University Of Balamand

Faculty of Engineering

Fuzzy Logic Control

Tuning Of Fuzzy PID Controllers

Presented to: Dr. Issam Dagher

Presented by: Kifah DAHER

Date: 19-12-2012

Introduction:

Fuzzy controllers are nonlinear, it is more difficult to set the controller gains compared to proportional-integral-derivative (PID) controllers. This assignment proposes a design procedure and a tuning procedure that carries tuning rules from the PID domain over to fuzzy single-loop controllers. The idea is to start with a tuned, conventional PID controller, replace it with an equivalent linear fuzzy controller, make the fuzzy controller nonlinear, and eventually fine-tune the nonlinear fuzzy controller. This is relevant whenever a PID controller is possible or already implemented.

Design Strategy:

The design strategy, which makes use of known PID design techniques before implementing the fuzzy controller:

1. Tune a PID controller
2. Replace it with an equivalent linear fuzzy controller
3. Make the fuzzy controller nonlinear
4. Fine-tune it

Finding parameters for a fuzzy PID or PD+I controller could be done as defined by Ziegler-Nichols.

a. Increase the proportional gain until the system oscillates; that gain is the ultimate gain K_u.
b. Read the time between peaks T_u at this setting.
c. Table 1 gives approximate values for the controller gains.

Controller	K_p	T_i	T_d
P	$0.5K_u$		
PI	$0.45K_u$	$T_u/1.2$	
PID	$0.6K_u$	$T_u/2$	$T_u/8$

Figure 1: The Ziegler-Nichols rules (frequency response method)

Assuming the following transfer function for our system

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

Experimentally K_u =8, T_u= 15/4 =>

K_p= 4.8

T_i= 15/8

T_d= 15/32

The input universes must be large enough for the inputs to stay within the limits (no saturation). Each input family should contain a number of terms, designed such that the sum of membership values for each input is 1. This can be achieved when the sets are triangular and cross their neighbor sets at the membership value $\mu = 0.5$; their peaks will thus be equidistant. Any input value can thus be a member of at most two sets, and its membership of each is a linear function of the input value.

The number of terms in each family determines the number of rules, as they must be the AND combination (outer product) of all terms to ensure completeness. The output sets should preferably be singletons the output sets may also be triangles, symmetric about their peaks, but singletons make defuzzification simpler.

To ensure linearity, we must choose the algebraic product for the connective AND. Using the weighted average of rule contributions for the control signal (corresponding to center of gravity defuzzification), the denominator vanishes, because all firing strengths add up to 1.

The gains are transferred from PID to Fuzzy controller. The below controller will be considered in our study.

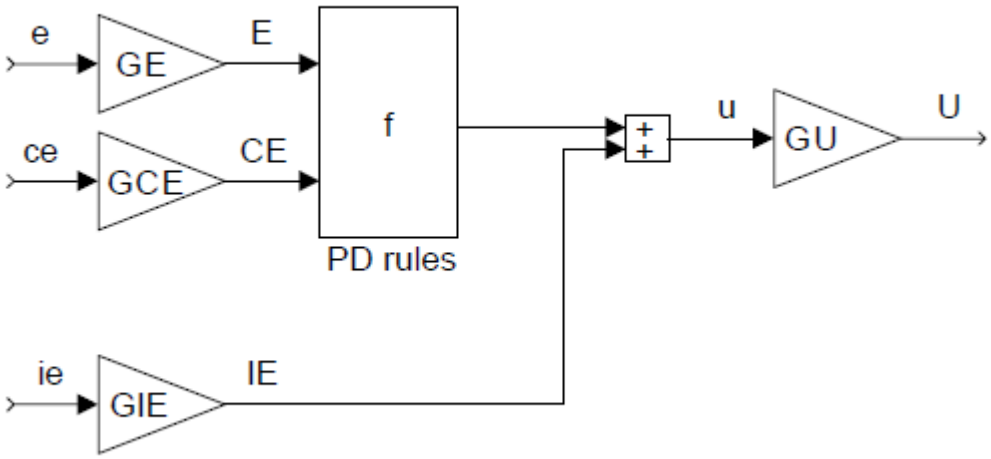

Figure 2: Fuzzy PD+I controller (FPD+I).

The gains are related as follows:

$$\begin{aligned} GE * GU &= K_p \\ \frac{GCE}{GE} &= T_d \\ \frac{GIE}{GE} &= \frac{1}{T_i} \end{aligned}$$

In our example we set GE= 100

- ⇨ GU = K_p/GE= 4.8/100 = 0.048
- ⇨ GCE = GE * T_d = 100*15/32 = 46.875
- ⇨ GIE = GE/T_i = 100*8/15 = 53.33

We have used linear triangular inputs to simulate a linear system and Gaussian inputs to simulate a nonlinear system.

