Industrial Applications of Fuzzy Logic
Homework-2

March 18, 2008

Contents

1 Introduction

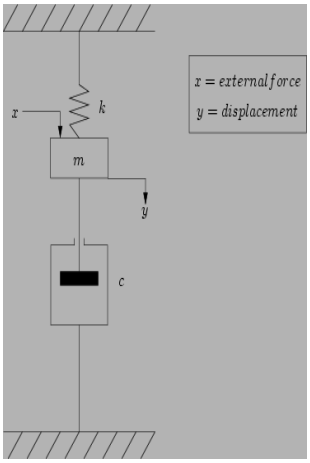
In this homework a conventional PD and a Fuzzy-PD controller will be designed and tuned for a chosen system. Then for certain non-linearities, a robustness test will be performed and results will be given

2 Initial Design

First thing to do is to design two controllers for the chosen system.

1. Conventional PD controller
2. A fuzzy PD controller

Following system is chosen to be controlled:



Differantial equation of the system:

$$m\ddot{y} + b\dot{y} + k\dot{y} = x \tag{1}$$

Laplace transformation is used to obtain the transfer function representation of the system:

$$\frac{Y(s)}{X(s)} = \frac{1}{ms^2 + bs + k} \tag{2}$$

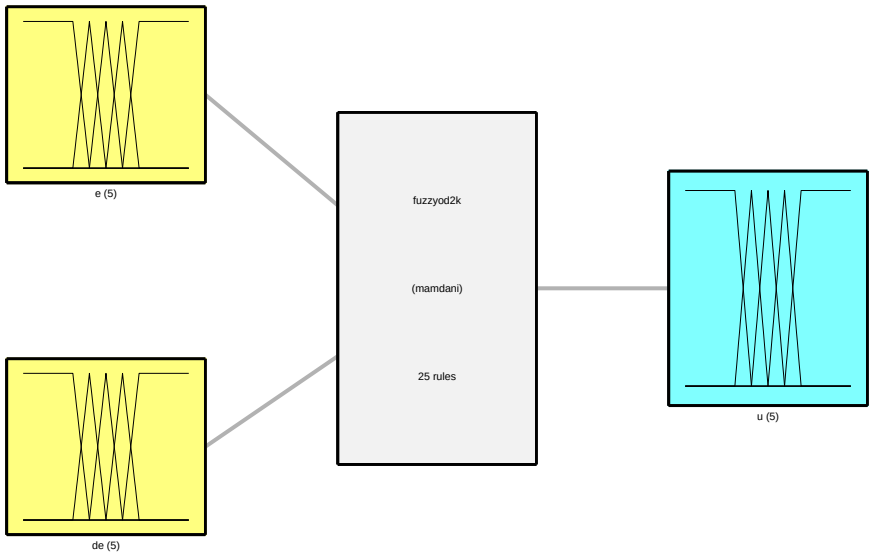
Here choosing coefficients as follows, Transfer function is obtained:
m=1;k=1;b=0.5

$$\frac{Y(s)}{X(s)} = \frac{1}{s^2 + 0.5s + 1} \tag{3}$$

2.1 Fuzzy-PD Design

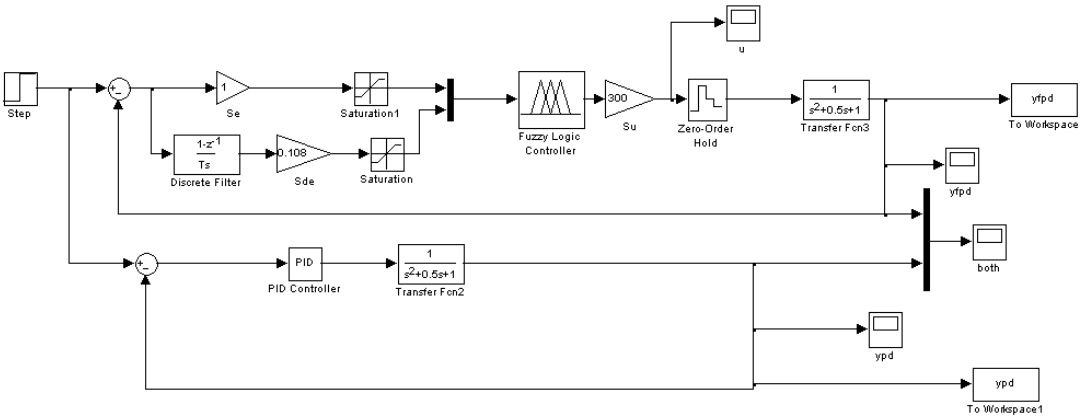
Following rule table is constituted for Fuzzy-PD controller:

U	NM	N	Z	P	PM
NM	NM	NM	N	N	Z
N	NM	N	N	Z	P
Z	N	N	Z	P	P
P	N	Z	P	P	PM
PM	Z	P	P	PM	PM



2.2 Simulink Block Diagram and Controller Tuning

Here is the siulation diagram builded on simulink:



2.2.1 Tuning parameter effects

Conventional PD:
For conventional PD controller there are two control coefficents to tune: P and D.

