Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Modeling and control of nonlinear systems using an Adaptive LAMDA approach

Luis Morales [a,*], Jose Aguilar [b,c], Andrés Rosales [a], Danilo Chávez [a], Paulo Leica [a]

[a] *Departamento de Automatización y Control Industrial, Escuela Politécnica Nacional, Calle Ladron de Guevara E11-253, 170517, Quito, Ecuador*
[b] *CEMISID, Facultad de Ingeniería, Universidad de Los Andes, Mérida 5101, Venezuela*
[c] *GIDITIC, Universidad EAFIT, Medellín, Colombia*

ARTICLE INFO

Article history:
Received 1 May 2020
Received in revised form 9 July 2020
Accepted 19 July 2020
Available online 25 July 2020

Keywords:
LAMDA
Intelligent control
Online learning
Adaptive fuzzy models

ABSTRACT

This paper presents a soft computing technique for modeling and control of nonlinear systems using the online learning criteria. In order to obtain an accurate modeling, and therefore a controller with good performance, a method based on the fundamentals of the artificial intelligence algorithm, called LAMDA (Learning Algorithm for Multivariate Data Analysis), is proposed, with a modification of its structure and learning method that allows the creation of an adaptive approach. The novelty of this proposal is that for the first time LAMDA is used for fuzzy modeling and control of complex systems, which is a great advantage if the mathematical model is not available, partially known, or variable. The adaptive LAMDA consists of a training stage to establish initial parameters for the controller, and the application stage in which the control strategy is computed and updated using an online learning that evaluates the closed-loop system. We validate the method in several control tasks: (1) Regulation of mixing tank with variable dead-time (slow variable dynamics), (2) Regulation of a Heating, Ventilation and Air-Conditioning (HVAC) system (multivariable slow nonlinear dynamics), and (3) trajectory tracking of a mobile robot (multivariable fast nonlinear dynamics). The results of these experiments are analyzed and compared with other soft computing control techniques, demonstrating that the proposed method is able to perform an accurate control through the proposed learning technique.

1. Introduction

The evolution of artificial intelligence has allowed the development of very powerful techniques, useful for modeling nonlinear systems whose dynamics are complex and unknown [1]. The development of these techniques has increased considerably due to the computational power of the computers, allowing the implementation of learning algorithms with high accuracy and fast in processing terms, considering the inherent uncertainty and changing conditions of the systems [2]. Due to the versatility of these methods, it is possible to perform offline and online system modeling [3], which are very useful in control schemes, especially when the mathematical model of the system is unknown or variable. Specifically, in the case of online learning, the main advantage is the adaptation to changes in the dynamics of the system to be modeled/controlled, since it learns constantly the behavior of the process based on the input and output data. Many of the applications related to industrial processes, aeronautics, robotics and power systems require the incorporation of artificial intelligence into the control schemes, due to the adaptive feature that it provides when the mathematical model is complex, unknown or inaccurate. The most used approaches for modeling and control of systems are Artificial Neural Networks (ANN) [4–6], Fuzzy Logic [7], and the hybrid models between the ANNs and the Fuzzy Inference Systems [8,9]. This approach is considered as a universal approximator [10], with the ability to represent any parameterized model. The adaptive neuro-fuzzy inference systems (ANFIS) are schemes that combine characteristics of both models (neuronal and fuzzy), with a fixed structure of nodes and layers, using the criteria of the neural networks for the learning process to perform the parametric adjustment, showing excellent results in different application fields related to model and control [11–15].

The aforementioned approaches are generally used in schemes of Adaptive Inverse Control (AIC). This control methodology has been studied for the last three decades, in which different researchers have made interesting proposals, applied to unknown plant dynamics [16]. Neural networks are used for this purpose, finding a great deal of information in the literature (for more

* Corresponding author.
E-mail addresses: luis.moralesec@epn.edu.ec (L. Morales), aguilar@ula.ve (J. Aguilar), andres.rosales@epn.edu.ec (A. Rosales), danilo.chavez@epn.edu.ec (D. Chávez), paulo.leica@epn.edu.ec (P. Leica).

https://doi.org/10.1016/j.asoc.2020.106571
1568-4946/© 2020 Elsevier B.V. All rights reserved.

detail, see [3,17–22]). One of the works where the potential of these methods is presented is [23]. In this work, an AIC strategy is proposed for variable speed wind turbine, which is a clear example of applying intelligent algorithms to model and control complex systems. The neural networks are used for inverse learning of the wind turbine, and additionally, this proposal requires as information the sensitivity of the process to the control input. The main problem of this methodology is that an additional neural network is required to estimate the process sensitivity, which increases the complexity and computational cost of the control scheme. The two neural networks are firstly offline trained; then, they are online updated with a backpropagation algorithm. A similar approach is presented in [24] based on support vector machines (SVM) for the excitation system with changing parameters in the actuator, improving the performance in the transient and system damping. Recurrent Neural Networks (RNN) are a variation of the previous ones used in the design of identifiers and controllers; however, they are difficult to train due to the problem of fading gradients computed with backpropagation through time [25]. An interesting solution to this issue is proposed by the reservoir computing paradigm, whose main idea is that only the output weights of the network are trained and the internal connections are randomly initialized, such that the dynamics of the network are at the edge of stability [6]. The main problem with this proposal is the stability analysis and the definition of the number of neurons in the reservoir, which must be heuristically calibrated.

Fuzzy logic is an excellent tool for modeling systems, it is generally rule-based and has had a wide field of application in system control, facilitating the design of applicable nonlinear controllers from simple systems to complex chaotic systems [26,27]. Fuzzy logic has also been used in the design of AIC schemes based on the creation of rules whose parameters can adapt automatically through learning criteria. For instance, in [28], to compensate the motion inaccuracies of a gripper rotating system owing to dead zones of unknown characteristics via the Takagi–Sugeno (T–S) fuzzy model. The adaptive updating is based on the gradient projection technique to update the inverse dead-zone parameters online, which improves the performance of the entire control system. A similar work applies T–S models to model and compensate for unknown dead zones in actuators [29].

Finally, the neuro-fuzzy systems have also been developed in AIC. In [30], the position control of a switched reluctance motor is presented. The proposed method, based on type-2 fuzzy neural networks, could realize high precise position control for the SRM under different working conditions. In [31], a neuro-fuzzy inference system with learning based on a particle swarm optimization algorithm has been developed, compared with ANFIS, and applied to control of the temperature of a water system. The use of hybrid AIC schemes, combined with PID, has been implemented to regulate an exothermic Continuous Stirred-Tank Reactor (CSTR) to eliminate the static error, which is trained through an improved nearest-neighborhood clustering algorithm and a gradient descent algorithm [32]. The authors of [33] have implemented the learning of the neuro-fuzzy model based on the global evolutionary Big Bang-Big Crunch (BB-BC) optimization algorithm. The model is used to generate the optimal fuzzy inverse model output as the control signal at every sample time, applying their method in two slow dynamic systems. However, no variation of plant parameters is observed to analyze the potential of the proposal. AIC based on neuro-fuzzy approaches are also used in the field of robotic, since kinematic, and especially dynamic models, are often complex to obtain due to the forces that interact in each degree of freedom, being adaptive models useful to compensate the errors in modeling the dynamics [34] . In this work, the authors demonstrate that the proposed adaptive inverse dynamics control scheme is effective to improve the control performance of the system with uncertainties. In the literature, several works have made efforts to improve the performance of neuro-fuzzy models through the use of hybrid learning [8], self-tuning ANFIS based in genetic programming [35], learning based on square-root cubature Kalman filter (SCKF) or recursive least squares (RLS) [36], or learning techniques that use input space partitioning through sub-clustering for higher dimensional regression problems (extreme learning [37]).

The background that has been analyzed related to AIC based on different learning methods, and the advantages that these systems present when working with systems of unknown or uncertain characteristics, have motivated in this work to propose a new approach based on fuzzy logic features. The technique we take as a starting point for our work is LAMDA (Learning Algorithm for Multivariate Data Analysis) [38], which has presented interesting results for both classification [39] and clustering [40] problems. Our research contribution consists in proposing an adaptive learning method for the LAMDA parameters update, which allows to control a system through the detection of functional states, the theory on which this algorithm is based, without requiring the process model.

The LAMDA algorithm can be considered as a white box because it involves simple mathematical operations, which reduce the complexity in the programming. The proposed approach has the additional advantage of handling a set of fixed hidden layers, which is usually a drawback because the designer must specify this parameter, as in the case of conventional neural networks. This algorithm originally adjusts its internal parameters in the learning stage, handling the concept of adequacy degree, for which is required the computation of two parameters. The first one is called the Marginal Adequacy Degree (MAD), which is calculated through the contribution of all descriptors of an individual using fuzzy probability functions [41]. The MADs of an individual in each class are combined through the use of fuzzy aggregation operators, resulting a second parameter, called Global Adequacy Degree (GAD), which corresponds to the membership degree of the individual to a class. Then, the GADs allow identifying the class or cluster where the individual must be assigned.

LAMDA has been used in several applications, especially in the identification of functional states of systems, with satisfactory results in classification and clustering tasks [40,42–47]. In the most recent works, we have proposed for the first time LAMDA as a non-adaptive controller, adding to the algorithm a T–S inference stage, obtaining a class-based controller. This proposal was tested in two different systems, the first, in a SISO system corresponding to the temperature control of a mixing tank with variable dynamics [48]. The second, in a MIMO system corresponding to a Heating, Ventilation and Air-Conditioning (HVAC) system for regulation of Temperature and Humidity [49]. In both studies, the proposal presented promising results, without considering the mathematical model of the plants for its design. The class-based controllers were defined and calibrated by an expert who has knowledge of the behavior of the systems. In general, the design of these controllers can be a complex and time-consuming process, depending on the system to be controlled. Another disadvantage is that the controllers have not the ability to automatically learn or be adaptive, because their internal values (class centers, exigency degree, and consequent parameters) are set at the design stage, and they do not change during the operation of the control in the processes, which is useful in applications where the system dynamics is unknown or variable.

In order to solve the problems of defining the LAMDA classes, calibration of input and output scaling gains, and non-adaptability of controller parameters, in this work, we propose a novel approach that consists in creating an Adaptive LAMDA model that allows making an online modeling of the plant and its control, for which the learning strategy of LAMDA is enhanced to replicate the behavior of the plant. The adaptability feature through online learning avoids performing the calibration stage that is time-consuming and often complex.

