FCTUC DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

MASTER IN INFORMATICS ENGINEERING
MASTER IN BIOMEDICAL ENGINEERING

MACHINE LEARNING
2023-2024

Assignment nº 4 Fuzzy and Neuro-Fuzzy Systems

(Theoretical content: Chapters 6,7,8)

Learning objectives:

1- To project and test a fuzzy controller (Mamdani and TSK)
2- To build a fuzzy model of a dynamic process, using clustering and optimization.

PART A – FUZZY CONTROL

We will implement, in SIMULINK environment, a Mamdani type1 and a Sugeno type-1 fuzzy controllers.

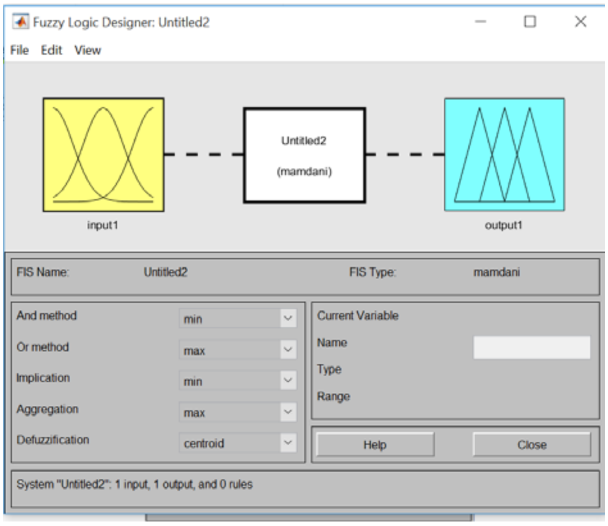
The theoretical material involved is composed by chapters 6, 7 and 8**. It is recommended the reading of the note "Conceitos Básicos de Controlo" and/or the slides "Introduction to Automatic Control" available in the course website**. The Fuzzy Logic Toolbox Users' Guide is also very useful.

Each group will work with a dynamic system representative of a hypothetical real process, given by its continuous transfer function in a Simulink block.

Using the *fisditor*, writing in the Matlab command line

>fuzzyLogicDesigner (or > fuzzy, previous version still working and more simple)

a very practical GUI is opened to design a rule based fuzzy system, either Mamdani or Sugeno type.



Basic window of *fiseditor*

The several menus of *fiseditor* allow

- to define the number of inputs and outputs (edit > add/remove variable),

- define the membership functions for each input and each output (edit> membership functions),

- to write the rules using the created membership functions (edit> rules).

The several operators for conjunction/disjunction of the antecedents (And/Or method), for implication, for aggregation of the outputs of each rule, the defuzzification method, studied in the classes, are chosen in the shown window.

After creating the fuzzy system, it is saved, for example named *myController.fis,* in a file (export it with file>export>to file), for posterior use, and it must also be exported to the Matlab working space (file > export > to workspace).

To use a controller previously developed, it must be firstly imported to the fiseditor (file>import>from file) and afterwards exported to the working space. If you try to execute 'myController.fis' in the command line, an error appears, because it takes the file as ASCII. **To import a fis from a directory to the working space may be done through the fiseditor or by the function readfis:**

FISMAT=readfis('filename') creates a FIS matrix in the workspace corresponding to the FIS file 'filename' on disk.

Too save a fis in a directory, the function writeFIS(FIS) can be used. See Matlab help.

Processes to be used: see the table.

Controllers to be implemented (minimum requirement):

- Mamdani with (i) 9 rules, (ii) 25 rules

- Sugeno with (i) 9 rules, (ii) 25 rules

You may also try 49 rules, if your process needs it to be well controlled.

Some advices:

1- The choice of the scale factors is probably the main challenge of the work. For the controller to work well, the error and its variation must be in the interval [-1 1], leading to the simultaneous firing of several rules. If they remain at the extreme of this interval, then the same rule is always applied, and the system may become instable. This may guide the choice of the scale factor 1 (at the input of the controller), such that the error and its derivative swing between positive and negative values. On the other side, the input to the process should not be always very big (positive or negative), and this gives some light to the choice of the scale factor 2. Note however that to some extend the scale factors “interact” with each other, and some experience is needed to get finally good values for them.

2- The visualization of the error, its derivative, and the control action, helps to verify if the rules are working well.