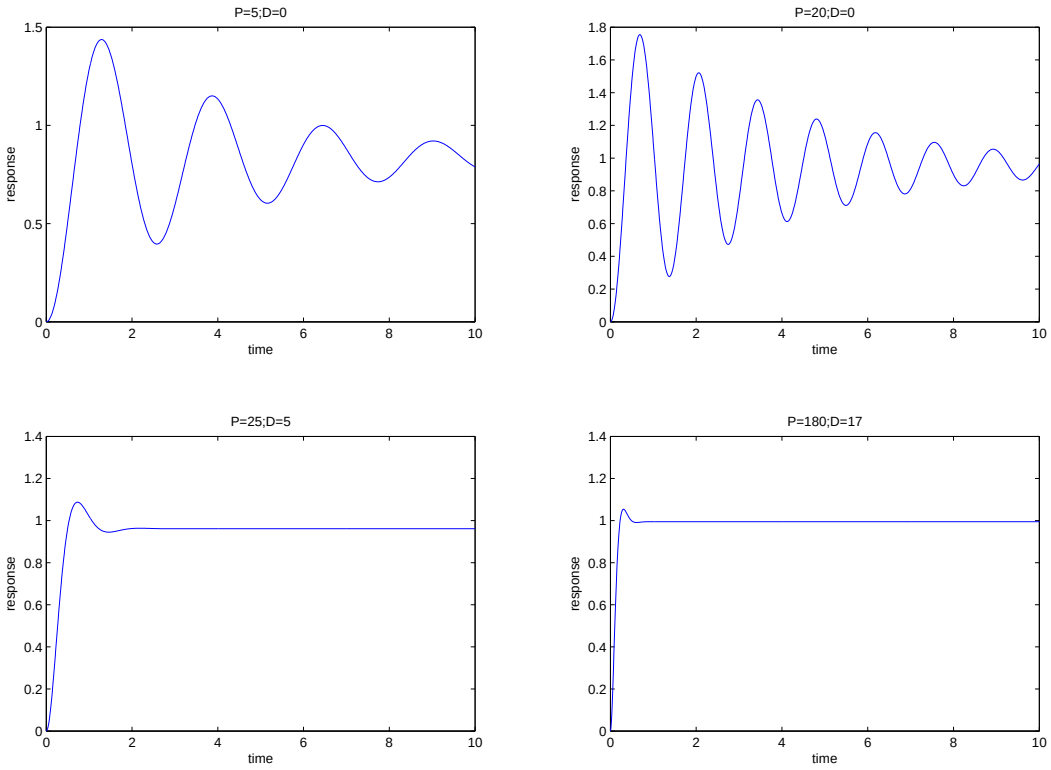
- Increasing the P we obtain a short rise time but meanwhile we increase overshoot of the system. The value of the P also determines the steady state response (value) of the system!
- Increasing D, we decrease the overshoot.

Fuzzy-PD:
For Fuzzy-PD controller, there are more than 2 tuning coefficents. Se, Sde, Su are global tuning parameters and also the MFs‘ shapes are again (local)tuning parameters. Se has nearly same but limited effect as P in conventional PD controller. It make the response quicker and decreases the steady state error. Sde is also very similar to D. It decreases the osilations and overshoot. Su has both properties inside. Increasing Su somehow means increasing Se and Sde both. It decreases the response time of the system, decreases the steady state error and also for large values of it (I just write this for my system here) you may see it also decreases the maximum overshoot somehow. But this point is not very clear and not true for all systems. We may say that Su has similar effects with Se and Sde both, but its dominant characteristic is like Se.