The proposed Adaptive LAMDA model adjusts its internal parameters, so that the output of the algorithm is as similar as possible to the real output data, based on the input data (descriptors/features). For this, it is essential to expand the LAMDA model, and give it an optimization criterion (a different learning technique) to adjust its internal parameters. A proper fit allows a high accuracy modeling, which will result in a good control system performance. Offline modeling schemes are suitable when considering invariant-time plants and without disturbances, since in the presence of these factors, the offline control scheme will result in an inappropriate behavior in the computed control actions, presenting steady-state error [50]. For this reason, the online learning scheme is appropriate in most control objectives. Regarding previous works, the contributions, novelties and advantages of our proposal are the following:

- Unlike LAMDA without self-learning presented in [49], the model parameter adjustment is proposed, avoiding the calibration stage and class parameterization.
- The implementation of LAMDA as an identifier is proposed for the first time, handling the concept of self-adjustment of the exigency (α) and the antecedent parameters used for the GADs calculation.
- A stability analysis of the learning algorithm is proposed to guarantee a rapid convergence of the estimated output towards the desired output.
- Our scheme, with respect to approaches such as those presented in [23,24,51], does not required to compute the output/input gradient of the system to be controlled, which reduces the computational cost.
- Our approach has a known number of hidden layers, which is an advantage with respect to algorithms such as those presented in [6,25], avoiding the heuristic definition of the number of internal layers.
- The proposed learning for LAMDA is based on a hybrid learning, which allows a quick convergence to the desired output, improving the learning time and preventing that solutions be trapped in local minima. This is a great advantage over learning methods that only work with gradient descent, which is generally slow [52].
- The modeling and control of nonlinear systems are based on the concept of classes or functional states established by the LAMDA theory.

The experiments carried out have demonstrated the ability to learn the control actions, for different dynamic behaviors. Due to this, our proposal can be applied to a wide variety of nonlinear systems. This paper is organized as follows, Section 2 presents a brief review of the fundamentals and operations performed in the original LAMDA for classification and clustering. Section 3 formalizes the proposed scheme and the established learning process, detailing the parameter adjustment methods for the Adaptive LAMDA. Section 4 shows in detail the proposed Adaptive LAMDA approach applied to control systems. Section 5 presents the test and results in three different case studies with different dynamics, to solve regulation and tracking trajectory problems, in order to validate our online control scheme. Finally, Section 6 concludes the paper.

2. LAMDA

In this section, we present a brief review of the theoretical framework of LAMDA. This algorithm is a fuzzy approach focused on the concept of the adequacy degree that can be used for classification and clustering. In supervised learning context, LAMDA performs a similarity evaluation (adequacy degree) of an individual $X = \left[x_1, \ldots, x_j, \ldots, x_n\right]^T$ (where n is the number of descriptors) to each class $C = \{C_1, \ldots, C_k, \ldots, C_m\}$ (where m is the number of classes), to define where the individual should be assigned.

To initialize the learning process, the descriptors are normalized $\bar{x}_j \in [0, 1]$. This procedure is performed considering the maximum x_{jmax} and minimum x_{jmin} values as:

$$\bar{x}_j = \frac{x_j - x_{jmin}}{x_{jmax} - x_{jmin}} \tag{1}$$

The normalized individual $\overline{X}$ represented for the normalized descriptors computed by (1) is used for calculating the adequacy degrees for each class.

2.1. Marginal Adequacy Degree (MAD)

This parameter computes the degree of similarity between the descriptor of an individual and the same descriptor in each class. To compute this parameter are considered probability density functions [53], one of the most used is the Gaussian function (2), which assumes a normal distribution of the descriptor.

$$MAD_{k,j}(\bar{x}_j, \rho_{k,j}, \sigma_{k,j}) = e^{-\frac{1}{2}\left(\frac{\bar{x}_j-\rho_{k,j}}{\sigma_{k,j}}\right)^2} \tag{2}$$

where $\rho_{k,j}$ is the average value of the descriptor j that belongs to the class k, and $\sigma_{k,j}$ is the standard deviation of the descriptor j that belongs to the class k.

2.2. Global Adequacy Degree (GAD)

The Global Adequacy Degree (GAD) is computed combining the MADs for each individual $\overline{X}$ in each class, using fuzzy logic connectors as aggregation operators. These connectors can be of intersection (t-norm "T") or union (t-conorm "S"). Thus, different aggregation operators can be used, among them, the Product-Probabilistic sum (see Eq. (3)) or Dombi (see Eq. (4)).

$$T(a, b) = ab; S(a, b) = a + b - ab \tag{3}$$

$$T(a, b) = \frac{1}{1 + \sqrt[p]{\left(\frac{1-a}{a}\right)^p + \left(\frac{1-b}{b}\right)^p}};$$

$$S(a, b) = 1 - \frac{1}{1 + \sqrt[p]{\left(\frac{a}{1-a}\right)^p + \left(\frac{b}{1-b}\right)^p}} \tag{4}$$

To classify the data in a strict or permissible manner, the exigency parameter $\alpha \in [0, 1]$ is required to calibrate the fuzzy partition data. If $\alpha = 1$, then the fuzzy partition data is calculated only by the t-norm. It means that the classification is stricter. If $\alpha = 0$, then the fuzzy partition data is calculated by the t-conorm. It means that classification is more permissible, therefore, samples are assigned to a class despite not having enough similarity with the individuals belonging to it. The exigency parameter produces a linear interpolation between the t-norm and t-conorm for the GADs, which are computed by:

$$\begin{aligned} &GAD_{\overline{X},k}\left(MAD_{k,1}, \ldots, MAD_{k,j}, \ldots, MAD_{k,n}, \alpha\right) \\ &\quad = \alpha T\left(MAD_{k,1}, \ldots, MAD_{k,j}, \ldots, MAD_{k,n}\right) \\ &\qquad + (1-\alpha) S\left(MAD_{k,1}, \ldots, MAD_{k,j}, \ldots, MAD_{k,n}\right) \end{aligned} \tag{5}$$

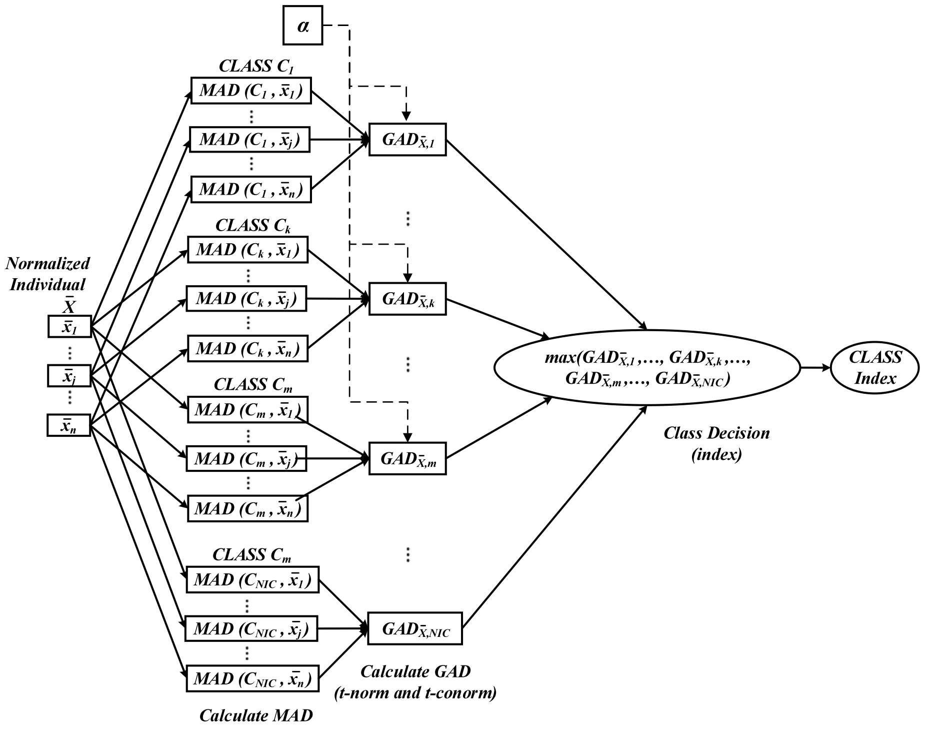


Fig. 1. LAMDA scheme.

LAMDA, with respect to other classifiers, has the capability to create new classes after the training stage, for which it is based on a threshold known as the Non-Informative Class (NIC). The possibility of finding MAD depends on how the data are distributed in each class. For the NIC, it is considered $\rho_{NIC,j} = 0.5$, and $\sigma_{k,j} = 0.25$.

The algorithm compares all the GADs, and assign the object to the class with the highest value (6). If the GAD of the NIC has the highest value, then a new class or cluster is created.

$$index = \max\left(GAD_{\overline{X},1}, GAD_{\overline{X},k}, \ldots, GAD_{\overline{X},m}, GAD_{\overline{X},NIC}\right) \tag{6}$$

The LAMDA operation scheme for classification is presented in Fig. 1, in which the steps described above are shown. They are sequentially interconnected from the acquisition of the standardized descriptors to determine the class to which the individual is assigned.

In control applications, the importance of the self-adjustment of parameters of each class of the original model is detailed in [49]. Furthermore, the exigency parameter α affects the controller output in terms of performance, especially if the dynamic of the plant is variable.

As seen in the LAMDA procedure, the functioning as a classification model is quite simple, however, in online learning control tasks an adaptive model is required. Hence, we propose the design of an Adaptive LAMDA, whose internal parameters are updated online based on the data of the system to be controlled, it implies the implementation of a learning method different from the one proposed for classification and clustering.

3. Adaptive LAMDA model

In this section, we formalize our proposed scheme and the learning process, detailing the modifications to the original structure of LAMDA and the parameter adjustment process of the model.

3.1. Structure of the proposed adaptive LAMDA

The original LAMDA presents a fairly good performance in classification and clustering applications, however, for modeling and control, the algorithm needs to work as a regressor with the feature of online self-adjustment of parameters, for which the addition of layers and a different learning method is required. In this paper, the addition of a first-order T–S fuzzy inference system to LAMDA is proposed, due to the excellent results that this method presents for systems modeling and control. This methodology establishes that the output of each class is represented as a linear combination of input descriptors, plus a constant parameter. Finally, the last output is the weighted average of each class output.

The implementation of the T–S fuzzy inference system applied to LAMDA requires the addition of layers 3, 4 and 5 to the original model presented in Fig. 1. Fig. 2 shows the proposed scheme for the adaptive model, which takes an individual $\overline{X}$ for the computation of the outputs from Layer 1 to Layer 5.

