Fuzzy Systems and Computational Intelligence: Satellite attitude control with fuzzy controllers

Dimitrios Karatis 10775, *Electrical and Computer Engineering, AUTH,*
14_Satellite.pdf

***Abstract*—This project focuses on the modeling, simulation, and control of a satellite attitude system using "classical" and fuzzy logic controllers. The first part introduces the mathematical model of the satellite orientation dynamics and analyzes its behavior under different input signals. In the second part, a fuzzy logic controller (FLC) is designed to regulate the satellite's angular position, taking into account nonlinearities and external disturbances. The controller is tuned to minimize steady-state error and improve transient response. Comparative results between classical PI controllers and the fuzzy-based approach are presented, demonstrating the advantages of fuzzy control in handling uncertainties and nonlinear dynamics.**

***Index Terms*—satellite attitude control, fuzzy logic controller, nonlinear systems, trajectory tracking, orientation dynamics, intelligent control.**

I. Design of a Classical PI Controller for Satellite Attitude Dynamics

In this section, we begin by modeling the satellite attitude system and designing a classical PI controller using MATLAB's Control Toolbox. The open-loop transfer function of the plant is given by

$$G_p(s) = \frac{10}{(s+1)(s+9)}.$$

The reference signal $r(t)$, output $y(t)$, and error $e(t) = r(t) - y(t)$ are used to evaluate the system response. The PI controller parameters K_p and K_I are tuned using root-locus methods, and the closed-loop stability is verified. The performance is analyzed in terms of step response, rise time, and overshoot. The obtained PI controller will serve as a benchmark for comparison with the fuzzy controller.

Design of Linear Controller

To achieve zero steady-state error in speed control, we select a PI linear controller of the form

$$G_c(s) = K_p + \frac{K_I}{s} = K_p \frac{s+c}{s}, \quad c = \frac{K_I}{K_p}.$$

The controller parameters are determined such that the following specifications are satisfied:

1) Overshoot for a step input less than 10%.
2) Rise time less than 1.2 seconds.

1) For a second-order system, the peak overshoot M_p is related to the damping ratio ζ by

This document was prepared by Dimitrios Karatis as part of the first assignment for the Simulation and Modeling of Dynamic Systems course at Aristotle University of Thessaloniki.

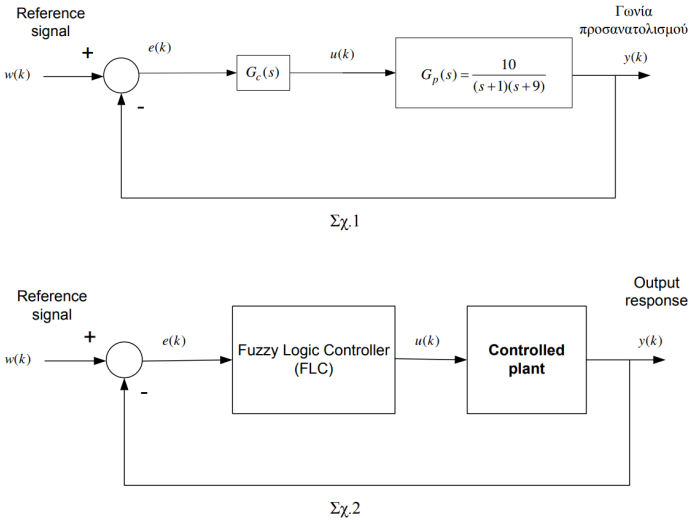

Fig. 1: Linear vs Fuzzy PI control

$$M_p = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}.$$

To satisfy the specification $M_p < 0.10$, we solve

$$0.10 = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \Rightarrow \ln(0.10) = \frac{-\pi\zeta}{\sqrt{1-\zeta^2}} \Rightarrow \quad \zeta \approx 0.59.$$

Hence, the damping ratio must satisfy

$$\zeta >\approx 0.59$$

to ensure the overshoot remains below 10%.

2) The rise time t_r is approximately related to the natural frequency ω_n by

$$t_r \approx \frac{1.8}{\omega_n}.$$

For a rise time specification of $t_r < 1.2$ s, this gives

$$\omega_n > \frac{1.8}{1.2} \approx 1.5 \text{ rad/s}.$$

Therefore, the system parameters must satisfy both conditions:

$$\zeta >\approx 0.59, \quad \omega_n >\approx 1.5 \text{ rad/s},$$

which will be useful when choosing a gain K from the root lotus below.

For the design of the linear controller, I followed the principles of classical automatic control using the `control toolbox` of MATLAB:

- Place the zero of the controller between the poles -1 and -9 of the plant, at a location close to the dominant pole, i.e., near -1: *I chose to place the zero at* $c = 1.5$
- Insert the open-loop transfer function into the system

$$\frac{10K(s+c)}{(s+1)(s+9)s}.$$

- Generate the root locus of the system using the `rlocus` command.
- Select a gain K from the root locus plot such that the closed-loop pole locations and damping ratio satisfy the design specifications: *To satisfy the desired damping ratio* $\zeta > 0.59$ *and natural frequency* $\omega_n > 1.5$*, I selected the gain* $K = 2.41$*, with the help of Fig.2*
- *With this value, the PI controller parameters were calculated as*

$$K_p = K = 2.41, \qquad K_I = K_p \cdot c = 2.41 \times 1.5 = 3.615.$$

- Compute the closed-loop transfer function (with unit negative feedback) using the MATLAB command `feedback(sys-open-loop, 1, -1)`.
- Compute the step response of the closed-loop system using the commands `step(sys-closed-loop)`, `lsim`, `plot`, etc.
- If the design specifications are satisfied, the procedure is complete. Otherwise, iterate by selecting another value of the gain K.
- For the optimal gain K obtained, calculate the proportional gain K_p and the integral gain K_I of the linear controller.

