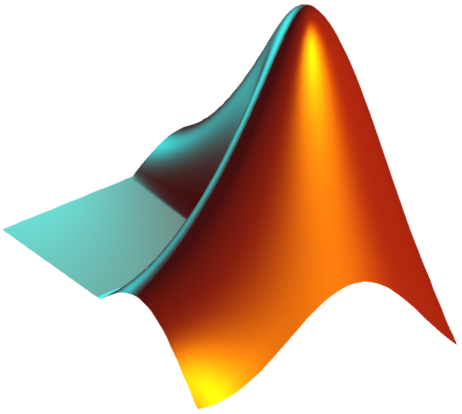
1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Informatics Engineering Department

Machine Learning
2020/2021 - 1º Semester

Assignment Nº3:

NEURONAL AND FUZZY COMPUTATION



Teacher:
António Dourado Pereira Correia

Students:

Renato Matos | 2015257602 | uc2015257602@student.uc.pt

Sérgio Machado | 2017265620 | uc2017265620@student.uc.pt

Index

Part A - Fuzzy Control

1. Introduction

This practical work has the purpose of being an introduction to fuzzy logic, fuzzy systems and to apply them to the modulation of a continuous process. This usage allows us to understand this concept and how they are useful to solve daily life issues.

Since these notions are sort of new to us we had to start by reviewing the concepts given in theoretical classes and other materials provided. We also watched videos provided by Matlab on how to design fuzzy controllers and searched for their documentation about fuzzy logic designer and simulink.

Later, we started by implementing the Mamdani and Sugeno fuzzy controllers with triangular and gaussian membership functions with a variable number of rules. Also, we used simulink to recreate the model of a continuous process. Then we ran the continuous process with the fuzzy controllers that we created and with the function that was given to us and evaluated their performance according to the integral squared error.

Further in this report, we show with more detail what we did, how it was done and the results we got. We also make discussion about the performance of the system according to each controller tested.

2. Membership Functions

Functions Selection

In this assignment, to design membership functions, we used two membership function types: triangular and gaussian. The triangular functions are easy to define and simple to understand since we only have to define three points. However, they might not give the best results since they are represented just by a triangle, which might not be very representative of real world situations. On the other side we have gaussian functions that are better because of their smoothness and concise notation. Also they never achieve the zero point.

We tested this with the same membership function for both Mamdani and Sugeno controllers.

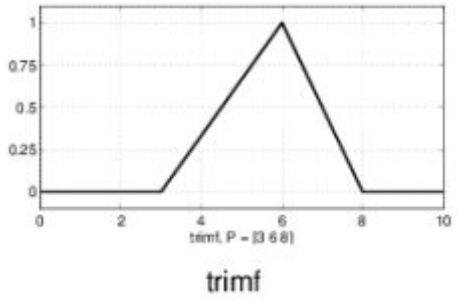


Figure 1. Triangular m.f.

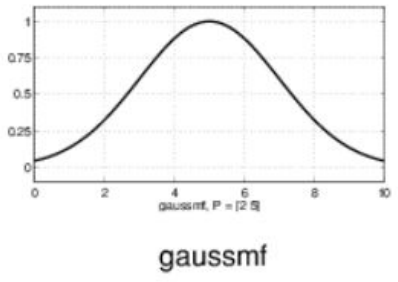


Figure 2. Gaussian m.f.

Functions Implementation

The implementation of a fuzzy controller was very simple with the fuzzy logic designer graphical interface. We just had to select the function type and number and the program will insert them automatically. Apart from that, we just had to define the intervals from -1 to 1 and define the rules. Below we have screenshots from the inputs with 3 and 5 fuzzy sets.

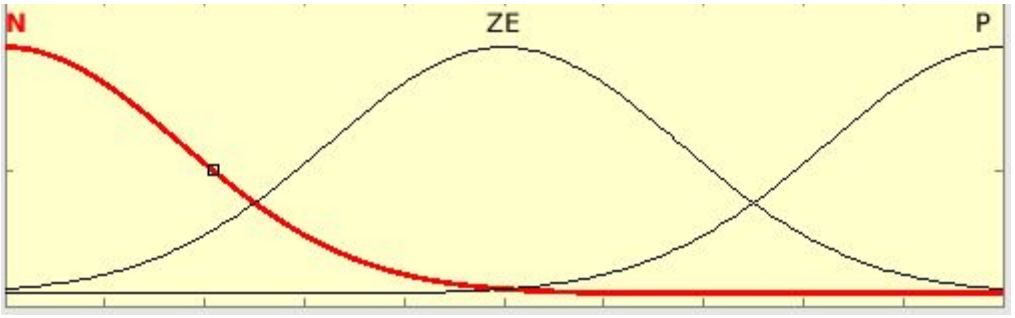


Figure 3. Input with 3 membership functions

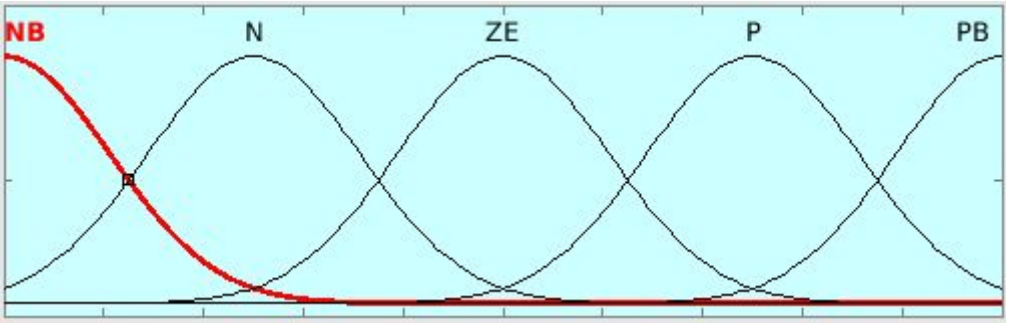


Figure 4. Output with 5 membership functions

3. Rules Set

In order to implement the fuzzy controllers we also need to identify the rules of the output that we would like to get from the combination of the two inputs. The following rules sets were tested for **both mamdani and sugeno** controllers.

We decided to implement the fuzzy controllers **with 9 and 25 rules** because we thought that it was enough to model the continuous process that will be shown further. Having 49 rules would be **too much** to represent a relatively simple continuous process and these presented here were enough to model the system.

The rules used were the ones presented in the theoretical classes. In the first table we have the rules for a controller with 2 inputs and 1 output as the three membership functions **contain 3 fuzzy sets**. In the second table we have the rule table that was proposed from previous students from this course in which we have 2 inputs with 3 fuzzy sets and 1 output **with 5 fuzzy sets**. Since the students that used it reported that it had good results we also tested it to have more tests that can be compared.

$e_k \mid \Delta e_k$	**N**	**ZE**	**P**
N	N	N	Z
ZE	N	ZE	P
P	ZE	P	P

Table 1

$e_k \mid \Delta e_k$	**N**	**ZE**	**P**
N	NB	N	Z
ZE	N	ZE	P
P	ZE	P	PP

Table 2

Also, we made fuzzy controllers in which both 2 input and output membership functions have 5 fuzzy sets as it was also presented in chapter 7.

$e_k \mid \Delta e_k$	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NS	ZE
NS	NB	NS	ZE	PS	PB
ZE	NB	NS	ZE	PS	PB
PS	NS	ZE	PS	PB	PB
PB	ZE	PS	PB	PB	PB

Table 3

Legend: **NB** = Negative Big, **NS** = Negative Small, **ZE** = Zero, **PS** = Positive Small, **PB** = Positive Big.

4. Fuzzy inference system

After building the fuzzy logic systems using mamdani and sugeno controllers with triangular or gaussian membership functions, with the presented rules, we can see the graphical representation of the output surface given both inputs. Here's two examples of the FIS surfaces that we built:

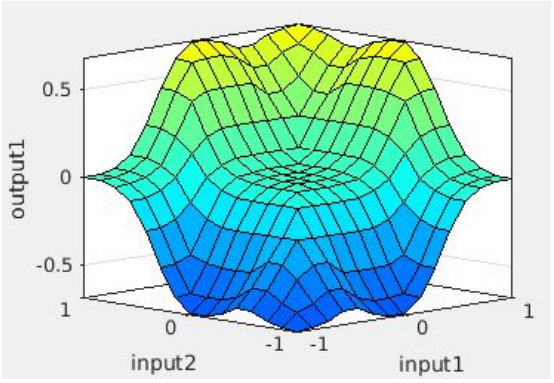


Figure 5. Mamdani - 9 rules surface (trimf)

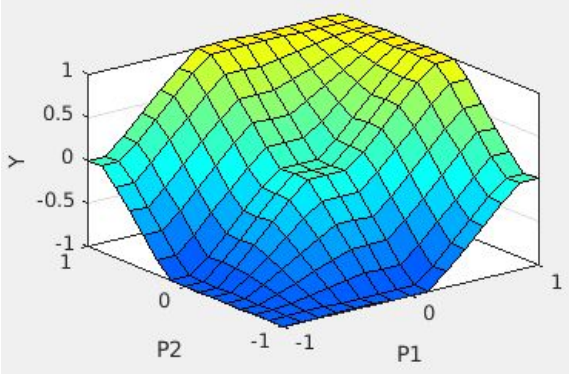


Figure 6. Mamdani FIS with 9 rules surface (trimf)

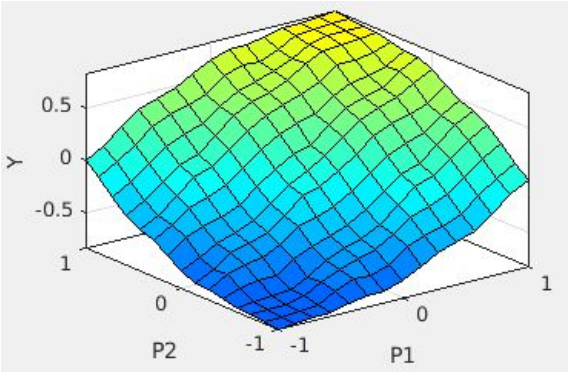


Figure 7. Mamdani - 25 rules surface (gauss)

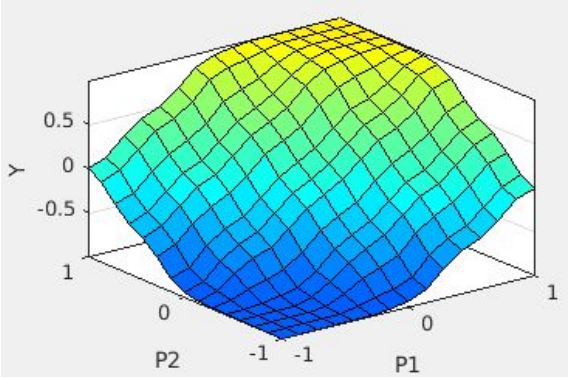


Figure 8. Sugeno - 25 rules surface (gauss)

5. Continuous Process Implementation

This is the transfer function that was proposed to us:

$$\frac{2s+3}{s^3+2s^2+2.5s+1.25}$$

Figure 9. Our transfer function

In order to test the impact of the characteristics of each membership function and of each controller, we built the model of a continuous process that was presented to us in the chapter 7 of the theoretical classes.