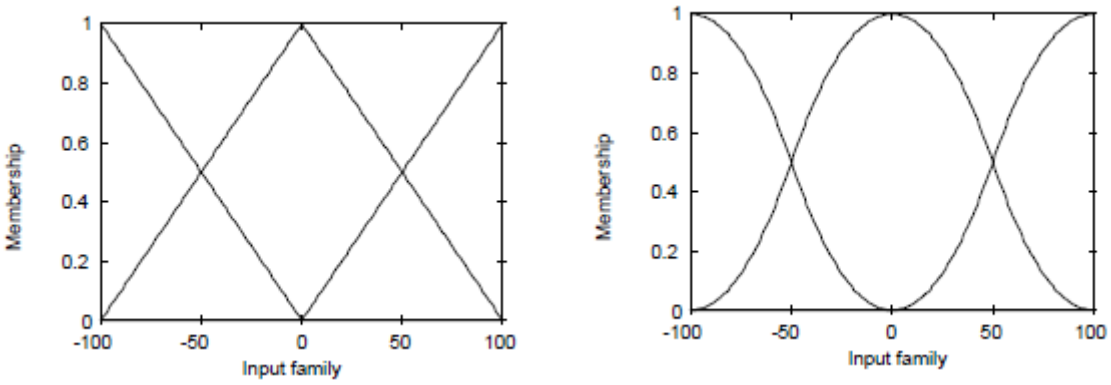

Figure 3: linear triangular M.F. (left) Gaussian M.F. (right)

The triangular M.F. gives linear surface whereas the Gaussian gives a bumpy surface.

The rule base are the following:

1. If *error* is Neg and *change in error* is Neg then output is -200
2. If *error* is Neg and *change in error* is Zero then output is -100
3. If *error* is Neg and *change in error* is Pos then output is 0
4. If *error* is Zero and *change in error* is Neg then output is -100
5. If *error* is Zero and *change in error* is Zero then output is 0
6. If *error* is Zero and *change in error* is Pos then output is 100
7. If *error* is Pos and *change in error* is Neg then output is 0
8. If *error* is Pos and *change in error* is Zero then output is 100
9. If *error* is Pos and *change in error* is Pos then output is 200

Simulation:

Different simulations were done to simulate the design:

(a) Using mamdani and triangular membership function input. The output is also triangular symmetric narrow to the center having the aspect of Singleton.

(b) Using Sugeno to implement constant singleton output.

(c) Using mamdani with Gaussian triangular membership function to have a nonlinear surface.

A- Triangular membership functions input.