3- The reference should vary from time to time, for example as a square wave, and the process output must follow it. A sinusoidal or a saw-tooth reference are also interesting, allowing to check if the controller is good for the tracking problem. The higher the reference, the higher must be the control action, and adjustments of the scale factors may be needed.

4- The performance of the controller must be analysed with respect to three objectives: follow the reference, compensate the load disturbances, and compensate the actuator disturbances.

5- A measure of the performance can be computed in Simulink, calculating the square of the error in each instant and integrating it. The evolution of the performance in real time can be observed in a display register (as it is implemented in the slide 547 of the course).

6- Another important measurement of performance is the control effort, since this reflects the cost of the control. The maximum value of the control signal (the Effective control action in the Simulink diagram), the integral of the squared control signal (than can be computed in a similar way, in the Simulink diagram, as the integral of the squared error), are measures that the control engineer must take into account. To have a balance between the error and the control cost, one can use the geometric mean of the two, which can be said it is the best criteria to compare controllers. The lower the geometric mean, the best the controller is. See slide 547 of the course.

7- An interesting data synthesising the properties of your system are the roots of the numerator polynomial (the zeros) and the roots of the denominator polynomial (the poles). Compute them and write them in your report. For that use the Matlab function roots: for example for $s^3+6.5s^2+13s+7.5$, > roots([1 6.5 13 7.5]) gives -1, -2.5, -3,. The roots may be real or complex conjugate.

Remarks:

1- The refresh rate in the Fuzzy Controller block controls the frequency with which the rule viewers are updated. A low value requires more time for graphication and the simulation will be slower.

Report for part A:

1- A pdf file describing the controllers and their performance, namely by plots.

2- The files *.fis with the controllers and the Simulink diagram *.slx.

Note that all delivered reports must be a single .zip or .rar file with the name ML2023FuzzyPLyGx. The report must also be named ML2023FuzzyPLyGx.pdf

Table. Transfer functions to be used.

Group 1 PL1 $\frac{3}{s^3+2s^2+2.5s+1.5}$	Group 2 PL1 $\frac{1.5}{s^3+2.8s^2+2.31s+0.54}$	Group 3 PL1 $\frac{3}{s^3+5s^2+6.75s+2.25}$	Group 4 PL1 $\frac{25}{s^3+9s^2+28s+30}$
Group 1 PL2 $\frac{5}{s^3+5s^2+9s+5}$	Group 2 PL2 $\frac{0.7}{s^3+4.6s^2+4.95s+0.45}$	Group 3 PL2 $\frac{5(s+1)}{s^3+6s^2+13s+10}$	
Group 4 PL2 $\frac{3}{s^3+6s^2+11s+6}$	Group 5 PL2 $\frac{6(s+3)}{s^3+9s^2+28s+19}$	Group 6 PL2 $\frac{15(s+3)}{s^3+8s^2+22s+20}$	Group 7 PL2 $\frac{2(s+1.5)}{s^3+2s^2+2.5s+1.5}$
Group 8 PL2 $\frac{1.5}{s^3+4.5s^2+5s+1.5}$			
Group 1 PL3 $\frac{s+2}{s^3+2s^2+2.5s+1.25}$	Group 2 PL3 $\frac{2(s+2.5)}{s^3+5s^2+9s+5}$	Group 3 PL3 $\frac{1.35}{s^3+4.6s^2+4.95s+0.45}$	Group 4 PL3 $\frac{15}{s^3+8s^2+22s+20}$
Group 5 PL3 $\frac{2s+1}{s^3+3s^2+4s+2}$	Group 6 PL3 $\frac{4s+8}{s^3+3s^2+4s+2}$	Group 7 PL3 $\frac{11}{s^3+7s^2+16s+12}$	Group 8 PL3 $\frac{0.9}{s^3+2.8s^2+2.31s+0.54}$
Group 9 PL3 $\frac{10}{s^3+8s^2+22s+20}$	Group10 PL3 $\frac{1.5(s+0.25)}{s^3+3s^2+2.25s+0.55}$		

ASSIGNMENT 4 PART B

NEURO-FUZZY SYSTEMS FOR MODELLING DYNAMIC PROCESSES

The neuro-fuzzy systems can be used for modelling dynamical systems. For a generic system, an input is applied, and it produces an output (Fig 1).

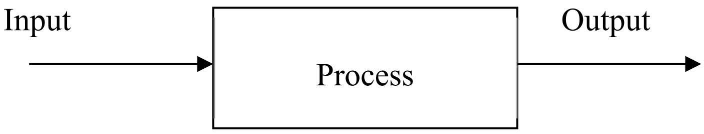


Figure 1.