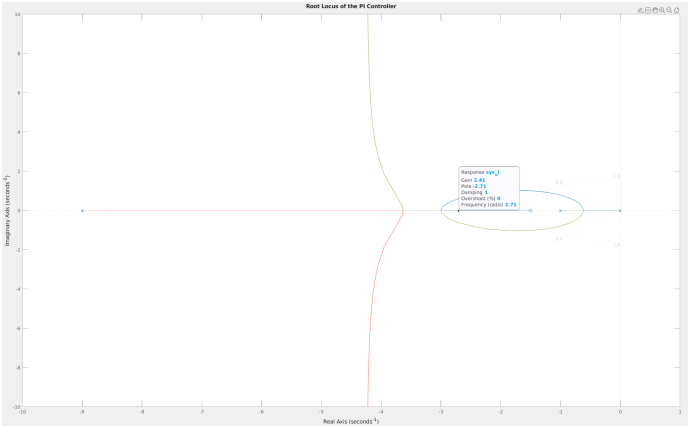

Fig. 2: Root locus of the Linear PI controller

Below we can see the step response of the system when using the linear PI controller, along with some relevant information for the system and also the control signal response as well. As observed, the rise time is

$$t_r = 0.4857 \text{ s} < 1.2 \text{ s},$$

and the peak overshoot is

$$M_p \approx 8.12\% < 10\%.$$

Hence, it is safe to say that the system satisfies the performance constraints.

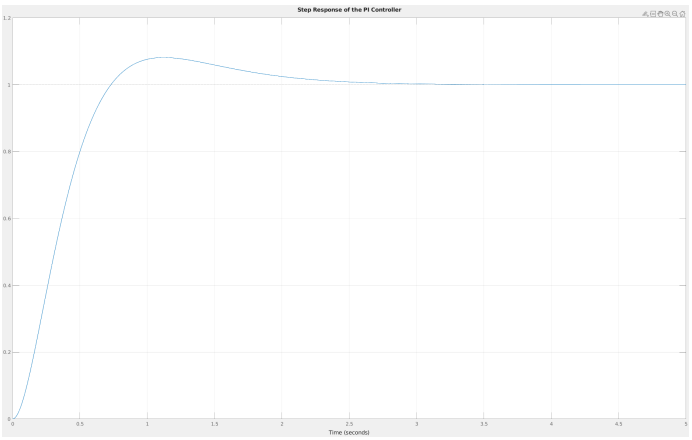

Fig. 3: Step Response of the Linear PI controller

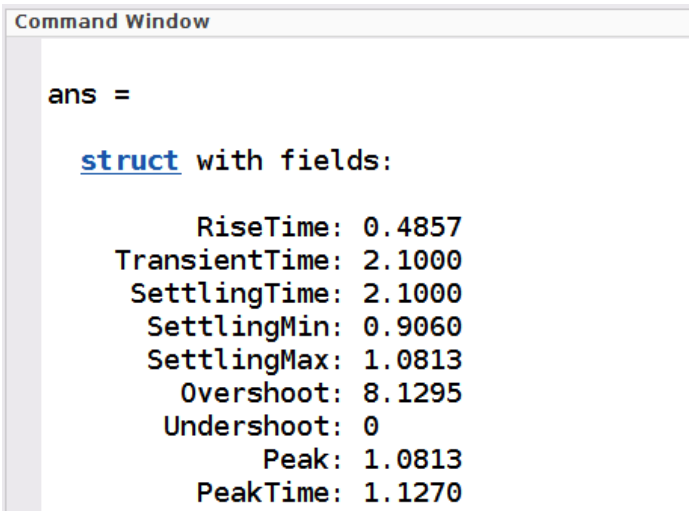

Fig. 4: Results of the Linear PI controller system, as shown using the *stepinfo* command, under a unit step input

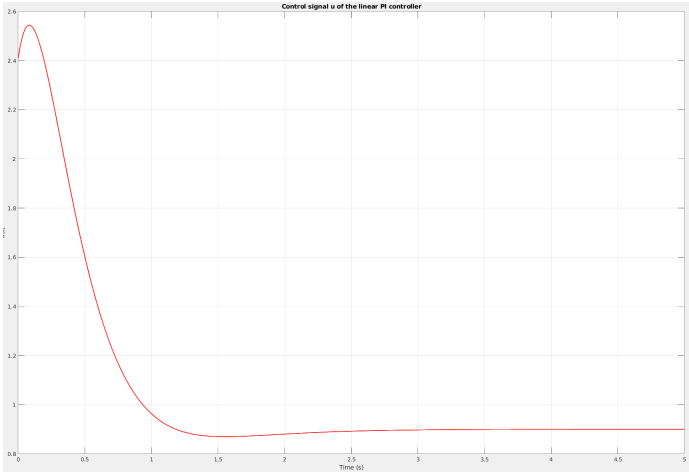

Fig. 5: Control signal u response of the linear PI controller

```
Command Window

ans =

  struct with fields:

        RiseTime: 0.4856
    TransientTime: 2.1000
     SettlingTime: 2.1000
      SettlingMin: 54.4888
      SettlingMax: 64.8782
        Overshoot: 8.1303
       Undershoot: 0
             Peak: 64.8782
         PeakTime: 1.1200
```

Fig. 6: Results of the Linear PI controller system, as shown using the *stepinfo* command, under maximum step input (60)

Detailed code can be found in the ***Project1_linear_unit.m*** *and* ***Project1_linear_maximum.m***

II. Design and Simulation of a Fuzzy Logic Controller

In this part, a Fuzzy Logic Controller (FLC) is developed for the same satellite attitude system. The inputs of the controller are the error $e(k)$ and its derivative $\Delta e(k)$, while the output is the control signal $u(k)$. Membership functions are defined for linguistic variables such as Negative Large (NL), Negative Medium (NM), Zero (ZR), Positive Small (PS), and Positive Large (PL). The inference mechanism is based on the Larsen method, and defuzzification is carried out using the Center of Area (COA) technique.

Simulation results illustrate the ability of the FLC to track reference trajectories more effectively than the PI controller, especially under parameter variations and external disturbances. The analysis confirms that fuzzy control provides robustness and adaptability to uncertain conditions in satellite attitude dynamics.

Design of the Fuzzy Controller

- The closed-loop system implementation will be carried out in discrete time with a sampling period of $T = 0.01$ sec.
- The linguistic variables of the error E are described by seven linguistic values, as shown in Fig. 7.
- The linguistic variables of the change of error $\dot{E}$ are described by nine linguistic values, as shown in Fig. 8.
- The linguistic variables of the change of the control signal $\dot{U}$ are described by nine linguistic values, as shown in Fig. 8.

The reference signal r may vary within the range $[0, 60]$.

