4

Examples

In this section we will apply our method to different illustrative examples in order to show its performance in comparison with other techniques. All of them have been used previously in many articles and can be considered as benchmarks. Apart from being useful to compare the different alternatives, each one will facilitate the understanding of different qualities which we also want to emphasize.

The first example is a nonlinear static function often used in the literature of fuzzy modeling [111, 58, 20] and will help us to review the whole method in detail.

The second example is about time series prediction. Here we will consider two popular time series, the Box and Jenkins' gas furnace data [7] cited in [113, 86, 122, 90, 110, 124, 111, 69, 118, 119, 58, 20] and also the Mackey-Glass chaotic time series [72] studied in [120, 47, 51, 18, 33]. Both of them are benchmarks for the prediction of dynamic systems. They are also useful to observe the trade-off between accuracy and intelligibility because in general they have been used to achieve models with high accuracy in spite of not being intelligible.

The third example is about fuzzy control. In this part we will study two popular examples of intelligent control able to operate like humans, the truck backer-upper control or even the truck and trailer [82, 83, 61, 97, 40, 62, 63, 120, 52, 103, 16] and the ball-beam control system. Here our method will be considered to show its capabilities when building an intelligible controller from input-output data and also to show how it can be applied to explain the control actions applied with a non-fuzzy controller.

Finally four short examples will be included: a real application of modeling about the electrical network maintenance cost [11]; a fuzzy model proposed in [111] in order to study the reproduction of the method and thus, to obtain the same model or at least a similar one; the simplification of FRBS

whose transfer function is quite linear; and finally a simple function in order to compare our method with probably the most popular one: ANFIS.

4.1 Criteria to evaluate the methodology

The criteria to evaluate the performance of the method must be according to the objectives of this work and consequently, they must evaluate the trade-off between accuracy and intelligibility. Furthermore, we must also analyze the computational cost because this is another objective of our research. Thus, in the following examples, we will do studies and make comparisons about these three aspects:

- Accuracy

  Accuracy is obviously defined in terms of the error, as the difference between the original samples and the output of the resulting model. Among the different possibilities for defining the overall error here we will consider the RMSE for being the most commonly used definition found in the literature and thus, it will help us to compare the performance of our models with other alternatives.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y_{model}(k) - y_{measured}(k)\right)^2} \tag{4.1}$$

  Nevertheless, recall that our preferred definition for the error is the NRMSE (Equation 3.42) and this is the parameter we consider in order to decide when the modeling process ends. We stop whether the overall error or the error of every linearized fuzzy curve, both defined with the NRMSE, reach the *desired error* value.

- Intelligibility

  We have explained the different criteria to be considered in order to obtain an intelligible fuzzy model and we have also explained how our method satisfies most of them: distinguishability, completeness, coverage, normality ... Thus, here we will just measure those which are more objective, the *number of linguistic labels*, which depend on the number of inputs, the number of sets and the number of rules.

  Furthermore we will detail the type of FRBS when we compare our method with some alternatives which use the Takagi-Sugeno's model.

- Computational cost

  In the examples where the computational cost should be compared, we have done all the measurements by using the same hardware and the same software. This allows us to compare the computational cost in terms of the elapsed time.

Nevertheless, we will not argue all these aspects in every example and only those with results which may be more significant will be commented in each case.

Furthermore and in order to clarify the results, most of the information will be provided by means of tables and by using graphics extracted directly from the implementation of the method with Matlab.

4.2 A nonlinear static function

4.2.1 Problem statement

Here we will consider the following double-input single-output function

$$y = \left(1 + x_1^{-2} + x_2^{-1.5}\right)^2 \qquad 1 \leq x_1, x_2 \leq 5 \tag{4.2}$$

plotted in figure 4.1. This function was proposed by M. Sugeno [111] and we will use exactly the same data which are given in table 4.1 in order to compare results.

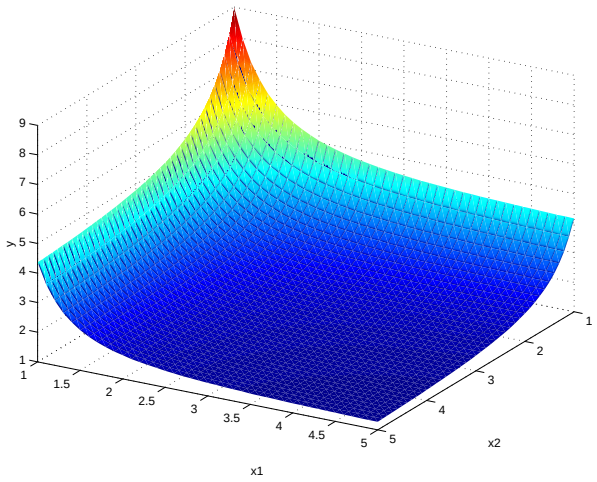


Figure 4.1: Original function proposed by M. Sugeno.

Data of the nonlinear function														
x_1	x_2	y	x_1	x_2	y	x_1	x_2	y	x_1	x_2	y	x_1	x_2	y
1.40	1.80	3.70	4.28	4.96	1.31	1.18	4.29	3.35	1.96	1.90	2.70	1.85	1.43	3.52
3.66	1.60	2.46	3.64	2.14	1.95	4.51	1.52	2.51	3.77	1.45	2.70	4.84	4.32	1.33
1.05	2.55	4.63	4.51	1.37	2.80	1.84	4.43	1.97	1.67	2.81	2.47	2.03	1.88	2.66
3.62	1.95	2.08	1.67	2.23	2.75	3.38	3.70	1.51	2.83	1.77	2.40	1.48	4.44	2.44
3.37	2.13	1.99	2.84	1.24	3.42	1.19	1.53	4.99	4.10	1.71	2.27	1.65	1.38	3.94
2.00	2.06	2.52	2.71	4.13	1.58	1.78	1.11	4.71	3.61	2.27	1.87	2.24	3.74	1.79
1.81	3.18	2.20	4.85	4.66	1.30	3.41	3.88	1.48	1.38	2.55	3.14	2.46	2.12	2.22
2.66	4.42	1.56	4.44	4.71	1.32	3.11	1.06	4.08	4.47	3.66	1.42	1.35	1.76	3.91
1.24	1.41	5.05	2.81	1.35	1.97	1.92	4.25	1.92	4.61	2.68	1.63	3.04	4.97	1.44
4.82	3.80	1.39	2.58	1.97	2.29	4.14	4.76	1.33	4.35	3.90	1.40	2.22	1.35	3.39

Table 4.1: Original samples proposed by M. Sugeno.

4.2.2 A whole case in detail

We will use this example in order to illustrate all the steps of the method in detail. Nevertheless, one should remember that the resulting model may change in every execution because we use a statistic of the optimal β which is computed from random partitions of the original samples. This occurs even with the same value for the *desired error* parameter.

For this reason, first we will show the steps of only one of these executions where the only adjustable parameter, the *desired error* (ε), was ε=10% and then we will analyze how the final models vary in different executions and thus, the robustness of the method.

Rounded values

At the beginning we compute the values which are necessary in order to round some numbers, basically the possible points of the universes of scope where the fuzzy sets may be placed. They are obtained by considering the range of the samples and the ε parameter.

In this example the range of the original variables is 3.80 for the input x_1 because it varies between 1.05 and 4.85, 3.91 for the input x_2 and 3.97 for the output y. With ε=10%, the numerical results of the three variables will be rounded close to multiples of 0.380 for x_1, equal to the 10% of 3.80, 0.391 for x_2 and 0.397 for the output. Nevertheless, as these values are in general poorly intelligible, they are finally rounded to multiples of 0.1.

Furthermore some of these values of the universes of scope will not be considered in order to reduce the computational cost, those without any original sample inside its neighborhood.

Observe in figure 4.2 how for the first input x_1 the values 3.9 and 5.0 will not be considered while for the second input x_2 the values 2.4, 3.0, 3.3, 3.4, 3.5 and 4.0 will also be ignored. The rest of the points are potentially possible points where the fuzzy sets can be placed.

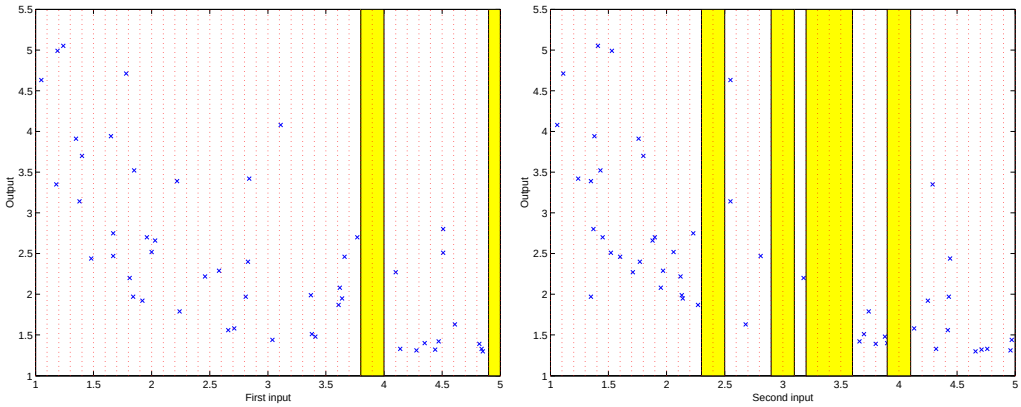


Figure 4.2: Original samples and possible points of the universes of scope.

Train set and test set for the optimal β

The next step consists in computing the optimal β for each input variable from which we will compute the optimal fuzzy curves. Remember that in fact we only obtain a statistic because the original samples must be grouped into a train set of samples and a test set of samples. Otherwise the trivial solution for the optimal β would be equal to zero.

Consequently, we make several partitions of the original samples from which we compute the mean of their optimal β until we assure this statistic with an error lower than 10% and a confidence interval higher than 90%. These two values are fixed as the *desired error* parameter (ε) for the first one and 100 minus the same parameter for the second one. Thus, in this case, in 90 of every 100 partitions the optimal β will be inside the range defined by the result $\pm 10\%$. A lower ε would give a more confident parameter.

These two sets of samples, the train set and the test set, are generated by taking randomly two samples close to each possible point of the universe of scope which have been computed in the former step, one for the train set and the other one for the test set. The points of the universe of scope without at least two samples inside its neighborhood are not considered.

Nevertheless, if less than half of these points do not have at least two samples inside its neighborhood they are temporarily reduced to the half by changing the rounded value to the double of its magnitude. In the current example this situation occurs for both input variables, as can be observed in the last figure 4.2.

Therefore, either the train set or the test set are generated by taking randomly two of the samples close to each point of the universe of scope rounded to 0.2. Figure 4.3 shows how the samples have been grouped for this purpose.

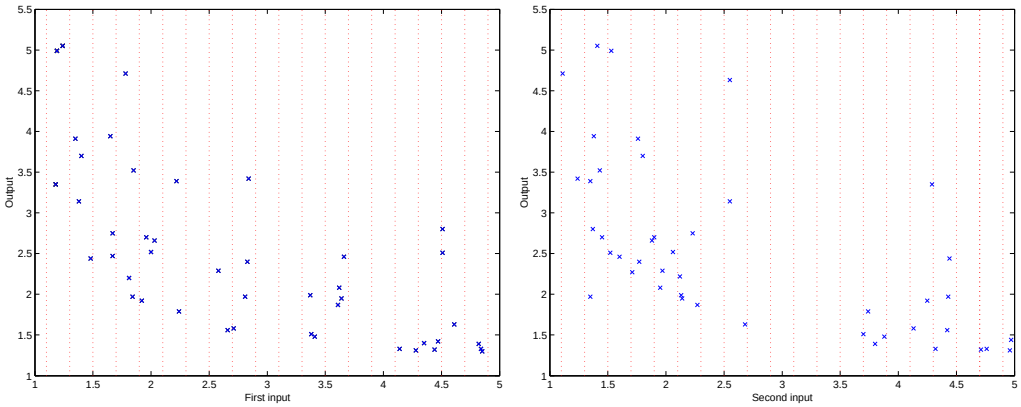
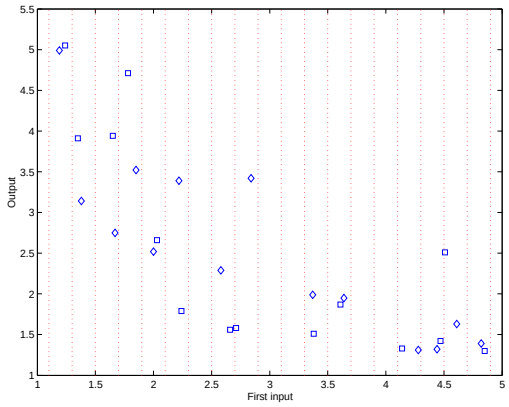


Figure 4.3: Groups of samples in order to generate the train set and test set.

Furthermore, observe how some of the original samples have been extracted because they do not have at least another sample inside its range, like for example $(x_1 = 3.11, y = 4.08)$ or $(x_2 = 3.18, y = 2.20)$.

Once we have these two random sets, we are ready to compute the optimal β of this partition. For instance we could consider the random samples shown in figure 4.4 for the first input x_1.



Train points													
1.24	1.35	1.65	1.78	2.03	2.24	2.66	2.71	3.38	3.61	4.14	4.47	4.51	4.85
5.05	3.91	3.94	4.71	2.66	1.79	1.56	1.58	1.51	1.87	1.33	1.42	2.51	1.30
Test points													
1.19	1.38	1.67	1.85	2.00	2.22	2.58	2.84	3.37	3.64	4.28	4.44	4.61	4.82
4.99	3.14	2.75	3.52	2.52	3.39	2.29	3.42	1.99	1.95	1.31	1.32	1.63	1.39

Figure 4.4: Random points for x_1.

We will show the next steps only for this variable. The same process would be done for the other one.

Boundaries of the optimal β

Once we have the train set and the test set, we are ready to compute the optimal β. Nevertheless, recall that we can know the boundaries between which the optimal β can be found in order to improve its search. If we consider the previous train set and test set, these boundaries are β_{min}=0.0417 and β_{max}=11.2746.

The first one results from the test point {4.44,1.32} by having its closest train points {4.47,1.42} and {4.51,2.51} at a distance equal to d_1=0.03 and d_2=0.07 respectively because this is the case with the minimum value for ${d_2}^2 - {d_1}^2$ and thus, $\beta_{min}=\sqrt{-\frac{0.07^2-0.03^2}{\ln(0.01\times 10)}} = 0.0417$.

The β_{max} results from the test point {1.19,4.99} by having its closest train point {1.24,5.05} and its farthest train point {4.85,1.30} at a distance equal to d_1=0.05 and d_∞=3.66 respectively because this is the case with the maximum value for ${d_\infty}^2 - {d_1}^2$ and thus, $\beta_{max}=\sqrt{-\frac{3.66^2-0.05^2}{\ln(0.01\times(100-10))}} = 11.2746$.

As the range between β_{min} and β_{max} is usually very large we compute the derivative of the square error $\frac{\partial\varepsilon}{\partial\beta}$ with some values for the β parameter inside this range in order to reduce the boundaries between which the optimal value is placed. For instance, if 3 points per decade are considered then a total amount of 9 values would be computed:

β	0.0417	0.0839	0.1690	0.3404	0.6855	1.3805	2.7801	5.5986	11.2746
$\partial\varepsilon/\partial\beta$	0.0278	0.0489	-0.2578	-0.6830	-1.2204	0.3026	1.9408	0.9268	0.2621

Therefore we conclude that there is a local minimum between 0.6855 and 1.3805 where the sign of the derivative changes from negative to positive. If all the derivatives would have shown the same sign then another group of train and test samples would have been generated.

Search of the optimal β with the bisection method

We finally use the bisection method in order to fit the optimal value with an error lower than the 10%. In this case only three iterations are necessary:

Iteration	Initial	1^{st}	2^{nd}	3^{rd}
$[\beta_{min},\beta_{max}]$	[0.6855,1.3805]	[1.0330,1.3805]	[1.2067,1.3805]	[1.2067,1.2936]
$\partial\varepsilon/\partial\beta$	[-1.2204,0.3026]	[-0.5156,0.3026]	[-0.1013,0.3026]	

Once we know that the optimal β is between 1.2067 and 1.2936, we can assure its value with an error lower than 10% and we conclude an optimal β equal to the mid point of this range, that is 1.2502. If we had found more than one local minimum we would have searched all of them in order to choose the one with the lowest error.

This process is repeated with different test and train sets of samples in order to compute the mean of their optimal values until we accomplish the required confidence interval and error.

Figure 4.5 shows the final results after computing 118 iterations for the first input and 124 iterations for the second one. In this case we obtained an optimal $\beta = 0.7774$ for the input x_1 and $\beta = 1.0306$ for the input x_2.

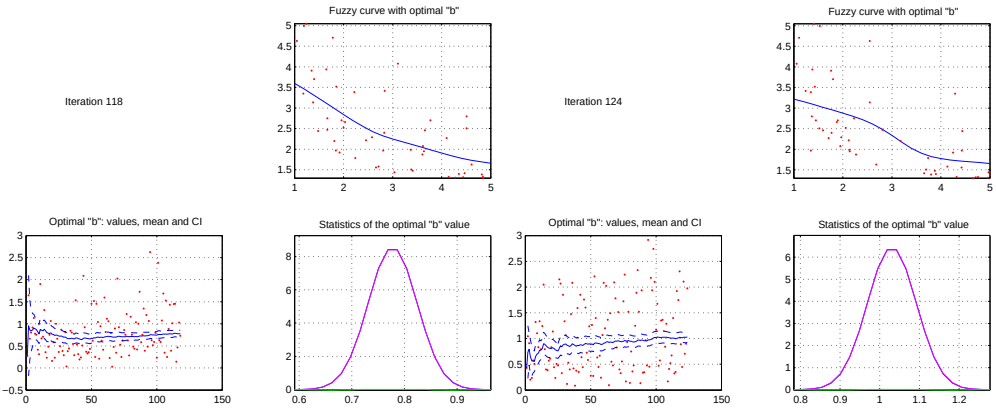


Figure 4.5: Statistics for β_{x_1} and β_{x_2}.

Optimal fuzzy curves

Once we have the optimal β for each input then we compute the fuzzy curves. For this purpose we only evaluate them in every point of the universes of scope where a fuzzy set can be placed, based on the original rounded values which were 0.1 for both inputs. Figure 4.6 shows these fuzzy curves.

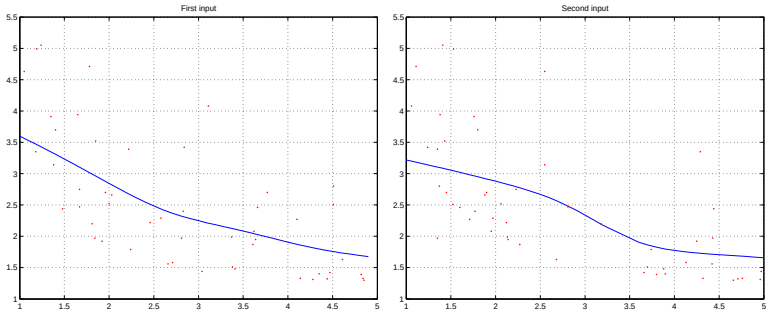


Figure 4.6: Fuzzy curves with optimal β.

Input partition based on the linearization of the fuzzy curves

Now we begin the hierarchical process of linearizing these fuzzy curves until the error due to the linearization is lower than the *desired error* parameter, in this case 10%. In the first iteration we only consider the boundaries of the universe of scope as fuzzy sets and thus, two fuzzy sets are considered for each input, placed at 1.0 and 4.9 for x_1 and 1.0 and 5.0 for x_2, as shows figure 4.7.

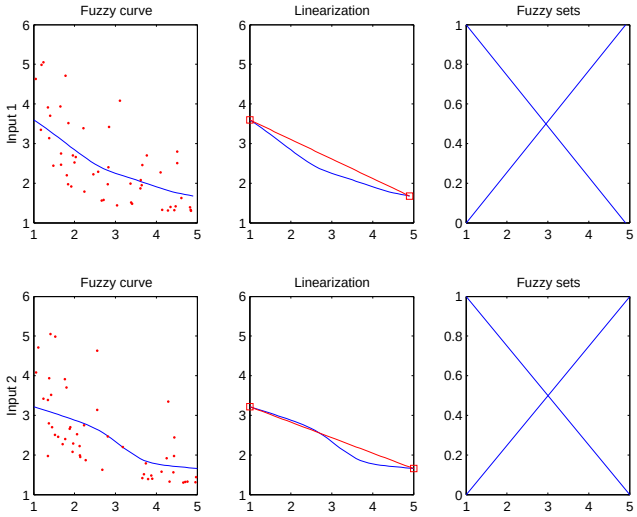


Figure 4.7: Initial iteration.