Dynamic systems have memory and are described by difference equations in a NARX way (see course slides 326 to 330). In general, the output at instant k, $y(k)$, depends on some outputs in previous instants k-1, k-2, …, and on past inputs. For systems with inertia, as are all the practical ones, the output at instant k does not depend on the input at that same instant k. Because of the inertia it takes some time until the input effect reaches the output. We have then the difference equation (1).

$$y(k) = f(y(k-1), y(k-2), \dots, u(k-1), u(k-2) \dots) \quad (1)$$

If, for example we have a system with the form (2), for illustration of the concept,

$$y(k) = f(y(k-1), y(k-2), y(k-3), u(k-1)) \quad (2)$$

the following fuzzy rules (3) will be needed (zero order TSK type).

IF *y(k-1)* is A_1 AND *y(k-2)* is A_2 AND *y(k-3)* is A_3 AND *u(k-1)* is A_4 THEN *y(k)* is α (3)

where A_1, A_2, A_3, and A_4 are fuzzy sets.

The learning phase of the fuzzy system consist in the determination of the membership functions of the antecedents and of the constant terms in the consequents.

Learning is done using input-output data conveniently processed by:

- a clustering technique to obtain the initial rules (for example *subtractive*, c-*means, or fuzzy c-means*),
- optimization (by one of the studied methods) of the configuration, looking to minimizing the obtained squared error (difference between the measures real output and the fuzzy output after defuzzification).

The Anfis architecture makes our task easier, because it implements the two phases (if the subtractive method is used). The Anfis GUI is executed by

>*neuroFuzzyDesigner* (ou >*anfisedit*, previous version still active and simpler to use).

The building of the initial *fis* can also be done by the functions *genfis* in the command line.

Consider the system given by the transfer function of Part A, for your group. Using Simulink, give it an input and register the output. Two temporal series are then obtained (two vectors), one with the input values, the other with the output values. As the systems are of third order and have inertia (because the degree of the denominator is higher than the degree of the numerator, i.e., more poles that zeros), it can be shown that after discretization the output memory goes until the instant *k-3* in the output and in the input, as given by

$$y(k) = f(y(k-1), y(k-2), y(k-3), u(k-1), u(k-2), u(k-3))$$

To generate a good data set for training is a fundamental issue in the design of a set of rules. The data must be representative of the system's dynamics, varying through all the admissible domain. A good technique is to use a random input sequence that makes the output to move in all the (output) space. The following block diagram allows to make this.
A discrete version of the transfer function is used to produce the temporal series with the same number of elements referred to the same time instants.

To find a discrete transfer function, corresponding to the continuous one, a discretization stage is needed. For that, the first step is to fix the discretization interval, for example in one second (the input and output are measured only every second), to obtain the discrete transfer function. For example consider the following G(s)

$$G(s) = \frac{Y(s)}{U(s)} = \frac{10s+5}{s^3+6s^2+16s+8}$$

Then make:

>> num=[10 5] % specify the vector of numerator coeficients
>> den=[1 6 16 8] %% sepecify the vector of denominator coeficients
>>[numd,dend]=c2dm(num,den,Ts,'zoh') % apply the discretization method 'zoh' % Ts is the sampling <u>time that must be computed as indicaded in the following</u> (see **Best value for the sampling time** on page 11). It is made here 1, just for illustration,.

And one obtains

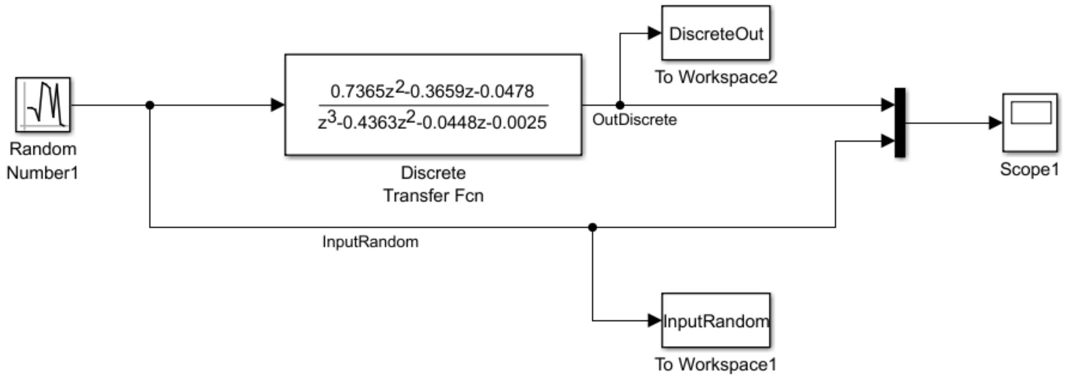
numd =
    0   0.7365   -0.3659   -0.0478   %the discrete numerator coeficients

dend =
   1.0000   -0.4363   -0.0448   -0.0025   % the discrete denominator coeficients

