University of Coimbra

Doctoral Program in Information Science and Technology

Real Time Learning in Intelligent Systems

Assignment #4 - Process Modelling with Neuro-Fuzzy Systems

Joaquim Pedro Bento Gonçalves Pratas Leitão - 2011150072

May 31, 2017

1   Introduction

Dynamic systems are often described and modelled by different equations, where the outputs in a given time instant (say k) are conditioned by previous outputs (in instants $k - 1$, $k - 2$, ...) and inputs (in instants k, $k - 1$, $k - 2$, ...). A system is also said to have *inertia* when the inputs in a given instant only affect its outputs in posterior instants (that is, input at instant k can only influence the output at instants $k + 1$, $k + 2$, ...).

The current assignment proposes the development of a Sugeno-type Neuro-Fuzzy System (NFS) to model the dynamics of a given process or system with inertia. NFSs of this nature are characterised by mapping their input space to an output space using a series of fuzzy *if-then* rules. In the particular case of Sugeno NFSs the output of each rule is written as a linear combination of the input variables. In simpler Sugeno systems, this linear combination consists in a constant value.

By collecting data containing pairs of the system's input and corresponding output values the rules that define the NFS can be learned in such a way that they describe the system's behaviour.

In other words, exploring collected data from the system the mentioned fuzzy rules can be learned, resulting in the development of a Sugeno NFS that models the dynamics of the desired process or system, as it is the objective of this assignment.

The remainder of this document is organised as follows: Section 2 presents the system to be modelled in this work; Section 3 describes the methodology followed throughout the project; Sections 4 and 5 cover the main steps of the work, presented in the Methodology section. Finally, Section 6 concludes the document.

2   Modelled System

The system to be modelled in this work can be defined by the following transfer function:

$$G(s) = \frac{2}{s^3 + 5s^2 + 6.75s + 2.25}$$

Since the provided transfer function features 3 poles and no zeros - therefore it has more poles than zeros - the presented system is a third-order system. It can be proved that the provided system has memory of 3 instants in both the inputs and outputs and *inertia* (meaning that inputs at a given time instant only influence future outputs). In this sense, the output of the system at any time instant k can be written as:

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

where $y(k)$ represents the system's output at instant k and $u(k)$ represents the system's input at instant k.

3   Methodology

The project and development of a Neuro-Fuzzy System can be a complex and extensive task. Nevertheless, in the scope of the current work, the following two main steps will be considered:

1. LEARNING STAGE, where a model of the system is learned using data collected from the system. This stage comprises three main sub-tasks:

   - *Data collection*, where pairs of system's *input-output* data are collected, which will be used as a labelled dataset during the learning. Collecting a good set of *input-output* data is critical in the final outcome. These data should be representative of the dynamics of the system, making the input and output vary over the entire permissible domain. A good technique is to use a random input sequence that forces the output to move through the entire space.

   - *Fuzzy rules initialisation.* An initial estimate for the NFS's rules is obtained by applying a *clustering* algorithm to the labelled dataset collected in the previous point. Common algorithms used at this stage include subtractive, c-means and fuzzy c-means clustering.

   - *Fuzzy rules optimisation.* A further optimisation of the fuzzy rules computed in the previous point is usually performed, aiming to reduce the modelling error. An *ANFIS* structure is commonly used along with either a *backpropagation* or *hybrid* optimisation method (although other alternatives maybe considered at this point).

2. ASSESSMENT STAGE, where different input signals are provided to both the developed model and the real system with the objective of comparing the real and simulated outputs. The ultimate goal of this task is to certify that the developed model of the system can accurately describe its behaviour.

4   Learning Stage

As presented in the previous section, the goal of the *Learning Stage* is to train a Neuro-Fuzzy System to model the target system's behaviour based on pairs of *input-output* signals collected from the target system.

Therefore the first step of this stage is related with data collection. Since no access to the "real", *physical* system is available data will be collected with *Simulink*, a graphical programming and simulation environment available with the *MATLAB* software.

4.1   Data Collection

Usually, the process of collecting data from a given system assumes that the collection is made at a given frequency, meaning that samples are collected with a given interval. It is recommended that the sampling interval (also referred to as discretization interval) be less than the inverse of the smaller pole of the system. As such, a sampling interval equal to one-third of the inverse of the smaller pole of the system was used.