The scheme of Fig. 2 corresponds to a MISO (Multiple-Input Single-Output) system, with 5 layers, each one with a specific function:

Layer 1 each node in this layer corresponds to compute the $MAD_{k,j}(\overline{x}_j, \rho_{k,j}, \sigma_{k,j})$ of each descriptor j in each class k, as described in the fundamentals of LAMDA (see Eq. (2)). The set of parameters $\rho_{k,j}, \sigma_{k,j}$ must be optimized, changing the bell shape by adjusting the classes of the model. These parameters are known as the premise parameters of the LAMDA structure.

Layer 2: Each node in this layer computes the $GAD_{\overline{X},k}(MAD_{k,1}, \ldots, MAD_{k,n}, \alpha)$ of each class k through the aggregation functions and the exigency parameter α. This parameter must be optimized, changing the exigency degree for the classes of the model, and therefore, the linear interpolation between the t-norm and t-conorm, which affects the behavior of the GADs.

Layer 3: In this node, the normalization of each GAD is computed, with respect to the sum of all the GAD for each class. The

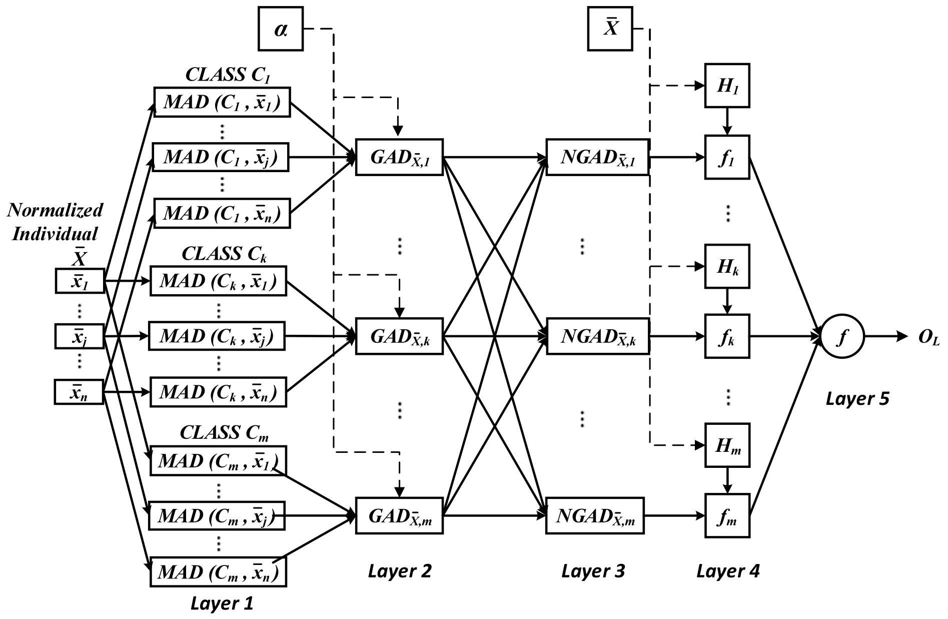


Fig. 2. Adaptive scheme for LAMDA.

normalization is performed by:

$$NGAD_{\overline{X},k}\left(GAD_{\overline{X},1},\ldots,GAD_{\overline{X},k},\ldots,GAD_{\overline{X},m}\right)=\frac{GAD_{\overline{X},k}}{\sum_{k=1}^{m}GAD_{\overline{X},k}} \quad (7)$$

Layer 4: Each node of this layer corresponds to the result of multiplying the $NGAD_{\overline{X},k}$ with a first-order T–S function $H_k(\cdot)$ for the class k that uses the descriptors of the analyzed individual, and it is defined by Eq. (8). This function has $n+1$ parameters, that is, it depends on the number of descriptors of $\overline{X}$. These values are known as consequent parameters.

$$H_k\left(\overline{X},h_{k1},\ldots,h_{kj},\ldots,h_{kn},h_k\right) = \overline{x}_1h_{k1}+\cdots+\overline{x}_jh_{kj}+\cdots+\overline{x}_nh_{kn}+h_k \quad (8)$$

and the output of layer 4 is computed by:

$$f_k\left(NGAD_{\overline{X},k},H_k\right)=NGAD_{\overline{X},k}H_k \quad (9)$$

Layer 5: This layer has only one node, which computes the sum of all the inputs, giving as a product the value O_L:

$$O_L\left(f_1,\ldots,f_k,\ldots,f_m\right)=\sum_{k=1}^{m}f_k \quad (10)$$

Using the previous expressions, the construction of an Adaptive LAMDA algorithm based on the T–S inference is proposed. This model must adjust the premise parameters that correspond to the calculation of the $MAD_{k,j}$, such as: $\rho_{k,j},\sigma_{k,j}$, the exigency parameter α, and the consequent parameters in the functions: $H_1,\ldots,H_k,\ldots,H_m$.

The number of nodes in each layer depends on the number of descriptors/features and their fuzzy sets (set by the designer). Based on the fact that all descriptors are considered to have the same number of classes "s", the total number of classes is $m=s^n$, and the number of nodes in each layer is, for layer 1: (ns)nodes, for layers 2, 3, 4: m nodes, and for layer 5: 1 node.

3.2. Hybrid learning algorithm

In the adaptive LAMDA model, each node fulfills an established function in a unidirectional manner. Some of these nodes have parameters that are adapted as a result of the learning process based on the input and output data. In this process, the method called *hybrid learning* [12,13] has been considered. It consists of a step forward and a step backward that considerably improves the learning time, preventing that solutions be trapped in local minima [54,55].

The proposed learning has been well studied in different works where adaptive networks are designed [13,14]. In the first stage, a forward pass is carried out with the least-squares estimate (LSE) method to adjust the consequent parameters, then a backward pass is performed using the gradient descent (GD) algorithm to adjust the antecedent parameters. The scheme of the hybrid learning algorithm is presented in Fig. 3, detailing the two steps for parameters update.

3.2.1. Forward pass

In the forward pass, the learning algorithm keeps fixed the antecedent parameters $\rho_{k,j},\sigma_{k,j},\alpha$ required for the calculation of $MAD_{k,j}$ and $GAD_{\overline{X},k}$ in the layers 1 and 2, respectively, and the process goes forward until the calculation of the nodes $NGAD_{\overline{X},k}$ in layer 3.

Layer 4 requires the consequent parameters for all the classes $C=\{C_1,\ldots,C_k,\ldots,C_m\}$. Here, the LSE method is used for their adjustment, considering that the function $H_k(\cdot)$ is linear in the consequent parameters. To demonstrate this, we develop Eq. (9), considering the dth individual $\overline{X}^d=\left[\overline{x}_1^d,\ldots,\overline{x}_j^d,\ldots,\overline{x}_n^d\right]$ that produces the output o_L^d:

$$o_L^d=f_1+\cdots+f_k+\cdots+f_m \quad (11)$$

Expressing (11) on the terms of $H(\cdot)$:

$$o_L^d=NGAD_{\overline{X}^d,1}H_1+\cdots+NGAD_{\overline{X}^d,k}H_k+\cdots+NGAD_{\overline{X}^d,m}H_m \quad (12)$$

Replacing (8) in (12) for the dth individual:

$$\begin{aligned}o_L^d=&\left(NGAD_{\overline{X}^d,1}\overline{x}_1^d\right)h_{11}+\cdots+\left(NGAD_{\overline{X}^d,1}\overline{x}_j^d\right)h_{1j}+\cdots\\&+\left(NGAD_{\overline{X}^d,1}\overline{x}_n^d\right)h_{1n}+\left(NGAD_{\overline{X}^d,1}\right)h_1\\&+\left(NGAD_{\overline{X}^d,k}\overline{x}_1^d\right)h_{k1}+\cdots+\left(NGAD_{\overline{X}^d,k}\overline{x}_j^d\right)h_{kj}+\cdots\\&+\left(NGAD_{\overline{X}^d,k}\overline{x}_n^d\right)h_{kn}+\left(NGAD_{\overline{X}^d,k}\right)h_k\end{aligned}$$

$$+\left(NGAD_{\overline{X}^d,m}\bar{x}_1^d\right)h_{m1}+\cdots+\left(NGAD_{\overline{X}^d,m}\bar{x}_j^d\right)h_{mj}+\cdots$$
$$+\left(NGAD_{\overline{X}^d,m}\bar{x}_n^d\right)h_{mn}+\left(NGAD_{\overline{X}^d,m}\right)h_m \tag{13}$$

The parameters $\{h_{11},\ldots,h_{1j},\ldots,h_1,h_{k1},\ldots,h_{kj},\ldots,h_k,h_{m1},\ldots,h_{mj},\ldots,h_m\}\in\mathbb{R}$ are constants, and the expression (13) shows that function $H_k(\cdot)$ is linear in the consequent, for all the *desired values* at the output $O=\left[o^1\ldots o^d\ldots o^D\right]^T$. Eq. (13) can be rewritten as:

$$O=Ah \tag{14}$$

In this case, the matrix A is invertible, and Eq. (15) can be used; otherwise, the pseudoinverse must be computed with (16), which minimize the difference $\left(\|Ah-O\|^2\right)$:

$$h=A^{-1}O \tag{15}$$

$$h=\left(A^TA\right)^{-1}A^TO \tag{16}$$

The fact that an inverse matrix must be calculated makes Eqs. (15) and (16) computationally expensive. For this reason, a sequential method is used to calculate h.

The recursive method applied to time-varying systems uses the dth row vector of matrix A, defined by a^T, and the dth element of O, defined by o^d. Then, h is iteratively computed using the covariance matrix$P(t+1)$ as follows:

$$P(k+1)=\frac{1}{\lambda}\left[P(k)-\frac{P(k)a(k+1)a^T(k+1)P(k)}{\lambda+a^T(k+1)P(k)a(k+1)}\right];$$
$$0<\lambda\le 1 \tag{17}$$

λ is the forgetting factor and is chosen close to 1 to achieve good stability [56].

