Report on Fuzzy Gain Scheduling of PID Controllers

ENPM 667 Project 1 Section (201)

1st Prathinav Karnala Venkata
Robotics Program
University of Maryland
College Park, Maryland, USA
pratkv@umd.edu

2nd Sarang Shibu
Robotics Program
University of Maryland
College Park, Maryland, USA
sarang@umd.edu

***Abstract*—This is a report based on the research paper 'Fuzzy Gain Scheduling of PID Controllers'. Here we are using fuzzy logic to determine the control parameters of the PID controller. This fuzzy logic controller takes the error signal and its first derivative as the inputs to determine the parameters. We also perform simulations of the fuzzy tuned PID and compare its performance with other PID tuning methods like Ziegler - Nichol's method.**

I. Introduction

The Proportional Integral Derivative Controller, also known as the PID controller, is a popular controller with a wide range of applications in control systems engineering. This controller is designed using three parameters which are the proportional gain, integral gain, and derivative gain.

Here we are using the fuzzy logic controller in MATLAB Simulink to simulate the fuzzy logic PID controller and determine the parameters of the PID controller using the error signal and its first difference. We will compare the results with Ziegler-Nichol's tuning method [19].

There is a drawback: The Kitamori tuning method which was used in the research is difficult to implement as the relevant research papers [7][26] based on it are all written in Japanese and it is hard to comprehend the approach taken and its implementation. Hence the implementation of Kitamori PID tuning has been excluded from the report.

Fuzzy control involves using linguistic descriptions of human expertise to create fuzzy rules that guide the control process [12]- [14]. These rules are used with information about the state of the process to make control decisions. Fuzzy logic controllers can be seen as nonlinear PID controllers whose parameters are adjusted online based on error signals and their derivatives. This paper presents a rule-based method for adjusting PID controller gains in process control. This method uses fuzzy rules and reasoning to determine controller parameters, while the PID controller generates the control signal. The main concept is that human expertise in PID gain scheduling can be captured through fuzzy rules, resulting in better control performance compared to PID controllers with fixed parameters.

II. PID Controller

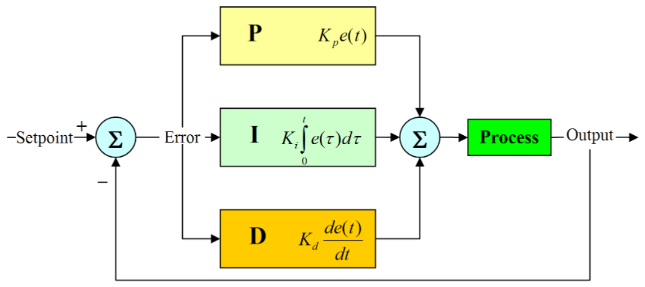


Fig. 1. PID controller block diagram.

The Transfer Function of the PID controller is given by:

$$G(s) = K_P + \frac{K_I}{s} + K_D s \tag{1}$$

A. *Proof:*

The PID controller is a combination of the proportional controller, the integral controller, and the derivative controller as shown in Fig. 1, and produces a control signal *u(t)*, which is proportional to the error signal, the integral of the error signal, and the derivative of the error signal.

$$u(t) \propto e(t) \tag{2}$$

$$u(t) \propto \int e(t)dt \tag{3}$$

$$u(t) \propto \frac{d}{dt}(e(t)) \tag{4}$$

Removing the Proportionality, we can assign the proportional constants K_P for Proportional gain, K_I for Integral gain, and K_D for Derivative gain.

$$u(t) = K_p e(t) \tag{5}$$

$$u(t) = K_I \int e(t)dt \tag{6}$$

$$u(t) = K_D \frac{d}{dt}(e(t)) \tag{7}$$

Combining the above equations, we can determine the PID controller's control signals,

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{d}{dt}(e(t)) \tag{8}$$

Taking the Laplace Transform of the following equation yields,

$$U(s) = K_p E(s) + K_l \frac{E(s)}{s} + K_D s E(s) \tag{9}$$

Taking *E(s)* common,

$$U(s) = \left(K_P + \frac{K_I}{s} + K_D s\right) E(s) \tag{10}$$

Dividing by *E(s)* on both sides gives Transfer function *G(s)*,

$$G(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s \tag{11}$$

Here we can take K_P common,

$$G(s) = K_P \left(1 + \frac{K_I}{K_P s} + \frac{K_D}{K_P} s\right) \tag{12}$$

Here we take taking Integral Time Constant as T_I, and Derivative Time Constant as T_D,

$$T_I = \frac{K_P}{K_I} \tag{13}$$

$$T_D = \frac{K_D}{K_P} \tag{14}$$

Therefore, we can write the Transfer Function *G(s)* as,

$$G(s) = K_P \left(1 + \frac{1}{T_I s} + T_D s\right) \tag{15}$$

The discrete time equivalent of the control signal *u(t)* is:

$$u(k) = K_P e(k) + K_I T_s \sum_{i=1}^{n} e(i) + \frac{K_D}{T_s} \Delta e(k) \tag{16}$$

Where Ts is the sampling period for the controller, and the error difference is:

$$\Delta e(k) := e(k) - e(k-1) \tag{17}$$

III. Fuzzy Logic and Gain Scheduling

Here we are first considering that the values of K_P and K_D (Proportional and Derivative gains) lie in a certain range,

$$K_p \in [K_{\text{Pmin}}, K_{\text{Pmax}}] \tag{18}$$

$$K_D \in [K_{\text{Dmin}}, K_{\text{Dmax}}] \tag{19}$$

The values of these ranges are found by heuristic methods, where we experimentally determine the gain Ku, and time-period Tu at which the system produces sustained oscillations according to Zeigler Nichols tuning method [19]. Here,

$$K_p = 0.6 K_u \tag{20}$$

$$T_I = 0.5 T_u \tag{21}$$

$$T_D = 0.125 T_u \tag{22}$$

From dividing (21) and (22), we see that,

$$T_I = 4 T_D \tag{23}$$

Here we have a relation between T_I and T_D as:

$$T_I = \alpha T_D \tag{24}$$

Where α can be obtained by dividing (13) with (14) yielding:

$$T_I = \frac{K_P^2}{K_I K_D} T_D \tag{25}$$

From here we can obtain,

$$\alpha = \frac{K_P^2}{K_I K_D} \tag{26}$$

For convenience of implementation of the Fuzzy Logic Controller we are normalizing the values of K_P and K_D to values between [0,1] using the linear transformations:

$$K'_P = \frac{K_P - K_{P_{\text{min}}}}{K_{P_{\text{max}}} - K_{P_{\text{min}}}} \tag{27}$$

$$K'_D = \frac{K_D - K_{D_{\text{min}}}}{K_{D_{\text{max}}} - K_{D_{\text{min}}}} \tag{28}$$

Where,

$$K_{P_{\text{min}}} = 0.32 K_u \tag{29}$$

$$K_{P_{\text{max}}} = 0.6 K_u \tag{30}$$

$$K_{D_{\text{min}}} = 0.08 K_u T_u \tag{31}$$

$$K_{D_{\text{max}}} = 0.15 K_u T_u \tag{32}$$

Now we start to design the Fuzzy Logic controller, Fig. 1 shows how the system's model would look like: The fuzzy

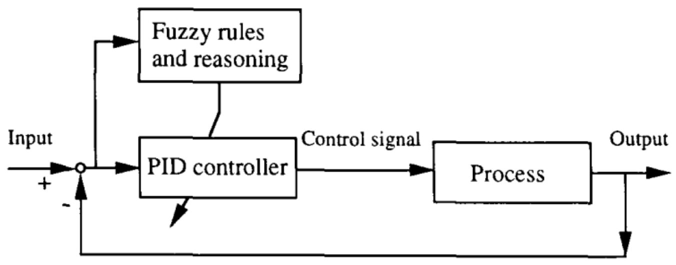


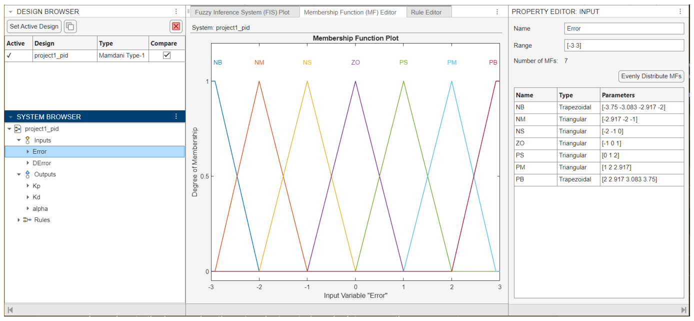
Fig. 2. PID control system with a fuzzy gain scheduler.

rules that the Fuzzy Logic Controller is implementing take the inputs $e(k)$ and its first difference $\Delta e(k)$, and yield the outputs K'_P, K'_D, and α'.

