Constructive On-line Learning for a Neuro-Fuzzy Network with Fuzzy Sets obtained by Delaunay Triangulation

C. Pereira[1,2], A. Dourado[1], R. Babuska[3]
[1]CISUC – Centro de Informática e Sistemas da Universidade de Coimbra, POLO II – Universidade de Coimbra, Pinhal de Marrocos, 3030 Coimbra, Portugal
Phone: +351 239 790000, Fax: +351 239 701266
[2]ISEC – Instituto Superior de Engenharia de Coimbra, Quinta da Nora, 3030 Coimbra, Portugal
Phone: +351 239 790200, Fax: +351 239 790270
[3]Delft University of Technology, Department of Information Technology and Systems, Control Engineering Laboratory, P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract - This paper addresses the design and gradual building of a rule based neuro-fuzzy network using piecewise linear multidimensional membership functions obtained by Delaunay partition of the input space. On-line growing and pruning techniques are used to obtain a parsimonious structure. The proposed network is shown to be useful in approximating unknown nonlinearities of dynamic systems. A control framework is applied, taking advantage of the piecewise linear property of the model. For each simplex, the local inverse model can easily be calculated. The operation of this adaptive control scheme using the on-line constructive algorithm and the inverse of the local linear model is demonstrated using a simulation example and a laboratory scale process.

Index Terms - Neuro-fuzzy networks, Delaunay triangulation, constructive learning, non-linear control, inverse model control.

1. INTRODUCTION

Artificial neural networks (NN) have become popular models for a broad variety of modelling and control problems. However, NN adaptive controllers cannot incorporate linguistic system descriptions, because it is difficult to extract knowledge from them in a comprehensible way. On the other hand, fuzzy inference systems represent knowledge using linguistic labels and may easily be interpreted. Recently, the neuro-fuzzy approach has become a popular research area. In contrast to pure neural or fuzzy methods, the neuro-fuzzy method possesses both of their advantages the learning and adaptation capabilities of the neural networks and also providing an inference approach that enables approximate reasoning capabilities [1].

One of the most important issues in neuro-fuzzy modelling is the search for an optimal partition of the input space [2]. The way the input space is partitioned determines the number of rules as well as the number of fuzzy sets on the universe of discourse of each input variable. Often, lattice networks are used, however these models do not comply with the principle of parsimony, which demands that the model should represent a given set of data with the fewest possible parameters. One effective approach to parsimonious interpolation networks is to allow the nodes to be arbitrarily distributed. The node density can then be adjusted to the local complexity of the represented function.

A good way to reduce the problem's dimension is to use multidimensional fuzzy sets. The Delaunay partition of the input space is an efficient method to construct the multidimensional fuzzy sets [3]. Two majors benefits of this are that the set of Delaunay simplices is non-intersecting and one can prove that if linear interpolation is applied, the worst case approximation error using the Delaunay triangulation is lowest compared with any other triangulation of the input space [4]. Also, the Delaunay partition of the input space has been shown to be more appropriate than lattice networks in obtaining a parsimonious and smooth model.

Usually, the approaches to generating fuzzy rules from input-output data consist of two learning phases, the structure learning phase and the parameter learning phase. Most of the times, both phases are performed off-line. One of the objectives of this paper is to provide a mechanism in which both structure and parameter learning are performed on-line. A constructive adaptation law, based on a minimal resource allocating algorithm [5] is applied in order to on-line adjust the structure and parameters of the neuro-fuzzy network.

Concerning control applications, in most neuro-fuzzy based control schemes adaptation laws are derived using a fixed structure network. The result is that these fixed structures often need a large number of elements even for simple problems. Thus, the use of variable structure neural networks can solve this problem [6]. In the current paper, the network structure is updated on-line; at each instant a piecewise linear model of the plant is obtained. Then, it becomes part of a model-based control scheme. The model inverse control technique is considered here. It takes advantage of the piecewise linear property of the model, which allows an efficient on-line computation of the inverse model.

The remainder of this paper is organised as follows. Section 2 presents network structure and the fuzzy sets

construction using the Delaunay partition of the input space. The last paragraph describes the on-line constructive algorithm. Modelling results to some benchmarks are presented in Section 3. In Section 4 a framework is proposed to inverse model based control and experimental results are presented using simulations and experiments on a laboratory scale process. Finally, conclusions are given in Section 5.

