Evolving Systems manuscript No.
(will be inserted by the editor)

Online Self-Evolving Fuzzy Controller with Global Learning Capabilities

Ana Belén Cara · Héctor Pomares · Ignacio Rojas · Zsófia Lendek · Robert Babuška

Received: date / Accepted: date

Abstract This paper presents an online self-evolving fuzzy controller with global learning capabilities. Starting from very simple or even empty configurations, the controller learns from its own actions while controlling the plant. It applies learning techniques based on the input/output data collected during normal operation to modify online the fuzzy controller's structure and parameters. The controller does not need any information about the differential equations that govern the plant, nor any offline training. It consists of two main blocks: a parameter learning block that learns proper values for the rule consequents applying a local and a global strategy, and a self-evolving block that modifies the controller's structure online. The modification of the topology is based on the analysis of the error surface and the determination of the input variables which are most responsible for the error. Simulation and experimental results are presented to show the controller's capabilities.

Keywords Evolving fuzzy system · Adaptive control · Evolutionary methodology · Adaptive fuzzy control

1 Introduction

Throughout the years, control has been one of the fields in which fuzzy systems have achieved a considerable success. The two main reasons for this are: 1) fuzzy controllers do not need an accurate model of the plant under control, and 2) they allow for the application of human expert knowledge (Ying 2000; Liu and Zheng 2009; Chen et al 2009; Rojas et al 2006; Park et al 2005; Gao and Er 2005; Angelov 2004). Furthermore, it has been proven that fuzzy controllers are universal approximators in the sense that they are able to approximate any continuous function on a compact set to any desired accuracy (Castro 1995).

There are different approaches to design a fuzzy controller. When the dynamics of the plant to be controlled are fully known (i.e., the differential equations that define the plant's behavior are available), *ad-hoc* controllers may be defined by setting their structure and determining the proper value of their parameters (Lalouni et al 2009; Galluzzo and Cosenza 2009). However, when such knowledge about the plant is not available, intelligent methods are required to automatically design the controller. Some of these methods are based on assumptions about the bounds of the unknown differential equations. Partial knowledge about the plant is used to compute the controller's parameters offline (Mucientes and Casillas 2007) or online (Liu and Zheng 2009; Wang et al 2008b), or even to modify the controller's structure (Phan and Gale 2008; Park et al 2005; Gao and Er 2003).

The design becomes even more challenging when it is not possible to make such assumptions. In this case, we find in the literature several offline algorithms based on pre-training with input/output data (Mingzhi et al 2009; Chen et al 2009; Lin and Xu 2006; Li and Lee 2003; Wang et al 2008a; Hoffmann and Nelles 2001). Works that solve the problem of the online adaptation of fuzzy control parameters, i.e., adaptation of the rule consequents and membership functions, also exist (Wang et al 2008b; Rojas et al 2006; Pomares et al 2002b). However, few results exist about the online adaptation of the structure of the fuzzy system when no prior knowledge about the plant is available (Angelov 2004; Lin et al 1995).

A.B. Cara · H. Pomares · I. Rojas
Dept. Computer Architecture and Computer Technology, University of Granada, C/ Periodista Saucedo Aranda s/n, 18071 Granada, Spain
Tel.: +34 958 241 778
E-mail: acara@atc.ugr.es, hector@ugr.es, irojas@atc.ugr.es

Zs. Lendek · R. Babuška
Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
Tel.: +31 (0)15 27 88573
E-mail: z.lendek@tudelft.nl, r.babuska@tudelft.nl

In (Angelov 2004) and (Lin et al 1995) the controller's structure is modified by using the information given by every new sample of training data. As a consequence, different sequences of training data may produce different structures for the controller. In some cases, e.g., when working in noisy environments, these methods may not be sufficiently robust. Furthermore, in (Lin et al 1995) the fuzzy controller is trained by using previously obtained data and is not actually used to control the plant until the learning has been completed. Hence, the method proposed in (Lin et al 1995) could be considered an offline learning method, in the sense that learning does not take place while controlling the plant. Therefore, the search for methods able to work with less information while controlling the plant is a very challenging issue for current research. Nevertheless, relaxing the assumptions and conditions imposed on the plant's dynamics is not without a price, as stability analysis becomes difficult. In many cases, the value of these methods can only be shown by experimental results (Angelov 2004).

The aim of this paper is to advance in the direction of these interesting works by proposing a methodology for an Online Self-Evolving Fuzzy Controller (OSEFC) with global learning capabilities. This paper is an extension of the work presented in (Cara et al 2010). The main idea is to apply global information gathered during the normal control operation to dynamically adapt the rule consequents and the fuzzy controller's topology. For this, we rely on the property of universal approximation of fuzzy controllers, which states that the proper addition of membership functions makes it possible to reach a desired accuracy level for a functional approximation. The main properties of the OSEFC proposed in this paper are the following:

- No model of the plant is needed.
- No previous knowledge about the control policy is needed. The fuzzy system can start working from an empty set of rules and will adapt itself as it controls the plant.
- All the decisions regarding the topology or parameters changes are based on historical information about the control process, gathered online while the controller is working.

The rest of the paper is organized as follows. In Section 2, the problem to be solved is introduced together with the structure of the fuzzy controller employed. In Section 3, the general architecture of the OSEFC is presented. Section 4 describes the Parameter Learning (PL) block, distinguishing between the local and the global learning techniques. Section 5 is dedicated to the description of the Self-Evolving (SE) block. Simulation and experimental results illustrating the capabilities of the algorithm are presented in Section 6. Finally, conclusions are drawn in Section 7.

2 Problem Formulation

The objective of the controller considered in this paper is to make the plant's output track a given reference signal r_k. For instance, consider a single-input single-output (SISO) plant, whose dynamics are given by

$$y_{k+1} = f(\mathbf{x}_k, u_k) \tag{1}$$

where $\mathbf{x}_k = (y_k, y_{k-1}, ..., y_{k-p}, u_{k-1}, u_{k-2}, ..., u_{k-q}) \in \Omega$ is the state of the plant, $u_k \in \mathbb{R}$ is the control signal exerted by the controller, f is an unknown continuous and differentiable function, and p and q are constants that determine the plant order.

In order to design a controller for this system, we impose the controllability condition, that is stated as (Phan and Gale 2008)

$$\frac{\partial f(\mathbf{x}, u)}{\partial u} \neq 0 \quad \forall \mathbf{x} \in \Omega, \forall u \in \mathbb{R} \tag{2}$$

where Ω is the operating region of the plant. This condition guarantees that there is no state in which the plant's output would not depend on the control signal. Furthermore, (2) implies that the partial derivative of the plant's output with respect to the control signal has a constant sign (Phan and Gale 2008; Pomares et al 2002b). Note that this restriction applies only to the partial derivative of the plant's output with respect to the control signal, but not with respect to the state.

The controller can be expressed as a function G such that

$$u_k = G(\hat{\mathbf{x}}_k; \Phi) \tag{3}$$

where $\hat{\mathbf{x}}_k = (r_k, y_k, y_{k-1}, ..., y_{k-p}, u_{k-1}, u_{k-2}, ..., u_{k-q})$ and Φ represents the set of parameters that define the controller. Note that the definition of $\hat{\mathbf{x}}_k$ is different from the one given for $\mathbf{x}_k$, as it also includes the reference value at time k. The controller has the objective of making the plant's output reach the set point as fast as possible.

According to (3), the control problem may be cast as a problem of function approximation of the plant's inverse function (Pomares et al 2004). We choose to use a zero-order Takagi-Sugeno fuzzy system with a complete set of rules (Lee 1990) to approximate (3). The *i-th* fuzzy rule $\Re_i$ is given by

$$\Re_i : \text{IF } x_1 \text{ is } X_1^{i_1} \text{ AND } x_2 \text{ is } X_2^{i_2} \text{ AND} \ldots x_N \text{ is } X_N^{i_N} \text{ THEN } u_i = Q_i \tag{4}$$

where $i = 1...N_r$, N_r is the number of fuzzy rules, N is the number of inputs to the fuzzy system, $X_v^{i_v} \in \{X_v^1, X_v^2, ..., X_v^{n_v}\}$ are the membership functions (MFs) of the input x_v, n_v is the number of membership functions for that input variable, u_i is the output of the *i-th* rule and Q_i is a scalar value representing the rule consequent.

Although our approach is independent of the type of membership function used, we have chosen triangular membership functions to provide better transparency and interpretability to the fuzzy system (Wang et al 2008a). Furthermore, by using normalized triangular membership functions only the centers of the membership functions need to be stored (Rojas et al 2000). At the same time, it is guaranteed that for any input within the input range, exactly two membership functions for each variable attain non-zero activation degrees. This reduces the complexity of the computations while assuring that the whole operating region is covered by fuzzy rules. The membership functions placed at both ends of the range of the input variables are trapezoidal. This allows responding to the issue of data that go beyond the range of the variables. If a point is out of range, the corresponding trapezoidal MF is activated with a degree equal to 1. This situation is identified as an out-of-range point and the controller responds to it by exerting a control action and modifying the consequents of the rules accordingly to the output error.

The selected fuzzy system uses the product as the T-norm for the conjunction and the weighted average as the defuzzification strategy. Thus, the output of the fuzzy system at instant k is given by

$$u_k = \hat{G}(\hat{\mathbf{x}}_k;\Phi_k) = \frac{\sum_{i=1}^{N_r} Q_i \cdot \alpha_i(\hat{\mathbf{x}}_k)}{\sum_{i=1}^{N_r} \alpha_i(\hat{\mathbf{x}}_k)} \tag{5}$$

where Φ_k is the set of parameters defining the fuzzy controller at time k and $\alpha_i(\hat{\mathbf{x}}_k)$ is the firing strength of the *i-th* rule, which is calculated by

$$\alpha_i(\mathbf{x}) = \alpha_i(x_1,x_2,...,x_N) = \prod_{j=1}^{N} \mu_j^i(x_j) \tag{6}$$

where $\mu_j^i(x_j)$ is the activation degree of the *j-th* MF in the antecedent part of the rule $\Re_i$.

The set of parameters of the fuzzy controller Φ_k is formed by the rule consequents Q_i $(i = 1,2,...,N_r)$ and the centers of the membership functions $\phi_{v,j}$ $(v = 1,2,...,N; j = 1,2,...,n_v)$. The use of normalized triangular membership functions simplifies (5) as follows:

$$u_k = \hat{G}(\hat{\mathbf{x}}_k;\Phi_k) = \sum_{i=1}^{N_r} Q_i \cdot \alpha_i(\hat{\mathbf{x}}_k) \tag{7}$$

since $\sum_{i=1}^{N_r} \alpha_i(\mathbf{x}) = 1, \forall \mathbf{x}$.