Possible output sets based on the Wang&Mendel's method

We only need to define the possible output sets of each possible rule in order to conclude the current model. Based on the method proposed by Wang&Mendel, for each possible rule we search the sample with the highest fuzzification value. In this case we have 4 possible rules:

Rule	Closest sample	Possible set
x_1 close to 1.0 and x_2 close to 1.0	{1.24,1.41,5.05}	5.05
x_1 close to 1.0 and x_2 close to 5.0	{1.18,4.29,3.35}	3.35
x_1 close to 4.9 and x_2 close to 1.0	{4.51,1.37,2.80}	2.80
x_1 close to 4.9 and x_2 close to 5.0	{4.85,4.66,1.30}	1.30

If there was a rule whose fuzzification values would have been equal to zero for all the samples, this rule would had been removed from the set of rules.

Clustering the output sets based on the Chiu's method

In order to improve the intelligibility of the model, the output sets are clustered based on the Chiu's method.

Nevertheless, when there are still few rules and obviously few possible output sets, the final output sets are in general the same which were initially proposed. This is what will happen here and during the following two iterations.

For this purpose, first we compute the possible points of the universe of scope where an output fuzzy set can be placed. As at the beginning we decided to round them to 0.1 we can consider four possible output sets placed at 1.3, 2.8, 3.4 and 5.0.

In this case the parameters used by the Chiu's method are $r_\alpha = r_\beta = \varepsilon\,(5.0 - 1.3) = 0.37$ and $\alpha = \beta = -\ln(\varepsilon)/{r_a}^2 = 16.82$.

First, we compute the potential P for each possible output set with equation 3.39. The possible sets placed at 2.8 and 3.4 show the highest value of P equal to 1.0023 and they are chosen as the initial clusters. We then compute the new potential P for each possible output set once subtracted a magnitude proportional to the former P and to its distance to the last clusters by considering the equation 3.40. Now the possible sets with the highest P are those placed at 1.3 and 5.0 with P equal to 1. Therefore, we choose both of them as new clusters. As the new potential P of all the possible output sets is lower than ε=10%, we conclude that the final possible output sets must be placed at 1.3, 2.8, 3.4 and 5.0.

Finally we assign to each possible rule the possible output set which is closest to it.

Computation of the NRMSE in order to decide if the process ends

Once we have the final model we compute its RMSE=0.9698. Nevertheless as its NRMSE=93.98% is higher than 10% and also the NRMSE of each linearized fuzzy curve ($NRMSE_{x_1}$=24.10%, $NRMSE_{x_2}$=13.40%) we must do another iteration by seeking the highest error in both linearizations in order to diminish it.

Search of the new input partition

As the highest error of the first input is 0.39 when x_1 is 2.7 and the highest error of the second input is 0.30 when x_2 is either 3.6, 3.7 or 3.8, we conclude that a new set should be placed for the input variable x_1 at 2.7. Figure 4.8 shows the model proposed for the second iteration.

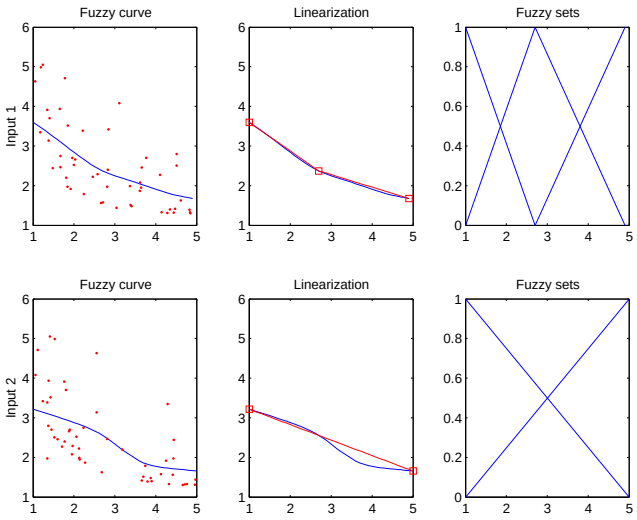


Figure 4.8: Second iteration.

Hierarchical process by repeating the same steps until a satisfactory model is achieved

We repeat the previous process. Therefore, first we look for the output values of the samples which adapt better to each new possible rule based on the Wang&Mendel's method. Here we must compute only two new rules because the other four possible rules are obviously the same than before:

Rule	Closest sample	Possible set
x_1 close to 2.7 and x_2 close to 1.0	{2.84,1.24,3.42}	3.42
x_1 close to 2.7 and x_2 close to 5.0	{3.04,4.97,1.44}	1.44

Due to the rounded values, we must cluster the possible output sets placed at: 1.3, 1.4, 2.8, 3.4 and 5.0. In this case the initial cluster is placed at 1.4 with P=1.8452, the next cluster is placed at 3.4 with P=1.0023, the third cluster is placed at 5.0 with P=1, the fourth cluster is placed at 2.8 with P=0.9999 and the fifth cluster is placed at 1.3 with P=0.2857. We stop here because the potential P in all the possible output sets have a potential P lower than 10%. Thus, the final output sets are placed at 1.3, 1.4, 2.8, 3.4 and 5.0.

We compute the RMSE of this model which is equal to 0.6074. Nevertheless we must do another iteration because either its NRMSE=58.86% or the NRMSE of the linearized fuzzy curve of the second input $\text{NRMSE}_{x_2} = 13.40\%$ are still higher than 10%. The NRMSE_{x_1} of the linearized fuzzy curve of the first input is 3.23% and by being lower than 10% we conclude that no more sets will be considered for this variable. For this reason we only seek the highest error in the linearization of the second variable. This occurs when x_2 is either 3.6, 3.7 or 3.8 with an error equal to 0.30 and therefore, we will consider a new fuzzy set placed at the mean of these values. Figure 4.9 shows the model proposed for the third iteration.

We repeat all the steps again. First, we look for the possible output values of the new rules. In this case we must compute three new rules:

Rule	Closest sample	Possible set
x_1 close to 1.0 and x_2 close to 3.7	{1.05,2.55,4.63}	4.63
x_1 close to 2.7 and x_2 close to 3.7	{2.24,3.74,1.79}	1.79
x_1 close to 4.9 and x_2 close to 3.7	{4.82,3.80,1.39}	1.39

Now we must cluster seven values: 1.3, 1.4, 1.8, 2.8, 3.4, 4.6 and 5.0. In this case the inital cluster is placed at 1.4 with P=1.9130, the next cluster is placed at 4.6 with P=1.0678, the third cluster is placed at 2.8 with P=1.0023, the fourth cluster is placed at 3.4 with P=0.9999, the fifth cluster is placed at 5.0 with P=0.9954, the sixth cluster is placed at 1.8 with P=0.9532 and the last cluster is placed at 1.3 with P=0.2291. We stop here after all the possible output sets have a potential P lower than 10%. Thus, the final output sets are placed at 1.3, 1.4, 1.8, 2.8, 3.4, 4.6 and 5.0.

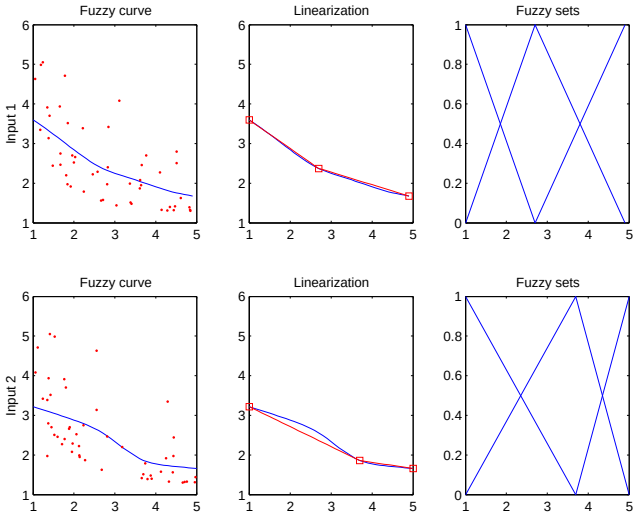


Figure 4.9: Third iteration.

With this model the RMSE=0.6192 and the NRMSE=60.00%. As either this NRMSE or the NRMSE of the linearization of the fuzzy curve of the second variable ($\text{NRMSE}_{x_2} = 10.54\%$) are still higher than 10% we look for another fuzzy set which will be placed at x_2=2.5 because this is the point with the highest error equal to 0.21. Figure 4.10 shows the model proposed for the fourth iteration.

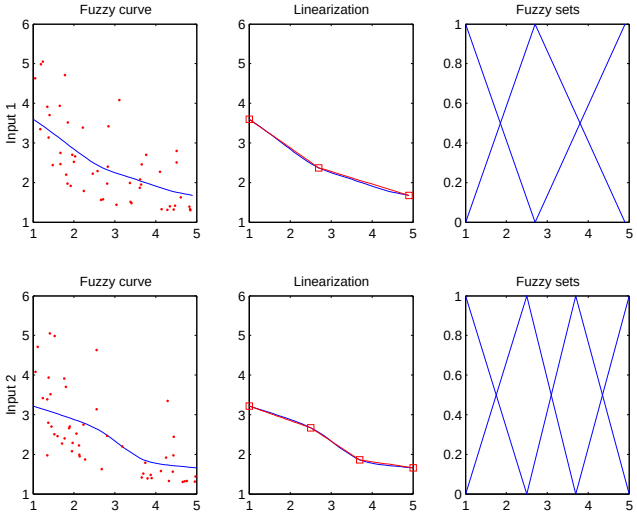


Figure 4.10: Fourth and last iteration.

We repeat the process again.

Therefore, we look for the possible output values of the new rules:

Rule	Closest sample	Possible set
x_1 close to 1.0 and x_2 close to 2.5	{1.05,2.55,4.63}	4.63
x_1 close to 2.7 and x_2 close to 2.5	{2.46,2.12,2.22}	2.22
x_1 close to 4.9 and x_2 close to 2.5	{4.61,2.68,1.63}	1.63

Now we must cluster the following nine output values: 1.3, 1.4, 1.6, 1.8, 2.2, 2.8, 3.4, 4.6 and 5.0. In this case the inital cluster is placed at 1.4 with P=2.4233, the second cluster is placed at 1.8 with P=1.4965, the third cluster is placed at 4.6 with P=1.0678, the fourth cluster is placed at 2.8 with P=1.0047, the fifth cluster is placed at 3.4 with P=0.9999, the sixth cluster is placed at 5.0 with P=0.9954 and the last cluster is placed at 2.2 with P=0.9686. The final output sets are placed at 1.4, 1.8, 2.2, 2.8, 3.4, 4.6 and 5.0. In fact this has been the first time the method has been able to remove some of the possible output sets and the nine original values have been reduced to seven.

This model has a RMSE=0.5409 and a NRMSE=52.42% which is higher than 10%. Nevertheless we stop the hierarchical process here because the NRMSE of the linearized fuzzy curve of the second variable (NRMSE_{x_2} = 2.21%) is finally lower than 10%.

Final model

We choose as result the model of the last iteration because it had the lowest RMSE. The final rules in this case are:

if x_1 is close to 1.0 and x_2 is close to 1.0 then y is close to 5.0
if x_1 is close to 1.0 and x_2 is close to 2.5 then y is close to 4.6
if x_1 is close to 1.0 and x_2 is close to 3.7 then y is close to 4.6
if x_1 is close to 1.0 and x_2 is close to 5.0 then y is close to 3.4
if x_1 is close to 2.7 and x_2 is close to 1.0 then y is close to 3.4
if x_1 is close to 2.7 and x_2 is close to 2.5 then y is close to 2.2
if x_1 is close to 2.7 and x_2 is close to 3.7 then y is close to 1.8
if x_1 is close to 2.7 and x_2 is close to 5.0 then y is close to 1.4
if x_1 is close to 4.9 and x_2 is close to 1.0 then y is close to 2.8
if x_1 is close to 4.9 and x_2 is close to 2.5 then y is close to 1.8
if x_1 is close to 4.9 and x_2 is close to 3.7 then y is close to 1.4
if x_1 is close to 4.9 and x_2 is close to 5.0 then y is close to 1.3

Many authors have proposed several techniques able to convert the previous result into a full linguistic result by applying to each fuzzy set a linguistic name. Early works were done by M. Sugeno [111] and since then several alternatives have appeared. Nevertheless this is not the objective of the current work and here we will not make more considerations about it.

4.2.3 Analysis of results

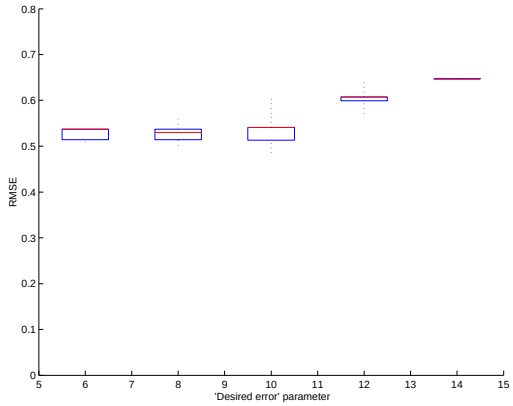
In general we may obtain better results in terms of error if the ε parameter is reduced but then we will probably have a higher number of sets and consequently a poorer linguistic intelligibility.

Anyway this is not always true by the fact to work with statistics of the optimal β which can differ in every execution. Therefore, we can analyze the robustness of the resulting models due to different ε values.

For this purpose we will consider five values for the ε equal to 6%, 8%, 10%, 12% and 14%. For each one we will perform 100 executions in order to have enough information to study the robustness of the method.

Due to the final amount of information, the results of the different analyses are shown with box plots together with the values for the upper quartile (75-th percentile), the median, the lower quartile (25-th percentile), the mean (μ), the standard deviation (σ) and the relation between these two last parameters (σ/μ).

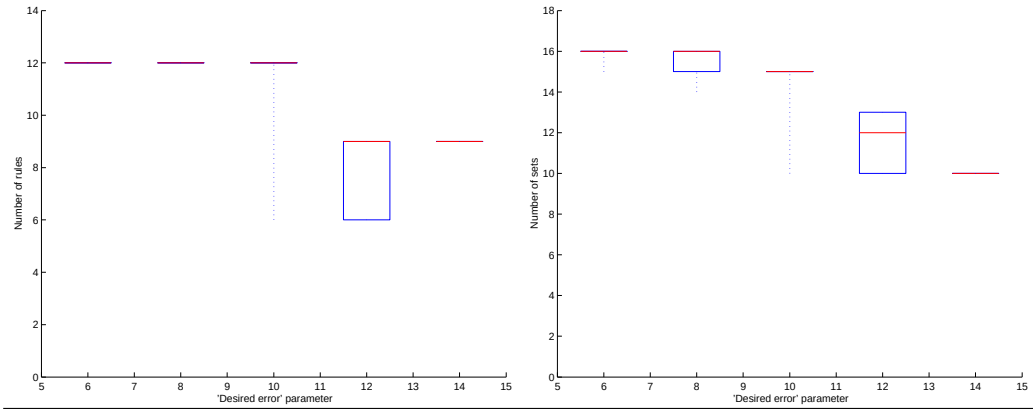
Figure 4.11 shows the box plot of the RMSE obtained with different ε together with their mean (μ_{RMSE}) and standard deviation (σ_{RMSE}). As we expected, a higher value in the ε yields in general a higher RMSE. Nevertheless, observe how there are extreme values for which the results are neither better nor worse. We will discuss these results after watching the rest of box plots.



Desired error	6%	8%	10%	12%	14%
Upper quartile	0.5369	0.5369	0.5409	0.6073	0.6469
Median	0.5369	0.5300	0.5409	0.6073	0.6469
Lower quartile	0.5139	0.5139	0.5130	0.5991	0.6469
μ_{RMSE}	0.5284	0.5290	0.5326	0.6041	0.6469
σ_{RMSE}	0.0111	0.0129	0.0222	0.0118	0.0000
σ_{RMSE}/μ_{RMSE}	0.0210	0.0244	0.0416	0.0196	0.0000

Figure 4.11: Box plot for the RMSE.

In order to compare the intelligibility of the models, the number of rules and the number of sets are given together again with their statistics in figure 4.12.



6%	8%	10%	12%	14%	Desired error	6%	8%	10%	12%	14%
12	12	12	9	9	Upper quartile	16	16	15	13	10
12	12	12	9	9	Median	16	16	15	12	10
12	12	12	6	9	Lower quartile	16	15	15	10	10
12	12	11.7300	7.5600	9	μ	15.7900	15.6400	14.6500	11.4300	10
0	0	1.1358	1.5064	0	σ	0.4094	0.5949	1.0088	1.4160	0
0	0	0.0968	0.1993	0	σ/μ	0.0259	0.0380	0.0689	0.1239	0

Figure 4.12: Box plots for the number of rules and number of sets.

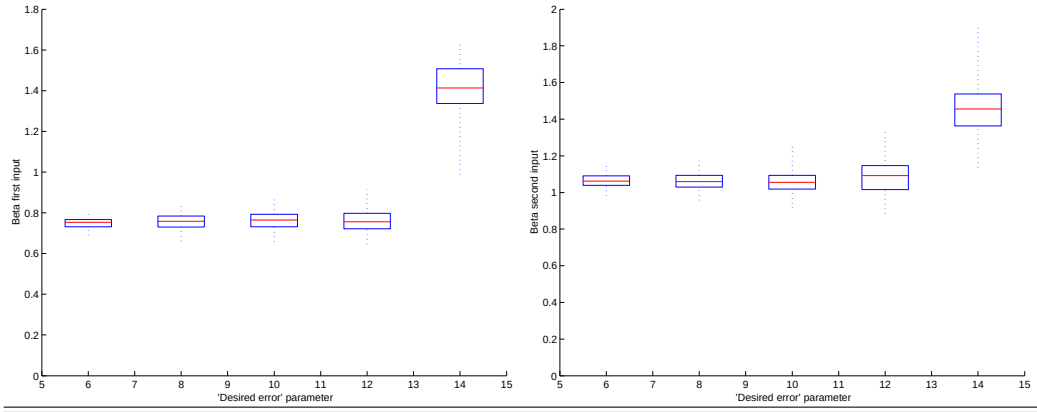
In fact the differences between the executions with the same *desired error* parameter are due to the different values of the optimal β. Figure 4.13 shows the box plots for β_{x_1} and β_{x_2}. A lower ε gives more confident parameters.

Finally the elapsed time[1] in seconds for each execution is given in figure 4.14 together again with their mean (μ_{etime}) and standard deviation (σ_{etime}). Obviously a higher value of ε decreases the elapsed time for each execution by considering less possible points in the universe of scope where the fuzzy sets can be placed.

After watching these box plots, we conclude that the models are quite robust in spite of working with statistical parameters because most of the boxes have the same value, either for the 25-th percentile or the 75-th percentile.

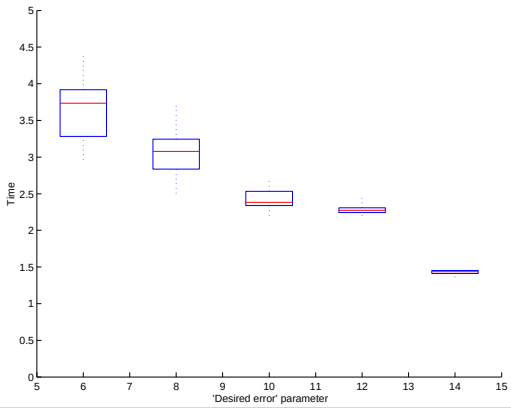
In general a higher value for the ε decreases the necessary number of rules and sets but increases the RMSE. But if the ε is very high then the inefficient grid of the universe of scope gives always the same model despite changing the values of the β parameter. If on the other hand the ε is very low, the very similar values for the β parameter give obviously the same model too.

[1]The method has been tested in Matlab R12.1 (Microsoft XP Professional) under a Mobile Pentium 4 at 1.7GHz with 512MB of RAM.