2. STRUCTURE AND PARAMETER LEARNING

A local basis function neural network has been adopted. The model belongs to a general class of function approximators, called the basis function expansion, taking the form:

$$y = \sum_{i=1}^{m} f_i(\mathbf{x}) q_i \qquad (1)$$

where f_i are the basis function and q_i are the consequent parameters. The well-known radial basis function (RBF) neural networks belong to this class of models. These networks are under certain conditions functionally equivalent to the fuzzy model with constant consequent parts [7]:

$$R_i: \text{ If } \mathbf{x} \text{ is } A_i \text{ then } y = q_i, \quad i = 1,2,...,m. \qquad (2)$$

where $\mathbf{x}$ is the antecedent linguistic variable, which represents the input to the fuzzy system, and y is the consequent variable representing the output. A_i is the linguistic term defined by the multivariate membership function $m_{A_i}(\mathbf{x}): X \to [0,1]$, and m denotes the number of rules. The inference is given by:

$$y = \frac{\sum_{i=1}^{m} m_{A_i}(\mathbf{x}) q_i}{\sum_{i=1}^{m} m_{A_i}(\mathbf{x})} \qquad (3)$$

In this framework the linguistic terms satisfy the property of coverage and the sum of membership degrees equals one. Then the denominator in (3) can be dropped.

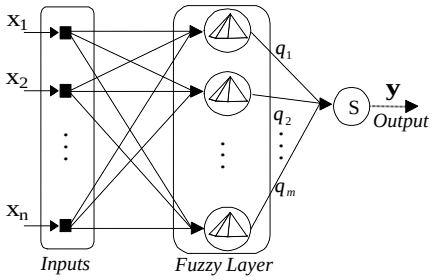

Figure 1. Network structure with multidimensional fuzzy sets.

The functional equivalence implies that the learning algorithms of RBF networks can be used to train models of this type. In most cases, the RBF training proceeds in two phases. The basis functions are constructed (e.g. by clustering the input training vectors in the input space) and the consequent parameters are estimated from the data using least-squares methods. In a similar way, TS fuzzy models are constructed by product-space fuzzy clustering. The way the input space is partitioned determines the number of fuzzy rules. Typically, a model structure is created from numerical data in the form of IF-THEN rules. However these automated modelling techniques may introduce unnecessary redundancy into the rule base. It is of great interest to reduce the number of fuzzy rules. A survey of some methods proposed in the literature is given in [3]. Here, the technique proposed uses multidimensional fuzzy sets obtained by Delaunay triangulation combined with growing and pruning techniques to reduce complexity.

2.1 Partitioning the input space

Consider $\mathrm{P} = \{P_1, P_2, ..., P_m\}$, a set of m points in an n-dimensional Euclidean input space X_n, where each point P_j is defined by its coordinate vector $[x_{j1}, x_{j2}, ..., x_{jn}]^T$. The Voronoi diagram associated with P is a sequence of convex polyhedra $V_1, V_2, ..., V_m$ covering the input space X_n where V_l consists of all the points $X \in \mathrm{X}_n$ that have P_l as a nearest point in the set P:

$$\forall X \in \mathrm{X}_n, X \in V_l \text{ iff } \forall P_j \in \mathrm{P}, \; \partial(X, P_l) \le \partial(X, P_j), \; j \ne l, l = 1,2,...,m \qquad (4)$$

where ∂ denotes the Euclidean distance $\partial(X, P_j) = \sqrt{\sum_{i=1}^{n} (x_i - x_{ji})^2}$. The Delaunay triangulation of P is the geometrical dual of the Voronoi diagram and can be obtained by linking the points P_j whose Voronoi polyhedra are adjacent. The Delaunay triangulation is a graph that connects the simplices of $(n+1)$ nodes. The maximum number of simplices that can be constructed from a set of m nodes is $S_{\max} = \binom{m}{n+1}$.

These n-dimensional simplices satisfy two important constraints. First, the union of all simplices must cover the complete input space. Second, the intersection of two simplices must be empty. The first condition avoids extrapolation, whereas the second constraint assures that a concrete query point always activate the same subset of nodes.

