**Open Ended lab ----- PID controller**

**Lab report #10**

****

Fall 2022

CSE-310L Control Systems

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr: Muniba Ashfaq**

**January** 13, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objective(s):**

* To learn about PID controller.
* To learn about kp, ki and kd.
* To learn about pid and pidtune function in matalb.
* To learn how to make pid controller in Simulink.

**PID controller:**

A PID controller is a control system that is used to control a process. The controller takes in a measured value (e.g. temperature), compares it to a desired value (setpoint) and then sends out a control signal to adjust the process (e.g. turn up or down a heater) to bring the measured value closer to the setpoint. The controller uses three types of calculations: proportional, integral, and derivative to make the adjustments. These calculations allow the controller to make fine adjustments to keep the process at the desired setpoint. The controller can be found in many applications like heating, ventilation, and air conditioning (HVAC), robotics, and process control systems.

**Pid function:**

In MATLAB pid function is used to design PID controller.

**Syntax:**

pid(Kp, Ki, Kd)

**Where:**

Kp, Ki, Kd: are the proportional, integral, and derivative gains, respectively. These are adjustable parameters that can be fine-tuned to optimize the controller's performance.

**Pidtune:**

In MATLAB, the "pidtune" function is used to automatically tune the gains of a PID controller for a specific response. The syntax for using the "pidtune" function is as follows:

**Syntax:**

[controller, info] = pidtune(sys, method, options)

**Where:**

**sys:** is the system to be controlled, represented as a transfer function or state-space model.

**method:** is a string specifying the tuning method to use. Common methods include 'pid' (for traditional PID tuning) and 'pidf' (for frequency-domain PID tuning).

**options:** is a structure that contains additional options for the tuning method.

**The function returns two outputs:**

**controller:** is the tuned PID controller object

**info:** is a structure that contains information about the tuning process, such as the tuning method used, the final controller gains, and the closed-loop response of the system.

For example, to automatically tune the gains of a PID controller for a system represented by a transfer function "sys" using the 'pid' method, the following command can be used:

[controller, info] = pidtune(sys, 'pid');

It is worth noting that the "pidtune" function uses a default set of options, however you can customize the options to fit your specific needs by creating an options structure and passing it as an argument.

**Task01:** Design a PID controller for given system both in MATLAB as well as in Simulink and show the step response before after PID controller that you design.

**Source code:**

clc

clear

close all

%system

num=[1];

denum=[1 3 1];

sys=tf(num,denum);

%negative feedback system

feedback\_system=feedback(sys,1);

%step response before PID controller

step(feedback\_system);

%PID controller

kp=1;

ki=1;

kd=1;

PID\_controller=pid(kp,ki,kd);

%negative feedback system with PID controller in series.

%we will connect the PID controller with system in series.

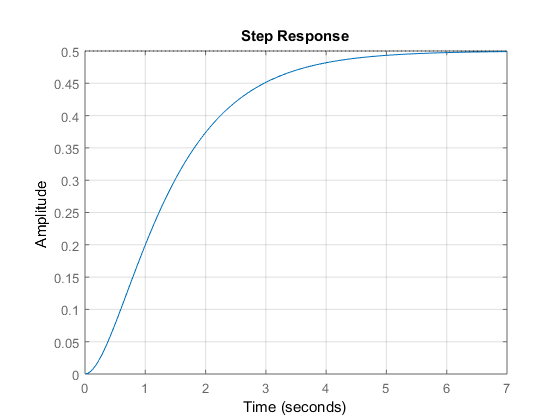
feedback\_system\_pid=feedback(sys\*PID\_controller,1);

%step response after PID controller.

