MINI PROJECT REPORT

MCTA 3371

DATE : 17TH JUNE 2025

SECTION : 2

SEMESTER 2, 2024/2025

NO	NAME	MATRIC
1.	AKMAL FAUZAN BIN AZHAR	2319531
2.	AIMAN SAIFULLAH BIN AMINNURLLAH	2319185
3.	MUHAMMAD DANISH FARHAN BIN AMIRUDDIN	2315423

INSTRUCTED BY :

AMIR AKRAMIN BIN SHAFIE

AZHAR BIN MOHD IBRAHIM

TABLE OF CONTENTS

1. INTRODUCTION

DC motors are extensively used in industrial and consumer applications requiring precise speed and position control. Traditional control methods like PID are reliable under linear and ideal conditions, but struggle with disturbances, noise, or nonlinearity. Computational Intelligence (CI) techniques such as Fuzzy Logic and Adaptive Neuro-Fuzzy Inference Systems (ANFIS) have emerged as robust alternatives that can learn and adapt to uncertainties in system behavior.

2. PROBLEM STATEMENT

While PID controllers are effective for linear time-invariant systems, their fixed gains limit performance under varying loads or disturbances. In contrast, fuzzy logic and ANFIS can handle nonlinearity and uncertainty more effectively. Therefore, this project investigates the comparative performance of PID, Fuzzy, and ANFIS controllers applied to a DC motor to evaluate their practical suitability.

3. OBJECTIVES

- To model and simulate a DC motor system.
- To design and implement intelligent control strategies using Fuzzy Logic and ANFIS.
- To compare the control performance of PID, Fuzzy Logic, and ANFIS controllers.

4. SYSTEM MODELLING & DATA COLLECTION

The DC motor is modeled using physical parameters based on its differential equations. The transfer function derived is:

$$P(s) = \frac{K}{(Js+b)(Ls+R)+K^2}$$

PARAMETERS	SYMBOL	VALUE
RESISTANCE	R	1 Ω
INDUCTANCE	L	0.5 H
INERTIA	J	0.01 kg·m^2
DAMPING	b	0.1 N·m·s
MOTOR CONSTANT	K	0.01

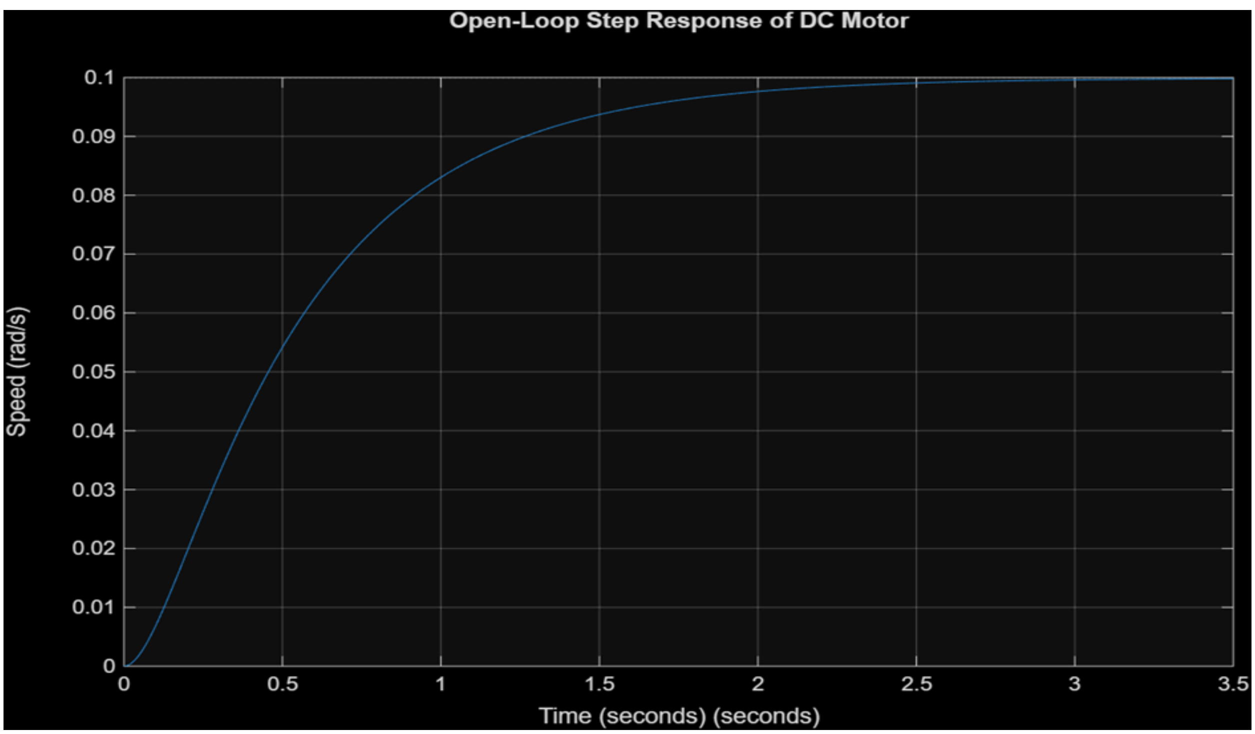


FIGURE 1 : GRAPH FOR OPEN-LOOP STEP RESPONSE OF DC MOTOR

5. MODELLING & SOLUTION

5.1 PID CONTROLLER

To design and implement a Proportional-Integral-Derivative (PID) controller for a DC motor to regulate its angular velocity in response to a step input. The aim is to tune the controller gains to minimize overshoot, reduce settling time, and ensure accurate tracking of the desired speed.