6%	8%	10%	12%	14%	Desired error	6%	8%	10%	12%	14%
0.7670	0.7843	0.7933	0.7977	1.5075	Upper quartile	1.0902	1.0937	1.0933	1.1459	1.5377
0.7538	0.7583	0.7639	0.7561	1.4137	Median	1.0619	1.0589	1.0546	1.0920	1.4562
0.7310	0.7302	0.7317	0.7220	1.3366	Lower quartile	1.0395	1.0288	1.0187	1.0153	1.3633
0.7501	0.7564	0.7617	0.7622	1.4133	μ	1.0638	1.0610	1.0567	1.0908	1.4562
0.0231	0.0389	0.0435	0.0566	0.1247	σ	0.0359	0.0490	0.0664	0.0937	0.1441
0.0309	0.0515	0.0571	0.0742	0.0883	σ/μ	0.0338	0.0462	0.0628	0.0858	0.0990

Figure 4.13: Box plots for the β parameters.



Desired error	6%	8%	10%	12%	14%
Upper quartile	3.9200	3.2440	2.5340	2.3090	1.4520
Median	3.7355	3.0790	2.3840	2.2730	1.4370
Lower quartile	3.2800	2.8340	2.3380	2.2435	1.412
μ_{etime}	3.6578	3.0378	2.4246	2.2922	1.4296
σ_{etime}	0.4389	0.3024	0.1443	0.0751	0.0284
$\sigma_{etime}/\mu_{etime}$	0.1200	0.0996	0.0596	0.0327	0.0199

Figure 4.14: Box plot for the elapsed time.

Observe how in spite of having slight differences between models we can find a model with a medium value for the ε whose RMSE is the lowest one because the accuracy of β is proportional in general to the RMSE but the fact to work with nonlinear systems can break sometimes this rule.

4.2.4 Model with highest accuracy

Here it is interesting to observe that among the 100×5 executions, the result with the lower RMSE was obtained when ε=10% and whose optimal values for the β parameters were equal to 0.699 for x_1 and 1.190 for x_2. This model plotted in figure 4.15 had three sets for x_1, four sets for x_2, 8 output sets, 12 rules and a RMSE=0.485. Table 4.2 shows the 12 rules of this model where the numbers are the cores of the fuzzy sets.

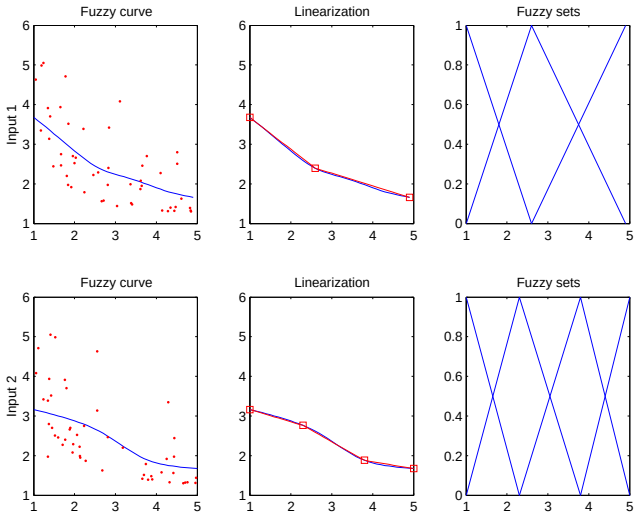


Figure 4.15: Best model in terms of RMSE.

Rule matrix				
$x_1 \backslash x_2$	**1.0**	**2.3**	**3.8**	**5.0**
1.0	5.0	4.6	4.6	3.4
2.6	3.4	2.2	1.8	1.4
4.9	2.8	1.8	1.4	1.3

Table 4.2: Best model in terms of RMSE.

4.2.5 Comparisons

The results are compared with other methods in table 4.3 where, apart from the error, the number of sets, the number of rules and the type of the fuzzy model (Takagi-Sugeno or Sugeno-Yasukawa[2] are given in order to compare their linguistic capabilities. The values given with our method are the ones computed with the median β parameters of the previous executions.

[2]Sugeno-Yasukawa is the output singleton FRBS which we consider.

Method	Type	Sets	Rules	RMSE
Sugeno [111]	S-Y	18	6	0.564
Delgado EST1[20]	S-Y	15	25	0.493
Our method with de=6%	S-Y	16	12	0.537
Our method with de=8%	S-Y	16	12	0.530
Our method with de=10%	S-Y	15	12	0.541
Our method with de=12%	S-Y	12	9	0.607
Our method with de=14%	S-Y	10	9	0.647
Our method (best model)	S-Y	15	12	0.485
Kim [58]	T-S	3 eqs. with 6 vars.	3	0.281

Table 4.3: Comparisons between different methods.

It is difficult to compare a Sugeno-Yasukawa's model with a Takagi-Sugeno's one. As cited in [20] *the first one is easier to implement and intuitively more persuasive towards human beings because its consequent parts are expressed by linguistic variables, not linear equations* and for this reason we considered this option.

Anyway, our method shows similar performances to others with a simple algorithm which can also adjust its error and consequently its linguistic capabilities based on the ε parameter.

4.2.6 Summary

Despite working with statistical parameters which can give different models in different executions, the box plots have shown how the results are quite robust and in many cases the models are very similar between them or even exactly the same.

We have also observed how the trade-off between intelligibility and accuracy can be adjusted with the *desired error* value. Nevertheless there are some extreme values for which the models do not vary.

4.3 Predicting two popular time series

4.3.1 Box and Jenkins' Gas Furnace Data

In this section we will study the data appeared in [7] which are 296 values given in table 4.4 of a gas furnace system with a sampling interval of 9 seconds.

The data were recorded from a combustion process of an air-methane mixture. The only input $u(k)$ is the gas flow rate into the furnace and the output measurement $y(k)$ is de CO2 concentration in outlet gas.

These data are considered as a benchmark in prediction analyses.

Box and Jenkins' Data														
k	u	y	k	u	y	k	u	y	k	u	y	k	u	y
1	-0.109	+53.8	61	+1.146	+51.5	121	-0.876	+59.4	181	-1.218	+52.3	241	-1.794	+57.8
2	+0.000	+53.6	62	+1.155	+51.6	122	-0.885	+58.4	182	-1.183	+53.0	242	-1.302	+58.3
3	+0.178	+53.5	63	+1.112	+51.2	123	-0.800	+57.6	183	-0.873	+53.8	243	-1.030	+58.6
4	+0.339	+53.5	64	+1.121	+50.5	124	-0.544	+56.9	184	-0.336	+54.6	244	-0.918	+58.8
5	+0.373	+53.4	65	+1.223	+50.1	125	-0.416	+56.4	185	+0.063	+55.4	245	-0.798	+58.8
6	+0.441	+53.1	66	+1.257	+49.8	126	-0.271	+56.0	186	+0.084	+55.9	246	-0.867	+58.6
7	+0.461	+52.7	67	+1.157	+49.6	127	+0.000	+55.7	187	+0.000	+55.9	247	-1.047	+58.0
8	+0.348	+52.4	68	+0.913	+49.4	128	+0.403	+55.3	188	+0.001	+55.2	248	-1.123	+57.4
9	+0.127	+52.2	69	+0.620	+49.3	129	+0.841	+55.0	189	+0.209	+54.4	249	-0.876	+57.0
10	-0.180	+52.0	70	+0.255	+49.2	130	+1.285	+54.4	190	+0.556	+53.7	250	-0.395	+56.4
11	-0.588	+52.0	71	-0.280	+49.3	131	+1.607	+53.7	191	+0.782	+53.6	251	+0.185	+56.3
12	-1.055	+52.4	72	-1.080	+49.7	132	+1.746	+52.8	192	+0.858	+53.6	252	+0.662	+56.4
13	-1.421	+53.0	73	-1.551	+50.3	133	+1.683	+51.6	193	+0.918	+53.2	253	+0.709	+56.4
14	-1.520	+54.0	74	-1.799	+51.3	134	+1.485	+50.6	194	+0.862	+52.5	254	+0.605	+56.0
15	-1.302	+54.9	75	-1.825	+52.8	135	+0.993	+49.4	195	+0.416	+52.0	255	+0.501	+55.2
16	-0.814	+56.0	76	-1.456	+54.4	136	+0.648	+48.8	196	-0.336	+51.4	256	+0.603	+54.0
17	-0.475	+56.8	77	-0.944	+56.0	137	+0.577	+48.5	197	-0.959	+51.0	257	+0.943	+53.0
18	-0.193	+56.8	78	-0.570	+56.9	138	+0.577	+48.7	198	-1.813	+50.9	258	+1.223	+52.0
19	+0.088	+56.4	79	-0.431	+57.5	139	+0.632	+49.2	199	-2.378	+52.4	259	+1.249	+51.6
20	+0.435	+55.7	80	-0.577	+57.3	140	+0.747	+49.8	200	-2.499	+53.5	260	+0.824	+51.6
21	+0.771	+55.0	81	-0.960	+56.6	141	+0.900	+50.4	201	-2.473	+55.6	261	+0.102	+51.1
22	+0.866	+54.3	82	-1.616	+56.0	142	+0.993	+50.7	202	-2.330	+58.0	262	+0.025	+50.4
23	+0.875	+53.2	83	-1.875	+55.4	143	+0.968	+50.9	203	-2.053	+59.5	263	+0.382	+50.0
24	+0.891	+52.3	84	-1.891	+55.4	144	+0.790	+50.7	204	-1.739	+60.0	264	+0.922	+50.0
25	+0.987	+51.6	85	-1.746	+56.4	145	+0.399	+50.5	205	-1.261	+60.4	265	+1.032	+52.0
26	+1.263	+51.2	86	-1.474	+57.2	146	-0.161	+50.4	206	-0.569	+60.5	266	+0.866	+54.0
27	+1.775	+50.8	87	-1.201	+58.0	147	-0.553	+50.2	207	-0.137	+60.2	267	+0.527	+55.1
28	+1.976	+50.5	88	-0.927	+58.4	148	-0.603	+50.4	208	-0.024	+59.7	268	+0.093	+54.5
29	+1.934	+50.0	89	-0.524	+58.4	149	-0.424	+51.2	209	-0.050	+59.0	269	-0.458	+52.8
30	+1.866	+49.2	90	+0.040	+58.1	150	-0.194	+52.3	210	-0.135	+57.6	270	-0.748	+51.4
31	+1.832	+48.4	91	+0.788	+57.7	151	-0.049	+53.2	211	-0.276	+56.4	271	-0.947	+50.8
32	+1.767	+47.9	92	+0.943	+57.0	152	+0.060	+53.9	212	-0.534	+55.2	272	-1.029	+51.2
33	+1.608	+47.6	93	+0.930	+56.0	153	+0.161	+54.1	213	-0.871	+54.5	273	-0.928	+52.0
34	+1.265	+47.5	94	+1.006	+54.7	154	+0.301	+54.0	214	-1.243	+54.1	274	-0.645	+52.8
35	+0.790	+47.5	95	+1.137	+53.2	155	+0.517	+53.6	215	-1.439	+54.1	275	-0.424	+53.8
36	+0.360	+47.6	96	+1.198	+52.1	156	+0.566	+53.2	216	-1.422	+54.4	276	-0.276	+54.5
37	+0.115	+48.1	97	+1.054	+51.6	157	+0.560	+53.0	217	-1.175	+55.5	277	-0.158	+54.9
38	+0.088	+49.0	98	+0.595	+51.0	158	+0.573	+52.8	218	-0.813	+56.2	278	-0.033	+54.9
39	+0.331	+50.0	99	-0.080	+50.5	159	+0.592	+52.3	219	-0.634	+57.0	279	+0.102	+54.8
40	+0.645	+51.1	100	-0.314	+50.4	160	+0.671	+51.9	220	-0.582	+57.3	280	+0.251	+54.4
41	+0.960	+51.8	101	-0.288	+51.0	161	+0.933	+51.6	221	-0.625	+57.4	281	+0.280	+53.7
42	+1.409	+51.9	102	-0.153	+51.8	162	+1.337	+51.6	222	-0.713	+57.0	282	+0.000	+53.3
43	+2.670	+51.7	103	-0.109	+52.4	163	+1.460	+51.4	223	-0.848	+56.4	283	-0.493	+52.8
44	+2.834	+51.2	104	-0.187	+53.0	164	+1.353	+51.2	224	-1.039	+55.9	284	-0.759	+52.6
45	+2.812	+50.0	105	-0.255	+53.4	165	+0.772	+50.7	225	-1.346	+55.5	285	-0.824	+52.6
46	+2.483	+48.3	106	-0.229	+53.6	166	+0.218	+50.0	226	-1.628	+55.3	286	-0.740	+53.0
47	+1.929	+47.0	107	-0.007	+53.7	167	-0.237	+49.4	227	-1.619	+55.2	287	-0.528	+54.3
48	+1.485	+45.8	108	+0.254	+53.8	168	-0.714	+49.3	228	-1.149	+55.4	288	-0.204	+56.0
49	+1.214	+45.6	109	+0.330	+53.8	169	-1.099	+49.7	229	-0.488	+56.0	289	+0.034	+57.0
50	+1.239	+46.0	110	+0.102	+53.8	170	-1.269	+50.6	230	-0.160	+56.5	290	+0.204	+58.0
51	+1.608	+46.9	111	-0.423	+53.3	171	-1.175	+51.8	231	-0.007	+57.1	291	+0.253	+58.6
52	+1.905	+47.8	112	-1.139	+53.0	172	-0.676	+53.0	232	-0.092	+57.3	292	+0.195	+58.5
53	+2.023	+48.2	113	-2.275	+52.9	173	+0.033	+54.0	233	-0.620	+56.8	293	+0.131	+58.3
54	+1.815	+48.3	114	-2.594	+53.4	174	+0.556	+55.3	234	-1.086	+55.6	294	+0.017	+57.8
55	+0.535	+47.9	115	-2.716	+54.6	175	+0.643	+55.9	235	-1.525	+55.0	295	-0.182	+57.3
56	+0.122	+47.2	116	-2.510	+56.4	176	+0.484	+55.9	236	-1.858	+54.1	296	-0.262	+57.0
56	+0.122	+47.2	116	-2.510	+56.4	176	+0.484	+55.9	236	-1.858	+54.1			
57	+0.009	+47.2	117	-1.790	+58.0	177	+0.109	+54.6	237	-2.029	+54.3			
58	+0.164	+48.1	118	-1.346	+59.4	178	-0.310	+53.5	238	-2.024	+55.3			
59	+0.671	+49.4	119	-1.081	+60.2	179	-0.697	+52.4	239	-1.961	+56.4			
60	+1.019	+50.6	120	-0.910	+60.0	180	-1.047	+52.1	240	-1.952	+57.2			

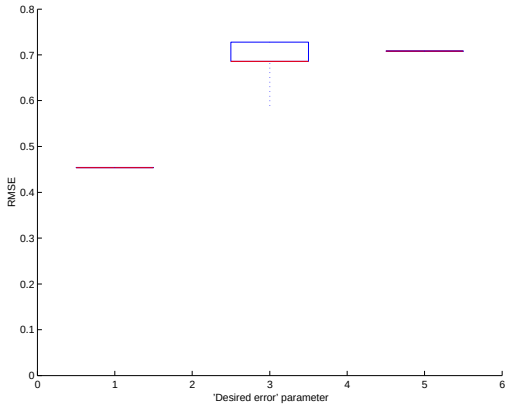
Table 4.4: Box and Jenkins' Data.

Conventional models have considered $u(k), u(k-1), u(k-2), \ldots, y(k-1), y(k-2), y(k-3), \ldots$ as possible inputs in order to predict the current output $y(k)$.

First, we will consider two input variables, $y(k-1)$ and $u(k-4)$ by being one of the most correlated variables with $y(k)$[3]. These two variables have been commonly used in literature. In fact the furnace considered here is believed to exhibit a slow dynamic response and old control inputs must significantly influence the newest output.

This example has normally been applied to emphasize the precision of prediction techniques. For this reason and in order to compare results, we will consider low values for the *desired error* parameter ε in spite of not being interested in models with high accuracy but with a satisfactory intelligibility. Here we will consider three different values for the *desired error* parameter ε equal to 1%, 3% and 5%. For each value we computed 100 executions in order to verify the robustness of the results.

We will show the same box plots as before. Figure 4.16 shows the box plot for the RMSE, figure 4.17 shows the box plots for the number of rules and sets and figure 4.18 shows the box plot for the β parameter.

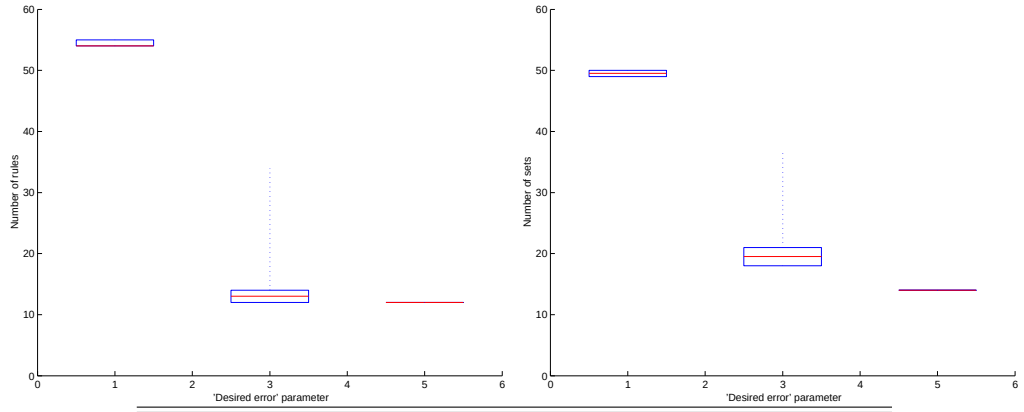


Desired error	1%	3%	5%
Upper quartile	0.4541	0.7279	0.7080
Median	0.4539	0.6860	0.7080
Lower quartile	0.4537	0.6860	0.7080
μ_{RMSE}	0.4539	0.6931	0.7080
σ_{RMSE}	0.0001	0.0420	0.0000
σ_{RMSE}/μ_{RMSE}	0.0004	0.0607	0.0000

Figure 4.16: Box plot for the RMSE.

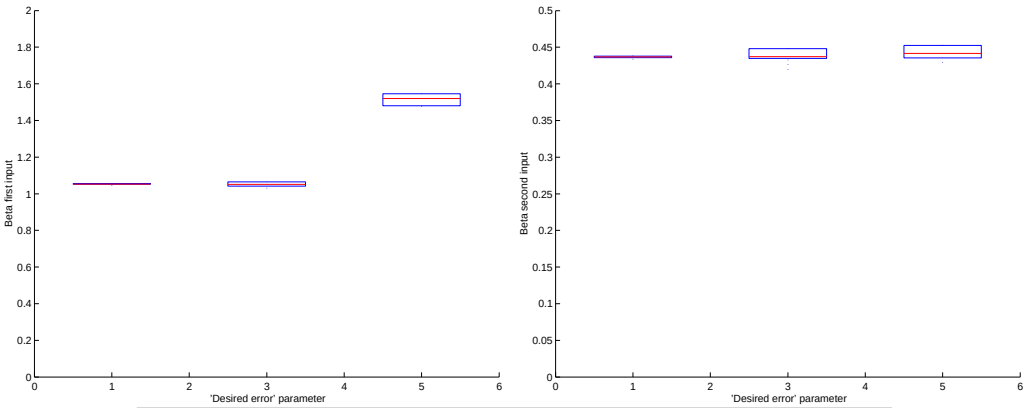
Observe how the results are quite robust like in the previous example and in fact even more because we have more samples. The conclusions about how the accuracy and intelligibility vary based on the value of the β parameter are the same than before.

[3]The correlation coefficient between $y(k)$ and $y(k-1)$ is 0.97 while between $y(k)$ and $u(k-4)$ is 0.92.



1%	3%	5%	Desired error	1%	3%	5%
55	14	12	Upper quartile	50	21	14
54	13	12	Median	49.5	19.5	14
54	12	12	Lower quartile	49	18	14
54.4	15.1	12	μ	49.5	21.1	14
0.5164	6.7404	0.0000	σ	0.5271	5.7822	0.0000
0.0095	0.4464	0.0000	σ/μ	0.0106	0.2740	0.0000

Figure 4.17: Box plots for the number of rules and the number of sets.