The Delaunay triangulation can be used for the construction of multidimensional fuzzy sets. Selecting a set of m points P such that they describe a linguistic term A, the fuzzy set describing that linguistic term is defined from the Delaunay triangulation of P as follows:

i) For $j = 1,2,...,m$, if P_j describes the linguistic term A then $m_A(P_j) = 1$, otherwise $m_A(P_j) = 0$.

ii) On each simplex of the Delaunay triangulation, the membership function $m_A(X)$ is defined by a linear interpolation between the ($n+1$) vertices of the n-simplex:

$$m_A(X) = a_1 x_1 + a_2 x_2 + ... + a_n x_n + a_{n+1} \qquad (5)$$

The coefficients $a_1, a_2, ..., a_{n+1}$ are calculated from the ($n+1$) vertices $P_l = (x_{l1}, x_{l2}, ..., x_{\mathrm{ln}})^T$ by solving:

$$\mathbf{a} = \mathbf{M}^{-1}\mathbf{b} \qquad (6)$$

where $\mathbf{a} = [a_1, a_2, ..., a_{n+1}]^T$, $\mathbf{M} = \begin{bmatrix} x_{1,1} & \dots & x_{1,n} & 1 \\ \dots & \dots & \dots & \dots \\ x_{n+1,1} & \dots & x_{n+1,n} & 1 \end{bmatrix}^T$, and $\mathbf{b} = [m_A(P_1), ..., m_A(P_{n+1})]^T$.

Figure 2 shows the Delaunay partition for a set of interpolation nodes $P = \{P_1, P_2, ..., P_7\}$ in the input space. For a given input vector **x**, the simplex which contains **x** is selected and its *n+1* vertices are the active nodes, contributing to the model response. Figure 3 shows the fuzzy set associated with the node P_5.

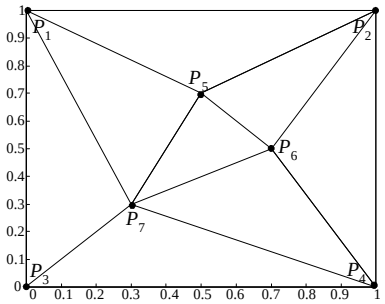

Figure 2. Delaunay partition of the input space for a given set of points.

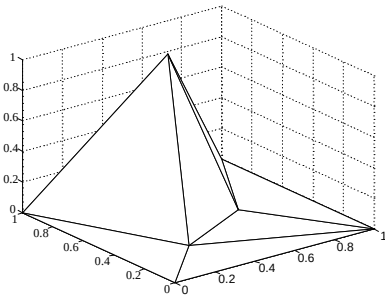
Figure 3. Fuzzy set associated with the point P_5.

There is an important advantage in using this type of triangulation when considering the task of learning nonlinear functions. It is proved that if interpolation is implemented by piecewise linear approximation, then the Delaunay triangulation has a smaller worst case error at each point than any other triangulation [4].

2.2 On-line learning algorithm

The extraction of an appropriate set of rules from data is an important problem. Moreover, given a set of rules, there is a need for on-line learning or tuning the rules to achieve the desired performance level. In this work, constructive growing and pruning techniques, commonly used in RBF neural networks are applied. Several learning algorithms have been proposed to train the RBF networks, including the well-known orthogonal least squares [8] and the hybrid training proposed by Moody and Darken [9]. In these classical approaches, the number of hidden units is fixed a-priori. This may result in too many hidden neurons. Constructive methods have been proposed to determine this number effectively.

The resource allocating network (RAN) starts with no hidden units and allocates a new unit whenever an unusual pattern is presented to the network [10]. A drawback is that once a hidden unit is created it can never be removed. Therefore, a natural improvement is to detect and remove the hidden units contributing little to the RBFN response [5]. The method presented here is based on the Delaunay partition of the input space combined with the minimal resource allocating network (M-RAN), which uses a pruning strategy to remove units together with an additional growth criterion compared to RAN. To ensure a smooth transition in the number of hidden units due to growing and pruning, M-RAN augments the basic novelty of RAN with an additional condition based on the RMS value of the output error over a sliding window.

In this framework, the criteria for adding units are the distance of the new input pattern to the closest node and the RMS error of the activated simplex. This makes the method more robust to noise. Also a more parsimonious structure can be achieved. The pruning strategy for removing units works as follows: for every observation *k*, compute all the hidden unit outputs $m_{A_i}(\mathbf{x}_k)$. If $m_{A_i}(\mathbf{x}_k) = 0$ for *M* consecutive observations then remove the *i-th* unit. This means that all the simplices that include this point are not activated. After removing the point, the Delaunay triangulation of the new reduced set of points is calculated.