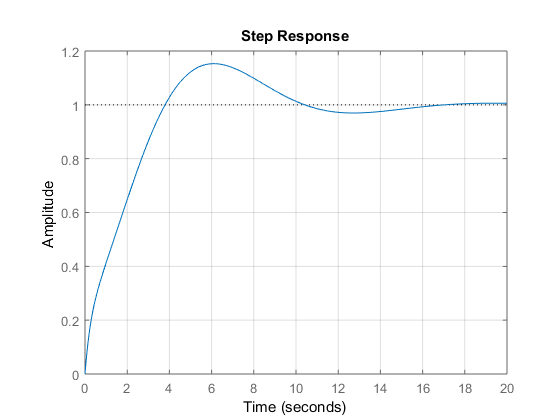
step(feedback\_system\_pid);

**Graph:**

**without PID controller:**

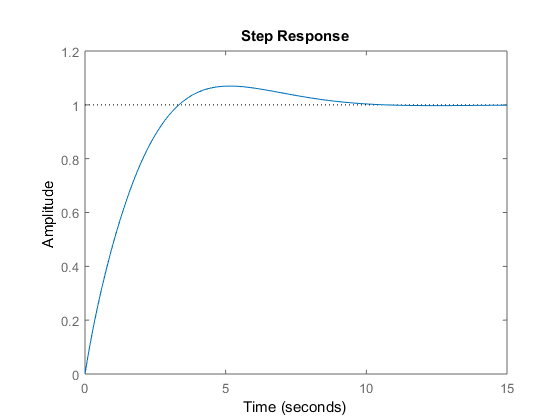


**With PID controller:**



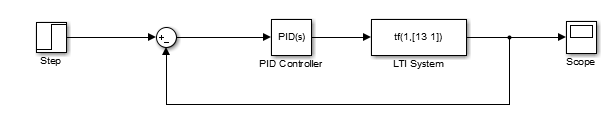
**Description:** As the final value of step response before PID controller is 0.5 which is not the desired output as desired final value of step function is 1. After PID controller we got the desired output 1. We can also change the desired final value or shape of the graph by changing the values of kp, ki and kd.