The PID controller was implemented with this tuned gains:

- Proportional gain (K_p) : 100
- Integral gain (K_i) : 200
- Derivative gain (K_d) : 10

The system responded with a fast rise time, minimal overshoot, and low steady-state error. Using MATLAB's step response analysis :

METRIC	**VALUE**
Rise Time (s)	0.132
Settling Time (s)	0.199
Overshoot (%)	1.03
Steady-State Value	1.00
Mean Squared Error	0.00642

The PID controller performed optimally due to its ability to quickly correct error (proportional), eliminate steady-state offset (integral), and dampen oscillations (derivative). This method is easy to implement but requires tuning for each system.

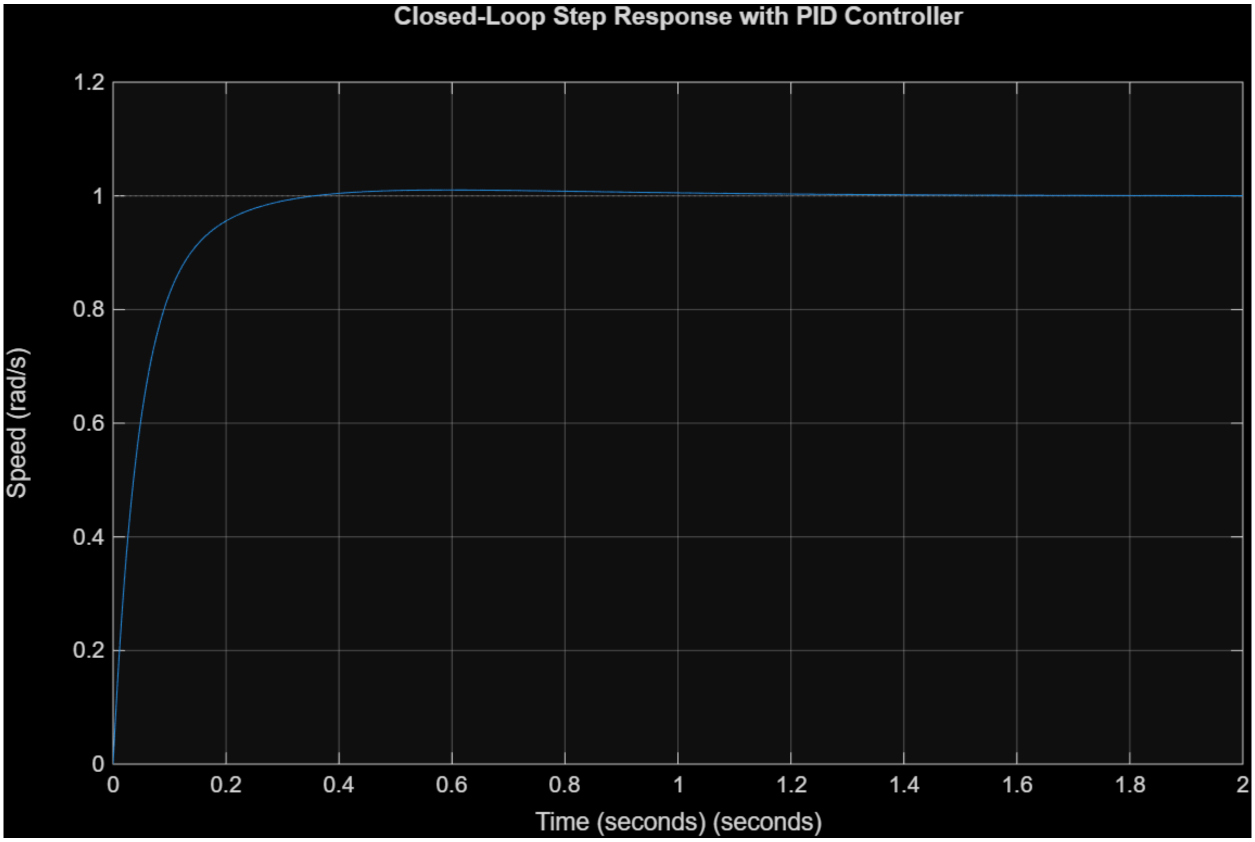


FIGURE 2 : GRAPH FOR CLOSED-LOOP STEP RESPONSE WITH PID CONTROLLER

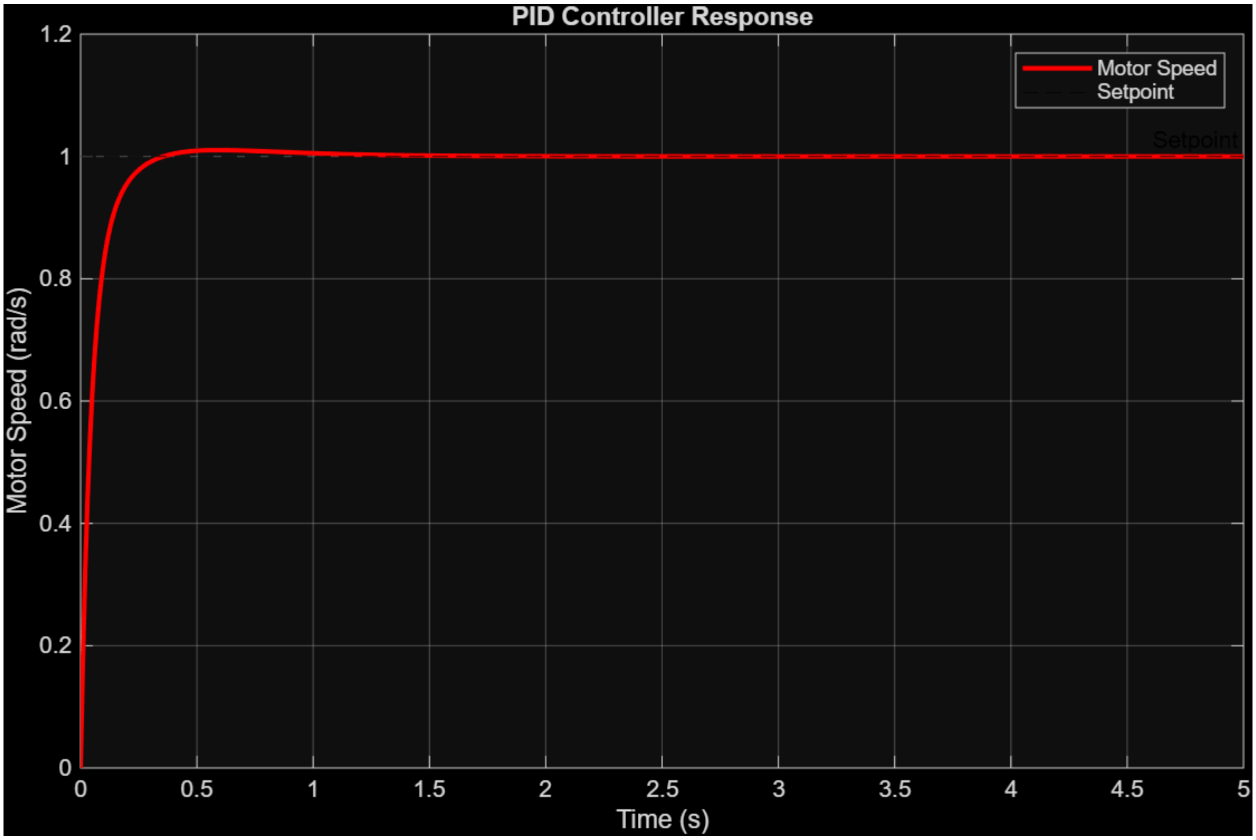


FIGURE 3 : GRAPH FOR PID CONTROLLER RESPONSE

5.2 MAMDANI FUZZY LOGIC CONTROLLER

The objective here is to implement a Mamdani-type fuzzy logic controller (FLC) for regulating the speed of a DC motor, and evaluate its performance compared to traditional PID control.

A fuzzy controller was built in MATLAB using the Fuzzy Logic Designer app. The controller has two input variables and one output variable:

- Input 1: Error (e) = reference speed - actual speed
- Input 2: Change in error (Δe)
- Output: Control signal (normalized, later scaled to voltage)

The input and output variables were normalized in the range [-1, 1]. Later, the output was scaled by multiplying by 12 to simulate voltage. For the membership functions (MF) is as the following:

- Type: Triangular
- Number of MFs per input 5
- Labels: NB (Negative Big), NS (Negative Small), ZE (Zero), PS (Positve Small), PB (Positive Big)

There are 9 rules which are:

1. If error (e) is NB (Negative Big) and change in error (Δe) is NB (Negative Big) then control is NB (Negative Big)