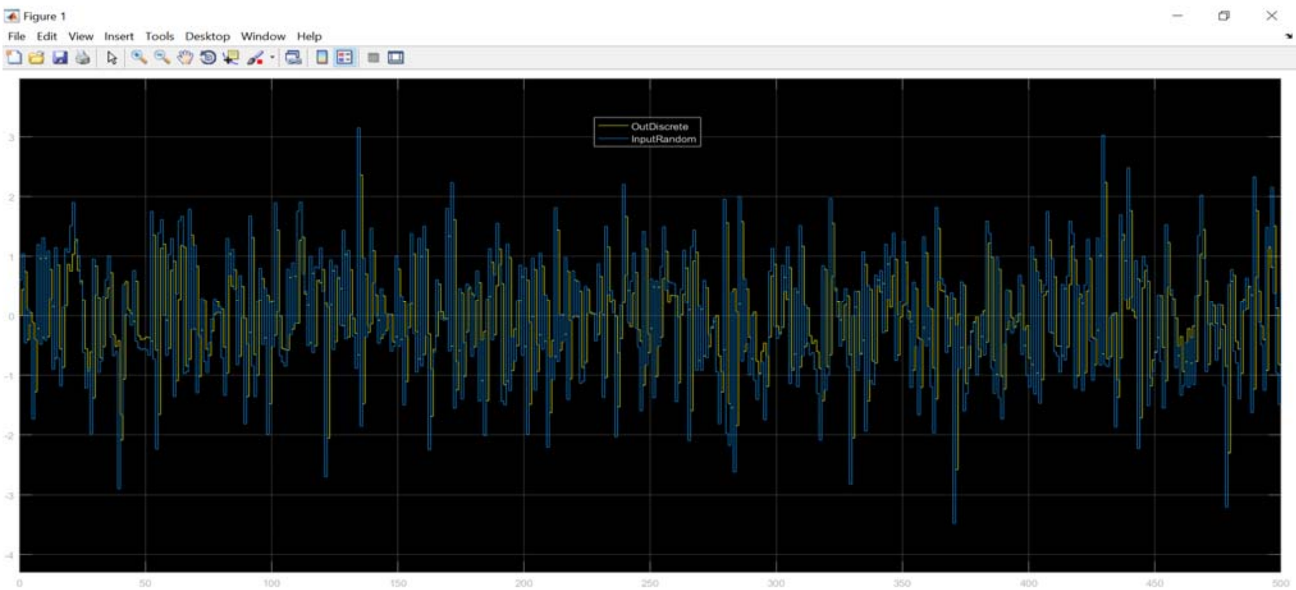
c2dm means "continuous to discrete transformation with a specified method"; its arguments are the numerator and denominator polinomials of the s-transfer function, the discretization interval (1 in the case), and the discretization method ('zoh' in the case, zero-order hold). The following transfer function is obtained:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0z^3+0.7365z^2-0.3659z-0.0478}{z^3-0.4363z^2-0.0448z-0.0025}$$

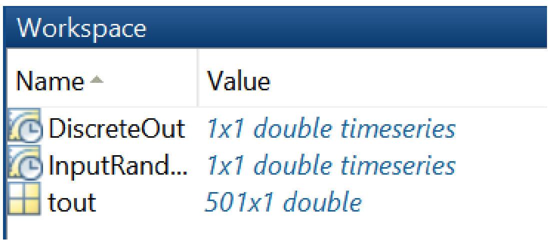
This transfer function in powers of z (the z-transfer function) is written in the bloc "Transf function" in the Discrete collection of Simulink. Note the order of the coefficients and the powers of z. Note also that the coefficient of z^3 in the numerator is zero; this is a consequence of the system's inertia.