Concerning the computation of the consequent parameters, the local property of the model should be considered. In fact, for a given input, only some rules are activated. The selective forgetting factor (SFF) algorithm is applied in this situation. It guarantees that the forgetting is proportional to the amount of information received in each direction [11]. The method consists of the following steps:

Step 1: Measurement update:

$$\mathrm{q}(k) = \mathrm{q}(k-1) + \mathbf{P}(k) f(k) e(k) \qquad (7)$$

where **P**(*k*) is the covariance matrix and $f(k)$ represents the output of the hidden units.

Step 2: Update the covariance matrix

$$\mathbf{P}(k+1) = \sum_{i=1}^{m} \frac{a_i(k)}{l_i(k)} n_i(k) n_i(k)^T \qquad (8)$$

where a_i and n_i are the eigenvalues and the unitary eigenvectors of **P**(*k*), respectively. In the conventional forgetting factor method all the forgetting factors l_i are equal. In this directional forgetting the chosen l_i can be a function of the amount of information received in the direction n_i as an increasing function of a_i. The forgetting factor $l_i(k)$ associated with the direction $n_i(k)$, can then be evaluated by:

$$l_i(k) = \begin{cases} 1 & \text{if } a_i(k) > a_{\max} \\ a_i(k)\left[a_{\min} + a_i(k)\dfrac{a_{\max} - a_{\min}}{a_{\max}}\right]^{-1} & \text{if } a_i(k) \le a_{\max} \end{cases} \qquad (9)$$

The proposed constructive algorithm starts from a simple set of nodes (edges covering the input space). Then, for each input pattern $\mathbf{x}_k$:

i) Compute the modelling error e_k.

ii) Compute the simplex activated by the input pattern. This can be done using the barycetric coordinates [12]. The (*n*+1) barycetric coordinates $[b_1, b_2, ..., b_{n+1}]$ of the query point $\mathbf{x}_k$ in relation to the simplex *q* are calculated by solving the following equations:

$$\begin{aligned} &i)\ \sum_{i=1}^{n+1} b_i(\mathbf{x}_k) x_{qi} = \mathbf{x}_k \\ &ii)\ \sum_{i=1}^{n+1} b_i(\mathbf{x}_k) = 1 \end{aligned} \tag{10}$$

where x_{qi} denotes the position of the *i-th* node of the simplex *q*. Determination of the active simplex is enabled by the fact that the barycetric coordinates of the query point in relation to the active simplex are non-negative if and only if this simplex contains the input vector.

iii) Updates the RMS error e_{Tk} for the active simplex.

iv) Find the closest centre $\mathbf{c}_n$.

v) If $|e_k| > e$ and $\|\mathbf{x}_k - \mathbf{c}_n\| > d$ and $e_{Tk} > e'$ then allocate a new unit ($\mathbf{x}_k$), increase the dimensionality of SFF, and find a new Delaunay partition.
Else if a center is not activated during M consecutive samples, remove that unit, decrease the dimensionality of SFF, and find a new Delaunay partition.
Else estimate consequent parameters by SFF.

3. MODELLING EXAMPLES

This section includes two numeric examples illustrating the design of the neuro-fuzzy model. The first example concerns synthetic two-dimensional data, which is useful for visualisation of the method. The second one is a non-linear identification problem.

3.1 Modelling of a 3-D surface

Consider a three dimensional data set generated by the equation [13]:

$$y = (x_1 - 1)^5 + (x_2 - 1)^5 \tag{11}$$

for 100 data points $(x_1, x_2) \in [-1,1]^2$. Both the inputs and output are normalised to the interval [0,1]. The function surface is given in figure 4. Starting from the points (0,0), (0,1), (1,0), (1,1) in order to assure the coverage property, three more nodes have been added. The obtained piecewise linear fuzzy model is represented in the figure 5. The Delaunay triangulation achieved is also visible. It can be seen that the model approximates the non-linear function by a piecewise linear surface.

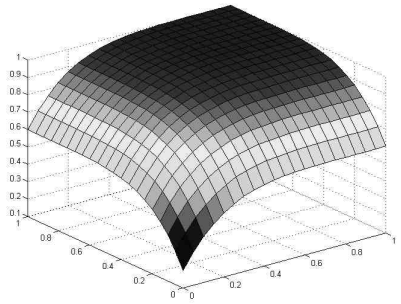

Figure 4. Function 3-D surface.

