the integral and derivative settings at the same time.

$$ K_p + \frac {K_i} {s} + K_d s = \frac{K_d s^2 + K_p s + K_i} {s} $$

Use help command to find details of following commands.

1. **tf:**

 To convert state-space or zero-pole-gain models to transfer functionform. sys = tf(num,den) creates a continuous-time transfer function with numerator(s) and denominator(s) specified by num and den .

1. **Feedback:-**

The closed-loop model sys has u as input vector and y as output vector. The models sys1 and sys2 must be both continuous or both discrete with identical sample times

1. **Plot:-**

MATLAB allows you to add title, labels along the x-axis and y-axis, grid lines and also to adjust the axes to spruce up the graph. The xlabel and ylabel commands generate labels along x-axis and y-axis. The title command allows you to put a title on the graph.

1. **Subplot :-**

MATLAB® numbers subplot positions by row. The firstsubplot is the first column of the first row, the secondsubplot is the second column of the first row, and so on.

1. **Step :-**

Generating a Step Response in MATLAB. The step function is one of most useful functions in MATLAB for control design. Given a system representation, the response to a step input can be immediately plotted, without need to actually solve for the time response analytically.

1. **Grid :-**

The grid function turns the current axes' grid lines on and off. grid on adds major gridlines to the current axes. grid off removes major and minor grid lines from the current axes. grid(axes\_handle,...) uses the axes specified by axes\_handle instead of the current axes.

1. **Text**

text is the low-level function for creating text graphics objects. Use text to place character strings at specified locations. text(x,y,'string') adds the string in quotes to the location specified by the point (x,y) . text(x,y,z,'string') adds the string in 3-D coordinates.

1. **For**

This MATLAB function executes a group of statements in a loop for a specified number of times.

1. **PID**

This requirement ensures a stable derivative filter pole. C = pid( sys ) converts the dynamic system sys to a parallel form pid controller object. C = pid( Kp ) creates a continuous-time proportional (P) controller with Ki = 0, Kd = 0, and Tf = 0.

**EXERCISE**

**Task # 1 : :** Find the step response of the given closed loop transfer function.

G(s) =

***Coding :***

n=[1]

d=[1 10 21]

g=tf(n,d)

step(g)

y=stepinfo(g)

***Result:***

n = 1

d = 1 10 21

g = 1

---------------

s^2 + 10 s + 21

Continuous-time transfer function.

y RiseTime: 0.8330

SettlingTime: 1.4903

SettlingMin: 0.0431

SettlingMax: 0.0476

Overshoot: 0

Undershoot: 0

Peak: 0.0476

PeakTime: 2.7441

***Figure :***

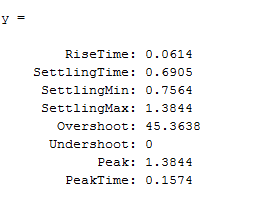
**Task # 2(a):** Find the step response of the given closed loop transfer function with a proportional controller for given values of Kp = 200, 300, 400, assume Ki=Kd=1 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

G(s) =

***Coding :***

|  |  |
| --- | --- |
| kp=200  n=[kp]  d=[1 10 20+kp]  g=tf(n,d)  step(g)  y=stepinfo(g)  hold on  kp=300  n=[kp]  d=[1 10 20+kp] | g=tf(n,d)  step(g)  y=stepinfo(g)  hold on  kp=400  n=[kp]  d=[1 10 20+kp]  g=tf(n,d)  step(g)  y=stepinfo(g) |

***Result:***

******

***Figure :***

**Task # 2(b):** Find the step response of the given closed loop transfer function with a derivative controller for given values of Kd = 5, 10, 15, assume Kp=Ki=1 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

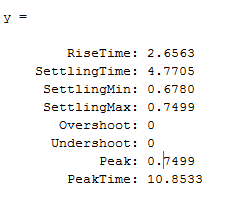
G(s) =

**Coding :**

|  |  |
| --- | --- |
| kd=5  n=[kd]  d=[1 10+kd 20]  g=tf(n,d)  step(g)  hold on  kd=10  n=[kd]  d=[1 10+kd 20 | g=tf(n,d)  step(g)  hold on  kd=15  n=[kd]  d=[1 10+kd 20]  g=tf(n,d)  step(g) |