2.2.2 Parameter tuning trials and results

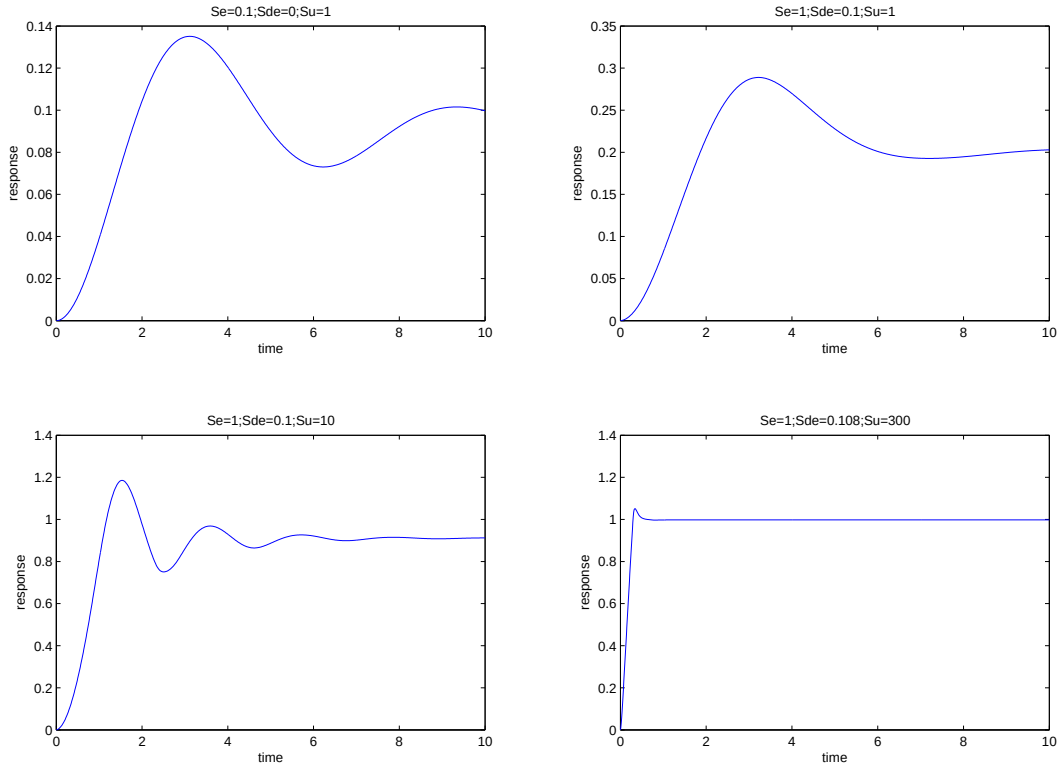
Here are the some trials to tune conventional PD and Fuzzy-PD controllers:

Conventional PD



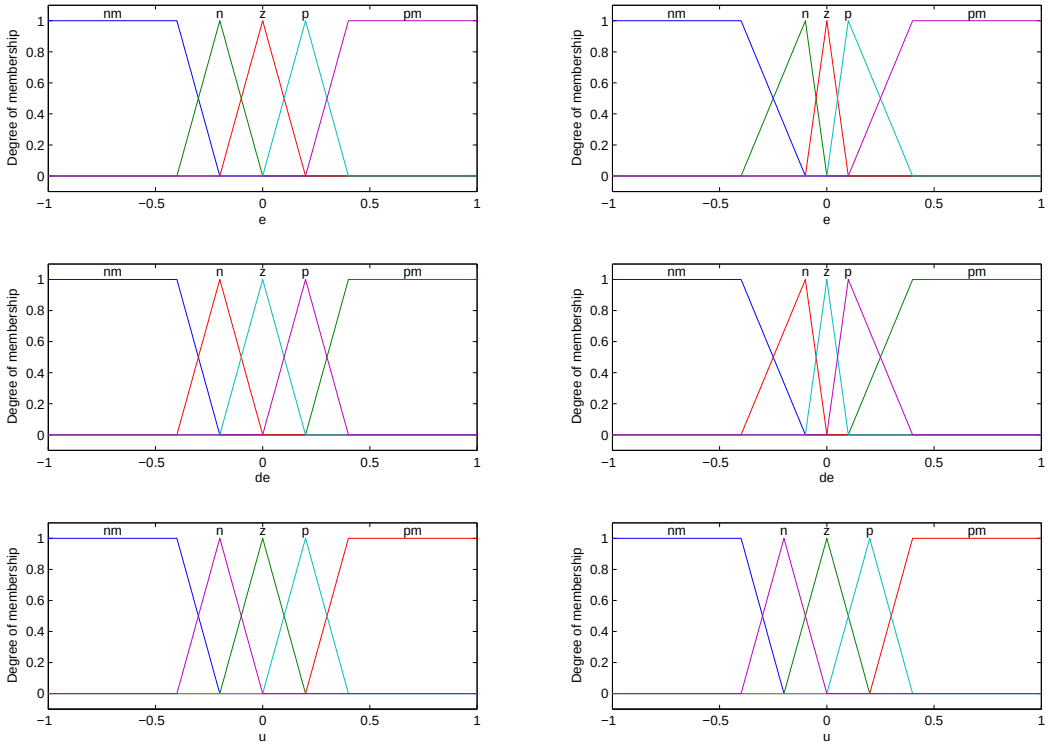
Conventional PD tuning trials

Fuzzy-PD

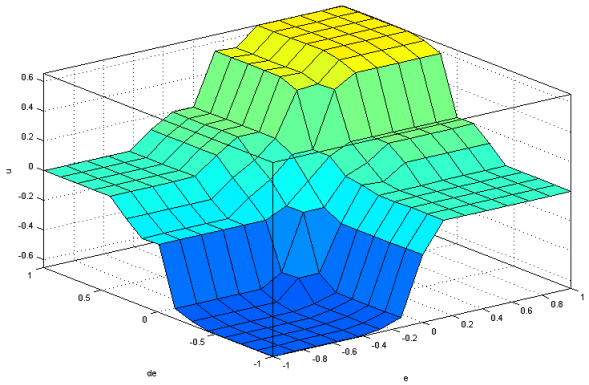


Fuzzy PD tuning trials

To decrease the steady state error and to make the system quicker, also the shapes of MFs can be manipulated. The next figure shows such a manipulation



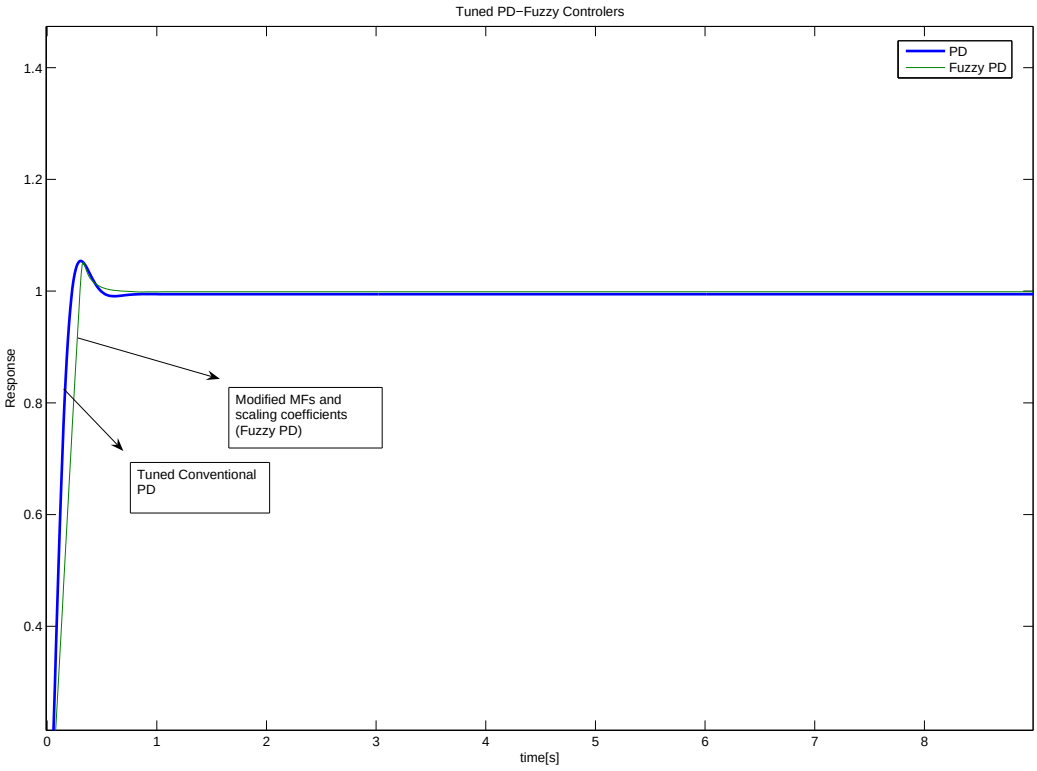
MFs tuning (very initial design - a step ahead)