The rules have the following format [26]:

If $e(k)$ is A_i, and $\Delta e(k)$ is B_i, then K'_P is C_i, K'_D is D_i, and α is α_i, where $i = 1, 2, 3, \ldots, m$.

Where A_i and B_i are fuzzy variables that correspond to the fuzzy sets: {NB, NM, NS, ZO, PS, PM, PB}, observe Fig. 3, and 4. Where C_i and D_i are fuzzy variables that correspond to the fuzzy sets: {Small, Big}, observe Fig. 5, and 6. Where α_i is a fuzzy variable with a singleton membership function and corresponds to the fuzzy set: {2, 3, 4, 5}, observe Fig. 7



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

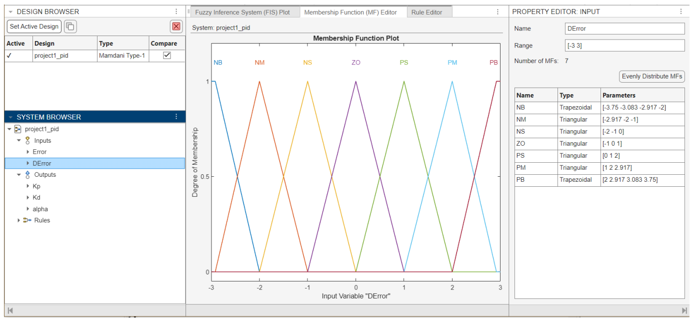
PROPERTY EDITOR: INPUT

Name: Error
Range: [-3 3]
Number of MFs: 7

Evenly Distribute MFs

Name	Type	Parameters
NB	Trapezoidal	[-3.75 -3.083 -2.917 -2]
NM	Triangular	[-2.917 -2 -1]
NS	Triangular	[-2 -1 0]
ZO	Triangular	[-1 0 1]
PS	Triangular	[0 1 2]
PM	Triangular	[1 2 2.917]
PB	Trapezoidal	[2 2.917 3.083 3.75]

Fig. 3. Membership function of Error.



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

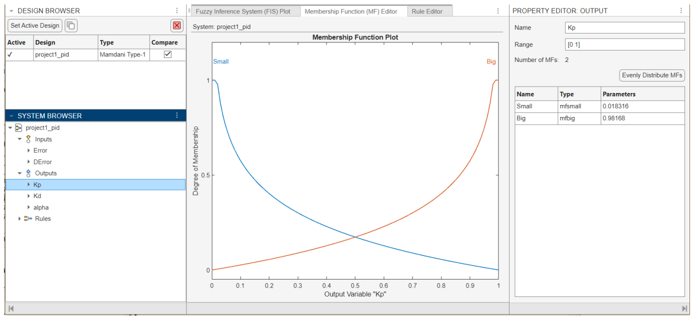
PROPERTY EDITOR: INPUT

Name: DError
Range: [-3 3]
Number of MFs: 7

Evenly Distribute MFs

Name	Type	Parameters
NB	Trapezoidal	[-3.75 -3.083 -2.917 -2]
NM	Triangular	[-2.917 -2 -1]
NS	Triangular	[-2 -1 0]
ZO	Triangular	[-1 0 1]
PS	Triangular	[0 1 2]
PM	Triangular	[1 2 2.917]
PB	Trapezoidal	[2 2.917 3.083 3.75]

Fig. 4. Membership function of Error difference.



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

PROPERTY EDITOR: OUTPUT

Name: Kp
Range: [0 1]
Number of MFs: 2

Evenly Distribute MFs

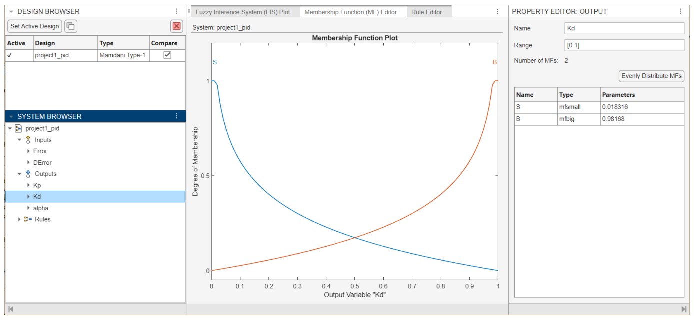
Name	Type	Parameters
Small	mfsmall	0.018316
Big	mfbig	0.98168

Fig. 5. Membership function of K_P'.

N is Negative, P is Positive, ZO - approximately zero, S is small, M is medium, and B is Big. Therefore, NB = Negative Big, PM = Positive Medium, and so on, are called the fuzzy sets. These fuzzy sets have membership functions. The fuzzy sets are partitions of the total range of values for a particular universe, values falling within those partitions will have a certain membership value associated with it and are determined by their membership functions.

According to the fuzzy rules, the antecedent part of a rule determines the truth value of that specific rule. i.e., from the rule:

If $e(k)$ is A_i, and $\Delta e(k)$ is B_i, then Kp' is C_i, Kd' is D_i,



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

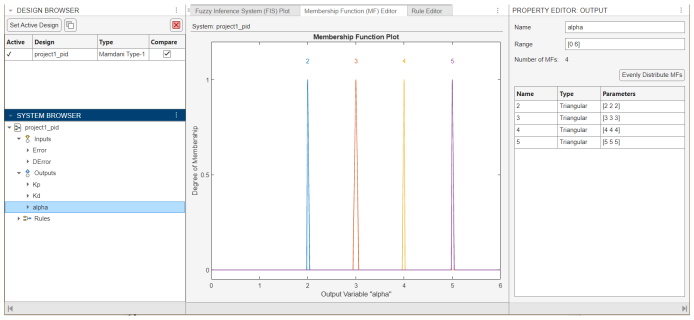
PROPERTY EDITOR: OUTPUT

Name: Kd
Range: [0 1]
Number of MFs: 2

Evenly Distribute MFs

Name	Type	Parameters
S	mfsmall	0.018316
B	mfbig	0.98168

Fig. 6. Membership function of K_D'.



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

PROPERTY EDITOR: OUTPUT

Name: alpha
Range: [0 6]
Number of MFs: 4

Evenly Distribute MFs

Name	Type	Parameters
2	Triangular	[2 2 2]
3	Triangular	[3 3 3]
4	Triangular	[4 4 4]
5	Triangular	[5 5 5]

Fig. 7. Membership function of α.

and $\alpha = \alpha_i$.

The truth value is calculated by taking the product of the input membership functions of A_i and B_i, which is the antecedent part of the rule i.

$$\mu_i = \mu_{A_i}(e(k)) \cdot \mu_{B_i}(\Delta e(k)) \tag{33}$$

Where μ_{A_i} is the membership function of fuzzy set A_i and μ_{B_i} is the membership function of fuzzy set B_i given a particular value $e(k)$ and $\Delta e(k)$ for a particular rule i.

According to fuzzy logic theory [18], the sum of the truth values of all the rules is equal to 1.

$$\sum_{i=1}^{m} \mu_i = 1 \tag{34}$$

A. *Fuzzy Example*

Let us consider the example of a washing machine, which uses fuzzy logic to compute the wash time (output) by taking the inputs of Dirt and Grease of the cloth to be washed. Let the Membership functions used for Dirt and Grease be defined using triangular functions. A triangular function is defined as follows:

$$f(x; a, b, c) = \begin{cases} 0, & \text{if } x \leq a \\ \frac{x-a}{b-a}, & \text{if } a \leq x \leq b \\ \frac{c-x}{c-b}, & \text{if } b \leq x \leq c \\ 0, & \text{if } x \geq c \end{cases}$$

TABLE I
FUZZY RULES FOR WASHING TIME T

		Grease		
		Low	Medium	High
Dirt	Low	VS	M	L
	Medium	S	M	L
	High	M	L	VL

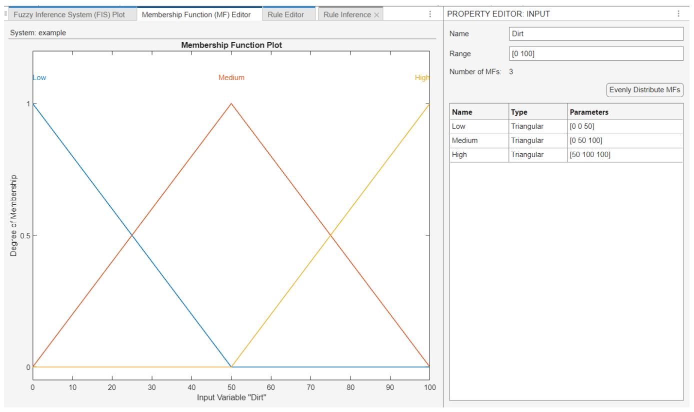
Or for simplicity we can write it as:

$$f(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right)$$

The process of a fuzzy can be boiled down to the following steps:

1 The fuzzy system takes a crisp input(s), and performs fuzzification to yield the membership function values of the fuzzy sets.
2 Then we obtain the truth value for each rule.
3 Then implication is performed to yield output fuzzy sets for each rule.
4 Then aggregation is performed to yield a single output fuzzy set from all the output fuzzy sets obtained from the rules.
5 then the defuzzification is done to the final output fuzzy set to yield a crisp output.