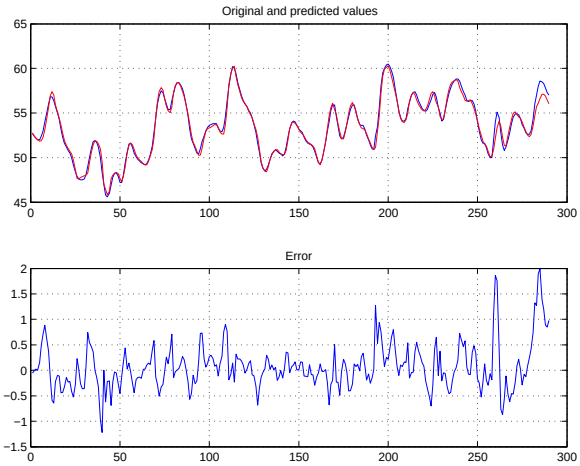


1%	3%	5%	Desired error	1%	3%	5%
1.0553	1.0644	1.5461	Upper quartile	0.4380	0.4480	0.4524
1.0530	1.0517	1.5200	Median	0.4368	0.4371	0.4414
1.0519	1.0410	1.4798	Lower quartile	0.4357	0.4347	0.4354
1.0536	1.0520	1.5175	μ	0.4366	0.4382	0.4425
0.0060	0.0210	0.0319	σ	0.0015	0.0095	0.0088
0.0057	0.0114	0.0210	σ/μ	0.0035	0.0217	0.0198

Figure 4.18: Box plots for the optimal β.

For this reason we will just show the resulting predictions with the median values of β: $\beta_{y(k-1)} = 1.0530$ and $\beta_{u(k-4)} = 0.4368$ when ε=1%, $\beta_{y(k-1)} = 1.0517$ and $\beta_{u(k-4)} = 0.4371$ when ε=3% and $\beta_{y(k-1)} = 1.5200$ and $\beta_{u(k-4)} = 0.4414$ when ε=5%.

Figures 4.19, 4.20 and 4.21 show the original and predicted values for each model when ε=1%, 3% and 5% respectively. Observe how among the different combinations of input sets, there are some possible rules which are not defined (NaR) because they don't have any sample inside its neighborhood.



Rule matrix

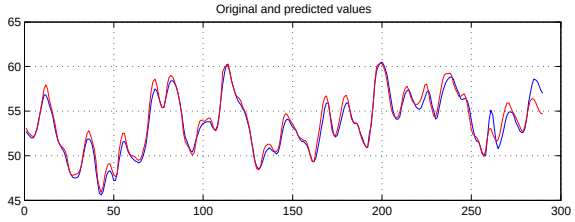
$y(k-1)\backslash u(k-4)$	**-2.8**	**-2.4**	**-1.4**	**-0.7**	**0.1**	**1.2**	**1.8**	**2.1**	**2.6**	**2.9**
45.6	53.0	NaR	NaR	52.0	NaR	47.8	48.1	46.9	46.0	45.6
48.1	59.4	58.0	54.4	53.0	49.4	48.3	47.9	47.2	47.0	47.0
49.4	NaR	56.0	54.4	52.0	50.6	49.3	48.8	47.9	47.0	47.0
51.4	NaR	56.0	54.4	53.0	51.6	51.0	49.4	NaR	NaR	NaR
55.7	59.4	58.0	56.8	55.3	54.6	54.5	54.5	NaR	47.0	47.0
59.8	60.2	60.0	58.4	57.6	58.3	57.8	NaR	NaR	NaR	NaR
60.5	60.0	60.5	58.4	57.6	NaR	57.8	52.8	NaR	NaR	57.8

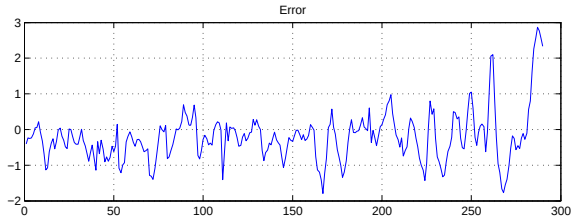
Figure 4.19: Resulting prediction when ε=1%.

The RMSE was 0.4537 when ε=1%, 0.7279 when ε=3% and 0.7080 when ε=5%. Despite not being very usual, in this case the RMSE when ε is 5% is lower than the RMSE when this parameter is 3%. This is due to the fact to work with statistical results because in general the lower value for the ε the lower RMSE, as can be observed in the box plots of figure 4.16.

Finally we can compare these results with other alternatives appeared in the literature which are given in table 4.5.

Observe that we achieve a similar performance in comparison with other methods in spite of the simplicity of the method. The accuracy and intelligibility are in the average of other techniques which also consider the Sugeno-Yasukawa's model. In fact we can obtain the lowest RMSE among these alternatives by considering a low value for the ε parameter in spite of increasing the number of sets. On the other hand we can obtain a simple and fast model by increasing this parameter but then the RMSE is slightly higher than the other methods. Suprisingly, the results are even better than some

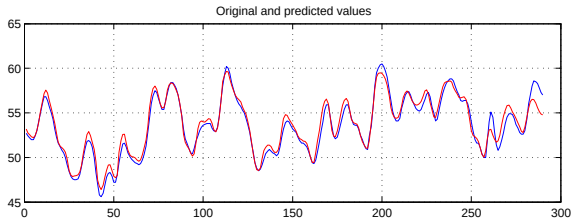


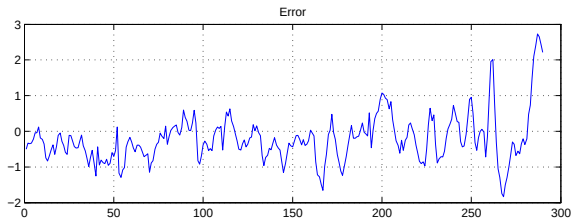


Rule matrix

$y(k-1)\backslash u(k-4)$	**-2.8**	**-2.4**	**-0.7**	**1.8**	**2.9**
45.6	53.0	NaR	52.0	48.1	45.6
48.1	59.4	58.0	53.0	47.9	47.0
60.5	60.0	60.5	57.6	52.8	57.8

Figure 4.20: Resulting prediction when ε=3%.





Rule matrix

$y(k-1)\backslash u(k-4)$	**-2.8**	**-0.7**	**1.8**	**2.9**
45	53	52	48	46
48	58	53	48	47
61	60	58	53	58

Figure 4.21: Resulting prediction when ε=5%.

Takagi-Sugeno's models which nevertheless, can obtain the lowest RMSE but with a very complex model with up to six input variables.

Method	Type	Inputs	Rules	RMSE
Tong [113]	S-Y	2	19	0.685
Xu [122]	S-Y	2	25	0.573
Pedrycz [86]	S-Y	2	81	0.566
Peng [90]	S-Y	2	49	0.548
Our method with ε=1%	S-Y	2	54	0.454
Our method with ε=3%	S-Y	2	14	0.728
Our method with ε=5%	S-Y	2	12	0.708
Yoshinari [124]	T-S	2	6	0.546
Sugeno-Yasukawa[111]	T-S	3	6	0.436
Wang-Langari [119]	T-S	2	5	0.415
Sugeno-Tanaka [110]	T-S	6	2	0.261
Wang-Langari [118]	T-S	6	2	0.257

Table 4.5: Comparisons between different methods.

4.3.2 Mackey-Glass chaotic time series

Chaotic systems are based on deterministic maps which generate sequences which seem to be *random* series. Anyway its behavior seems to be so unpredictable that its characterization is usually very arduous. Here we will study the capabilities of the proposed method in order to model the Mackey-Glass chaotic time series defined with the following delay differential equation 4.3.

$$\frac{dx\left(t\right)}{dt} = \frac{0.2x\left(t-\tau\right)}{1+x^{10}\left(t-\tau\right)} - 0.1x\left(t\right) \tag{4.3}$$

It describes the arterial CO2 concentration in the case of normal and abnormal respiration [72]. Under the premise that $\tau > 17$ it exhibits a chaotic behavior. Higher values of τ yield to higher dimensional chaos.

These series were already used by L.X. Wang and J.M. Mendel [120] with $\tau = 30$. Nevertheless and in order to facilitate the comparisons with other methods, here we will consider $\tau = 17$ like J.R. Jang [47, 51] because this is the value with more references since then in the literature.

We will use exactly the same data of Jang which are 1000 samples of four past values with the time lag of 6 seconds {x(t-18), x(t-12), x(t-6), x(t)} in order to predict a value 6 seconds ahead {x(t+6)}. The first 500 pairs are used as train set while the remaining 500 are used as test set. The values we have considered have been computed with the fourth-order Runge-Kutta method with a fixed time step of 0.1 seconds and an initial condition x(0)=1.2. The 1000 data pairs have been stored every second between t=118 and t=1117 and are plotted in figure 4.22.

Despite willing to *predict* its behavior, we will not consider a very low value for the *desired error* parameter ε in order to obtain a satisfactory model in terms of intelligibility. The fact to have four inputs increases the

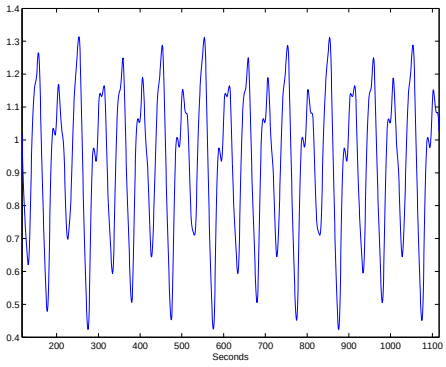


Figure 4.22: Samples of the chaotic time series.

complexity of model and with a low value for the ε the resulting model could be inintelligible if a high accuracy was required. Thus, we will consider only ε=5% and ε=10% for comparisons.

After computing 100 different executions with the train data, the median values for the β parameters were $\beta_{x(t-18)} = 0.0652$, $\beta_{x(t-12)} = 0.0877$, $\beta_{x(t-6)} = 0.1378$ and $\beta_{x(t)} = 0.1519$ when ε=5% while $\beta_{x(t-18)} = 0.1183$, $\beta_{x(t-12)} = 0.1633$, $\beta_{x(t-6)} = 0.2659$ and $\beta_{x(t)} = 0.2434$ when ε=10%. Observe that the values in the second case are close to the double of those found in the first one. This is due to the grid partition of the universe of scope. Remember that when computing the optimal β, the data are grouped in the test set and the train set, which are made by taking two samples close to each possible point of the universe of scope where a fuzzy set can be placed. Thus, a lower ε yields a narrow grid and therefore, as the resulting train and test sets have their values closer between them, the optimal β is lower.

Figures 4.23 and 4.24 show the final models when ε=5% and ε=10%, respectively. In the first case we obtained 4 sets for x(t-18), x(t-12) and x(t-6), 2 sets for x(t) and 27 sets for the output x(t+6) with a total amount of 124 rules. In the second case we obtained 5 sets for x(t-18), 4 sets for x(t-12), 3 sets for x(t-6), 4 sets for x(t) and 10 sets for the output x(t+6) with a total amount of 179 rules.

Observe how these models are surprisingly different from those computed in the other examples because in this case the model with the low ε requires less sets in order to achieve the *desired error*. This happened because the model with the high ε had a wide grid in the universe of scope and then the method demanded a high number of sets in order to achieve the *desired error*. On the other hand, when the ε was low then the narrow grid of the universe of scope helped the model to place the sets in less but better points and thus, it needed less sets in order to satisfy the *desired error*.

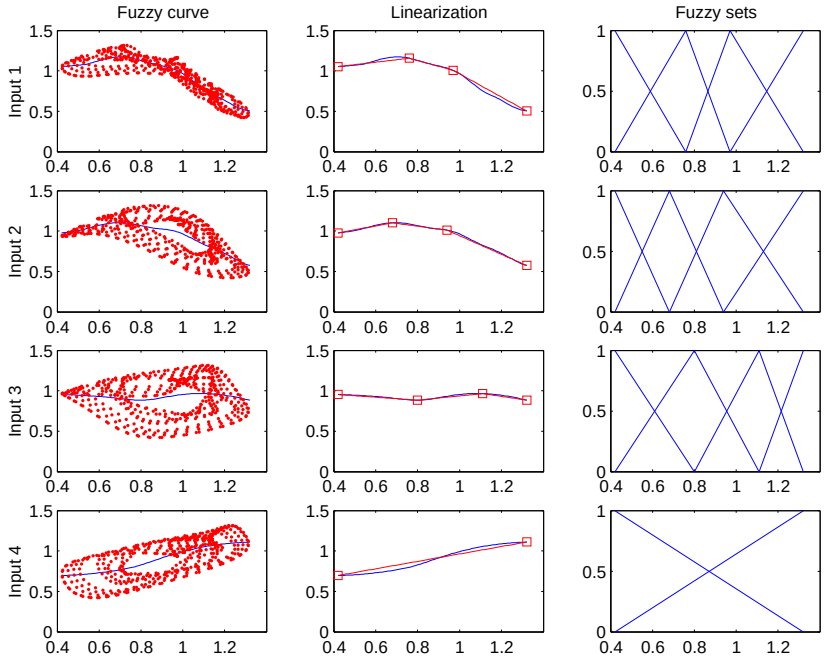


Figure 4.23: Resulting model when ε=5%.

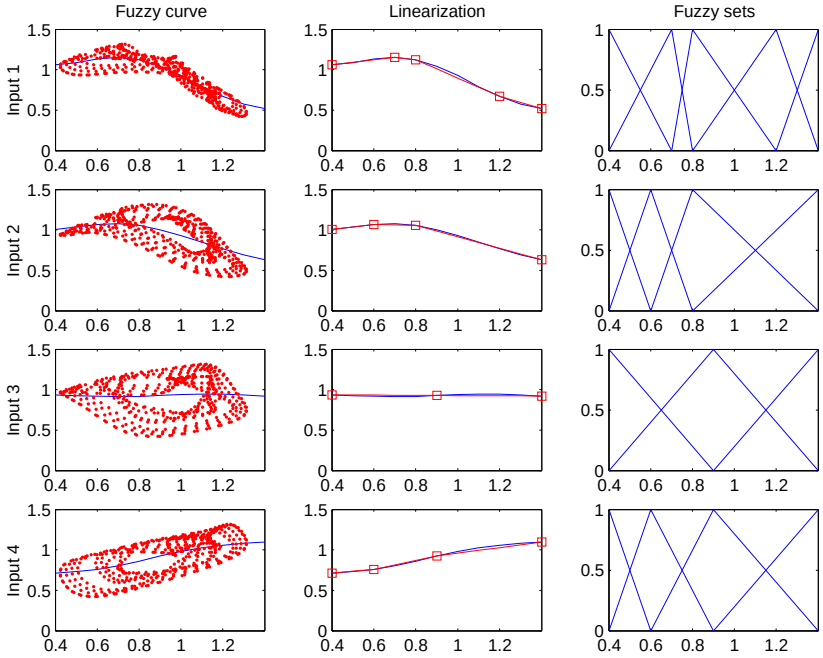


Figure 4.24: Resulting model when ε=10%.

Figures 4.25 and 4.26 show the resulting predictions when ε=5% and ε=10% respectively, either for the train set or the test set. In the first case RMSE$_{\text{train}}$=0.0413 and RMSE$_{\text{test}}$=0.0407. In the second case we obtained a RMSE$_{\text{train}}$=0.0647 and RMSE$_{\text{test}}$=0.0614.

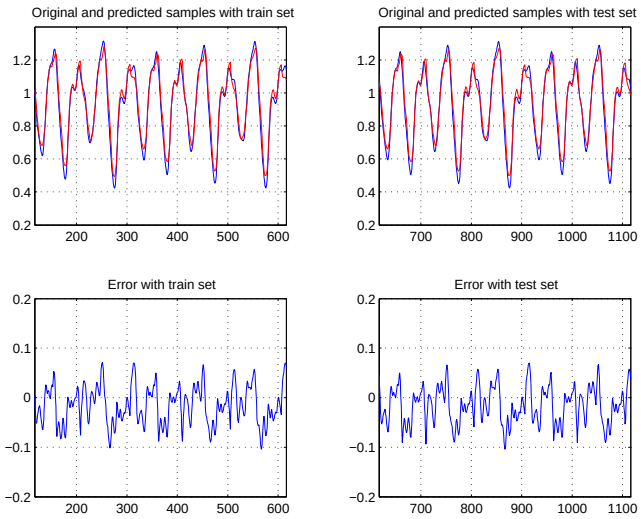


Figure 4.25: Resulting predictions when ε=5%.

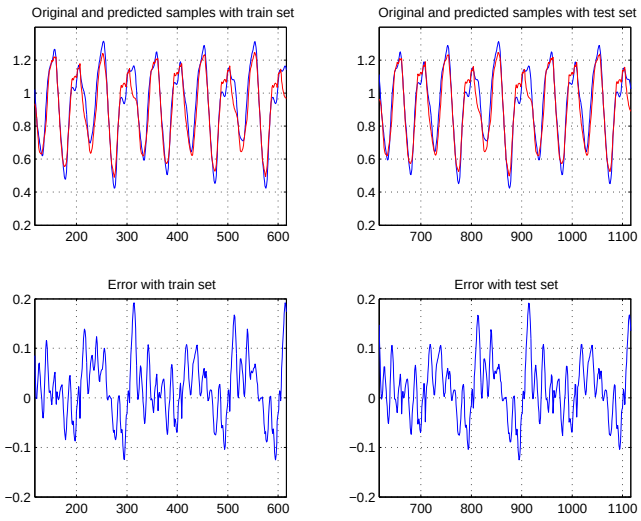


Figure 4.26: Resulting predictions when ε=10%.

We can compare these results with other methods found in the literature. Table 4.6 shows their main differences.

Method	Type	RMSE
S. Chiu [18]	T-S with 9 clusters and 9 linear eqs. with 4 vars.	0.0005
R. Jang (ANFIS)	T-S with 2 sets per input and 16 linear eqs. with 4 vars.	0.0016
Gaweda [33]	3 relational fuzzy rules with 39 parameters	0.0071
AR model	$x(t+6) = a_0 + a_1x(t) + a_2x(t-6) + \ldots + a_{103}x(t-102\times 6)$	0.0432
Linear model	$x(t+6) = a_0 + a_1x(t) + a_2x(t-6) + a_3x(t-12) + a_4x(t-18)$	0.1251
Our method with ε=5%	S-Y with 14 input sets	0.0413
Our method with ε=10%	S-Y with 16 input sets	0.0647

Table 4.6: Comparisons between different methods.

As we expected, our method is worse in terms of the error than many other techniques. Chiu and Jang (ANFIS) obtain the lowest RMSE because this is their prior objective; nevertheless its computational cost is very high. For example ANFIS needs a media of 123 seconds per epoch and if 500 epochs are computed, which is the same number used by Jang, more than 17 hours are necessary. On the contrary we just need 3.5 seconds plus 0.033 seconds for every iteration when computing the β parameters when ε=10%. The slowest cases were found when 68 iterations were necessary for $\beta_{x(t-18)}$, 113 $\beta_{x(t-12)}$, 536 for $\beta_{x(t-6)}$ and 112 for $\beta_{x(t)}$, and the whole method lasted 30.8 seconds. Both time computations have been done under the same hardware[4] and software[5]. Gaweda obtained very good results but nevertheless, prior information about the correlation between the variables is necessary in order to group them. Linear models perform in general worse than our method in terms of the error.

4.3.3 Summary

Models for making predictions with a high accuracy have always been demanded without taking into account its intelligibility. Thus, when fuzzy logic was considered as a possible method, their linguistic benefits began to be forgotten. These were in fact the responsibles of loosing for years the main virtues of fuzzy logic proposed by Zadeh.

Despite not being very interested in accurate predictions we have also shown how our method can obtain similar results while keeping the intelligibility of the models. In fact its main virtue is getting a first intelligible model to undergo further analyses with low computational cost.

[4]Mobile Pentium 4 at 1.7GHz with 512MB of RAM.

[5]Matlab (R12.1) and ANFIS has been tested with the application developed by Jang for the Matlab's fuzzy logic toolbox (version 2.1.1).

4.4 Two popular control problems

4.4.1 Truck and trailer

With this problem we will analyze our method from two points of view. First, we will study the truck backer-upper control in order to show how it obtains a fuzzy control from a set of required control actions. Then we will study the truck and trailer backer-upper control in order to show how it explains the actions that a very efficient non-fuzzy control applies to the truck.