2. If error (e) is NB (Negative Big) and change in error (Δe) is ZE (Zero) then control is NB (Negative Big)
3. If error (e) is NB (Negative Big) and change in error (Δe) is PB (Positive Big) then control is NS (Negative Small)
4. If error (e) is ZE (Zero) and change in error (Δe) is NB (Negative Big) then control is NS (Negative Small)
5. If error (e) is ZE (Zero) and change in error (Δe) is ZE (Zero) then control is ZE (Zero)
6. If error (e) is ZE (Zero) and change in error (Δe) is PB (Positive Big) then control is PS (Positive Small)
7. If error (e) is PB (Positive Big) and change in error (Δe) is NB (Negative Big) then control is PS (Positive Small)
8. If error (e) is PB (Positive Big) and change in error (Δe) is ZE (Zero) then control is PB (Positive Big)
9. If error (e) is PB (Positive Big) and change in error (Δe) is PB (Positive Big) then control is PB (Positive Big)

The output from the fuzzy controller was applied to the motor model using a closed-loop simulation with Euler integration over 5 seconds. Results are as follows:

Metric	Value
Rise Time (s)	0.805
Settling Time (s)	—
Overshoot (%)	0.00
Steady-State Value	0.43
Mean Squared Error	0.38658

The fuzzy controller did not overshoot, but it significantly undershot the desired speed. The system never reached the 90% setpoint threshold, so MATLAB returned NaN for settling time.

From there we can see that the fuzzy controller was stable but conservative. The low output voltage (due to limited output scaling and weak rule gain) resulted in slow convergence and underperformance. Performance could be improved by redesigning membership functions or tuning the rule base. Despite underperforming compared to PID, the FLC is easier to design intuitively and does not require a mathematical model of the system.