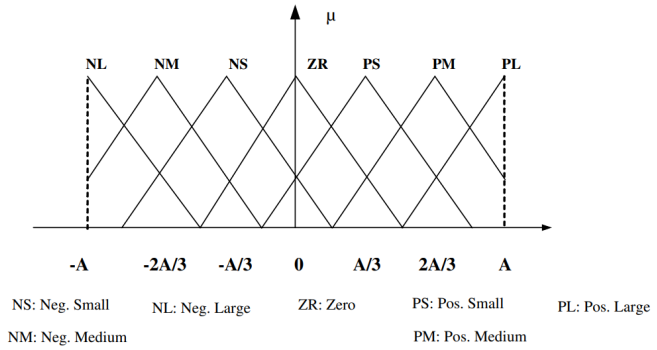

Fig. 7: Linguistic variables of the error E

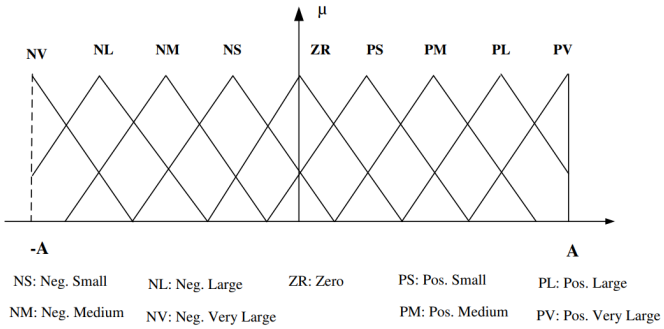

Fig. 8: The linguistic variables of the change of error $\dot{E}$ and of the control signal $\dot{U}$

Characteristics of the FLC

The characteristics of the fuzzy controller are the following:

- Singleton fuzzifier
- The AND operator is implemented using the *min* operator
- The implication function is implemented using Larsen's product
- The ALSO operator is implemented using the *max* operator
- The defuzzification is carried out using the COA (Center of Area) technique

Project Requirements

- Perform scaling of the error and error change so that the normalized variables vary within the range $[-1, 1]$.

- Develop the rule base of the fuzzy controller according to meta-rules ensuring proper closed-loop performance.
- Write a program in MATLAB that implements the fuzzy controller–plant closed-loop system.

A. Scaling of Input Variables

Given that

$$e(k) = r(k) - y(k)$$

the maximum error at the input of the controller will be

$$e_{\max} = r_{\max} - y_{\min} = r_{\max} - 0 = r_{\max}$$

and the minimum

$$e_{\min} = r_{\min} - y_{\max} = 0 - y_{\max} = y_{\max}$$

And since, the reference signal r may vary within the range of $[0, 60]$:

$$r_{\max} = 60 \qquad r_{min} = 0$$

Therefore, $e_{\max} = 150$ rad/sec, and thus

$$e(k) \in [-60,\ 60]$$

For the error change

$$\Delta e(k) \in [-e_{min},\ e_{max}] = [-60,\ 60]$$

For the normalization of the input variables, we multiply $e(k)$ and $\Delta e(k)$ by $1/50$, and then:

$$\Delta e_n(k),\ e_n(k) \in [-1,\ 1]$$

B. Fuzzy Rule Base

The controller used is of type fuzzy PI (FZ-PI). The rule base of the controller is derived from the relation:

$$\dot{U} = e + \Delta e$$

Since one input is divided into seven (7) linguistic terms and the others into nine (9), the fuzzy rule base will consist of a total of $9 \times 7 = 63$ fuzzy rules. Thus, the fuzzy rule base of the controller is obtained as follows:

$\Delta e_n \downarrow e_n \rightarrow$	NL	NM	NS	ZR	PS	PM	PL
PV	PS	PM	PL	PV	PV	PV	PV
PL	ZR	PS	PM	PL	PV	PV	PV
PM	NS	ZR	PS	PM	PL	PV	PV
PS	NM	NS	ZR	PS	PM	PL	PV
ZR	NL	NM	NS	ZR	PS	PM	PL
NS	NV	NL	NM	NS	ZR	PS	PM
NM	NV	NV	NL	NM	NS	ZR	PS
NL	NV	NV	NV	NL	NM	NS	ZR
NV	NV	NV	NV	NV	NL	NM	NS

TABLE I: The 63 fuzzy rules forming the fuzzy controller rule base

or for better visualization:

$\Delta e_n \downarrow e_n \rightarrow$	NL	NM	NS	ZR	PS	PM	PL
NV	NV	NV	NV	NV	NL	NM	NS
NL	NV	NV	NV	NL	NM	NS	ZR
NM	NV	NV	NL	NM	NS	ZR	PS
NS	NV	NL	NM	NS	ZR	PS	PM
ZR	NL	NM	NS	ZR	PS	PM	PL
PS	NM	NS	ZR	PS	PM	PL	PV
PM	NS	ZR	PS	PM	PL	PV	PV
PL	ZR	PS	PM	PL	PV	PV	PV
PV	PS	PM	PL	PV	PV	PV	PV

TABLE II: The 63 fuzzy rules forming the fuzzy controller rule base

*Detailed code can be found in the **createFIS_PI.m** file.*

C. Control Loop Simulation

The `fuzzyPI_simulation` function implements a discrete-time fuzzy proportional-integral (PI) controller for a linear system described in state-space form. At each simulation step, the tracking error e between the reference signal r and the system output y, as well as the error derivative Δe, are computed. These signals are scaled by tuning gains (K_e and K_d) to normalize them within the fuzzy controller's input range and clamped to $[-1, 1]$, ensuring the Mamdani-type fuzzy inference system (FIS) operates effectively. The FIS evaluates e and Δe to produce an incremental control action Δu_{fuzzy}, which is scaled and integrated to generate the control signal u. This incremental approach mimics a traditional PI controller, while the fuzzy logic provides a nonlinear mapping from error states to control adjustments. The control input is applied to the system via the discrete-time state-space equations $x = Ax + Bu$ and $y = Cx + Du$. Alternatively, the same fuzzy PI controller can be implemented using a similar `Simulink` model to the one described in the course's material, although I did not choose that approach.

