University of Coimbra

Machine Learning Report

Fuzzy Control and Fuzzy Modeling

Authors:

Adolfo Pinto, nº 2013138622, uc2013138622@student.uc.pt
Jani Hilliaho, nº 2016167354, uc2016167354@student.uc.pt

Faculty of Sciences and Technology
Department of Informatics Engineering

November 2016

Part A

Introduction

In this project, we were asked to implement two types of fuzzy inference controllers: Mamdani and Sugeno. Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Sugeno is similar to the Mamdani method in many aspects, except that in Sugeno the output membership functions are either linear or constant. Beneath the system, fuzzy logic rules are implemented.

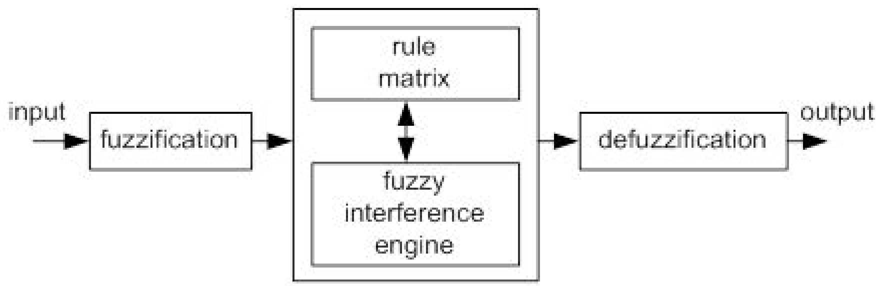

Figure 1. Fuzzy logic controller

As we can see in Figure 1, a fuzzy controller has an input unit, a processing unit and an output unit. In the input phase we measure the system's conditions. The processing phase, the longest one, is used to determine the action to be taken based on human determined fuzzy "if-then" rules, combined with non-fuzzy rules. The output is the "decision" or signal with a specific value.

Simulink, Fuzzy Logic Toolbox and *Matlab* were used to implement and test these functionalities.

In this work, we implemented controllers Mamdani and Sugeno with 9 and 25 rules.

Application

In this project, we implemented the fuzzy controllers stated before, using the *Fuzzy Logic Toolbox* from *Simulink.*

Using the >**fuzzyLogicDesigner** command in MATLAB, it lets us design and test fuzzy inference systems for modeling complex system behaviors.

Each group from class had to work with a specific transfer function *T*, in which our function was:

$$T = 12 / (s^3 + 6s^2 + 11s + 6)$$

References

As suggested in the assignment, it was implemented two types of references: **square wave** and **sinusoidal wave**. What he hope to see by using different types of signal waves is how it affects the performance of the controller.

Controllers and rules

The implemented controllers have 2 types with 2 types of rules.

- Mamdani controller with 9 rules
- Mamdani controller with 25 rules
- Sugeno controller with 9 rules
- Sugeno controler with 25 rules

The following figures represent the rules implemented by the controllers (extracted from the course's slides), where (abreviation - meaning):

- N - Negative
- ZE - Zero
- P - Positive
- NB - Negative Big
- NS - Negative Small
- PS - Positve Small
- PB - Positive Big

e_k \ Δe_k	N	ZE	P
N	N	N	Z
ZE	N	Z	P
P	Z	P	P

e_k \ Δe_k	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NS	ZE
NS	NB	NB	NS	ZE	PS
ZE	NB	NS	ZE	PS	PB
PS	NS	ZE	PS	PB	PB
PB	ZE	PS	PB	PB	PB

Figure 2. Rules implemented by a controller with 9 rules and a controller with 25 rules.

As one can except, by implement controllers with 25 rules, we hope to see an improvement in the controller's performance. By having a smal set of rules, the controller will yield a bigger error than using a controller with more rules. In some of the models implemented, only the rules changed and the other variables remained constant. In this case, we expected to see some improvement.