Note that the challenge tackled in this paper, i.e., the online evolution and control starting from no knowledge about the plant, is more complex than typical function approximation problems, because the approximation of (3) must be obtained while controlling the plant in a proper way. Although in this paper we have considered only SISO plants, the proposed methodology can easily be extended to multiple-input multiple-output (MIMO) problems.

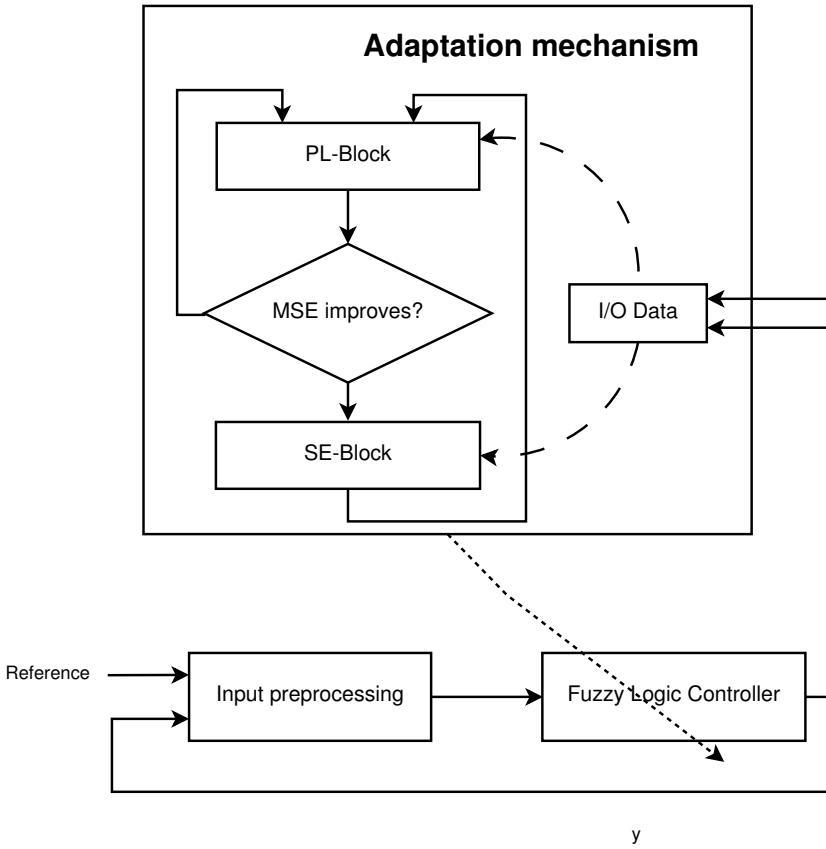


Fig. 1 Representation of the closed-loop system

3 Architecture of the OSEFC

The complete closed-loop system is presented in Fig. 1. The proposed adaptation mechanism of the OSEFC is shown on the top. Its internal structure consists of two blocks (parameter learning and self-evolution of the topology) that will be explained in the sequel and a memory to store the I/O data collected during the operation of the plant for the use in the two adaptive blocks. Below, the fuzzy logic controller that is modified by the adaptation mechanism (dotted line) provides the control signal that is applied to the plant. The OSEFC is the combination of the adaptation mechanism and the fuzzy logic controller. Finally, the plant produces the output y that is fed back to the controller. The Input Preprocessing block is optional. It simply represents the possibility of using different sets of inputs for the fuzzy controller.

Fig. 2 depicts the general flowchart of the OSEFC. There are two main blocks: 1) the PL-block uses the control evolution information for parameter learning; 2) the SE-block

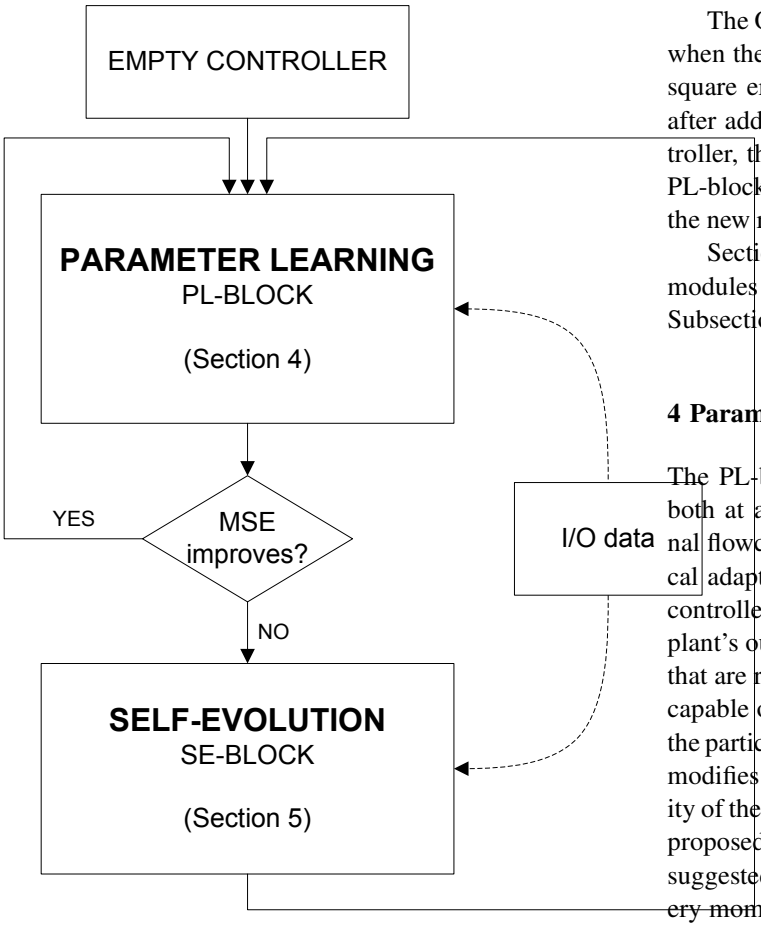


Fig. 2 General flowchart for OSEFC

uses this I/O information to modify the topology of the fuzzy controller.

The PL-block performs the tuning of the rule consequents at two different levels. On the one hand, the Local Learning module (LL-module) combines information about the current state of the control and the monotonicity of the plant's output with respect to the control signal to modify the rule consequents. The aim of these changes is to direct the plant's current output to its target value. On the other hand, the Global Learning module (GL-module) uses the I/O data collected during operation to fine tune the rule consequents. These changes come from a global perspective, as their objective is not only to direct the current output to the current set point, but also to guarantee that the control in other operating areas has a reasonable performance.

The second main block is the SE-block, which is responsible for the evolution of the topology of the fuzzy controller. This block also uses the I/O data collected during the normal operation of the control system. In this case, the information is used to determine which input variable is responsible for the current approximation error and, therefore, should receive an extra membership function.

The OSEFC switches from the PL-block to the SE-block when the former does not further improve the global mean square error (MSE) in the control performance. However, after adding a new membership function to the fuzzy controller, the OSEFC switches back to the LL-module in the PL-block to restart the procedure of consequent learning for the new rules.

Section 4 describes in detail the PL-block and its two modules (LL-module in Subsection 4.1 and GL-module in Subsection 4.2). The SE-block is presented in Section 5.

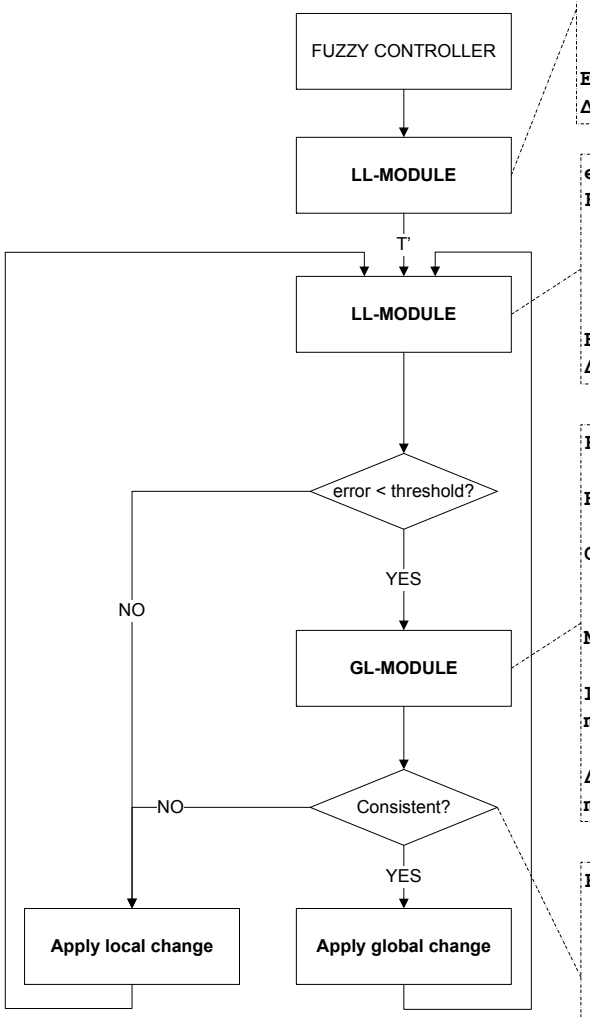
4 Parameter learning

The PL-block is in charge of tuning the rule consequents both at a local and a global level. Fig. 3 shows the internal flowchart and the pseudo-code of the PL-block. The local adaptation takes care of the short-term response of the controller. At every time instant it analyzes the error at the plant's output and applies a reward or penalty to those rules that are responsible for the error. This type of action is only capable of a coarse tuning of the parameters, which makes the participation of the GL-module necessary. The GL-module modifies the consequents of the rules so that the global quality of the control policy is preserved. Obviously, the changes proposed by the GL-module must be consistent with those suggested by the LL-module, as the main objective at every moment is to control the plant. The pseudo-code of the routines has been included to clarify the operation of the PL-block.

The change from local to global learning is performed as follows: first, only the LL-module works during a period *T'*. During this period, the control policy is rough, but the local learning allows the gathering of truly useful data from the plant. After this period, and while the plant is controlled locally, the GL-module starts tuning the consequents to achieve a finer learning. It is important to note that during the global learning stage, the LL-module continues working as a supervisor of the GL-module.