In the washing machine fuzzy system, as shown in Fig. 12, we use the rules as shown in Table I. We can see the same rules set into the fuzzy system in Fig. 11.



PROPERTY EDITOR: INPUT

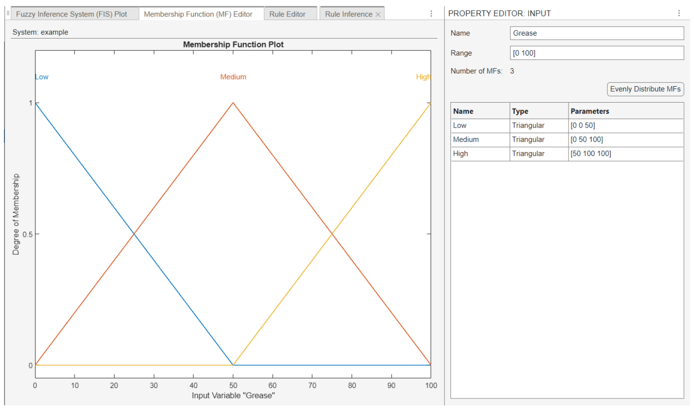
Name: Dirt
Range: [0 100]
Number of MFs: 3

Name	Type	Parameters
Low	Triangular	[0 0 50]
Medium	Triangular	[0 50 100]
High	Triangular	[50 100 100]

Fig. 8. Membership function of Dirt.

Low, Medium, and High are fuzzy sets of Dirt and Grease. VS - Very Short, S - Short, M - Medium, L - Long, and VL - Very Long, are fuzzy sets of Washing Time. The membership functions of Dirt and Grease are shown in Fig. 8, and Fig. 9 respectively. The range of the input values that they take are in percentages ranging from [0,100]. The fuzzy sets of the output range from [0,60] minutes. The membership functions of the output fuzzy sets are shown in Fig. 10.

Let us consider the crisp inputs Dirt = 60%, and Grease = 70%. The Membership function values of Dirt that uses the triangular membership function:

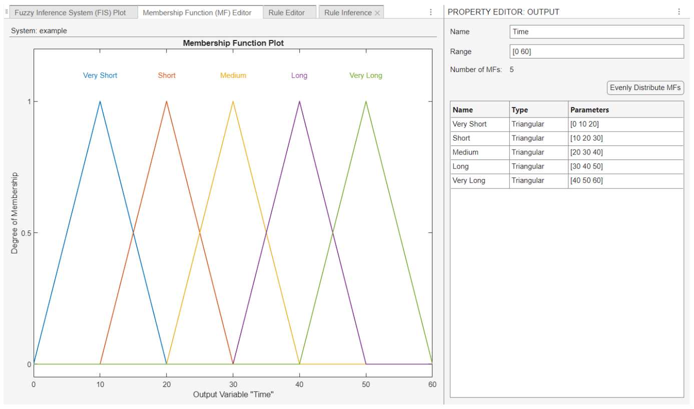


PROPERTY EDITOR: INPUT

Name: Grease
Range: [0 100]
Number of MFs: 3

Name	Type	Parameters
Low	Triangular	[0 0 50]
Medium	Triangular	[0 50 100]
High	Triangular	[50 100 100]

Fig. 9. Membership function of Grease.

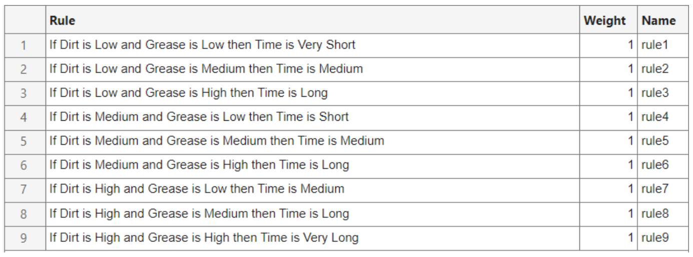


PROPERTY EDITOR: OUTPUT

Name: Time
Range: [0 60]
Number of MFs: 5

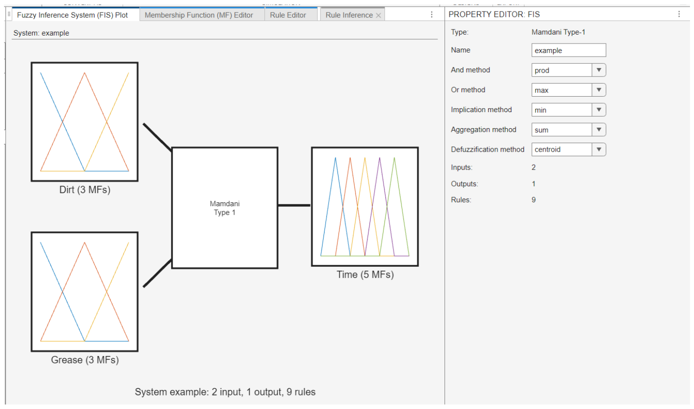
Name	Type	Parameters
Very Short	Triangular	[0 10 20]
Short	Triangular	[10 20 30]
Medium	Triangular	[20 30 40]
Long	Triangular	[30 40 50]
Very Long	Triangular	[40 50 60]

Fig. 10. Membership function of Washing Time T.



	Rule	Weight	Name
1	If Dirt is Low and Grease is Low then Time is Very Short	1	rule1
2	If Dirt is Low and Grease is Medium then Time is Medium	1	rule2
3	If Dirt is Low and Grease is High then Time is Long	1	rule3
4	If Dirt is Medium and Grease is Low then Time is Short	1	rule4
5	If Dirt is Medium and Grease is Medium then Time is Medium	1	rule5
6	If Dirt is Medium and Grease is High then Time is Long	1	rule6
7	If Dirt is High and Grease is Low then Time is Medium	1	rule7
8	If Dirt is High and Grease is Medium then Time is Long	1	rule8
9	If Dirt is High and Grease is High then Time is Very Long	1	rule9

Fig. 11. Rule set of the Fuzzy system.



System example: 2 input, 1 output, 9 rules

PROPERTY EDITOR: FIS

Type: Mamdani Type-1
Name: example
And method: prod
Or method: max
Implication method: min
Aggregation method: sum
Defuzzification method: centroid
Inputs: 2
Outputs: 1
Rules: 9

Fig. 12. The whole washing machine Fuzzy system.

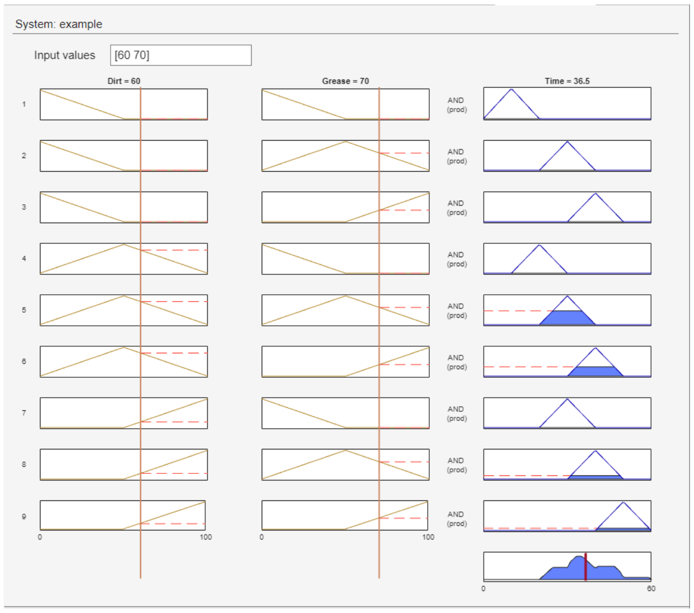


Fig. 13. The Fuzzy inference of the values Dirt = 60% and Grease = 70%

1 For low dirt membership function μ_{Ld} which lies in range [0,0,50], the
$$f(60;0,0,50)=0$$
2 For medium dirt membership function μ_{Md} which lies in range [0,50,100], the
$$f(60;0,50,100)=\frac{4}{5}$$
3 For high dirt membership function μ_{Hd} which lies in range [50,100,100], the
$$f(60;50,100,100)=\frac{1}{5}$$

Similarly, the Membership function values of Grease:

1 Low grease membership function μ_{Lg} which lies in range [0,0,50], the
$$f(70;0,0,50)=0$$
2 Medium grease membership function μ_{Mg} which lies in range [0,50,100], the
$$f(70;0,50,100)=\frac{3}{5}$$
3 High grease membership function μ_{Hg} which lies in range [50,100,100], the
$$f(70;50,100,100)=\frac{2}{5}$$

To obtain the truth value of all the rules which are of the form: If Dirt is A_i, and Grease is B_i, then Time is C_i, where $i=1,2,3,\ldots,m$..