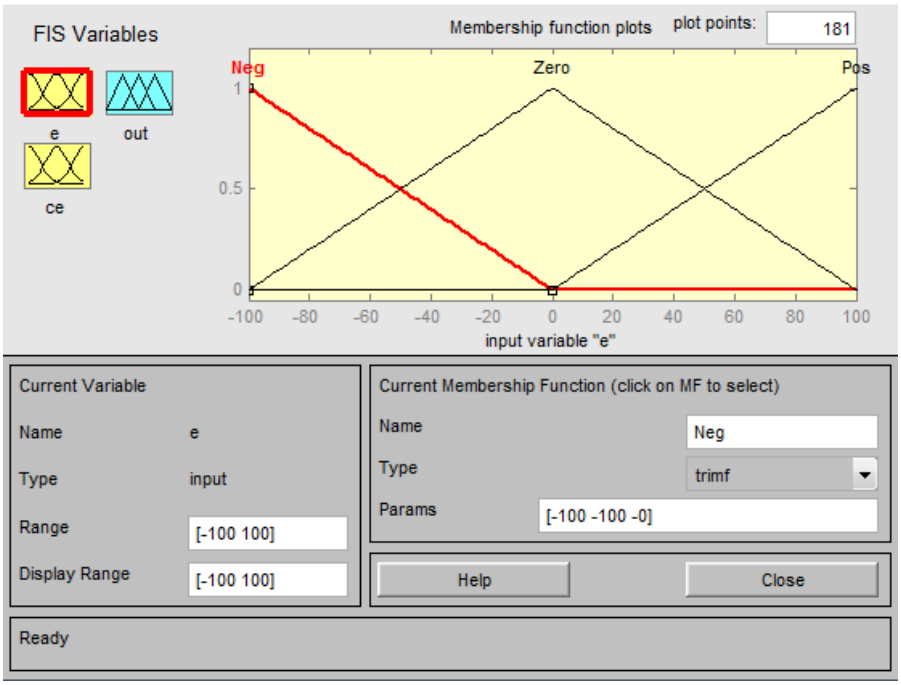

Figure 4: Triangular M.F. Negative, Zero& Positive

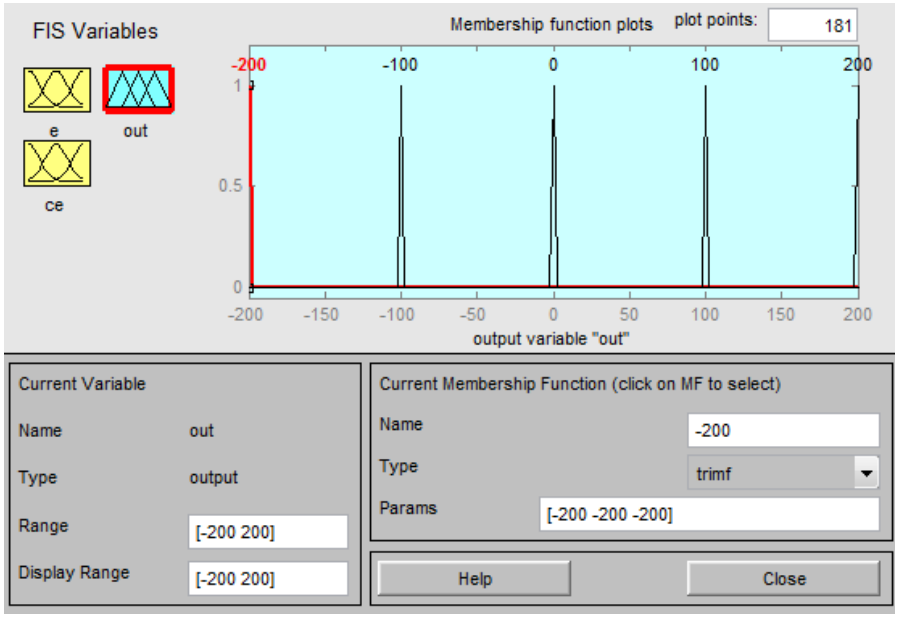

Figure 5: Singleton output

The rule base is the following for all simulations which consist of 9 rules:

de/dt / e	N	Z	P
N	NB	N	Z
Z	N	Z	P
P	Z	P	PB

1. If (e is Neg) and (ce is Neg) then (out is -200) (1)
2. If (e is Neg) and (ce is Zero) then (out is -100) (1)
3. If (e is Neg) and (ce is Pos) then (out is 0) (1)
4. If (e is Zero) and (ce is Neg) then (out is -100) (1)
5. If (e is Zero) and (ce is Zero) then (out is 0) (1)
6. If (e is Zero) and (ce is Pos) then (out is 100) (1)
7. If (e is Pos) and (ce is Neg) then (out is 0) (1)
8. If (e is Pos) and (ce is Zero) then (out is 100) (1)
9. If (e is Pos) and (ce is Pos) then (out is 200) (1)

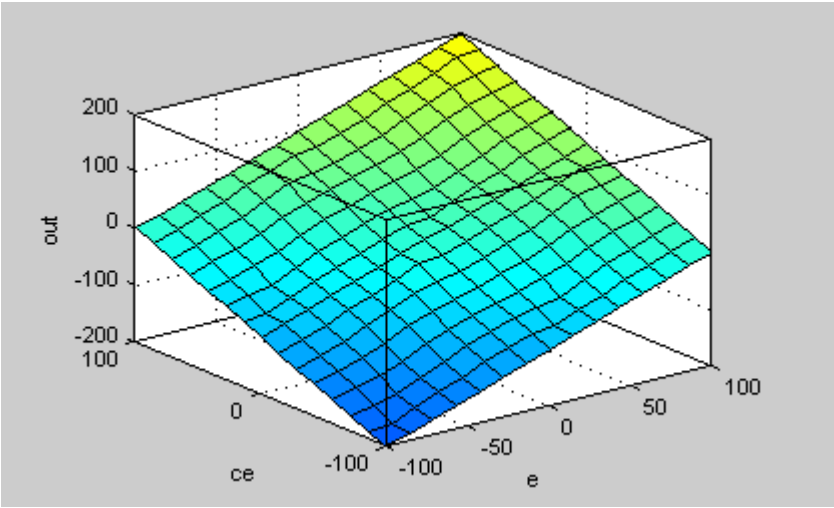

Figure 6: The control surface

As we can see the control surface is almost linear and smooth equivalent to the summation of two inputs the error and the change of error.