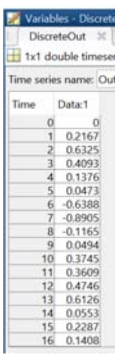
Simulating, with a discretization interval of 1s specified in the RandomNumber block dialog window, the following figure is obtained:



At this moment the Matlab working space has the following:



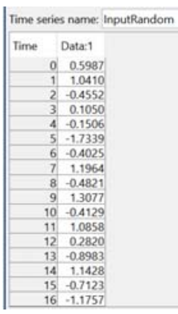
Name	Value
DiscreteOut	1x1 double timeseries
InputRand...	1x1 double timeseries
tout	501x1 double



Variables - DiscreteOut

Time series name: Out

Time	Data:1
0	0
1	0.2167
2	0.6325
3	0.4093
4	0.1376
5	0.0473
6	-0.6388
7	-0.8905
8	-0.1165
9	0.0494
10	0.3745
11	0.3609
12	0.4746
13	0.6126
14	0.0553
15	0.2287
16	0.1408



Time series name: InputRandom

Time	Data:1
0	0.5987
1	1.0410
2	-0.4552
3	0.1050
4	-0.1506
5	-1.7339
6	-0.4025
7	1.1964
8	-0.4821
9	1.3077
10	-0.4129
11	1.0858
12	0.2820
13	-0.8983
14	1.1428
15	-0.7123
16	-1.1757

From the z-transfer function we can obtain:

$$\frac{Y(z)}{U(z)} = \frac{0z^3 + 0.7365z^2 - 0.3659 - 0.0478}{z^3 - 0.4363z^2 - 0.0448z - 0.0025}$$

by cross multiplying

$$(z^3 - 0.4363z^2 - 0.0448z - 0.0025)Y(z) = (0z^3 + 0.7365z^2 - 0.3659z - 0.0478)U(z)$$
$$z^3Y(z) - 0.4363z^2Y(z) - 0.0448zY(z) - 0.0025Y(z) = 0.7365z^2U(z) - 0.3659zU(z) - 0.0478U(z)$$

in discrete time domain, applying the inverse transform, the following difference equation appears

$$y(k+3) - 0.4363y(k+2) - 0.0448y(k+1) - 0.0025y(k) = 0.7365u(k+2) - 0.3659u(k+1) - 0.0478u(k)$$

now subtracting 3 to all time indices,

$$y(k) - 0.4363y(k-1) - 0.0448y(k-1) - 0.0025y(k-3) = 0.7365u(k-1) - 0.3659u(k-2) - 0.0478u(k-3)$$

and resolving for y(k)

$$y(k) = 0.4363y(k-1) + 0.0448y(k-1) + 0.0025y(k-3) + 0.7365u(k-1) - 0.3659u(k-2) - 0.0478u(k-3)$$

So in general, for the nonlinear case we can write,

$$y(k) = f(y(k-1), y(k-2), y(k-3), u(k-1), u(k-2), u(k-3))$$

The first task is to construct the matrix with data for clustering. Each line of this matrix contains the antecedents and the consequent of each rule. So it will have 7 columns: six for the antecedents and the last one for the consequents.

The 7th column is the time series of the output. It is our target. The learning should start only at instant 3, because of the past values of the input and the output it depends on (there are no negative values for *k*).

If we have the generic rule

IF *y(k-1)* is A_1 AND y*(k-2)* is A_2 AND *y(k-3)* is A_3 AND *u(k-1)* is A_4 AND *u(k-2)* is A_5 AND *u(k-3)* is A6 THEN *y(k)* is α,

for *k=3* it will be

IF *y(2)* is A_1 AND *y(1)* is A_2 AND *y(0)* is A_3 AND *u(2)* is A_4 AND *u(1)* is A_5 AND *u(0)* is A_6 THEN *y(3)* is α

Corresponding to the first line of the data matrix made of

y(2)	*y(1)*	*y(0)*	*u(2)*	*u(1)*	*u(0)*	*y(3)*

Then, for *k*=4,5, …, the following lines will be

y(3)	*y(2)*	*y(1)*	*u(3)*	*u(2)*	*u(1)*	*y(4)*
y(4)	*y(3)*	*y(2)*	*u(4)*	*u(3)*	*u(2)*	*y(5)*
…	…	…	…			…

Paying attention to this shape, the columns of the matrix are easily constructed:

- the first column is the output time series shifted two lines upwards,
- the second column is the output time series shifted one line upwards,
- the third column is the output time series as obtained in the simulation,
- the fourth column is the input time series shifted two lines upwards,
- the fifth column is the input time series shifted one line upwards,
- the sixth column is the input time series as obtained in the simulation.
- the seventh column (the target) is the output time series shifted thrice upwards

(this type of construction is what the *preparets* function makes in training dynamic neural networks).

