Fuzzy Systems - Satellite (16)

Nikolaos Giakoumoglou 9043

25/08/2020

Abstract

Design of a conventional and a fuzzy PI controller to control the orientation angle of a satellite (assignment 16) with `MATLAB 2019a`. In the first part of the assignment we design a conventional linear PI controller for the given plant system. For more information see Automatic Control Systems. In the second part, we design a Fuzzy PI cntroller and regulate it with the results from the first part. The FZ-PI is tested for a number of given inputs and observe the ability to match the input signal.

Contents

1 Conventional PI Controller (script $conventional\_PI\_controller.m$**) 3**

2 Fuzzy PI Controller (script $satellite\_FIS.m$**) 6**

2.1 Preparation 6
2.2 Membership Functions 6
2.3 Rule Base 8
2.4 Custom Defuzzifier: COS (function $customdefuzz.m$) 8
2.5 Save the Model 10

3 Simulations 11

3.1 Scenario No.1 (simulink $control\_scenario\_1.xls$) 11
3.1.1 Design the controller and responses (a) 11
3.1.2 Operation of the Rule Base and conclusions (b) 14
3.1.3 Interpretation of the FLC control law (c) 15
3.2 Scenario No.2 (simulink $control\_scenario\_1.xls$) 16
3.2.1 Input 1 (pulse) 17
3.2.2 Input 2 (trapezoidal) 17

1 Conventional PI Controller (script *conventional_PI_controller.m*)

Our aim is to design a conventional linear PI controller which satisfies the following requirements:

- Overshoot M_P less than 10 %
- Rising time t_r less than 1.2 seconds

The linear PI controller can be modeled as

$$G_c(s) = K_P + \frac{K_I}{s} = \frac{K_P(s+c)}{s}, c = \frac{K_I}{K_P}$$

We arbirarily choose the zero denoted as c between -1 and -9 and closer to -1 e.g.

$$c = 2$$

The open loop function is

$$A(s) = G_c(s) \cdot G_P(s) = \frac{K_P(s+c)}{s} \cdot \frac{10}{(s+1)\cdot(s+9)} \Rightarrow$$

$$\Rightarrow A(s) = \frac{10 \cdot K_P \cdot (s+2)}{s \cdot (s+1) \cdot (s+9)}$$

We are ready to plot the root locus figure.

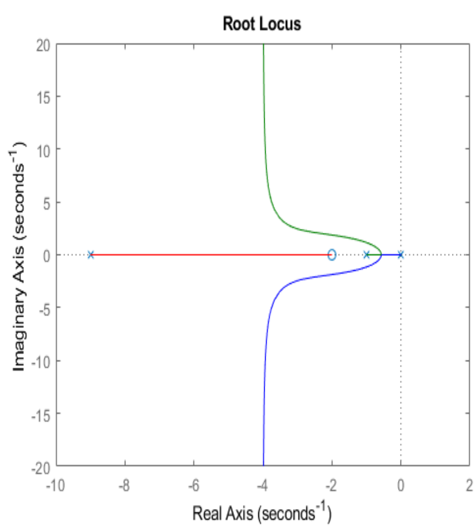

Figure 1: Root locus

We know that the overshoot is

$$M_p = e^{\frac{-\zeta \cdot \pi}{\sqrt{1-\zeta^2}}} \times 100\%$$

where ζ is the dumping ratio. Moreover we know that

$$\omega_n = \frac{1.8}{t_r}$$

where ω_n is the natural frequency. Solving the above equations we get

$$\begin{cases} \zeta = 0.5911 \\ \omega_n = 1.5 \end{cases}$$

and we can plot the feasible area in MATLAB in order to choose the gain K_P. After trial and error we set

$$K_P = 0.8$$

Indeed we can verify our choices after ploting the step response function where we can see that the rise time and the overshoot are acceptable