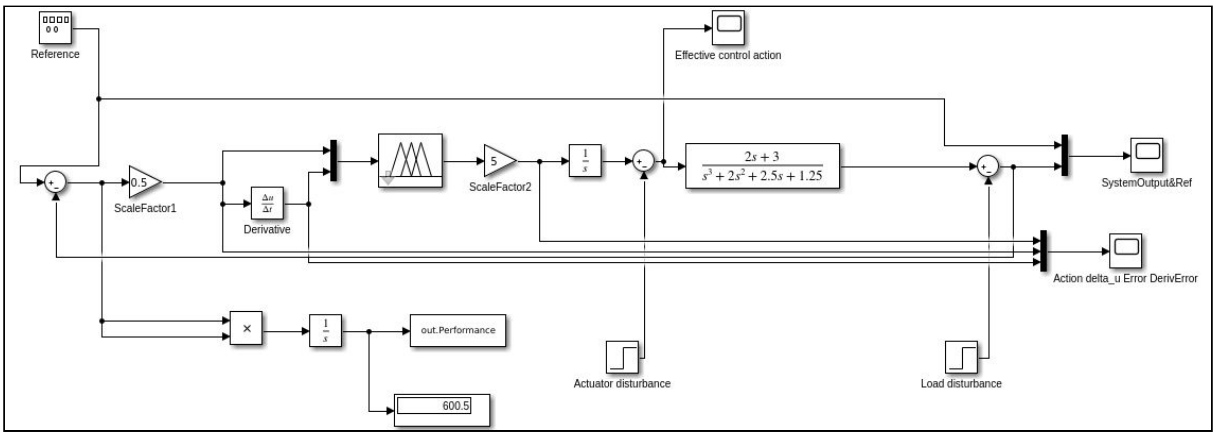


Figure 10. Model for a continuous process

The model starts in the signal generator that outputs a signal with a selected wave frequency of **0.05 rad/s** that could be sinusoidal, square or sawtooth. Then we have two gain blocks, representing the scale factor for both input and output. These scale factors must be changed for each test in order to achieve the lowest error. We can also see in the figure the fuzzy controller which has a **refresh rate of 2 s** (for faster graphication) and the transfer function. We found this model particularly good because it gives us graphic information of the system actions and an instant display of the integral squared error which evaluates the performance of the model.

6. Results and discussion

In this section we present our results with each fuzzy controller. This task was particularly hard and time consuming since we had to adjust both scale factors for each controller test. As so, we decided to include only the best results since it would be easier to compare and give a better idea of the system behaviour with each controller. It is relevant to say that all tests were made with the system with perturbations at 75 s and 150 s. For each test the simulation lasts for 250 seconds.

5.1 Square Signal

Membership Functions		Rules Number	Input factor	Output Factor	Integral Squared Error
Mamdani	Triangular	9	0.6	5.1	24.56
		9 (5 outputs)	0.3	5.1	29.22
		25	0.1	5.2	23.04
	Gauss	9	0.7	5.4	22.83
		9 (5 outputs)	0.3	6	22.89
		25	0.2	3.7	22.96
Sugeno	Triangular	9	0.5	1.9	24.20
		9 (5 outputs)	0.7	3	24.22
		25	0.4	1	26.83
	Gauss	9	0.7	1.5	22.54
		9 (5 outputs)	0.7	1.7	23.22
		25	0.7	1	24.89

The tests with the square signal were the ones in which the integral squared error turned out to have the biggest values from all the tests. This is because the squared signal has **big changes** when compared to sinusoidal or sawtooth signals.

Using the triangular membership function, Sugeno performed better with 9 rules, both 3 and 5 outputs and Mamdani was better with 25 rules. Using the gaussian membership function, Sugeno performed better with 9 rules, and Mamdani in the other two. But the **difference in values was very small**, and therefore for this signal type we cannot see much difference between the Mamdani and Sugeno controller type. Between the membership functions, we can conclude the **gaussian performed better than the triangular**. The **smallest error** was achieved with a Sugeno controller with gaussian membership function and 9 rules. The smallest error for the Mamdani controller was obtained with the same variables and a similar result.

From the smallest errors we concluded that, for this signal, there wasn't **any advantage** of having a 9 rule fuzzy controller with an **output with five fuzzy sets**, as the best results were obtained with an output of only three fuzzy sets.

5.2 Sinusoidal Signal

Membership Functions		Rules Number	Input factor	Output Factor	Integral Squared Error
Mamdani	Triangular	9	0.8	4.8	11.07
		9 (otp. with 5 f.)	0.7	4.4	10.94
		25	0.1	5.2	4.35
	Gauss	9	0.9	5.2	6.52
		9 (otp. with 5 f.)	0.6	5	3.83
		25	0.2	5.1	4.33
Sugeno	Triangular	9	1	1	8.81
		9 (otp. with 5 f.)	1.3	0.9	8.04
		25	0.5	1	4.49
	Gauss	9	0.7	1.8	3.87
		9 (otp. with 5 f.)	1	3.1	3.90
		25	0.6	1	4.54

With a sinusoidal signal we had the smallest integral squared errors when compared to the other signal types. This might be justified with the fact that this signal doesn't have such abrupt changes when compared to the others.

In the test with this type of signal, the **Sugeno controller was the one that generally had the best values,** with the Mamdani performing very slightly better using the triangular membership function with 25 rules and using the gaussian with 9 rules and 5 outputs. As can be seen, the **gaussian was the membership function that always obtained the better results**, with the best results in both controllers belonging to this function.

The **best value** was obtained with the Mamdani controller using 9 rules and 5 outputs. With little difference compared with this result, the combination that performed the best with the Sugeno controller was 9 rules with 3 outputs.

In this case, the suggestion of having 9 outputs with 5 rules was helpful, considering that generally had better values than using 9 rules with only 3 outputs and especially because the best result was obtained using it.

5.3 Sawtooth Signal

Membership Functions		Rules Number	Input factor	Output Factor	Integral Squared Error
Mamdani	Triangular	9	0.8	4.9	15.93
		9 (otp. with 5 f.)	0.6	4.1	19.11
		25	0.1	5.7	11.71
	Gauss	9	0.9	5.2	11.42
		9 (otp. with 5 f.)	0.5	5.7	11.18
		25	0.2	4	11.21
Sugeno	Triangular	9	1	1.5	15.96
		9 (otp. with 5 f.)	0.9	1.3	17.05
		25	0.5	1	11.97
	Gauss	9	0.7	1.4	10.68
		9 (otp. with 5 f.)	0.6	2.5	10.28
		25	0.6	1	11.67

For the sawtooth signal, the performance was better than the squared signal but worse than sinusoidal signal.

Between the controllers the results don't differ too much, with the biggest difference recorded using the triangular membership function with 9 rules and 5 outputs. Regarding the difference between membership functions, **the gaussian one performed always better**. Excluding the 25 rules, the difference between triangular and gaussian was considerable.

The **best result** was achieved with the Sugeno controller type, gaussian membership functions and 9 rules with 5 outputs. With the Mamdani controller type this was also the combination with the best result.

Again, this was a case where the **output with 5 fuzzy sets and 9 rules had better results** than the output with just 3 fuzzy sets.