FIGURE 4 : MEMBERSHIP FUNCTION PLOTS

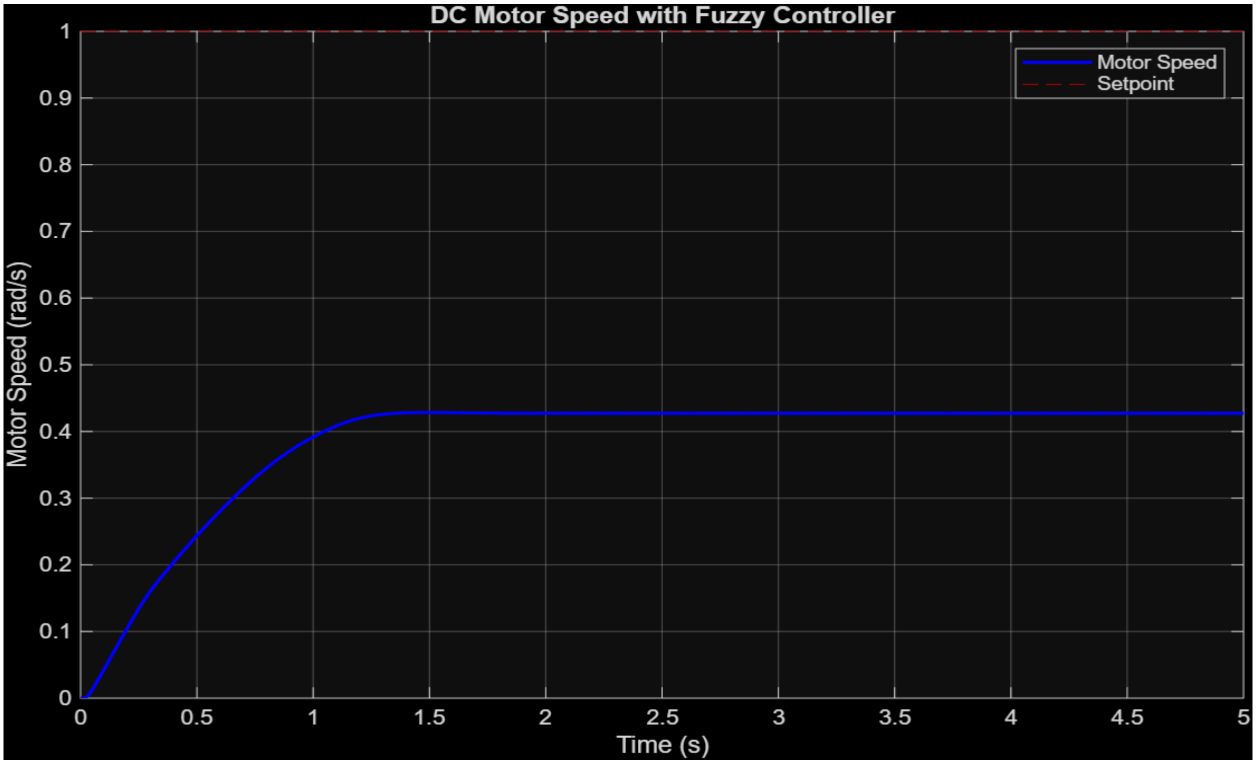


FIGURE 5 : GRAPH FOR DC MOTOR SPEED WITH FUZZY LOGIC CONTROLLER

5.3 ANFIS HYBRID CONTROLLER

To implement an Adaptive Neuro-Fuzzy Inference System (ANFIS) as a hybrid controller for the DC motor, combining fuzzy logic and neural network training. The goal is to improve upon the standard fuzzy logic controller by allowing data-driven learning of membership functions and rules.

ANFIS is a Sugeno-type inference system trained using supervised learning.

The training input is : Error (e) and Change in Error (Δe)

The target output is : Control signal

The ANFIS Hybrid Controller is achieved as below:

- Dataset generated from fuzzy controller using error range [-2, 2] and delta error [-2, 2]
- 2000 training samples created with MATLAB script (generate_anfis_data_ strong.m)
- Trained using grid partition (3 MFs/input), 20 epochs
- New dataset saved as anfis_training_data_strong.mat
- Trained FIS: anfis_fuzzy_strong.fis
- Output scaled and clamped: u = max(min(output x 30, 12), -12)

Results are as follows:

Metric	Value
Rise Time (s)	0.090
Settling Time (s)	—
Overshoot (%)	0.00
Steady-State Value	0.41
Mean Squared Error	0.2244

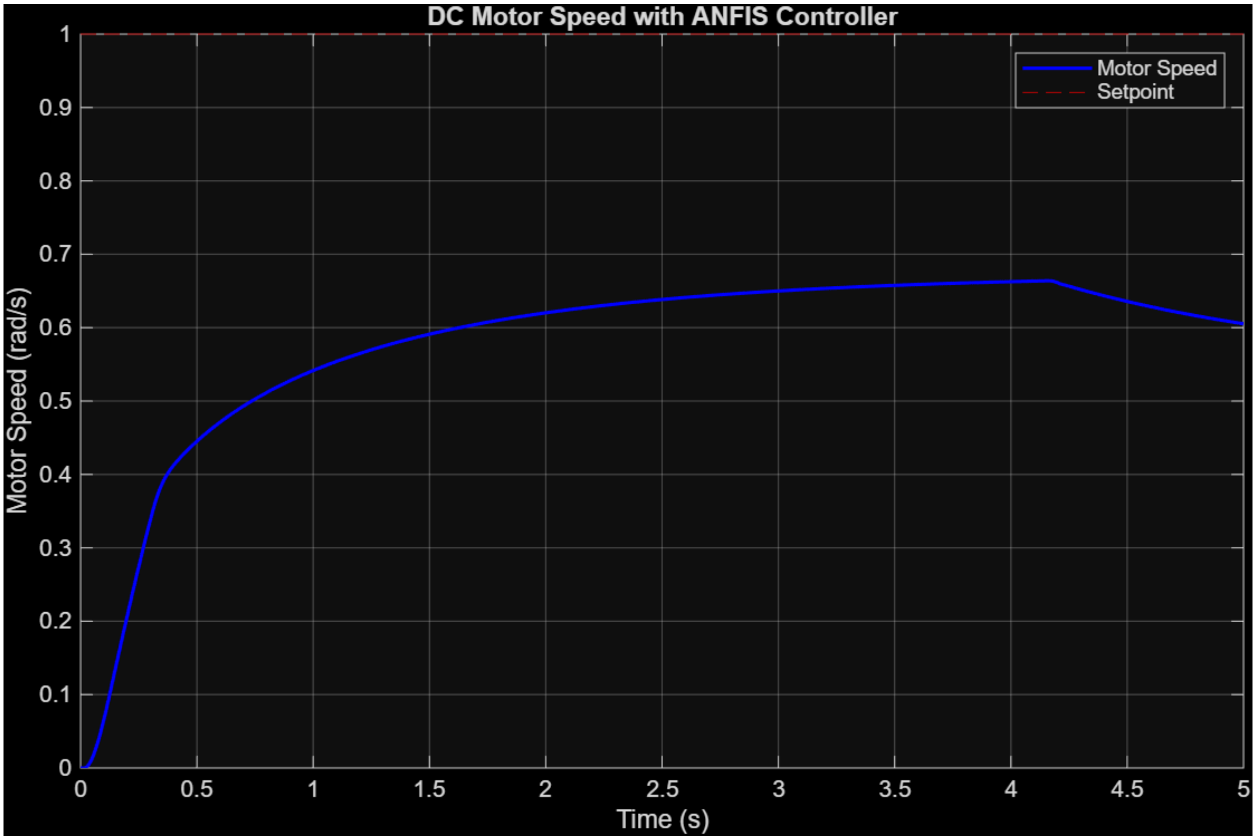


FIGURE 6: GRAPH FOR DC MOTOR SPEED WITH ANFIS CONTROLLER

6. SIMULATION & IMPLEMENTATION

All simulations were done in MATLAB :