$$t_r = 1.0906 < 1.2\, sec$$
$$M_P = 9.3554 < 10\%$$

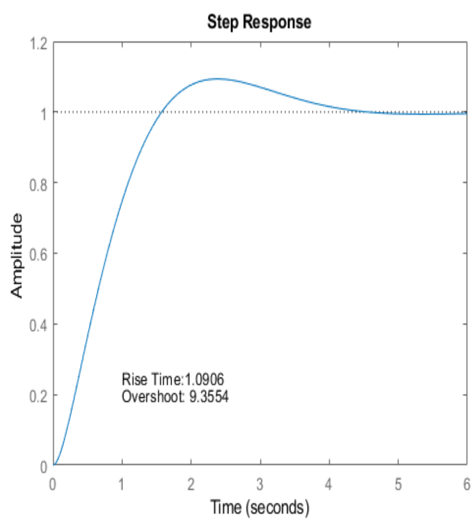

Figure 2: Step response

Summing up, we have

$$\begin{cases} K_P = 0.8 \\ K_I = 1.6 \end{cases}$$

2 Fuzzy PI Controller (script *satellite_FIS.m*)

2.1 Preparation

Taking into consideration the following:

- Fuzzifier: Singleton
- Defuzzifier: COS
- AND $\rightarrow$ min
- Fuzzy Rule $\rightarrow$ Larsen (product)
- ALSO $\rightarrow$ max

we are ready to design the FLC. The FIS is the following

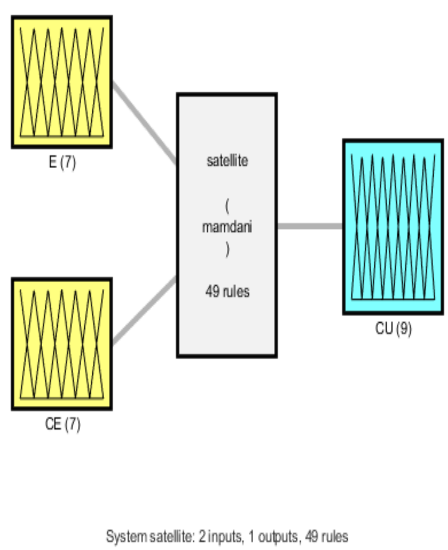

Figure 3: FIS

2.2 Membership Functions

The design of the membership functions includes 7 triangular functions for error and change of error and 9 for $\dot{U}$. Results are as following