After setting the sampling interval, a discrete version of the system's transfer function must be computed. Resorting to the *MATLAB* software environment, such computation can be performed with the function *c2dm*. The following discrete transfer function was obtained:

$$G(z) = \frac{0.1079z^2 + 0.1413z + 0.009003}{z^3 - 0.8794z^2 + 0.1766z - 0.006738}$$

In this sense, the block diagram presented in Figure 1 was created. A random number generator is used to produce random inputs to the system to which the corresponding outputs are obtained. Both the generated inputs and outputs are stored in memory. The system dynamics are simulated during a total simulation time of 500 instants. The generated data was further divided into a *training* and *testing* dataset, with a $70 - 30$ division.
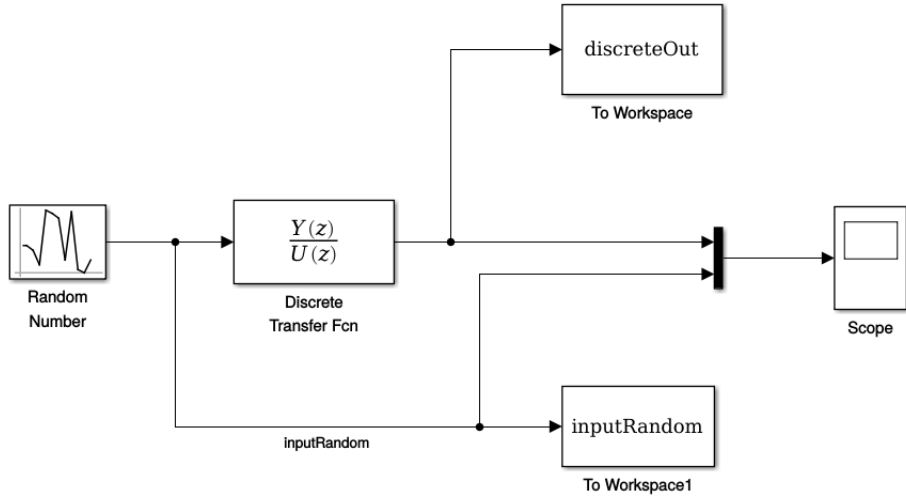
Figure 1: Block diagram developed in Simulink to generate training and testing data.

The generated input signal and correspondent system's output is presented in Figure 2:
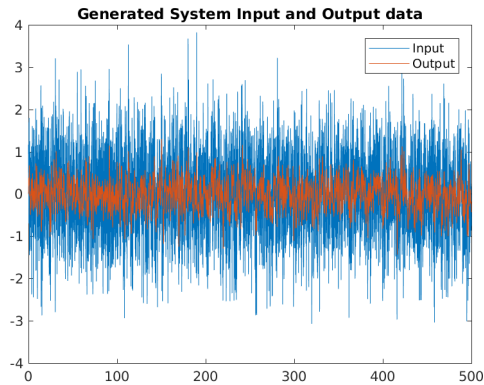
Figure 2: Generated input signal and system output.

4.2 Fuzzy Rules Initialisation

Having generated input and output samples representative of the dynamics of the system being modelled, the actual learning process can be started. As mentioned in Section 3, in an initial step, estimations for the fuzzy rules are produced as a result of a clustering process of the collected data. Two clustering algorithms were studied at this point, featuring the following parameters:

- SUBTRACTIVE CLUSTERING, where a *cluster influence range* of 0.5 was defined; s *squash factor* of 1.25 was considered, along with and an accept ratio of 0.5 and reject ratio of 0.15.

- FUZZY C-MEANS CLUSTERING, considering an exponent for the fuzzy partition matrix of 2.0, maximum number of iterations of 100, minimum improvement in objective function between two consecutive iterations of 10^{-5} and the number of clusters being automatically estimated using subtractive clustering.

In terms of implementation, the initial estimation of the fuzzy rules was performed using the *MATLAB's genfis* function, properly specifying the training data, the type of clustering to be performed and the respective parameters. The last two arguments (type of clustering and clustering parameters) were defined by creating a proper *genfisOptions* object.

The remainder of the current subsection presents a detailed view of the fuzzy inference systems obtained after the initial rules estimation.

Table 1: Fuzzy Inference Systems obtained after the initial rules estimation.

| Cluster Technique	AND Method	OR Method	Order
Subtractive	Product	Probabilistic OR of fuzzified input values	1
Fuzzy C-Means	Product	Probabilistic OR of fuzzified input values	1

Table 2: Input Membership Functions - Subtractive Clustering.