- Simulation time : 5s
- Setpoint = 1 rad/s
- Performance metrics calculated: rise time, settling time, overshoot, mean squared error (MSE)
- Scripts created: dc_motor_model.m, anfis_motor_control.m, pid_motor_control.m, generate_anfis_data_strong.m

7. PERFORMANCE EVALUATION & COMPARISON

Metric	PID	Fuzzy	ANFIS
Rise Time (s)	0.132	0.805	0.090
Settling Time (s)	0.199	—	—
Overshoot (%)	1.03	0.00	0.00
Steady-State (rad/s)	1.00	0.43	0.41
MSE	0.00642	0.38658	0.2244

8. DISCUSSION

The PID controller delivered the most accurate and fastest response. The fuzzy controller, while stable and safe, suffered from underperformance due to limited control authority. ANFIS improved this significantly by adapting the fuzzy behavior through training. Although ANFIS did not reach the precision of PID, it showed strong potential for learning-based adaptation.

9. CONCLUSION

All three controllers were successfully implemented and compared. PID remained superior in precision, while fuzzy logic provided safety. ANFIS offered a promising hybrid method balancing intelligence and control adaptability. Future work may include disturbance rejection tests and real hardware validation.

10. REFERENCES

MathWorks. (nd). MATLAB Documentation. MathWorks

https://www.mathworks.com/help/matlab/

MathWorks. (n.d.). Design Fuzzy Inference Systems.

https://www.mathworks.com/help/fuzzy/

SUKMA Project. (2020, July 24). *Tutorial on how to create do Mamadani Method in fuzzy logic using MATLAB very simple way* [Video]. YouTube.

https://www.youtube.com/watch?v=K_Zu2wROa9E

11. APPENDIX

- DC Motor Model Parameters :

% DC Motor Parameters

R = 2.0; % Armature resistance (Ω)

L = 0.5; % Armature inductance (H)

J = 0.02; % Rotor inertia (kg·m^2)

B = 0.1; % Viscous damping (N·m·s)

K = 0.05; % Motor constant (Nm/A or V/rad/s)

%Transfer function of DC motor: P(s) = $\frac{K}{(Js+b)(Ls+R)+K^2}$

```
num = K;

den = [(J*L) (J*R + L*B) (B*R + K^2)];

motor_tf = tf(num, den);
```

- PID Controller Code :

% PID Controller Design

K_p = 100;

K_i = 200;

K_d = 10;

```
pid_ctrl = pid(Kp, Ki, Kd);

% Closed-loop system
sys_cl = feedback(pid_ctrl * motor_tf, 1);

% Step response and performance
step(sys_cl);
title('PID Step Response');
[y, t] = step(sys_cl);
ref = ones(size(y));
mse_pid = immse(y, ref);
stepinfo(sys_cl)
```

- Mamdani Fuzzy Logic Controller (Simulink Controlled) If using .fis file in Simulink, include :

```
% Load FIS controller in Simulink block (assumes 'fuzzy_motor_control.fis')
% Input: error, change in error
% Output: control (scaled by x18 before applying to plant)
```

- ANFIS Controller Training Data Generation :

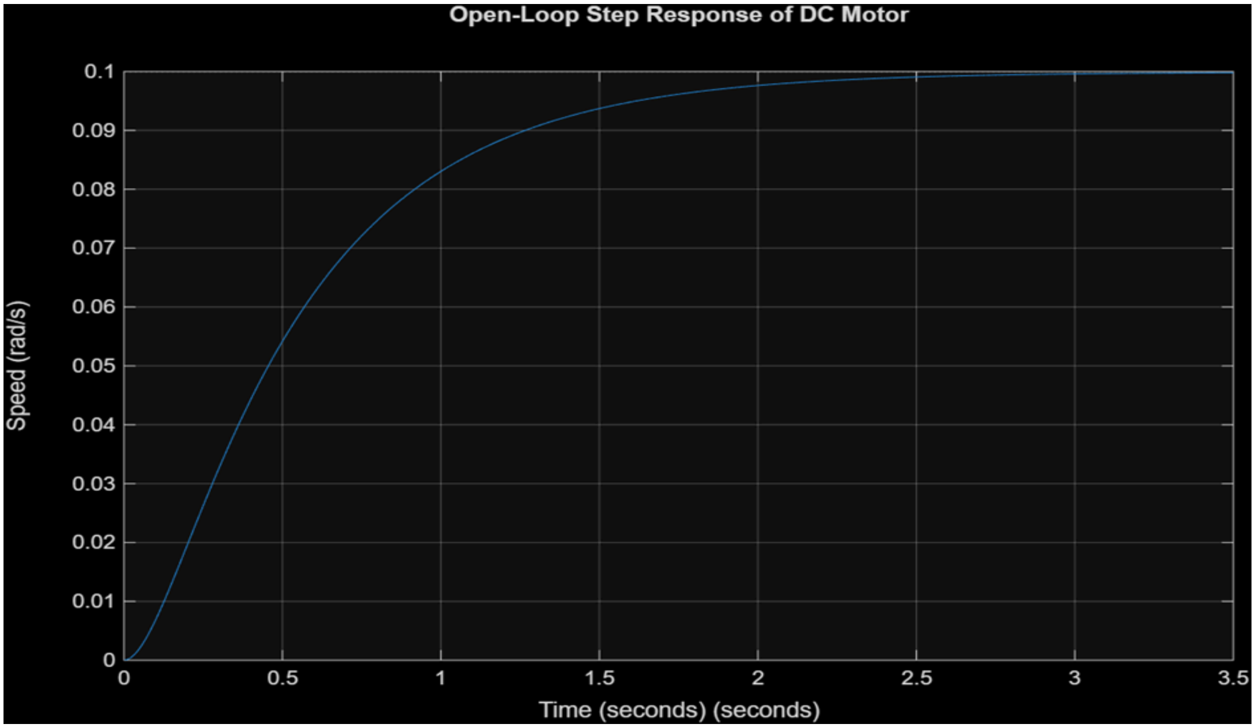
```
% Generate training data using Mamdani fuzzy logic controller

error_range = linspace(-2, 2, 100);

delta_error_range = linspace(-2, 2, 100);

[X, Y] = meshgrid(error_range, delta_error_range);


data = [];

for i = 1:numel(X)

    e = X(i);

    de = Y(i);

    u = evalfis([e, de], readfis('fuzzy_motor_control_strong.fis'));

    data(end+1, :) = [e, de, u];

end


% Save training data

save('anfis_training_data.mat', 'data');
```