As we said before, the controller, in addition to its rules, has other variables. That said, the controller's implemented are: **Mamdani** or **Sugeno**, with **9** or **25** rules, **gaussmf** or **trimf** as membership functions, with **Centroid** or **MedMax** defuzzification, in the case of **Mamdani** and **Wtsum** or **Wtaver** in **Sugeno.**

How to run

To run the application, the user must run, first, the **main.m** matlab script, which will load all the fuzzy inference systems *(.fis*). After that, the user can run the *Simulink* models in the models folder. This operation, will open a new window the model implemented and the user can run the model and see its performance in the **Integral Squared Error** field. The output can be observered too, clicking on the **System Output & Ref.** The following figure illustrates one of the models implented in this project:

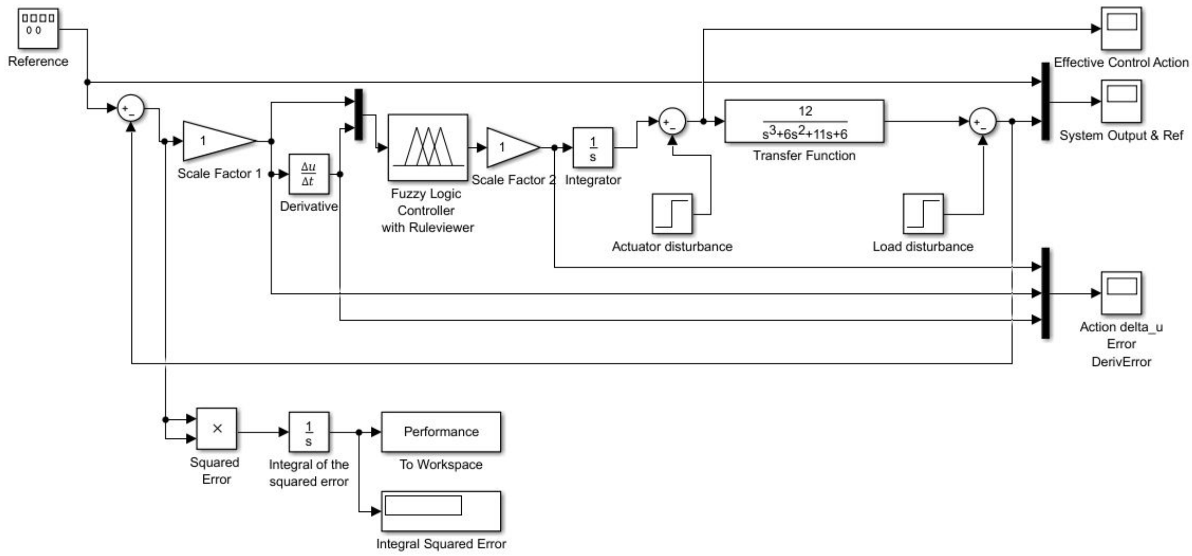

Figure 3. Model in Simulink

Experiments

The following tables describe the tests and simulations performed in the models implemented.

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller's Performance
Mamdani	9	sin	gaussmf	centroid	✘	✘	11.85
Mamdani	9	square	gaussmf	centroid	✘	✘	1498
Mamdani	9	sin	gaussmf	medmax	✘	✘	86.48
Mamdani	9	square	gaussmf	medmax	✘	✘	67.22
Mamdani	9	sin	trimf	centroid	✘	✘	5.108
Mamdani	9	square	trimf	centroid	✘	✘	1490
Mamdani	9	sin	trimf	medmax	✘	✘	72.38
Mamdani	9	square	trimf	medmax	✘	✘	377.1
Mamdani	25	sin	gaussmf	centroid	✘	✘	0.04255
Mamdani	25	square	gaussmf	centroid	✘	✘	192.6
Mamdani	25	sin	gaussmf	medmax	✘	✘	21.91
Mamdani	25	square	gaussmf	medmax	✘	✘	1592
Mamdani	25	sin	trimf	centroid	✘	✘	0.01734
Mamdani	25	square	trimf	centroid	✘	✘	1481
Mamdani	25	sin	trimf	medmax	✘	✘	14.87
Mamdani	25	square	trimf	medmax	✘	✘	1591