The first case, originally proposed by D.H. Nguyen and B. Widrow [82], is a typical nonlinear control problem to back up a truck to the loading dock from any reasonable initial location. Figure 4.27 shows the loading zone and also the variables involved in this control. The truck position is exactly determined by the three state variables $x \in [-50m, +50m]$, $y \in [0, +100m]$ and $\theta \in [-90, +270]$. The first two variables (x, y) specify the position of the rear center of the truck in the plane. The last variable θ is the angle of the truck with the horizontal.

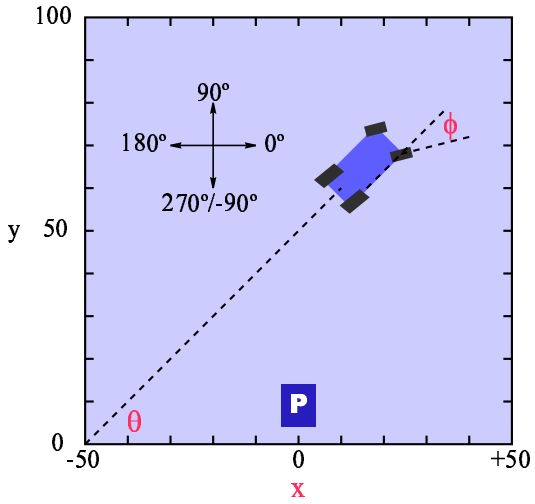


Figure 4.27: Loading zone for the truck.

The goal is to make the truck arrive to the loading dock ($x_f = 0, y_f = 0$) and to have the truck aligned with $\theta_f = 90$. Only backing up is considered and the truck moves backward by a fixed distance every step. The controller must produce the appropiate steering angle $\phi \in [-45, +45]$ at every step from the former input variables. Positive angles of ϕ are counterclockwise rotations while negative ones are clockwise. Furthermore, the y-position can be omitted as input if enough clearance between the truck and the loading dock is given.

We will use our method in order to build a fuzzy controller able to solve the truck backer-upper case.

A common approach in control problems is to work with the samples given by an expert. This approach deals with fuzzy logic theory because experts can bring out linguistic relations which easily can turn into the necessary samples to build the controller.

Nevertheless the required amount of samples is difficult to know. As we need enough data to fit a good representation of the necessary operations to drive the truck, here we will consider the following input values: from -50 meters to +50 meters with a step of 5 meters for the x-position and from -90º to +270º with a step of 30º for the truck angle. Finally a steering angle from -45º to +45º with a step of 15º is assigned to each possible input pair. From the total amount of 273 samples only those with $x \geq 0$ are given in table 4.7 due to the simmetry of the steering angle when $x < 0$.

Samples											
$\theta \setminus x$	**0**	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
-90	-30	-30	-30	-30	-15	-15	-15	-15	-15	-15	-15
-60	-30	-30	-30	-30	-15	-15	-15	-15	-15	-15	-15
-30	-30	-30	-30	-30	-15	-15	-15	-15	-15	-15	-15
0	-30	-30	-30	-30	-15	-15	-15	-15	-15	-15	-15
30	-30	-15	-15	-15	+15	+15	+15	+15	+15	+15	+15
60	-15	+15	+15	+15	+30	+30	+30	+30	+30	+30	+30
90	0	+30	+30	+30	+30	+30	+30	+30	+30	+30	+30
120	+15	+30	+30	+30	+45	+45	+45	+45	+45	+45	+45
150	+30	+45	+45	+45	+45	+45	+45	+45	+45	+45	+45
180	+30	+45	+45	+45	+45	+45	+45	+45	+45	+45	+45
210	+30	+45	+45	+45	+45	+45	+45	+45	+45	+45	+45
240	+30	+45	+45	+45	+45	+45	+45	+45	+45	+45	+45
270	+30	+45	+45	+45	+45	+45	+45	+45	+45	+45	+45

Table 4.7: Samples for the truck.

First, we compute for both inputs the statistic of the β parameter which diminishes the square error of the fuzzy curves. The results with a level of confidence of 90% and a relative error of 10% are $\beta_x = 7.0571$ and $\beta_\theta = 27.5219$ and are plotted in figures 4.28 and 4.29, respectively.

Once we have the optimal values for the β parameter then the fuzzy curves are linearized until a high relative error of 20% is achieved in order to obtain a simple but intelligible model.

Observe that we have doubled the value of the *desired error* parameter in order to emphasize the fact that the method can easily be tuned based on each problem.

Furthermore, we have rounded the values for the input x to 10 meters, the values for the input θ to 30º and the values for the output ϕ to 5º.

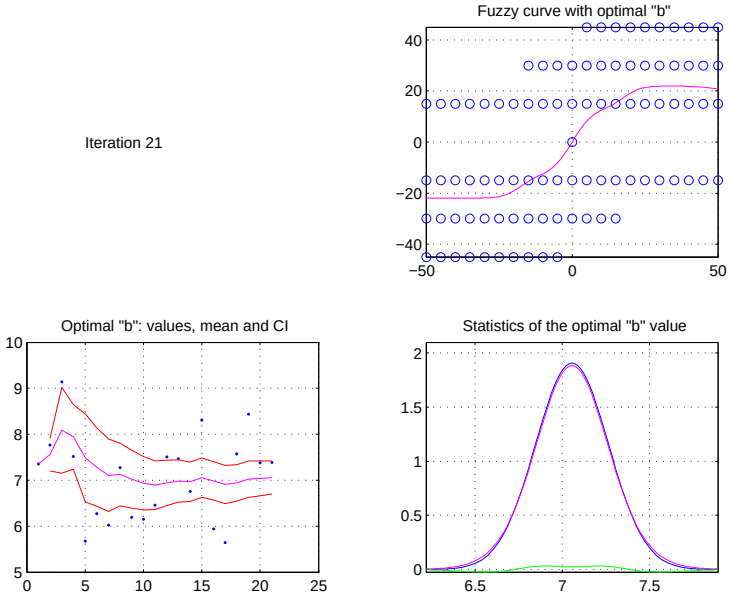


Figure 4.28: Statistics for the β_x parameter.

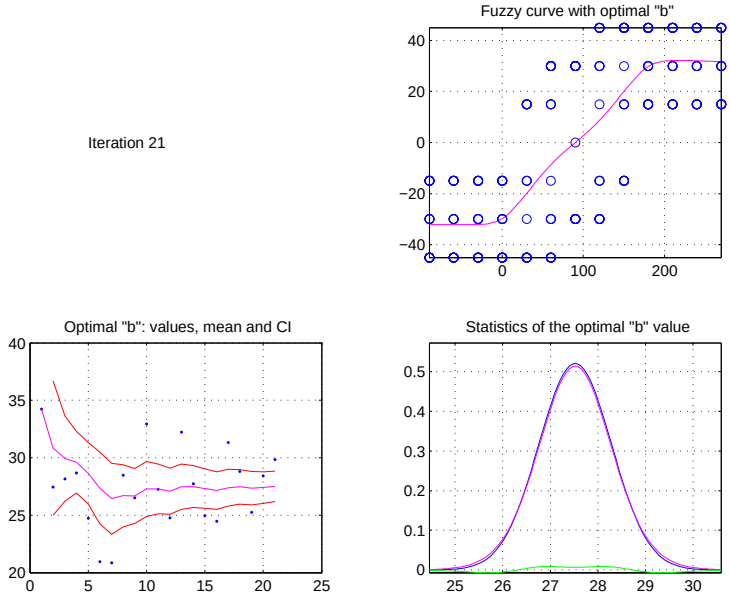


Figure 4.29: Statistics for the β_θ parameter.

We have plotted the resulting fuzzy sets in figures 4.30 and 4.31. They are placed at -50, -20, 20 and 50 for the input x and -90, 0, 180 and 270 for the input θ.

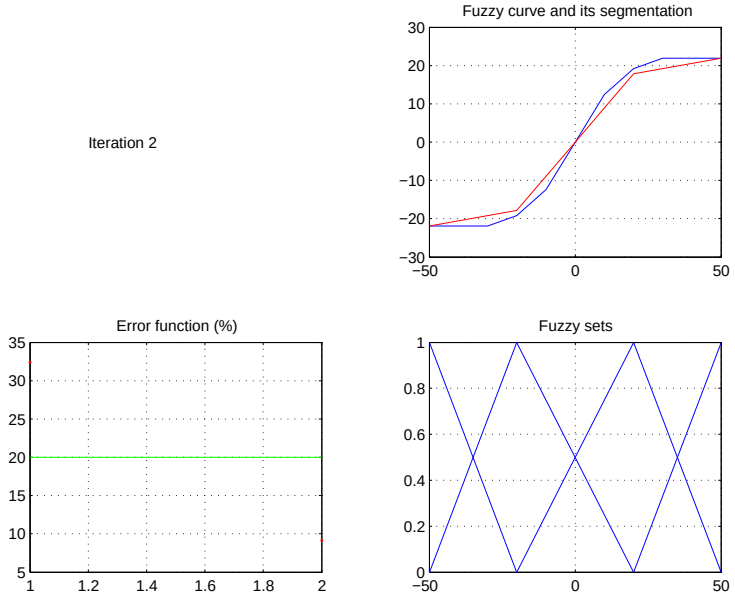


Figure 4.30: Linearization of the fuzzy curve for the input x.

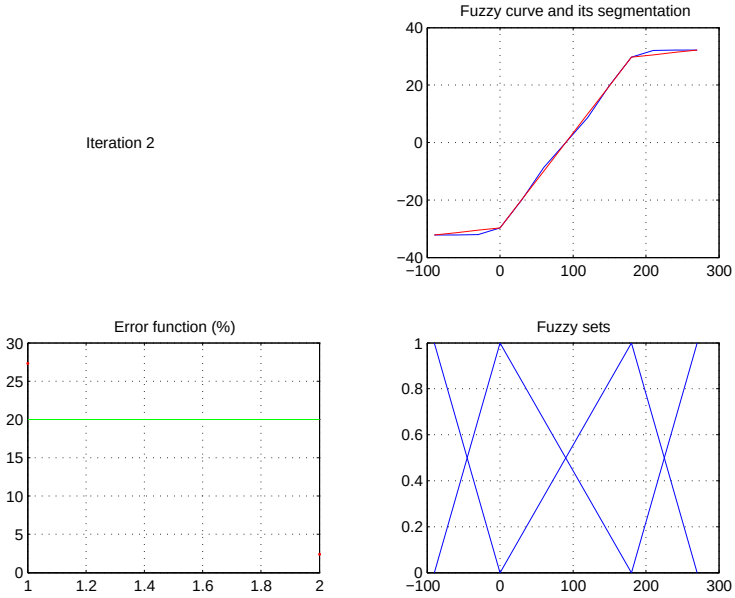


Figure 4.31: Linearization of the fuzzy curve for the input θ.

Finally the rule matrix with the previous linearization is given in table 4.8 which could be simplified if we consider trapezoidal membership functions. In this case we could join the sets of x placed at {-50,-20} as one trapezoidal

set and also those placed at {20,50}. Obviously, we could do the same for the variable θ with the sets placed at {-90,0} and also at {180,270}.

Rule matrix				
$\theta \setminus x$	**-50**	**-20**	**20**	**50**
-90	-45	-45	-15	-15
0	-45	-45	-15	-15
180	15	15	45	45
270	15	15	45	45

Table 4.8: Rules of the controller.

This controller has succesfully been tested with Simulink except for those situations where the y-position should have been considered as input because the original position of the truck was too close to the loading dock. Figure 4.32 shows the Simulink's program and figure 4.33 displays some simulations.

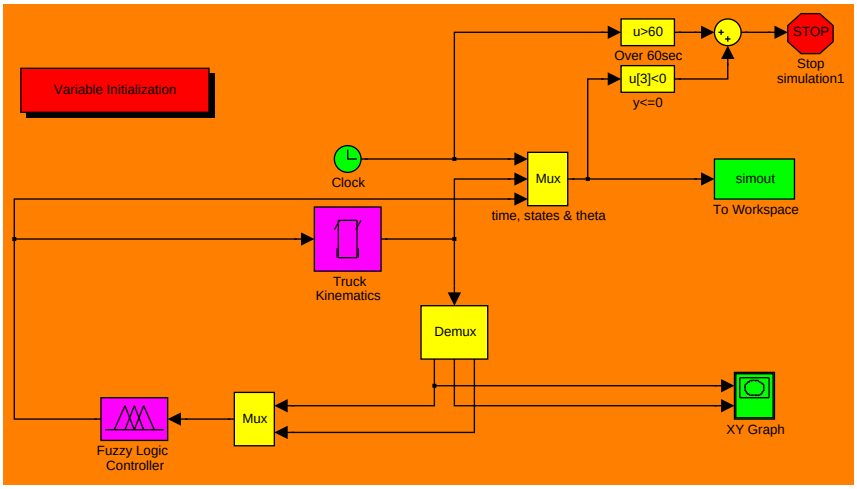


Figure 4.32: Truck's model with Simulink.

In this problem the linearized fuzzy curves are odd functions reason why we suggest choosing the mid point of the universe of scope as a fuzzy set. This option is always recommended in control systems in order to define the accomplishment of the target in spite of not having any error due to linearization in this point.

In this case the model would have had 5 sets for each input and 7 output sets whose rules are given in table 4.9.

This controller works exactly like the previous one and obviously it can also be simplified with trapezoidal membership functions in order to have 3 fuzzy sets for each input and 9 rules.

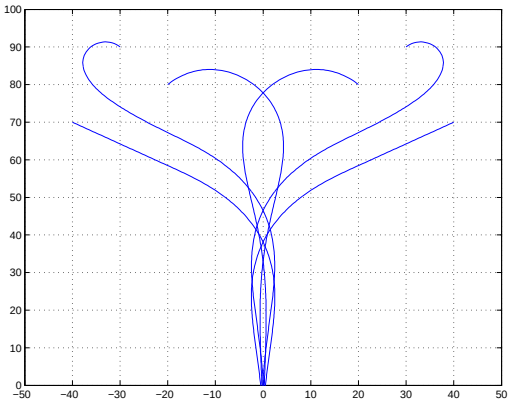


Figure 4.33: Simulations for the truck.

Rule matrix					
$\theta \setminus x$	**-50**	**-20**	**0**	**20**	**50**
-90	-45	-45	-30	-15	-15
0	-45	-45	-30	-15	-15
90	-30	-30	0	30	30
180	15	15	30	45	45
270	15	15	30	45	45

Table 4.9: Rules of the controller by considering odd fuzzy curves.

Now we will consider the truck and trailer backer-upper problem. In fact this problem has already been solved with different techniques and between them, we will take the efficient controller proposed by K. Chellapilla [16] which was developed with evolutionary programming. If we have decided to consider this controller is because it will help us to clarify one of the main purposes of our method: the capacity of *explaining* a complex process by means of an intelligible model.

The controller proposed by Chellapilla computes the desired control actions with the following expression[6]: (ADD (ADD (ADD (IFLTZ 0.378274 -0.419594 TANG) Y) DIFF) (ADD (MUL (SUB X -0.239431) (SUB DIFF TANG)) (DIV (MUL X TANG) (ADD 0.229157 (DIV (MUL (SUB TANG Y) (MUL Y (ADD (ATG (SUB Y TANG) (SUB X (DIV -0.447904 Y))) (DIV (IFLTZ X DIFF TANG) (MUL Y Y))))))) (IFLTZ (MUL TANG TANG) (ATG (IFLTZ (MUL (SUB (DIV Y X) (ADD TANG -0.432548)) (SUB (ADD (ATG (IFLTZ DIFF 0.650782 DIFF) (SUB Y -0.534967)) (DIV (MUL X -0.698540) (SUB 0.357931 -0.079263)))

[6]The variables are X, Y, TANG (θ_t) and DIFF ($\theta_c - \theta_t$). The operators are ADD, SUB (substraction), MUL (multiplication), DIV (division), ATG ($\tan^{-1}(y/x)$ producing an angle in the range $-\pi$ to $+\pi$) and IFLTZ (with 3 arguments in order to represent the operation *if* arg_1 *is less than zero then return* arg_2 *else return* arg_3).

(MUL (DIV -0.262260 (IFLTZ (DIV Y DIFF) (ADD (MUL (SUB (ATG TANG (SUB (SUB (IFLTZ Y TANG Y) (ADD (SUB TANG DIFF) (SUB Y TANG))) (DIV DIFF (SUB (ADD DIFF TANG) (MUL TANG -0.418766))))) (ATG TANG DIFF)) (ADD (DIV DIFF (IFLTZ TANG 0.156674 TANG)) (IFLTZ TANG (ADD -0.118834 0.893238) DIFF))) DIFF) X)) (SUB -0.345314 DIFF)) (DIV (ADD -0.629664 Y) (DIV Y -0.702695))))) TANG DIFF) DIFF) TANG))))))

Thus, we will try to obtain an intelligible model which explains how the previous controller works by generating several samples from it and by applying them to our method. Figure 4.34 shows the loading zone for this problem. Here we have changed the boundaries of the loading zone in comparison with the previous problem without the trailer in order to leave them like those which are used by Chellapilla [16].

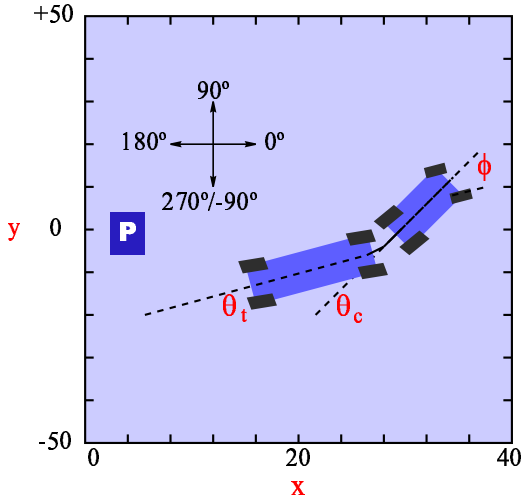


Figure 4.34: Loading zone for the truck and trailer.

The kinematics of this problem can be solved with the following equations:

$$A = r\cos\phi(t) \tag{4.4}$$

$$B = A\cos\left(\theta_c(t) - \theta_t(t)\right) \tag{4.5}$$

$$C = A\sin\left(\theta_c(t) - \theta_t(t)\right) \tag{4.6}$$

$$x(t+1) = x(t) - B\cos\theta_t(t) \tag{4.7}$$

$$y(t+1) = y(t) - B\sin\theta_t(t) \tag{4.8}$$

$$\theta_c(t+1) = \tan^{-1}\left(\frac{d_c\sin\theta_c(t) - r\cos\theta_c(t)\sin\phi(t)}{d_c\cos\theta_c(t) + r\sin\theta_c(t)\sin\phi(t)}\right) \tag{4.9}$$

$$\theta_t(t+1) = \tan^{-1}\left(\frac{d_s\sin\theta_c(t) - C\cos\theta_c(t)}{d_s\cos\theta_c(t) + C\sin\theta_c(t)}\right) \tag{4.10}$$

In these equations $\tan^{-1}(y/x)$ is the two-argument arctangent function producing an angle in the range $-\pi$ to $+\pi$. The length of the tractor d_c is 6 meters and the length of the trailer d_s is 14 meters. Time is measured in steps of 0.02 seconds and the speed of the truck is constant and the distance moved backward in one time step r is 0.2 meters.

First, we have sampled the controller proposed by Chellapilla. Due to the symmetry of the problem we have only considered the values with $y \geq 0$. Thus, we have varied the variable x from 0 to 40 meters with a step of 5 meters, the variable y from 0 to +50 meters with a step of 5 meters and the variables θ_c and θ_t from 0º to 120º with a step of +15º . The total amount of samples has been $9 \times 11 \times 9 \times 9 = 8019$.