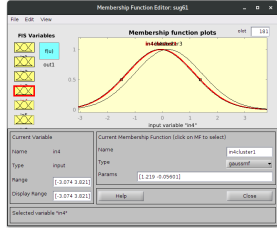| Input Number	Membership Function Type	Parameters
in1	Gaussian Curve (gaussmf)	$\sigma = [0.1095; 0.1095; 0.1095]$ $C = [-0.0221; 0.1294; -0.1326]$
in2	Gaussian Curve (gaussmf)	$\sigma = [0.1095; 0.1095; 0.1095]$ $C = [-0.0242; 0.1236; -0.1426]$
in3	Gaussian Curve (gaussmf)	$\sigma = [0.1095; 0.1095; 0.1095]$ $C = [-0.0276; 0.1155; -0.1513]$
in4	Gaussian Curve (gaussmf)	$\sigma = [1.2188; 1.2188; 1.2188]$ $C = [-0.056; -0.0963; 0.2091]$
in5	Gaussian Curve (gaussmf)	$\sigma = [1.2188; 1.2188; 1.2188]$ $C = [0.1841; -0.2404; -0.0825]$
in6	Gaussian Curve (gaussmf)	$\sigma = [1.2188; 1.2188; 1.2188]$ $C = [0.0674; -0.021; -0.0921]$
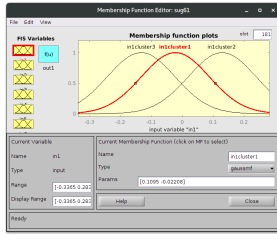
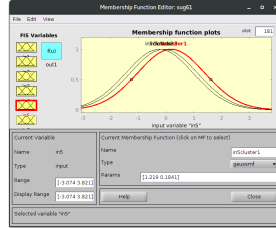(a) Membership Functions for input $in1$
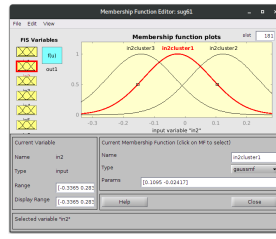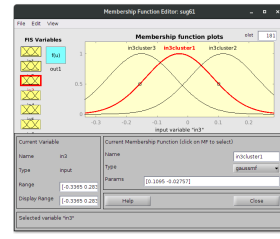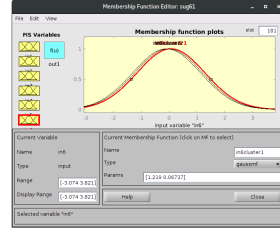
(b) Membership Functions for input $in2$

(c) Membership Functions for input $in3$

(d) Membership Functions for input $in4$
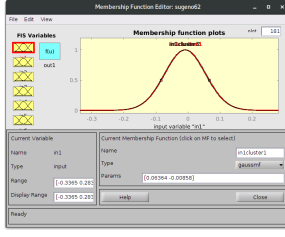
(e) Membership Functions for input $in5$

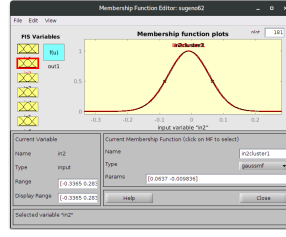(f) Membership Functions for input $in6$

Figure 3: Input Membership Functions for the initial rules obtained with the Subtractive Clustering method.

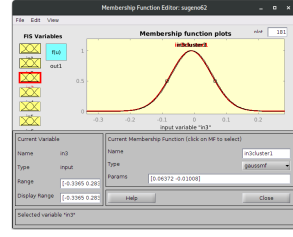Table 3: Input Membership Functions - Fuzzy C-Means Clustering.

| Input Number	Membership Function Type	Parameters
in1	Gaussian Curve (gaussmf)	$\sigma = [0.0636; 0.0638; 0.0636]$ $C = [-0.0086; -0.0071; -0.0108]$
in2	Gaussian Curve (gaussmf)	$\sigma = [0.0637; 0.0637; 0.0636]$ $C = [-0.0098; -0.0069; -0.0099]$
in3	Gaussian Curve (gaussmf)	$\sigma = [0.0637; 0.0637; 0.0635]$ $C = [-0.0101; -0.0069; -0.0097]$
in4	Gaussian Curve (gaussmf)	$\sigma = [0.5497; 0.5851; 0.5863]$ $C = [-0.0334; -0.4924; 0.4953]$
in5	Gaussian Curve (gaussmf)	$\sigma = [0.5592; 0.5909; 0.5669]$ $C = [-0.2425; 0.5136; -0.3109]$
in6	Gaussian Curve (gaussmf)	$\sigma = [0.6152; 0.5591; 0.5746]$ $C = [0.6764; -0.2763; -0.4373]$