- ANFIS Simulation with Trained Controller

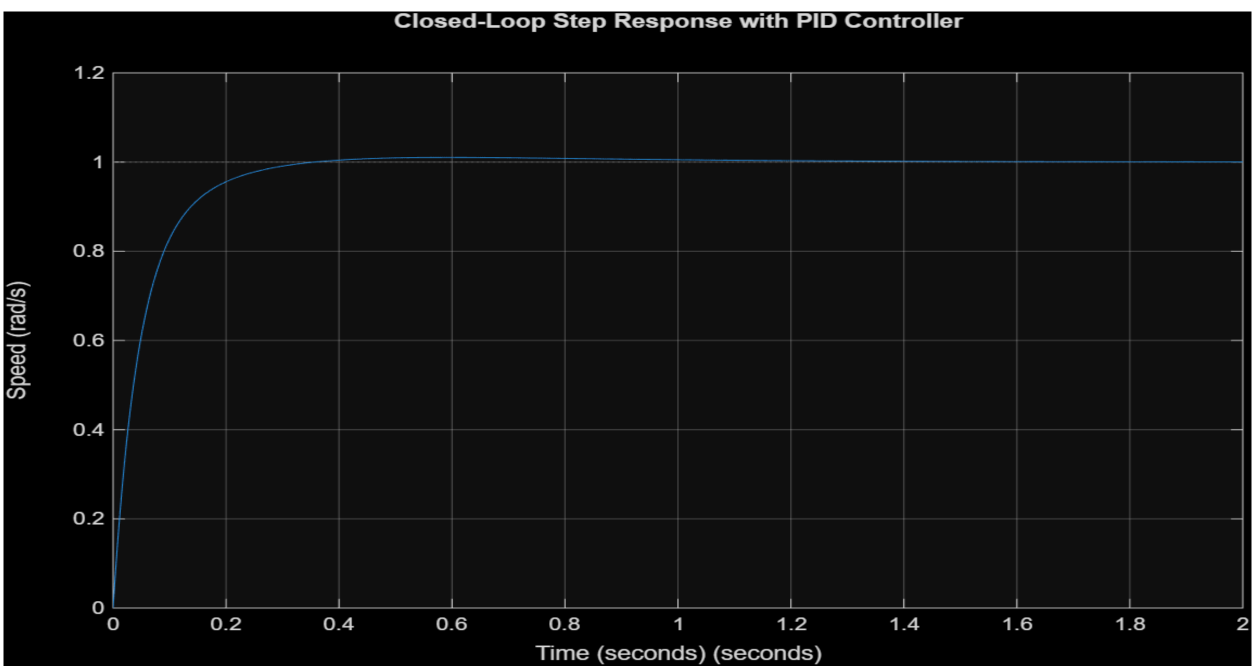
```
% Load ANFIS FIS file

anfis_fis = readfis('anfis_fuzzy_strong.fis');



% Simulation Parameters

t = 0:0.01:2;

ref = ones(size(t));

y = zeros(size(t));

e_prev = 0;

u = 0;

y_prev = 0;

for k = 2:length(t)

    e = ref(k) - y(k-1);

    de = e - e_prev;

    % Evaluate ANFIS controller output

    ctrl = evalfis([e, de], anfis_fis);

    u = max(min(ctrl * 30, 12), -12);  % Scale and clamp

% Simulate DC motor (Euler method)

    dy = (-B/J)*y(k-1) + (K/J)*u;
```

```
    y(k) = y(k-1) + 0.01*dy;


    e_prev = e;
end
% Plot results
plot(t, y, 'LineWidth', 2);
hold on; plot(t, ref, 'r--');
legend('Motor Speed', 'Reference');
title('ANFIS DC Motor Response');
xlabel('Time (s)'); ylabel('Speed (rad/s)');
% Performance
mse_anfis = immse(y, ref);
```