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller’s Performance
Sugeno	9	sin	gaussmf	wtaver	✘	✘	0.02732
Sugeno	9	square	gaussmf	wtaver	✘	✘	25.21
Sugeno	9	sin	gaussmf	wtsum	✘	✘	0.01219
Sugeno	9	square	gaussmf	wtsum	✘	✘	49.46
Sugeno	9	sin	trimf	wtaver	✘	✘	0.01292
Sugeno	9	square	trimf	wtaver	✘	✘	1.07e08
Sugeno	9	sin	trimf	wtsum	✘	✘	0.01292
Sugeno	9	square	trimf	wtsum	✘	✘	1.07e08
Sugeno	25	sin	gaussmf	wtaver	✘	✘	0.01547
Sugeno	25	square	gaussmf	wtaver	✘	✘	24.79
Sugeno	25	sin	gaussmf	wtsum	✘	✘	0.006899
Sugeno	25	square	gaussmf	wtsum	✘	✘	1621
Sugeno	25	sin	trimf	wtaver	✘	✘	0.008758
Sugeno	25	square	trimf	wtaver	✘	✘	1.07e08
Sugeno	25	sin	trimf	wtsum	✘	✘	0.008688
Sugeno	25	square	trimf	wtsum	✘	✘	1.07e08

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller's Performance
Mamdani	9	sin	gaussmf	centroid	✓	✘	14.52
Mamdani	9	square	gaussmf	centroid	✓	✘	1500
Mamdani	9	sin	gaussmf	medmax	✓	✘	65.29
Mamdani	9	square	gaussmf	medmax	✓	✘	2046
Mamdani	9	sin	trimf	centroid	✓	✘	7.74
Mamdani	9	square	trimf	centroid	✓	✘	1290
Mamdani	9	sin	trimf	medmax	✓	✘	62.46
Mamdani	9	square	trimf	medmax	✓	✘	2048
Mamdani	25	sin	gaussmf	centroid	✓	✘	1.202
Mamdani	25	square	gaussmf	centroid	✓	✘	206.7
Mamdani	25	sin	gaussmf	medmax	✓	✘	15.54
Mamdani	25	square	gaussmf	medmax	✓	✘	1547
Mamdani	25	sin	trimf	centroid	✓	✘	1.18
Mamdani	25	square	trimf	centroid	✓	✘	1490
Mamdani	25	sin	trimf	medmax	✓	✘	15.54
Mamdani	25	square	trimf	medmax	✓	✘	1548

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller's Performance
Sugeno	9	sin	gaussmf	wtaver	✓	✘	0.9333
Sugeno	9	square	gaussmf	wtaver	✓	✘	25.81
Sugeno	9	sin	gaussmf	wtsum	✓	✘	0.6601
Sugeno	9	square	gaussmf	wtsum	✓	✘	49.82
Sugeno	9	sin	trimf	wtaver	✓	✘	0.917
Sugeno	9	square	trimf	wtaver	✓	✘	1.07e08
Sugeno	9	sin	trimf	wtsum	✓	✘	0.7047
Sugeno	9	square	trimf	wtsum	✓	✘	1.07e08
Sugeno	25	sin	gaussmf	wtaver	✓	✘	0.8099
Sugeno	25	square	gaussmf	wtaver	✓	✘	25.36
Sugeno	25	sin	gaussmf	wtsum	✓	✘	0.6459
Sugeno	25	square	gaussmf	wtsum	✓	✘	1509
Sugeno	25	sin	trimf	wtaver	✓	✘	0.8318
Sugeno	25	square	trimf	wtaver	✓	✘	1.07e08

Sugeno	25	sin	trimf	wtsum	✓	✘	0.6744
Sugeno	25	square	trimf	wtsum	✓	✘	1.07e08

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller's Performance
Mamdani	9	sin	gaussmf	centroid	✘	✓	13.59
Mamdani	9	square	gaussmf	centroid	✘	✓	1503
Mamdani	9	sin	gaussmf	medmax	✘	✓	62.23
Mamdani	9	square	gaussmf	medmax	✘	✓	2013
Mamdani	9	sin	trimf	centroid	✘	✓	6.67
Mamdani	9	square	trimf	centroid	✘	✓	1494
Mamdani	9	sin	trimf	medmax	✘	✓	67.2
Mamdani	9	square	trimf	medmax	✘	✓	2014
Mamdani	25	sin	gaussmf	centroid	✘	✓	1.463
Mamdani	25	square	gaussmf	centroid	✘	✓	206.5
Mamdani	25	sin	gaussmf	medmax	✘	✓	14.8