Where A_i and B_i correspond to the fuzzy sets: Low, Medium, High, and C_i corresponds to the fuzzy sets:Very Short, Short, Medium, Long, Very Long.

The truth value μ_i is the truth value of the ith rule, which depends on the product of the membership functions of A_i and B_i, which is the antecedent part of the rule.

Therefore, the truth values of the 9 rules are:

$$\sum_{i=1}^{9}\mu_i=\mu_1+\mu_2+\mu_3+\mu_4+\mu_5+\mu_6+\mu_7+\mu_8+\mu_9$$

$$\begin{aligned}\mu_1&=0\\ \mu_2&=0\\ \mu_3&=0\\ \mu_4&=0\\ \mu_5&=\frac{4}{5}\cdot\frac{2}{5}=\frac{12}{25}\\ \mu_6&=\frac{4}{5}\cdot\frac{3}{5}=\frac{8}{25}\\ \mu_7&=0\\ \mu_8&=\frac{1}{5}\cdot\frac{2}{5}=\frac{3}{25}\\ \mu_9&=\frac{1}{5}\cdot\frac{3}{5}=\frac{2}{25}\end{aligned}$$

$$\sum_{i=1}^{9}\mu_i=\frac{12}{25}+\frac{8}{25}+\frac{3}{25}+\frac{2}{25}=1$$

This proves (34), that the sum of the truth values of the rules yields unity.

Now we perform the implication process of the fuzzy rule, which applies the $\min()$ function to the output fuzzy set of the corresponding fuzzy rule. In this case, it yields 9 different output fuzzy sets which are reshaped using the corresponding truth values.

We then use aggregation on these output fuzzy sets to yield a single output fuzzy set. The aggregation function uses the sum() function which adds the fuzzy sets together to yield a new output fuzzy set that contains all possible values of the crisp output value.

We now Defuzzify the output fuzzy set to yield a crisp output value using the Centroid function:

$$T=\frac{\int x\cdot\mu(x)\,dx}{\int\mu(x)\,dx}$$

the corresponding discrete function is:

$$T=\frac{\sum x\cdot\mu(x)}{\sum\mu(x)}$$

Since we found that the sum of the truth values for all the rules yields unity. The denominator is also unity so that the Time can be calculated as:

$$T=\int x\cdot\mu(x)\,dx$$

the corresponding discrete function is:

$$T=\sum x\cdot\mu(x)$$

By inputting the crisp inputs Dirt = 60% and Grease = 70% into the fuzzy inference system of MATLAB we can directly obtain the crisp output values of Time. In Fig. 13, we observe that the crisp value obtained in the Fuzzy Inference system in MATLAB is 36.5 minutes.

In Fig. 13, we see that the implication process that uses the min() function on the output fuzzy set of the rules scales it using the truth value, and we see 9 output fuzzy sets corresponding to each of the 9 rules. The final output fuzzy set is seen at the bottom, which is obtained using the sum() function, which adds up all the output fuzzy sets of the 9 rules to yield a single output fuzzy set. The defuzzification is done onto the final output fuzzy set, which applies the Centroid function, and then the final crisp output is obtained.

B. Returning to the PID Fuzzy Tuner

According to fuzzy logic theory, the membership functions of the fuzzy sets are used to first transform the input values, $e(k)$ and $\Delta e(k)$, into fuzzy values. We refer to this as fuzzification. Next, each rule's truth values are calculated. Once the truth values are computed, the implication process is done using the min() function on the corresponding output fuzzy sets of the rules, the basic idea of it is given in Fig. 14. Aggregation is then done which sums up the output fuzzy sets of all the rules, to yield a new output fuzzy set that contains the range of all the crisp output values. Then Defuzzification process is initiated in which the Centroid function is used to determine the final optimal crisp value of the output. In the end, we find that the values of K'_P, K'_D, and α_i are determined using the equations:

$$K'_P = \sum_{i=1}^{m} \mu_i K'_{P_i} \tag{35}$$

$$K'_D = \sum_{i=1}^{m} \mu_i K'_{D_i} \tag{36}$$

$$\alpha = \sum_{i=1}^{m} \mu_i \alpha_i \tag{37}$$

Where K'_{P_i}, K'_{D_i}, and α_i are the range of values of K'_P, K'_D, and α for the ith rule.

Once the values of K'_P, K'_D, and α are obtained, we can find the values of K_P, K_D, and K_I using the equations (26), (27), and (28):

$$K_P = K'_P(K_{P_{\max}} - K_{P_{\min}}) + K_{P_{\min}} \tag{38}$$

$$K_D = K'_D(K_{D_{\max}} - K_{D_{\min}}) + K_{D_{\min}} \tag{39}$$

$$K_I = \frac{K_P^2}{\alpha K_D} \tag{40}$$

Usually, the membership functions chosen while designing the Fuzzy Logic Controller use a predefined set of membership functions whose values range from [0,1]. The custom membership function of the fuzzy sets of K'_P and K'_D chosen by the author are:

$$\mu_{\text{small}}(x) = -\frac{1}{4}\ln(x) \quad \text{or} \quad x_{\text{small}}(\mu) = e^{-4\mu} \tag{41}$$

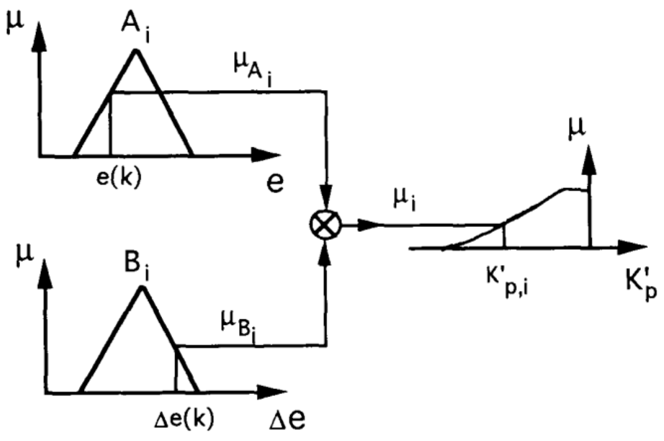


Fig. 14. To represent the idea of how defuzzification takes place.

For Fuzzy set Small or S.

$$\mu_{\text{big}}(x) = -\frac{1}{4}\ln(1-x) \quad \text{or} \quad x_{\text{big}}(\mu) = 1 - e^{-4\mu} \tag{42}$$

For Fuzzy set Big or B.

We have to cap the value of the log function so that its range stays within [0, 1]. So we find the x value at which the $\mu_{\text{small}} = 1$, which is found to be 1.831564e-02.

$$x_{\text{small}} = e^{-4} = 1.831564 \times 10^{-2} \tag{43}$$

We have to cap the value of the log function so that its range stays within [0, 1]. So we find the x value at which the $\mu_{\text{big}} = 1$, which is found to be 9.816844e-01.

$$x_{\text{big}} = 1 - e^{-4} = 9.816844 \times 10^{-1} \tag{44}$$

The code to implement these functions and plug them into the Fuzzy Logic Controller in MATLAB is shown below:

For the fuzzy set Small (S), whose membership function is given by (41).

```
function [y] = mfsmall(x,params)
% MF of Small for Kp' or Kd'
% values of x range from [0,1]
% values of y also range from [0,1]
% to find the value of MF small is Kp' and Kd'
    equal 1 is given by eq(20)
% the value of x when MF value = 1, is equal
    to exp(-4) = 0.018316
y(:) = 0;
for i = 1 : length(x)
    if x(i) <= params
        y(i) = 1;
    else
        y(i) = log(x(i))*(-1/4);
    end
end
end
```

Listing 1. MATLAB Code for Defining the membership function Small for K'_P and K'_D

For the fuzzy set Big (B), whose membership function is given by (42).