Finally, $h(k+1)$ is computed by:

$$h(k+1)=h(k)+P(k+1)a(k+1)$$
$$\times\left[o(k+1)-a^T(k+1)h(k)\right];d=\{1,\ldots,D-1\} \tag{18}$$

3.2.2. Backward pass

According to our LAMDA model, if a data output set $O=\left[o^1\ldots o^d\ldots o^D\right]^T$ is available, then a supervised learning process can be carried out, propagating backward the error from layer 5 to layer 1 by the chain rule, after computing the consequent parameters $h(t+1)$, with Eq. (19). Considering that o^d is the dth data of the desired outputs O, and o_L^d the output calculated by the LAMDA model corresponding to the individual $\overline{X}^d$, the error in layer 5 is defined as:

$$E_d(k)=\frac{1}{2}\left[o^d(k)-o_L^d(k)\right]^2 \tag{19}$$

For online learning, the aim is to propagate backward the error E_d, through each layer and each node, until obtaining the derivative of the error E_d with respect to each of the adjustment terms $\theta=\left\{\rho_{k,j},\sigma_{k,j},\alpha\right\}$ required in Eqs. (2) and (4) to compute *MADs* and *GADs*, respectively.

In this way, the adjustment of θ in an instant of time $(k+1)$ by the gradient descent method is done through Eq. (20), and the updated with (21):

$$\Delta\theta(k)=-\eta\frac{\partial E_d(k)}{\partial\theta(k)} \tag{20}$$

$$\theta(k+1)=\theta(k)+\Delta\theta(k)+\beta(\theta(k)-\theta(k-1))$$
$$=\theta(k)-\eta\frac{\partial E_d}{\partial\theta}+\beta(\theta(k)-\theta(k-1)) \tag{21}$$

Where $\eta\in[0,1]$ corresponds to the learning rate, and $\beta\in[0,1]$ is the momentum term.

The learning process using the gradient descent method through the backpropagation of the error E_d from layer 5 to layer 1 is:

Layer 5:

$$\epsilon^{(5)}=\frac{\partial E_d}{\partial o_L^d}=\frac{\partial}{\partial o_L^d}\left[\frac{1}{2}\left(o^d-o_L^d\right)^2\right]=-\left(o^d-o_L^d\right) \tag{22}$$

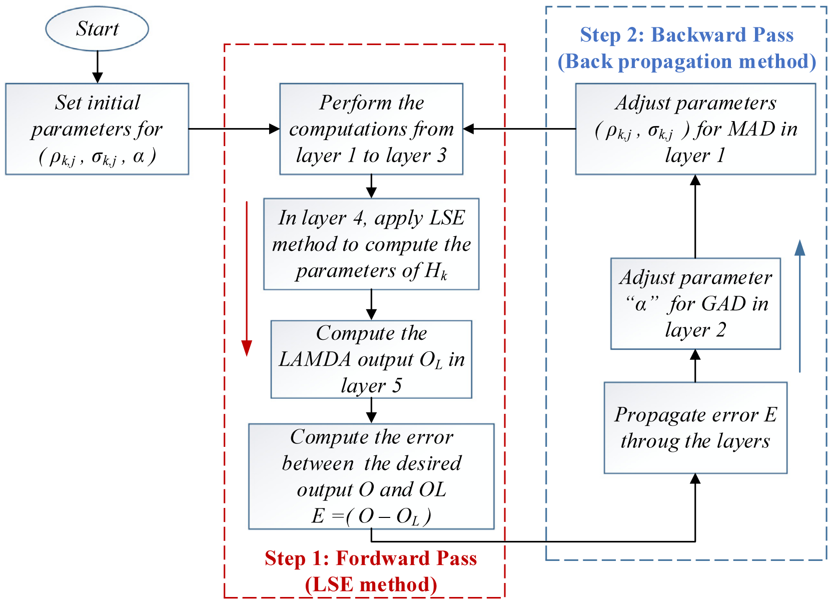
Layer 4: From (10), the derivative of $o_L^d$with respect to ∂f_k is:

$$\frac{\partial o_L^d}{\partial f_k}=\frac{\partial[f_1+\cdots f_k+\cdots f_m]}{\partial f_k}=1;\forall k=1,\ldots,m \tag{23}$$

$$\epsilon_k^{(4)}=\frac{\partial E_d}{\partial o_L^d}\frac{\partial o_L^d}{\partial f_k}=\epsilon^{(5)} \tag{24}$$

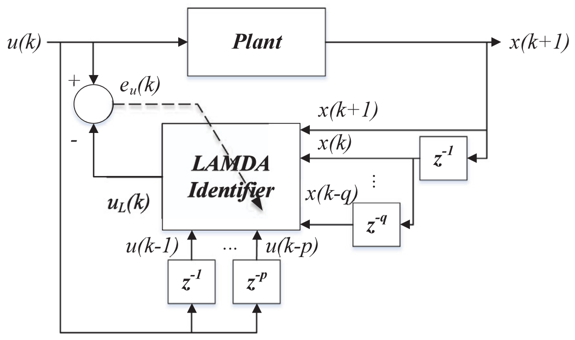
Layer 3: From (9), the derivative is:

$$\frac{\partial f_k}{\partial NGAD_{\overline{X},k}}=\frac{\partial\left[NGAD_{\overline{X},k}\times H_k\right]}{\partial NGAD_{\overline{X},k}}=H_k;\forall k=1,\ldots,m \tag{25}$$



Start
Set initial parameters for (ρ_{k,j} , σ_{k,j} , α)
Perform the computations from layer 1 to layer 3
In layer 4, apply LSE method to compute the parameters of H_k
Compute the LAMDA output O_L in layer 5
Compute the error between the desired output O and OL E =(O – O_L)
Step 1: Fordward Pass (LSE method)
Step 2: Backward Pass (Back propagation method)
Adjust parameters (ρ_{k,j} , σ_{k,j}) for MAD in layer 1
Adjust parameter "α" for GAD in layer 2
Propagate error E throug the layers

Fig. 3. Hybrid Learning Scheme.



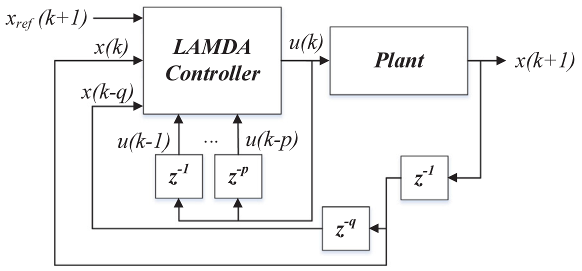


Fig. 4. Block diagram of the inverse control method: a) Learning stage, (b) Application Stage.

$$\epsilon_k^{(3)} = \frac{\partial E_d}{\partial o_L^d}\frac{\partial o_L^d}{\partial f_k}\frac{\partial f_k}{\partial NGAD_{\overline{X},k}} = \epsilon^{(5)} H_k; \forall k = 1,\ldots,m \quad (26)$$

Layer 2: In this layer, the partial derivatives of layer 3 are calculated with respect to the outputs of layer 2. Because each node$k$ of layer 3 depends on all the outputs of layer 2, as is shown in (7), the term k_2 is used to refer to the nodes of layer 2.

$$\frac{\partial NGAD_{\overline{X},k}}{\partial GAD_{\overline{X},k_2}} = \begin{cases} \dfrac{\left(\sum_{k_2=1}^{m} GAD_{\overline{X},k_2}\right) - GAD_{\overline{X},k_2}}{\left(\sum_{k_2=1}^{m} GAD_{\overline{X},k_2}\right)^2} & if: k_2 = k \\ -\dfrac{GAD_{\overline{X},k}}{\left[\sum_{k=1}^{m} GAD_{\overline{X},k_2}\right]^2} & if: k_2 \neq k \end{cases} \quad (27)$$

$$\epsilon_k^{(2)} = \frac{\partial E_d}{\partial o_L^d}\frac{\partial o_L^d}{\partial f_k}\frac{\partial f_k}{\partial NGAD_{\overline{X},k}}\sum_{k=1}^{m}\frac{\partial NGAD_{\overline{X},k}}{\partial GAD_{\overline{X},k_2}} \quad (28)$$

$$\epsilon_k^{(2)} = \epsilon^{(5)} H_k \sum_{k=1}^{m}\frac{\partial NGAD_{\overline{X},k}}{\partial GAD_{\overline{X},k_2}}; \forall k = 1,\ldots,m \quad (29)$$

Layer 1: The partial derivatives of layer 2 are computed with respect to the outputs of layer 1. Because the *GADs* are calculated recursively by Eq. (5), we use the term j_1 to refer to each of the nodes of the layer 1, in order to facilitate the mathematical expression of the derivative.

$$\frac{\partial GAD_{\overline{X},k}}{\partial MAD_{k,j}} = \alpha T\left(MAD_{k,1},\ldots,MAD_{k,j_1},\ldots,MAD_{k,n}\right) + (1-\alpha)\left(1 - S\left(MAD_{k,1},\ldots,MAD_{k,j_1},\ldots,MAD_{k,n}\right)\right); \forall j_1 \neq j \quad (30)$$

In (30), the derivative of $GAD_{\overline{X},k}$ respect to $MAD_{k,j}$ is equal to the calculation of the GAD without considering this term. Now, the propagated error in layer 1 is:

$$\epsilon_k^{(1)} = \epsilon_k^{(2)}\frac{\partial GAD_{\overline{X},k}}{\partial MAD_{k,j}} \quad (31)$$

The parameters $\rho_{k,j}, \sigma_{k,j}$ are adjusted in each class k and each descriptor j with Eqs. (32) and (33), respectively, and α is adjusted for all the model with (34).

$$\frac{\partial MAD_{k,j}}{\partial \rho_{k,j}} = \frac{\left(\overline{x}_j - \rho_{k,j}\right)}{\left(\sigma_{k,j}\right)^2}\partial MAD_{k,j} \Longrightarrow \frac{\partial E_d}{\partial \rho_{k,j}} = \epsilon_k^{(1)}\frac{\partial MAD_{k,j}}{\partial \rho_{k,j}} \quad (32)$$

$$\frac{\partial MAD_{k,j}}{\partial \sigma_{k,j}} = \frac{\left(\overline{x}_j - \rho_{k,j}\right)^2}{\left(\sigma_{k,j}\right)^3}\partial MAD_{k,j} \Longrightarrow \frac{\partial E_d}{\partial \sigma_{k,j}} = \epsilon_k^{(1)}\frac{\partial MAD_{k,j}}{\partial \sigma_{k,j}} \quad (33)$$

$$\frac{\partial GAD_{\overline{X},k}}{\partial \alpha} = \left[T\left(MAD_{k,1},\ldots,MAD_{k,j},\ldots,MAD_{k,n}\right) - S\left(MAD_{k,1},\ldots,MAD_{k,j},\ldots,MAD_{k,n}\right)\right] \Longrightarrow \frac{\partial E_d}{\partial \alpha} = \sum_{k=1}^{m}\epsilon_k^{(2)}\frac{\partial GAD_{\overline{X},k}}{\partial \alpha} \quad (34)$$

Finally, the terms are updated as:

$$\rho_{k,j}(k+1) = \rho_{k,j}(k) + \eta\left(-\frac{\partial E_d}{\partial \rho_{k,j}}\right) + \beta\left(\rho_{k,j}(k) - \rho_{k,j}(k-1)\right) \quad (35)$$

$$\sigma_{k,j}(k+1) = \sigma_{k,j}(k) + \eta\left(-\frac{\partial E_d}{\partial \sigma_{k,j}}\right) + \beta\left(\sigma_{k,j}(k) - \sigma_{k,j}(k-1)\right) \quad (36)$$

$$\alpha(k+1) = \alpha(k) + \eta\left(-\frac{\partial E_d}{\partial \alpha}\right) + \beta\left(\alpha(k) - \alpha(k-1)\right) \quad (37)$$

The proposed procedure for online learning is performed at every sample time.