With a *desired error* equal to 5% we have obtained the fuzzy sets plotted in figure 4.35 and the following 81 rules which relate them:

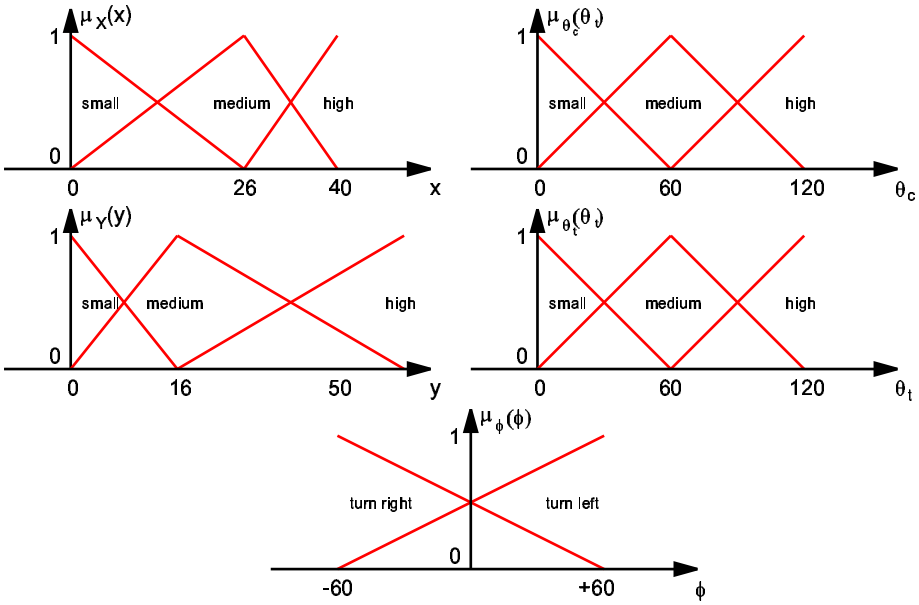


Figure 4.35: Fuzy sets for the truck and trailer problem.

if x is small and y is small and θ_c is small and θ_t is small then ϕ is turn left

if x is small and y is small and θ_c is small and θ_t is medium then ϕ is turn left

if x is small and y is small and θ_c is small and θ_t is high then ϕ is turn left

if x is small and y is small and θ_c is medium and θ_t is small then ϕ is turn left

if x is small and y is small and θ_c is medium and θ_t is medium then ϕ is turn left

if x is small and y is small and θ_c is medium and θ_t is high then ϕ is turn left

if x is small and y is small and θ_c is high and θ_t is small then ϕ is turn left

if x is small and y is small and θ_c is high and θ_t is medium then ϕ is turn left

if x is small and y is small and θ_c is high and θ_t is high then ϕ is turn left
if x is small and y is medium and θ_c is small and θ_t is small then ϕ is turn left
if x is small and y is medium and θ_c is small and θ_t is medium then ϕ is turn left
if x is small and y is medium and θ_c is small and θ_t is high then ϕ is turn left
if x is small and y is medium and θ_c is medium and θ_t is small then ϕ is turn left
if x is small and y is medium and θ_c is medium and θ_t is medium then ϕ is turn left
if x is small and y is medium and θ_c is medium and θ_t is high then ϕ is turn left
if x is small and y is medium and θ_c is high and θ_t is small then ϕ is turn left
if x is small and y is medium and θ_c is high and θ_t is medium then ϕ is turn left
if x is small and y is medium and θ_c is high and θ_t is high then ϕ is turn left
if x is small and y is high and θ_c is small and θ_t is small then ϕ is turn left
if x is small and y is high and θ_c is small and θ_t is medium then ϕ is turn left
if x is small and y is high and θ_c is small and θ_t is high then ϕ is turn left
if x is small and y is high and θ_c is medium and θ_t is small then ϕ is turn left
if x is small and y is high and θ_c is medium and θ_t is medium then ϕ is turn left
if x is small and y is high and θ_c is medium and θ_t is high then ϕ is turn left
if x is small and y is high and θ_c is high and θ_t is small then ϕ is turn left
if x is small and y is high and θ_c is high and θ_t is medium then ϕ is turn left
if x is small and y is high and θ_c is high and θ_t is high then ϕ is turn left
if x is medium and y is small and θ_c is small and θ_t is small then ϕ is turn left
if x is medium and y is small and θ_c is small and θ_t is medium then ϕ is turn left
if x is medium and y is small and θ_c is small and θ_t is high then ϕ is turn left
if x is medium and y is small and θ_c is medium and θ_t is small then ϕ is turn left
if x is medium and y is small and θ_c is medium and θ_t is medium then ϕ is turn left
if x is medium and y is small and θ_c is medium and θ_t is high then ϕ is turn left
if x is medium and y is small and θ_c is high and θ_t is small then ϕ is turn left
if x is medium and y is small and θ_c is high and θ_t is medium then ϕ is turn left
if x is medium and y is small and θ_c is high and θ_t is high then ϕ is turn left
if x is medium and y is medium and θ_c is small and θ_t is small then ϕ is turn left
if x is medium and y is medium and θ_c is small and θ_t is medium then ϕ is turn right
if x is medium and y is medium and θ_c is small and θ_t is high then ϕ is turn right
if x is medium and y is medium and θ_c is medium and θ_t is small then ϕ is turn left
if x is medium and y is medium and θ_c is medium and θ_t is medium then ϕ is turn right
if x is medium and y is medium and θ_c is medium and θ_t is high then ϕ is turn right
if x is medium and y is medium and θ_c is high and θ_t is small then ϕ is turn left
if x is medium and y is medium and θ_c is high and θ_t is medium then ϕ is turn left
if x is medium and y is medium and θ_c is high and θ_t is high then ϕ is turn right
if x is medium and y is high and θ_c is small and θ_t is small then ϕ is turn left
if x is medium and y is high and θ_c is small and θ_t is medium then ϕ is turn right
if x is medium and y is high and θ_c is small and θ_t is high then ϕ is turn right
if x is medium and y is high and θ_c is medium and θ_t is small then ϕ is turn left

if x is medium and y is high and θ_c is medium and θ_t is medium then ϕ is turn left
if x is medium and y is high and θ_c is medium and θ_t is high then ϕ is turn right
if x is medium and y is high and θ_c is high and θ_t is small then ϕ is turn left
if x is medium and y is high and θ_c is high and θ_t is medium then ϕ is turn left
if x is medium and y is high and θ_c is high and θ_t is high then ϕ is turn right
if x is high and y is small and θ_c is small and θ_t is small then ϕ is turn left
if x is high and y is small and θ_c is small and θ_t is medium then ϕ is turn left
if x is high and y is small and θ_c is small and θ_t is high then ϕ is turn left
if x is high and y is small and θ_c is medium and θ_t is small then ϕ is turn left
if x is high and y is small and θ_c is medium and θ_t is medium then ϕ is turn left
if x is high and y is small and θ_c is medium and θ_t is high then ϕ is turn left
if x is high and y is small and θ_c is high and θ_t is small then ϕ is turn left
if x is high and y is small and θ_c is high and θ_t is medium then ϕ is turn left
if x is high and y is small and θ_c is high and θ_t is high then ϕ is turn left
if x is high and y is medium and θ_c is small and θ_t is small then ϕ is turn left
if x is high and y is medium and θ_c is small and θ_t is medium then ϕ is turn right
if x is high and y is medium and θ_c is small and θ_t is high then ϕ is turn right
if x is high and y is medium and θ_c is medium and θ_t is small then ϕ is turn left
if x is high and y is medium and θ_c is medium and θ_t is medium then ϕ is turn right
if x is high and y is medium and θ_c is medium and θ_t is high then ϕ is turn right
if x is high and y is medium and θ_c is high and θ_t is small then ϕ is turn left
if x is high and y is medium and θ_c is high and θ_t is medium then ϕ is turn left
if x is high and y is medium and θ_c is high and θ_t is high then ϕ is turn right
if x is high and y is high and θ_c is small and θ_t is small then ϕ is turn left
if x is high and y is high and θ_c is small and θ_t is medium then ϕ is turn right
if x is high and y is high and θ_c is small and θ_t is high then ϕ is turn right
if x is high and y is high and θ_c is medium and θ_t is small then ϕ is turn left
if x is high and y is high and θ_c is medium and θ_t is medium then ϕ is turn left
if x is high and y is high and θ_c is medium and θ_t is high then ϕ is turn right
if x is high and y is high and θ_c is high and θ_t is small then ϕ is turn left
if x is high and y is high and θ_c is high and θ_t is medium then ϕ is turn left
if x is high and y is high and θ_c is high and θ_t is high then ϕ is turn right

We have observed how this fuzzy controller works in a very similar way to the one proposed by Chellapilla. Figure 4.36 shows two cases with initial values $(x = 20, y = 50, \theta_c = \theta_t = 90)$ and also $(x = 40, y = 50, \theta_c = \theta_t = 90)$ where the position of the truck is plotted in red with the original controller and in blue with the fuzzy controller. These are the same cases tested by Chellapilla apart from some other cases which are not considered here because

they produce *jack-knifing*[7] with both controllers.

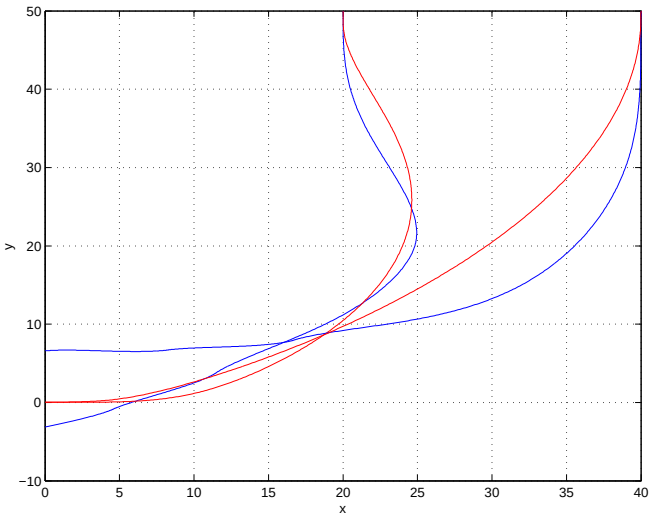


Figure 4.36: Simulations for the truck and trailer.

Despite not working exactly like the original controller, the most important conclusion is that our method explains in a very clear manner how the original controller works. Recall that we are interested in intelligible models in spite of its exactitude. Anyway our solution only gives intelligible rules in a local manner (rule by rule) but without assuring the intelligibility of the whole linguistic explanation if there are many rules. Later in the conclusions, we will review this aspect.

4.4.2 Ball and beam

We can repeat the last application by building a fuzzy controller from the samples of a more complex fuzzy controller covering the whole input space. We will demonstrate the capabilities of the proposed method in order to simplify the original controller with the popular ball and beam problem and to improve the intelligibility of the required control actions. Due to its non-linearity, this problem has been addressed quite often in control literature.

The problem plotted in figure 4.37 consists in placing a ball to any desired location on a horizontal beam by controlling its angle.

[7]Jack Knife is a term applied to the dangerous situation when a large 18 wheel truck and its trailer go into a skid and the trailer swings out and stops to form an angle of 90 degrees with each other. This term comes from a description of how the blade of a jack knife forms the angle with its protective handle. In this situation the vehicle is likely to roll over. For this reason we will not consider the cases for which $|\theta_c - \theta_t| \geq 90$.

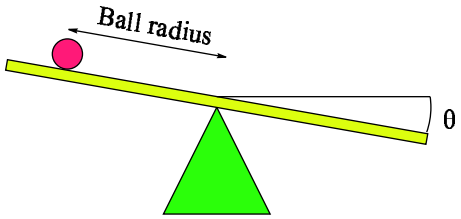


Figure 4.37: Problem statement.

This system can be described with the state space model proposed by Hauser, Sastry and Kokotović [39]. If r is the radius of the ball from the center of the beam and θ is the angle of the beam, the dynamics of the system can be expressed with equation 4.11

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B\left(x_1 {x_4}^2 - G\sin\left(x_3\right)\right) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{4.11}$$

where $(x_1, x_2, x_3, x_4) = \left(r, \dot{r}, \theta, \dot{\theta}\right)$, G represents the gravity acceleration and B=0.7143. We desire to bring the ball radius $y = x_1 = r$ to any set point by controlling the torque of the beam $u = \ddot{\theta}$. The set point can be either a constant or a function of time.

In fact this system and a possible controller have already been developed with Simulink in the Fuzzy Logic Toolbox for Matlab (figure 4.38). From a prompt window the user can choose between a sinusoid, square or saw wave or even any desired set point with the mouse (figure 4.39).

The original fuzzy controller proposed in the Fuzzy Logic Toolbox is a Takagi-Sugeno system with 4 inputs: $in_1 = x_1 - x_{1_{target}}$, $in_2 = x_2$, $in_3 = x_3$, $in_4 = x_4$; 2 fuzzy sets for each one which are plotted in figure 4.40 and 16 rules given in equation 4.12 by using a matrix format in order to compact them[8].

[8]For example the 5-th rule says that "if in_1 is Set_{1_1} and in_2 is Set_{2_2} and in_3 is Set_{3_1} and in_4 is Set_{4_1} then θ is $0.734+2.234\times\text{in}_1-12.853\times\text{in}_2-6.110\times\text{in}_3-1.034\times\text{in}_4$".

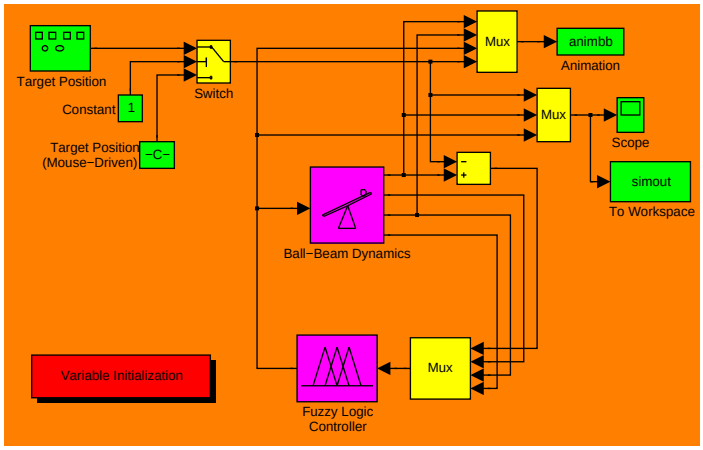


Figure 4.38: Description with Simulink of the ball and beam.

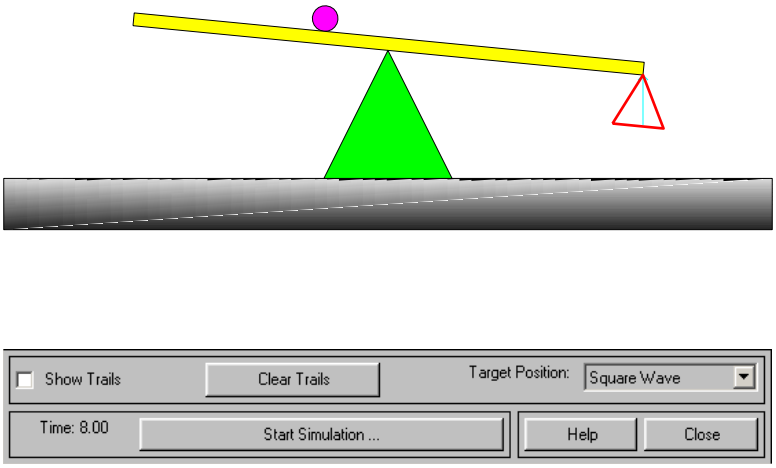


Figure 4.39: Simulator with Simulink for the ball and beam problem.

$$\begin{pmatrix} \mathrm{Set}_{1_1} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_1} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_1} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_1} & \mathrm{Set}_{4_2} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_1} \\ \mathrm{Set}_{1_2} & \mathrm{Set}_{2_2} & \mathrm{Set}_{3_2} & \mathrm{Set}_{4_2} \end{pmatrix} \begin{pmatrix} \mathrm{in}_1 \\ \mathrm{in}_2 \\ \mathrm{in}_3 \\ \mathrm{in}_3 \end{pmatrix} \rightarrow \begin{pmatrix} 1.015 & 2.234 & -12.665 & -4.046 & 0.026 \\ 1.161 & 1.969 & -9.396 & -6.165 & 0.474 \\ 1.506 & 2.234 & -12.990 & -1.865 & 1.426 \\ 0.734 & 1.969 & -9.381 & -4.688 & -0.880 \\ 0.734 & 2.234 & -12.853 & -6.110 & -1.034 \\ 1.413 & 1.969 & -9.485 & -6.592 & 1.159 \\ 1.225 & 2.234 & -12.801 & -3.929 & 0.366 \\ 0.985 & 1.969 & -9.291 & -5.115 & -0.195 \\ 0.985 & 1.969 & -9.292 & -5.115 & 0.195 \\ 1.225 & 2.234 & -12.802 & -3.929 & -0.366 \\ 1.413 & 1.969 & -9.485 & -6.592 & -1.159 \\ 0.734 & 2.234 & -12.853 & -6.110 & 1.034 \\ 0.734 & 1.969 & -9.381 & -4.688 & 0.880 \\ 1.506 & 2.234 & -12.990 & -1.865 & -1.426 \\ 1.161 & 1.969 & -9.396 & -6.165 & -0.474 \\ 1.015 & 2.234 & -12.665 & -4.046 & -0.026 \end{pmatrix} \begin{pmatrix} 1 \\ \mathrm{in}_1 \\ \mathrm{in}_2 \\ \mathrm{in}_3 \\ \mathrm{in}_4 \end{pmatrix} \tag{4.12}$$

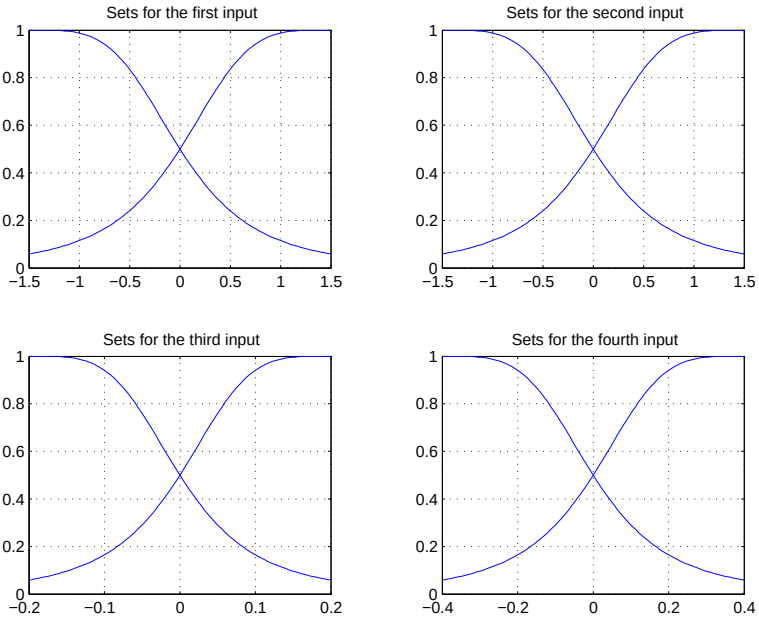


Figure 4.40: Fuzzy sets of the controller defined in the Fuzzy Logic Toolbox.

We have computed 625 samples from this controller equally spaced and covering the whole input space in order to test our method with $\text{in}_1 \in \{$-1.5, -0.5, 0, 0.5, 1.5$\}$, $\text{in}_2 \in \{$-1.5, -0.5, 0, 0.5, 1.5$\}$, $\text{in}_3 \in \{$-0.2, -0.1, 0, 0.1, 0.2$\}$ and $\text{in}_4 \in \{$-0.4, -0.2, 0, 0.2, 0.4$\}$.

After 20 executions with the *desired error* parameter equal to 3% the median values for the optimal β parameters were $\beta_{in_1} = 1.840$, $\beta_{in_2} = 1.410$, $\beta_{in_3} = 0.226$ and $\beta_{in_4} = 0.478$.

The linearization is accomplished with very few segments because the resulting fuzzy curves are *quite linear*. For example only two sets for each variable are necessary if the value for the ε parameter is equal or higher than 3%. Figure 4.41 shows these sets when $\varepsilon = 3\%$. In fact the NRMSE between the linearized function and the original fuzzy curves is only 0.74% for the first variable, 2.74% for the second one, 1.20% for the third one and 0.79% for the last one.

The difference between these sets and those proposed in the Matlab's Toolbox is its linearity and furthermore the accomplishment of the normal property of the fuzzy sets which is recommended in order to improve the intelligibility of the resulting model.