B- Singleton output using Sugeno

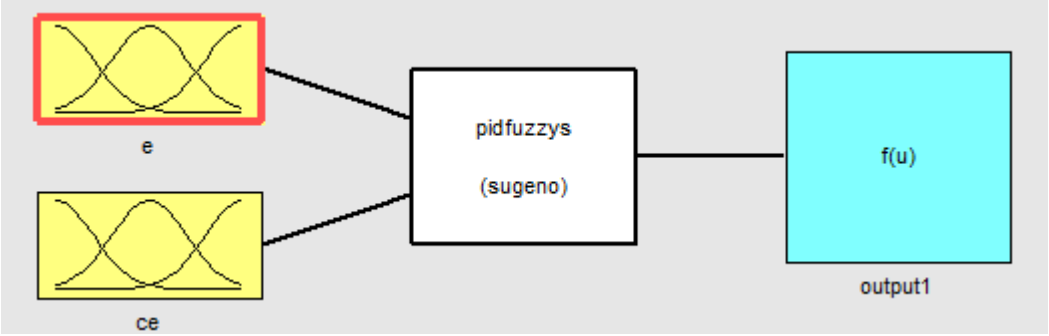

Figure 7: Sugeno FIS

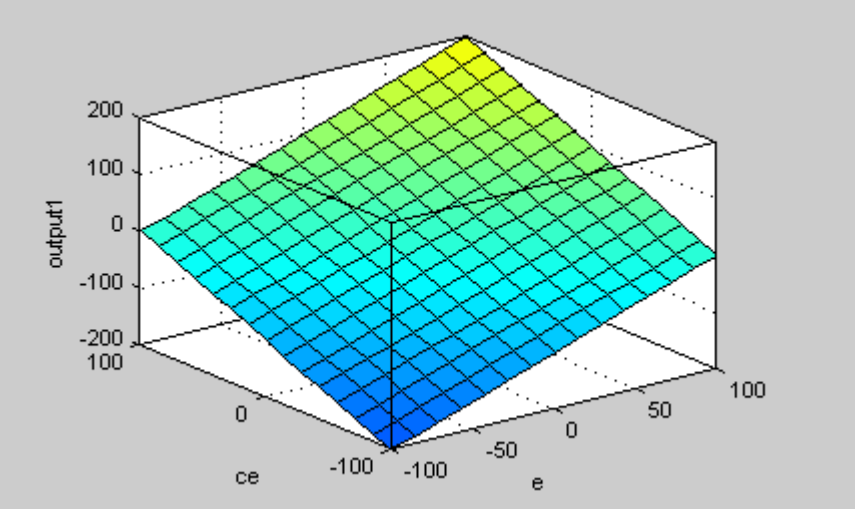

Figure 8: control surface

By implementing a constant singleton output using Sugeno method and for the same triangular membership function as in the previous simulation A, we have got a linear control surface representing the error, change of error and the output.

C- Using Gaussian Membership function:

in order to represent a non linear control surface we have implemented Gaussian membership functions where sigma is defined at $\mu = 0.5$ which is $\sigma = 42.73$