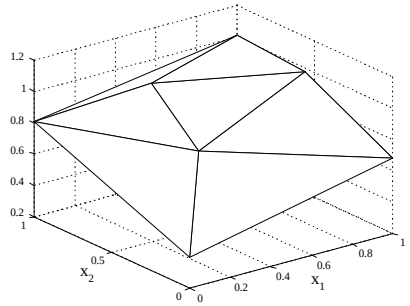

Figure 5. Piecewise linear model after 100 points presentation.

3.2 Nonlinear identification problem

As an example of a nonlinear identification problem consider the first-order non-linear system, described by:

$$y(k) = \frac{y(k-1)}{1 + y^2(k-1)} + u^3(k-1) \tag{12}$$

The goal is to learn the model on-line. The input consists of a 400 points PRBS signal in the interval [-1,1] and 800 points of a sinusoidal input signal $u(k) = \sin(2pk/100)$. All the data are normalised to the interval [0,1]. The learning algorithm starts with the following four nodes (0,0), (0,1), (1,0) and (1,1), the edges of the normalised input space.

In order to evaluate the performance of the proposed method, a comparison to standard RBF networks with Gaussian activation functions has been made. In that method [14], the learning algorithm also works on-line, the centres of the Gaussian functions are updated using a recursive version of the K-means clustering algorithm and the number of units is fixed. The selective forgetting algorithm was applied to compute the second layer weights. Table 1 summarises the one step ahead prediction results. It can be seen that the proposed neuro-fuzzy learning algorithm has a similar performance to the RBF with 20 hidden units. These results indicate a good approximation capability of the method keeping a parsimonious network structure.

Method	*Number of Units*	*RMS error*
Proposed	8	0.0059
Adaptive RBF	5	0.0862
	10	0.0142
	20	0.0054

Table 1. Comparison to Gaussian RBF with on-line learning.

4. INVERSE MODEL BASED CONTROL

Once an accurate model of the plant is available it may become a part of a model-based control scheme. The model inverse control technique is considered here. This type of control can be applied to stable systems whose inverse dynamics is also stable. In the model construction strategy presented in Section 3 a linear model is given in each model simplex.

4.1 Controller design

The considered fuzzy rule based model corresponds to the following regression NARX model:

$$y(k+1) = f\left(y(k),...,y(k-n_a),u(k),...,u(k-n_b)\right) \quad (13)$$

The control objective is to compute the control input $u(k)$, such that the system output at the next sampling time equals the reference $r(k+1)$. By inverting the model of the process and replacing $y(k+1)$ by $r(k+1)$ one obtains:

$$u(k) = f^{-1}\left(r(k+1), y(k),..., y(k-n_a), u(k-1),..., u(k-n_b)\right) \quad (14)$$

Generally, it is not an easy task to find the inverse function. However this method makes use of the piecewise linear property of the model. For each particular state $\mathbf{z}_s(k) = \left[y(k),...,y(k-n_a),u(k-1),...,u(k-n_b)\right]$, a linear difference equation describes the process dynamic behavior. In a matrix notation the model output (equation 1) is given by:

$$y(k+1) = \mathbf{z}^T \mathbf{A}^T \mathrm{q} \quad (15)$$

where $\mathbf{A} = \begin{bmatrix} a_{1,1} \ldots a_{1,n+1} \\ \ldots \ \ldots \ \ldots \\ a_{m,1} \ldots a_{m,n+1} \end{bmatrix}$,

$\mathbf{z} = \left[u(k),u(k-1),...,u(k-n_b),y(k),...,y(k-n_a),1\right]^T$,

$n = n_a + n_b$, and $\mathrm{q} = \left[q_1, q_2, .q_m\right]^T$.

By matrix manipulation:

$$y(k+1) = \mathrm{q}^T \mathbf{a}^+ u(k) + \mathrm{q}^T \mathbf{A}^+ \mathbf{z}^+ \quad (16)$$

where, $\mathbf{A}^+ = \begin{bmatrix} a_{1,2} \ldots a_{1,n+1} \\ \ldots \ \ldots \ \ldots \\ a_{m,2} \ldots a_{m,n+1} \end{bmatrix}^T$, $\mathbf{a}^+ = \left[a_{1,1},...,a_{m,1}\right]^T$,

$\mathbf{z}^+ = \left[u(k-1),...,u(k-n_b),y(k),...,y(k-n_a),1\right]$.