After building the data matrix, clustering is made (eventually) with the support of the GUI

> findcluster

Allowing to choose between subtractive and fuzzy c-means, and to visualize the data in two dimensions (chosen among the seven possible). The *k-means* is made by the function *kmeans* (see >*help kmeans*).

The parameters of the methods (finally our degrees of freedom in controlling the clustering) are introduced in the respective window (or as arguments for those that prefer the command line).

For the subtractive clustering:

The `options` vector can be used for specifying clustering algorithm parameters to override the default values. These components of the vector `options` are specified as follows:

- `options(1) = quashFactor`: This factor is used to multiply the radii values that determine the neighbourhood of a cluster centre, so as to quash the potential for outlying points to be considered as part of that cluster. (default: 1.25)
- `options(2) = acceptRatio`: This factor sets the potential, as a fraction of the potential of the first cluster centre, above which another data point is accepted as a cluster centre. (default: 0.5)

- `options(3) = rejectRatio`: This factor sets the potential, as a fraction of the potential of the first cluster centre, below which a data point is rejected as a cluster centre. (default: 0.15)

For the fcm:

- `options(1)`: exponent for the partition matrix U (default: 2.0)
- `options(2)`: maximum number of iterations (default: 100)
- `options(3)`: minimum amount of improvement (default: 1e-5), unless stops.

The clustering methods give us the centres of the clusters. With them the fuzzy sets for the antecedents and the constants for consequents can be defined. ANFIS makes this automatically, with the subtractive clustering. The rules can be seen in the Anfis GUI, or exporting the *fis* structure created by ANFIS to the fuzzyLogicDesigner(or >fuzzy) used in part A.

Be careful in defining appropriately the training and testing sets. One and the other must be part of the temporal series respecting the temporal order. For example, take the first 70% lines of the data matrix for training and the remaining 30% for testing.

To optimize the *fis* constructed using another clustering technique, the following can be made:

a) Manually construct a *fis* (form the previous data matrix)
   a. clustering, suppose it gives 5 centres. Each centre will give a rule with 6 antecedents and one consequent. The antecedents are centred in the coordinates of the six dimensions and the consequent is the seventh dimension (case of zero order TSK).
   b. in Anfis (or in fuzzyLogicDesigner) design the membership functions with these centres issued from clustering, using the menu Edit>membership functions, and exporting afterwards the structure as the file *myManualFis.fis*

b) Load the *myManualFis.fis* in the anfisedit GUI (Generate fis > load from file).

Alternatively to the manual procedure, you may use the genfis (generate fis) function:
"FIS = genfis(XIN,XOUT,OPTIONS) creates an FIS using the specified OPTIONS returned by GENFISOPTIONS function. You can specify three different sets of option values using GENFISOPTIONS function as follows:
OPTIONS = GENFISOPTIONS('GRIDPARTITION',NAME1,VALUE1,...) creates options for grid partition using the specified parameter NAME/VALUE
pairs. OPTIONS= GENFISOPTIONS('SUBTRACTIVECLUSTERING',NAME1, VALUE1,...) creates options for subtractive clustering method with the specified parameter NAME/VALUE pairs.
OPTIONS = GENFISOPTIONS('FCMCLUSTERING',NAME1,VALUE1,...) creates options for FCM clustering method with the specified parameter NAME/VALUE pairs.
For more information on creating option sets, see GENFISOPTIONS function." (Matlab > help genfis)

c) Optimize (Train FIS) by the hybrid or retropropagation method.
d) Save the *myManualOptimizedFis.fis*

   For those who prefer the command line:

[FIS,ERROR] = anfis(TRNDATA) tunes the FIS parameters using the input/output training data stored in TRNDATA. For an FIS with N inputs, TRNDATA is a matrix with N+1 columns where the first N columns contain data for each FIS input and the last column contains the output data. ERROR is the array of root mean square training errors (difference between the FIS output and the training data output) at each epoch. anfis uses GENFIS1 to create a default FIS that is used as the starting point for anfis training.

[FIS,ERROR] = anfis(TRNDATA,INITFIS) uses the FIS structure, INITFIS, as the starting point for anfis training.