7. Graphical representation of our best results

- **Mamdani controller + gaussian functions (9 rules): 22.83 s.e.**

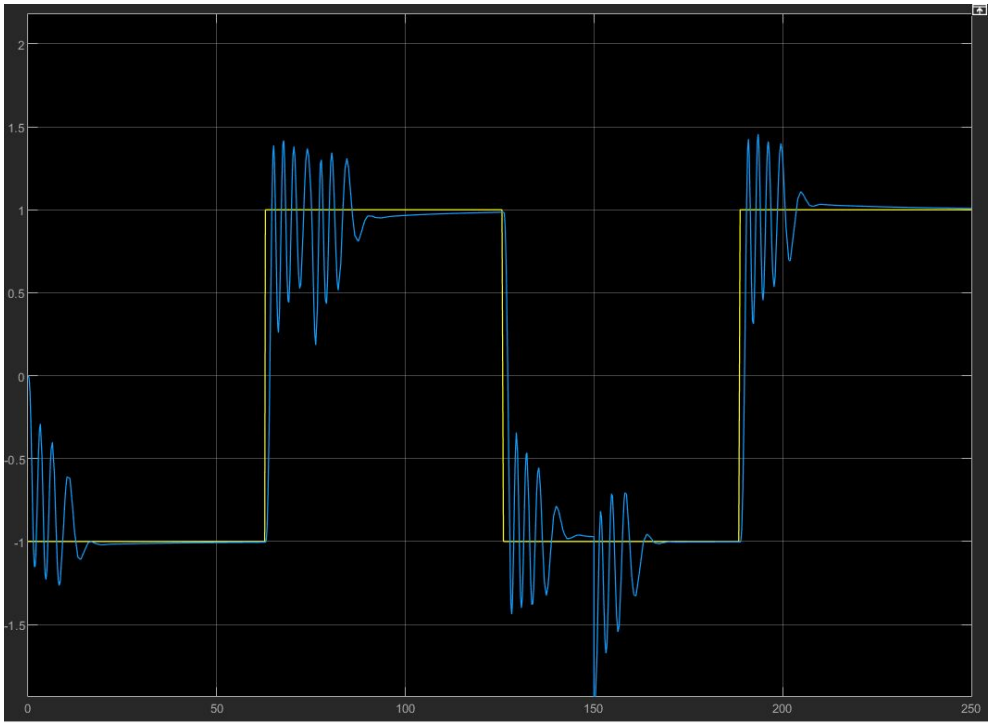


Figure 11. Process model output

- **Sugeno controller + gaussian functions (9 rules): 22.54 s.e.**

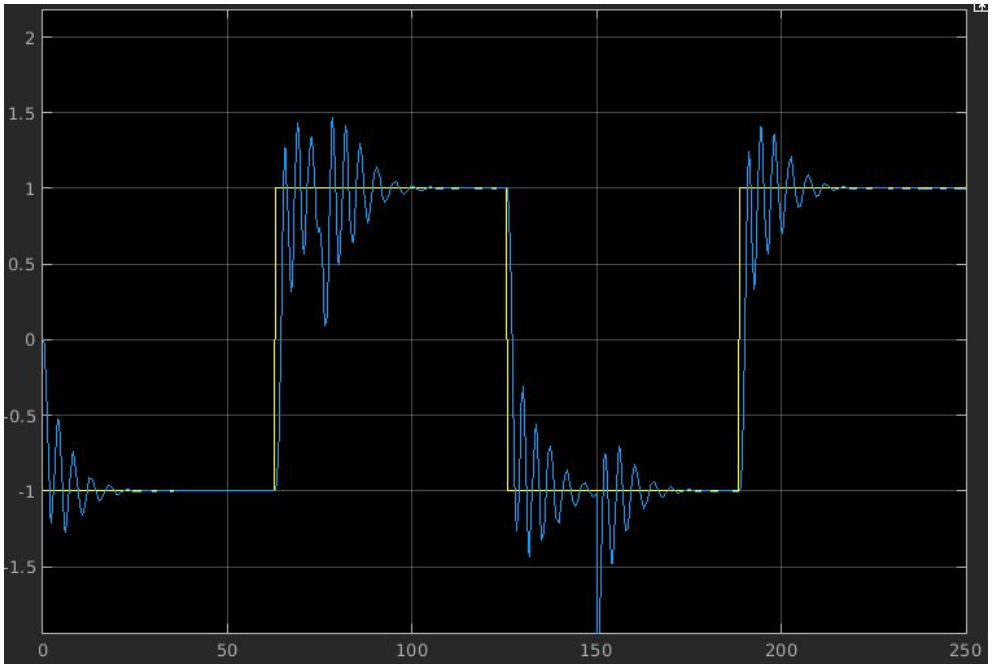


Figure 12. Process model output

- **Mamdani controller + gaussian functions (9 rules - output with 5 fuzzy sets): 3.83 s.e.**

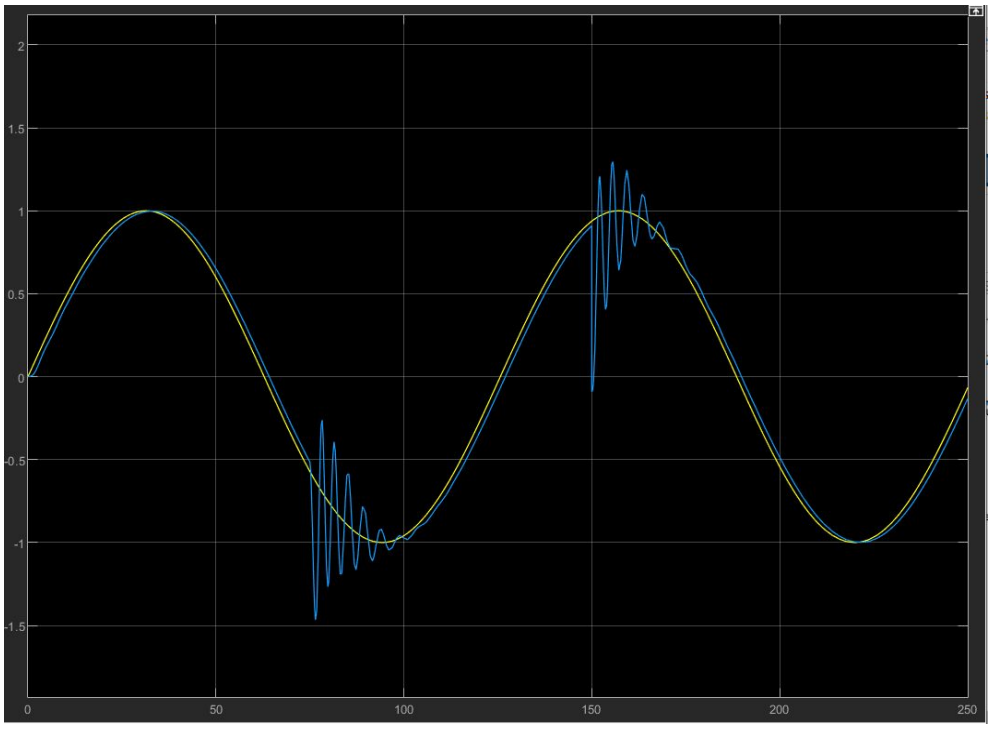


Figure 13. Process model output

- **Sugeno controller + gaussian functions (9 rules): 3.87 s.e.**

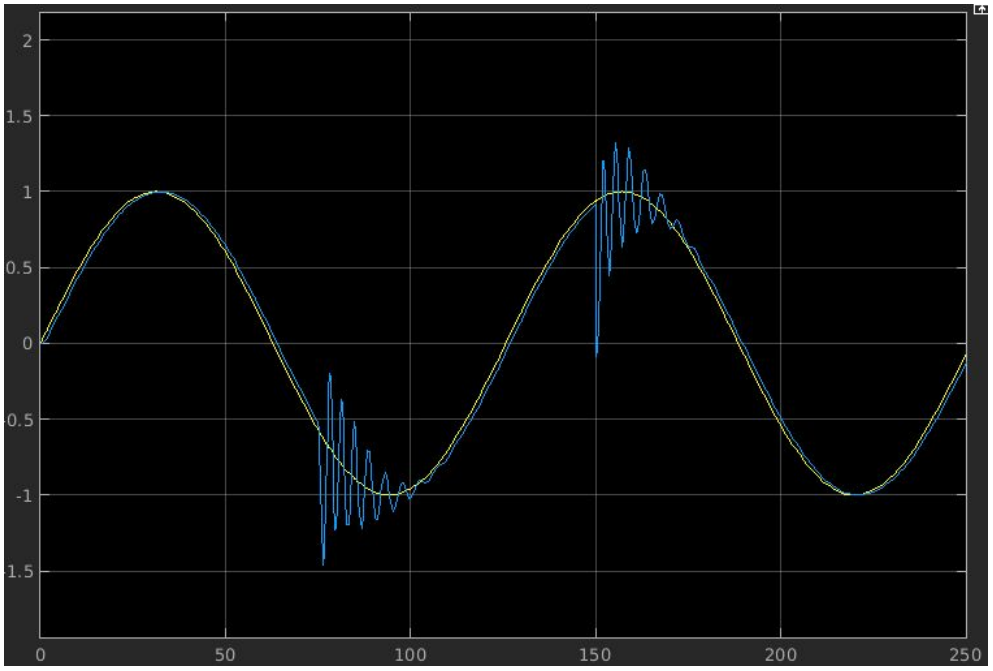


Figure 14. Process model output

- **Mandani controller + gaussian functions (9 rules-output with 5 fuzzy sets): 11.18 s.e.**

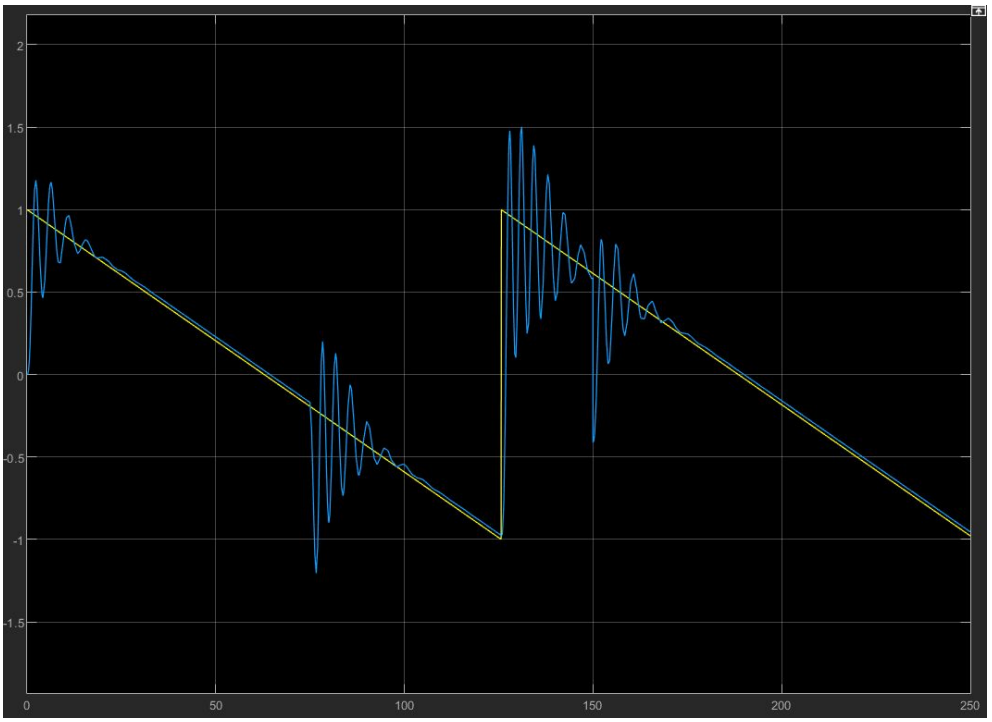


Figure 15. Process model output

- **Sugeno controller + gaussian functions (9 rules-output with 5 fuzzy sets): 10.28 s.e.**

Figure 16. Process model output

8. Effective control action

As important as the system response/output to a given signal there is also the effective control action of the system in order to "follow the signal". Here we have represented the control action of two of our best fuzzy controllers.