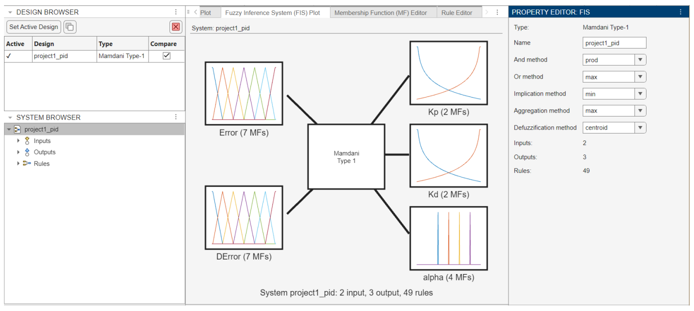
```
function [y] = mfbig(x,params)
% MF of Big for Kp’ or Kd’
% values of x range from [0,1]
% values of y also range from [0,1]
% to find the value of Kp’ and Kd’ for MF = 1
    is given by eq(21)
% the value of x when MF value = 1, is = 1 -
    exp(-4) = 0.98168

y(:) = 0;
for i = 1 : length(x)
    if x(i) >= params
        y(i) = 1;
    else
        y(i) = log(1-x(i))*(-1/4);
    end
end
end
```

Listing 2. MATLAB Code for Defining the membership function Big for K'_P and K'_D

IV. Fuzzy Inference System

The Fuzzy Inference System was created using the Fuzzy Logic Designer, an application of MATLAB. This application was used to design the complete fuzzy logic system for the PID controller. The structure of the Fuzzy inference system is shown in Fig. 15.



Active	Design	Type	Compare
✓	project1_pid	Mamdani Type-1	✓

System: project1_pid

System project1_pid: 2 input, 3 output, 49 rules

Type:	Mamdani Type-1
Name	project1_pid
And method	prod
Or method	max
Implication method	min
Aggregation method	max
Defuzzification method	centroid
Inputs:	2
Outputs:	3
Rules:	49

Fig. 15. Structure of the Fuzzy Inference System.

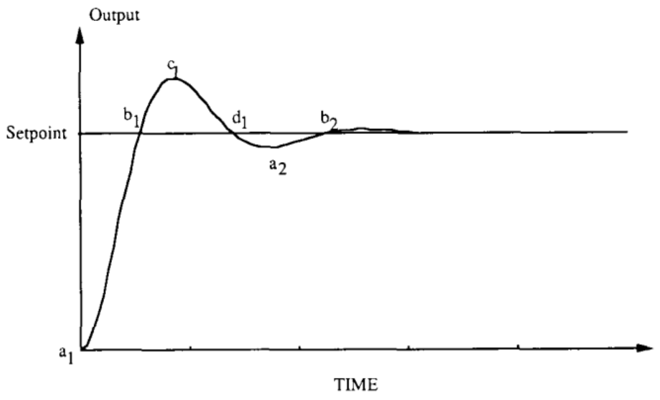


Fig. 16. Structure of the Fuzzy Inference System.

TABLE II
Rules for Kp'

		$\Delta(e(k))$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	ZO	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

TABLE III
Rules for Kd'

		$\Delta(e(k))$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZO	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

According to the paper, a desirable response by the user is shown in Fig. 16, in this response at point $a1$, to achieve a fast rise time, a strong control signal is needed. This implies that the Proportional gain and Integral gain must be large, and the Derivative gain must be small for a fast rise time. Therefore, Kp' can be represented by the fuzzy set Big, and Kd' by Small. Ki can be calculated according to equation (40), implying that when α decreases, the integral gain increases. Thus, taking a small value for α yields a large integral gain. Therefore, the rule for $a1$ would be as follows:

If $e(k)$ is PB, and $\Delta e(k)$ is ZO, then Kp' is Big, Kd' is Small, and $\alpha = 2$.

At point $b1$, the expert would know that to prevent an overshoot of the signal, a small control signal would be required. This implies the Proportional gain and Integral gain have to be small, and the Derivative gain must be large. So, the rule at $b1$ would be as follows:

If $e(k)$ is ZO, and $\Delta e(k)$ is NB, then Kp' is Small, Kd' is Big, and $\alpha = 5$.

The primary consideration for designing the rules was to leverage expert knowledge, enabling the fuzzy logic to adjust the parameters for generating the desired signal response.

In such a manner, the rules for the fuzzy controller were designed based on the experience and knowledge of how the signal should behave. Understanding the signal in this way led to the development of rules for each of the inputs and outputs of the Fuzzy Logic Controller.

Here, we have implemented two methods for tuning the PID controller: fuzzy tuning and Zeigler-Nichols method. Both tuning methods were applied to second-order, third-order, and fourth-order systems.

TABLE IV
RULES FOR α

		$\Delta(e(k))$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	2	2	2	2	2	2	2
	NM	3	3	2	2	2	3	3
	NS	4	3	3	2	3	3	4
	ZO	5	4	3	3	3	4	5
	PS	4	3	3	2	3	3	4
	PM	3	3	2	2	2	3	3
	PB	2	2	2	2	2	2	2

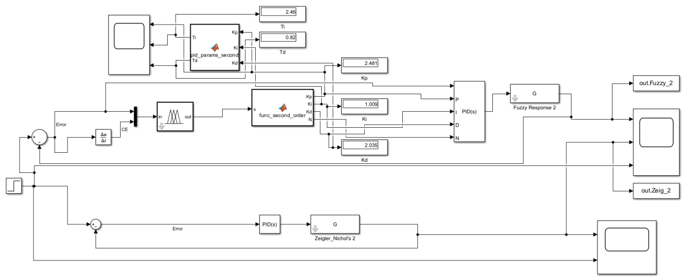


Fig. 17. Structure of the Second-order System.

A. *Creating the Second, Third, and Fourth, Order Systems on MATLAB*

The code presented below shows the definition of the second-order system(G), third-order system(P), and fourth-order system(Q).

```
s = tf('s');

% Second order system
G = exp(-0.5 * s) / ((s + 1)^2);

% Third order system
P = 4.228 / ((s + 0.5) * (s^2 + 1.64 * s +
    8.456));

% Fourth Order System
Q = 27 / ((s + 1) * (s + 3)^3);
```

Listing 3. MATLAB Code for Defining Transfer Functions

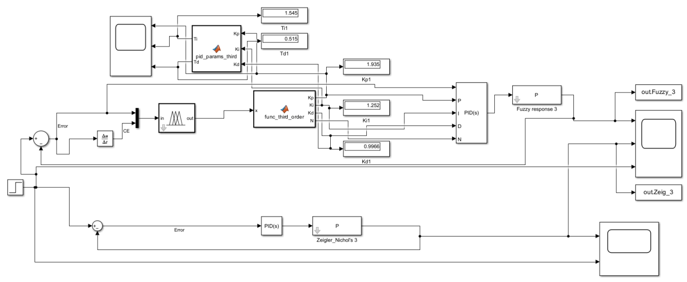


Fig. 18. Structure of the Third-order System.

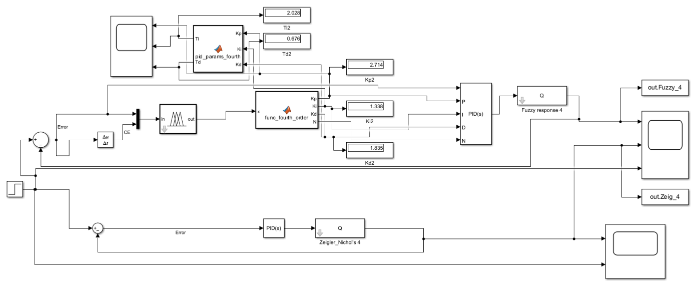


Fig. 19. Structure of the Fourth-order System.

B. *Second Order System*

For the second-order system with the transfer function:

$$G(s) = \frac{e^{-0.5s}}{(s+1)^2} \tag{45}$$

Structure for the second order system is shown in Fig. 17. According to Zeigler-Nichol's method [19], the tuning process involves the following steps:
1. Initially, set the parameters $K_P = 0$, $K_D = 0$, and $K_I = 0$ for the PID controller and check the step response.
2. Gradually increase the value of K_P until sustained oscillation is observed in the step response of the system.
3. The gain at which these sustained oscillations occur is the Ultimate gain (K_u).
4. The time period of oscillation (T_u) can also be obtained from the graph.