Mamdani	25	square	gaussmf	medmax	✘	✓	1916
Mamdani	25	sin	trimf	centroid	✘	✓	1.408
Mamdani	25	square	trimf	centroid	✘	✓	1492
Mamdani	25	sin	trimf	medmax	✘	✓	17.54
Mamdani	25	square	trimf	medmax	✘	✓	1659

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzificatio n Method	Perturbatio n in actuator	Perturbatio n in load	Controller's Performanc e
Sugeno	9	sin	gaussmf	wtaver	✘	✓	1.352
Sugeno	9	square	gaussmf	wtaver	✘	✓	28.39
Sugeno	9	sin	gaussmf	wtsum	✘	✓	1.251
Sugeno	9	square	gaussmf	wtsum	✘	✓	51.23
Sugeno	9	sin	trimf	wtaver	✘	✓	1.308
Sugeno	9	square	trimf	wtaver	✘	✓	1.07e08
Sugeno	9	sin	trimf	wtsum	✘	✓	1.301
Sugeno	9	square	trimf	wtsum	✘	✓	1.07e08

Sugeno	25	sin	gaussmf	wtaver	✘	✓	1.306
Sugeno	25	square	gaussmf	wtaver	✘	✓	27.45
Sugeno	25	sin	gaussmf	wtsum	✘	✓	1.269
Sugeno	25	square	gaussmf	wtsum	✘	✓	1550
Sugeno	25	sin	trimf	wtaver	✘	✓	1.297
Sugeno	25	square	trimf	wtaver	✘	✓	1.07e08
Sugeno	25	sin	trimf	wtsum	✘	✓	1.396
Sugeno	25	square	trimf	wtsum	✘	✓	1.07e08

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller's Performance
Mamdani	9	sin	gaussmf	centroid	✓	✓	19.44
Mamdani	9	square	gaussmf	centroid	✓	✓	1799
Mamdani	9	sin	gaussmf	medmax	✓	✓	69.17
Mamdani	9	square	gaussmf	medmax	✓	✓	1296
Mamdani	9	sin	trimf	centroid	✓	✓	11.58

Mamdani	9	square	trimf	centroid	✓	✓	1797
Mamdani	9	sin	trimf	medmax	✓	✓	76.39
Mamdani	9	square	trimf	medmax	✓	✓	74.67
Mamdani	25	sin	gaussmf	centroid	✓	✓	5.307
Mamdani	25	square	gaussmf	centroid	✓	✓	2560
Mamdani	25	sin	gaussmf	medmax	✓	✓	20.66
Mamdani	25	square	gaussmf	medmax	✓	✓	300.5
Mamdani	25	sin	trimf	centroid	✓	✓	4.898
Mamdani	25	square	trimf	centroid	✓	✓	3526
Mamdani	25	sin	trimf	medmax	✓	✓	19.5
Mamdani	25	square	trimf	medmax	✓	✓	3131

Type of controller	Nº of rules	Signal Wave	Membership Function	Defuzzification Method	Perturbation in actuator	Perturbation in load	Controller’s Performance
Sugeno	9	sin	gaussmf	wtaver	✓	✓	4.136
Sugeno	9	square	gaussmf	wtaver	✓	✓	31.78