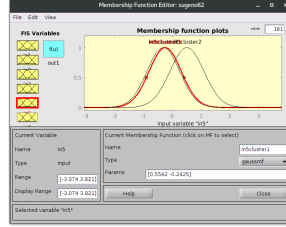(a) Membership Functions for input $in1$

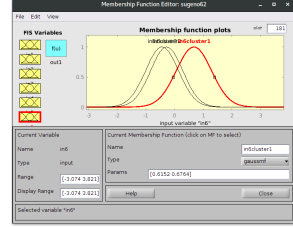(b) Membership Functions for input $in2$

(c) Membership Functions for input $in3$
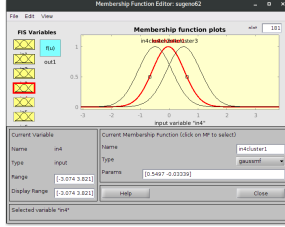
(d) Membership Functions for input $in4$

(e) Membership Functions for input $in5$

(f) Membership Functions for input $in6$

Figure 4: Input Membership Functions for the initial rules obtained with the Fuzzy C-Means Clustering method.

4.3   Fuzzy Rules Optimisation

In the final step of the learning stage an ANFIS structure is used to optimise the estimations for the fuzzy rules obtained in the previous subsection. Two optimisation methods were considered at this point: the well-known *Backpropagation* and an *Hybrid* method.

In terms of implementation, the optimisation of the inference rules was performed using *MATLAB*'s *anfis* function, properly specifying the training data, the type of optimisation method and the number of epochs[1]. The last two arguments (type of optimisation and the number of epochs) were defined by creating a proper *anfisOptions* object.

Analysing the input membership functions of the optimised Neuro-Fuzzy Systems (NFS) no changes were registered in the NFSs where the backpropagation optimisation method was used ; however, visible changes in the mean and deviation of the gaussians were registered when the hybrid optimisation method was used along with both fuzzy c-means and subtractive clustering. Tables 4 and 5 presents the new input membership functions.
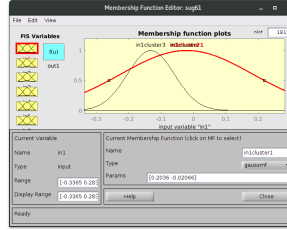
---

[1]A value of 200 was defined for the epoch number.

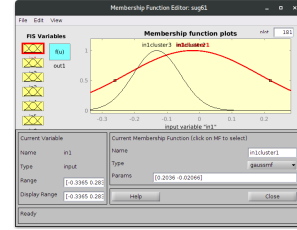Table 4: Final Input Membership Functions - Subtractive Clustering and Hybrid Optimisation Method.

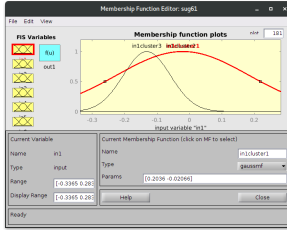| Input Number	Membership Function Type	Parameters
in1	Gaussian Curve (gaussmf)	$\sigma = [0.2036; 6.5753e - 05; 0.0723]$ $C = [-0.0207; 0.2147; -0.1315]$
in2	Gaussian Curve (gaussmf)	$\sigma = [0.2036; 0.0641; 0.0702]$ $C = [-0.0219; 0.1512; -0.1613]$
in3	Gaussian Curve (gaussmf)	$\sigma = [0.2032; 0.0692; -2.2305e - 05]$ $C = [-0.0237; 0.1266; -0.2394]$
in4	Gaussian Curve (gaussmf)	$\sigma = [1.2305; 1.2134; 1.2130]$ $C = [-0.0553; -0.0965; 0.2101]$
in5	Gaussian Curve (gaussmf)	$\sigma = [1.2313; 1.2134; 1.2135]$ $C = [0.1815; -0.2415; -0.0827]$
in6	Gaussian Curve (gaussmf)	$\sigma = [1.2294; 1.2135; 1.2138]$ $C = [0.0644; -0.0212; -0.0924]$
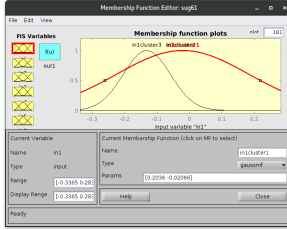
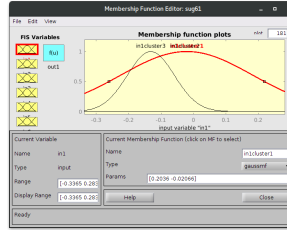(a) Membership Functions for input $in1$