Resultant command surface

2.3 Comparison of Controllers

After tuning, similar performance values are reached for both controllers



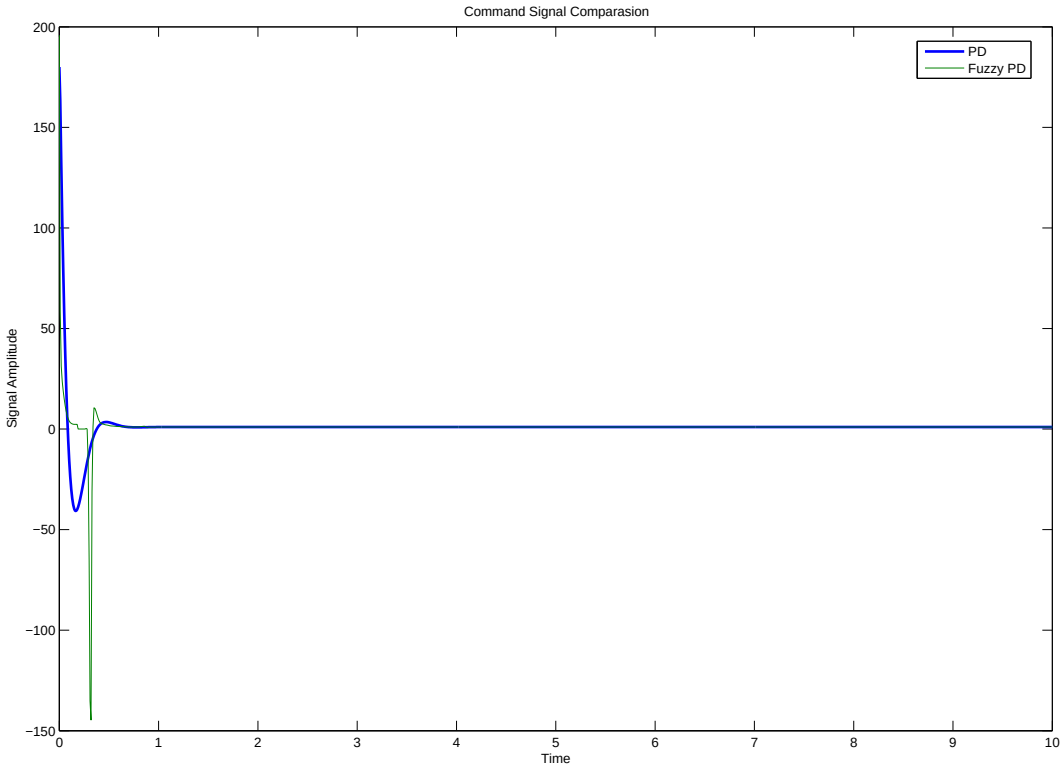
Step responce of controllers

Comparison table:

	Fuzzy PD	PD
Max. Overshoot	0.052	0.054
Sum of Abs. Err.	17.79	17.27
Steady State Err.	0.0012	0.0055
Rise Time	0.3	0.23
Settling Time	0.4	0.41

2.3.1 Generated command signal magnitudes

What is also important in real life is the magnitude of the generated command signal. I will inspect this situation putting a saturation to actuator later. Here I just give the command signal graphics below:



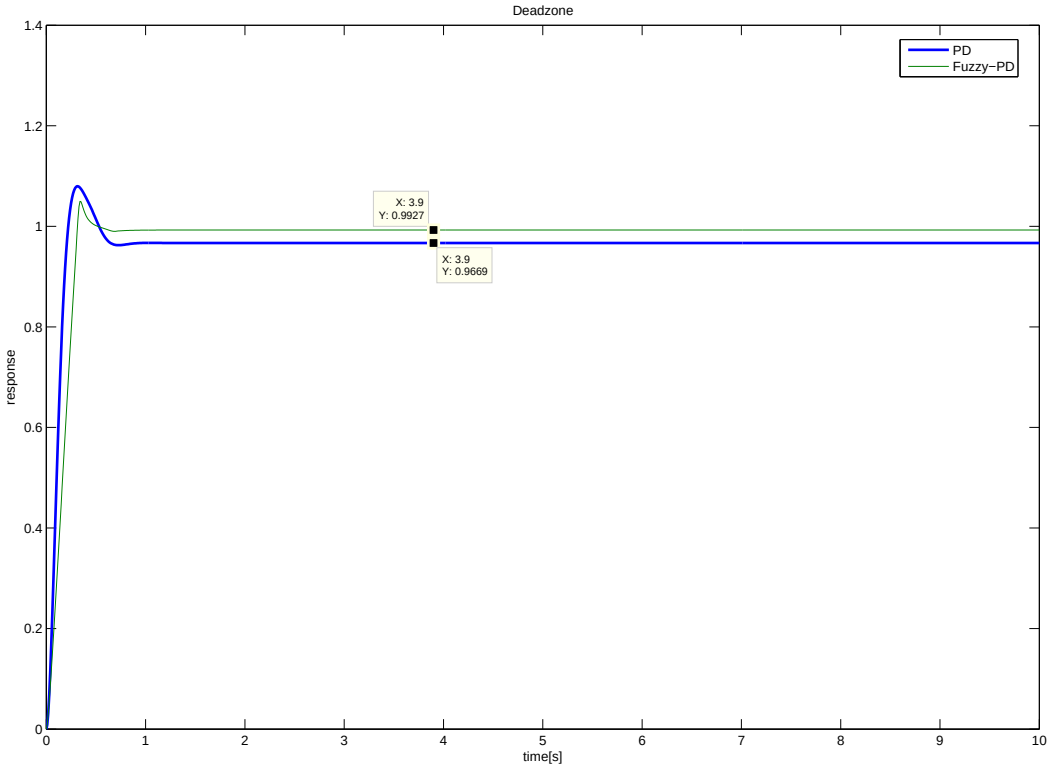
Step responce of controllers

3 Robustness Tests

After initial design and tuning phase we test both controllers for their robustness. First we add a dead-zone to command signal u. Then we assume there is a noise on command signal and last we assume the system parameters are changed. We compare the controller responses for these three nonlinearities.

3.1 Deadzone Response

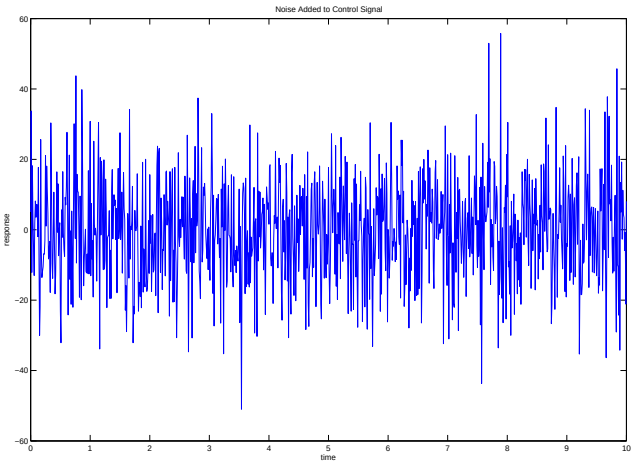
It is obvious that Fuzzy PD is more robust than normal PD against deadzone. Steady state error and maximum overshoot is less than conventional PD



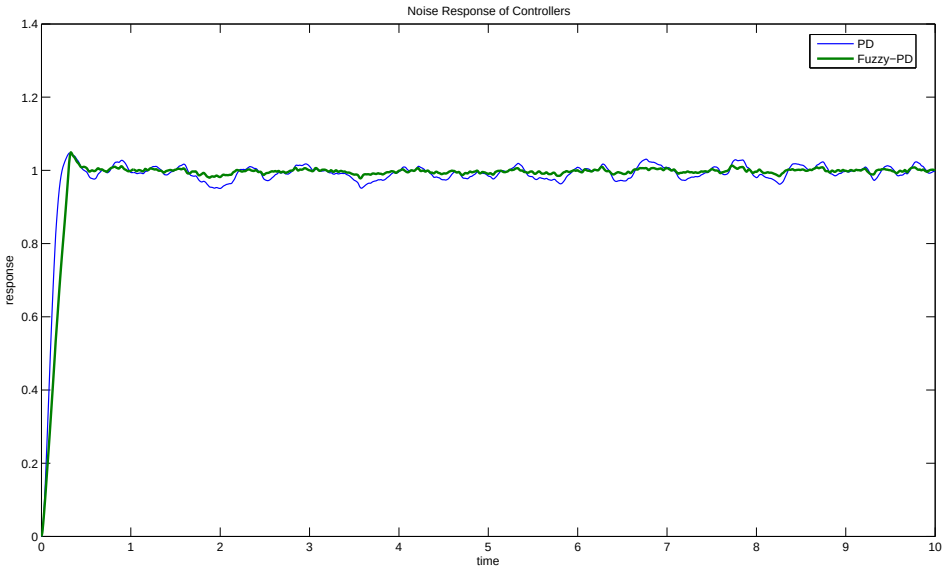
Deadzone response comparison

3.2 Noise addition to command signal

Again Fuzzy PD wins again conventional PD, looking the step responce results.