4.1 Local learning

From a "local" point of view, the goal of the controller is to bring the plant's output from its current value to the desired reference value at the next time step, i.e., $y_{k+1} = r_k$. According to the controllability condition (2), the partial derivative of the plant's output with respect to the control signal has a definite constant sign. Therefore, the combination of the error at the plant's output and the sign of the monotonicity of the plant w.r.t. the control signal, gives us information about the right direction in which the rule consequents have to be moved to achieve the local control objective (Rojas et al 2006).




```
e(k) = r(k-1) - y(k)
FOR each rule consequent Qi
   ΔQi = C · α(k-1) · e(k)
   IF (u = umin AND ΔQi<0) OR (u = umax AND ΔQi>0)
      ΔQi = 0
   END IF
END FOR
Δ = {ΔQ1, ΔQ2, ... ΔQNr}
```

```
e(k) = r(k-1) - y(k)
FOR each rule consequent Qi
   ΔQi = C · α(k-1) · e(k)
   IF (u = umin AND ΔQi<0) OR (u = umax AND ΔQi>0)
      ΔQi = 0
   END IF
END FOR
ΔL = {ΔLQ1, ΔLQ2, ... ΔLQNr} ={ΔQ1, ΔQ2, ... ΔQNr}
```

```
FOR each I/O datum M(m) = [x(m), u(m)]in M
      u(m) = fuzzy controller(x(m))
END FOR
Compute Jk according to equation (13):
      Jk = sum((u(m) - ũ(m))²)
Minimize Jk according to equation (14)

Γ* = {Q1*,Q2*, ... ,QNr*} are the consequents that
minimize Jk
ΔG = {Q1-Q1*, Q2-Q2*, ... ,QNr-QNr*} are the
modifications proposed for the global learning
```

```
FOR each rule consequent Qi (i=1,2, ... Nr)
   IF (ΔL(Qi) and ΔG(Qi) have the same sign)
      ΔQi= ΔG(Qi)
   ELSE
      ΔQi= ΔL(Qi)
   END IF
END FOR
Δ = {ΔQ1, ΔQ2, ... ΔQNr}
```

Fig. 3 Flowchart and pseudo-code of the PL-block

Let us assume that the derivative in (2) is positive. This means that given the state x_k of the system if we apply $u_1 > u_0$, then the output obtained for u_1 is larger than the one obtained for u_0, i.e., $f(k_k, u_1) > f(x_k, u_0)$. If we define the error at the plant's output as $e_k = r_{k-1} - y_k$, there are the three possible cases:

- If the error is negative, i.e., $y_k > r_{k-1}$, the control signal applied at time k was too large. To correct this behaviour, the consequents of the fuzzy rules must be decreased.
- In the opposite case, $e_k > 0$ (i.e., $y_k < r_{k-1}$), the control signal was too small and the consequents must be increased.
- Finally, if $e_k = 0$, the plant's output has reached the reference value and no correction is needed in the rule consequents.

From the previous analysis, we conclude that the sign of the error e_k indicates the direction in which the consequents have to be moved. If the plant's monotonicity with respect to the control signal is negative, the previous study still holds, but the changes go in the opposite direction. On the other hand, not all the rules contribute in the same degree to reaching the current value at the plant's output. Therefore, the modification applied to each of them must be proportional to their contributions. Thus, we apply the following modification to the consequent of the *i-th* fuzzy rule at instant k:

$$\Delta Q_i(k) = C \cdot \alpha_i(k-1) \cdot e_k \tag{8}$$

where $\alpha_i(k-1)$ is the firing strength of the rule at the previous time step, when the rule was fired to obtain r_{k-1}. C is a normalization constant with the same sign as $\partial y/\partial u$. Its absolute value is set online as

$$|C| = \Delta u/\Delta r \tag{9}$$

where Δu is the range of variation of the control signal and Δr is the range of possible reference values. It is to be clarified that Δr does not refer exclusively to the range in which the reference signal is actually taking its values, but rather refers to the range of values that are allowed by the system. Note that Δu is also known *a priori*, since the user must know the limitations of the actuators before using them.

The rationale behind (9) is as follows: the objective of the adaptation law is to decrease the error at the plant's output. In this case, it can be proven that the absolute value of C must be

$$|C| \leq \frac{1}{|\frac{\partial f(\mathbf{x},u)}{\partial u}|_{max}} \tag{10}$$

Since the differential equations that govern the plant are unknown, the maximum value of the partial derivative shown in the denominator of (10) is also unknown and this expression cannot be applied. We use (9) as an approximation of this value.

Most real life controllers are limited in their operation, which highly affects the control process. For instance, if the actuator is only able to operate within the range $[u_{min}, u_{max}]$ and at a given moment the optimal control input to reach the reference value is $u(k) > u_{max}$, the input that is finally applied to the plant is u_{max}, so it is not possible to reach the desired set point at the next time step. However, we cannot apply any penalty to the rules, as they are already giving the best possible answer. Therefore, no adjustment is applied to

the rules when the error is due to the actuator's limitations:

$$\Delta Q_i(k) = \begin{cases} 0 & \text{if } u_{k-1} = u_{min} \;\&\; \Delta Q_i(k) < 0 \\ 0 & \text{if } u_{k-1} = u_{max} \;\&\; \Delta Q_i(k) > 0 \\ \Delta Q_i(k) & \text{otherwise} \end{cases} \quad (11)$$

The LL-module performs the consequent learning process online, while the controller is operating over the real plant. Control actions are applied from the first moment, with increasing accuracy as the adaptation evolves. Nevertheless, note that the adaptation proposed here is based primarily on qualitative information and, therefore, only a coarse tuning of the consequents is obtained by the LL-module.

4.2 Global learning

The local learning procedure focuses on the control at the current time instant. However, it is also necessary to assure the global performance of the control policy. In this way, the controller will be able to respond to the changes in the value of the reference signal. This means that we cannot simply concentrate on the rules that are being fired at the moment, but also have to consider the remaining rules. To do so, we need historical data about the behaviour of the plant.

Since our hypothesis here is that the dynamics of the plant are unknown, we cannot compute the partial derivative $\partial y/\partial u$. Therefore, it is not possible to apply any gradient-based techniques to minimize the error at the plant's output. However, such methodologies can be applied if we use the error in the controller's output instead of that in the plant's output (Pomares et al 2004). As mentioned before, the fuzzy controller is an approximator of the plant's true inverse function. Therefore, minimizing the error made by the controller indirectly minimizes the tracking error made at the plant's output. In this way, we avoid the need for the model of the plant.

We propose a method that exploits the fact that the very operation of the control system provides the I/O data about the plant's inverse function needed to perform the aforementioned approximation. Hence, if the control signal u_k exerted at time k produces the plant's output y_{k+1} we know that, if the system is ever in the same state again and the reference value happens to be $r_{k'} = y_{k+1}$, the optimum control signal is precisely u_k. Therefore, at every sampling time the plant produces a value of the plant's true inverse function.

In order to use this information, we store in a memory M the I/O data collected during the normal control operation. The memory is organized as a grid that divides the input space. Each datum has the form $[\tilde{\mathbf{x}}_k, u_k]$, where $\tilde{\mathbf{x}}_k = (y_{k+1}, y_k, ..., y_{k-p})$. Every new datum is stored in the hypercube of the grid corresponding to $\tilde{\mathbf{x}}_k$, replacing the datum already stored in that position, if there was any. The data stored in this memory is used with two different purposes in our approach: 1) for the global learning of the rule consequents described in this section, and 2) for the modification of the controller's topology. For the latter, a uniform representation of the function to be approximated is required. With this organization, the data stored in M provide such uniform representation of the inverse function of the plant. Moreover, the organization of the memory as a grid reduces the memory costs, as only one I/O pair is stored in each hypercube.

Now it is possible to compute the error at the controller's output at time k for the *m-th* datum as

$$e_u(m) = u_m - \tilde{u}_m \quad (12)$$

where u_m is the control signal stored in the memory M and $\tilde{u}_m$ is the output produced by the current controller for the input vector $\tilde{\mathbf{x}}_m$.

At each control step k, the objective of the GL-module is to minimize the mean square error for all the data in M:

$$J_k = \sum_{m=1}^{\#M} e_u^2(m) = \sum_{m=1}^{\#M} (u_m - \tilde{u}_m)^2 = \sum_{m=1}^{\#M} (u_m - \hat{G}(\tilde{\mathbf{x}}_m; \Phi_k))^2 \quad (13)$$

where $\#M$ is the number of elements stored in the memory M.

The new consequents proposed for the fuzzy controller are those that minimize (13), that is:

$$\Gamma^* = \arg\min_{Q}(J_k) = \arg\min_{Q}(\sum_{m=1}^{\#M} (u_m - \hat{G}(\tilde{\mathbf{x}}_m; \Phi_k))^2) \quad (14)$$

where $\Gamma^* = \{Q_1^*, Q_2^*, ..., Q_{N_r}^*\}$ is a set with the new values for all the consequents.

Before replacing the old consequents in the fuzzy controller by the new ones, it must be kept in mind that we have minimized the error at the controller's output and not the error at the plant's output. An additional check is necessary to verify whether the modification proposed by the GL-module actually improves the tracking of the reference signal. In this sense, we use the LL-module as a supervisor: as stated before, the monotonicity of the plant allows us to determine the right direction in which the consequents have to be moved. Therefore, if the modification proposed for a consequent by the GL-module is in the same direction as that indicated by the LL-module, the change is accepted. Otherwise, we apply the local modification.

Finally, it is important to note that the global learning cannot be used without the existence of the local module as the latter is the one which is initially capable of starting to control the plant and obtaining truly useful I/O data.

5 Self-evolving topology

In the previous section we introduced a method for the online learning of the rule consequents of a fuzzy controller with a predefined configuration of membership functions. However, it still is necessary to fix the structure of the fuzzy controller beforehand (i.e. control inputs, number of membership functions for each input, parameters of the membership functions, etc.). This may not be a trivial task, as it requires certain knowledge about the system to be controlled (Rojas et al 2006; Phan and Gale 2008).

In order to operate without any previous knowledge, a true self-evolving adaptive controller must be able to adapt not only some of its parameters, but its full structure as well. However, most adaptive fuzzy controllers proposed in the literature use a predefined structure (Liu and Zheng 2009; Wang et al 2008b; Mucientes and Casillas 2007), that often leads to systems whose structure is unnecessarily large or too small to adequately represent the plant (Phan and Gale 2008).