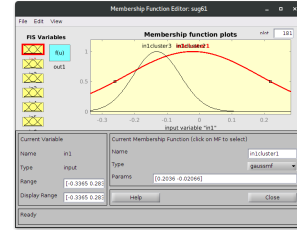(b) Membership Functions for input $in2$

(c) Membership Functions for input $in3$

(d) Membership Functions for input $in4$

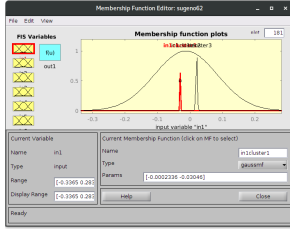(e) Membership Functions for input $in5$
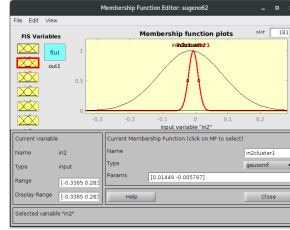

(f) Membership Functions for input $in6$

Figure 5: Input Membership Functions after Hybrid optimisation when Subtractive Clustering method was used in the rules initialisation.

Table 5: Final Input Membership Functions - Fuzzy C-Means Clustering and Hybrid Optimisation Method.

| Input Number	Membership Function Type	Parameters
in1	Gaussian Curve (gaussmf)	$\sigma = [-2.3364e - 04; 0.0964; -0.0021]$ $C = [-0.0305; -0.0128; 0.0204]$
in2	Gaussian Curve (gaussmf)	$\sigma = [0.0145; 0.0966; 2.1969e - 04]$ $C = [-0.0058; -0.0133; -0.0145]$
in3	Gaussian Curve (gaussmf)	$\sigma = [-1.5273e - 04; 0.0969; 0.0059]$ $C = [-0.0029; -0.014; -0.0093]$
in4	Gaussian Curve (gaussmf)	$\sigma = [0.5424; 0.6011; 0.5833]$ $C = [-0.0328; -0.4880; 0.4956]$
in5	Gaussian Curve (gaussmf)	$\sigma = [0.5521; 0.6047; 0.5634]$ $C = [-0.242; 0.5106; -0.3113]$
in6	Gaussian Curve (gaussmf)	$\sigma = [0.6123; 0.5757; 0.5718]$ $C = [0.6761; -0.2744; -0.4362]$

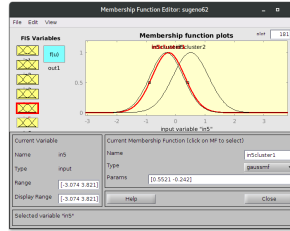

(a) Membership Functions for input $in1$
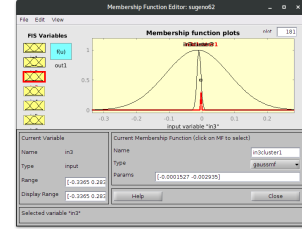
(b) Membership Functions for input $in2$

(c) Membership Functions for input $in3$
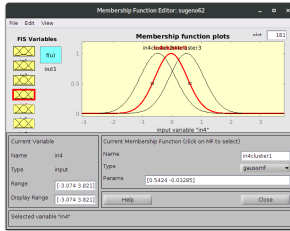
(d) Membership Functions for input $in4$

(e) Membership Functions for input $in5$
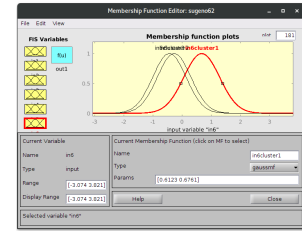
(f) Membership Functions for input $in6$

Figure 6: Input Membership Functions after Hybrid optimisation when the Fuzzy C-Means Clustering method was used in the rules initialisation.