- **Sinusoidal signal: Sugeno controller + gaussian functions (9 rules)**

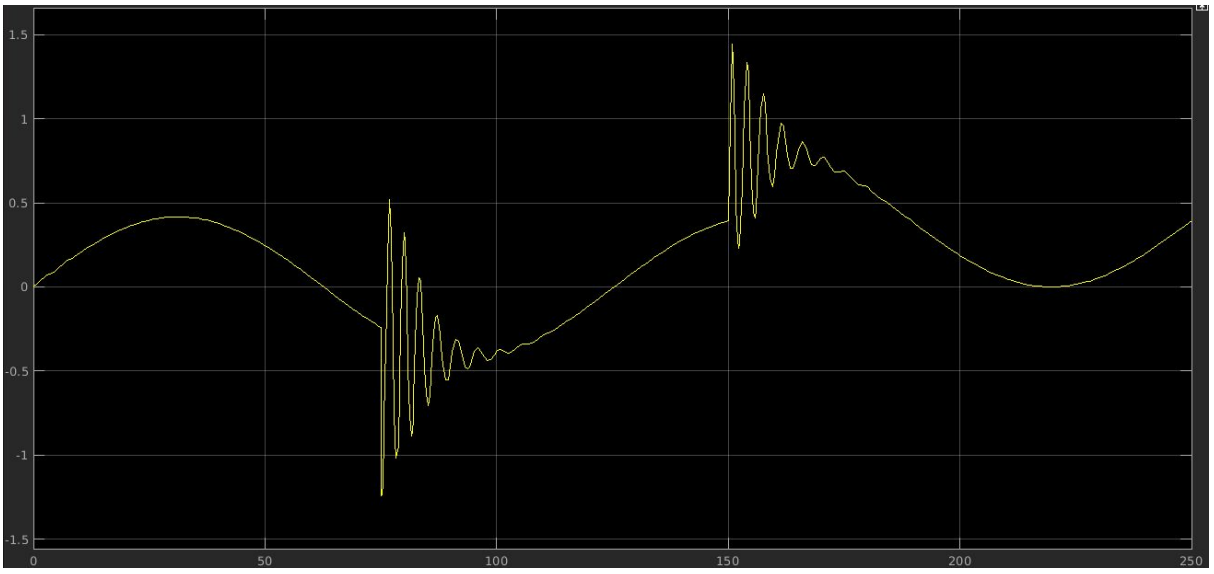


Figure 17. System effective control action

- **Sawtooth signal: Sugeno controller + gaussian functions (9 rules-output with 5 fuzzy sets)**

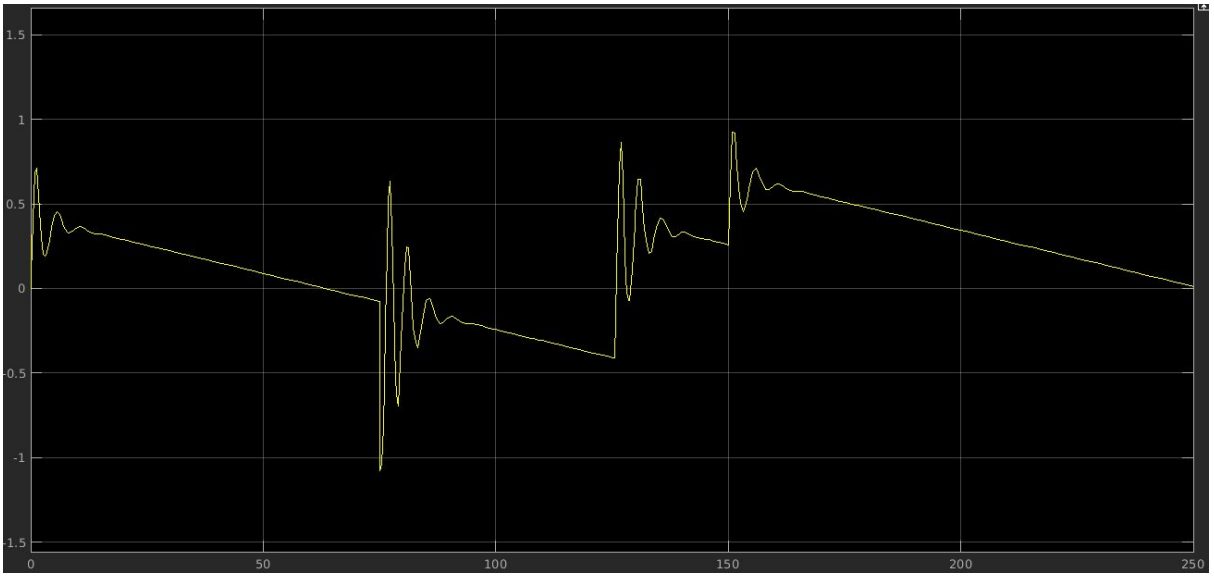


Figure 18. System effective control action

9. Conclusions

Looking at the tables and comparing the different signals we can say that, **generally, the modulations with less integral squared error were obtained using the sinusoidal signal**, then the sawtooth and at last the squared one. That was expected since the sinusoidal signal, compared to the square and sawtooth signals, has smoother signal changes. The other two signals have abrupt changes which might result in bigger error values, as verified in test results. The sawtooth having better results than the squared maybe can also be derived from the abrupt changes, because as we can observe in the graphs, the sawtooth signal has less vertices compared to the squared one. The results in some cases might be quite big because the system has perturbations which directly result in bigger errors.

The results between each controller didn't differ as much as expected. With the square signal, the difference wasn't much significant. Using the sinusoidal signal Sugeno generally performed better. For the sawtooth signal, again, they didn't differ too much. Once more, although the values were very similar, the best results were obtained using Sugeno for two of the three signals. So, we can conclude that Sugeno has a little advantage.

The gaussian membership function was recurrently the best function to model the systems. This can be concluded as in the tables, with exception of one time, the results using the gaussian were always better than the triangular (and in that exception the difference was small). Reenforcing this, the best results, from which later we showed the graphs, were always from using the gaussian function.

Regarding the number of rules, **using 9 rules was constantly the best option**, with all the best results using this quantity of rules. The rule set proposed by previous students of this course with **9 rules and 5 mf's in the output** really achieved **better results** compared to the one with 3 mf's in the output **in some cases**. Considering the best results, later used to produce the graphs, half of them were obtained using this ruleset (sinusoidal with Sugeno and sawtooth with both Mamdani and Sugeno). However, with the 25 rule fuzzy system we didn't notice much difference and this ruleset ended not being significant.

We came to the conclusion that building the membership function for the input and outputs it's quite easy, however having to modify the scale factor for each test makes the job much harder and we might never achieve the optimal values (the minimum possible error).

10. References

[1] A. Dourado, ML Chapter 6: Fuzzy Logic
[2] A. Dourado, ML Chapter 7: Fuzzy Systems
[3] Matlab documentation: Fuzzy Logic User's Guide
[4] Matlab documentation: Simulink User's Guide

Part B - Neuro-fuzzy systems for modeling dynamic processes

1. Introduction

This practical work part intends to give a deeper knowledge in fuzzy systems construction and application in model simulation.

We started by calculating a discrete transfer function for the actual transfer function that was given to us. Then we used it in a simulation to create a sufficient large dataset to train and create fuzzy inference systems. First we create these FIS by using three clustering methods which obviously have different properties and results. Then we train those fuzzy systems with our dataset so it is able to give a precise response to the system input with a minimal error. We used the backpropagation and the hybrid method. We present and discuss all the results that we achieve and then we select the best to test in a dynamic process simulation and the results were really impressive.

In this report is a detailed step-by-step description of what we did, how and the results we got. The report structure is in the same order as the described work previously.

2. Dataset generation

In order to obtain a good dataset that is able to train and design a set of rules in order to get a fuzzy system we had to start by finding a discrete function. Then the data might be used to learn the fuzzy systems by determining the membership function of the antecedents and of the consequent terms. As so, we need to find the discrete function in order to produce a temporal series in which there are a number of elements referred to at the same time instants, as referred in the assignment statement.

This was the function we had:

$$\frac{2s+3}{s^3+2s^2+2.5s+1.25}$$

After converting to discrete time assuming a zero order hold on the inputs:

$$\frac{0.0147z^2+0.0022z-0.0120}{1.0000z^3-2.7480z^2+2.5323z-0.7823}$$

After having the discrete function, this is the model that will give us the data that we will use to find the fuzzy systems.