*Detailed code can be found in the **fuzzyPI _simulation.m** matlab file.*

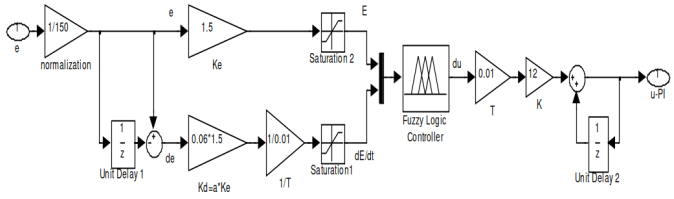

Fig. 9: The Simulink model used in the course's presentations

SCENARIO 1

A. Controller Design and Responses

Initial Gains of the Fuzzy PI Controller

The initial gains of the fuzzy logic PI controller (FLC) are calculated based on the gains of the linear PI controller, as follows

$$K_e = 1$$

$$K = \frac{K_p}{F[a \cdot K_e]} = \frac{2.41}{0.667} = 3.615$$

where

$$a = T_i = \frac{K_p}{K_I} = \frac{1}{c} = \frac{1}{1.5} = 0.667 \; and \; K_d = a \cdot K_e$$

Thus, the initial gains of the fuzzy PI FLC are:

K_e	1.0000
K_d	0.667
K	3.615

TABLE III: The initial gains of the fuzzy PI controller

The results for the FLC PI under maximum step input (60) are the following:

```
ans =

  struct with fields:

        RiseTime: 1.9557
   TransientTime: 3.6779
    SettlingTime: 3.6779
     SettlingMin: 54.0674
     SettlingMax: 59.7523
       Overshoot: 0
      Undershoot: 0
            Peak: 59.7523
        PeakTime: 4.9900

fx >>
```

Fig. 10: Results of the Fuzzy PI controller system under maximum step input, as shown using the *stepinfo* command, for the initial gains

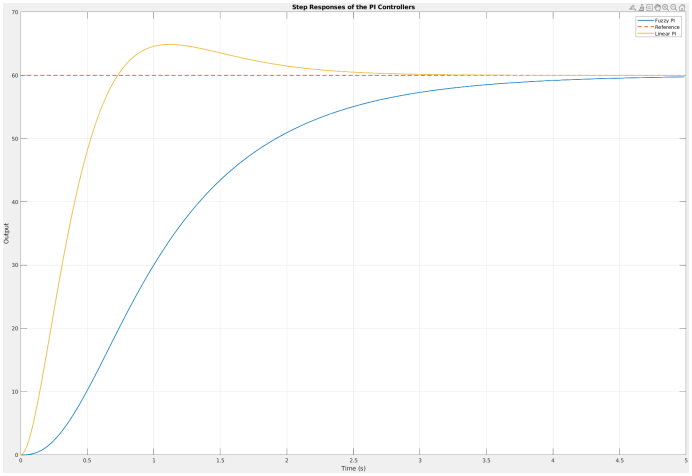

Fig. 11: Step Responses comparison of the PI Controllers under maximum step input for the initial gains

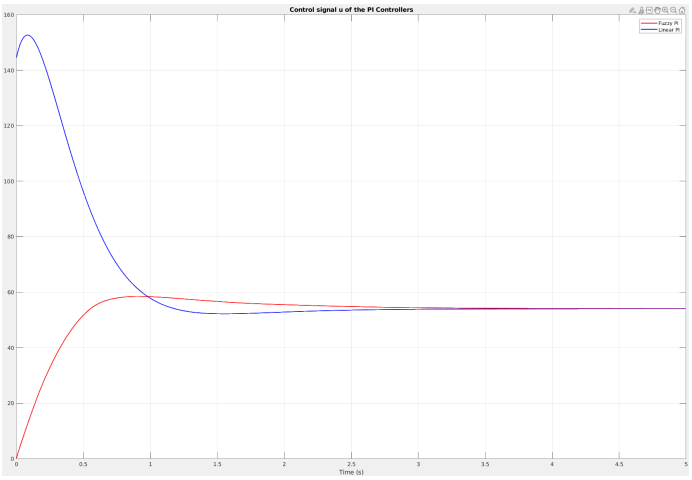

Fig. 12: Control signal u comparison of the PI under maximum step input for the initial gains

Detailed code can be found in the ***Project1_fuzzy_initial.m*** *file.*

From the comparison of the above with the corresponding plots and tables for the linear PI controller (under maximum step input 60), it is evident that the fuzzy PI controller with the initial gains $K_e = 1$, $K = 3.615$ $K_d = 0.667$ is slower (in terms of rise time, settling time, and peak) compared to the linear PI controller. Therefore the input gains E (K_e) and $\dot{E}$ (K_d) as well as the output gain $\dot{U}$ (K) of the fuzzy PI controller need to be adjusted (tuned).

Comparative Gain Design

In order to improve the behavior of the PI FLC, we will apply the comparative method of tuning the controller gains:

COMPARATIVE TUNING METHOD FOR FZ-PI

1) The gains K_e/K are adjusted to achieve a fast response and small steady-state error. The value of K_e should not be too large in order to avoid saturation of the input. By keeping the product K_eK constant, we can adjust K_e and K to achieve better performance of the controller.
2) The ratio α is adjusted to achieve a fast response and small steady-state error in FZ-PI control.

After determining the initial scaling gains, steps (1) and (2) are repeated until a satisfactory performance is achieved.

Since we want to increase the response speed of the controller, I adjusted the values of K_e/K (increase) as well as the ratio α (decrease), as follows:

$$\begin{cases} K'_e = 1.5 \cdot K_e \\ K' = 1.2 \cdot K \end{cases}$$

$$\alpha' = \frac{\alpha}{2.5}$$

$$K'_d = \alpha' \cdot K'_e = \frac{1.5}{2.5} \cdot \alpha \cdot K_e = 0.6 \cdot K_d$$

Therefore, the gains of the PI FLC will be:

K'_e	1.5
K'_d	0.4
K'	4.338

TABLE IV: First attempt at tuning the gains of the fuzzy PI controller

and the results were:

```
Command Window

ans =

  struct with fields:

        RiseTime: 0.8345
    TransientTime: 2.5711
     SettlingTime: 2.5711
      SettlingMin: 54.3334
      SettlingMax: 67.0110
        Overshoot: 11.6850
       Undershoot: 0
             Peak: 67.0110
         PeakTime: 1.7900
```

Fig. 13: Results of the Fuzzy PI controller system with the above set of gains

The fuzzy controller continues to show weaker performance compared to the linear controller under a maximum step input. I then tested different gain values and determined a set of parameters that improve the fuzzy controller's performance:

$$\begin{cases} K'_e = 1.35 \cdot K_e \\ K' = 4.3 \cdot K \end{cases}$$

$$\alpha' = \frac{\alpha}{2.7}$$

$$K'_d = \alpha' \cdot K'_e = \frac{1.35}{2.7} \cdot \alpha \cdot K_e = 0.5 \cdot K_d$$