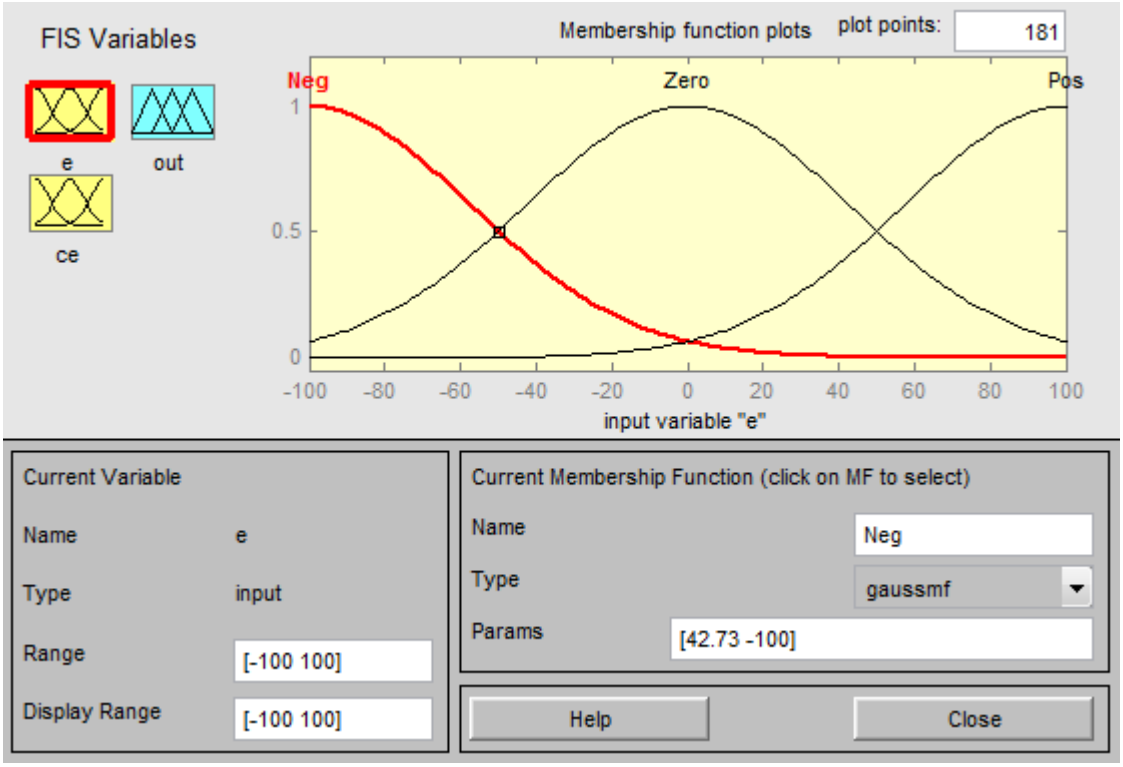

Figure 9: Gaussian membership functions

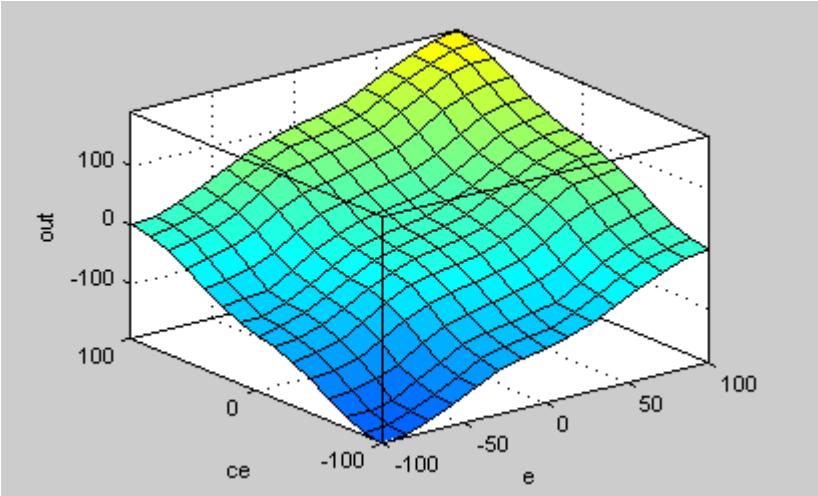

Figure 10: Bumpy control surface

As we can see using Gaussian membership functions we have got a bumpy surface. Built by using the all set of rules (9 rules) with non linear input, it has a flat plateau near the center and bumps in several other places. Even this surface has the same values as the other surfaces in the four corners.

Simulink simulation:

After building the FIS our block diagram is built in Simulink as shown in figure 11. We have watched the output of the controller (control signal) and the overall output of the system (process output).

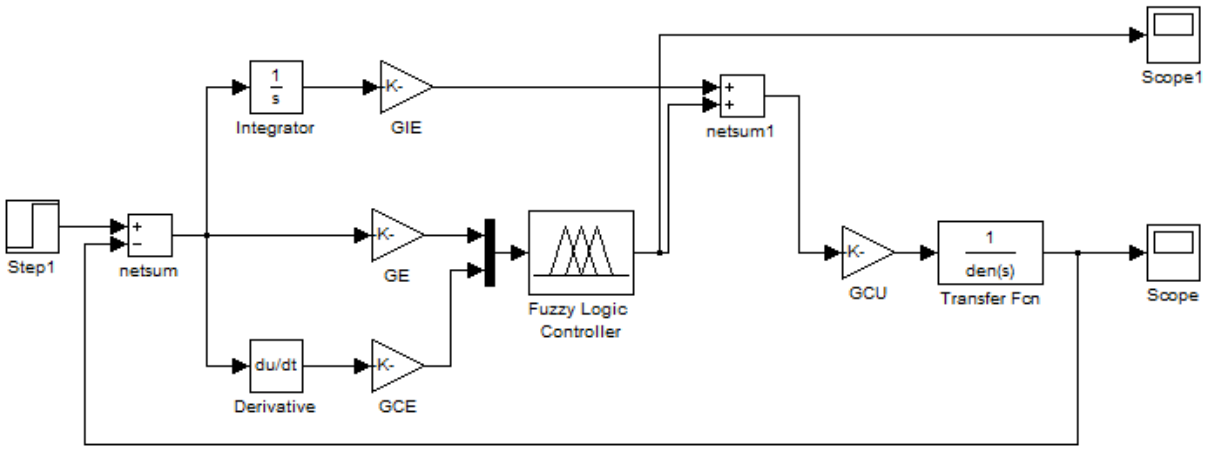

Figure 11: Block diagram on Simulink

The block diagram is formed by using a step input, gains to set parameters as GE, GCE, GIE & GCU, derivative and integrator. The fuzzy controller where our fuzzy logic membership functions and rules are set. At the end we have the plant which is the transfer function containing the $1/(1+s)^3$ transfer function and two scopes one to sketch the control signal from the controller output and the other to sketch the overall process signal.

As an output of our simulation, to outputs are considered one for Mamdani and Sugeno using the triangular membership function which gave the same simulation results below.

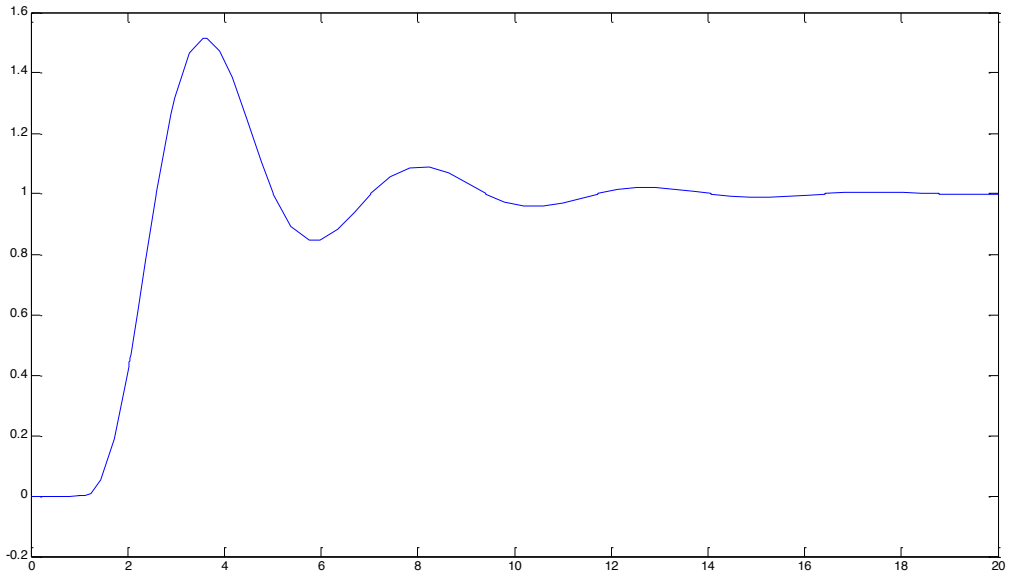

Figure 12: process signal using triangular M.F.