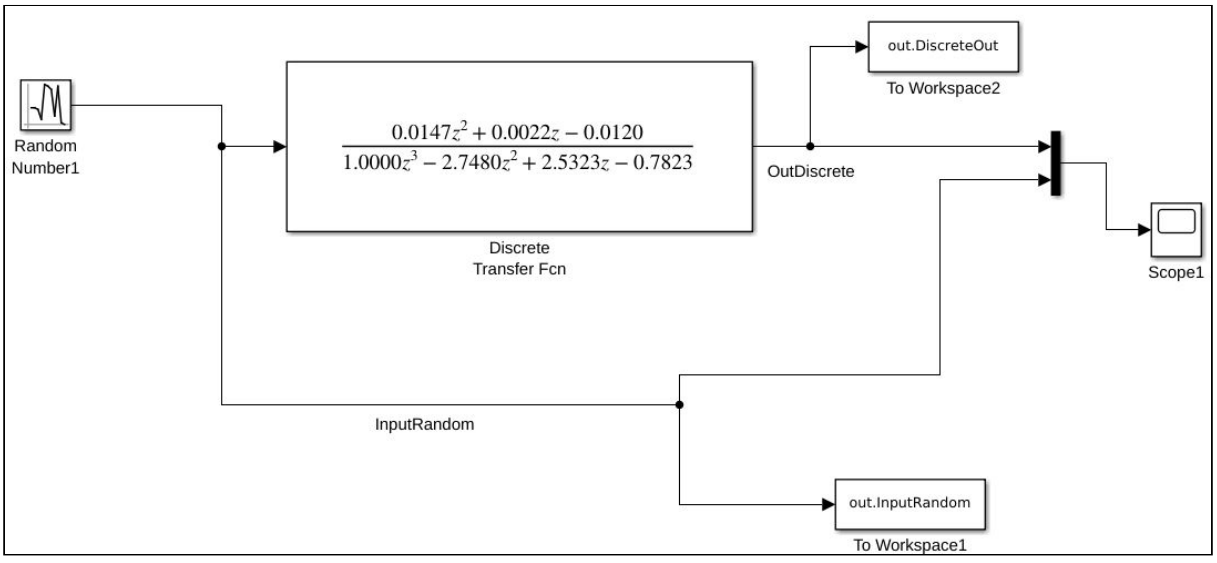


Figure 19. Model used to generate the dataset

3. Building the matrix

After running the simulation we got the following output signal:

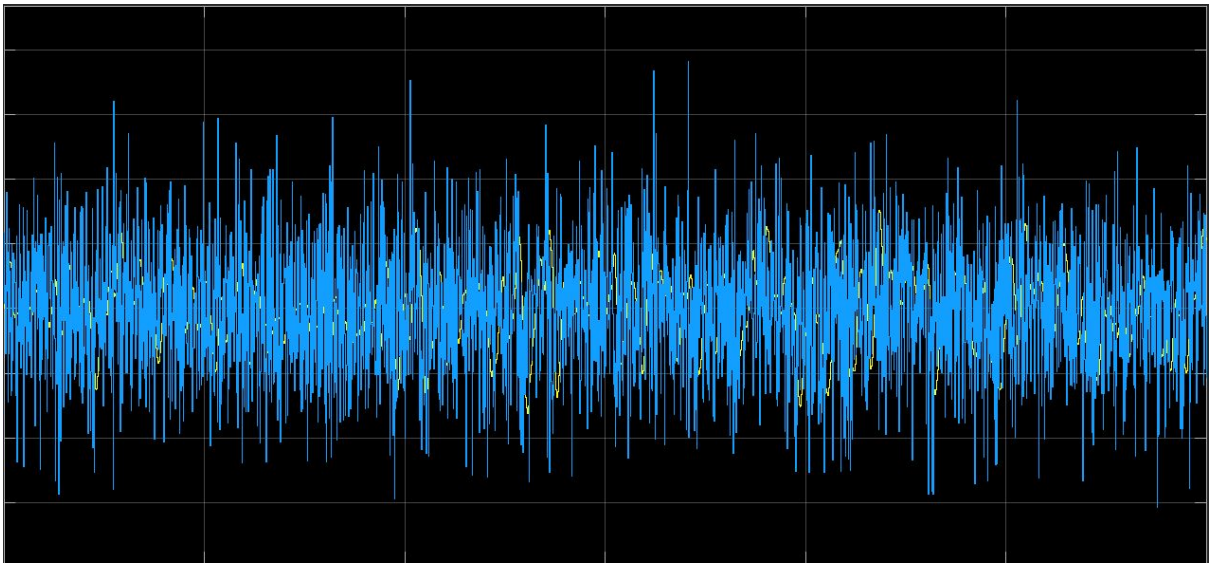


Figure 20. Figure 13 model output

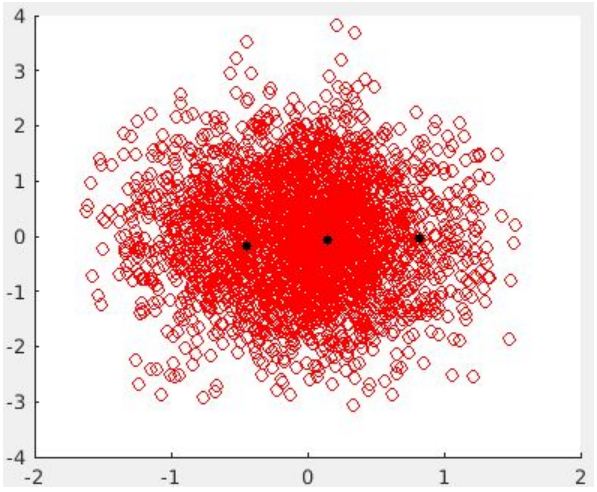
Also from the input and output time series that we got from the simulation we built the matrix following the instructions presented in the statement. The matrix has 6 columns for the antecedents and 1 for the consequents. This was the code that we used to built it:

```
matrix(:,1) = discreteOut(3 : end-1); %y(k-1)
matrix(:,2) = discreteOut(2 : end-2); %y(k-2)
matrix(:,3) = discreteOut(1: end-3);  %y(k-3)
matrix(:,4) = inputRandom(3 : end-1); %u(k-1)
matrix(:,5) = inputRandom(2 : end-2); %u(k-2)
matrix(:,6) = inputRandom(1: end-3);  %u(k-3)
matrix(:,7) = discreteOut(4:end);   %Output shifted
```

Figure 21. Code used to create the dataset matrix

4. Find clusters with a GUI

Using the command *findcluster* we explored the clustering techniques proposed: the subtractive and fuzzy c-means methods having the first column on X axis and the fourth column on Y axis.



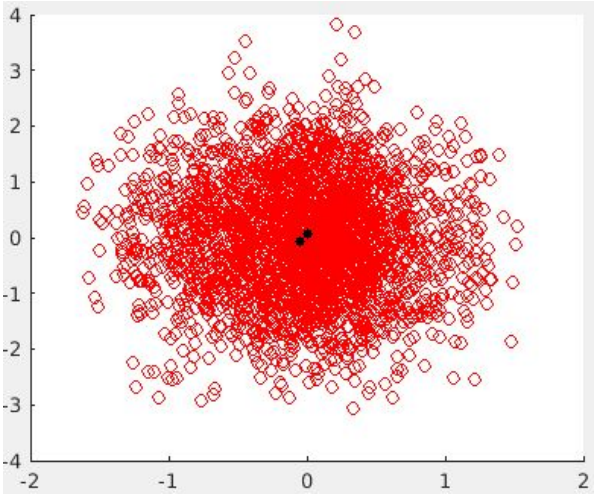


Figure 22. Clustering with subtractive method (default) **Figure 23.** Clustering with fuzzy c-means (default)

However this was just an exploratory test to have an idea of the difference between the methods. The systems that will be tested and evaluated in the simulations with the matrix data were obtained using commands. We will talk about how these different methods work in the following section.

5. Training the fuzzy systems

First of all, we started by divide the matrix data in two matrices: one for training with 70% of the data and the other for testing with the 30% left, as was proposed in the assignment.

```
div = floor(0.7 * R);
trainData = matrix(1:div,:);
testData = matrix(div+1:end,:);
```

Then we obtained the fuzzy inference systems by using the function *genfis*. We did this for three clustering methods: (1) subtractive clustering, (2) fuzzy c-means clustering (fcm) and (3) grid partition clustering. The goal is to use different clustering techniques on a given data in order to produce the systems behaviour representation.

5.1. Subtractive clustering

When we don't know how many clusters there are, subtractive clustering is usually faster and one-pass algorithm provides a quick estimation of the number of clusters existent in one dataset and their cluster centers [3].

The cluster estimates are obtained with the function *subclust* which defines the clusters centers based on the density of neighbouring data points. Then the points within the clustering range are removed and the process is repeated until there are no points left.

This is the plot of the fuzzy system that we got from the subtractive clustering:

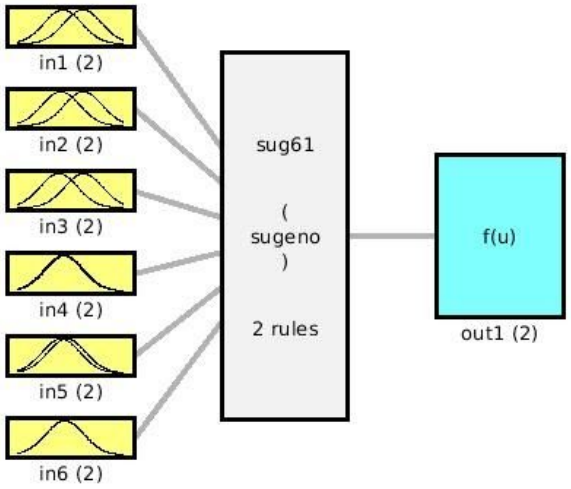


Figure 24. Subtractive clustering Fis plot

5.2. Fuzzy c-means clustering

This technique finds out where each data point belongs to a cluster to some degree that is specified by a membership grade. The function starts with an initial guess for the cluster centers and every data point has a membership grade for each cluster. Then, the algorithm iteratively moves the location of the clusters center with the intention of minimizing the objective function that represents the distance from any given data point to a cluster center [3]. This is fuzzy system that we got:

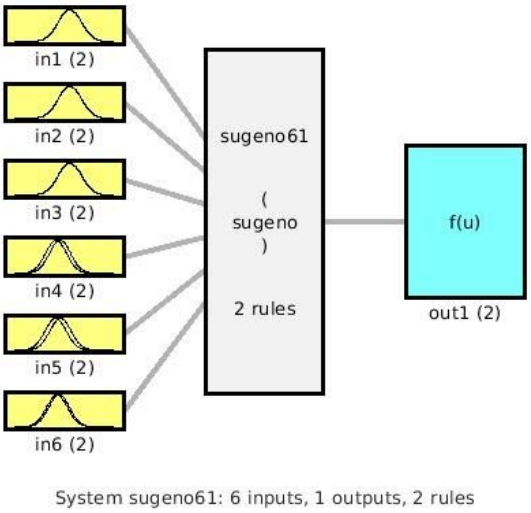


Figure 25. Fuzzy c-means clustering Fis plot

5.1. Grid Partition clustering

Generates the input membership functions by uniformly partitioning the input variable ranges, and then creates a single-output Sugeno fuzzy inference system. The fuzzy rule base contains one rule for each input membership function combination [3].