Therefore, the final gains of the PI FLC will be:

K'_e	1.35
K'_d	0.334
K'	15.5445

TABLE V: The final gains of the fuzzy PI controller

and the results are now:

```
Command Window

ans =

  struct with fields:

        RiseTime: 0.4819
    TransientTime: 1.1467
     SettlingTime: 1.1467
      SettlingMin: 54.1279
      SettlingMax: 62.3831
        Overshoot: 3.9718
       Undershoot: 0
             Peak: 62.3831
         PeakTime: 0.9800
```

Fig. 14: Results of the Fuzzy PI controller system with the final set of gains

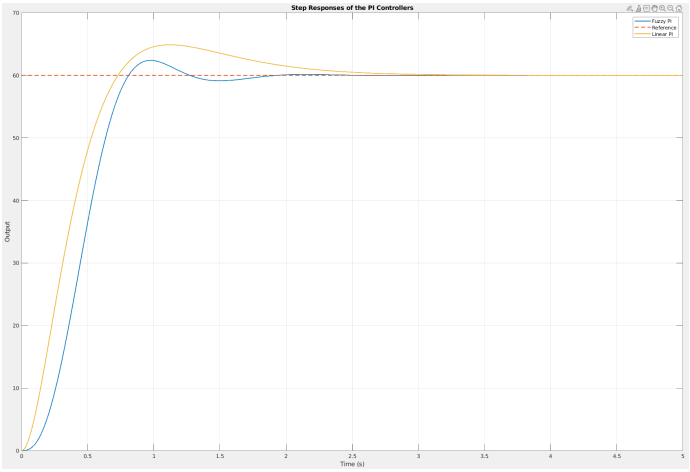

Fig. 15: Step Responses comparison of the PI Controllers with the final gains

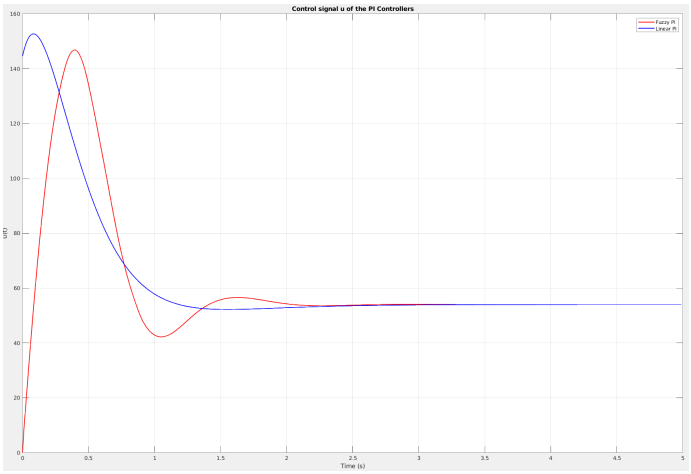

Fig. 16: Control signal u comparison of the PI Controllers with the final gains

As we can see, the constraints (rise time $< 0.6s$ and overshoot $< 7\%$) are being satisfied, while the system also demonstrates improved characteristics compared to the linear controller.

Detailed code can be found in the ***Project1_fuzzy_final.m*** *file.*

EFFECTS OF SCALING GAINS ON THE RESPONSE

- Increasing the value of the gain K is equivalent to increasing the value of K_P. The response becomes faster and the steady-state error decreases. However, a large value of K may cause oscillations or even instability.
- Decreasing the value of α in an FZ-PI controller is equivalent to increasing the value of the fuzzy gain K_I. The response becomes faster and the steady-state error decreases. However, a very small value of α may cause large oscillations and tends to destabilize the system.
- Increasing K_e makes the controller more sensitive to small errors, leading to a faster response and reduced steady-state error. However, if K_e is set too large, the input to the fuzzy controller may saturate, causing aggressive control action and possible instability. Conversely, if K_e is too small, the controller reacts slowly, which increases the steady-state error and reduces overall performance.

B. Operation of the controller's rule base and conclusions

In this part of the assignment, we are asked to analyze the operation of the fuzzy controller for the case where the error e belongs to the fuzzy set NS and the error derivative Δe belongs to the fuzzy set ZR. The procedure involves identifying which fuzzy rules are activated under this excitation and representing them graphically, followed by determining the partial conclusions of each rule. Next, the overall fuzzy inference result must be obtained using the corresponding defuzzification method. Finally, we need to interpret and comment on the controller's response in this specific case.

FUZZY RULE BASE

The complete fuzzy rule base is given below:

1) If ($error$ is NL) and ($\Delta error$ is NV) then (Δu is NV)
2) If ($error$ is NM) and ($\Delta error$ is NV) then (Δu is NV)
3) **If ($error$ is NS) and ($\Delta error$ is NV) then (Δu is NV)**
4) If ($error$ is ZR) and ($\Delta error$ is NV) then (Δu is NV)
5) If ($error$ is PS) and ($\Delta error$ is NV) then (Δu is NL)
6) If ($error$ is PM) and ($\Delta error$ is NV) then (Δu is NM)
7) If ($error$ is PL) and ($\Delta error$ is NV) then (Δu is NS)
8) If ($error$ is NL) and ($\Delta error$ is NL) then (Δu is NV)
9) If ($error$ is NM) and ($\Delta error$ is NL) then (Δu is NV)
10) **If ($error$ is NS) and ($\Delta error$ is NL) then (Δu is NV)**
11) If ($error$ is ZR) and ($\Delta error$ is NL) then (Δu is NL)
12) If ($error$ is PS) and ($\Delta error$ is NL) then (Δu is NM)
13) If ($error$ is PM) and ($\Delta error$ is NL) then (Δu is NS)
14) If ($error$ is PL) and ($\Delta error$ is NL) then (Δu is ZR)
15) If ($error$ is NL) and ($\Delta error$ is NM) then (Δu is NV)
16) If ($error$ is NM) and ($\Delta error$ is NM) then (Δu is NV)
17) **If ($error$ is NS) and ($\Delta error$ is NM) then (Δu is NL)**
18) If ($error$ is ZR) and ($\Delta error$ is NM) then (Δu is NM)
19) If ($error$ is PS) and ($\Delta error$ is NM) then (Δu is NS)
20) If ($error$ is PM) and ($\Delta error$ is NM) then (Δu is ZR)
21) If ($error$ is PL) and ($\Delta error$ is NM) then (Δu is PS)
22) If ($error$ is NL) and ($\Delta error$ is NS) then (Δu is NV)
23) If ($error$ is NM) and ($\Delta error$ is NS) then (Δu is NL)
24) **If ($error$ is NS) and ($\Delta error$ is NS) then (Δu is NM)**
25) If ($error$ is ZR) and ($\Delta error$ is NS) then (Δu is NS)
26) If ($error$ is PS) and ($\Delta error$ is NS) then (Δu is ZR)
27) If ($error$ is PM) and ($\Delta error$ is NS) then (Δu is PS)
28) If ($error$ is PL) and ($\Delta error$ is NS) then (Δu is PM)
29) **If ($error$ is NL) and ($\Delta error$ is ZR) then (Δu is NL)**
30) **If ($error$ is NM) and ($\Delta error$ is ZR) then (Δu is NM)**
31) **If ($error$ is NS) and ($\Delta error$ is ZR) then (Δu is NS)**
32) **If ($error$ is ZR) and ($\Delta error$ is ZR) then (Δu is ZR)**
33) **If ($error$ is PS) and ($\Delta error$ is ZR) then (Δu is PS)**
34) **If ($error$ is PM) and ($\Delta error$ is ZR) then (Δu is PM)**
35) **If ($error$ is PL) and ($\Delta error$ is ZR) then (Δu is PL)**
36) If ($error$ is NL) and ($\Delta error$ is PS) then (Δu is NM)
37) If ($error$ is NM) and ($\Delta error$ is PS) then (Δu is NS)
38) **If ($error$ is NS) and ($\Delta error$ is PS) then (Δu is ZR)**
39) If ($error$ is ZR) and ($\Delta error$ is PS) then (Δu is PS)
40) If ($error$ is PS) and ($\Delta error$ is PS) then (Δu is PM)
41) If ($error$ is PM) and ($\Delta error$ is PS) then (Δu is PL)
42) If ($error$ is PL) and ($\Delta error$ is PS) then (Δu is PV)
43) If ($error$ is NL) and ($\Delta error$ is PM) then (Δu is NS)
44) If ($error$ is NM) and ($\Delta error$ is PM) then (Δu is ZR)
45) **If ($error$ is NS) and ($\Delta error$ is PM) then (Δu is PS)**
46) If ($error$ is ZR) and ($\Delta error$ is PM) then (Δu is PM)
47) If ($error$ is PS) and ($\Delta error$ is PM) then (Δu is PL)
48) If ($error$ is PM) and ($\Delta error$ is PM) then (Δu is PV)
49) If ($error$ is PL) and ($\Delta error$ is PM) then (Δu is PV)
50) If ($error$ is NL) and ($\Delta error$ is PL) then (Δu is ZR)
51) If ($error$ is NM) and ($\Delta error$ is PL) then (Δu is PS)
52) **If ($error$ is NS) and ($\Delta error$ is PL) then (Δu is PM)**
53) If ($error$ is ZR) and ($\Delta error$ is PL) then (Δu is PL)
54) If ($error$ is PS) and ($\Delta error$ is PL) then (Δu is PV)
55) If ($error$ is PM) and ($\Delta error$ is PL) then (Δu is PV)

56) If ($error$ is PL) and ($\Delta error$ is PL) then (Δu is PV)
57) If ($error$ is NL) and ($\Delta error$ is PV) then (Δu is PS)
58) If ($error$ is NM) and ($\Delta error$ is PV) then (Δu is PM)
59) **If ($error$ is NS) and ($\Delta error$ is PV) then (Δu is PL)**
60) If ($error$ is ZR) and ($\Delta error$ is PV) then (Δu is PV)
61) If ($error$ is PS) and ($\Delta error$ is PV) then (Δu is PV)
62) If ($error$ is PM) and ($\Delta error$ is PV) then (Δu is PV)
63) If ($error$ is PL) and ($\Delta error$ is PV) then (Δu is PV)

Considering that we use a singleton fuzzifier, the rules above have been underlined when at least one of their two inputs is fully activated ($DoF_E = 1$ or $DoF_{\dot{E}} = 1$). The rule highlighted in yellow corresponds to rule 31, for which the degree of fulfillment of both inputs is equal to 1, and therefore the overall DoF of the rule is given by:

$$DoF = \min(1, 1) = 1.$$

Based on the above, for the inputs $E = 0.333334$ and $\dot{E} = 0.00$, the output of the rule base is illustrated graphically below. The values for the inputs E and $\dot{E}$ were found using the `Ergasia1_fis.fis`. And more specifically, from the peaks of `MF3` and `MF5` respectfully.

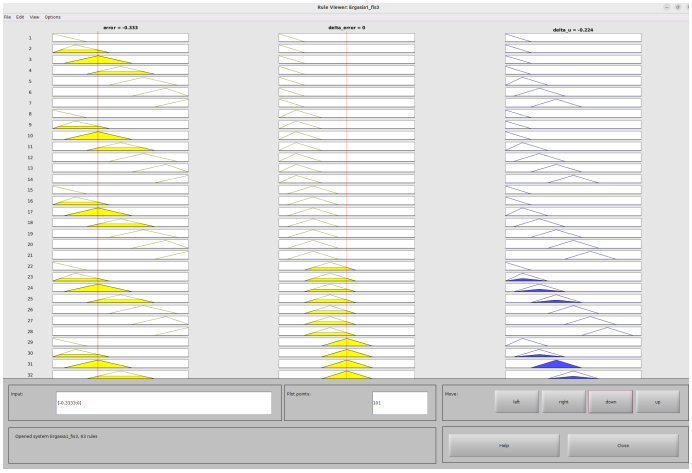

Fig. 17: Visualization of rule activation of the fuzzy rule base for input $(E, \dot{E}) = (0.3334, 0.00)$

As shown from the rules using the *Fuzzy Logic Designer*, only rule 31 is activated. This was expected, since although the ALSO method is implemented with the max operator for the inputs at the core ("peak") of the triangular MFs, for crisp input values only the rules corresponding to those values will be activated and no others, because the AND operator between the rules is implemented with the min operator.

Thus, for the inputs 0.33334 and 0.00, which are fuzzified as singleton MFs, the Degree of Fulfillment (DoF) will be 1 for the rules that have the conditions $E = NS$ and $\dot{E} = ZR$, respectively. Therefore, only rule 31 is activated with DoF = 1, while for all the other rules the DoF will be zero.