To tackle the problem of the evolution of the topology of the fuzzy controller, information about all the operating regions reached by the plant is required. Again, this information must be gathered during the normal operation of the controller. Unlike other methods proposed to solve this problem (Angelov 2004; Lin et al 1995), OSEFC considers the entire operating region when modifying the controller's topology, thereby providing more robustness.

To do this, we again make use of the I/O data stored in M during the parameter learning phase. In this case, the information is used to decide which input variable needs most a new membership function. Fig. 4 depicts the elements that form the SE-block:

- First, an input variable has to be selected to receive an additional membership function. Adding membership functions to all the inputs is not feasible as the number of rules grows exponentially with the number of membership functions (Pomares et al 2002a).
- After selecting the most appropriate input variable, we must select a good position to place it. To reduce the method's sensibility to noise, we analyze the entire error distribution, instead of placing the new function at the point of highest error.
- Finally, the new fuzzy rules created from the new MF have to be initialized. The goal of this initialization is that the controller's performance is minimally affected by the topology change.

All these steps are detailed in the following subsections. It is important to note that to avoid an uncontrollable growth of the number of MFs (and thereby of the number of rules) we set a threshold for the desired accuracy in the approximation. Once this is met, no new membership functions are added. However, if the control performance decreases below the desired level (e.g. a change in the plant's dynamics happens), the SE-block starts operating again.

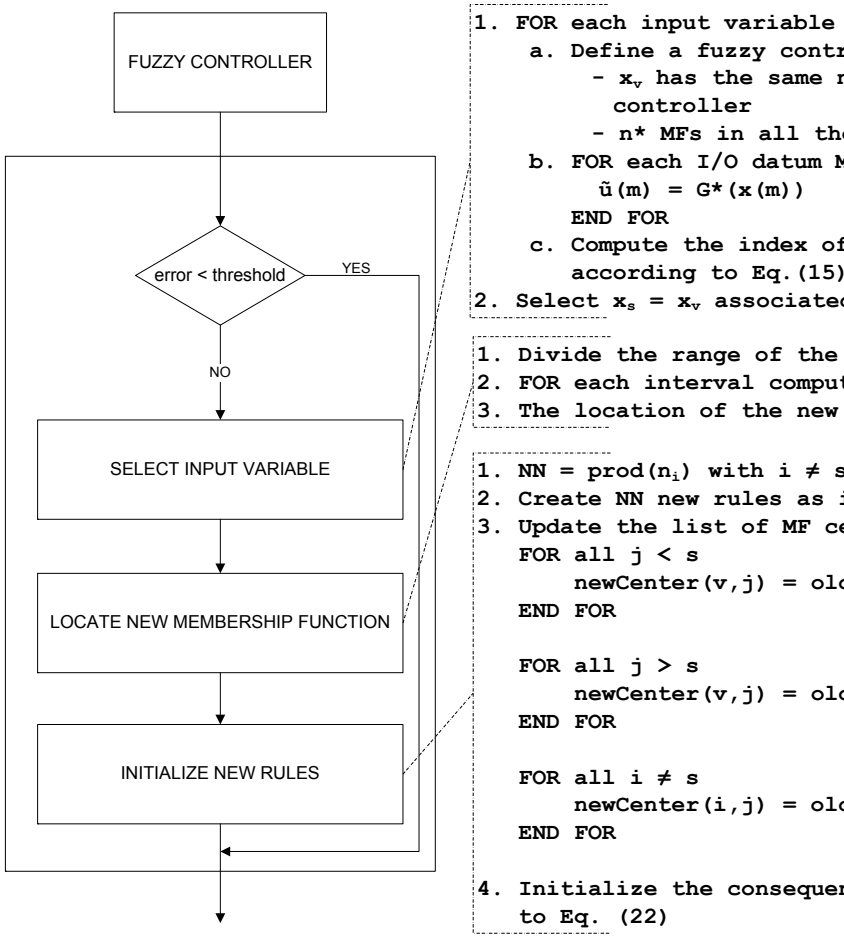


Fig. 4 Flowchart and pseudo-code of the SE-block

5.1 Selection of the input variable

In the proposed method we only add a membership function to one input variable at a time. The selection of this input is based on the study of the complete error surface reached by the current configuration (Pomares et al 2000), instead of considering only the point of maximum error, which would lead to a method with high sensibility to noise (Phan and Gale 2008).

The main idea is to check independently the degree of responsibility of each input variable on the approximation error and to select the one with the highest error. To do so, we assume that the approximation is perfect in all the other dimensions and that only the variable under analysis is responsible for the approximation error. To perform this analysis, the following algorithm is applied:

1. For each input x_v:

(a) Construct an auxiliary fuzzy system G_v^* such that
   i. The input x_v has the same number of membership functions as on the current fuzzy controller.
   ii. Associate to all the other inputs a large number n^* of MFs.

(b) The approximation of the plant's inverse function produced by this fuzzy controller is $G_v^*(\tilde{\mathbf{x}}_m; \Phi^*)$, where $\tilde{\mathbf{x}}_m$ represents the input data stored in M and Φ^* is the set of parameters with the properties described in step 1a.

(c) Compute the Index of Responsibility for input x_v (IR_v) as

$$IR_v = \sum_{m=1}^{\#M} (u_m - G_v^*(\tilde{\mathbf{x}}_m; \Phi^*))^2 \tag{15}$$

2. Select the input variable x_s with the largest value of IR_v, i.e. $s = \arg\max_v(IR_v)$.

Under the assumption that the memory M provides a uniform distribution of the data in the state-space, that is, all hypercubes contain data, the number of data points along one axis is indeed the *n-th* root of the total number. Therefore, a possible value for n^* is

$$n^* = \frac{\sqrt[N]{\#M}}{2} \tag{16}$$

where $\#M$ is the number of I/O data stored in the memory M and N is the number of input variables.

This value is based on the idea of having neither too few MFs (which would lead to a structure far from the ideal case, in which all the inputs have infinite membership functions) nor too many (in which case, many of them would not be activated by any of the input vectors stored in the memory M).

Note that the fuzzy system G_v^* is used solely to determine which input variable is receiving the new membership function. The SE-block operates in parallel with the normal operation of the controller. For this reason, the auxiliary fuzzy controller only works with the data stored in M, and is never applied to the real plant.

The algorithm proposed allows us to select the variable with the highest responsibility for the current approximation error and, hence, the one which needs to have one more membership function. The next step is to find a proper location for this new function.

5.2 Location of the new membership function

After selecting the input variable, we have to find a good location for the new MF. As mentioned before, selecting the point of largest error leads to methods sensible to noise. Hence, we analyze the error distribution through the whole variable's range. The position of the new membership function is the center of gravity of the error distribution.

With this aim, we divide the range of the input variable x_s selected in Section 5.1 in K small intervals of width Δx_s and compute the mean square error in each of them. Let us define the *j-th* interval of the input x_s as $[x_{s_{min}} + (j-1)\cdot \Delta x_s,\ x_{s_{min}} + j\cdot\Delta x_s]$ and the set of data points stored in M with component x_s within the *j-th* interval defined in the range of the variable x_s as

$$\chi_j^s = \{\mathbf{x} \mid x_s \in [x_{s_{min}} + (j-1)\cdot\Delta x_s,\ x_{s_{min}} + j\cdot\Delta x_s]\} \tag{17}$$

where $j = 1, 2, ..., K$, $\mathbf{x}$ is a data vector stored in the memory M, x_s is the *s-th* component of that data and $x_{s_{min}}$ is the left end of the range of the variable x_s.

The mean square error of all the data in χ_j^s is given by

$$\overline{e^2}(\chi_j^s) = \frac{\sum_{\mathbf{x}\in\chi_j^s} e_u^2(\mathbf{x})}{\#\chi_j^s} \tag{18}$$

where e_u is computed as in (12) and $\#\chi_j^s$ represents the number of elements in the set χ_j^s.

The center of the new membership function is located at the gravity center of the error distribution, which is given by

$$\phi_s^* = \frac{\sum_{j=1}^{K} c_j \cdot \overline{e^2}(\chi_j^s)}{\sum_{j=1}^{K} \overline{e^2}(\chi_j^s)} \tag{19}$$

where c_j is the central value of the interval $[x_{s_{min}} + (j-1)\cdot \Delta x_s,\ x_{s_{min}} + j\cdot\Delta x_s]$.

5.3 Initialization of the new fuzzy rules

Since we are using a complete set of rules, the addition of one MF to the input x_s implies the creation of $\prod_{\substack{i=1\\ i\neq s}}^{N} n_i$ new rules. It is possible to initialize these new rules to any random value, as the PL-block is able to adapt them properly. However, this choice may cause a sudden decrease in the controller's performance in the first time steps after the topology change. This undesirable effect is avoided by initializing the consequents of the new rules to values that guarantee a minimum quality degradation.

The new rules have the form

$$\Re^*: \text{IF } x_1 \text{ is } X_1^{i_1} \text{ AND...AND } x_s \text{ is } X_s^{j_s} \text{ AND...} x_N \text{ is } X_N^{i_N} \text{ THEN } u = Q^* \tag{20}$$

where j_s is the position of the new membership function within the list of ordered centers.

To explain the changes applied to the parameters of the fuzzy system, let Φ be the set of old parameters (before the new membership function was added) and Φ^* the new set of parameters. First, we include the new membership function ϕ_{s,j_s} in the list of ordered centers of the variable x_s. Note that the MFs belonging to all the others variables do not undergo any changes:

$$\begin{aligned} \phi^*_{s,j} &= \phi_{v,j} && if\ j < j_s \\ \phi^*_{s,j} &= \phi_{v,j-1} && if\ j > j_s \\ \phi^*_{i,j} &= \phi_{i,j} && if\ i \neq s \end{aligned} \tag{21}$$

Secondly, the consequents of the new rules have to be initialized. The idea is that the global function represented by the fuzzy controller keeps the same outline as before the change, i.e. $\hat{G}(\mathbf{x};\phi^*) = \hat{G}(\mathbf{x};\phi)\,\forall\mathbf{x}$. The new rules are activated only when the input value for variable x_s is within the range $[\phi_{v,j_s-1}, \phi_{v,j_s+1}]$, which is the partition created by the new membership function. Therefore, we only need to assign values to these rules. To obtain the new consequents we impose the condition that, at the rule's point of maximum activation, the consequent equals the output of the fuzzy controller under its previous configuration for the same input. As this maximum activation happens when all the inputs are located at the centers of the MFs in the premise, we have that:

$$Q^* = \hat{G}(\mathbf{c};\Phi) \tag{22}$$

where $\mathbf{c} = (\phi_{1,i_1}, ..., \phi_{s,i_s=j_s}, ..., \phi_{N,i_N})$, with $i_1 = 1, ..., n_1$, ..., $i_N = 1, ..., n_N$.