This is the plot of the fuzzy system that we got from the grid partition clustering:

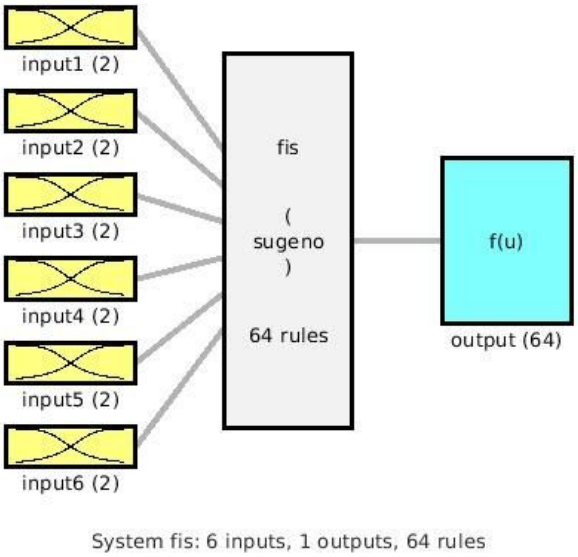


Figure 26. Grid partition clustering Fis plot

6. Optimizing the fuzzy inference systems

Now that we generated the fuzzy inference system successfully with the help of the three clustering techniques the next step was to optimize (train) the FIS with the training data previously selected from the built matrix. We did this using ANFIS which uses a back-propagation or a combination of algorithms. The training process is done in a way that the system models our input and output data. Further we will explain how each optimization method works and what results we got.

6.1. Backpropagation Method

The backpropagation model calculates the gradient of the loss function with the respect to the network weights for a single input-output example. The "backwards" refers to the back that the calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last [5].

6.2. Hybrid Method

The hybrid method includes the backpropagation method that we spoke about previously associated with the input membership function and least squares estimation for the values associated with the output membership functions [3].

7. Optimization results and discussion

After training the fuzzy inference systems we used the function *evalfis* to get an output from the testing dataset input. Then we calculate the mean square error by adding the square of the differences between the output from the FIS and the actual output. Below we can see the code that does that just for one example.

```
%Backpropagation GridPartition
bGridRes.output = evalfis(bGrid, input);
bGridRes.mse = sum((output-bGridRes.output).^2)/L;
```

This are the values of the mean squared error that we got from each FIS evaluation:

Optimization Method	Clustering Method	Mean Squared Error
Backpropagation	Subtractive	7.026356e-05
	Fuzzy c-means	4.972599e-05
	Grid Partition	3.048162e-01

Hybrid	Subtractive	4.023544e-11
	Fuzzy c-means	3.060494e-11
	Grid Partition	1.272667e-07

From the results above we can see that for all the clustering methods, the hybrid propagation method resulted in a smaller mean squared error by an evident difference. Also, when compared the fuzzy c-means clustering was the one with best results. So we decided to test it in a dynamic process simulation which will be shown next.

8. Fuzzy systems rules set

Below we have the ruler viewer of each trained FIS. We can see them by using Matlab's function *ruleview*.