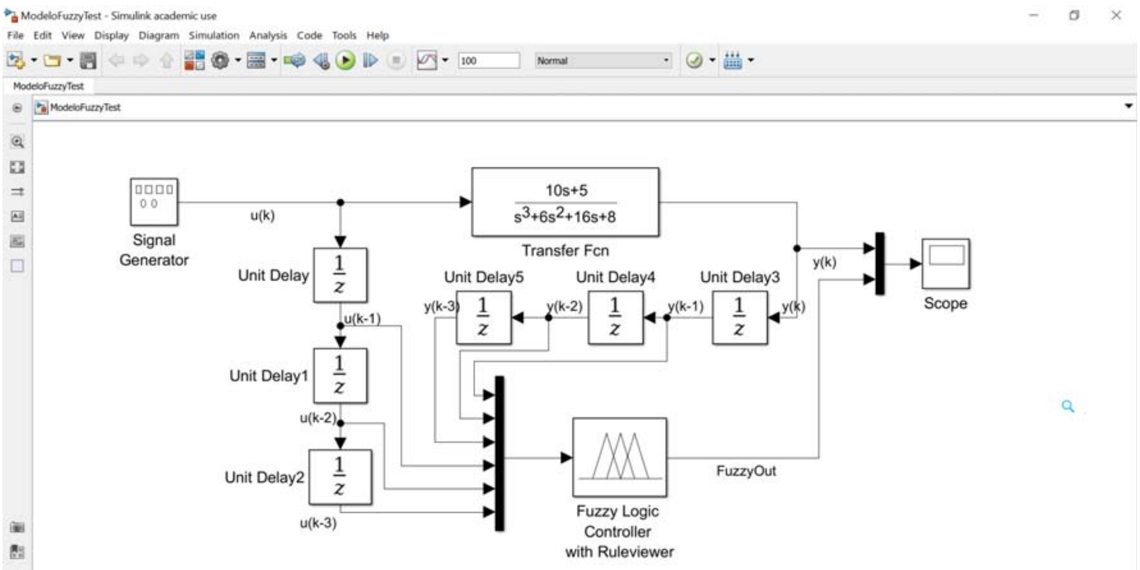
To compute the performance of *myManualOptimizedFis*, the GUI can be used selecting Load data>Testing, or, alternatively, run *evalfis* for the test data. Then compute the output, compare with the target, compute the squared error in each instant, sum them and divide by the number of instants and the mse is obtained.

Two clustering techniques should be used and compared, from the point of view of the precision of the obtained fuzzy model.

The following block diagram can be built. In the FuzzyController write *myManualOptimizedFis*. The sample time of the delays must be the same for all.

In general, the **best value for the sampling time** can be computed as follows.

1st- compute the roots of the denominator of the s- transfer function (the system poles p_i) with > roots(den).
2nd- the time constants τ_i of the system are the inverses of the real part of the poles, $\tau_i = -1/p_i$
3rd- the "Sample time" must be less than the lowest time constant, for example one quarter (1/4) of it. This leads to small integration errors in Simulink. In other way, the simulation may evolve to very big values because of the cumulative effect of errors. In Simulink you may configure the integration method (the solver selection) in Simulation>Configuration parameters>Solver selection>Type:Fixed-step>Solver>Ode4 (Runge Kutta), Solver details: Fixed-step size: Ts (your sample time value).



Applying a square wave or a sinus, or a saw tooth, the scope plots the curves. Note that the order of the inputs in the multiplexer must be the same of the antecedents of the rules.

Note that the fuzzy block here is not a controller, but a model of the process. You can change its name in the diagram. It is a *fis* that you can name at your convenience. To rename it double click in the name and change it.

Report Part B (in the same file as Part A)

1- File describing briefly the obtained neuro-fuzzy systems and their performance, namely the membership functions obtained for each of the six inputs of the fuzzy system, plotted in the GUI fuzzy.

2- files *.*fis* with the fuzzy models optimized by Anfis.

3- Files *.m needed to perform the training and test.

4- The built Simulink diagram (if you made it).

Note that all delivered reports must be a single .zip or .rar file with the name AC2022FuzzyPLxGy and the report itself must have the same name.

You do not need to repeat in the report what is written in the assignment or in the slides. The principle of *parsimonia* (Chapter 2) will be applied. Concentrate on your critical thinking and original contributions.

Have a nice fuzzy work. Coimbra, 22 November 2023