5.4 Related work

In this section we present some comments about the comparison of the proposed OSEFC with other methods found in the literature. As mentioned in the introduction, many of the approaches proposed for the online adaptation of fuzzy controllers only modify the parameters of the fuzzy controller, but use a fixed structure that has to be chosen a priori (Wang et al 2008b; Rojas et al 2006; Pomares et al 2002b). Others also modify the structure, but are based on assumptions about the differential equations that govern the plant (Phan and Gale 2008; Park et al 2005; Gao and Er 2003). Comparing the proposed methodology with this type of algorithms is very difficult, as the assumptions and the initial information in every case are very different.

To the best of our knowledge, the methodology proposed by Angelov (2004) is the most relevant work concerning a problem similar to the one tackled in this paper, i.e., the online development of a fuzzy controller while the controller is operating and without previous knowledge about the plant. However, the way the problem is solved is very different in the two cases. Firstly, the underlying ideas are different. In (Angelov 2004), the structure of the controller may evolve with each new incoming data; that is, every new data is studied to decide whether or not a new fuzzy rule should be created. In our case, we gather data in a global memory and use a set of data points to decide when and where to add new membership functions. Therefore, our topology changes are less frequent in time than in the approach proposed by Angelov (2004). Moreover, in (Angelov 2004), each topology change consists only of the addition or modification of one fuzzy rule. In our proposal, a change in the topology consists of the addition of one membership function to one input and all its related rules, since a complete set of rules is being used.

Another difference regarding the evolution of the topology is the way the decision of adding MFs/rules is made: in the method proposed in (Angelov 2004), this decision is based on the potential of the data, which is related to the distance between the data points. In our case, we analyze the surface of the approximation error, trying to reduce the error.

The methods used for the adaptation of the rule consequents are also different. In (Angelov 2004) the recursive weighted least square approach (Kalman filter) is applied, while we base the modification of the consequents on two different concepts: on the one hand, the local learning module uses the error at the plant's output to reduce it in the short term. On the other hand, the global learning module uses global information to reduce the error at the controller's output and, hence, indirectly reduce the error at the plant's output.

Finally, there are also some minor differences regarding the structure of the fuzzy controllers used. Firstly, the method of Angelov (2004) defines a fixed set of input variables (the reference, the plant's output and some measure of the external perturbations). In our case, the set of input variables may differ from one problem to another. For instance, in the first example (Section 6.1) the inputs are the reference value and the plant's output, while in the second experiment (Section 6.2) we use the reference value, the plant's output and the first derivative of the plant's output (the angular velocity). Secondly, in (Angelov 2004) a first-order TSK fuzzy system is used, while in our approach we have chosen a zero-order TSK. Although the approximation power of the first-order TSK systems may be higher, the number of parameters to be estimated and adapted is also larger.

6 Test results

With the aim of providing a better insight into the operation of the proposed OSEFC, two examples are presented in this section. The first one (presented in Section 6.1) is a simulation example. In this case we only apply the parameter

learning procedure, described in Section 4, to a controller with a fixed topology. To visualize the positive effect of the global learning, we have developed two simulations in the same conditions: the first one only applies the local learning module, while the second one applies both the local and the global learning policies. In this experiment we also show the response of the OSEFC to an unexpected change in the dynamics of the plant to be controlled.

On the other hand, Section 6.2 presents an experiment with a real setup consisting of a nonlinear servo system. In this case, we apply the full OSEFC (parameter learning and self-evolving topology, described in Section 5) to control the system. The objective of this experiment is to show the capabilities of the proposed approach to control a real system.

6.1 Parameter learning: local learning vs. global learning

In this section we illustrate the parameter learning methodology using a fuzzy controller with a fixed structure. Let us consider the plant given by (Pomares et al 2002b):

$$y_{k+1} = -0.3\sin(y_k) + u_k + u_k^3 \tag{23}$$

This plant shows a nonlinear behavior both with respect to the control signal and the output variable. Note that a control signal $u_k = 0$ does not guarantee a stationary plant output. On the other hand, it is clear that the plant has a positive monotonicity with respect to u.

Consider a fuzzy controller with two input variables, the reference signal r_k and the plant's output y_k. For this example we assign five evenly distributed MFs to each input, as shown in Fig. 5. Therefore, 25 fuzzy rules are used. According to (7) and given this structure, the output of the fuzzy controller is given by

$$u_k = \hat{G}(\hat{\mathbf{x}}_k;\Phi_k) = \sum_{i=1}^{25} Q_i \cdot \alpha_i(\mathbf{x}_k) \tag{24}$$

where $\hat{\mathbf{x}}_k = (r_k, y_k)$. Initially, all the rule consequents are set to zero. The reference signal is a random step function in the range [-4,4] and the actuator's range is [-0.4,0.4]. According to (9), the constant C is set to 0.1.

Fig. 6 compares the evolution of the MSE for the cases of applying only local learning (solid line) and both local and global learning (dotted line). For the latter case, the period of initial data gathering has been $T' = 10000$ samples. This is the reason why in the beginning both alternatives show the same evolution. However, the effect of the global learning is evident: although the minimum error achieved by the local learning is 0.039, the addition of the GL-module allows reaching a value of 0.0038, i.e., the global learning performance is ten times better.

Another interesting fact is the response of the OSEFC to unexpected changes in the dynamics of the plant. To show

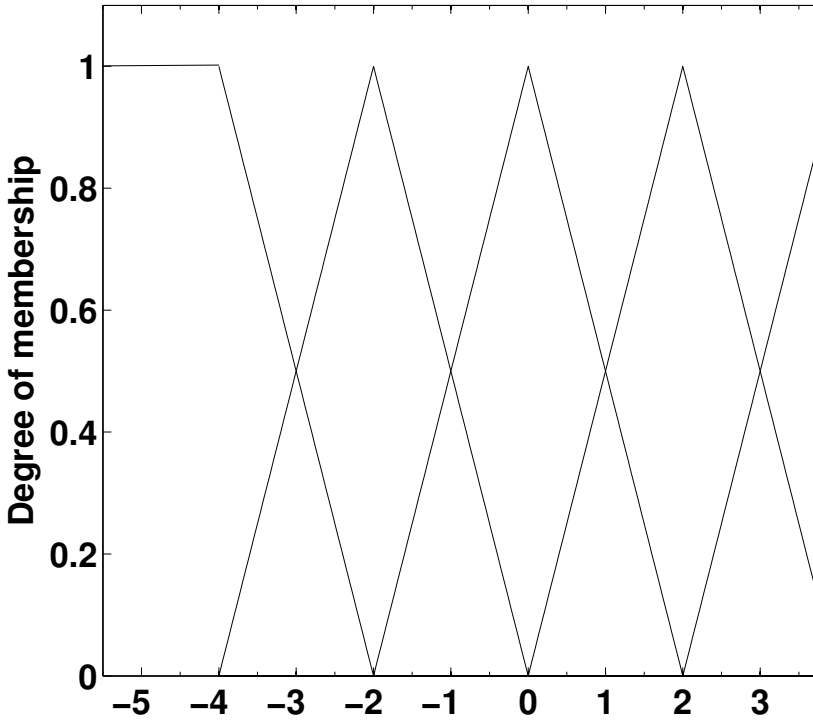


Fig. 5 Location of the membership functions used for both inputs r_k and y_k in Example 1

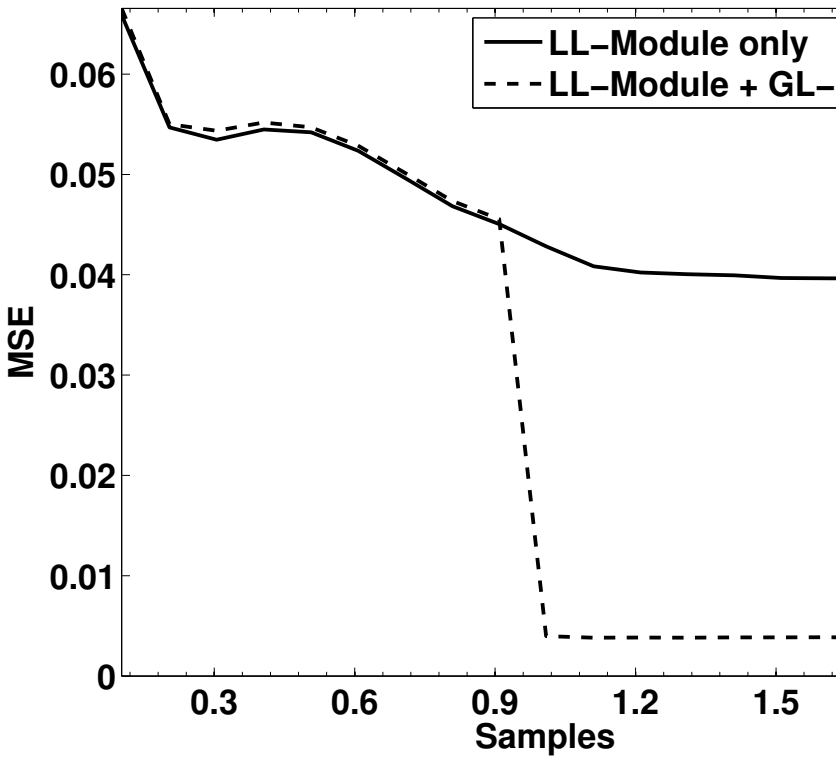


Fig. 6 Local learning performance versus local and global learning performance for Example 1