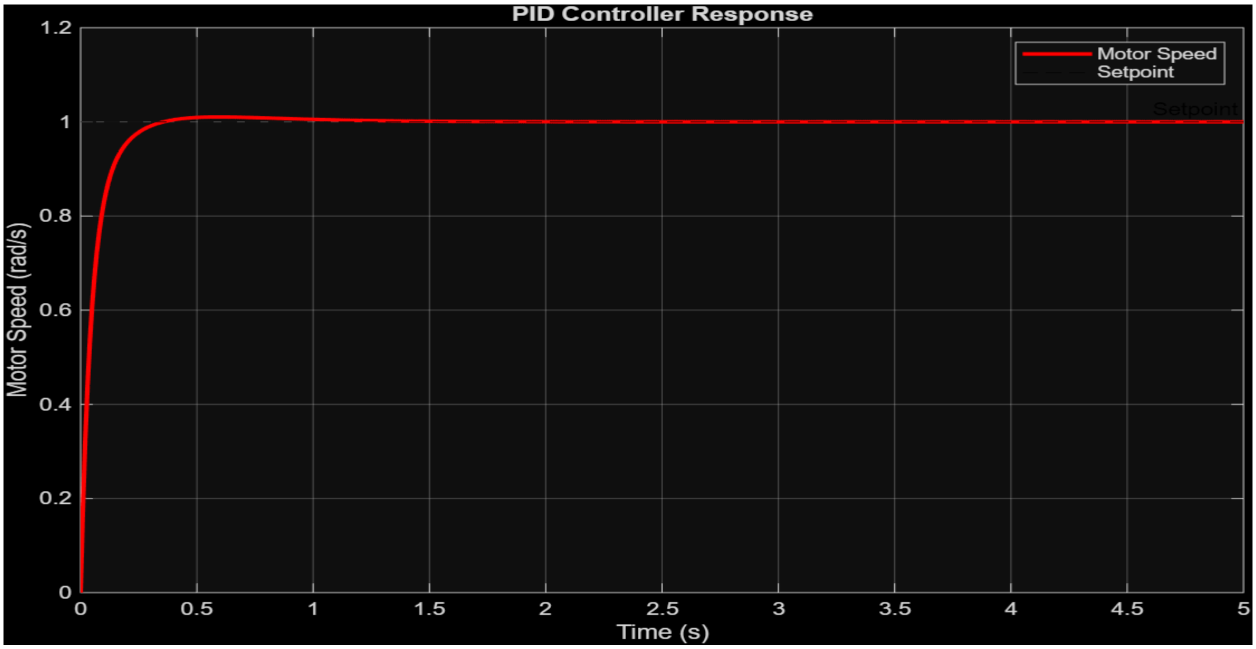
- GRAPH FOR OPEN-LOOP STEP RESPONSE OF DC MOTOR



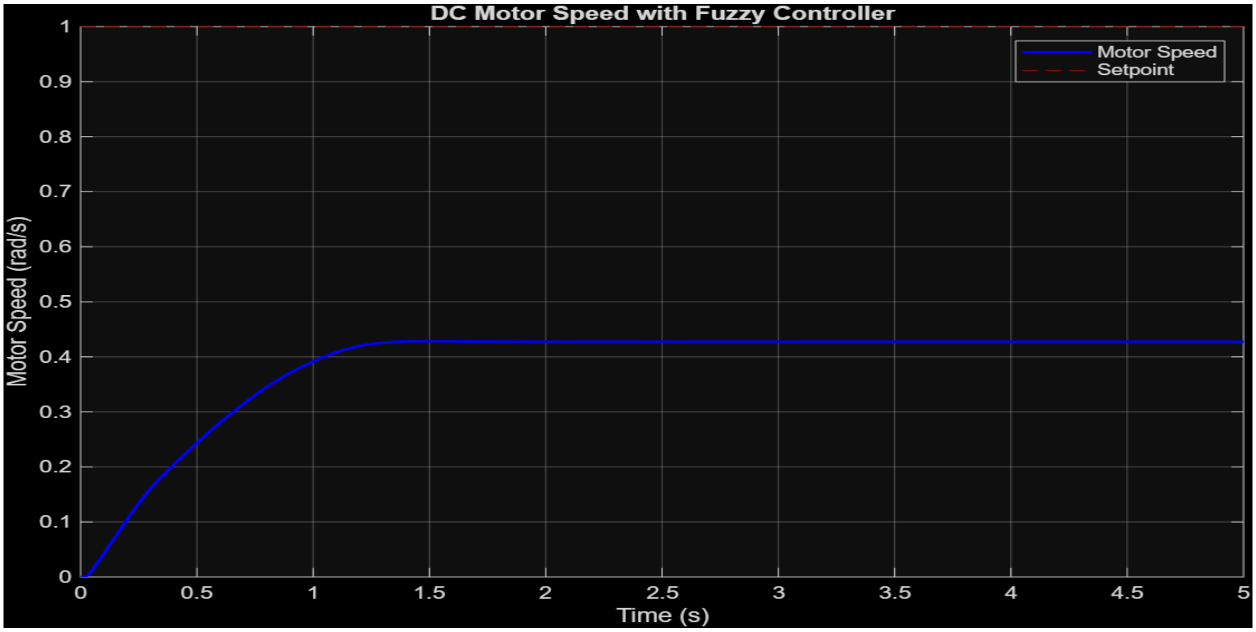
- GRAPH FOR CLOSED-LOOP STEP RESPONSE WITH PID CONTROLLER



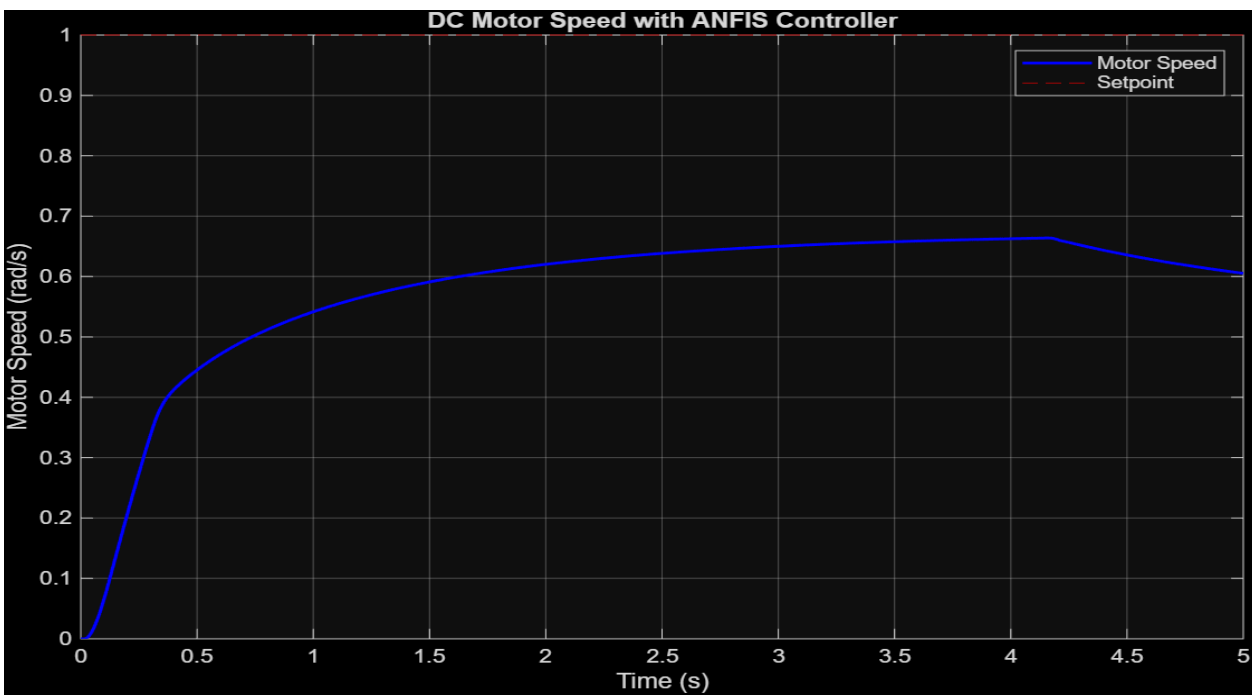
- GRAPH FOR PID CONTROLLER RESPONSE



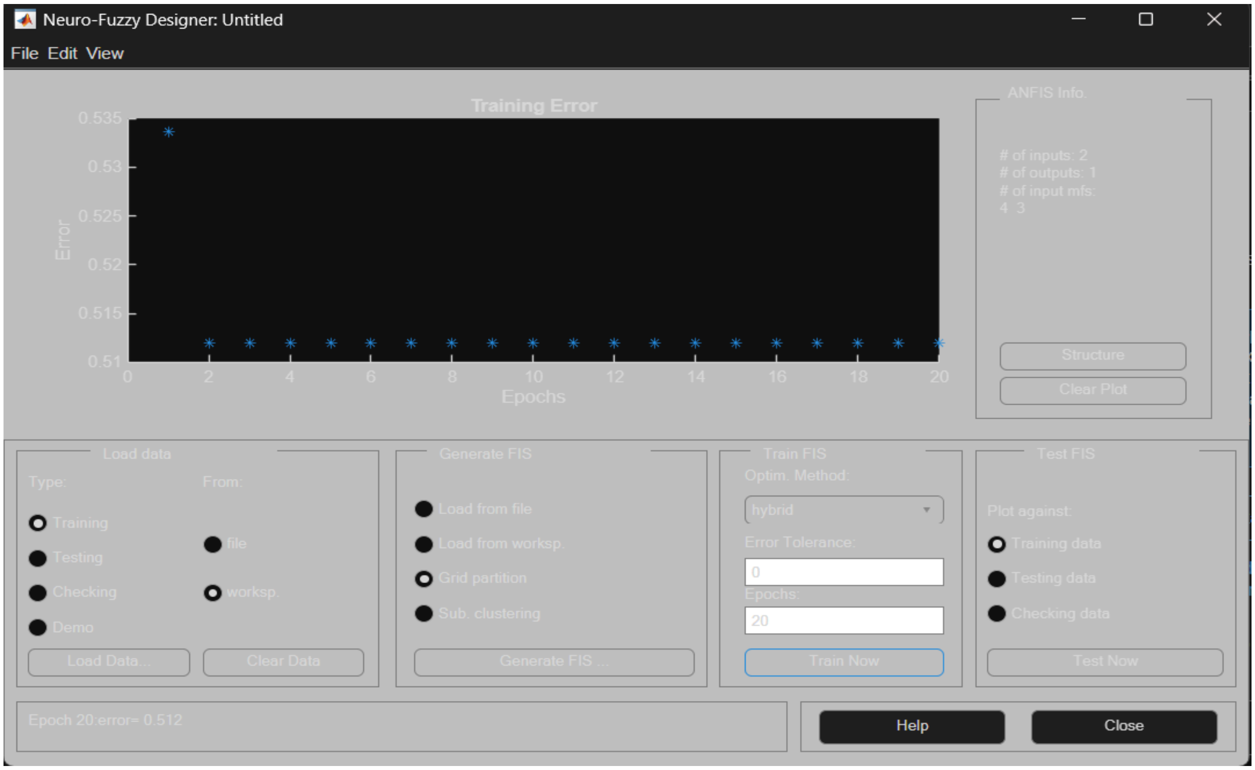
- GRAPH FOR DC MOTOR SPEED WITH FUZZY LOGIC CONTROLLER



- GRAPH FOR DC MOTOR SPEED WITH ANFIS CONTROLLER



- ANFIS Training GUI



12. INDIVIDUAL CONTRIBUTIONS

- AKMAL FAUZAN BIN AZHAR (2319531) : DC motor modelling, fuzzy rule base, ANFIS training and correction
- AIMAN SAIFULLAH BIN AMINNURLLAH (2319185) : PID tuning, performance metrics scripts, fuzzy scaling tuning
- MUHAMMAD DANISH FARHAN BIN AMIRUDDIN (2315423) : Documentation writing, report formatting, experimental retesting