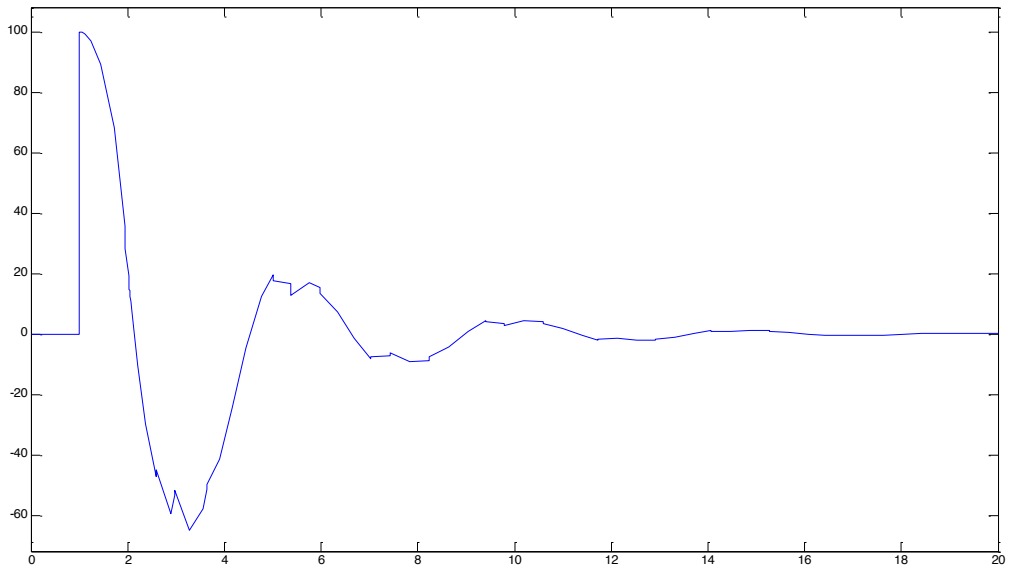

Figure 13: control signal using triangular M.F.

As we can see the process signal have an overshoot reaching around 1.5 with few oscillations reaches back to 1 steady state after 5 seconds.

As when using Gaussian membership function the simulation results for our system are presented below.

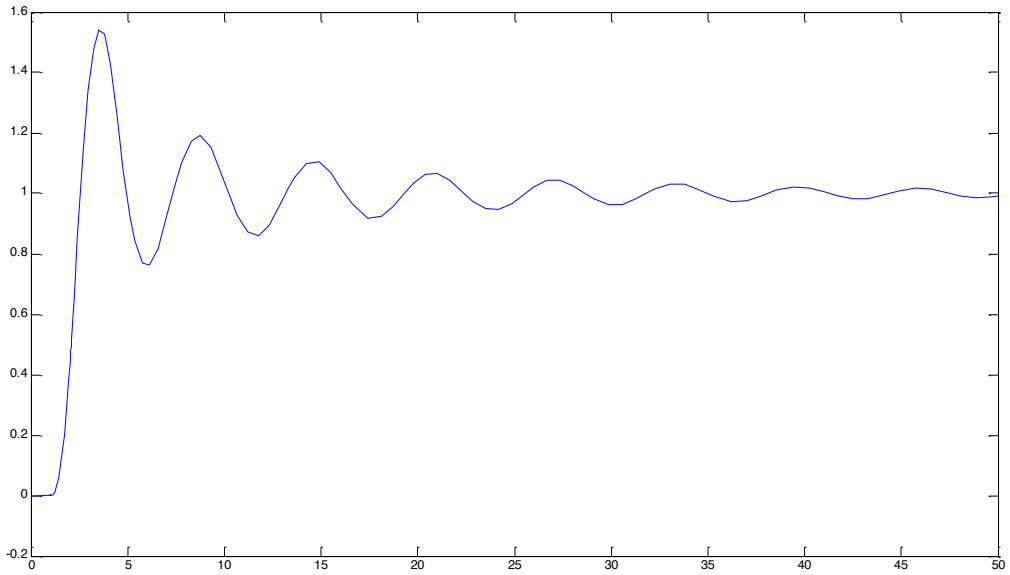

Figure 14: the process signal using Gaussian M.F.