It's worth noting that using an analog or digital oscilloscope is often more suitable for accurately determining K_u and T_u than doing it in Simulink.

For the second-order system, the gain K_u and period of oscillation T_u are found to be:

$$K_u = 4.68$$

$$T_u = 3.28\,\text{s}$$

Hence, the tuning parameters for the PID controller can be obtained using the following equations and plugging in the values of K_u and T_u:

Proportional Gain (K_P) = 2.808
Derivative Gain (K_D) = 1.15128
Integral Gain (K_I) = 1.7122
Integral Time Constant (T_I) = 3.28 s
Derivative Time Constant (T_D) = 0.41 s

The function block `func_second_order` is used for converting the outputs of the fuzzy logic controller to obtain the parameters K_P, K_I, and K_D of the PID is shown below.

```
function [Kp, Ki, Kd, N] = func_second_order(x
    )
% Function to convert the Fuzzy output to
    obtain the PID parameters
% Using equations (38), (39), and (40). We
    plug the values of Kp', Kd', and alpha.
```

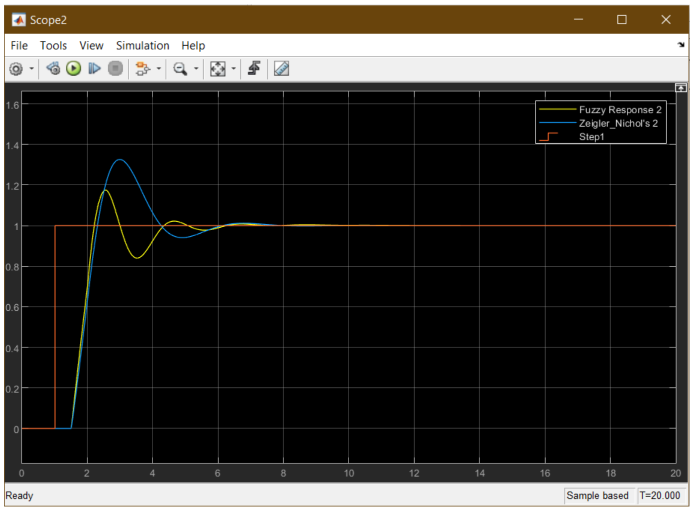


Fig. 20. Step response of the second-order System.

```
Ku = 4.68;
Tu = 3.28;
Kpmax = 0.6 * Ku;
Kpmin = 0.32 * Ku;
Kdmax = 0.15 * Ku * Tu;
Kdmin = 0.08 * Ku * Tu;

Kp = x(1) * (Kpmax - Kpmin) + Kpmin;
Kd = x(2) * (Kdmax - Kdmin) + Kdmin;

alpha = x(3);
Ki = (Kp^2) / (alpha * Kd);

% Here, N is the filter coefficient whose
    value has been set by default to 100 for
    all the simulations

N = 100;
```

Listing 4. MATLAB Code to Convert Fuzzy Output to PID Parameters

It takes the output of the fuzzy logic controller, which are K'_P, K'_D, and α, and using the equations (38), (39), and (40), we can then evaluate the values of the proportional gain K_P, integral gain K_I, and derivative gain K_D. The values of the Ultimate gain K_u and Time-Period of oscillation T_u are obtained using the Zeigler-Nichols method.

Function `pid_params_second` is used to obtain the variation of the parameters T_I, and T_D of the Second-order system using (13), and (14):

```
function [Ti, Td] = pid_params_second(Kp, Ki,
    Kd)
% Using equations eq (13) and (14)

Ti = Kp / Ki;
Td = Kd / Kp;
end
```

Listing 5. MATLAB Code for Calculating PID Parameters Ti and Td

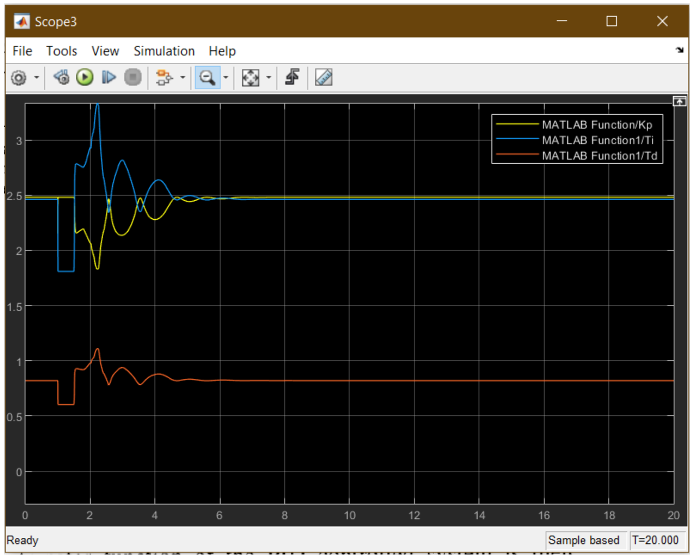


Fig. 21. The variation of parameters K_P, T_I, and T_D for the second order system

We then use the PID values we obtained from the function code block (`func_second_order`) and plug them into a PID controller and then connect it to the 2nd order system (G). The step response of Fuzzy logic and Zeigler-Nichols tuning methods for 2nd order system is shown in Fig.20. The variation of parameter values K_P, T_I, and T_D of the 2nd order fuzzy PID is shown in Fig.21.

C. Third Order System

For the third-order system with the transfer function:

$$G(s) = \frac{4.228}{(s+0.5)(s^2+1.64s+8.456)} \tag{46}$$

Structure for the third order system is shown in Fig. 18.
Applying the Zeigler-Nichols method,
We have found that the ultimate gain $K_u = 3.65$,
Time period of the sustained oscillation $T_u = 2.06s$
Proportional Gain $K_P = 2.19$,
Derivative gain $K_D = 0.56502$,
Integral gain $K_I = 2.1262$,
Integral Time constant $T_I = 1.03s$,
Derivative Time constant $T_D = 0.258s$.

The function block `func_third_order` is used for converting the outputs of the fuzzy logic controller to obtain the parameters K_P, K_I, and K_D of the PID is shown below.

```
function [Kp, Ki, Kd, N] = func_third_order(x)
% Function to convert the Fuzzy output to
    obtain the PID parameters
% Using equations eq (38), (39), and (40). We
    plug the values of Kp', Kd', and alpha

Ku = 3.65;
Tu = 2.06;
Kpmax = 0.6 * Ku;
```

```
Kpmin = 0.32 * Ku;
Kdmax = 0.15 * Ku * Tu;
Kdmin = 0.08 * Ku * Tu;

Kp = x(1) * (Kpmax - Kpmin) + Kpmin;
Kd = x(2) * (Kdmax - Kdmin) + Kdmin;

alpha = x(3);
Ki = Kp^2 / (alpha * Kd);

% Here N is the filter Coefficient whose value
    has been set by default to
% 100 for all the simulations

N = 100;
end
```

Listing 6. MATLAB Code for Calculating PID Parameters Kp, Ki, Kd, and N for Third Order System

It takes the output of the fuzzy logic controller, which are K'_P, K'_D, and α, and using the equations (38), (39), and (40), we can then evaluate the values of the proportional gain K_P, integral gain K_I, and derivative gain K_D. The values of the Ultimate gain K_u and Time-Period of oscillation T_u are obtained using the Zeigler-Nichols method.

Function `pid_params_third` is used to obtain the variation of the parameters T_I, and T_D of the Second-order system using eq.(13), and (14):

```
function [Ti, Td] = pid_params_third(Kp, Ki,
    Kd)
% Using equations eq (13) and (14)

Ti = Kp / Ki;
Td = Kd / Kp;
end
```

Listing 7. MATLAB Code for Calculating PID Parameters Ti and Td for Third Order System

We then use the PID values we obtained from the function code block (`func_third_order`) and plug them into a PID controller and then connect it to the 3rd order system (P). The step response of Fuzzy logic and Zeigler-Nichols tuning methods for 3rd order system is shown in Fig.22. The variation of parameter values K_P, T_I, and T_D of the 3rd order fuzzy PID is shown in Fig.23.