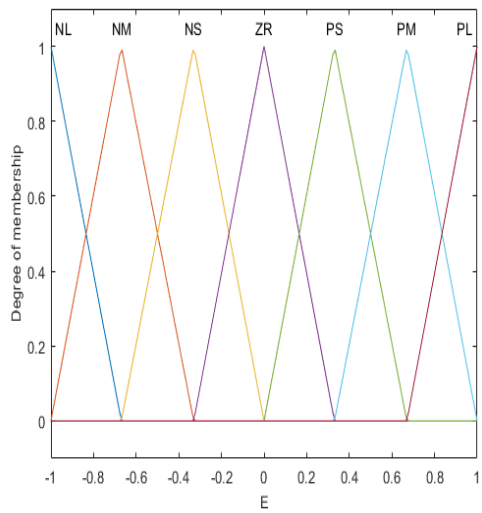

Figure 4: Membership function of error - E

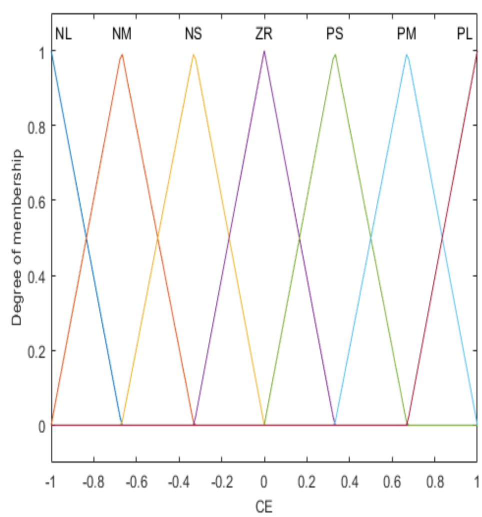

Figure 5: Membership function of change of error - CE

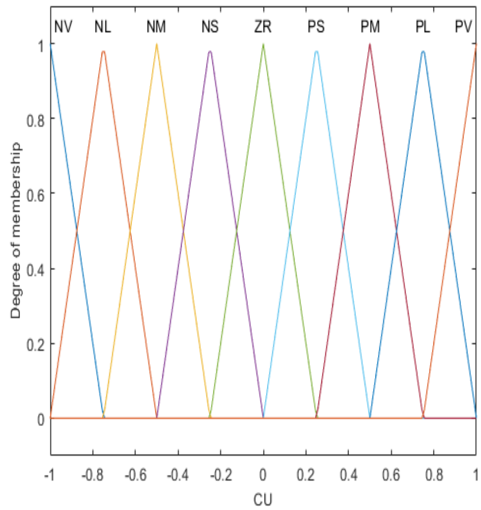

Figure 6: Membership function of $\dot{U}$ - CU

2.3 Rule Base

Taking into consideration a PI controller's Rule Base, we have

		Error						
		NL	**NM**	**NS**	**ZR**	**PS**	**PM**	**PL**
Change of Error	**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

Table 1: Fuzzy Rule Base

2.4 Custom Defuzzifier: COS (function $customdefuzz.m$)

In order to complete our FIS model, we have to define a fuction for the center of sums defuzzification method since *MATLAB* does not have a build in function. For that reason, we created a function *customdefuzz* where the inputs are the values x, y of the membership functions of the controller's output, and the output is

$$y^*_{COS} = \frac{\sum_{i=1}^{N}(y_i \cdot V_i)}{\sum_{i=1}^{N} V_i}$$

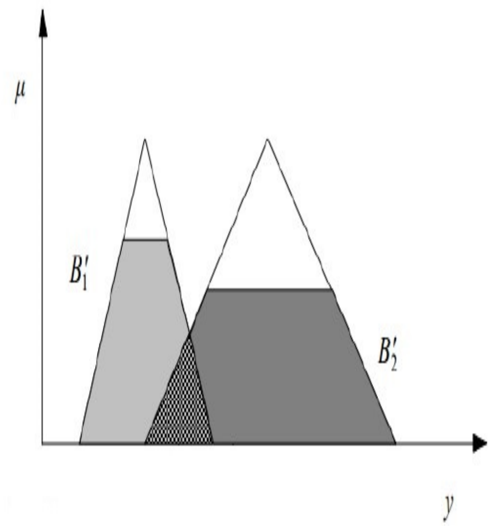

Figure 7: Center of Area - COA defuzzifier

In the script $customdefuzz.m$, first we find the peaks of ymf and their positions so as we can calculate the distance which the ymf has the same value as the peak. Having the height and the length of the bases we can calculate the area of the trapezoid: $area = \frac{x_1+x_2}{2}$, where $x_1$is the x-coordinate of the first point of the upper base and x_2 the x-coordinate of the second point of the upper base. In case the membership function is the far left or right (NL and PL respectively), we need to add an exception since *findpeaks* fails to locate the local maximal and add the result to the final sum. In the latter case, we check if the value is different than zero, calculate the integral between a line and a triangle and then calculate the COA by dividing the area of the integral to the total areal and applying the COS equation.

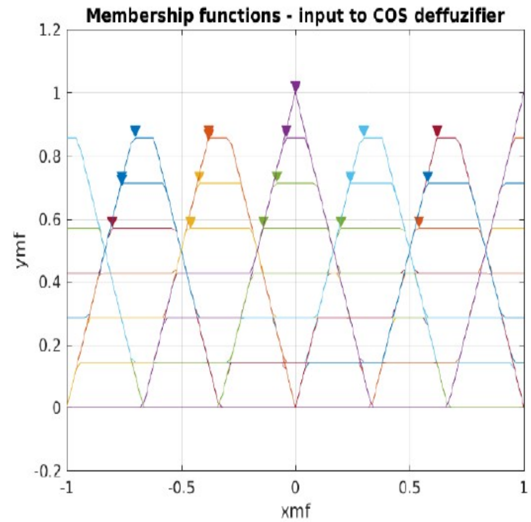

Figure 8: Custom defuzzifier input

2.5 Save the Model

The final step is to save our model to `satellite.fis`with the command

```
writefis(fis,'satellite.fis')
```

3 Simulations

3.1 Scenario No.1 (simulink $control\_scenario\_1.xls$)

First of all, we create the models in simulink. The control system and the FZ-PI controller are as following