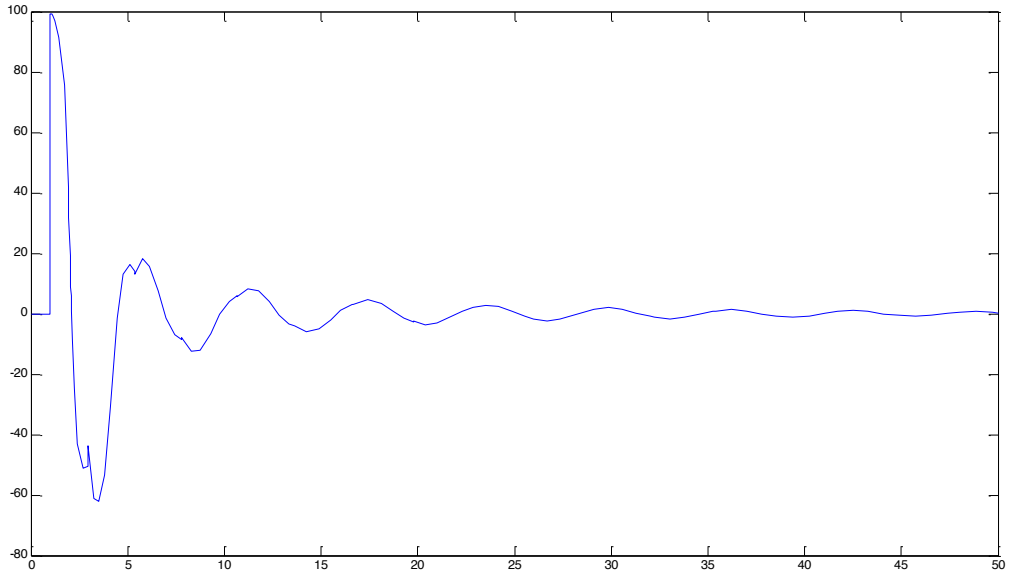

Figure 15: the control signal using Gaussian M.F.

As we can see since membership function are non linear for the same overshoot but rising time took more time as well settling time was delayed.

Now we wish to do the same design but using the lookup table instead of the fuzzy controller and this by implementing the logic in Matlab code then running the simulation using the results of the look up table which should give the same result in a faster time.

The code describing the simulation using Gaussian membership functions is the following:

```
a = newfis('a','sugeno');


a = addvar(a,'input','E',[-100 100]);
a = addmf(a,'input',1,'Negative','gaussmf', [42.73 -100]);
a = addmf(a,'input',1,'zero','gaussmf', [-42.73 0]);
a = addmf(a,'input',1,'Positive','gaussmf', [42.73 100]);


a = addvar(a,'input','CE',[-100 100]);
a = addmf(a,'input',2,'Negative','gaussmf', [42.73 -100]);
a = addmf(a,'input',2,'zero','gaussmf', [-42.73 0]);
a = addmf(a,'input',2,'Positive','gaussmf', [42.73 100]);


a = addvar(a,'output','u',[-200 200]);
a = addmf(a,'output',1,'NB','constant',-200);
a = addmf(a,'output',1,'N','constant',-100);
a = addmf(a,'output',1,'Z','constant',0);
a = addmf(a,'output',1,'P','constant',100);
a = addmf(a,'output',1,'PB','constant',200);

ruleList = [1 1 1 1 1;
            1 2 2 1 1;
            1 3 3 1 1;
            2 1 2 1 1;
            2 2 3 1 1;
            2 3 4 1 1;
            3 1 3 1 1;
            3 2 4 1 1;
            3 3 5 1 1];
a = addrule(a,ruleList);
figure
plotmf(a,'input',1);
figure
plotmf(a,'input',2);
figure
```

```
gensurf(a)

Step = 50;
E = -100:Step:100;
CE = -100:Step:100;
N = length(E);
LookUpTableData = zeros(N);
for i=1:N
   for j=1:N
      % compute output u for each combination of break points
      LookUpTableData(i,j) = evalfis([E(i) CE(j)],a);
   end
end
```

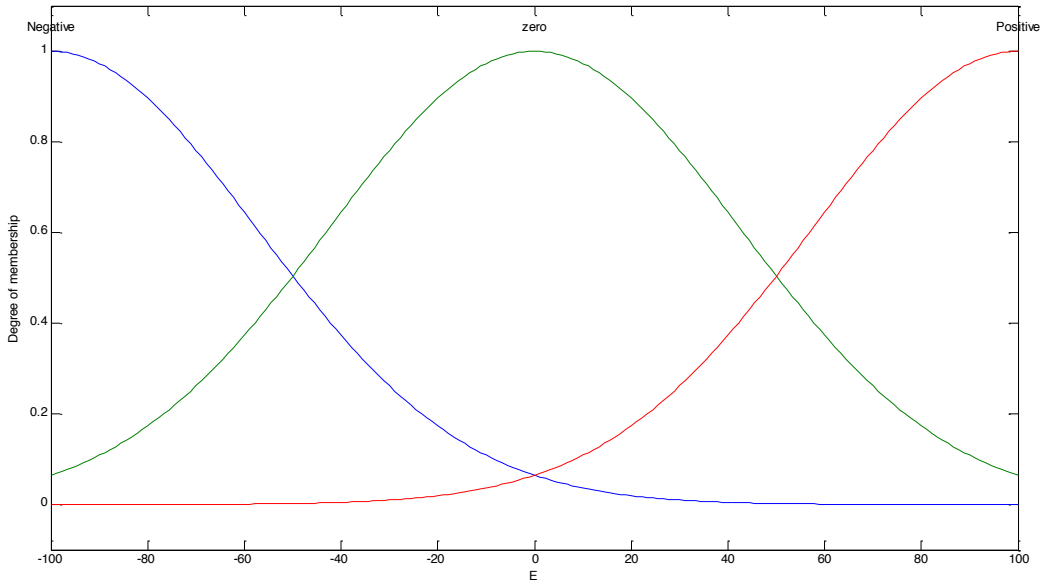

Figure 16: Gaussian membership function for both E (error) and CE (change of error)