Noise added to signal



Step responses with noise

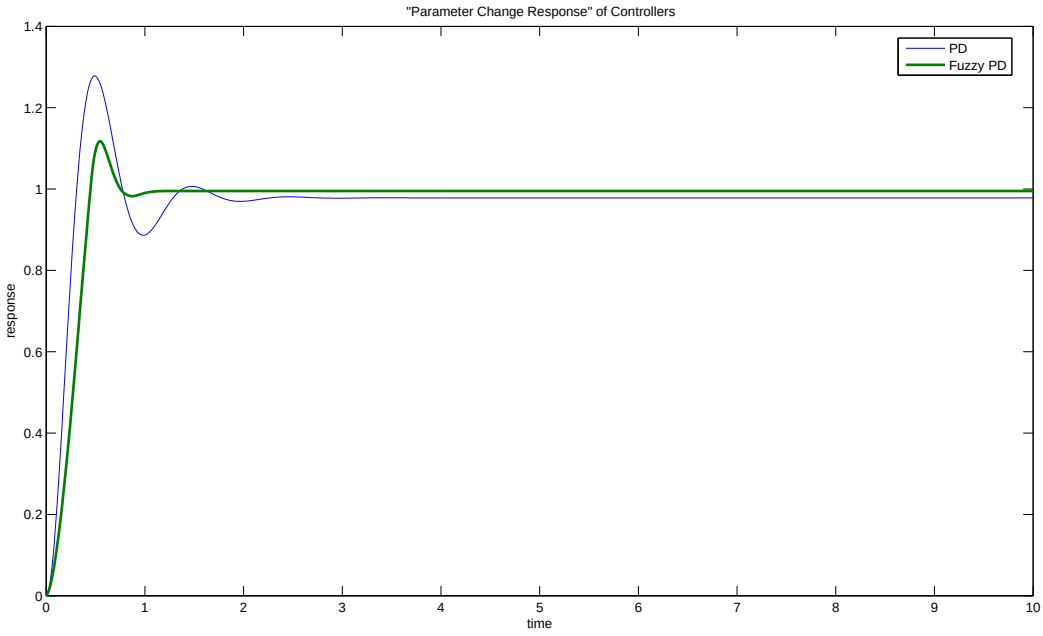
3.3 Parameter change

we initially designed and tuned our controllers for the original system below:

$$\frac{Y(s)}{X(s)} = \frac{1}{s^2 + 0.5s + 1} \tag{4}$$

Now we assume parameters changed like below:
$m \to 4$; $b \to 2$; $k \to 4$
and our system turns out to be:

$$\frac{Y(s)}{X(s)} = \frac{1}{4s^2 + 2s + 4} \tag{5}$$



Step responses against changed parameters

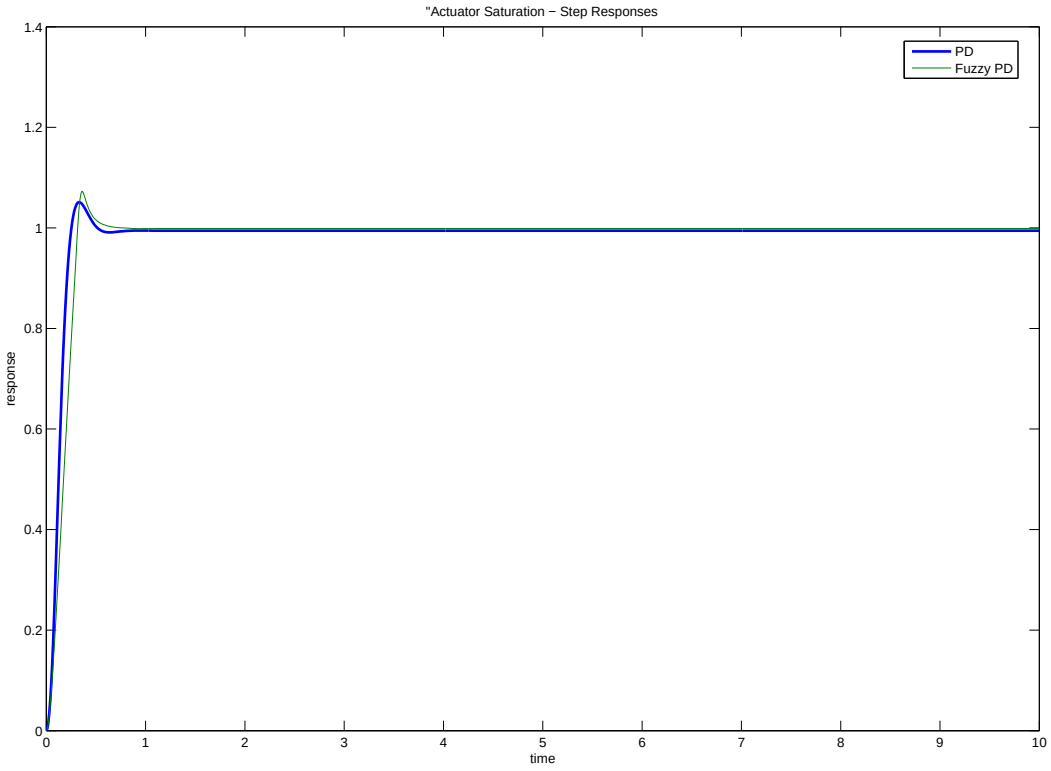
Inspection the above simulation result we can clearly say that Fuzzy-PD wins against conventinal PD. Less maximum overshoot,less osilations and less steady state error.