4. Design of the control with Adaptive LAMDA

As background, we have been used the AIC strategy [9]. This method requires an offline learning by using random values as training output, but also, the plant response to these values as training input, as is shown in Fig. 4a. Here, we propose to use the Adaptive LAMDA as an identifier, applying a random input $u(k)$ to the plant and taking the output $x(k+1)$, its previous values $[x(k); \ldots; x(k-q)]$, and the delayed values $[u(k-1); \ldots; u(k-p)]$ as descriptors. The reason for this delayed network input is to allow in the application stage the desired plant output and the current plant feedback as the network input. With training, the internal parameters of the LAMDA model are updated to minimize the error $e_u(k)$ through the process detailed in Section 3.2. After the training, the application stage is implemented with the trained LAMDA model, as is shown in Fig. 4b. This model takes as inputs the desired reference $x_{ref}(k+1)$, the states of the plant $[x(k); \ldots; x(k-q)]$, and the delayed values $[u(k-1),\ldots,u(k-p)]$. The main idea of this method is to estimate the inverse plant model based on past and current plant outputs and inputs, to obtain the feedback control. The selection of p and q depends on an estimation of the order of the plant.

4.1. Proposed feedback control with Adaptive LAMDA

To obtain online control with the Adaptive LAMDA, the feedback control scheme presented in Fig. 5 is proposed. Once the model has been trained, as shown in Fig. 4a, initial parameters are set in the identifier. In the application, the identifier is trained online in a supervised manner with the hybrid learning of Fig. 3.

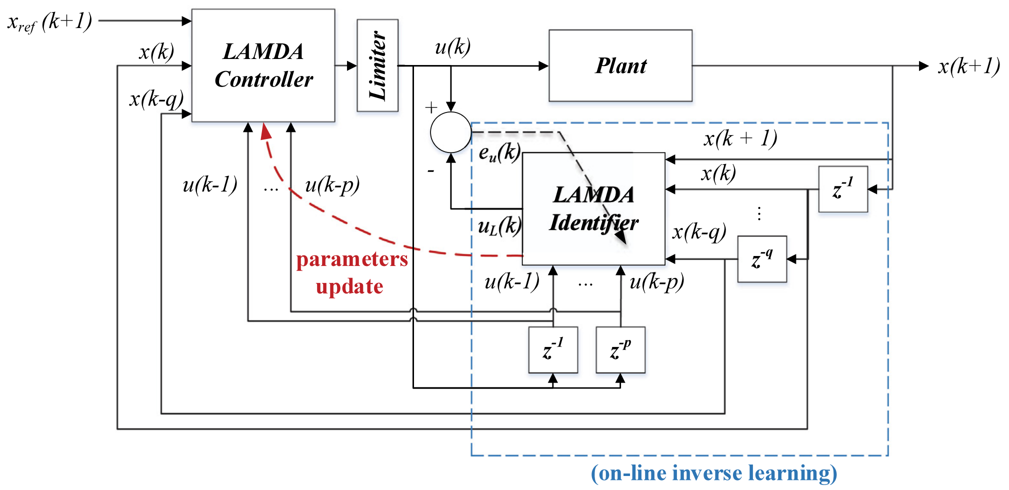


Fig. 5. Block diagram of the online inverse learning control with Adaptive LAMDA.

A duplicate LAMDA is used as controller, considering now the desired reference $x_{ref}(k+1)$, updating its internal parameters in each sample time based on the learning performed by the identifier.

Due to the online learning feature, the proposed scheme is able to bring the system output to the desired reference even in the presence of disturbances, or when the dynamic of the plant is variable. To validate the proposed controller, different control tasks have been simulated: (1) Regulation of mixing tank with variable dead-time (slow variable dynamic), (2) Regulation of an HVAC system (slow multivariable nonlinear dynamic), and (3) Trajectory tracking of a mobile robot (fast multivariable nonlinear dynamic).

4.2. Convergence of the learning algorithm

For online learning, we use the proposed hybrid algorithm presented in Section 3. In each iteration of the closed-loop of Fig. 5, the antecedent and consequent parameters are adjusted with the aim that the LAMDA output converges to a desired value. When GD and RLS are used to update the model parameters, then they are rapidly and effectively modified in order to minimize the error. From Eq. (19), we define the error $E_d(k)$ as:

$$E_d(k)=\frac{1}{2}\left[o^d(k)-o_L^d(k)\right]^2=\frac{1}{2}e(k)^2 \tag{38}$$

To demonstrate the convergence of the algorithm, we use Lyapunov's theory. The selected Lyapunov function is:

$$V(k)=E_d(k) \tag{39}$$

Then, the change of $V(k)$ is computed as:

$$\begin{aligned}\Delta V(k)&=V(k+1)-V(k)=E_d(k+1)-E_d(k)\\&=\frac{1}{2}\left(e(k+1)^2-e(k)^2\right)\\ \Delta V(k)&=\frac{1}{2}\left(e(k+1)-e(k)\right)\left(e(k+1)+e(k)\right)\\ \Delta V(k)&=\frac{1}{2}\Delta e(k)\left(\Delta e(k)+2e(k)\right)\end{aligned} \tag{40}$$

Grouping all the terms of the antecedent in vector form $\Upsilon(k)$ for the centers and $\Psi(k)$ for the standard deviation of the classes, we have:

$$\begin{aligned}\Upsilon(k)=\big[&\rho_{1,1}(k),\ldots,\rho_{1,j}(k),\ldots,\rho_{2,1}(k),\ldots,\rho_{2,j}(k),\ldots,\\&\rho_{k,1}(k),\ldots,\rho_{k,j}(k),\ldots,\rho_{m,n}(k)\big]^T\end{aligned} \tag{41}$$

$$\begin{aligned}\Psi(k)=\big[&\sigma_{1,1}(k),\ldots,\sigma_{1,j}(k),\ldots,\sigma_{2,1}(k),\ldots,\sigma_{2,j}(k),\ldots,\\&\sigma_{k,1}(k),\ldots,\sigma_{k,j}(k),\ldots,\sigma_{m,n}(k)\big]^T\end{aligned} \tag{42}$$

and the consequent parameters in the matrix $h(k)$, from Eq. (18):

$$h(k)=\begin{bmatrix}h_{11}&\ldots&h_{1j}&\ldots&h_{1n}&h_1\\h_{k1}&\ldots&h_{kj}&\ldots&h_{kn}&h_k\\h_{m1}&\ldots&h_{mj}&\ldots&h_{mn}&h_m\end{bmatrix} \tag{43}$$

The change in error $\Delta e(k)$can be approximated by [54]:

$$\begin{aligned}\Delta e(k)=&\left(\frac{\partial e(k)}{\Upsilon(k)}\right)^T\Delta\Upsilon(k)+\left(\frac{\partial e(k)}{\partial\Psi(k)}\right)^T\Delta\Psi(k)\\&+\left(\frac{\partial e(k)}{\partial\alpha(k)}\right)\Delta\alpha(k)+\operatorname{tr}\left(\left(\frac{\partial e(k)}{\partial h(k)}\right)\Delta h(k)\right)\end{aligned} \tag{44}$$

Updating the centers by the gradient descent method:

$$\Upsilon(k+1)=\Upsilon(k)+\eta\left(-\frac{\partial E_d(k)}{\partial\Upsilon(k)}\right)+\beta\left(\Upsilon(k)-\Upsilon(k-1)\right) \tag{45}$$

$$\frac{\partial E_d(k)}{\partial\Upsilon(k)}=\frac{\partial E_d(k)}{\partial e(k)}\frac{\partial e(k)}{\partial o_L^d(k)}\frac{o_L^d(k)}{\partial\Upsilon(k)}=-e(k)\frac{\partial o_L^d(k)}{\partial\Upsilon(k)} \tag{46}$$

The change of $\Upsilon(k)$, replacing (46) in (45), is:

$$\Delta\Upsilon(k)=\eta e(k)\frac{\partial o_L^d(k)}{\partial\Upsilon(k)}+\beta\Delta\Upsilon(k-1) \tag{47}$$

Applying the same procedure of (45)–(47) for $\Psi(k)$ and $\alpha(k)$, we have:

$$\Delta\Psi(k)=\eta e(k)\frac{\partial o_L^d(k)}{\partial\Psi(k)}+\beta\Delta\Psi(k-1) \tag{48}$$

$$\Delta\alpha(k)=\eta e(k)\frac{\partial o_L^d(k)}{\partial\alpha(k)}+\beta\Delta\alpha(k-1) \tag{49}$$

Now, rewriting Eq. (18) to compute the consequent parameters $h(k+1)$:

$$h(k+1)=h(k)+P(k+1)\,a(k+1)\,e_r(k) \tag{50}$$

with:

$$e_r(k) = o(k+1) - a^T(k+1)\,h(k) \tag{51}$$

From (50), the change of $h(k)$ is computed by:

$$\Delta h(k) = P(k+1)\,a(k+1)\,e_r(k) \tag{52}$$

Thus, the estimated output of LAMDA for the d-th sample from Eq. (18) is:

$$o_L^d(k+1) = a^T(k+1)\,h(k) \tag{53}$$

From Eqs. (38) and (53), we have the derivative:

$$\frac{\partial e(k)}{\partial h(k)} = \frac{\partial e(k)}{\partial o_L^d(k)}\frac{\partial o_L^d(k)}{\partial h(k)} = -a^T(k+1) \tag{54}$$