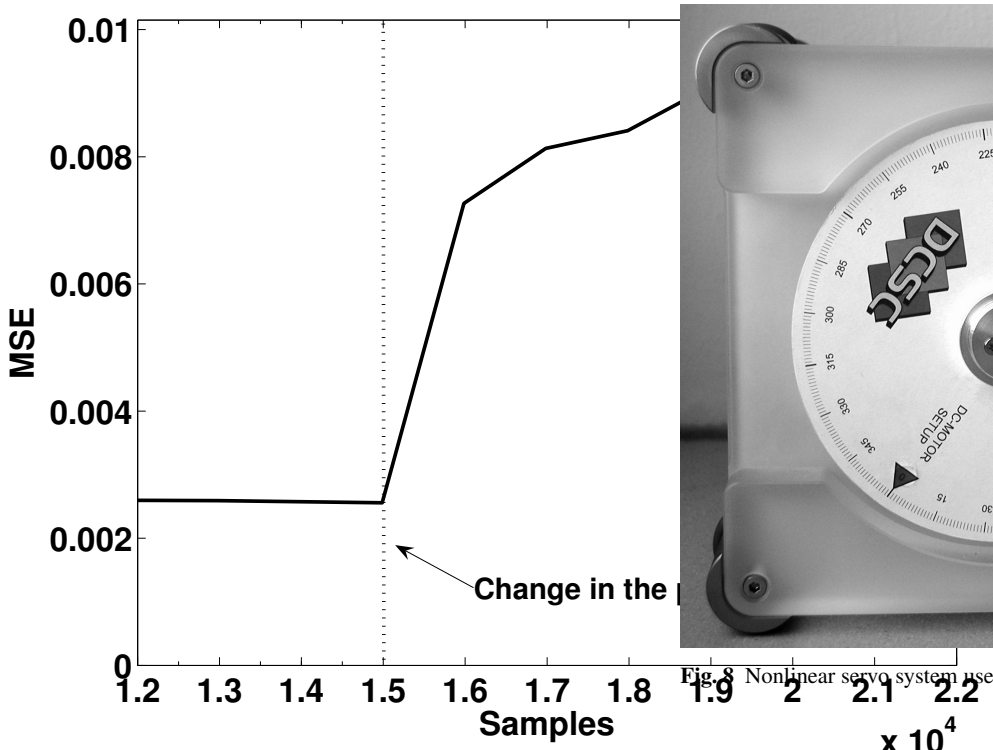


Fig. 7 Response of the OSEFC to an unexpected change in the dynamics of the plant for Example 1

this behavior, we have executed the same simulation as described above. However, at sample 15000, the definition of the equation of the plant was changed as follows:

$$y_{k+1} = -0.3\sin(y_k) + 0.8u_k + u_k^3 \tag{25}$$

The only difference with the previous definition given in (23) is the multiplying factor applied to u_k. This means that the actuator's response is decreased, for instance, due to an incipient fault. Fig. 7 depicts the evolution of the MSE before and after the change in the dynamics of the plant. It is observed that the change in the plant causes a sudden increase in the mean square error. However, after some time and due to the joint action of both the LL-module and the GL-module, the error decreases again.

6.2 OSEFC application to a real-world system

In this section the proposed OSEFC is applied to a nonlinear servo system consisting of a DC motor with an extra weight (see Fig. 8). The control objective is to compensate the nonlinearities caused by the gravity term caused by the extra weight and to make the system's output θ track a desired trajectory given by the reference signal r. The proposed controller does not need any information about the differential equations that govern the plant. However, this information is summarized in the Appendix for those readers who are interested in simulating the experiment. Note that all the results reported were obtained in real-time experiments.

Fig. 8 Nonlinear servo system used in Example 2

For this experiment we have applied a fuzzy controller with three inputs: the reference value r, the current angle θ, and the angular velocity $\dot{\theta}$, with values ranging in $[-\pi,\pi]$, $[-\pi,\pi]$, and $[-5,5]$, respectively. Initially, two membership functions are assigned to each input, thus 8 fuzzy rules form the initial controller. However, all the consequents are initialized to zero, so the situation is equivalent to having an empty controller. Fig. 9 presents the initial membership functions assigned to the input $\dot{\theta}$. The initial configuration for the inputs r and θ is the same, but in the range $[-\pi,\pi]$.

The actuator's range is limited to $[-10,10]$ V. As the monotonicity of the plant with respect to the control signal u is positive, the constant C for the adaptation of the consequents is positive. According to (9), its value is set to $C = 10/\pi$. The sampling period is chosen as $h = 0.05$s. The reference signal is given by a function of random values in the interval $[-\pi,\pi]$, forming a 1000-sample long pattern (i.e., 50-second long pattern). The total time for the experiment has been 45 minutes.

Fig. 10 (a), 11 (a) and 12 (a) show the tracking of the reference signal at three different moments of the execution. The solid line represents the reference signal and the dotted line corresponds to the plant's output. At the beginning of the execution (Fig. 10 (a)), the lack of knowledge about the plant causes oscillations in the plant's output. However, after only 20 s, the system starts following the trajectory marked by the reference signal, although the control is still not satisfactory. In Fig. 11 (a) the effect of the global learning can be

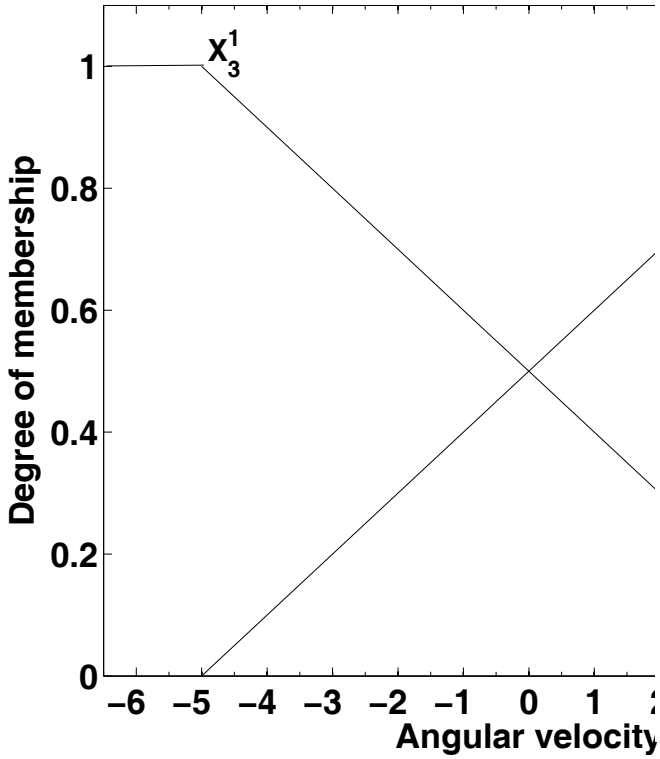


Fig. 9 Initial MFs for the input $\dot{\theta}$ in Example 2

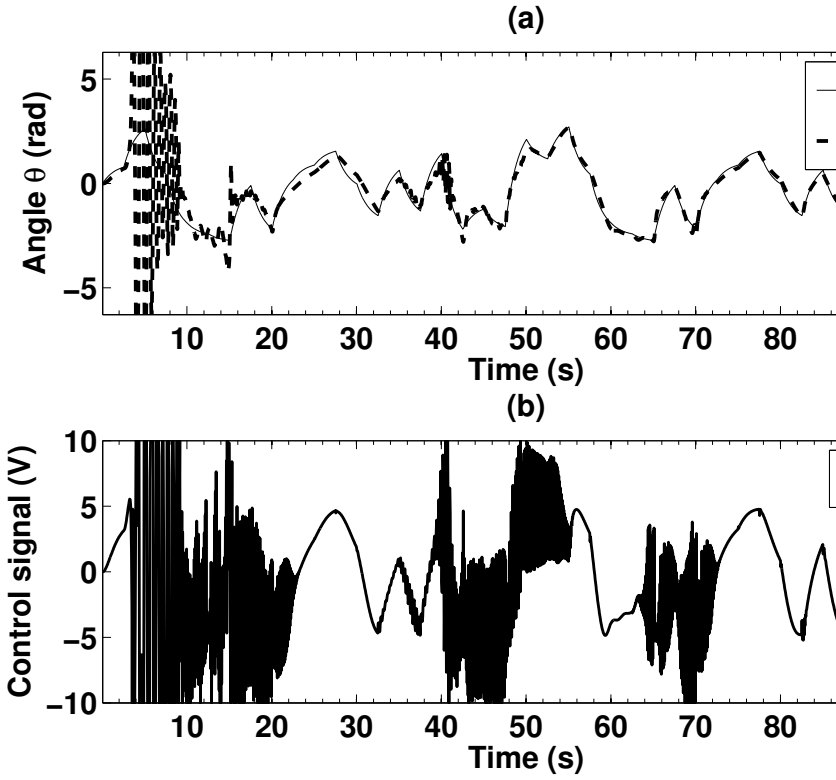


Fig. 10 Reference tracking during the first 100 seconds in Example 2. (a) Reference tracking. (b) Control signal

observed. At time 1238 s the GL-block starts operating. As it can be seen in Fig. 11 (a), the action of this block eliminates the small oscillations that were present before. Finally, Fig. 12 (a) depicts the control at the end of the experiment. It can be observed that the tracking at this point is virtually perfect. Fig. 10 (b), 11 (b) and 12 (b) show the control signal applied during the execution.

Table 1 summarizes the evolution of the controller's structure. Each row represents a topology change, showing the time of change, the new MF distribution, the values of the indexes IR_i used to select the variable to receive the new MF, and the MSE reached with that topology after tuning the new consequents. In the first two changes it is clear that the second input (the angle θ) is the most important, as the difference between its IR and the others is high. However, after optimizing the consequents of the topology 2x4x2, IR_1 and IR_2 have very similar values, the former being slightly higher. This means that, at this point, the error due to the plant's output has been corrected by the two extra MFs and the lack of precision on the first variable (the reference value) is more relevant. At the end of the execution, both inputs are equally important for the control, as they both have the same number of MFs. However, the initial configuration of the input $\dot{\theta}$ does not change. This means that a linear approximation is sufficient in that dimension.