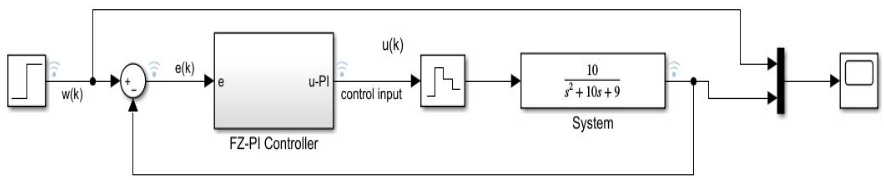

Figure 9: Control system

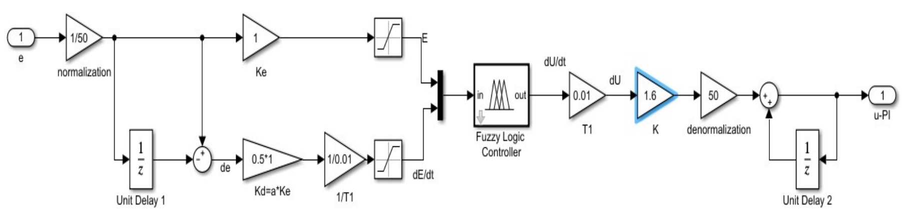

Figure 10: FZ-PI controller

We took into consideration the following:

- Sampling time $T = 0.01$ sec
- $r \in [0, 50]$

3.1.1 Design the controller and responses (a)

Our aim is the regulation of the PI controller to achieve

- Overshoot M_P less than 7 %
- Rising time t_r less than 0.6 seconds

In order to do that, we initialize

$$K_e = 1$$

$$a = T_i = \frac{K_P}{K_I} = 0.5$$

$$K = \frac{K_P}{\mathcal{F}\{a \cdot K_e\}} = \frac{K_P}{\mathcal{F}\{0.5 \cdot 1\}} = \frac{0.8}{0.5} = 1.6$$

The response to the maximum input $w(k) = 50$ is the following

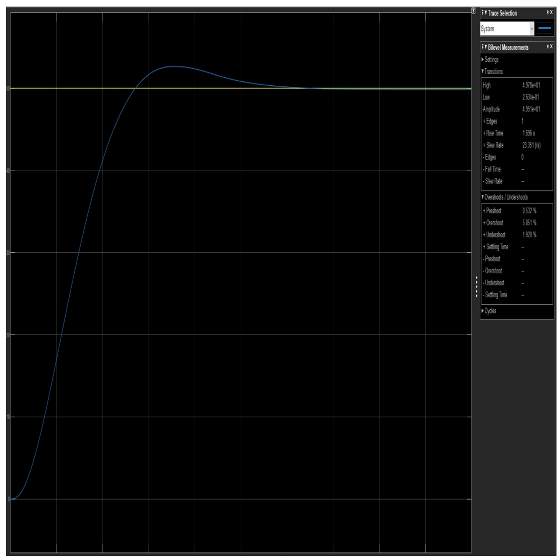

Figure 11: Initial regulation

As we observe, the overshoot is $M_P = 5.581\%$ and the rise time is $t_r = 1.696$ seconds which are not acceptable. As a result further regulation is needed. For that reason we increase the K_e and K while we decrease the a. By decreasing the a, the rise time is reduced significantly but the overshoot is inceased. That is the reason we increase the gains K_e, K. After trial and error, the requirements are fullfilled with the following values

$$K_e = 1.5$$

$$a = 0.25$$

$$K = 30$$

And the step response is

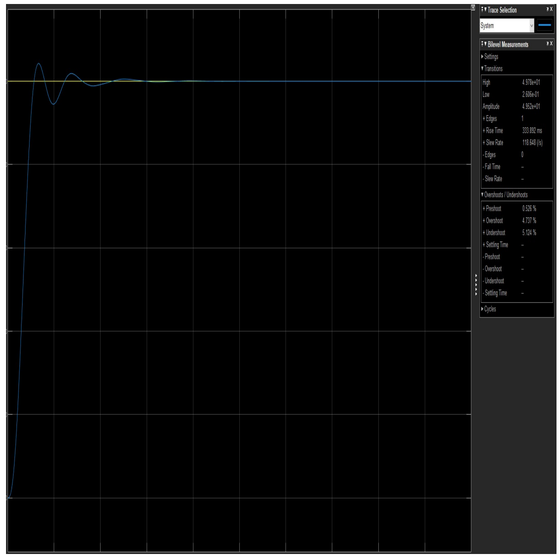

Figure 12: Final tuning

where the overshoot is $M_P = 5.124\%$ and the rise time is $t_r = 0.333$ seconds
The results are gathered in the following table