Replacing (16), (47)–(49) and (54) in (44), we can compute $\Delta e(k)$ as:

$$\begin{aligned}\Delta e(k) = &\left(-\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\left(\eta e(k)\frac{\partial o_L^d(k)}{\partial \Upsilon(k)} + \beta\Delta\Upsilon(k-1)\right)\\ &+\left(-\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\left(\eta e(k)\frac{\partial o_L^d(k)}{\partial \Psi(k)} + \beta\Delta\Psi(k-1)\right)\\ &+\left(-\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\left(\eta e(k)\frac{\partial o_L^d(k)}{\partial \alpha(k)} + \beta\Delta\alpha(k-1)\right)\\ &-a^T(k+1)\,P(k+1)\,a(k+1)\,e_r(k)\end{aligned} \tag{55}$$

$$\begin{aligned}\Delta e(k) = &-\eta e(k)\left(\left\|\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right\|_2\right)^2 - \left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\beta\Delta\Upsilon(k-1)\\ &-\eta e(k)\left(\left\|\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right\|_2\right)^2 - \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\beta\Delta\Psi(k-1)\\ &-\eta e(k)\left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)^2 - \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\beta\Delta\alpha(k-1)\\ &-a^T(k+1)\,P(k+1)\,a(k+1)\,e_r(k)\end{aligned} \tag{56}$$

Now, the following norms are replaced with the terms $N_\Upsilon, N_\Psi, N_\alpha$:

$$N_\Upsilon = \left(\left\|\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right\|_2\right)^2, N_\Psi = \left(\left\|\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right\|_2\right)^2, N_\alpha = \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)^2 \tag{57}$$

Replacing (57) in (56):

$$\begin{aligned}\Delta e(k) = &-\eta e(k)N_\Upsilon - \left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\beta\Delta\Upsilon(k-1) - \eta e(k)N_\Psi\\ &-\left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\beta\Delta\Psi(k-1)\\ &-\eta e(k)N_\alpha - \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\beta\Delta\alpha(k-1)\\ &-a^T(k+1)\,P(k+1)\,a(k+1)\,e_r(k)\end{aligned} \tag{58}$$

$$\begin{aligned}\Delta e(k) = -e(k)\Bigg[&\eta N_\Upsilon + \left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\frac{\beta}{e(k)}\Delta\Upsilon(k-1)\\ &+\eta N_\Psi + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\frac{\beta}{e(k)}\Delta\Psi(k-1)\\ &+\eta N_\alpha + \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\frac{\beta}{e(k)}\Delta\alpha(k-1)\\ &+a^T(k+1)\,P(k+1)\,a(k+1)\frac{e_r(k)}{e(k)}\Bigg]\end{aligned} \tag{59}$$

Replacing (59) in the equation of $\Delta V(k)$ presented in (40):

$$\begin{aligned}\Delta V(k) = \frac{1}{2}e^2(k)\Bigg[&\eta(N_\Upsilon + N_\Psi + N_\alpha)\\ &+\frac{\beta}{e(k)}\left(\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\Delta\Upsilon(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\Delta\Psi(k-1)\right.\\ &\left.+\left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\Delta\alpha(k-1)\right)\\ &+a^T(k+1)\,P(k+1)\,a(k+1)\frac{e_r(k)}{e(k)}\Bigg]\\ \times\Bigg[&\eta(N_\Upsilon + N_\Psi + N_\alpha)\\ &+\frac{\beta}{e(k)}\left(\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\Delta\Upsilon(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\Delta\Psi(k-1)\right.\\ &\left.+\left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\Delta\alpha(k-1)\right)\\ &+a^T(k+1)\,P(k+1)\,a(k+1)\frac{e_r(k)}{e(k)} - 2\Bigg]\end{aligned} \tag{60}$$

From (60), the following equalities are considered:

$$A(k) = \eta(N_\Upsilon + N_\Psi + N_\alpha) \tag{61}$$

$$\begin{aligned}B(k) = \frac{\beta}{e(k)}\Bigg(&\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\Delta\Upsilon(k-1)\\ &+\left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\Delta\Psi(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\Delta\alpha(k-1)\Bigg)\end{aligned} \tag{62}$$

$$C(k) = a^T(k+1)\,P(k+1)\,a(k+1)\frac{e_r(k)}{e(k)} \tag{63}$$

Replacing (61)–(63) in (60), to guarantee the convergence and stability, $\Delta V(k)$must meet the condition:

$$\begin{aligned}&\Delta V(k) < 0\\ &\Rightarrow \frac{1}{2}e^2(k)\,[A(k)+B(k)+C(k)]\,[A(k)+B(k)+C(k)-2] < 0\\ &\Rightarrow 0 < A(k)+B(k)+C(k) < 2\end{aligned} \tag{64}$$

A is always positive, while the signs of $B(k)$ and $C(k)$ must be evaluated to meet the condition presented in (64), if $B(k) > 0$:

$$\begin{aligned}\frac{\beta}{e(k)}\Bigg(&\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T\Delta\Upsilon(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T\Delta\Psi(k-1)\\ &+\left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right)\Delta\alpha(k-1)\Bigg) > 0\end{aligned} \tag{65}$$

Because $P(k+1)$ is Hermitanian semidefinite positive [54] , then:

$$\frac{e_r(k)}{e(k)} > 0 \Longrightarrow C(k) > 0 \tag{66}$$

For stability, it is sufficient to consider the same weights for all the terms in (64):

$$0 < A(k) < \frac{2}{3};\ 0 < B(k) < \frac{2}{3};\ 0 < C(k) < \frac{2}{3} \tag{67}$$

Thus, from (61) and (65)–(67), we obtain (68)–(70) given in Box I.

Considering the property: $XYX^T = tr(YX^TX)$, and applying it in (70):

$$0 < a^T(k+1)\,P(k+1)\,a(k+1)$$

$$0 < \eta < \frac{2}{3\left(N_{\Upsilon} + N_{\Psi} + N_{\alpha}\right)} \tag{68}$$

$$0 < \beta < \frac{2e(k)}{3\left(\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T \Delta\Upsilon(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T \Delta\Psi(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right) \Delta\alpha(k-1)\right)} \tag{69}$$

$$0 < a^T(k+1)\,P(k+1)\,a(k+1)\,\frac{e_r(k)}{e(k)} < \frac{2}{3} \tag{70}$$

Box I.

$$= \operatorname{tr}\left(P(k+1)\,a(k+1)\,a^T(k+1)\right) < \frac{2e(k)}{3e_r(k)} \tag{71}$$

Applying the property $W, Z \in H_0^+(n)$, then $0 \leq \operatorname{tr} WZ \leq \operatorname{tr} W \operatorname{tr} Z$, in (71), we have:

$$0 < \operatorname{tr}(P(k+1))\operatorname{tr}\left(a(k+1)\,a^T(k+1)\right) < \frac{2e(k)}{3e_r(k)}$$

$$0 < \operatorname{tr}(P(k+1))\left(\|a(k+1)\|_F\right)^2 < \frac{2e(k)}{3e_r(k)}$$

$$0 < \operatorname{tr}(P(k+1)) < \frac{2e(k)}{3e_r(k)\left(\|a(k+1)\|_F\right)^2} \tag{72}$$

Now, if $B(k) < 0$:, we have (73) and (74) given in Box II.

From Eq. (63), if $C(k) < 0$:

$$\frac{e_r(k)}{e(k)} < 0 \tag{75}$$

$$0 < -C(k) < \frac{2}{3} \tag{76}$$

Considering the same procedure from (71)–(72), in (76) we have:

$$0 < -a^T(k+1)\,P(k+1)\,a(k+1)\,\frac{e_r(k)}{e(k)} < \frac{2}{3}$$

$$0 > -a^T(k+1)\,P(k+1)\,a(k+1) > \frac{2}{3}\frac{e(k)}{e_r(k)}$$

$$0 < a^T(k+1)\,P(k+1)\,a(k+1) < -\frac{2}{3}\frac{e(k)}{e_r(k)}$$

$$0 < \operatorname{tr}(P(k+1)) < \frac{-2e(k)}{3e_r(k)\left(\|a(k+1)\|_F\right)^2} \tag{77}$$

Based on the Lyapunov function of Eq. (39), are obtained the Eqs. (68), (69), (72), (74) and (77), which guarantee the convergence of the error $e(k) \longrightarrow 0$ in the training stage for a controllable system independent of the application, system order, number of inputs, and number of classes.

As mentioned in [6], analyzing controller stability based on online learning is a complex task that is still an open field in adaptive inverse learning schemes that we will address in a future paper. However, the following aspects can be considered for local stability:

- Bounded Input–Bounded Output (BIBO) Stability: BIBO stability is guaranteed. The normalization of the *GADs*, through the computation of the normalized $NGAD_{\overline{X},k} \leq 1$ and the introduced limiter shown in Fig. 5, ensure that the adaptive LAMDA model is bounded for all inputs.

- We assume that the learning algorithm in the LAMDA Identifier has converged because a constant change in the parameters would make it hard to analyze stability. Under this assumption, we only need to take the LAMDA controller into account. If the error of the learning converges $e(k) \longrightarrow 0$, then the LAMDA model is an identical copy of the real process, so it is guaranteed that there is a solution to the inverse model, allowing to calculate a control action $u(k)$, which satisfies $x(k+1) \cong x_{ref}(x+1)$.

4.3. Computational complexity

We proceed to analyze the computational complexity of the fuzzy Adaptive LAMDA in terms of memory usage, computation time and number of operations [57] required to compute the learning and control output at each sample time. Our program is implemented in Matlab R2020a, running on an Intel (R) Core (TM) i7-8750H @ 2.2 GHz microprocessor. The computational complexity is computed based on the number of inputs of the algorithm, these are:

- n: The number of descriptors (inputs)
- s: The number of classes in each descriptor
- m: The number of total classes of the model

4.3.1. Memory usage

In this subsection, the permanent usage of memory is counted. The number of parameters to be computed by the algorithm in the learning stage for the antecedent and the consequent is based on the number of inputs and number of classes in each

$$0 < -B(k) < \frac{2}{3} \tag{73}$$

$$0 < \beta < \frac{-2e(k)}{3\left(\left(\frac{\partial o_L^d(k)}{\partial \Upsilon(k)}\right)^T \Delta\Upsilon(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \Psi(k)}\right)^T \Delta\Psi(k-1) + \left(\frac{\partial o_L^d(k)}{\partial \alpha(k)}\right) \Delta\alpha(k-1)\right)} \tag{74}$$

Box II.