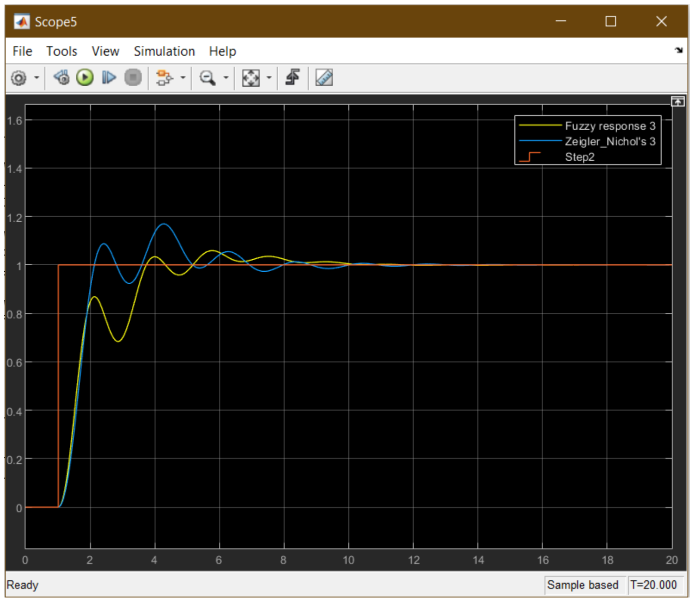


Fig. 22. Step response of the third-order System.

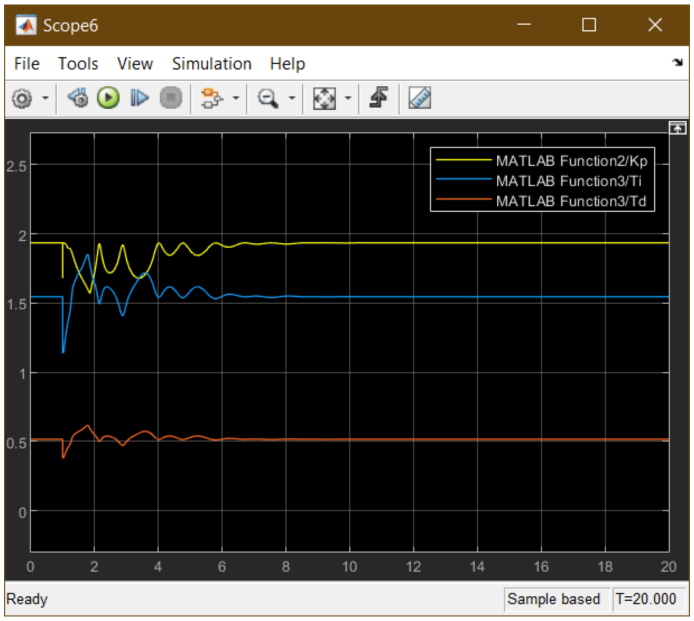


Fig. 23. The variation of parameters K_P, T_I, and T_D for the Third order system

D. Fourth Order System

For the fourth-order system with the transfer function:

$$G(s) = \frac{27}{(s+1)(s+3)^2} \tag{47}$$

Structure for the fourth order system is shown in Fig. 19.
Applying the Zeigler-Nichols method,
We have found that the ultimate gain $K_u = 5.12$,
Time period of the sustained oscillation $T_u = 2.704s$
Proportional Gain $K_P = 3.072$,
Derivative gain $K_D = 1.0383$,
Integral gain $K_I = 2.272189$,
Integral Time constant $T_I = 1.352s$,
Derivative Time constant $T_D = 0.338s$.

The function block `func_fourth_order` is used for converting the outputs of the fuzzy logic controller to obtain the parameters K_P, K_I, and K_D of the PID is shown below.

```
function [Kp, Ki, Kd, N] = func_third_order(x)
% Function to convert the Fuzzy output to
    obtain the PID parameters
% Using equations eq (38), (39), and (40). We
    plug the values of Kp', Kd', and alpha

Ku = 5.12;
```

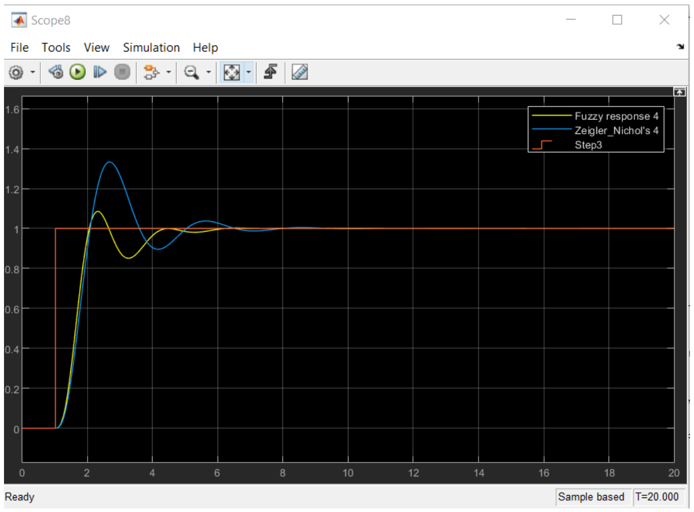


Fig. 24. Step response of the fourth-order System.

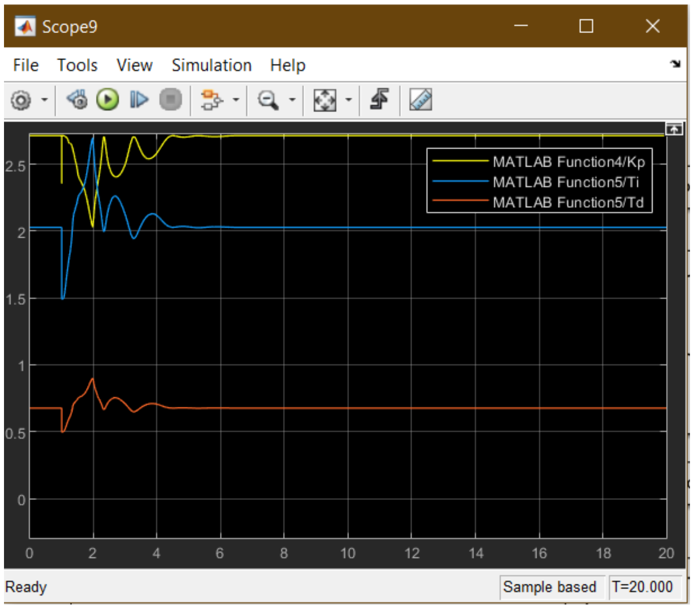


Fig. 25. The variation of parameters K_P, T_I, and T_D for the fourth order system

```
Tu = 2.704;
Kpmax = 0.6 * Ku;
Kpmin = 0.32 * Ku;
Kdmax = 0.15 * Ku * Tu;
Kdmin = 0.08 * Ku * Tu;

Kp = x(1) * (Kpmax - Kpmin) + Kpmin;
Kd = x(2) * (Kdmax - Kdmin) + Kdmin;

alpha = x(3);
Ki = Kp^2 / (alpha * Kd);

% Here N is the filter Coefficient whose value
    has been set by default to
% 100 for all the simulations

N = 100;
end
```

Listing 8. MATLAB Code for Calculating PID Parameters Kp, Ki, Kd, and N for Fourth Order System

It takes the output of the fuzzy logic controller, which are K'_P, K'_D, and α, and using the equations (38), (39), and (40), we can then evaluate the values of the proportional gain K_P, integral gain K_I, and derivative gain K_D. The values of the Ultimate gain K_u and Time-Period of oscillation T_u are obtained using the Zeigler-Nichols method.

Function `pid_params_third` is used to obtain the variation of the parameters T_I, and T_D of the Fourth-order system using (13), and (14):

```
function [Ti, Td] = pid_params_third(Kp, Ki,
    Kd)
% Using equations eq (13) and (14)

Ti = Kp / Ki;
Td = Kd / Kp;
end
```

Listing 9. MATLAB Code for Calculating PID Parameters Ti and Td for Fourth Order System

We then use the PID values we obtained from the function code block (`func_fourth_order`) and plug them into a PID controller and then connect it to the 4th order system (Q). The step response of Fuzzy logic and Zeigler-Nichols tuning methods for 4th order system is shown in Fig.24. The variation of parameter values K_P, T_I, and T_D of the 4th order fuzzy PID is shown in Fig.25.