******

***Result:***

******

***Figure :***

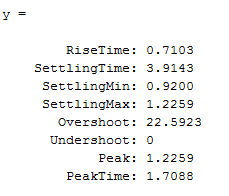
**Task # 2(c):** Find the step response of the given closed loop transfer function with an integral controller for given values of Ki = 20, 30, 40, assume Kp=Kd=1 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

G(s) =

**Coding :**

|  |  |
| --- | --- |
| ki=20  n=[ki]  d=[1 10 20 ki]  g=tf(n,d)  step(g)  y=stepinfo(g)  hold on  ki=30  n=[ki]  d=[1 10 20 ki] | g=tf(n,d)  step(g)  y=stepinfo(g)  hold on  ki=40  n=[ki]  d=[1 10 20 ki]  g=tf(n,d)  step(g)  y=stepinfo(g) |

***Result:***

******

***Figure :***

**Task # 3(a):** Find the step response of the given closed loop transfer function with a proportional controller for given value of Kp = 300.

G(s) =

***Coding :***

******kp=300

n=[kp]

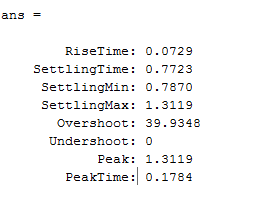
d=[1 10 20+kp]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

******

***Figure :***

**Task # 3(b):**

**Coding :**

kp=300

kd=10

n=[kd kp]

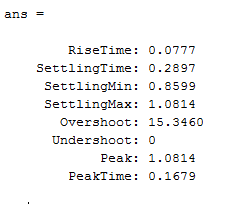
******d=[1 10+kd 20+kp]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

******

***Figure :***

**Task # 3(c):** Find the step response of the given closed loop transfer function with a proportional & derivative controllers for given values of Kp = 300 & Kd=10.

****Coding :**

kp=300

kd=20

n=[kd kp]

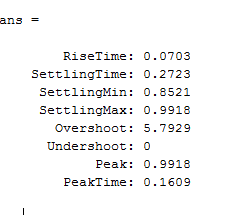
d=[1 10+kd 20+kp]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

******

**Task # 3(d):** Find the step response of the given closed loop transfer function with a proportional & derivative controllers for given values of Kp = 300 & Kd=30.

Compare the results of 3(b), 3(c), 3(d) in terms of Rise Time, Overshoot, Settling Time & Steady State Error. G(s) =

**Coding :**

kp=300

kd=30

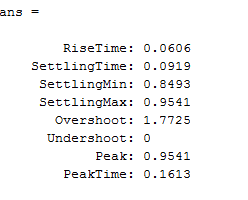
n=[kd kp]

******d=[1 10+kd 20+kp]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

***Figure :***

**Task # 4(a):** Find the step response of the given closed loop transfer function with a proportional & integral controllers for given values of Kp = 30 & Ki=70.

***Coding :***

kp=30

ki=70

******n=[kp+ki]

d=[1 10 20+kp ki]

g=tf(n,d)

step(g)

step info(g)

***Result:***

kp = 30

ki = 70

n = 100

d = 1 10 50 70

g = 100

------------------------

s^3 + 10 s^2 + 50 s + 70

Continuous-time transfer function.

ans =

RiseTime: 1.0448

SettlingTime: 2.0896

SettlingMin: 1.2866

SettlingMax: 1.4283

Overshoot: 0

Undershoot: 0

Peak: 1.4283

PeakTime: 4.2369

***Figure :***

**Task # 4(b):** Find the step response of the given closed loop transfer function with a proportional & integral controllers for given values of Kp = 30 & Ki=80.

**Coding :**

kp=30

ki=80

n=[kp+ki]

d=[1 10 20+kp ki]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

kp = 30

******ki = 80

n = 110

d = 1 10 50 80

g = 110

------------------------

s^3 + 10 s^2 + 50 s + 80

Continuous-time transfer function.

ans =

RiseTime: 0.8268

SettlingTime: 1.6963

SettlingMin: 1.2375

SettlingMax: 1.3741

Overshoot: 0

Undershoot: 0

Peak: 1.3741

PeakTime: 3.0048

***Figure :***

**Task # 4(c):** Find the step response of the given closed loop transfer function with a proportional & integral controllers for given values of Kp = 30 & Ki=90.