Sugeno	9	sin	gaussmf	wtsum	✓	✓	3.721
Sugeno	9	square	gaussmf	wtsum	✓	✓	56.99
Sugeno	9	sin	trimf	wtaver	✓	✓	3.878
Sugeno	9	square	trimf	wtaver	✓	✓	1.07e08
Sugeno	9	sin	trimf	wtsum	✓	✓	4.033
Sugeno	9	square	trimf	wtsum	✓	✓	1.07e08
Sugeno	25	sin	gaussmf	wtaver	✓	✓	3.951
Sugeno	25	square	gaussmf	wtaver	✓	✓	30.7
Sugeno	25	sin	gaussmf	wtsum	✓	✓	4.421
Sugeno	25	square	gaussmf	wtsum	✓	✓	2411
Sugeno	25	sin	trimf	wtaver	✓	✓	3.896
Sugeno	25	square	trimf	wtaver	✓	✓	1.07e08
Sugeno	25	sin	trimf	wtsum	✓	✓	6.081
Sugeno	25	square	trimf	wtsum	✓	✓	1.07e08

As we can see from the results, the **sin** function has better results than the **square** function. Looking at the membership function, **trimf** gives better results than the couterpart, **gaussmf,** but it's not too relevant**.** Generally, the type of controller that presents best results is the **Sugeno** type. In this project, besides the models implemented, we introduced some disturbances in the actuator, the load and both at the same time, which we can see the difference from the table, where all other variables remain constant.

Since we performed too many experiments, the resulting plots can be consulted in the same folder the project was delivered too. The next two images, illustrates the output from two models implemented:

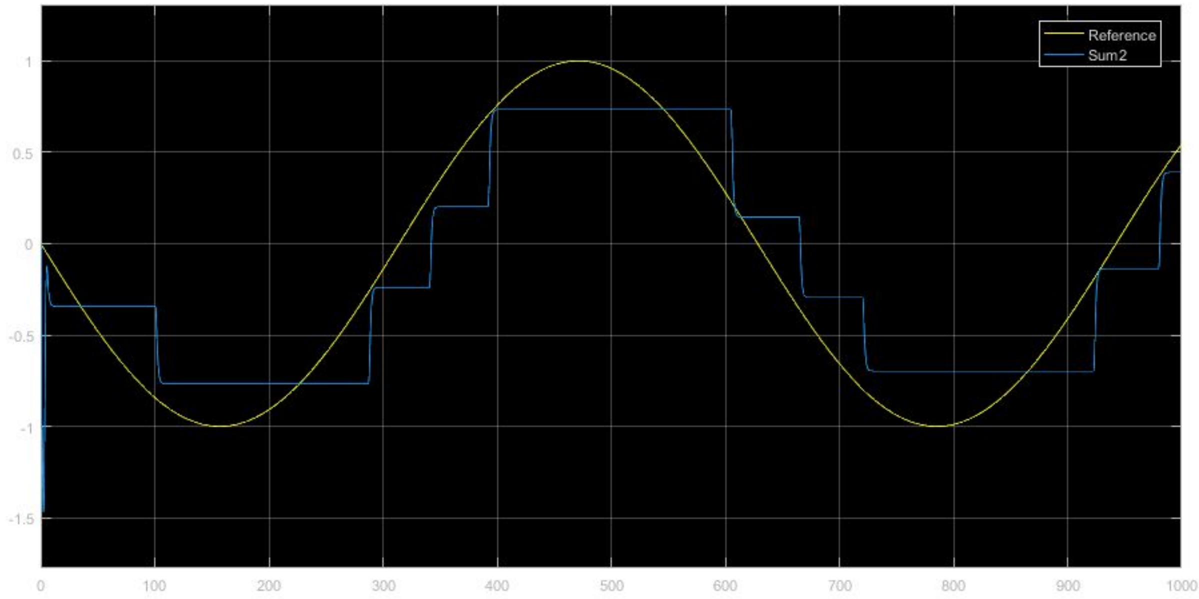

Figure 4. Output from the senoide signal.