	K_e	K	a	K_p	K_I	Overshoot(%)	Rise Time
Initial Regulation FZ-PI	1	1.6	0.5	-	-	5.581	1.696
Final Tuning FZ-PI	1.5	30	0.25	-	-	5.124	0.333
Linear PI	-	-	-	0.8	1.6	9.341	1.087

Table 2: FZ-PI vs conventional PI controller

As we can see below, the FZ-PI behaves much better than the conventional PI controller

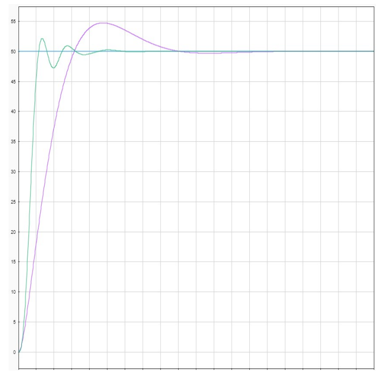

Figure 13: FZ PI vs PI controller for $w = 50$

3.1.2 Operation of the Rule Base and conclusions (b)

Lets assume that the error is PS e.g. 0.33 and the change of error is NM e.g. -0.67. In that case only 1 rule is active
IF E is PS and CE is NM THEN CU is NS
and thus the change of U is NS hence -0.75 after defuzzification.

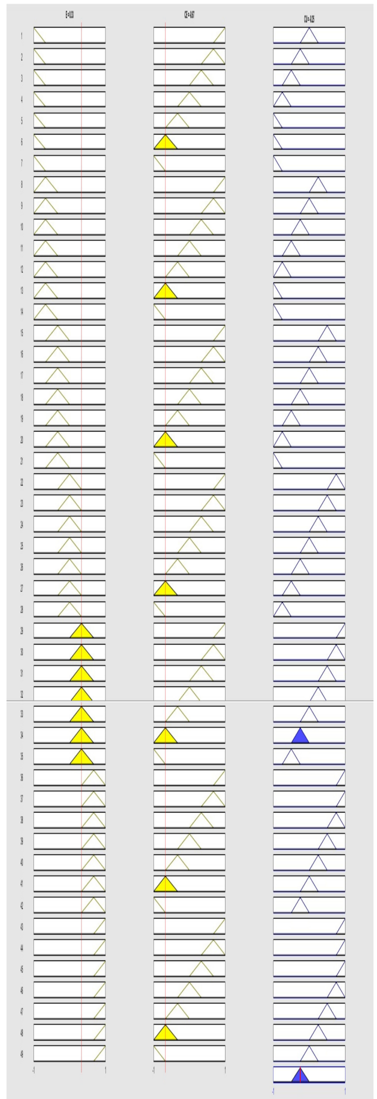

Figure 14: Operation of Rule Base

Script *satellite_FIS.m* ***MATLAB*** code:

```
ruleview(fis)
```

3.1.3 Interpretation of the FLC control law (c)

The 3D surface of the fuzzy controller is the following

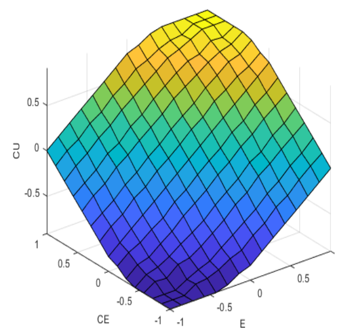

Figure 15: 3D surface of fuzzy controller

The controller aims to reduce the error to 0. If the error is negative and change of error is also negative, then the error tends to increase, hence the controller gives a negative output to fix the error. If the error is positive and the change of error is also positive, then the error tends to increase, hence the controller gives a positive output to fix the error. Lastly, if the error is auto-correcting itself or is zero, there is no need for additional output.