Finally the output sets and the rules obtained with our method are given

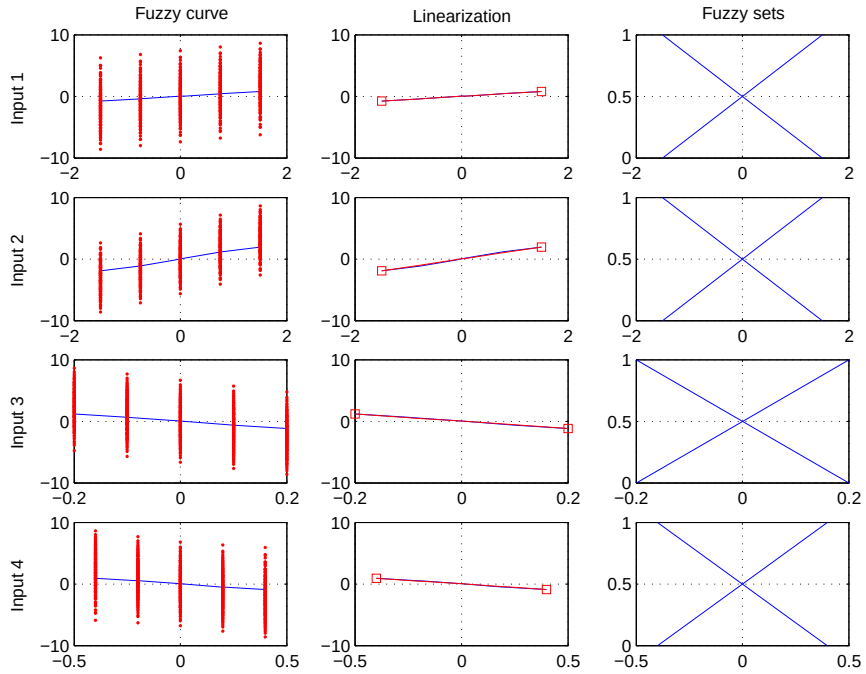


Figure 4.41: Linearized fuzzy curves when ε=3%.

in equation 4.13 by using the same matrix format than before.

$$\begin{pmatrix} \text{Set}_{1_1} & \text{Set}_{2_1} & \text{Set}_{3_1} & \text{Set}_{4_1} \\ \text{Set}_{1_1} & \text{Set}_{2_1} & \text{Set}_{3_1} & \text{Set}_{4_2} \\ \text{Set}_{1_1} & \text{Set}_{2_1} & \text{Set}_{3_2} & \text{Set}_{4_1} \\ \text{Set}_{1_1} & \text{Set}_{2_1} & \text{Set}_{3_2} & \text{Set}_{4_2} \\ \text{Set}_{1_1} & \text{Set}_{2_2} & \text{Set}_{3_1} & \text{Set}_{4_1} \\ \text{Set}_{1_1} & \text{Set}_{2_2} & \text{Set}_{3_1} & \text{Set}_{4_2} \\ \text{Set}_{1_1} & \text{Set}_{2_2} & \text{Set}_{3_2} & \text{Set}_{4_1} \\ \text{Set}_{1_1} & \text{Set}_{2_2} & \text{Set}_{3_2} & \text{Set}_{4_2} \\ \text{Set}_{1_2} & \text{Set}_{2_1} & \text{Set}_{3_1} & \text{Set}_{4_1} \\ \text{Set}_{1_2} & \text{Set}_{2_1} & \text{Set}_{3_1} & \text{Set}_{4_2} \\ \text{Set}_{1_2} & \text{Set}_{2_1} & \text{Set}_{3_2} & \text{Set}_{4_1} \\ \text{Set}_{1_2} & \text{Set}_{2_1} & \text{Set}_{3_2} & \text{Set}_{4_2} \\ \text{Set}_{1_2} & \text{Set}_{2_2} & \text{Set}_{3_1} & \text{Set}_{4_1} \\ \text{Set}_{1_2} & \text{Set}_{2_2} & \text{Set}_{3_1} & \text{Set}_{4_2} \\ \text{Set}_{1_2} & \text{Set}_{2_2} & \text{Set}_{3_2} & \text{Set}_{4_1} \\ \text{Set}_{1_2} & \text{Set}_{2_2} & \text{Set}_{3_2} & \text{Set}_{4_2} \end{pmatrix} \begin{pmatrix} \text{in}_1 \\ \text{in}_2 \\ \text{in}_3 \\ \text{in}_3 \end{pmatrix} \rightarrow \begin{pmatrix} \text{-0.6} \\ \text{-4.7} \\ \text{-5.9} \\ \text{-8.6} \\ \text{6.2} \\ \text{1.2} \\ \text{0.9} \\ \text{-2.5} \\ \text{2.5} \\ \text{-0.9} \\ \text{-1.2} \\ \text{-6.2} \\ \text{8.6} \\ \text{5.9} \\ \text{4.7} \\ \text{0.6} \end{pmatrix} \tag{4.13}$$

Our controller and the original controller have been tested with Simulink. We have tried to track a sine wave set point, a square wave and a also a saw

wave with an amplitude equal to the length of half beam (2 meters) and a period of 0.5 radians per second for the three cases.

The simulations for the first 40 seconds are plotted in figure 4.42 where the target position has been plotted in green, the ball's position with the Matlab's controller has been plotted in red and the ball's position with our controller has been plotted in blue.

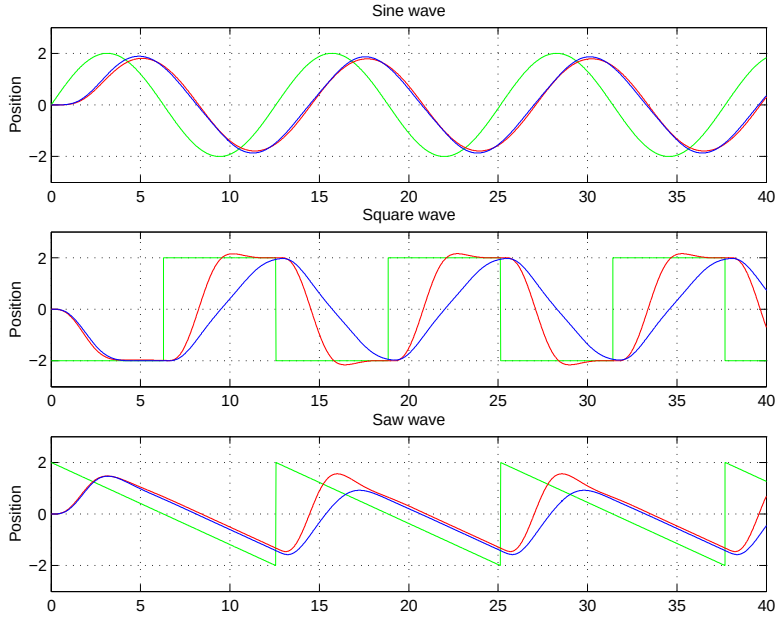


Figure 4.42: Simulations for the ball and beam.

Except for the sine wave target, where both results are very similar, in the other two simulations the original controller responds a little bit faster than our controller. Nevertheless observe how for this reason the ball would fall with the original controller with the square wave target. Therefore, the performance obtained with our controller is similar or even better in some cases by considering that is quite simple.

In any control application the output of the controller should be analyzed and for this reason this signal has been plotted for the three previous cases in figure 4.43.

In order to compare them the sum of their square values has been computed by being proportional to the necessary energy to control the beam. Due to a smoother signal, the results for the sine wave are better when the original control is considered (3.14) in comparison with our control (8.41), while our control is better in terms of this energy in the other two cases. For

the square wave this value is 1611 with the original control and 304 with our control. For the saw wave these values are 867 and 292, respectively.

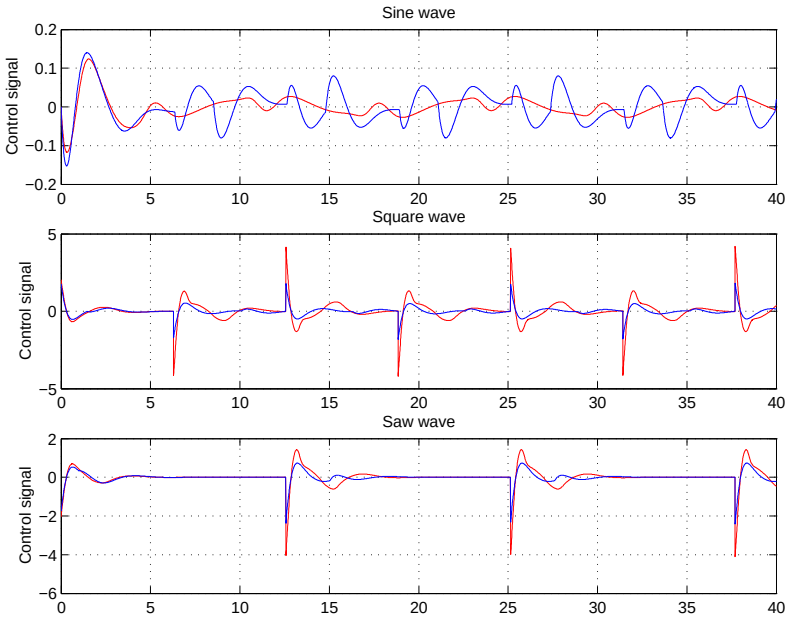


Figure 4.43: Control signals for the ball and beam.

Obviously a lower error could have been obtained by changing the *desired error* parameter but with a more complex model. For instance, if the *desired error* was only of 1%, then 4 sets for each input variable and a total amount of 256 rules would have been obtained. In this case the resulting control was very simlar to the original one.

4.4.3 Summary

Control problems have demanded for a long time a method able to obtain controllers from the input-output values given by an expert. In fact the foundation of the method we have exposed must be searched under this premise. In this section we have shown how our solution can give results very similar to the ones obtained with other techniques and furthermore by keeping a satisfactory intelligibility of the final controller which simplifies the typical latter adjustments.

4.5 Short examples

4.5.1 The electrical network maintenance

Here we will consider a real problem [11] in order to compare the performance of our method with other alternatives which are also focused on the intelligibility. Moreover we will take advantage of the data (495 samples with 2 inputs) in order to show some of the different alternatives to be considered with the proposed method. The problem states as follows [11]:

Sometimes, there is a need to measure the amount of electricity lines that an electric company owns. This measurement may be useful for several aspects such us the estimation of the maintenance costs of the network, which was the main goal of the problem presented here in Spain. High and medium voltage lines can be easily measured, but low voltage line is contained in cities and villages, and it would be very expensive to measure it. This kind of line used to be very convoluted and, in some cases, one company may serve more than 10,000 small nuclei. An indirect method to determine the length of line is needed. The problem involves finding a model that relates the total length of low voltage line installed in a rural town with the number of inhabitants in the town and the mean of the distances from the center of the town to the three furthest clients in it. This model will be used to estimate the total length of line being maintained.

Like in the former examples and due to the statistical analyses, several executions have been computed from which the median results are shown in table 4.10. Observe that we have considered four alternatives based on the technique used to define the output sets (Wang&Mendel or Takagi&Sugeno) and the clustering technique (Chiu's method or fuzzy C-means). For each alternative we have computed models by varying the *desired error* between 1% and 5%.

Observe how the results of the different alternatives are very similar. For this reason we usually prefer using the Wang&Mendel's method in order to define the possible output sets and the Chiu's clustering technique in order to cluster the final fuzzy sets because this is the fastest combination[9]. The Wang&Mendel's option is clearly faster than the Takagi&Sugeno's ε^2 optimal method while the Chiu's alternative is in general slightly faster than the fuzzy C-means but with a higher number of clusters.

Anyway, if we were interested in having less clusters like fuzzy C-means than we can increase the value of the parameter r_β slightly higher than the value of the parameter r_α. For instance Chiu suggests $r_\beta = 1.5 r_\alpha$ but we

[9] In this case the mean of the elapsed time when $\varepsilon = 1\%$ was 90.02sec with W&M+Chiu, 94.53sec with W&M+FCM, 213.80sec with T&S+Chiu and 230.75sec with T&S+FCM.

W&M + Chiu				
Error	**Sets**	**Rules**	**RMSE**	**Median model**
1%	35.5	44	601.79	6 × 7 in sets → 21 out sets (38 rules) with RMSE=593.3
2%	35	43	626.82	8 × 6 in sets → 18 out sets (38 rules) with RMSE=625.4
3%	39.5	52.5	638.32	9 × 7 in sets → 22 out sets (43 rules) with RMSE=637.9
4%	19	16	715.60	5 × 5 in sets → 15 out sets (24 rules) with RMSE=684.2
5%	13	8	746.98	4 × 2 in sets → 7 out sets (8 rules) with RMSE=747.0
W&M + FCM				
Error	**Sets**	**Rules**	**RMSE**	**Median model**
1%	27	44	600.79	7 × 6 in sets → 12 out sets (38 rules) with RMSE=598.5
2%	27	41.5	587.50	7 × 6 in sets → 12 out sets (35 rules) with RMSE=583.5
3%	27	40.5	583.19	8 × 6 in sets → 13 out sets (38 rules) with RMSE=581.9
4%	11.5	11	774.39	6 × 5 in sets → 11 out sets (28 rules) with RMSE=695.4
5%	9	6	775.15	3 × 2 in sets → 4 out sets (6 rules) with RMSE=775.2
T&S + Chiu				
Error	**Sets**	**Rules**	**RMSE**	**Median model**
1%	34	34	601.75	6 × 7 in sets → 21 out sets (34 rules) with RMSE=600.0
2%	35	36	623.25	8 × 7 in sets → 21 out sets (36 rules) with RMSE=621.1
3%	33.5	33	619.73	6 × 7 in sets → 19 out sets (30 rules) with RMSE=613.1
4%	8	4	633.20	2 × 2 in sets → 4 out sets (4 rules) with RMSE=633.2
5%	8	4	633.20	2 × 2 in sets → 4 out sets (4 rules) with RMSE=633.2
T&S + FCM				
Error	**Sets**	**Rules**	**RMSE**	**Median model**
1%	41.5	66.5	594.28	7 × 9 in sets → 15 out sets (48 rules) with RMSE=587.6
2%	31	42	595.34	8 × 8 in sets → 15 out sets (42 rules) with RMSE=594.4
3%	26	31	588.30	7 × 6 in sets → 12 out sets (31 rules) with RMSE=589.0
4%	9	6	778.17	3 × 2 in sets → 4 out sets (6 rules) with RMSE=775.2
5%	9.5	7	752.24	3 × 3 in sets → 4 out sets (8 rules) with RMSE=729.3

Table 4.10: Comparisons between different options in the real problem.

usually consider $r_\beta = r_\alpha$.

Figure 4.44 shows from different points of view the output values in blue together with original samples in red for the model with the *desired error* equal to 1% and with the W&M+Chiu's alternative.

Table 4.11 obtained from [11] shows the results with other common methods in order to observe how the our method assures a similar performance in spite of being very simple. For those methods where the data must be divided between a train set and a test set, the RMSE given in the table is the one obtained only with the train set.

Method	**Labels or sets**	**RMSE**
Linear regression	2	758.65
Exponential regression	2	682.27
2^{nd} order polynomial regression	6	686.95
3^{rd} order polynomial regression	10	686.93
3 layer perceptron 2-25-1	102	582.06
COR [11]	21	585.93

Table 4.11: Comparisons between different methods.

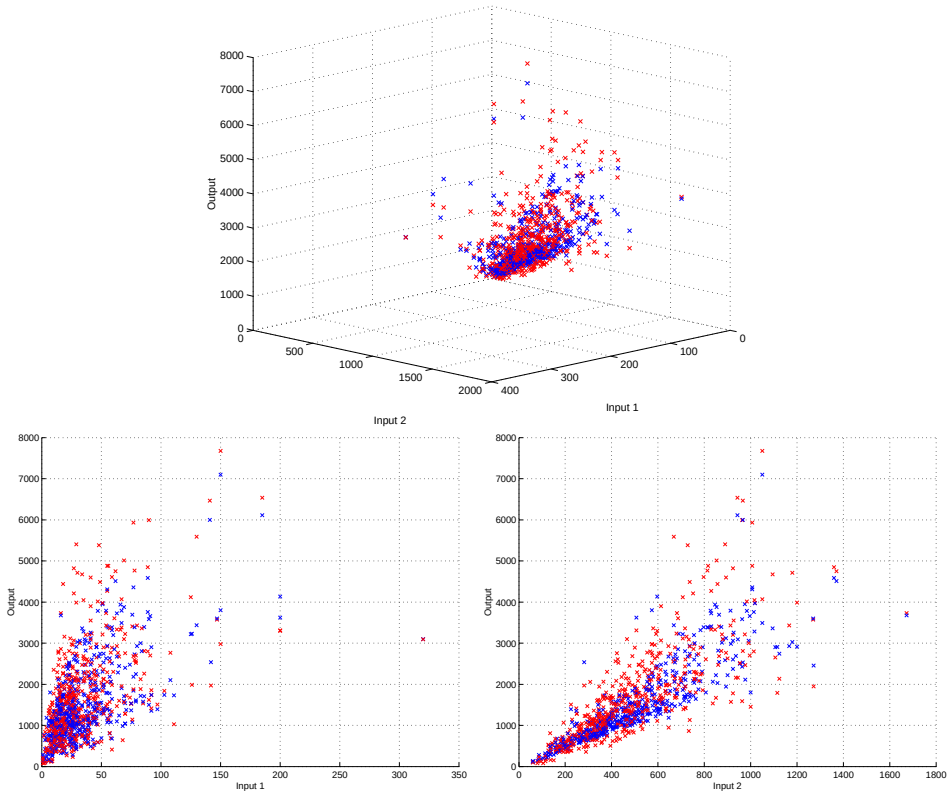


Figure 4.44: Output values in the real problem.

4.5.2 Reproduction of a fuzzy system

With this case we will study the capability of our method to reproduce a fuzzy system. This property is based on the generation of samples from a fuzzy model in order to evaluate if the method under test can give the same or at least a similar transfer function [111].

We will consider 100 equidistant values for each input from the fuzzy system given in figure 4.45 which was already used in [111].

We have considered four values for the *desired error* equal to 2%, 4%, 6% and 8%. The results are given in figures 4.46 and 4.47. Observe how the final transfer functions are similar to the original one.

The main difference between our models and the original one is the fact that we only work with triangular membership functions and consequently we need more sets for each variable. Nevertheless, observe in figure 4.48 how the model given in figure 4.46 with ε=2% could be simplified with trapezoidal sets given exactly the same structure.

In fact it is very easy to simplify results with trapezoidal sets. Once the

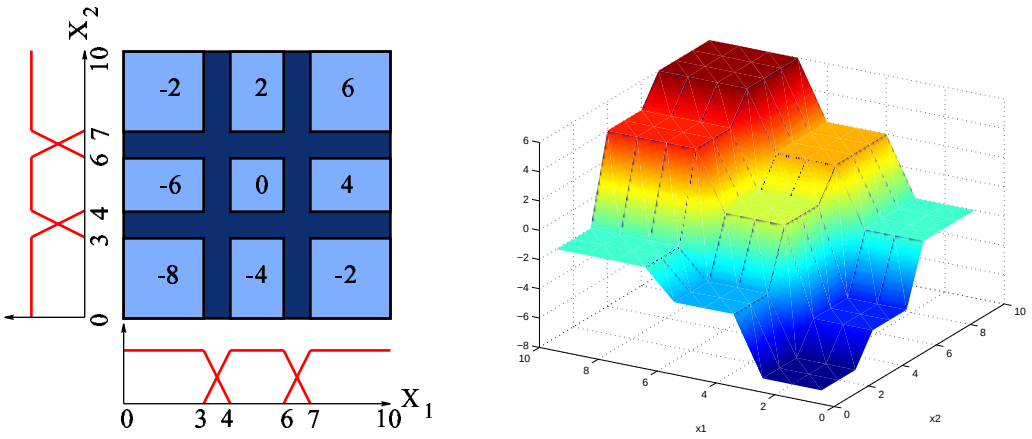
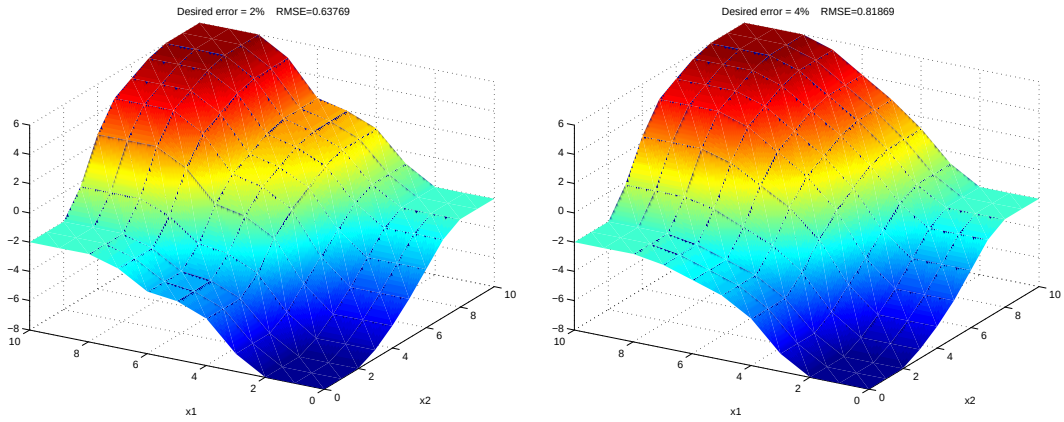


Figure 4.45: Original fuzzy system in order to study its reproductivity.