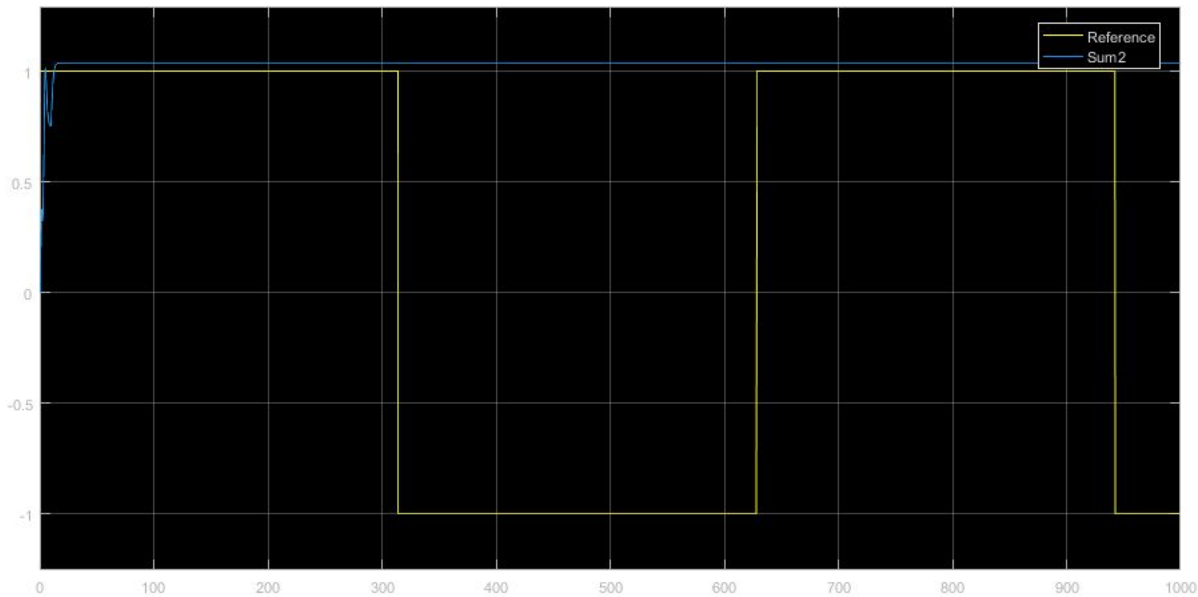

Figure 5. Output from the square signal.

Part B

Creating data matrix

The neuro-fuzzy systems can be used for modelling different dynamical systems. They can be trained by using input-output data by obtaining the initial rules with some clustering technique, and by optimization after it. The ANFIS architecture makes this work easier, because it implements these both phases. ANFIS GUI can be used by executing command *neuroFuzzyDesigner*.

To generate a good data set for training, the used data must represent the system's dynamics, varying through the used range. A good technique for creating such data is to use a random input sequence. Consider the transfer function used in Part A. During this work, a model was created from it in Simulink. In this model, a random number sequence was used as an input, and the input and output of the transfer function were saved to variables inputRandom and DiscreteOut. The model is saved to *model_simple.slx* and it is shown in figure 1.

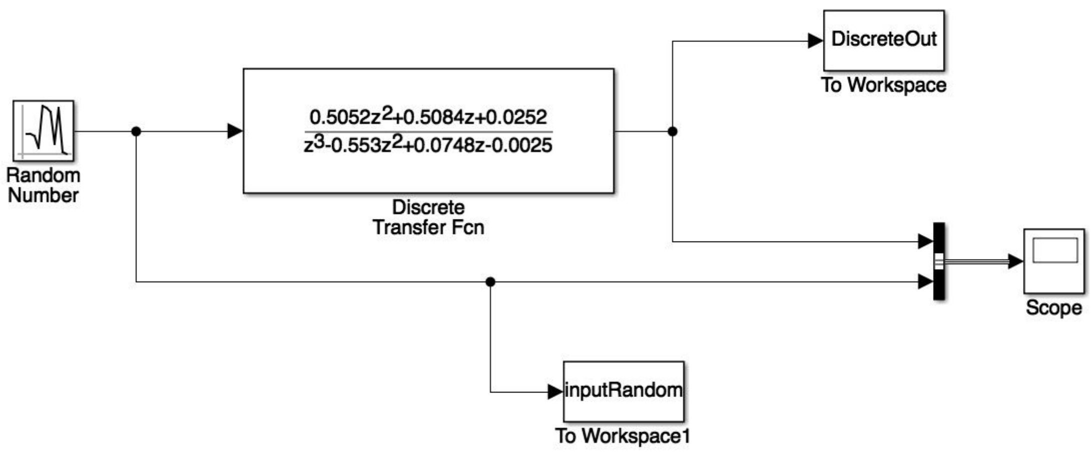

Figure 1: The Simulink model for creating the testing and training data

After simulation, a data matrix was created from the data in inputRandom and DiscreteOut. The data matrix consists of 7 columns and 99 rows, and every row of it was calculated with following formula:

row(i) = y(i-1) y(i-2) y(i-3) u(i-1) u(i-2) u(i-3) y(i),

where *i* means row number, *y* means DiscreteOut and *u* means inputRandom. In this matrix, 6 first columns of every row are inputs of created Fuzzy Inference System, and the last column is the output of it. After creating the data matrix, it was divided to training and testing data. The first 70 rows are used for training, and the last 29 rows for testing.

Creating and testing the systems

Fuzzy inference systems with two different ways of clustering were created during this work. The systems were created using functions genfis2 and genfis3 with their default settings so that *gaussmf* is used in all membership functions. Function genfis2 generates Fuzzy Inference System structure from the data using subtractive clustering, and function genfis3 using FCM (Fuzzy C-means) clustering. Both systems were created with training data. There is no need for training after running these functions because functions train the systems after creating them.

These systems were tested by testing data with function *evalfis*, which computes the output of the fuzzy inference systems from the used test input data. The outputs of the *evalfis* were then compared with the expected output data to calculate mean squared error. During the tests, MSE was 0.0563 with FCM clustering and 0.0349 with subtractive clustering.

Building the block diagram

After testing the two systems, a new Simulink model was created for simulating them and for comparing their outputs with the output of the transfer function. The model can be found from file *model.slx* and it is shown in figure 2.

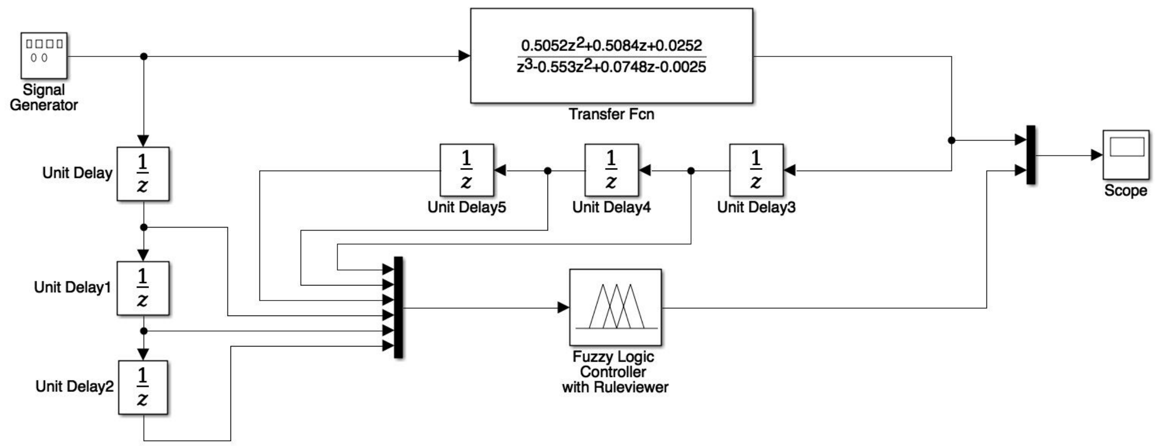

Figure 2: The final Simulink model

Before running the simulation, the sampling time must be set. The suitable range for sampling time was calculated from the transfer function, and it was decided to be 0.2.

Running the system

The two Fuzzy Inference Systems can be created by running the MATLAB script *program.m*. It will then create and save the systems to files *fcmfis.fis* (Clustered with FCM) and *scfis.fis* (Clustered with subtractive clustering). All details of these systems can be found from these files. It will also test them with the testing data and print MSE values to the console. After running the *program.m* the simulink model can be started by running the file *model.slx.* The input signal used in simulation can be selected from Signal Generator, and the input and output signals can be seen using the scope. The used FIS can be selected by changing its name from Fuzzy Logic Controller with Ruleviewer.