V. Discussion of Stability

Using the fuzzy gain scheduling technique, the transient response of the system is quite fast, we can obtain a smaller overshoot and faster settling time as is evident from the graphs when compared to the Zeigler-Nichols tuning method. So, a change in the setpoint will be dealt with faster using the fuzzy method. But in the case of using the Zeigler-Nichols tuning method, the transient response may take longer to settle but this tuning technique is very good at load disturbance rejection and can reach stability quickly [19][4]. The author of the paper suggests that a hybrid controller where the fuzzy gain scheduling is used to get a fast transient response will be good at dealing with the setpoint changes. After the fuzzy transient response and gaining stability, it is better to use the Zeigler-Nichols tuning parameters, to ensure that the system is well-equipped to deal with load disturbances as it stabilizes quickly. This form of a hybrid controller will yield superior results than optimizing using just one of the tuning methods.

VI. Conclusion

It is quite evident that the fuzzy gain scheduling results in a faster transient response in most cases as compared to the Zeigler-Nichols tuning method. The data from the step response of each of the systems is given below.

A. Second Order System

```
>> stepinfo(out.Fuzzy_2.Data, out.Fuzzy_2.Time)
ans =

  struct with fields:

         RiseTime: 0.5541
    TransientTime: 5.7650
     SettlingTime: 5.7650
      SettlingMin: 0.8397
      SettlingMax: 1.1749
        Overshoot: 17.4911
       Undershoot: 0
             Peak: 1.1749
         PeakTime: 2.5607
```

```
>> stepinfo(out.Zeig_2.Data, out.Zeig_2.Time)
ans =

  struct with fields:

         RiseTime: 0.6226
    TransientTime: 5.7314
     SettlingTime: 5.7314
      SettlingMin: 0.9101
      SettlingMax: 1.3257
        Overshoot: 32.5699
       Undershoot: 0
             Peak: 1.3257
         PeakTime: 3.0042
```

The fuzzy logic tuning method yields a superior response as compared with the Zeigler-Nichols tuning method with lesser settling time, transient time, rise time, peak, and overshoot.

B. Third Order System

```
>> stepinfo(out.Fuzzy_3.Data,out.Fuzzy_3.Time)
ans =

  struct with fields:

         RiseTime: 2.2246
    TransientTime: 8.0915
     SettlingTime: 8.0915
      SettlingMin: 0.9074
      SettlingMax: 1.0590
        Overshoot: 5.8957
       Undershoot: 0
             Peak: 1.0590
         PeakTime: 5.7669
```

```
>> stepinfo(out.Zeig_3.Data,out.Zeig_3.Time)
ans =

  struct with fields:

         RiseTime: 0.7175
    TransientTime: 7.6319
     SettlingTime: 7.6319
      SettlingMin: 0.9157
      SettlingMax: 1.1696
        Overshoot: 16.9708
       Undershoot: 0
             Peak: 1.1696
         PeakTime: 4.2840
```

Here it is observed that the Rise time, transient time, settling time, and peak time are a tad higher for the fuzzy tuning method as compared to the Zeigler-Nichols tuning method. But the results are still good.

C. Fourth Order System

```
>> stepinfo(out.Fuzzy_4.Data,out.Fuzzy_4.Time)
ans =

  struct with fields:

         RiseTime: 0.6242
    TransientTime: 4.1124
     SettlingTime: 4.1124
      SettlingMin: 0.8508
      SettlingMax: 1.0854
        Overshoot: 8.5360
       Undershoot: 7.4074e-31
             Peak: 1.0854
         PeakTime: 2.3215
```

```
>> stepinfo(out.Zeig_4.Data,out.Zeig_4.Time)
ans =

  struct with fields:

         RiseTime: 0.6554
    TransientTime: 6.1335
     SettlingTime: 6.1335
      SettlingMin: 0.8955
      SettlingMax: 1.3329
        Overshoot: 33.2852
       Undershoot: 1.6412e-32
             Peak: 1.3329
         PeakTime: 2.6809
```

The fuzzy tuning method has a superior response as compared to the Zeigler-Nichols tuning method.

References

[1] K. L. Anderson, G. L. Blankenship, and L. G. Lebow, ”A rule-based adaptive PID controller,” in Proc. 27th IEEE Conf. Decision, Control, 1988, pp. 564-569.

[2] P. J. Gawthrop and P. E. Nomikos, ”Automatic tuning of commercial PID controllers for single-loop and multiloop applications,” IEEE Control Syst. Mag., vol. 10, pp. 34-42, 1990.

[3] J. Gertler and H-S. Chang, ”An instability indicator for expert control,” IEEE Control Syst. Mag., vol. 6, pp. 14-17, 1986.

[4] C. C. Hang, ”The choice of controller zeros,” IEEE Control Syst. Mag., vol. 9, pp. 72-75, 1989.

[5] C. C. Hang, K. J. Astrom, and W. K. Ho, ”Refinements of the Ziegler-Nichols tuning formula,” Proc. IEE, Pt. D., vol. 138, pp.

[6] T. Iwasaki and A. Morita, ”Fuzzy auto-tuning for PID controller with model classification,” in Proc. NAFIPS ’90, Toronto, Canada, June 6-8, 1990, pp. 90-93.

[7] T. Kitamori, ”A method of control system design based upon partial knowledge about controlled processes,” Trans. SICE Japan, vol. 15, pp. 549-555, 1979 (in Japanese).

[8] B. C. Kuo, Automatic Control Systems, 5th ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[9] C. C. Lee, ”Fuzzy logic in control systems: Fuzzy logic controller, Part I,” IEEE Trans. Syst., Man, Cybern., vol. SMC-20, pp. 404-418, 1990.

[10] C. C. Lee, ”Fuzzy logic in control systems: Fuzzy logic controller, Part II,” IEEE Trans. Syst., Man, Cybern., vol. SMC-20, pp. 419-435, 1990.

[11] C. G. Nesler, ”Experiences in applying adaptive control to thermal processes in buildings,” in Proc. Amer. Control Conf., Boston, MA, 1985, pp. 1535-1540.

[12] T. J. Procyk and E. H. Mamdani, ”A linguistic self-organizing process controller,” Automatica, vol. 15, pp. 15-30, 1979.

[13] M. Sugeno, ed., Industrial Applications of Fuzzy Control. Amsterdam, The Netherlands: North-Holland, 1985.

[14] T. Takagi and M. Sugeno, ”Fuzzy identification of systems and its applications to modeling and control,” in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 1, pp. 116-132, Jan.-Feb. 1985, doi: 10.1109/TSMC.1985.6313399.

[15] Y. Takahashi, M. J. Rabins, and D. M. Auslander. Control and Dynamic Systems.

[16] J. G. Truxal, Automatic Feedback Control System Synthesis. New York: McGraw-Hill, 1955.

[17] T. Yamamoto, S. Omatu, and H. Ishihara, “A construction of self-tuning PID control system,” Trans. SICE Japan, vol. 25, pp. 39-45. 1989 (in Japanese).

[18] Z. Y. Zhao, M. Tomizuka, and S. Sagara, “A fuzzy tuner for fuzzy logic controllers,” in Proc. 1992 Amer. Control Conf., Chicago, IL, June 24-26, 1992, pp. 2268-2272.

[19] J. G. Ziegler and N. B. Nichols. ”Optimum settings for automatic controllers.” Trans. ASME, vol. 64, pp. 759-768, 1942. Menlo Park, NJ: Addison-Wesley, 1970.

[20] Nasyrov, Rinat Aljendy, Raseel. (2018). Comprehensive comparison between hybrid fuzzy-PI and PSO-PI controllers based active power filter for compensation of harmonics and reactive power under different load conditions. 725-730. 10.1109/EIConRus.2018.8317195.

[21] MATLAB Fuzzy Logic Toolbox Documentation, ”Building Fuzzy Inference Systems Using Custom Functions.” https://www.mathworks.com/help/fuzzy/building-fuzzy-inference-systems-using-custom-functions.html

[22] MATLAB Fuzzy Logic Toolbox Documentation, ”What is Fuzzy Logic?” https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html

[23] TutorialsPoint, ”Fuzzy Logic - Membership Function.” https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_membership_function.htm

[24] YouTube, ”Fuzzy Logic Control System.” https://www.youtube.com/watch?v=__0nZuG4sTw&t=838s

[25] Fuzzy Logic Controller https://www.mathworks.com/help/fuzzy/fuzzylogiccontroller.html

[26] Sivanandam, S.N., S. Sumathi, and S.N Deepa, Introduction to Fuzzy Logic using Matlab, Springer-Verlag Berlin Heidelberg, 2007.

[27] Application of Kitamori’s method to stage positioning control using open loop transfer function. / Wakui, Shinji; Hoshino, Masashi; Akatsu, Kan. In: Nippon Kikai Gakkai Ronbunshu, C Hen/Transactions of the Japan Society of Mechanical Engineers, Part C, Vol. 70, No. 8, 08.2004, p. 2235-2242.