**Step response using pidtune function:**

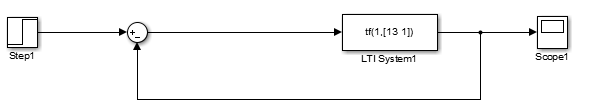


**Simulink Block:**

**Without PID controller:**

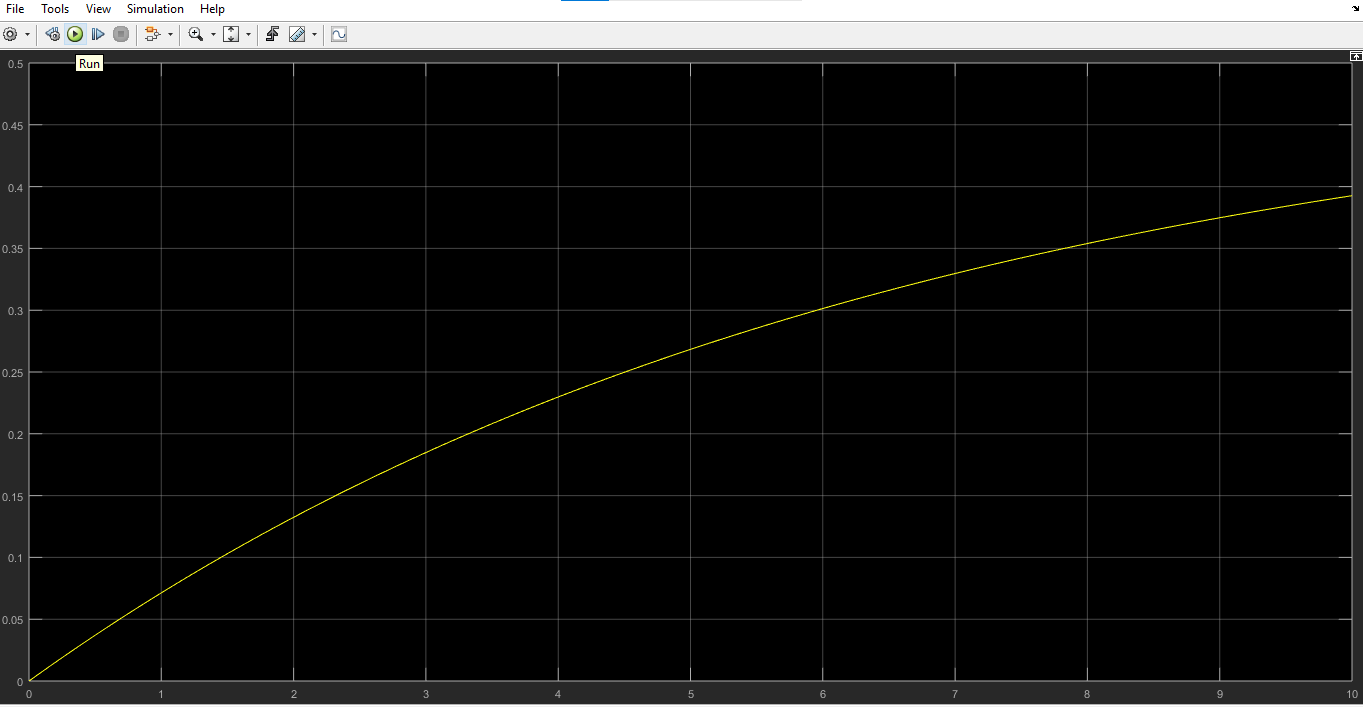


**With PID controller:**

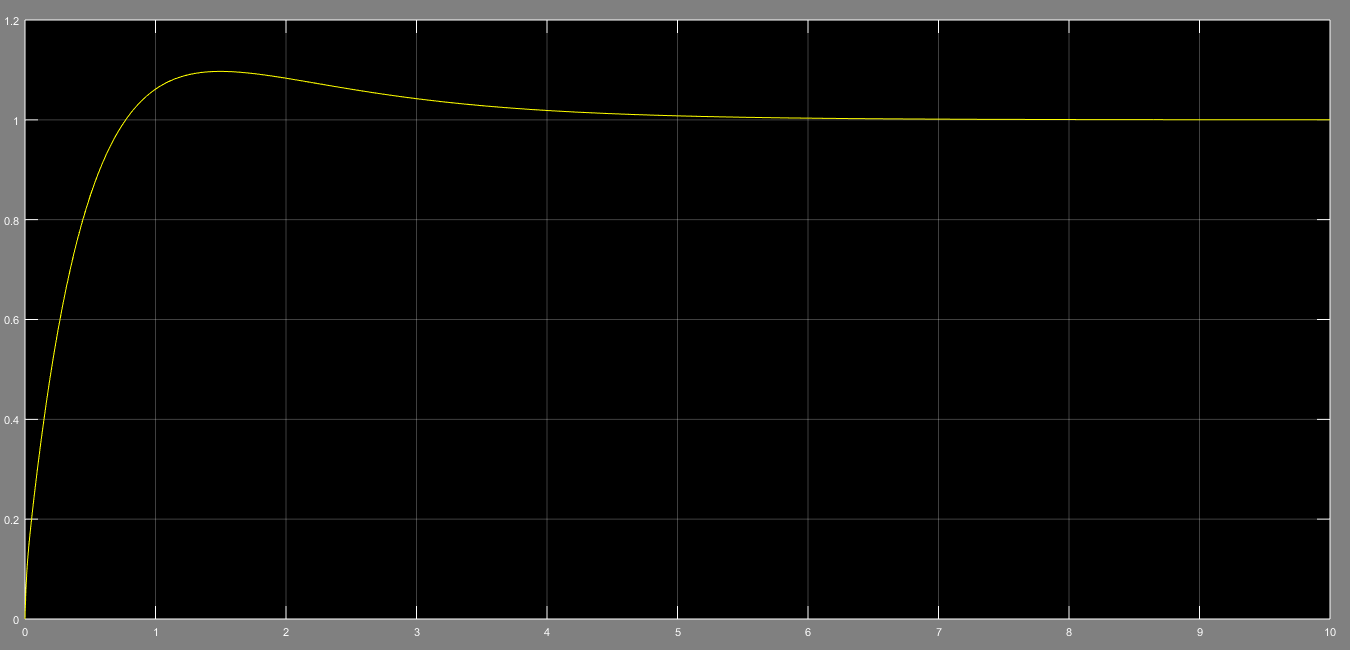


**Graph:**

**Without PID controller:**



**With PID controller:**



**Project Tasks:** Design a PID controller and show all the responses.

**Step response:**

**Source code:**

clc

clear

close all

num=[0.0003 0.0001242 0.000002364];

denum=[1 0.849 0.1274 0.0005188];