Inverting this linear model, and replacing $y(k+1)$ by $r(k+1)$, the control signal is given by:

$$u(k) = \frac{r(k+1) - \mathrm{q}^T \mathbf{A}^+ \mathbf{z}^+}{\mathrm{q}^T \mathbf{a}^+} \quad (17)$$

The difficulty is that the space partitioning of the inverse model may consist of overlapping simplices. Also, we do not know in advance whether the resulting $u(k)$ actually leads to an input vector lying inside the considered simplex. To solve these problems, the following procedure has been applied to chose the control signal:

i) Compute $u(k)$ using the simplex of the previous iteration (if possible).

ii) Check whether the result is inside of the simplex used. If so, use this control signal. Otherwise apply equation (17) to each simplex, check the results and among the possible solutions use the $u(k)$ closer to $u(k-1)$ in order to achieve smooth actuation signals.

4.2 Application example

Consider the first-order non-linear system described by equation 12. The parameters for the proposed algorithm are the same as in example 3.2. There it can be observed that the network is able to capture the process dynamics with a reasonable accuracy. Here, the model captured at every instant is on-line inverted according to equation 17. The inverse model can be employed in an open-loop control scheme. However, a model-mismatch will cause a steady state error. To compensate this error, some type of feedback is necessary. Usually the internal model control scheme is applied. In this work, another strategy has been applied (figure 6). Set-point following is achieved through introduction of an integrator [15]. This ensures that the steady-state output follows the constant set-point and at the same time it makes the control system robust against perturbations in the process model.

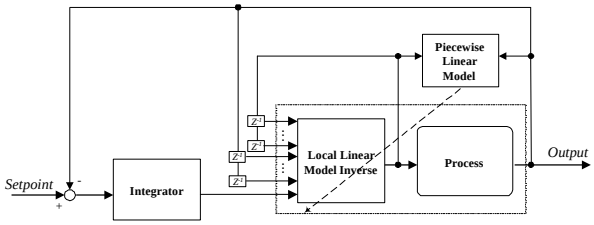

Figure 6. Inverse model with feedback control scheme.

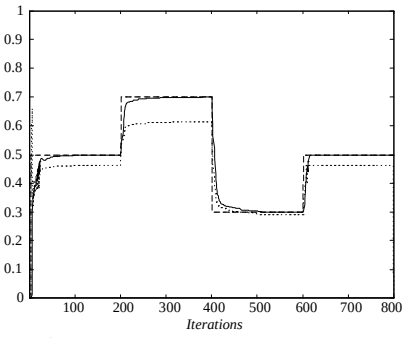

Figure 7. Reference (dashed), actuation signal (dotted) and output.

Figure 7 shows the actuation signal and the process response for different set-points. It can be observed that the controller performed well in controlling the system for a wide range of set-points. Also it can be said that the overall model is nonlinear, since at each time instant we can have different piecewise linear models. This gives good approximation capabilities keeping a parsimonious neuro-fuzzy network structure.

4.3 Laboratory process control

The presented control framework was applied to a lab scale process, the process trainer PT326, (Figure 8). In the PT326 air is forced by a fan blower through a tube and heated at the inlet. This is a non-linear system with a pure time delay. The pure time delay depends on the position of the temperature sensor (position I, II, or III) and the damper position (Ω). The system input, is the voltage applied to the power electronic circuit feeding the heating resistor, and the output is the outlet air temperature, expressed by a voltage, between -10 and 10 volts, issued from the transducer and the conditioning electronics.

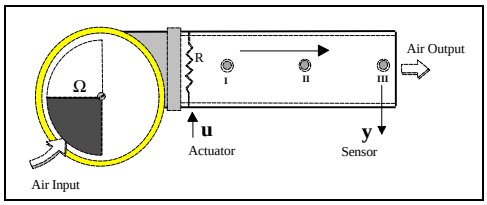

Figure 8. Experimental bench, PT326. Schematic diagram.