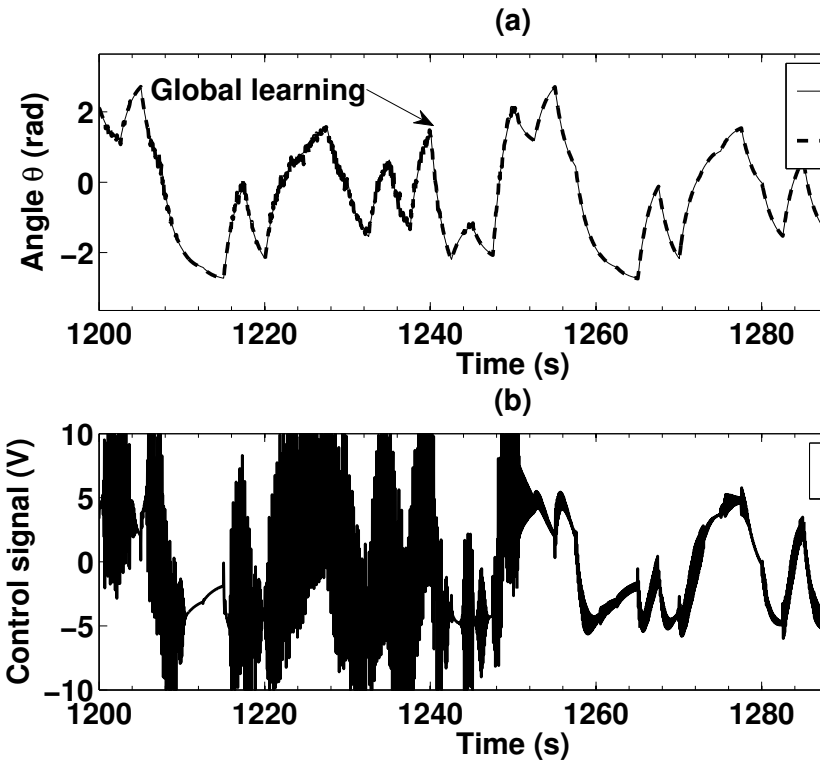


Fig. 11 Effect of the global learning on the reference tracking in Example 2. (a) Reference tracking. (b) Control signal

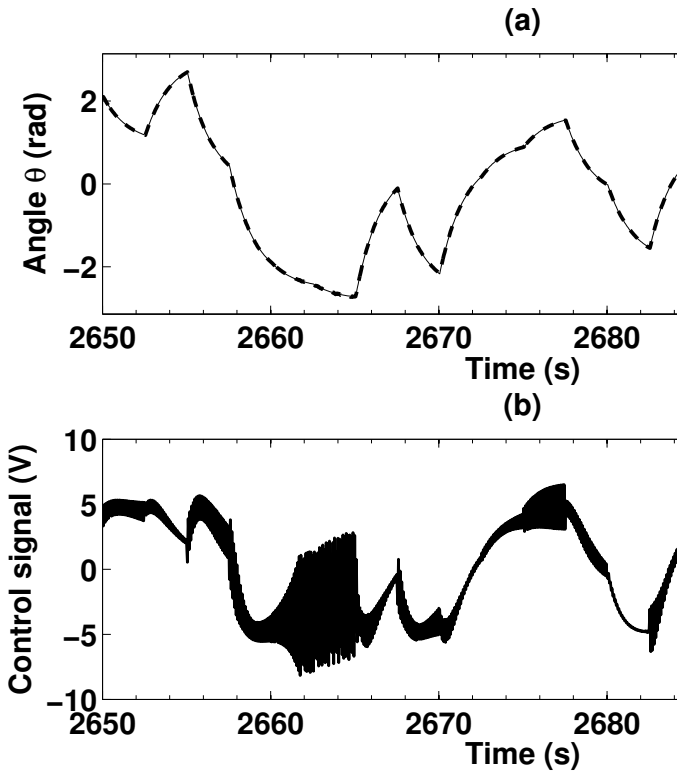


Fig. 12 Reference tracking at the end of the experiment 2. (a) Reference tracking. (b) Control signal

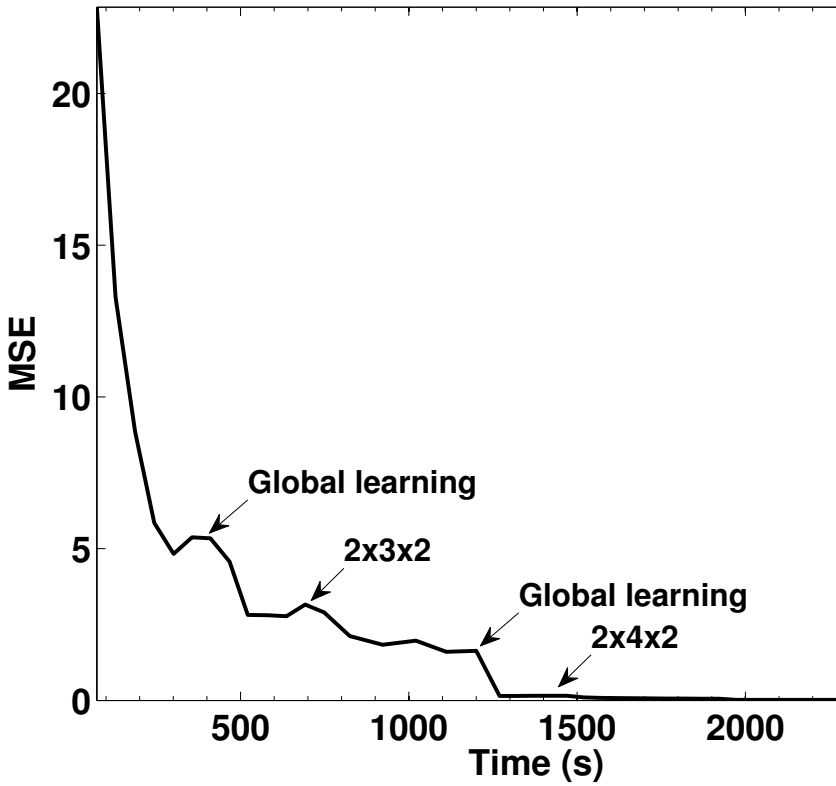


Fig. 13 Evolution of the mean square error in Example 2

Table 1 Topology evolution for the nonlinear servo system

Time (s)	Configuration	IR_1	IR_2	IR_3	MSE
0	2x2x2	7.761	18.531	4.538	2.897
748.3	2x3x2	13.038	29.533	7.512	0.153
1469.5	2x4x2	12.899	12.609	7.318	0.191
2173.7	3x4x2	12.033	12.470	7.135	0.020
2274.5	3x5x2	11.875	6.714	6.997	0.019
2376.4	4x5x2	11.703	6.652	6.978	0.015
2685.6	5x5x2				0.014

Fig. 13 shows the evolution of the MSE during the execution and Fig. 14 depicts the detail of this evolution once the error reaches small values. The moments in which topology changes have occurred are pointed out, as well as the moments in which global learning starts for some topologies. In both figures a clear decreasing tendency of the error is observed, which is faster at the beginning but continues until the end. Again, the global learning improves the performance of each topology, helping to reduce the number of topology changes.

Finally, we show the structure of the final fuzzy controller. Fig. 15 and 16 depict the final location of the membership functions of the two first input variables (r_k and θ). The final MFs for the third input ($\dot{\theta}$) are the same as at the beginning (see Fig. 9). At the end of the experiment, the controller is formed by $5 \cdot 5 \cdot 2 = 50$ rules. These are de-

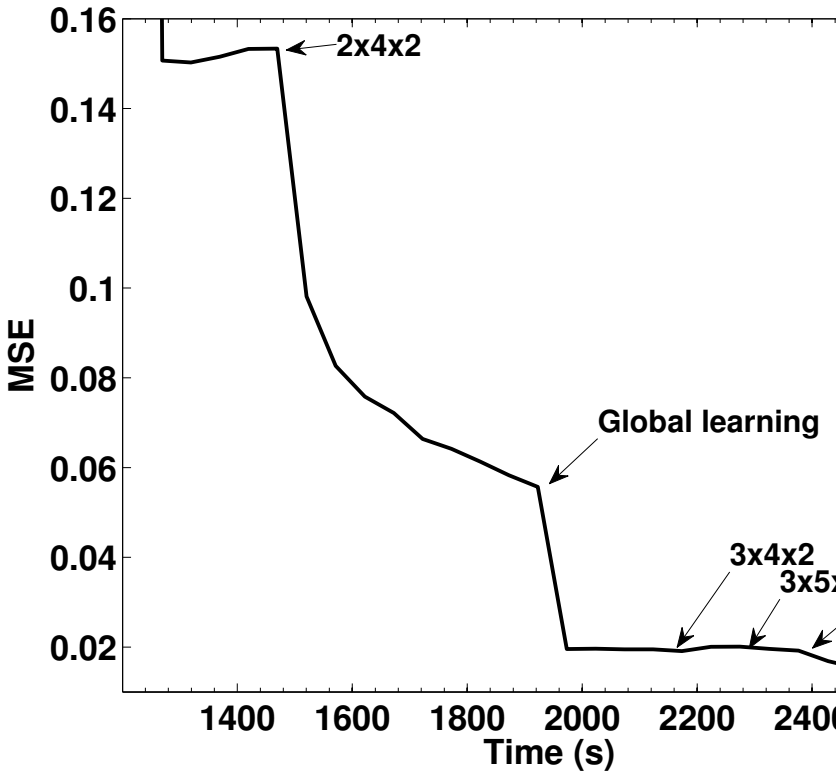


Fig. 14 Details of the evolution of the MSE in Example 2

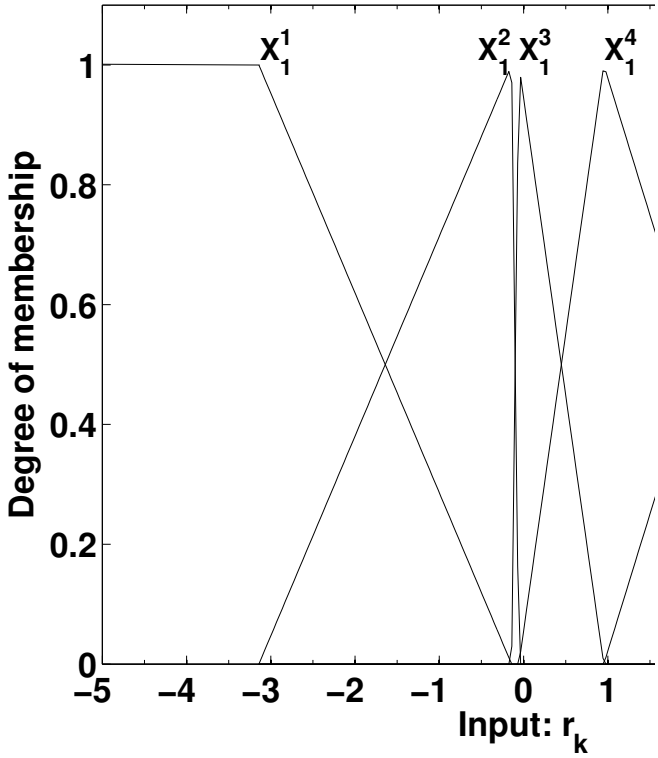


Fig. 15 Final location of the membership functions assigned to the input $r(k)$ in Example 2