G=tf(num,denum);

%pid controller

kp=92.83;

ki=1;

kd=1;

pid\_controller=pid(kp,ki,kd);

%step response before PID controller for compersion.

feedback\_system=feedback(G,1);

[Y\_step,T\_step]=step(feedback\_system);

%step response after PID controller.

pid\_controlled\_feedback\_system=feedback(pid\_controller\*G,1);

[Y\_step\_pid,T\_step\_pid]=step(pid\_controlled\_feedback\_system);

%plotting

subplot(2,1,1);

plot(T\_step,Y\_step,'b','LineWidth',2);

title('step response before PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

subplot(2,1,2);

plot(T\_step\_pid,Y\_step\_pid,'r','LineWidth',2);

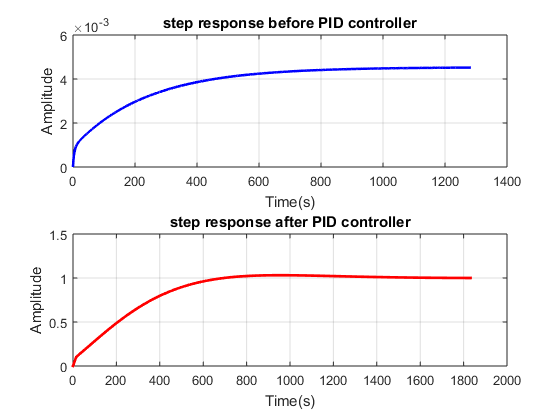
title('step response after PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

**Graph:**



**Impulse response:**

**Source code:**

clc

clear

close all

num=[0.0003 0.0001242 0.000002364];

denum=[1 0.849 0.1274 0.0005188];

G=tf(num,denum);

%pid controller

kp=9122576.98335;

ki=1;

kd=1;

pid\_controller=pid(kp,ki,kd);

% Impulse response before PID controller for compersion.

t = 0:0.01:10;

feedback\_system=feedback(G,1);

impulse\_signal = [1, zeros(1, length(t)-1)];

[Y\_impulse, T\_impulse] = lsim(feedback\_system, impulse\_signal, t);

% Impulse response after PID controller.

pid\_controlled\_feedback\_system\_i=feedback(pid\_controller\*G,1);

[Y\_impulse\_pid, T\_impulse\_pid] = lsim(pid\_controlled\_feedback\_system\_i, impulse\_signal, t);

%plotting

subplot(2,1,1);

plot(T\_impulse, Y\_impulse, 'r', 'LineWidth', 2);

title('Impulse response without PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

axis([-2 12 0 0.0000044]);

grid on

subplot(2,1,2);

plot(T\_impulse\_pid, Y\_impulse\_pid, 'r', 'LineWidth', 2);

title('Impulse respnse with PID controller');

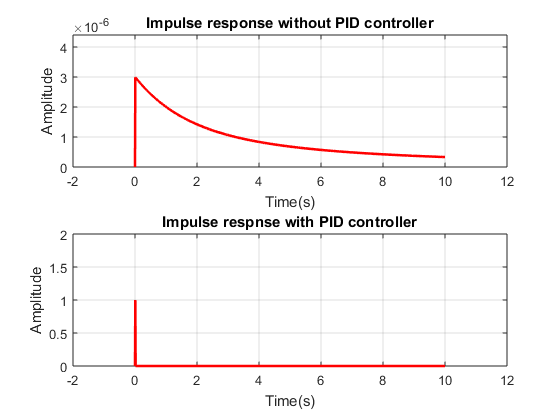
xlabel('Time(s)');

ylabel('Amplitude');

axis([-2 12 0 2]);

grid on

**Graph:**



**Ramp response:**

**Source code:**

clc

clear

close all

num=[0.0003 0.0001242 0.000002364];

denum=[1 0.849 0.1274 0.0005188];