Once the optimisation was complete, the performance of the final Neuro-Fuzzy Inference System was assessed using the test data. Table 6 presents a summary of the results of the optimisation process. The error metric used to compute both the training and testing error was the *Root Mean Squared Error* (RMSE).

Table 6: Final results of the optimisation process.

Cluster Technique	Optimisation Type	Minimum Training Error	Test Error	AND Method	OR Method	Number Rules	Order
Subtractive	Backpropagation	9.1984e-17	0.0053	prod	probor	3	1
Subtractive	Hybrid	1.5467e-06	8.9969e-04	prod	probor	3	1
Fuzzy C-Means	Backpropagation	2.1481e-16	0.0044	prod	probor	3	1
Fuzzy C-Means	Hybrid	1.5467e-06	1.5086e-06	prod	probor	3	1

5   Assessment Stage

Having developed and trained several NFSs to model the behaviour of the system presented in Section 2 more diverse and intensive tests must be performed. With the goal of determining how close the developed NFSs were from the actual system a *Simulink* model was developed and is presented in Figure 7.
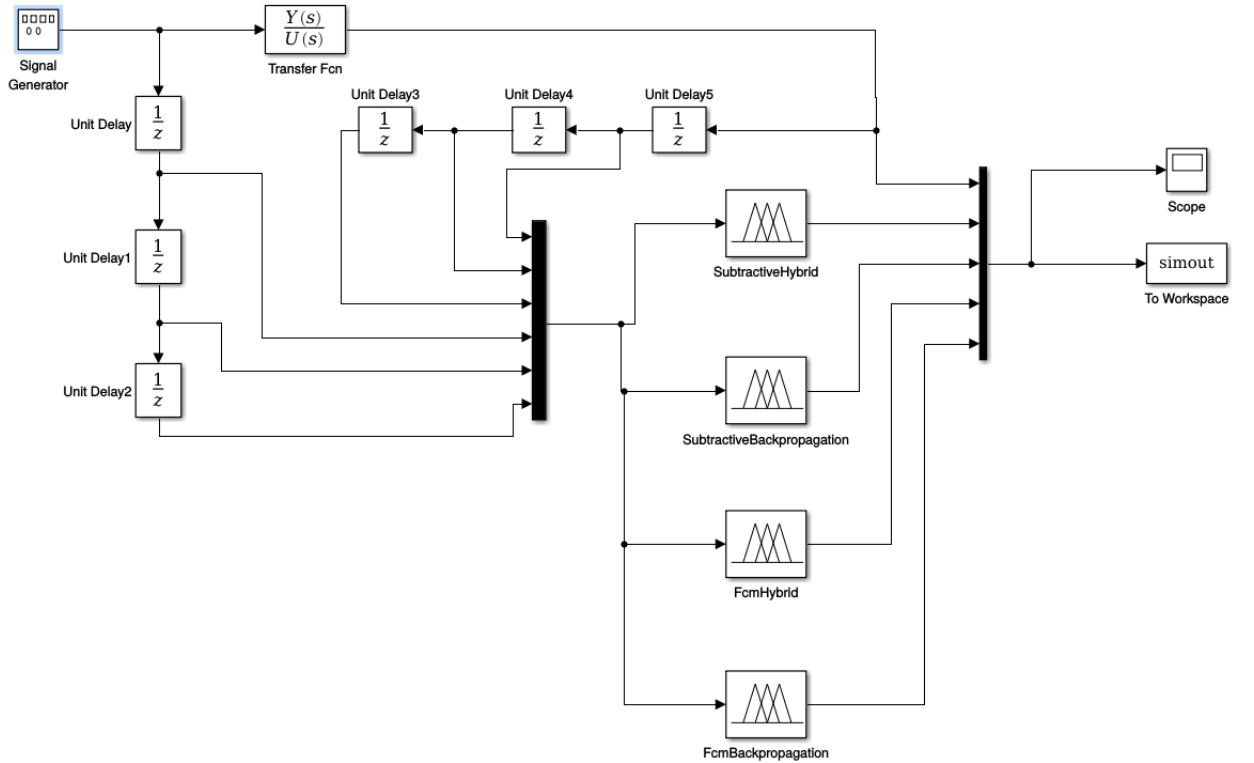

Figure 7: Block diagram developed in Simulink to assess how close the developed NFSs were to the actual system to be modelled.