A more verbal explanation of the above is that the input $E = NS$ means the output signal is slightly bigger than the reference signal, while the input $\dot{E} = ZR$ means that the

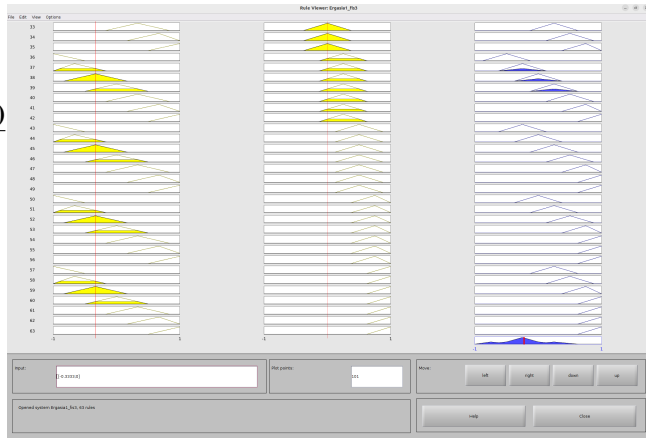

Fig. 18: Visualization of rule activation of the fuzzy rule base for input $(E, \dot{E}) = (0.3334, 0.00)$

error tends to remain almost constant. It is therefore logical to apply a small negative correction so that the output starts moving closer to the reference input. Hence, the control output is $\Delta U = NS$.

C. Interpretation of the Control Law of the FLC

Fuzzy Controller Output Surface

Below is the three-dimensional surface of the output ($\dot{U}$) of the fuzzy controller with respect to its inputs, E and $\dot{E}$.

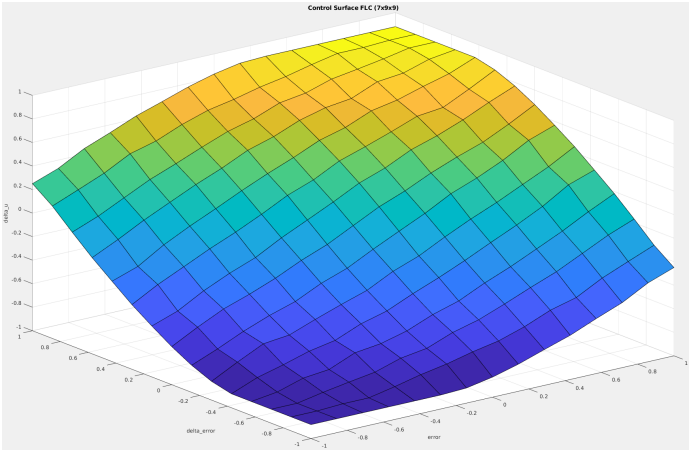

Fig. 19: Output Surface of the Fuzzy PI controller

This surface represents a three-dimensional visualization of the developed rule base. The development of the rules is based on the logic that $\dot{U} = E + \dot{E}$.

Since the two inputs of the rules are described by a different number of linguistic terms, it is reasonable that the output surface does not exhibit complete symmetry.

It can be observed that when E approaches its extreme values, meaning that the output significantly deviates from the reference signal, the controller output changes considerably (depending on the error direction) in order to approach the desired response. Conversely, when $\dot{E}$ approaches extreme values, meaning when the rate of change of the output signal is high and diverges from the reference signal, then the control

signal again takes large values to compensate for the rate of change of the output and to avoid overshoot or undershoot phenomena.

Furthermore, it becomes evident that when the error is small and its rate of change is also low, the variation of the control signal is correspondingly small. In these cases, the control signal remains close to or exactly at the reference signal.

*Detailed code can be found in the **createFIS_PI.m** and **Project1_fis.fis** files.*

SCENARIO 2

In this scenario, two different profiles of the reference signal are examined, as shown in Figures 20 and 21 respectfully.

- For the chosen parameters of the fuzzy controller, the closed-loop system response and control signal will be simulated and graphically presented for both reference signal scenarios.
- Based on the obtained responses, the ability of the FLC to track ramp-type input signals will be analyzed and discussed.

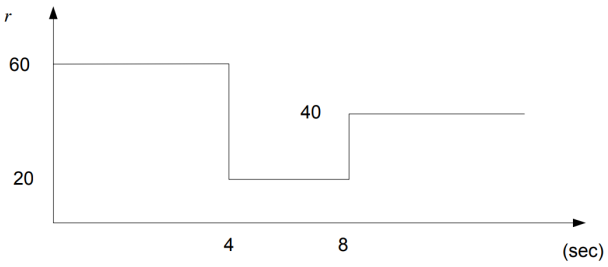

Fig. 20: Reference signal A

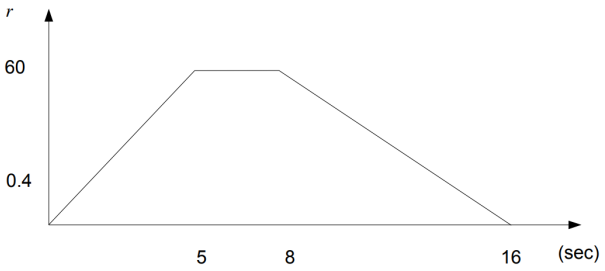

Fig. 21: Reference signal B

*Detailed code can be found in the **Project1_scenario2.m** file.*

A. Reference signal A (Fig. 20)

RESPONSE TO STEP-TYPE REFERENCE SIGNAL A

Figure 20 illustrates how the system reacts when the reference signal changes in steps with different amplitudes. The output (blue line) follows the reference (black dashed line)