Compare the above results in terms of Rise Time, Overshoot, Settling Time & Steady State Error

G(s) =

**Coding :**

kp=30

ki=90

n=[kp+ki]

d=[1 10 20+kp ki]

g=tf(n,d)

step(g)

******stepinfo(g)

***Result:***

kp =30

ki = 90

n = 120

d = 1 10 50 90

g = 120

------------------------

s^3 + 10 s^2 + 50 s + 90

Continuous-time transfer function.

ans = RiseTime: 0.6836

SettlingTime: 1.2320

SettlingMin: 1.2070

SettlingMax: 1.3331

Overshoot: 0

Undershoot: 0

Peak: 1.3331

PeakTime: 2.7896

***Figure :***

**Task # 5(a):** Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 350, Ki=300 & Kd=50.

***Coding :***

kp=350

ki=300

kd=50

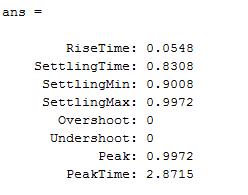
n=[kd kp ki]

d=[1 10+kd 20+kp ki]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

******

***Figure :***

**Task # 5(b):** ): Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 280, Ki=320 & Kd=70.

**Coding :**

kp=280

ki=320

kd=70

n=[kd kp ki]

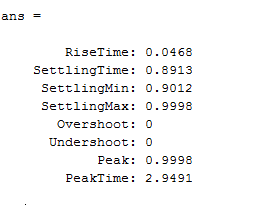
d=[1 10+kd 20+kp ki]

g=tf(n,d)

step(g)

******stepinfo(g

***Result:***

******

***Figure :***

**Task # 5(c):** Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 390, Ki=280 & Kd=90.

Compare the above results in terms of Rise Time, Overshoot, Settling Time & Steady State Error

G(s) =

****Coding :**

kp=390

ki=280

kd=90

n=[kd kp ki]

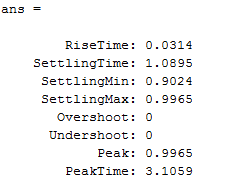
d=[1 10+kd 20+kp ki]

g=tf(n,d)

step(g)

stepinfo(g)

***Result:***

******

***Figure :***

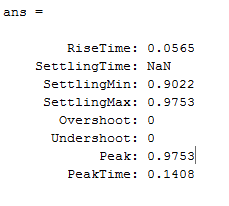
**Task # 6(a):** Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 280, 350, 400, 430, Ki=200 & Kd=40 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

***Coding :***

|  |  |
| --- | --- |
| kp=280  ki=200  kd=40  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  kp=350  ki=200  kd=40  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on | kp=400  ki=200  kd=40  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  kp=430  ki=200  kd=40  n=[kd kp ki]  d=[1 10+k  d 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on |

***Result:***

******

******

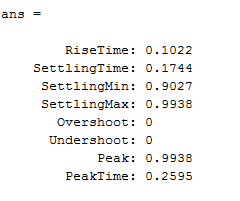
***Figure :***

**Task # 6(b):** Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 220, Ki=260, 310, 360, 400 & Kd=20 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

**Coding :**

|  |  |
| --- | --- |
| kp=220  ki=260  kd=20  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  ki=310  kp=220  kd=20  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on | kp=220  ki=360  kd=20  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  kp=400  ki=220  kd=20  n=[kd kp ki]  d=[1 10+k  d 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on |

***Result:***

******

***Figure :***

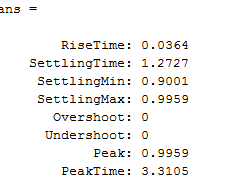
**Task # 6(c):** Find the step response of the given closed loop transfer function with a proportional-integral-derivative controllers for given values of Kp = 360, Ki=240 & Kd=50, 60, 70, 80 and compare the results in terms of Rise Time, Overshoot, Settling Time & Steady State Error.

G(s) =

**Coding :**

|  |  |
| --- | --- |
| kp=360  ki=240  kd=50  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  kp=360  ki=240  kd=60  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on | kp=360  ki=240  kd=70  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on  kp=360  ki=240  kd=80  n=[kd kp ki]  d=[1 10+kd 20+kp ki]  g=tf(n,d)  step(g)  stepinfo(g)  hold on |

***Result:***

******

***Figure :***

**Conclusion:**

In this lab I learnt how to determine the step response of the closed loop system using PID algorithms.