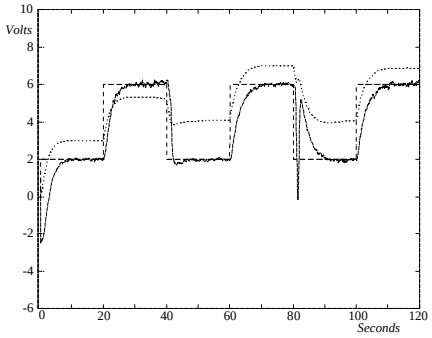

Figure 9. Set-point trajectory (dashed), actuation signal (dotted) and output.

The type of experiment carried out evaluates set point tracking and disturbance rejection by variations in the flow rate. The experiment duration was 120 seconds. The initial conditions were: blower inlet throttle Ω=20º, detector probe in position III. The sampling time was 200 ms. To test the disturbance rejection capabilities of the controller, variations were considered in the flow rate of the input air and in the sensor position: Ω=60º at 40 seconds, and at 80 seconds the detector probe is repositioned to position II. Figure 9 shows the controller behaviour. It can be seen that the adaptive controller performs well in maintaining the output at the set-point. The proposed control structure showed robustness in these situations of process disturbances.

5. CONCLUSION

In this paper, a constructive on-line learning algorithm for a neuro-fuzzy network with piecewise linear multidimensional membership functions has been considered. The method is based on constructive techniques usually applied to RBF neural networks. When modelling nonlinear functions, the results indicate a good approximation capability of the method keeping a parsimonious network structure. An inverse model with integrator feedback is also presented. In this scheme the inverse model of the process is easily computed taking benefit of the piecewise linear property of the model. The results achieved in simulation examples and using a lab scale process show the effectiveness and robustness of the method.

ACKNOWLEDGEMENTS

This work was partially supported by the Portuguese Ministry of Science and Technology, under program PRAXIS/P/EEI/14155/1998. Part of this methodology was investigated when the first author was visiting the Control Laboratory, in Delft, The Netherlands. We wish to thank Prof. Henk Verbruggen and Dr. Janos Abonyi for useful discussions held during the visit.

REFERENCES

[1] M. Brown and C. Harris, *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall, New York, 1994.

[2] J. Yen and L. Wang, "Constructing optimal fuzzy models using statistical information criteria," *Journal of Intelligent and Fuzzy Systems*, vol. 7, pp. 185-201, 1999.

[3] M. Setnes, V. Lacrose, and A. Titli, "Complexity Reduction Methods for Fuzzy Systems," Fuzzy Algorithms for Control, Kluwer, Boston, Chapter 8, 1999.

[4] S. Omohundro, "The delaunay triangulation and function learning," TR 90-001, ICS, berkeley, CA, 1989.

[5] L. Yingwei and N. Sundararajan, "Performance Evaluation of a Sequential Minimal Radial Basis Function (RBF) Neural Network Learning Algorithm," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 308-318, 1998.

[6] G. Liu, V. Hadirkamanathan, and S. Billlings, "Variable neural networks for adaptive control of nonlinear systems," *IEEE Trans. on Systems, Man and Cybernetics – Part C*, vol. 29, pp. 34-43, 1999.

[7] J. Jang and C. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Networks*, vol. 4, no. 1, pp. 156-159, 1993.

[8] S. Chen, S. Billings, and P. Grant, "Non-linear system identification using neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1191-1214, 1990.

[9] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281-294, 1989.

[10] J. Platt "A resource allocating network for function interpolation," *Neural Computation*, vol.4, no.2, pp. 213-225, 1991

[11] J. Parkum, N. Poulsen, and J. Holst, "Recursive Forgetting Algorithms," *Int. Journal of Control*, vol. 55, no. 1, pp. 109-128, 1992.

[12] T. Ullrich and T. Holle, "Delaunay Networks for Modelling of Non-linear Systems," Proc. of the IASTED Conference on Modelling and Simulation, Pittsburgh, USA, 1996.

[13] R. Babuska, *Fuzzy Modeling for Control*. Kluwer, Boston, 1998

[14] C. Pereira, J. Henriques and A. Dourado, "Adaptive RBFNN versus conventional self-tuning: comparison of two parametric model approaches for non-linear control," *Control Enginnering Practice*, 8(2), pp.3-12.2000.

[15] M. Ahmed and I. Tasadduq, "Neural servocontroller for nonlinear MIMO plant," *IEE Proc.-Control Theory Appl.*, vol. 145, no. 3, 1998.