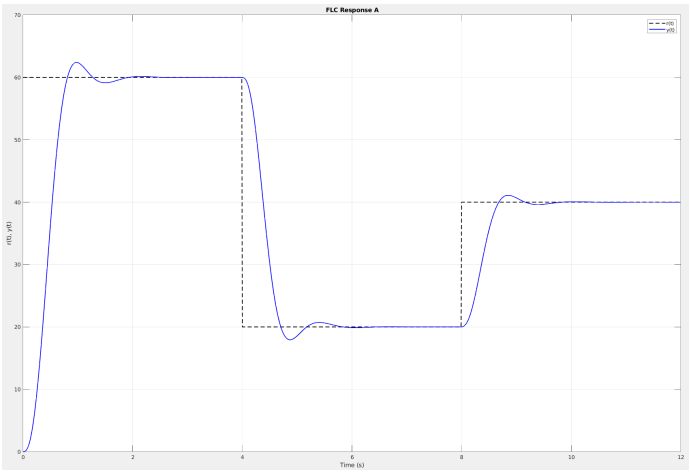

Fig. 22: FLC response to reference signal A

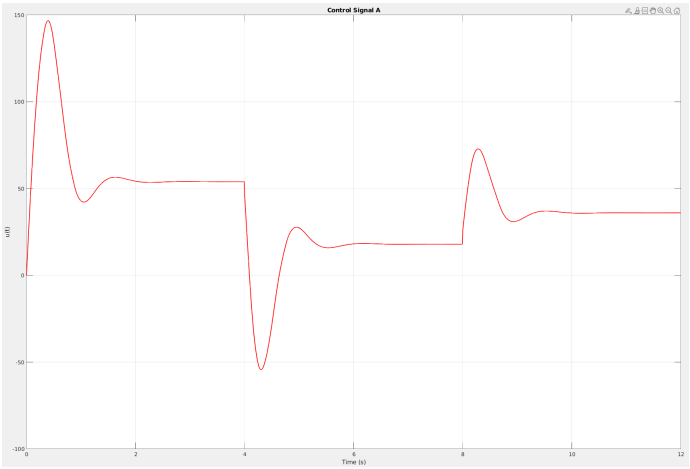

Fig. 23: Control signal u of the FLC to reference signal A

well, reaching the target after each change. The response is fairly quick, with only a small overshoot at the first step, where the output briefly goes above the reference before settling. Later step changes also cause short overshoots and undershoots, but the system stays stable and quickly settles to the new values with almost no steady-state error.

The controller also shows that it can handle sudden jumps in the reference signal quite well. Even though some oscillations appear right after the step changes, they fade away quickly, which means the system is being damped effectively. In general, the fuzzy PI controller manages to follow the step inputs with good accuracy and stability.

B. Reference signal B (Fig. 21)

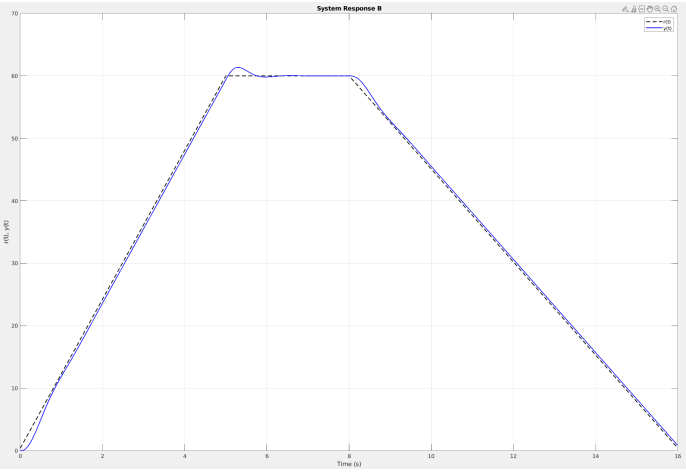

Fig. 24: FLC response to reference signal B

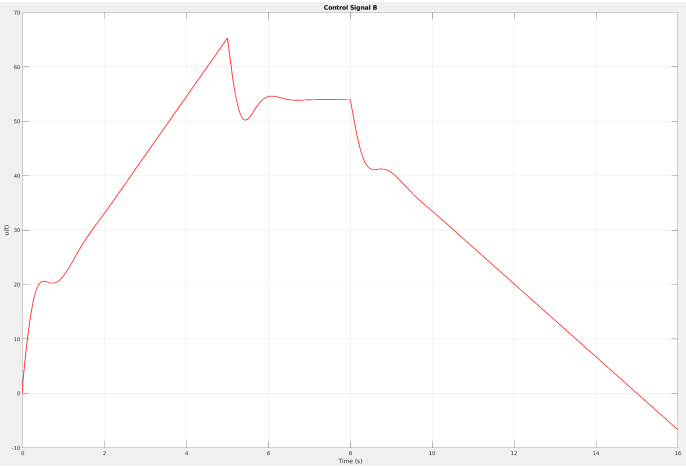

Fig. 25: Control signal u of the FLC to reference signal B

Response to Ramp-Type Reference Signal

The second case, shown in Figure 21, looks at how the system reacts to a ramp-type input. Unlike the sudden jumps in the step signal, the ramp changes more gradually, which is normally a tougher test for the controller since it has to follow a moving target. The results, however, show that the fuzzy PI controller actually handles the ramp signal even better than the step signal. During both the rising and falling parts, the output follows the reference very closely, with only a small delay visible when the slope changes quickly.

At the top of the ramp, there is a small overshoot before the system settles back onto the reference, but overall the tracking is smoother and more accurate than in the step case. This happens because fuzzy controllers, like PI controllers in general, are designed to reduce steady-state error and work best when the error changes in a gradual and predictable way. With a step input, the sudden jump produces a large and abrupt error that pushes the controller to react strongly, which can lead to overshoot or short oscillations. In contrast, with a ramp input, the error develops more slowly, which allows the fuzzy rules and the integral action to adjust the control effort more smoothly.

Overall, this means the fuzzy PI controller is very well-suited for tracking signals that evolve continuously over time. The ramp case highlights its strength in balancing quick reaction with stability, showing that it can achieve smoother and more precise tracking than with sudden step changes.

Discussion on Tracking Performance

Looking at Figures 22 and 24, it's clear that the fuzzy PI controller performs well in both situations. For step inputs, it shows that it can handle sudden changes in the reference with decent transient behavior, although some overshoot and short oscillations are present after each jump. For ramp inputs, however, the performance is noticeably better: the controller follows the reference more smoothly and with smaller errors, showing less overshoot and a more stable response overall.

This difference can be explained by how fuzzy controllers — and controllers in general — react to different kinds of errors. Step inputs create very large and sudden errors, which make the controller react strongly, often leading to overshoot or small oscillations before things settle. Ramp inputs, on the other hand, create errors that change more gradually, which gives the fuzzy logic rules and the PI action more time to adjust the control effort smoothly. As a result, the ramp case looks cleaner and shows the controller's ability to balance responsiveness with stability.

In summary, the fuzzy PI controller works reliably in both cases, but its strength really shows in the ramp scenario, where it achieves more accurate and smoother tracking. This makes it especially suitable for applications where the reference changes gradually over time, though it still handles sudden step changes well enough to remain stable and accurate.