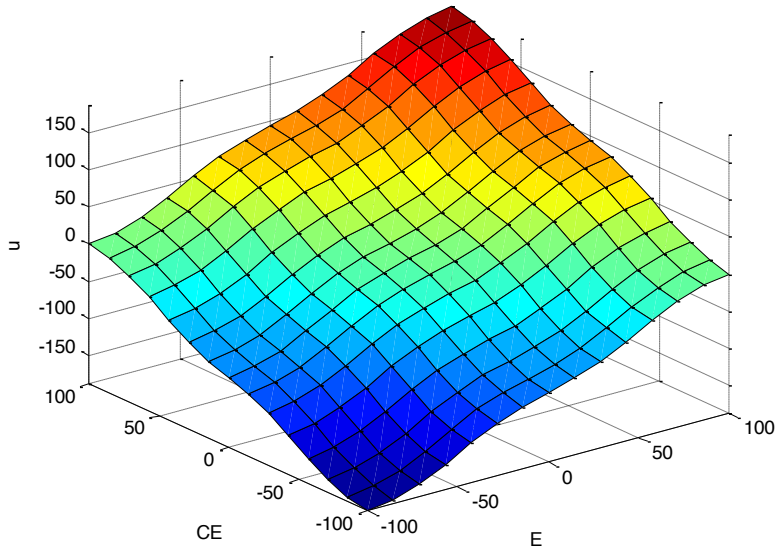

Figure 17: the control surface

As we can see the control surface is the same gotten when using fuzzy logic toolbox shown in figure 10 which is a bumpy surface having flat plateau near the centre and bumps in several other places. Even this surface has the same values as the other surfaces in the four corners.

The new block diagram using the look-up table is the following:

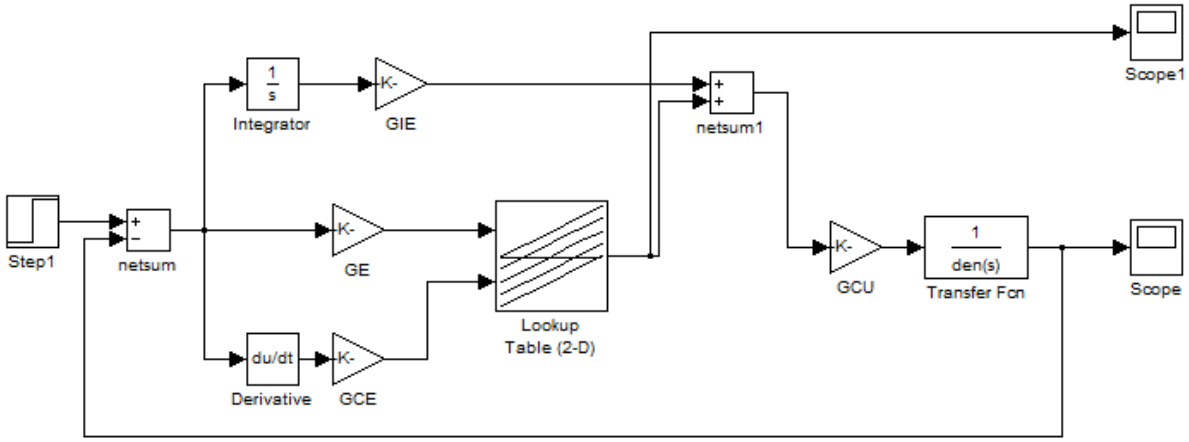

Figure 18: block diagram using look-up table

The look-up table advantage is to give a fast result since it doesn't have to go over all the values in the controller giving the same result in less simulation time.

Conclusion:

The typical FIS inputs are the signals of error (e(k)) and change of error (e(k)-e(k-1)). The FIS output is the control action inferred from the fuzzy rules. Fuzzy Logic Toolbox™ provides commands and GUI tools to design a FIS for a desired control surface.

Nonlinear control surfaces can often be approximated by lookup tables to simplify the generated code and improve execution speed. For example, a Fuzzy Logic Controller block in Simulink can be replaced by a set of Lookup Table blocks, one lookup table for each output defined in the FIS. Fuzzy Logic Toolbox provides command such as evalfis to compute data used in those lookup tables.

We also demonstrate how to implement the fuzzy inference system with a 2-D lookup table that properly approximates the control surface and achieves the same control performance.

After verifying that the linear fuzzy PID controller is properly designed, adjust FIS settings such as its style, membership functions and rule base to obtain a desired nonlinear control surface.

Compared with the traditional linear PID controller (the response curve with large overshoot), the nonlinear fuzzy PID controller reduces the over shoot by 50%, which is what we expected. The two response curves from the nonlinear fuzzy controllers almost overlap each other (using fuzzy toolbox Gaussian entry & using Matlab code to prepare for the look-up table method), which indicates that the 2-D lookup table approximates the fuzzy inference system very well.