In this exercise, a signal generator produces an input signal that is provided to the system being modelled (contained in the *Transfer Func* block, described by the transfer function presented in Section 2) and to each one of the four NFSs developed in the previous stage. The five computed outputs were then inspected and plotted in the *Scope* block.

Functions presented in the listing that follows were used in the *Signal Generator* block to produce the input signals. Table 7 presents the *Root Mean Squared* (rmse) errors registered with each developed Neuro-Fuzzy System, for each input wave.

- *Sawtooth* wave, with an amplitude of 1, units defined in *Hertz* and a frequency set to the sampling interval whose computation was described in Section 4.1.

- *Sine* wave, with an amplitude of 1, units defined in *Hertz* and a frequency set to the sampling interval whose computation was described in Section 4.1.

- *Square* wave, with an amplitude of 1, units defined in *Hertz* and a frequency set to the sampling interval whose computation was described in Section 4.1.

Table 7: Root Mean Squared Errors of the different Neuro-Fuzzy Systems for the three input waves.

Wave Type	Cluster Technique	Optimisation Type	RMSE
Sawtooth	Subtractive	Hybrid	3.0734e-04
Sawtooth	Subtractive	Backpropagation	0.0051
Sawtooth	Fuzzy C-Means	Hybrid	3.0734e-04
Sawtooth	Fuzzy C-Means	Backpropagation	0.0031
Sine	Subtractive	Hybrid	8.4137e-04
Sine	Subtractive	Backpropagation	0.0056
Sine	Fuzzy C-Means	Hybrid	5.7272e-05
Sine	Fuzzy C-Means	Backpropagation	0.0041
Square	Subtractive	Hybrid	3.4867e-04
Square	Subtractive	Backpropagation	0.0059
Square	Fuzzy C-Means	Hybrid	3.4867e-04
Square	Fuzzy C-Means	Backpropagation	0.005

Analysing the results presented in Table 7 considerably low *rmse* error values were registered for the different wave types and trained NFSs. Such results suggest that the NFSs can accurately reproduce the dynamics of the system being modelled. Indeed, the plots presented in Figures 8, 9, 10 support these statements.

The smallest *rmse* value was registered for the *Sine* wave, using an hybrid optimisation method and initial cluster estimations with Fuzzy C-Means. Overall, hybrid optimisation lead to errors about one order of magnitude lower than performing optimisation with the backpropagation method.
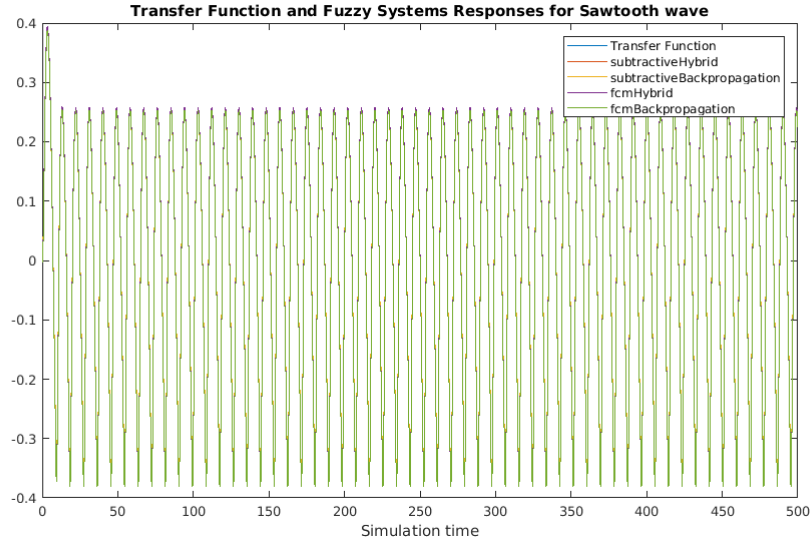
Figure 8: Outputs of the system being modelled and the four NFSs developed when using the *Sawtooth* wave to produce the inputs.
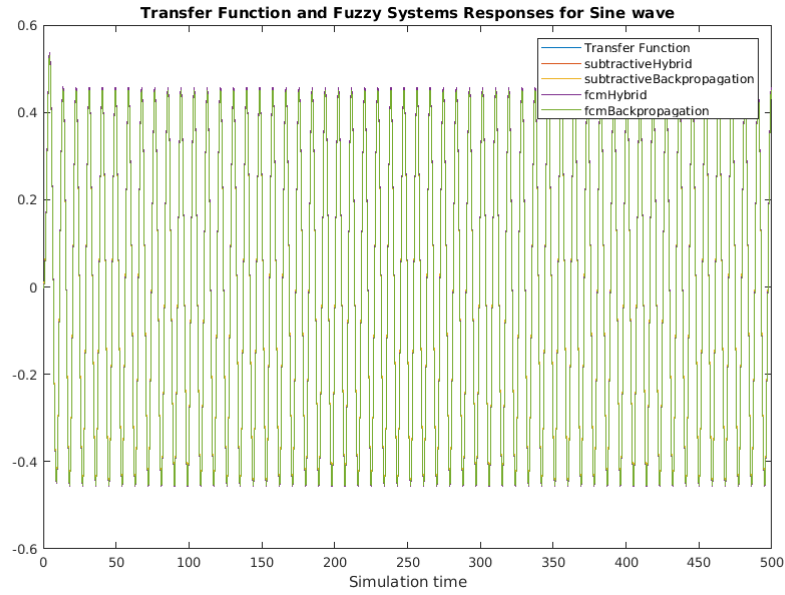