3.4 Actuator Saturation

In real world we have limited actuators. So for large values of command signal, actuator will not be able to impose what it has to impose to the system. What

will be the response of our controllers for such a case is just below.

I limited the command value between 75 and -75 (means above and below this value, actuator saturates)



Step responses against actuator saturation

In this case, as seen on the above figure, conventional PD comes out to be a little better than Fuzzy-PD

4 Conclusion

In this homework, we designed two controllers, a conventional PD and a fuzzy PD controller. After the initial design, we test them for their robustness against nonlinearities such as deadzone, noise, parameter change and actuator saturation. For the first three, Fuzzy-PD is superior to the cnventional PD controller. Looking this results I may say that providing a good design, a Fuzzy-PD is generally superior to a conventional PD controller in many ways.

A Matlab codes

There are two main m files to evalute grapichs and response performance

A.1 Performance function code

```
function [rise_time,oversht,sse,S_abs_err,settling_time]=performance(y,t)
Tsmp=t(2)-t(1);
r=1; while y(r)<1.0001;r=r+1;end
rise_time=(r-1)*Tsmp

oversht=max(y)-1
s=length(t);while y(s)>0.98 & y(s)<1.02;s=s-1;end
 settling_time=(s-1)*Tsmp

S_abs_err=sum(abs(1-y))
sse=1-y(length(t))
end
```

A.2 General graphic m file

```
 %Fuzzy tuning
SUBPLOT(2,2,1),plot(t,yfpd1),
Title('Se=0.1;Sde=0;Su=1'),xlabel('time'),ylabel('response')


SUBPLOT(2,2,2),plot(t,yfpd2),
Title('Se=1;Sde=0.1;Su=1'),xlabel('time'),ylabel('response')
SUBPLOT(2,2,3),plot(t,yfpd3),
Title('Se=1;Sde=0.1;Su=10'),xlabel('time'),ylabel('response')
SUBPLOT(2,2,4),plot(t,yfpd4),
Title('Se=1;Sde=0.108;Su=300'),xlabel('time'),ylabel('response')

%PD tuning

SUBPLOT(2,2,1),plot(t,ypd1),Title('P=5;D=0'),xlabel('time'),ylabel('response')


SUBPLOT(2,2,2),plot(t,ypd2),Title('P=20;D=0'),xlabel('time'),ylabel('response')
SUBPLOT(2,2,3),plot(t,ypd3),Title('P=25;D=5'),xlabel('time'),ylabel('response')
SUBPLOT(2,2,4),plot(t,ypd4),Title('P=180;D=17'),xlabel('time'),ylabel('response')

%initial command signals
plot(t,uypd1,t,uyfpd1),
Title('Command Signal Comparasion'),xlabel('Time'),ylabel('Signal Amplitude')
```

```
%Step responses of controllers:

plot(t,ypd4,t,ymfpd)

%performance of controllers:
performance(ypd4,t)
performance(ymfpd,t)

%fuzzy sistem grafii
plotfis(fuzzyod2k)

%updated mfs
subplot(3,2,1),plotmf(fuzzyod2k,'input',1)
subplot(3,2,2),plotmf(fuzzyod2kv3,'input',1)
subplot(3,2,3),plotmf(fuzzyod2k,'input',2)
subplot(3,2,4),plotmf(fuzzyod2kv3,'input',2)
subplot(3,2,5),plotmf(fuzzyod2k,'output',1)
subplot(3,2,6),plotmf(fuzzyod2kv3,'output',1)

%surface
surfview(fuzzyod2k)

%deadzone -5 +5
plot(t,dzypd,t,dzyfpd),Title('Deadzone')
,xlabel('time[s]'),ylabel('response')

%noise added to u
plot(t,noises),Title('Noise Added to Control Signal')
,xlabel('time'),ylabel('response')
plot(t,noiseypd,t,noiseyfpd),
Title('Noise Response of Controllers'),xlabel('time'),ylabel('response')

%parameter change all*4 m=4 b=2 k=4
plot(t,parypd,t,paryfpd),
Title('"Parameter Change Response" of Controllers'),xlabel('time'),ylabel('response')

%Actuator Saturation
plot(t,acsatypd,t,acsatyfpd),
Title('"Actuator Saturation - Step Responses '),xlabel('time'),ylabel('response')
```