Table 1
Computation time in seconds (s) of adaptive LAMDA for learning and control.

		Number of classes "s" in each descriptor							
		2		3		4		5	
		Learning	Control	Learning	Control	Learning	Control	Learning	Control
Number of Descriptors "n"	2	1.40e−4 s	4.08e−5 s	1.69e−4 s	4.48e−5 s	4.90e−4 s	5.13e−5 s	8.82e−4 s	5.65e−5 s
	3	2.08e−4 s	5.24e−5 s	1.10e−3 s	6.43e−5 s	1.71e−3 s	1.18e−4 s	10.7e−3 s	1.09e−3 s
	4	7.56e−4 s	5.67e−5 s	6.64e−3 s	7.85e−4 s	93.5e−3 s	7.84e−3 s	1.02e1 s	46.5e−3 s
	5	1.87e−3 s	9.07e−5 s	162e−3 s	11.6e−3 s	852e−3 s	182e−3 s	NT	NT
	6	6.70e−3 s	8.74e−4 s	624e−3 s	134.9	NT	NT	NT	NT
	7	45.1e−3 s	4.70e−3 s	NT	NT	NT	NT	NT	NT
	8	343e−3 s	25.8e−3 s	NT	NT	NT	NT	NT	NT

input, n and s, respectively. As we mentioned in Section 3.1., in layer 1 there are ns nodes, and at each node, two parameters are required (ρ, σ). Additionally, the parameter α is required in layer 2, therefore the number of parameters in the antecedents $\#parameters_{\theta(k)}$ and $\#parameters_{\theta(k-1)}$ are:

$$\#parameters_{\theta(k)} = 2ns + 1 \tag{78}$$

$$\#parameters_{\theta(k-1)} = 2ns + 1 \tag{79}$$

The number of parameters of the consequent in the vector $h(k)$ is $\#parameters_{h(k)}$, and the number of parameters of the covariance matrix is $\#parameters_{P(k)}$. These values are computed based on Eqs. (13) and (17), respectively, as:

$$\#parameters_{P(k)} = [m\,(n+1)]^2 \tag{80}$$

$$\#parameters_{h(k)} = m\,(n+1) \tag{81}$$

It is assumed that each value is stored in 2 bytes of memory [57].

4.3.2. Computation time

The temporal complexity is a parameter that allows us to verify the increase of the operations performed in each iteration. Table 1 shows the computation time of the proposed adaptive LAMDA in the control and learning stage of the proposed methodology.

As observed, the computation time of the Adaptive LAMDA in the learning block increases considerably with respect to the controller block. For these tests, eight system inputs have been considered, which corresponds to a high order system, for which it would be recommended to work with two classes per descriptor, so as not to affect machine time. NT (not-tested) cells have not been evaluated due to the high computation time required, especially for learning. Generally, real systems can be approximated to first or second-order systems [58], which implies working with a maximum of 4 or 5 inputs (descriptors), each with up to 4 classes, although based on the tests carried out, 2 or 3 classes per descriptor are enough to have a good performance in the control system.

4.3.3. Number of operations

The temporal complexity depends on the type of processor and memory characteristics in which the program is executed, for this reason, it is most appropriate to evaluate the number of arithmetic operations (arithmetic complexity) used to solve a problem. Subtraction, addition, multiplication, division, squared and exponent are considered as basic operations. To obtain the controller output (LAMDA controller block of Fig. 5), several operations are performed sequentially in each layer. Particularly, Eqs. (2), (3), (7), (9) and (10) are used, and their arithmetic complexity are determined in Eqs. (82) to (86), respectively.

$$\#op_{LAYER1c} = 5ns \tag{82}$$

$$\#op_{LAYER2c} = 4mn \tag{83}$$

$$\#op_{LAYER3c} = m^2 \tag{84}$$

$$\#op_{LAYER4c} = m(2n+2) \tag{85}$$

$$\#op_{LAYER5c} = m - 1 \tag{86}$$

Where $\#op_{LAYER1c}$ is the number of operations of the controller in layer 1, and so on for the rest of layers, until layer 5 ($\#op_{LAYER5c}$).

Finally, adding and simplifying the equations from (82) to (86), we have the number of operations used to calculate the controller output:

$$\#op_{control} = m^2 + 6mn + 3m + 5ns - 1 \tag{87}$$

The hybrid learning process, as mentioned in Section 3, corresponds to the "LAMDA identifier" block shown in Fig. 5. In the backward pass is calculated the gradient descent. The number of arithmetic operations in each layer is computed considering Eqs. (19)–(37):

$$\#op_{LAYER5gd} = 2 \tag{88}$$

$$\#op_{LAYER4gd} = 0 \tag{89}$$

$$\#op_{LAYER3gd} = m \tag{90}$$

$$\#op_{LAYER2gd} = m^3 + 3m^2 - 2m \tag{91}$$

$$\#op_{LAYER1gd} = m(4n+1) \tag{92}$$

$$\#op_{update_\rho} = 4(ns+5) \tag{93}$$

$$\#op_{update_\sigma} = 4(ns+5) \tag{94}$$

$$\#op_{update_\alpha} = 3mn - 2m + 5 \tag{95}$$

Where $\#op_{LAYER5gd}$ is the number of operations of the identifier in layer 5 for the gradient descent process, and successively, until layer 1 ($\#op_{LAYER1gd}$), and $\#op_{update_\rho}$ is the number of operations to update ρ, and the same for σ ($\#op_{update_\sigma}$) and $\alpha(\#op_{update_\alpha})$.

Adding and simplifying the equations from (88) to (95), the number of arithmetic operations used for computing the gradient descent in the learning stage is:

$$\#op_{gd} = m^3 + 3m^2 - 2m + n\,(7m + 8s) + 47 \tag{96}$$

In the learning, the forward pass is calculated using RLS. In this step, matrix operations are performed, which involve the addition and multiplication of elements. Considering that the dimension of the vector $a(k+1)$ depends on the inputs, we have:

$$\Omega\,(m, n) = \dim\,(a\,(k+1)) = m\,(n+1) \tag{97}$$

The number of arithmetic operations performed by the RLS with Eqs. (17)–(18) is:

$$\#op_{P(k+1)} = 4\Omega^3 + 4\Omega^2 + \Omega + 1 \tag{98}$$

$$\#op_{h(k+1)} = 2\Omega^2 + 3\Omega \tag{99}$$

Adding and simplifying the equations from (98) to (99), the number of arithmetic operations used for RLS in the learning stage based on the number of inputs is:

$$\#op_{RLS} = 4\Omega^3 + 6\Omega^2 + 4\Omega + 1 \tag{100}$$

Table 2
Arithmetic complexity of the fuzzy control algorithms.

	Arithmetic complexity
Conventional fuzzy [57]	$4089m + 37mn + 31ns + 59n + 21$
LAMDA without learning (87)	$m^2 + 6mn + 3m + 5ns - 1$
Adaptive ANFIS [8]	$m^3\left(4n^3 + 12n^2 + 12n + 5\right) + m^2\left(6n^2 + 12n + 10\right) + 8ns + 46$
Adaptive LAMDA (87)+(101)	$m^3\left(4n^3 + 12n^2 + 12n + 5\right) + m^2\left(6n^2 + 12n + 9\right) + m\,(11n + 2) + 8ns + 48$

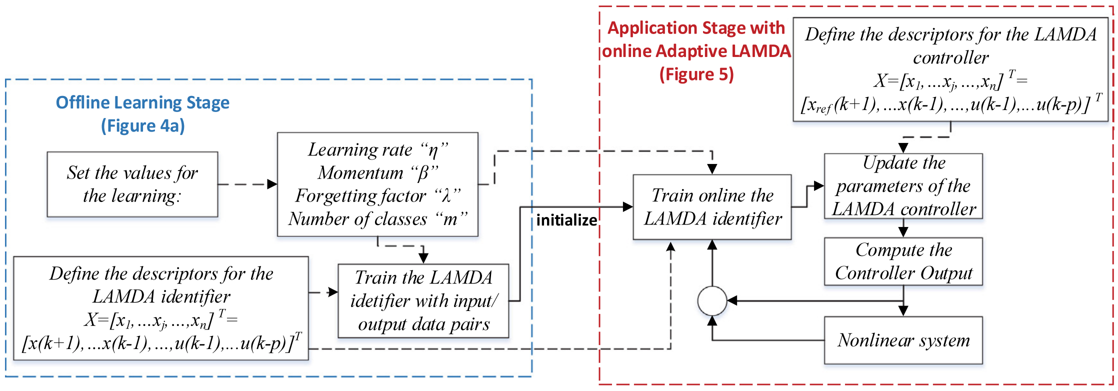

Offline Learning Stage (Figure 4a)
Set the values for the learning:
Learning rate "η" Momentum "β" Forgetting factor "λ" Number of classes "m"
Define the descriptors for the LAMDA identifier X=[x1, ...xj, ...,xn] T= [x(k+1), ...x(k-1), ...,u(k-1),...u(k-p)]T
Train the LAMDA idetifier with input/ output data pairs
initialize
Application Stage with online Adaptive LAMDA (Figure 5)
Define the descriptors for the LAMDA controller X=[x1, ...xj, ...,xn] T= [xref (k+1), ...x(k-1), ...,u(k-1),...u(k-p)] T
Train online the LAMDA identifier
Update the parameters of the LAMDA controller
Compute the Controller Output
Nonlinear system

Fig. 6. Implementation scheme of the online learning adaptive controllers.

Replacing $\Omega = m(n+1)$ in (100), adding and simplifying Eqs. (96)–(100), the total number of arithmetic operations in the learning process of the identifier is:

$$\#op_{iden} = m^3\left(4n^3 + 12n^2 + 12n + 5\right) + m^2\left(6n^2 + 12n + 9\right) + L\,(11n + 2) + 8ns + 48 \qquad (101)$$

Considering Eq. (87), the complexity order of the controller is quadratic in the total number of classes, while for the identifier, based on Eq. (101), is cubic, considering the term with the highest exponent. It is concluded then that the controller has a low arithmetic complexity that strongly depends on the number of classes established in each descriptor. On the other hand, in the case of the arithmetic complexity of the identifier, the growth rate of the operations in the algorithm increases rapidly, depending on the number of descriptors and the number of classes in a similar proportion, being this term the one that should be set to a low value to avoid carrying out a large number of operations.