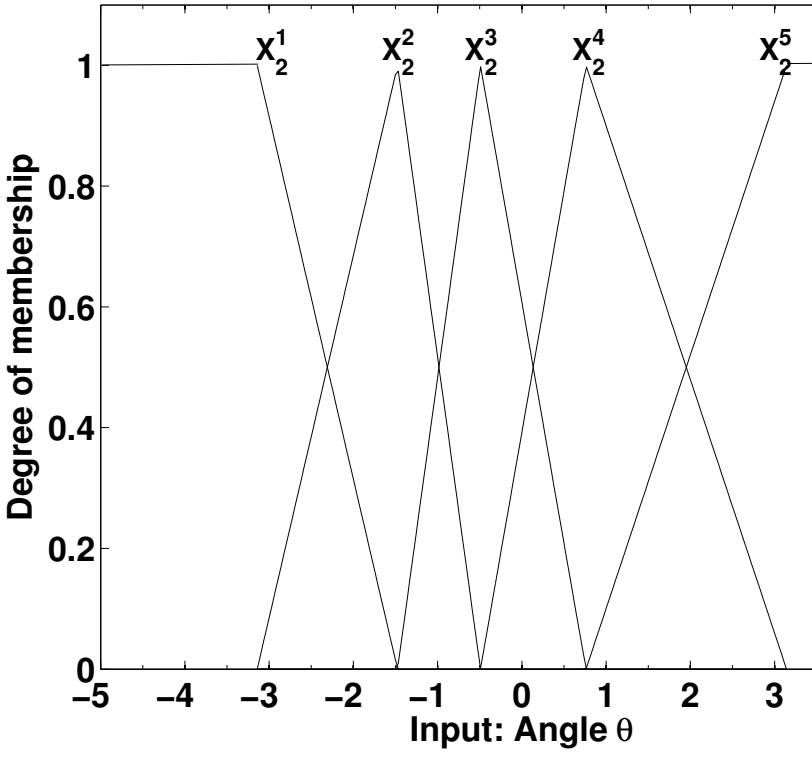


Fig. 16 Final location of the membership functions assigned to the input θ in Example 2

Table 2 Final fuzzy rules for $\dot{\theta} = X_3^1$

	Input r				
Input θ	X_1^1	X_1^2	X_1^3	X_1^4	X_1^5
X_2^1	4.7721	52.1996	53.6901	69.7312	104.5669
X_2^2	-30.8451	23.5035	24.9537	43.3321	83.2436
X_2^3	-48.4456	8.3943	10.3832	29.4678	70.9134
X_2^4	-70.1487	-10.0520	-8.4621	11.4745	54.7989
X_2^5	-90.8356	-45.5182	-44.1779	-28.9516	4.1149

Table 3 Final fuzzy rules for $\dot{\theta} = X_3^2$

	Input r				
Input θ	X_1^1	X_1^2	X_1^3	X_1^4	X_1^5
X_2^1	-4.7673	43.3722	44.9777	61.2631	96.6294
X_2^2	-39.8072	15.1472	16.5961	35.0585	75.1525
X_2^3	-56.5496	0.2467	2.2119	21.1992	62.4334
X_2^4	-77.4427	-18.3305	-16.7001	2.9824	45.7843
X_2^5	-98.5657	-53.8703	-52.5698	-37.5362	-4.8880

picted in two tables: Table 2 represents the consequents of the 25 rules for MF X_3^1, while Table 3 represents the consequentes for the remaining 25 rules, for MF X_3^2. Note that we identify the rules by the membership functions that form their antecedent part. In these tables, only the centers of the MFs are shown because we are using normalized triangular functions.

7 Conclusions

In this work, an online self-evolving fuzzy controller has been presented. This controller learns based on I/O data obtained while controlling the plant, so no pretraining is required. The proposed method adapts the values of the fuzzy rule consequents in two different ways: local learning is applied to direct the plant's output to the current target value and global learning is used to improve the control in the remaining operating space. Besides, the controller's structure is adjusted by the addition of more membership functions in the controller's inputs with higher responsibility in the approximation error. Thanks to these structural changes, the controller is able to start working with an empty or very simple structure and evolves to provide satisfactory results. Simulation and experimental results have been used to show the capabilities of the controller developed.

Appendix

The nonlinear servo system used for the experimentation in Section 6.2 is modeled by the following differential equa-

Table 4 Nonlinear servo system's parameters

Symbol	Parameter	Value
J	Moment of inertia of the rotor	$1.91 \cdot 10^{-4} kg \cdot m^2$
b	Damping of the mechanical system	$3 \cdot 10^{-6} kg/s$
K	Electromotive force constant	$0.0536 Nm/A$
R	Electric resistance	9.5Ω
m	Mass of the weight	$0.055 kg$
l	Length center of disk-center of mass	$0.042 m$
g	Gravity acceleration	$9.81 m/s^2$

tions:

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x},u) = A\mathbf{x} + Bu + G \\ &= \begin{bmatrix} 0 & 1 \\ 0 & \frac{-K^2-bR}{RJ} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \frac{K}{RJ} \end{bmatrix} u - \begin{bmatrix} 0 \\ \frac{mgl}{J} \sin(\theta) \end{bmatrix}\end{aligned} \tag{26}$$

where $\mathbf{x} = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T \in \mathbb{R}^2$ is the state of the plant, given by the angle θ and the angular velocity $\dot{\theta}$, u is the control input, and G is the gravity term. The plant's output is the angle θ. The meaning and values of the parameters used for the DC motor are given in Table 4.

As mentioned before, the proposed OSEFC does not require the differential equations that govern the plant to be controlled. However, we have included this information in this appendix with the purpose of allowing the interested reader to simulate the experiment.

References

Angelov P (2004) A fuzzy controller with evolving structure. Information Sciences 161(1-2):21–35

Cara A, Lendek Z, Babuska R, Pomares H, Rojas I (2010) Online self-organizing adaptive fuzzy controller: application to a nonlinear servo system. In: Proc. 2010 IEEE World Congress on Computational Intelligence, pp 2491–2498

Castro J (1995) Fuzzy logic controllers are universal approximators. IEEE Trans Syst, Man, Cybern 25(4):629–635

Chen CH, Lin CJ, Lin CT (2009) Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. IEEE Trans Syst, Man, Cybern C 39(4):459–473

Galluzzo M, Cosenza B (2009) Control of the biodegradation of mixed wastes in a continuous bioreactor by a type-2 fuzzy logic controller. Computers & Chemical Engineering 33(9):1475–1483

Gao Y, Er MJ (2003) Online adaptive fuzzy neural identification and control of a class of MIMO nonlinear systems. IEEE Trans Fuzzy Syst 11(4):462–477

Gao Y, Er MJ (2005) An intelligent adaptive control scheme for postsurgical blood pressure regulation. IEEE Trans Neural Netw 16(2):475–483

Hoffmann F, Nelles O (2001) Genetic programming for model selection of TSK-fuzzy systems. Information Sciences 136(1-4):7–28

Lalouni S, Rekioua D, Rekioua T, Matagne E (2009) Fuzzy logic control of stand-alone photovoltaic system with battery storage. Journal of Power Sources 193(2):899–907

Lee C (1990) Fuzzy logic in control systems: fuzzy logic controller. II. IEEE Trans Syst, Man, Cybern 20(2):419–435

Li C, Lee C (2003) Self-organizing neuro-fuzzy system for control of unknown plants. IEEE Trans Fuzzy Syst 11(1):135–150

Lin C, Xu Y (2006) A novel genetic reinforcement learning for nonlinear fuzzy control problems. Neurocomputing 69(16-18):2078–2089

Lin CT, Lin CJ, Lee CSG (1995) Fuzzy adaptive learning control network with on-line neural learning. Fuzzy Sets and Systems 71(1):25–45

Liu Y, Zheng Y (2009) Adaptive robust fuzzy control for a class of uncertain chaotic systems. Nonlinear Dynamics 57(3):431–439

Mingzhi H, Jinquan W, Yongwen M, Yan W, Weijiang L, Xiaofei S (2009) Control rules of aeration in a submerged biofilm wastewater treatment process using fuzzy neural networks. Expert Systems with Applications 36(7):10,428–10,437

Mucientes M, Casillas J (2007) Quick design of fuzzy controllers with good interpretability in mobile robotics. IEEE Trans Fuzzy Syst 15(4):636–651

Park J, Park G, Kim S, Moon C (2005) Direct adaptive self-structuring fuzzy controller for nonaffine nonlinear system. Fuzzy Sets and Systems 153(3):429–445

Phan PA, Gale TJ (2008) Direct adaptive fuzzy control with a self-structuring algorithm. Fuzzy Sets and Systems 159(8):871–899

Pomares H, Rojas I, Ortega J, Gonzalez J, Prieto A (2000) A systematic approach to a self-generating fuzzy rule-table for function approximation. IEEE Trans Syst, Man, Cybern B 30(3):431–447

Pomares H, Rojas I, Gonzalez J, Prieto A (2002a) Structure identification in complete rule-based fuzzy systems. IEEE Trans Fuzzy Syst 10(3):349–359

Pomares H, Rojas I, Gonzalez J, Rojas F, Damas M, Fernandez FJ (2002b) A two-stage approach to self-learning direct fuzzy controllers. International Journal of Approximate Reasoning 29(3):267–289

Pomares H, Rojas I, Gonzalez J, Damas M, Pino B, Prieto A (2004) Online global learning in direct fuzzy controllers. IEEE Trans Fuzzy Syst 12(2):218–229

Rojas I, Pomares H, Ortega J, Prieto A (2000) Self-organized fuzzy system generation from training examples. IEEE Trans Fuzzy Syst 8(1):23–36

Rojas I, Pomares H, Gonzalez J, Herrera L, Guillen A, Rojas F, Valenzuela O (2006) Adaptive fuzzy controller: Application to the control of the temperature of a dynamic room in real time. Fuzzy Sets and Systems 157(16):2241–2258

Wang D, Zeng XJ, Keane JA (2008a) An incremental construction learning algorithm for identification of t-s fuzzy systems. In: Proc. 2008 IEEE International Conference on Fuzzy Systems (FUZZ 2008), vol 1-5, pp 1662–1668

Wang W, Chien Y, Li I (2008b) An on-line robust and adaptive T-S fuzzy-neural controller for more general unknown systems. International Journal of Fuzzy Systems 10(1):33–43

Ying H (2000) Fuzzy Control and Modeling: Analytical Foundations and Applications. Wiley-IEEE Press