Figure 9: Outputs of the system being modelled and the four NFSs developed when using the *Sine* wave to produce the inputs.
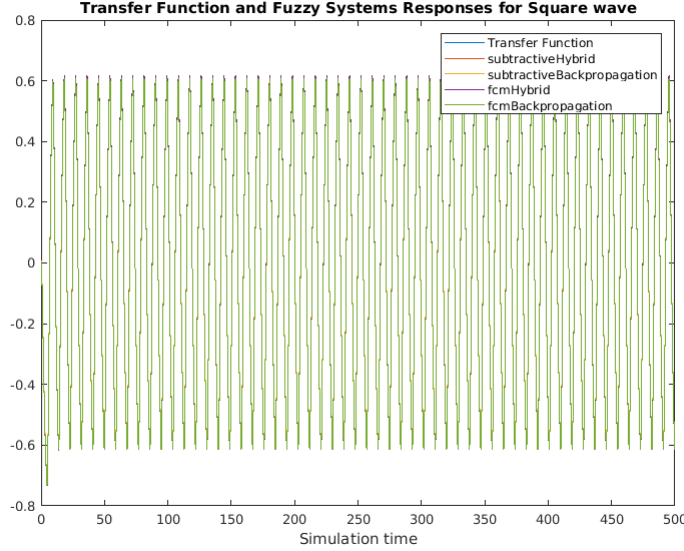
Figure 10: Outputs of the system being modelled and the four NFSs developed when using the *Square* wave to produce the inputs.

6 Conclusion

The current work focused on the task of modelling dynamic processes by means of Neuro-Fuzzy Systems. In the course of this work, such fuzzy logic controllers were projected and developed, based on the dynamics of a given system, described by a transfer function.

In an initial stage, a random input signal was provided to the "real" system (that is, the one to be modelled) and its corresponding outputs were recorded. A dataset containing tuples with input signals and the respective system outputs was created to be used in the development of the Neuro-Fuzzy Systems. This development consisted in two main stages:

At first, estimates to the fuzzy rules that govern the operation of these systems were obtained by means of a clustering technique - two distinct clustering algorithms were considered: *Subtractive* and *Fuzzy C-Means* clustering.

In a second step, an optimisation of the initial rules estimates was performed using an *ANFIS* structure. Two optimisation methods were also considered at this point - *Back-propagation* and an *Hybrid* optimisation method.

After the training, an initial performance assessment was conducted with part of the data initially collected. As presented in Table 7, the use of the *Backpropagation* optimisation technique registered the lowest errors during training; however substantially worst results were registered during testing, suggesting a lack of generalisation for this optimisation method. On the other hand, the *Hybrid* optimisation method was still capable of producing acceptable results in training which remained more stable during testing, suggesting a more general (and therefore desirable) model.

Based on the information mentioned, models developed using the *Hybrid* optimisation method could be seen as good candidates to model the dynamics of the system in question. Indeed, further tests conducted with different input signals support such suggestions: using

either a *Sawtooth, Square* or *Sine* wave the use of *Hybrid* optimisation always registered lower errors than when using *Backpropagation.*