Script *satellite_FIS.m* ***MATLAB*** code:

```
figure;
gensurf(fis)
```

3.2 Scenario No.2 (simulink *control_scenario_1.xls*)

We have modeled this scenario in simulink.
In order to get both responses, we disconnect and re-connect the 2 inputs of the signal.

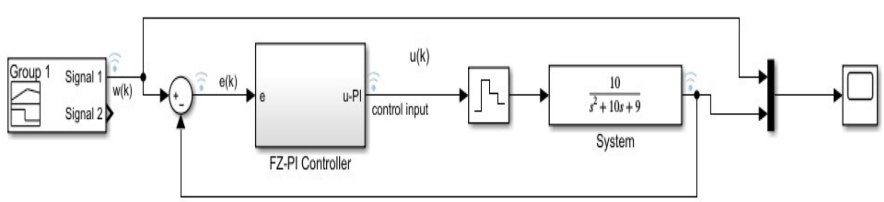

Figure 16: FZ-PI controller

3.2.1 Input 1 (pulse)

The responce of the fuzzy controller for the pulse as input is the following

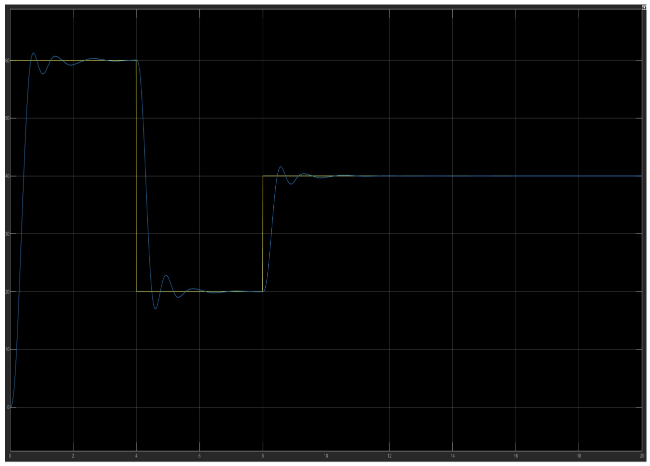

Figure 17: Pulse input

3.2.2 Input 2 (trapezoidal)

The responce of the fuzzy controller for the trapezoidal as input is the following

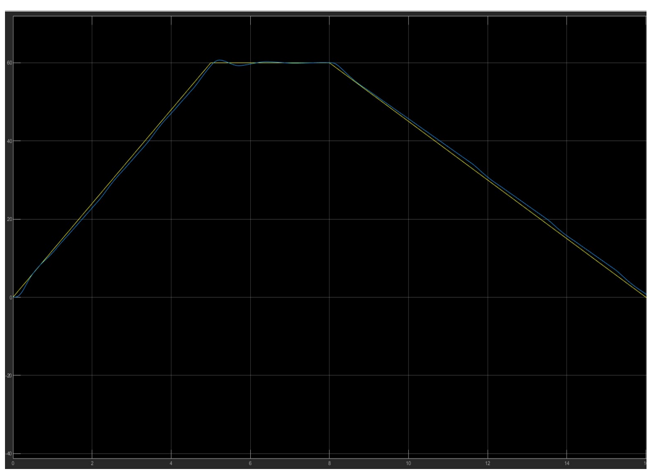

Figure 18: Trapezoidal input

We observe that the controller has better response to the pulse rather than the trapezoidal since there are no abrapt transitions in the first case.

List of Figures

1 Root locus 3
2 Step response 4
3 FIS 6
4 Membership function of error - E 7
5 Membership function of change of error - CE 7
6 Membership function of $\dot{U}$ - CU 8
7 Center of Area - COA defuzzifier 9
8 Custom defuzzifier input 10
9 Control system 11
10 FZ-PI controller 11
11 Initial regulation 12
12 Final tuning 13
13 FZ PI vs PI controller for $w = 50$ 14
14 Operation of Rule Base 15
15 3D surface of fuzzy controller 16
16 FZ-PI controller 16
17 Pulse input 17
18 Trapezoidal input 17

List of Tables

1 Fuzzy Rule Base 8
2 FZ-PI vs conventional PI controller 13