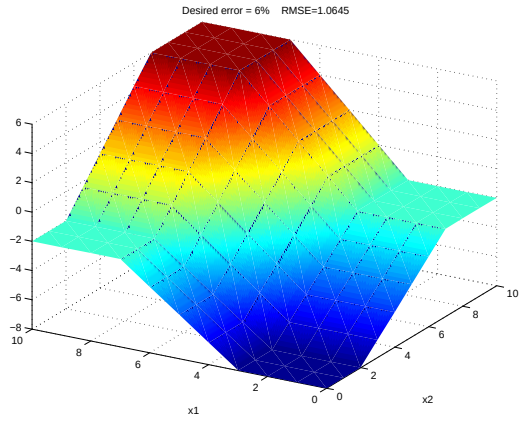


Rules with ε=2%					
$x_1 \backslash x_2$	**0.0**	**2.4**	**4.5**	**7.5**	**10.0**
0.0	-8.00	-8.00	-6.00	-2.00	-2.00
2.4	-8.00	-8.00	-6.00	-2.00	-2.00
4.4	-4.00	-4.00	0.00	2.00	2.00
5.8	-4.00	-4.00	0.00	2.00	2.00
7.5	-2.00	-2.00	4.00	6.00	6.00
10.00	-2.00	-2.00	4.00	6.00	6.00

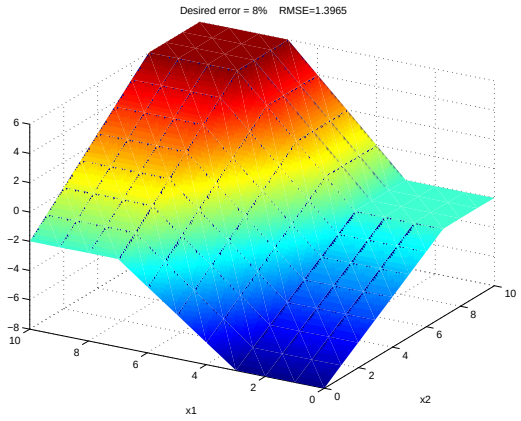
Rules with ε=4%					
$x_1 \backslash x_2$	**0.0**	**2.4**	**4.5**	**7.5**	**10.0**
0.0	-8.00	-8.00	-6.00	-2.00	-2.00
2.4	-8.00	-8.00	-6.00	-2.00	-2.00
4.4	-4.00	-4.00	0.00	2.00	2.00
7.5	-2.00	-2.00	4.00	6.00	6.00
10.00	-2.00	-2.00	4.00	6.00	6.00

Figure 4.46: Resulting models with ε=2% and ε=4%.

rule matrix with triangular sets is obtained we search, for each input set, if the output set assigned to its neighbor rule with the same input sets, apart from the one belonging to the current variable, is exactly the same. If this situation is repeated for all the rules with the same input values then both sets can be grouped into a trapezoidal one.



Rules with ε=6%			
$x_1 \backslash x_2$	**0.0**	**7.0**	**10.00**
0.0	-8.00	-2.00	-2.00
3.0	-8.00	-2.00	-2.00
7.0	-2.00	6.00	6.00
10.00	-2.00	6.00	6.00



Rules with ε=8%			
$x_1 \backslash x_2$	**0.0**	**7.0**	**10.00**
0.0	-8.00	-2.00	-2.00
3.0	-8.00	-2.00	-2.00
7.0	-2.00	6.00	6.00
10.00	-2.00	6.00	6.00

Figure 4.47: Resulting models with ε=6% and ε=8%.

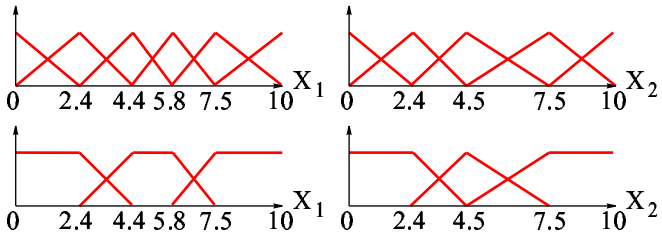


Figure 4.48: Trapezoidal membership functions from triangular ones.

4.5.3 Simplifying quasi-linear fuzzy systems

Some of the original fuzzy controllers were developed by copying the basis of classical control into FRBS. Thus, most of the original fuzzy controllers were PD-fuzzy controllers, PI-fuzzy controllers, PID-fuzzy controllers and so on. Here we will consider one of them (PD-fuzzy) in order to show how our method can simplify its structure without degrading its performance.

A proportional-derivative (PD) classical control considers two inputs: the error between the target and the current value and also the derivative of this error. The relation between these variables (e,d) and the control signal (u) is defined with a linear equation such as $u = K_e e + K_d d$. The constants K_e and K_d are adjusted depending on each problem.

Many people implements PD controls with fuzzy theory. Thus, they define for each input several fuzzy sets which are normally odd and equally

spaced along the universe of scope. Figure 4.49 shows an example of this kind of sets[10].

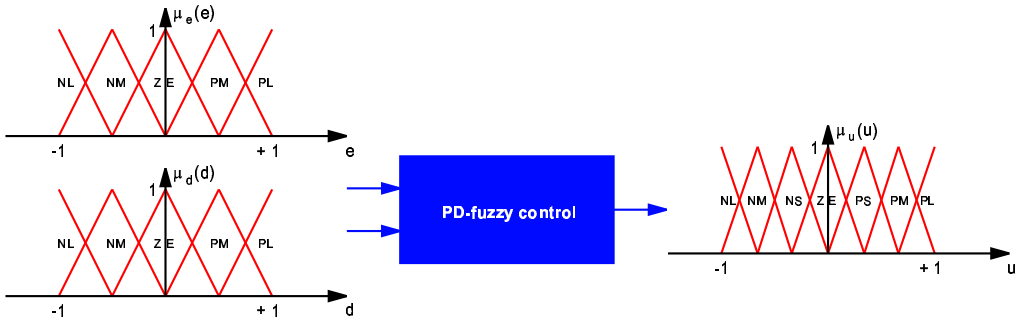


Figure 4.49: Fuzzy sets for the PD-fuzzy control.

The rule matrix which relates the input sets and the output sets is in general symmetrical. Table 4.12 shows the rule matrix for a PD-fuzzy with the previous fuzzy sets.

Rule matrix					
$d \setminus e$	**NL**	**NM**	**ZE**	**PM**	**PL**
NL	NL	NL	NM	NS	ZE
NM	NL	NM	NS	ZE	PS
ZE	NM	NS	ZE	PS	PM
PM	NS	ZE	PS	PM	PL
PL	ZE	PS	PM	PL	PL

Table 4.12: Rules of the PD-fuzzy control.

The transfer function of this FRBS is plotted in figure 4.50 where we can observe how this is a quasi-linear function.

The fact that our method gives simple FRBS by linearizing the set of samples, suggests that it may be applied in order to simplify the original PD-fuzzy control. Therefore, we will show how in this case the number of fuzzy sets is not very important because the same transfer function or at least a very similar one can be obtained with a simpler and more intelligible model.

We have considered three values for the *desired error* equal to 1%, 5% and 10%. The results are given in figures 4.51 and 4.52. Observe how the final transfer functions are simpler but very similar to the original one.

[10]NL means *negative large*, NM means *negative medium*, NS means *negative small*, ZE means *zero*, PS means *positive small*, PM means *positive medium* and PL means *positive large*.

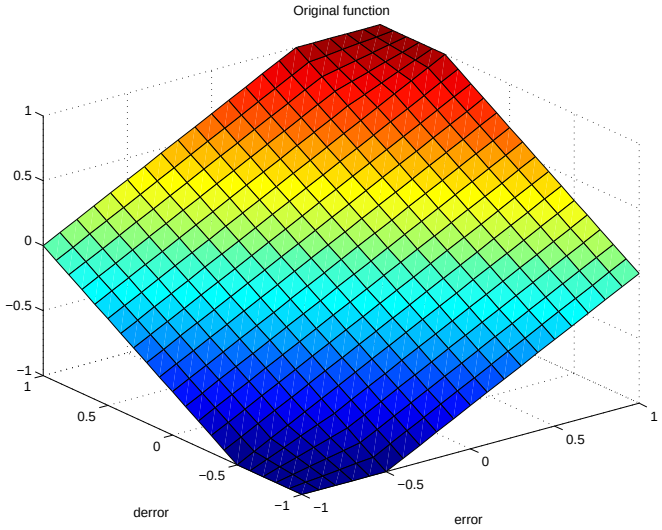
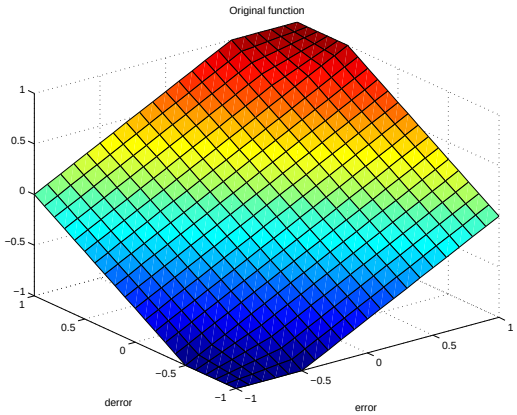
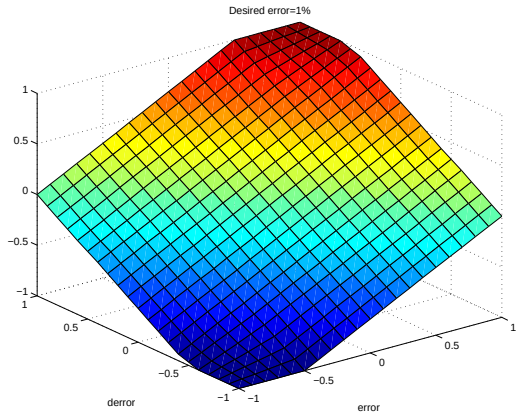


Figure 4.50: Original transfer function of the PD-fuzzy control.



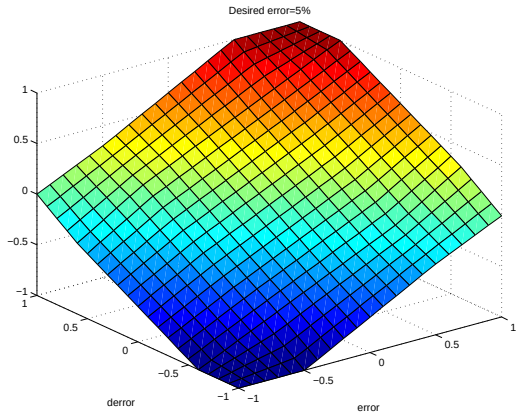


Original rules					
$d \setminus e$	**NL**	**NM**	**ZE**	**PM**	**PL**
NL	NL	NL	NM	NS	ZE
NM	NL	NM	NS	ZE	PS
ZE	NM	NS	ZE	PS	PM
PM	NS	ZE	PS	PM	PL
PL	ZE	PS	PM	PL	PL

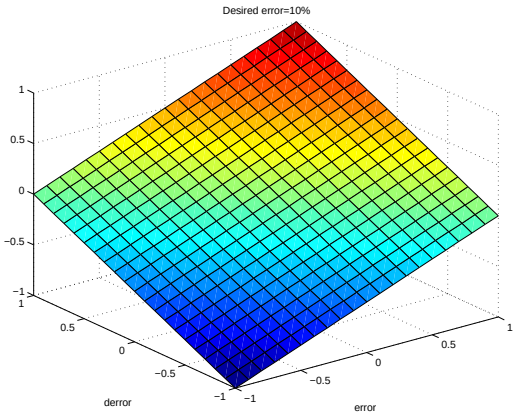
Rules with ε=1%						
$d \setminus e$	**-1.0**	**-0.7**	**-0.5**	**0.5**	**0.7**	**1.0**
-1.0	-1.00	-1.00	-1.00	-0.33	-0.20	0.00
-0.6	-1.00	-0.84	-0.73	-0.07	0.07	0.27
-0.4	-0.93	-0.73	-0.60	0.07	0.20	0.40
0.4	-0.40	-0.20	-0.07	0.60	0.73	0.93
0.6	-0.27	-0.07	0.07	0.73	0.84	1.00
1.0	0.00	0.20	0.33	1.00	1.00	1.00

Figure 4.51: Original and resulting model with ε=1%.

The model with ε=1% has 6 sets for each input, 21 output sets, 36 rules and a NRMSE=0.48%; the model with ε=5% has 4 sets for each input, 9 output sets, 16 rules and a NRMSE=3.66%; and the model with ε=10% has 2 sets for each input, 3 output sets, 4 rules and a NRMSE=22.06%.



Rules with ε=5%				
$d \setminus e$	**-1.0**	**-0.5**	**0.5**	**1.0**
-1.0	-1.0	-1.0	-0.3	0.0
-0.6	-1.0	-0.7	-0.1	0.3
-0.6	-0.3	0.1	0.7	1.0
1.0	0.0	0.3	1.0	1.0



Rules with ε=10%		
$d \setminus e$	**-1.0**	**1.0**
-1.0	-1.0	0.0
1.0	0.0	1.0

Figure 4.52: Resulting models with ε=5% and ε=10%.

Observe how we can play with the trade-off between accuracy and intelligibility by varying the *desired error* parameter. The fact is that we can simplify the necessary rules to *explain* how the PD-fuzzy works without degrading significantly its transfer function.

In this example we have not considered the different treatment of the fuzzy curves if they are odd. If we had considered this option we would have obtained very similar results in every execution which would have differed only by the inclusion of a fuzzy set placed at 0 for each input. The overall error is very similar or even the same, depending on the statistical result of every execution.

4.5.4 Comparisons with ANFIS

There is no doubt that the most popular identification method by means of fuzzy logic has been ANFIS for a long time. ANFIS (artificial neuro-fuzzy inference system) was presented in 1993 [47, 50] by J.R. Jang who proposed a hybrid learning rule in order to improve the learning process based on gradient descent techniques. Its main feature was getting the best model in terms of square error, avoiding becoming trapped in local minima. Thus, ANFIS applies gradient descent together with least-squares in order to identify the parameters in an adaptative network until a certain error has been achieved. In spite of the latter investigations [48, 49], the model

structure must be introduced in advance (number of inputs and number of sets) and the process can spend a long time if many samples must be considered, basically due to the least-squares computations.

It has been for years one of the best methods because most investigations were focused on the error but not on the intelligibility of the resulting models. In fact ANFIS is usually the technique we prefer when we search a model with a high accuracy. But in spite of being based on a completely different objective from our current investigation, we have considered interesting to do some comparisons between ANFIS and our method in order to emphasize their different targets.

Here we will consider a circular function of three and four variables with a different numbers of samples which have been generated randomly:

$$y = \sqrt{1 - x_1^2 - x_2^2 - x_3^2} \quad \text{with } 10^3, 10^4 \text{ and } 10^5 \text{ random samples in } \mathbb{R}$$

$$y = \sqrt{1 - x_1^2 - x_2^2 - x_3^2 - x_4^2} \quad \text{with } 10^4, 10^5 \text{ and } 10^6 \text{ random samples in } \mathbb{R}$$

We have done several executions from which their median values have been considered in order to give results independent from the random generation of the samples. For each simulation, first we have applied our method and when the model has been obtained then we have applied ANFIS with the same fuzzy structure (number of input sets for each input and singleton sets as outputs). Results from different cases are given in table 4.13.

	Our method			**ANFIS**		
Settings	**Time (sec)**	**RMSE**	**Sets**	**Time (sec)**	**RMSE**	**Sets**
3 inputs with 10^3 samples and ε=20%	3.585	28.67%	13	0.201	16.14%	26
3 inputs with 10^4 samples and ε=20%	5.968	23.43%	13	3.184	10.92%	46
3 inputs with 10^5 samples and ε=20%	32.376	26.43%	13	129.727	10.85%	46
4 inputs with 10^4 samples and ε=20%	8.523	27.26%	16	9.073	15.19%	65
4 inputs with 10^5 samples and ε=20%	55.591	30.51%	15	188.882	18.07%	46
4 inputs with 10^6 samples and ε=20%	532.176	33.62%	15	Stopped after three days		
3 inputs with 10^3 samples and ε=10%	6.289	18.08%	22	0.921	7.35%	88
3 inputs with 10^4 samples and ε=10%	8.582	13.32%	23	10.495	7.03%	93
3 inputs with 10^5 samples and ε=10%	53.847	14.27%	21	201.520	7.11%	93
4 inputs with 10^4 samples and ε=10%	15.161	23.12%	20	1105.470	9.88%	134
4 inputs with 10^5 samples and ε=10%	127.363	26.33%	22	Stopped after three days		
4 inputs with 10^6 samples and ε=10%	1020.388	31.47%	21	Stopped after three days		

Table 4.13: Comparisons between ANFIS and our method which ends by watching the error of the linearized fuzzy curves.

Observe how ANFIS achieves always better results in terms of the error because this is its main purpose but many more sets are in general necessary and thus, the final model is less comprehensible. Furthermore, it may give

absurd models by placing the sets outside the logical range of possible values because it does not consider the boundaries of the variables.

On the other hand our method is faster basically if many samples are given but still gives a satisfactory error with quite less fuzzy sets. For example if $\varepsilon = 10\%$, the case with four inputs and 10^6 samples took half an hour with our method but more than three days with ANFIS. In fact we stopped ANFIS in some executions after being computing the model for more than 3 days (more than 25000 seconds), basically when we have a high number of samples.

Nevertheless recall that our method ends either for a global NRMSE of for an error of all the linearized fuzzy curves lower than the *desired error*. In most cases this last condition is the one that prevails and then the final global NRMSE is higher than the *desired error* parameter. In order to compare our method with ANFIS, which always ends when it reaches a certain global error, we have considered some executions which stop only if a certain global NRMSE is achieved. The results are given in table 4.14.

	Our method			**ANFIS**		
Settings	**Time (sec)**	**RMSE**	**Sets**	**Time (sec)**	**RMSE**	**Sets**
3 inputs with 10^3 samples and ε=20%	5.277	13.14%	23	0.201	16.14%	26
3 inputs with 10^4 samples and ε=20%	17.034	11.25%	24	3.184	10.92%	46
3 inputs with 10^5 samples and ε=20%	151.167	11.60%	25	129.727	10.85%	46
4 inputs with 10^4 samples and ε=20%	59.105	13.11%	26	9.073	15.19%	65
4 inputs with 10^5 samples and ε=20%	522.481	13.10%	27	188.882	18.07%	46
4 inputs with 10^6 samples and ε=20%	5072.163	13.22%	27	Stopped after three days		
3 inputs with 10^3 samples and ε=10%	13.660	9.50%	36	0.921	7.35%	88
3 inputs with 10^4 samples and ε=10%	46.737	8.16%	37	10.495	7.03%	93
3 inputs with 10^5 samples and ε=10%	359.557	7.69%	38	201.520	7.11%	93
4 inputs with 10^4 samples and ε=10%	238.122	11.13%	39	1105.470	9.88%	134
4 inputs with 10^5 samples and ε=10%	2612.606	10.25%	41	Stopped after three days		
4 inputs with 10^6 samples and ε=10%	22173.123	10.00%	42	Stopped after three days		

Table 4.14: Comparisons between ANFIS and our method which ends by watching only the global NRMSE.

We conclude that we can reach also the same or similar *error* in comparison with ANFIS but with a more comprehensible model by having quite less sets (linguistic labels). Furthermore our method is clearly faster than ANFIS when a high number of samples are considered.