- **Backpropagation + Subtractive Clustering (2 rules)**

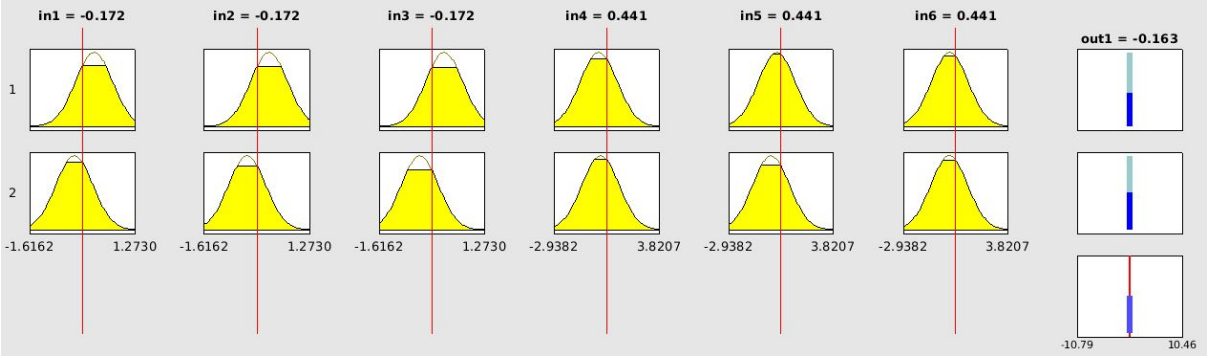


Figure 27. Backpropagation + Subtractive Clustering rule viewer

- **Backpropagation + Fuzzy c-means Clustering (2 rules)**

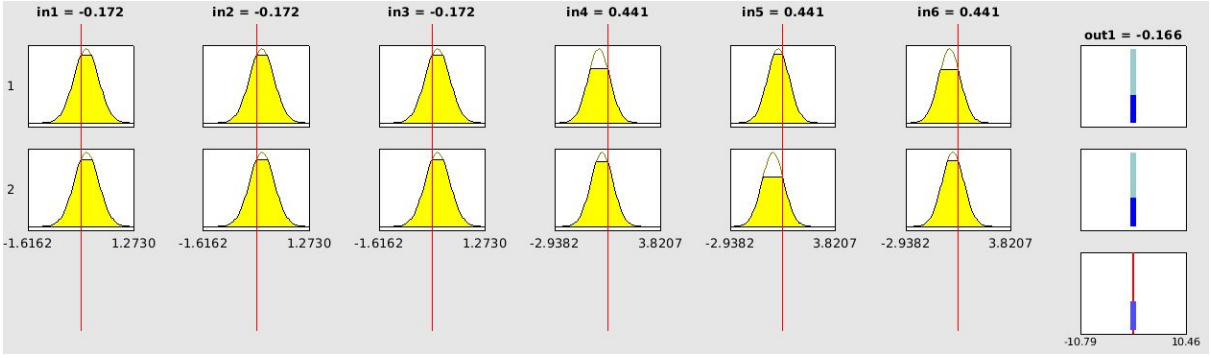


Figure 28. Backpropagation + Fuzzy c-means Clustering

- **Backpropagation + Grid partition Clustering (64 rules)**

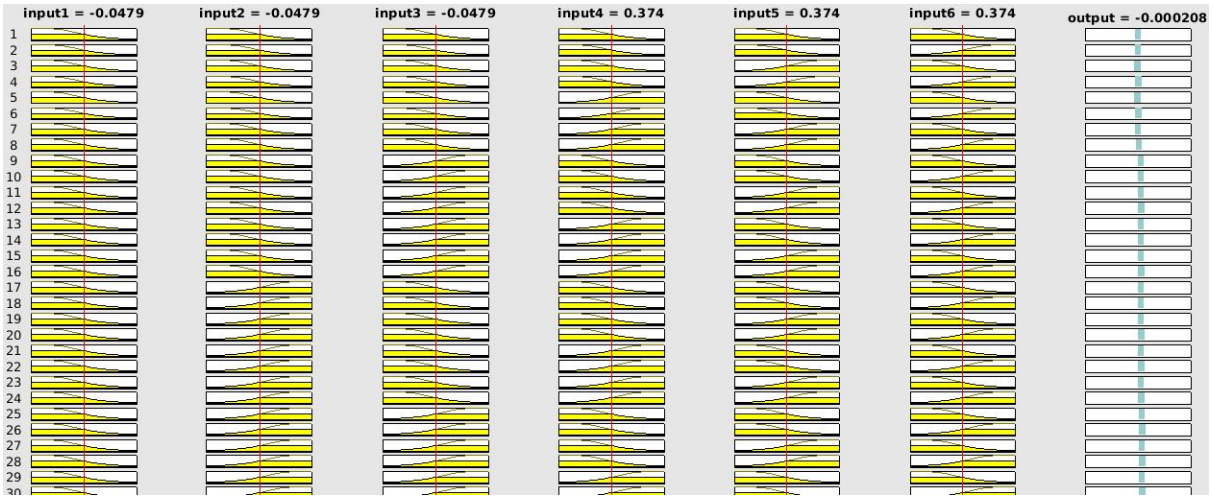


Figure 29. Backpropagation + Grid partition Clustering

- **Hybrid + Subtractive Clustering (3 rules)**

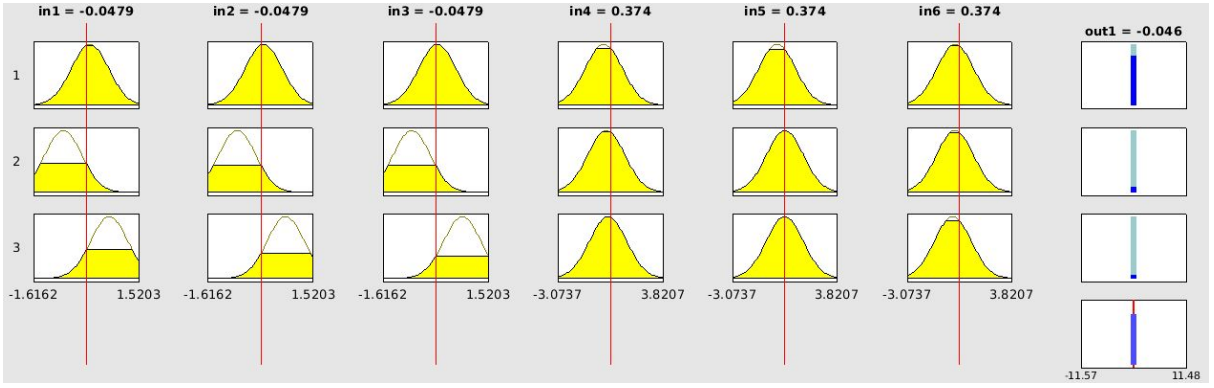


Figure 30. Hybrid + Subtractive Clustering

- **Hybrid + Fuzzy c-means Clustering (3 rules)**

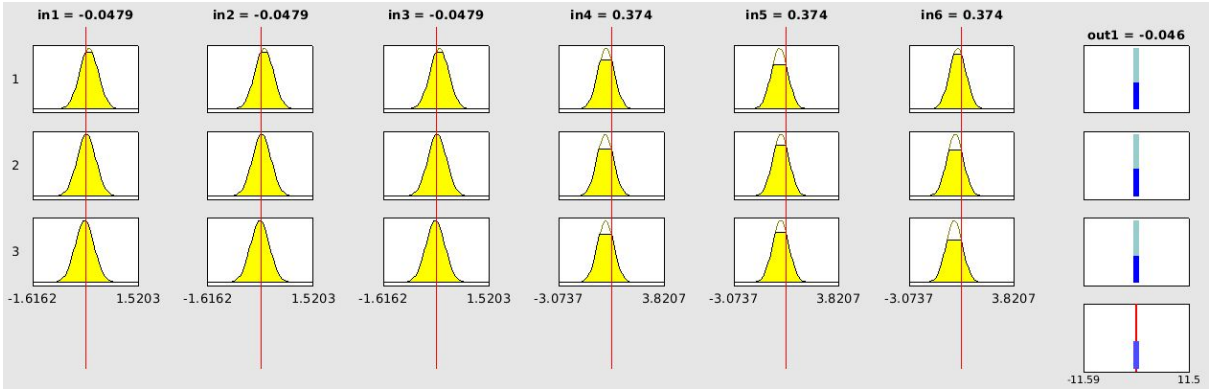


Figure 31. Hybrid + Fuzzy c-means Clustering

- **Hybrid + Grid partition Clustering**

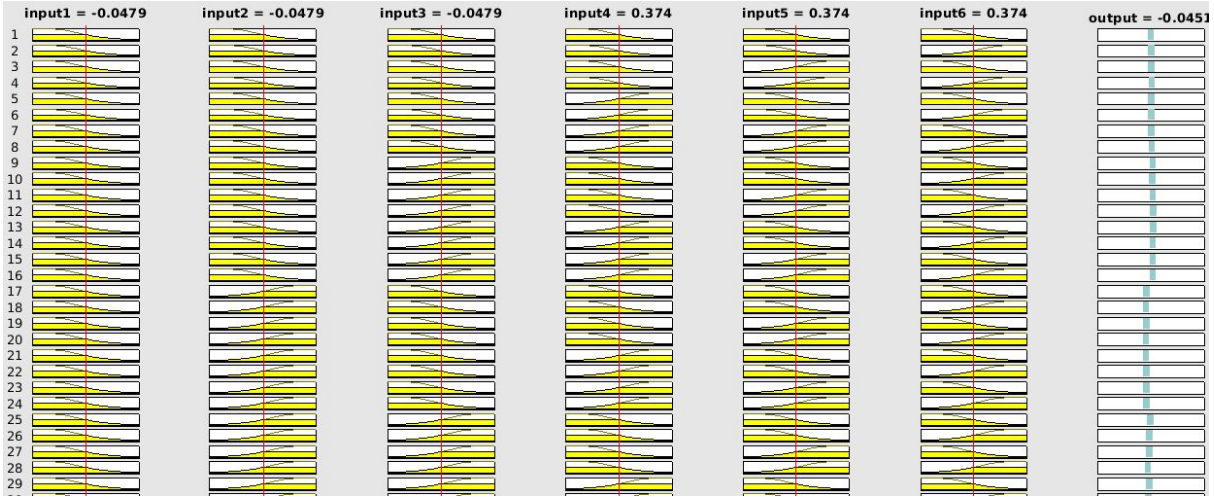


Figure 32. Hybrid + Grid partition Clustering

9. Testing the fuzzy systems in a simulation

9.1. Implementation

To test the fuzzy systems that we created and trained in previously, we replicated the model presented on the assignment statement.

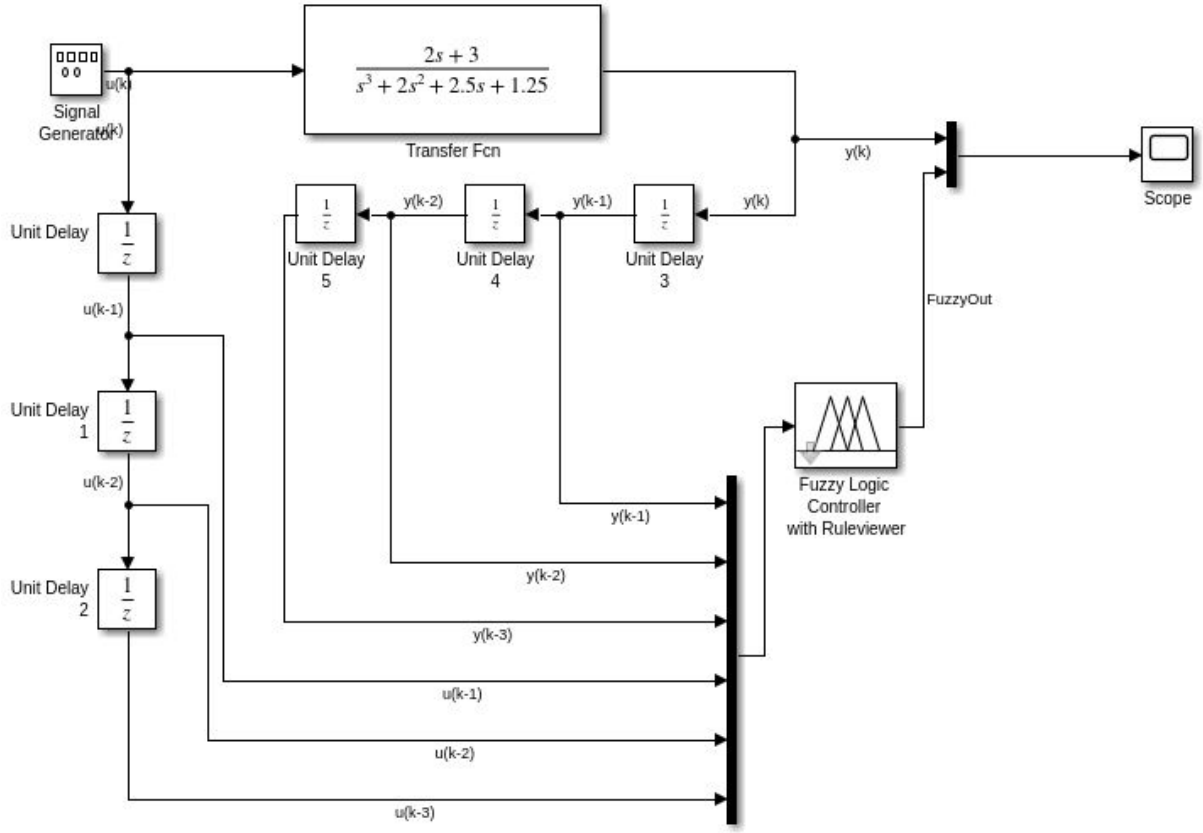


Figure 33. Model of a dynamic process used to test our FIS

Also, to have a small integration error we followed the suggestion and calculated the out sample time.

```
%Calculate de sampling time
poles = roots(den);
x = -1./real(poles);
Ts=min(x)*1/10;
```

For our transfer function the calculated function time output was **Ts = 0.1227**.

Figure 34. Simulation Ts simulation

Also in the delay units and as suggested we selected a sample time **four times** the Ts. In this case it was **0.4908 rad/s**. The signal generator has a frequency of **0.05 rad/s**. Also the fuzzy controller block has a **refresh rate of 0.9816 s**, eight times the sampling time (Ts).

9.2. Results

Below we can see the output graphical representation of the fuzzy systems with smaller mean squared error.