G=tf(num,denum);

%pid controller

kp=2122332.98335;

ki=0;

kd=0;

pid\_controller=pid(kp,ki,kd);

% Ramp response before PID controller

t = 0:0.01:10;

feedback\_system=feedback(G,1);

ramp\_signal = t;

[Y\_ramp, T\_ramp] = lsim(feedback\_system, ramp\_signal, t);

% Ramp response after PID controller.

pid\_controlled\_feedback\_system=feedback(pid\_controller\*G,1);

[Y\_ramp\_pid, T\_ramp\_pid] = lsim(pid\_controlled\_feedback\_system, ramp\_signal, t);

%plotting

subplot(2,1,1);

plot(T\_ramp, Y\_ramp, 'r', 'LineWidth', 2);

title('Ramp response without PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

subplot(2,1,2);

plot(T\_ramp\_pid, Y\_ramp\_pid, 'r', 'LineWidth', 2);

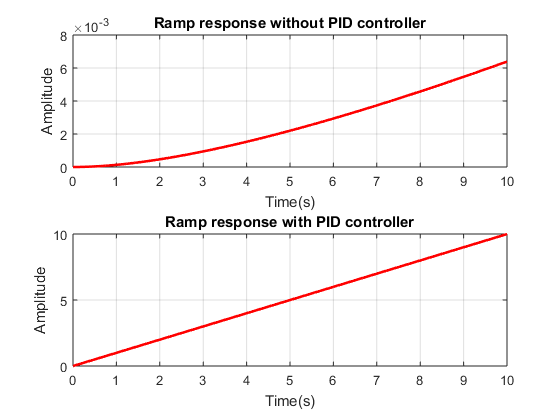
title('Ramp response with PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

**Graph:**



**Parabola response:**

**Source code:**

clc

clear

close all

% Transfer function of the system

num=[0.0003 0.0001242 0.000002364];

denum=[1 0.849 0.1274 0.0005188];

G=tf(num,denum);

% Design PID controller

kp=2122337.98335;

ki=0;

kd=0;

pid\_controller=pid(kp,ki,kd);

% Response to parabola signal without PID controller

t = 0:0.01:10;

parabola\_signal = t.^2;

feedback\_system=feedback(G,1);

[Y\_parabola, T\_parabola] = lsim(feedback\_system, parabola\_signal, t);

% Response to parabola signal with PID controller

pid\_controlled\_feedback\_system=feedback(pid\_controller\*G,1);

[Y\_parabola\_pid, T\_parabola\_pid] = lsim(pid\_controlled\_feedback\_system, parabola\_signal, t);

% Plot responses

figure

subplot(2,1,1);

plot(T\_parabola, Y\_parabola, 'r', 'LineWidth', 2);

title('parabola response without PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

subplot(2,1,2);

plot(T\_parabola\_pid, Y\_parabola\_pid, 'r', 'LineWidth', 2);

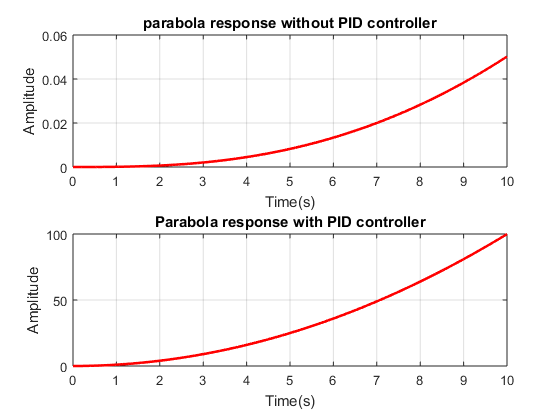
title('Parabola response with PID controller');

xlabel('Time(s)');

ylabel('Amplitude');

grid on

**Graph:**



**The end**