- **Hybrid Optimization + Fuzzy c-means (sinusoidal signal)**

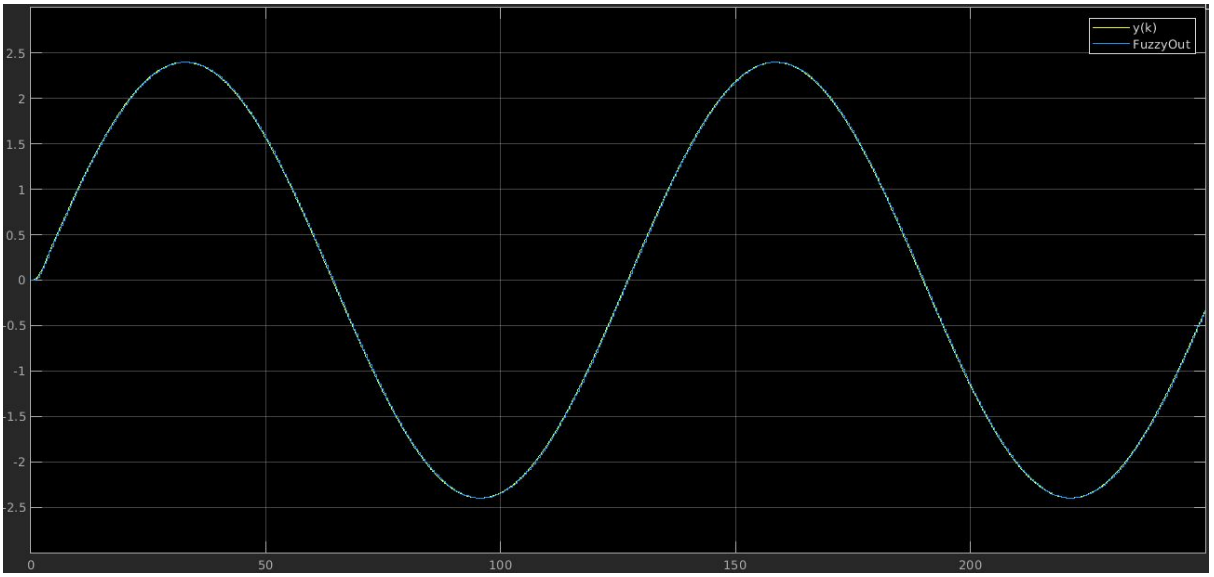


Figure 35. Simulation output with a sinusoidal signal

- **Hybrid Optimization + Fuzzy c-means (sawtooth signal)**

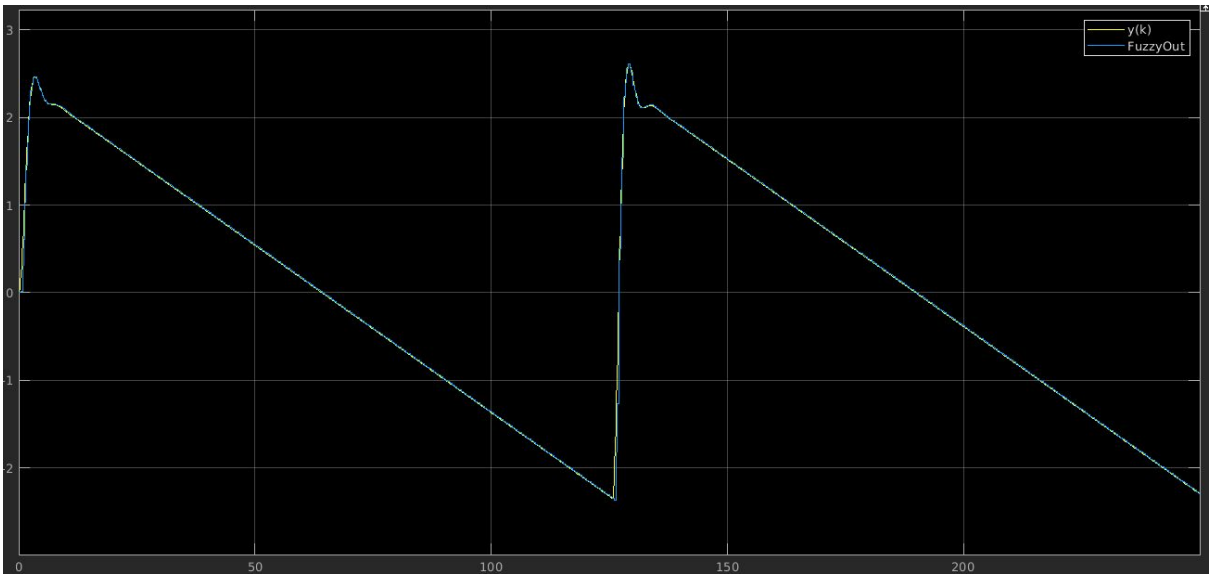


Figure 36. Simulation output with a sawtooth signal

- **Hybrid Optimization + Fuzzy c-means (square signal)**

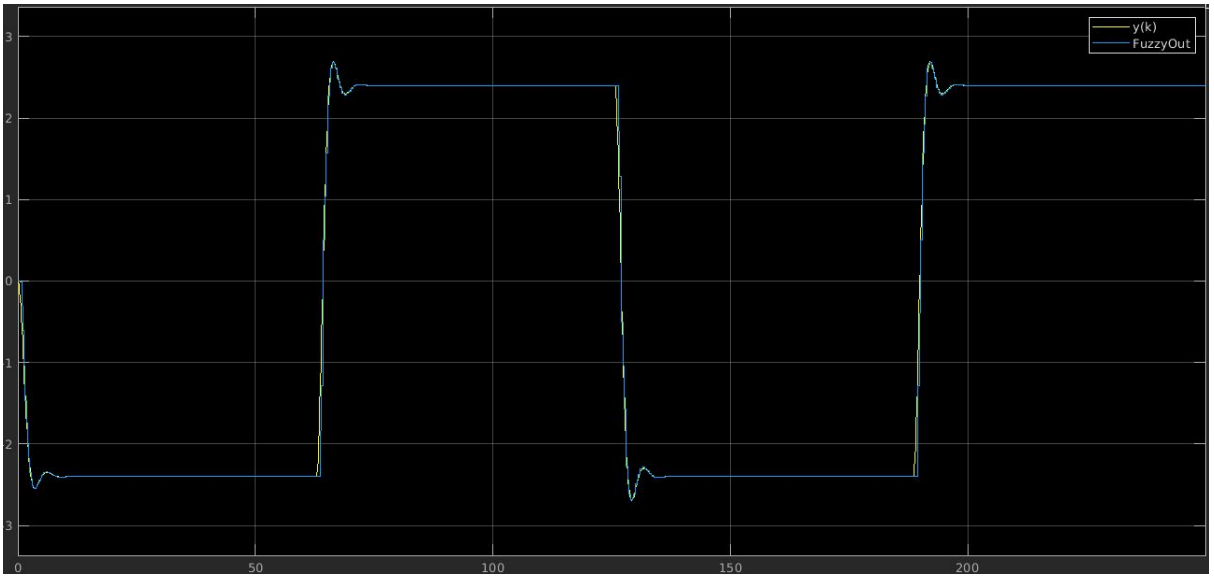


Figure 37. Simulation output with a squared signal

- **Hybrid Optimization + Fuzzy c-means (random signal)**

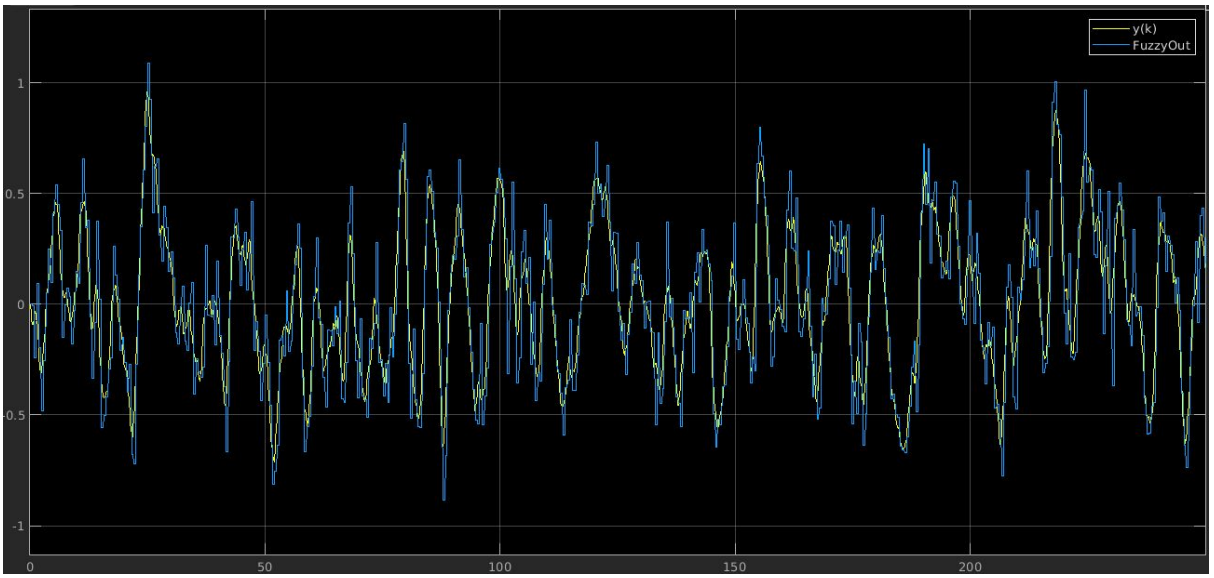


Figure 39. Simulation output with a random signal

As we can see from the graphical results above **this fuzzy controller has great results for the sinusoidal, square, sawtooth and even for random signals**.

10. Conclusions

From the results that we got from the clustering techniques we can see that the subtractive and fuzzy c-means are better than the grid partition since they were able to build fuzzy systems with a smaller number of rules. Also, after training the FIS with the backpropagation and hybrid methods from data that we build, we confirmed the hypothesis that we just referred to. Also, by looking at the mean squared error we can see that the fuzzy c-means clustering is the one with the smallest mse. Still, the hybrid optimization method turned out to be the best for every clustering technique.

After testing the fuzzy inference system with a dynamic process we were surprised how the graphical results show that with our best FIS the process has a perfect response to the generated signal. But this is not just for the squared, sinusoidal or sawtooth signals. We even tested the fuzzy system for a random signal and the results were fantastic.

Overall we were happy to learn more, not just about fuzzy systems, but also about different types of processes models. The work was also helpful to learn how to build the fuzzy systems with less work and greater results.

11. References

[1] A. Dourado, ML Chapter 6: Fuzzy Logic
[2] A. Dourado, ML Chapter 7: Fuzzy Systems
[3] A. Dourado, ML Chapter 8: Neuro Fuzzy Systems
[3] Matlab documentation: Fuzzy Logic User's Guide
[4] Matlab documentation: Simulink User's Guide
[5] McGonagle et al., Backpropagation