The computational complexity of our algorithm, compared to proposals such as conventional Fuzzy (number of discretization of output universe of discourse suggested $(M_{OD}) = 32$ [57]), LAMDA without learning [49], and ANFIS, are presented in Table 2.

It is evident that the adaptive proposals are the most complex in computational terms, which is logical due to the learning algorithm that they incorporate, these being of cubic order with respect to the number of classes. Our proposal is quite similar to ANFIS when analyzing the number of operations that these require for the learning and the control output computation, the difference lies in the linear term m, so it cannot be considered a relevant difference. As it has been observed in the results of Table 1, when working with a low number of classes in each descriptor, the algorithm is more efficient and requires less computation time. Therefore, it can be applied in systems with 8 descriptors at a speed of 0.4 s in the worst case, which shows the viability in the use of our controller.

Finally, to summarize in this section the design criteria of our proposal, the designer must take into account the following points:

- Choose the number of inputs (n descriptors) of the object X for the LAMDA Identifier and the LAMDA controller.

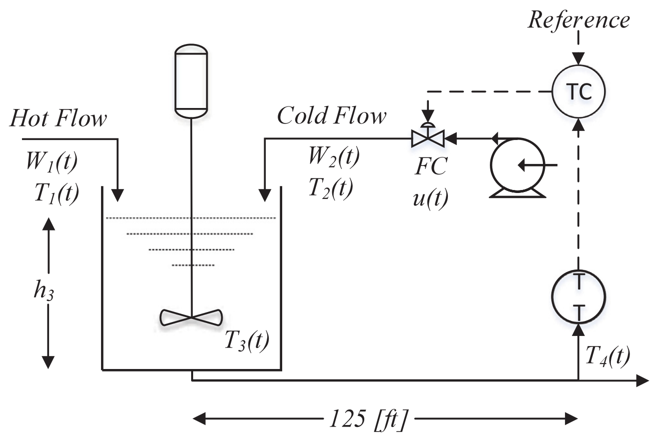


Fig. 7. Studied Process (Mixing Tank).

- Define the number of classes in each descriptor, generally 2 or 3, in order to obtain an efficient algorithm, capable of using a low machine time to solve learning and control operations.
- Carry out an offline training stage defining the learning parameters η, β, and λ with the scheme shown in Fig. 4a, which is trained using the proposed hybrid learning.
- With the initial values of the model obtained offline, and maintaining the previously defined learning parameters, implement the scheme of Fig. 5 in the system to be controlled.

In the next section, we present several examples to show the design and implementation of the proposed scheme

5. Tests and results

To validate the proposed controller, we address three tasks, each with different interesting properties. In the experiments, we demonstrate that our control strategy is experimentally stable and can be applied in systems with different dynamics. The results of these experiments are analyzed and compared with

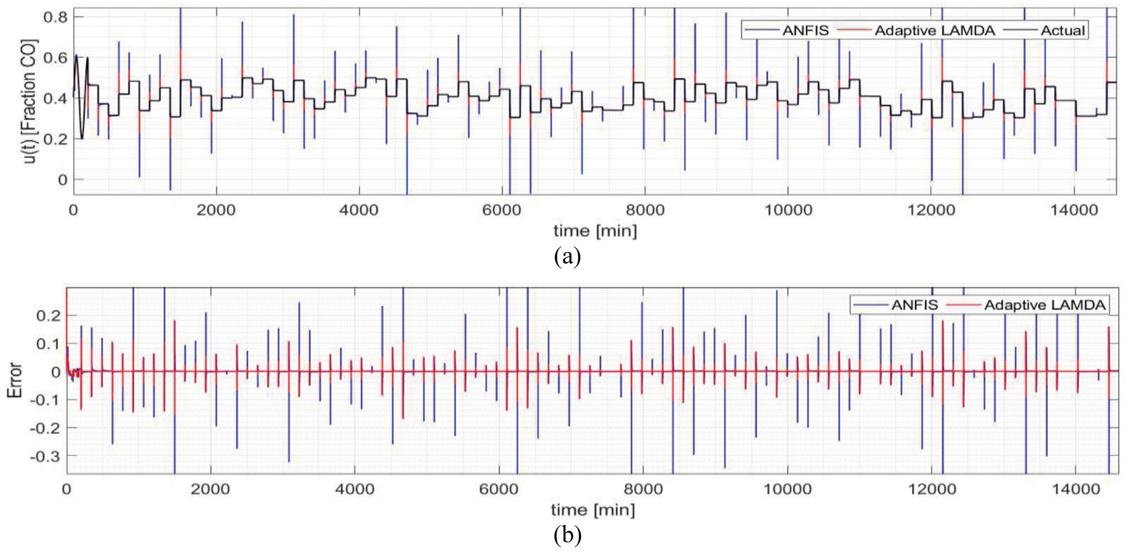


Fig. 8. Comparison of learning algorithms a) adjust in the training stage with actual output data (b) error.

other intelligent controllers that do not require the exact model of the plant to be designed, such as Fuzzy-PI (rule-based) and the recent LAMDA-PI (class-based [48,49]). These two methods are static (non-adaptive) and are designed and calibrated based on the designer's expertise, which is generally a complex and time-consuming process. The aforementioned process is not required by our adaptive method, which is the main advantage, especially when the system has unknown or variable dynamics. Additionally, the comparison is made with the online inverse learning control with ANFIS [13]. Comparative analysis allows identifying the advantages of our proposed method in the different systems. In the case of the adaptive schemes of LAMDA and ANFIS, the procedure for the validation is shown in Fig. 6, which summarizes the training and application stages of the controllers based on the schemes of Figs. 4 and 5. The black dashed lines represent the learning and controller configuration parameters, and the definition of the inputs. The solid lines represent the sequential operation of our proposal in the tests.

5.1. Case study 1: Mixing tank with variable dead time

The process consists of the mixing of two fluids in a tank with a constant liquid volume (Fig. 7). The system has a hot stream $W_1(t)$ that mixes with a cold stream $W_2(t)$, controlled with the valve position FC $(u(t))$. The resulting mixture requires to remain at a desired temperature $T_4 = 150[°F]$. The temperature transmitter is installed at a distance of $125[ft]$ from the tank outlet, and has been calibrated to operate in a range of 100 to $200[°F]$. The distance between the tank outlet and the location of the temperature transmitter generates a dead time in the measurement. The equations of the complete nonlinear model and the operating conditions of this process are presented by Camacho and Smith [59].

The dead time t_0 changes as:

$$t_0\,(t) = \frac{LA\rho}{W_1\,(t) + W_2\,(t)} \tag{102}$$

Where, $W_1\,(t)$ is the mass flow of hot stream (lb/min), $W_2\,(t)$ is the mass flow of cold stream (lb/min), $\rho = 62.4lb/ft^3$ is the density of the mixing tank contents, $A = 0.2006ft^2$ is the pipe cross-section, and $L = 125ft$ is the pipe length. This system has been used to test different types of controllers due to the variable dead time. In general, the model can be approximated to a FOPDT [59]:

$$\frac{x(s)}{u(s)} = \frac{Ke^{-tos}}{\tau s + 1} \tag{103}$$

The dead time can be modeled using a first-order Taylor series approximation as [59]:

$$e^{-tos} \cong \frac{1}{t_0 s + 1} \tag{104}$$

Substituting (104) into (103), we have:

$$\frac{x(s)}{u(s)} \cong \frac{K}{(\tau s + 1)\,(t_0 s + 1)} \tag{105}$$

Resulting in a second-order system, in a discrete-time of the form:

$$\frac{x(z)}{u(z)} \cong \frac{az + b}{cz^2 + dz + f} \tag{106}$$

where a, b, cd and f are functions dependent on K, τ, t_0. The discrete-time model is variable and dependent on these parameters. Developing (106), according to the time k, we obtain:

$$u\,(k) = (1/a)\,[cx\,(k+1) + dx\,(k) + fx\,(k-1) - bu\,(k-1)] \tag{107}$$

It is important to clarify that if the plant model is completely unknown, the number of previous states (p and q) of both x and u could be taken experimentally, until obtaining an adequate adjustment in the training stage.

For the initial training process, the algorithm parameters have been set with the following values $\eta = 0.00025$, $\beta = 0.001$, $\lambda = 1$, and a sampling period $T_s = 0.4$ min. The inputs are $[x\,(k+1)\,; x\,(k)\,; x\,(k-1)\,; u(k-1)]$, each with two classes. A random input is generated for the plant that consists of a sinusoidal signal from 0 to 200 min, and 100different random step values of 143.6 min duration. Fig. 8 presents a comparison between the two techniques that require a training stage (ANFIS and Adaptive LAMDA). It is observed that LAMDA presents less error with better adjust to the actual data $u(t)$ applied to the system.

Once LAMDA has been trained, the proposed controller is tested in the plant under disturbances produced by varying hot

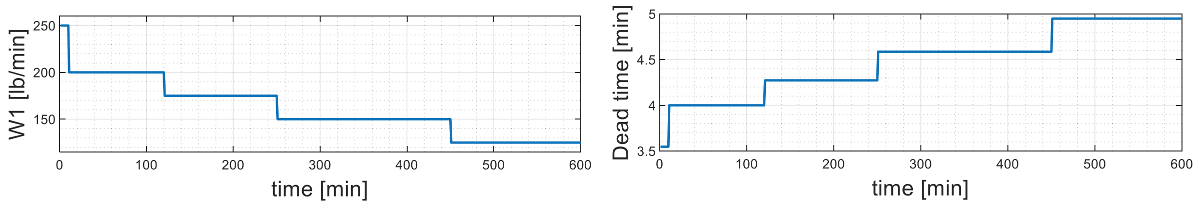


Fig. 9. Change of: (a) W_1 and (b) dead time.

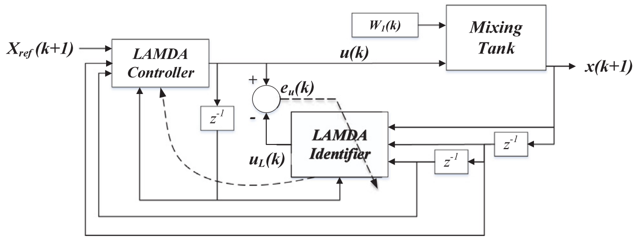


Fig. 10. Adaptive control structure for the Mixing Tank.

Table 3
IAE of the controllers applied to the mixing tank process.

Index	*Fuzzy PI*	*LAMDA PI*	*ANFIS*	*Adaptive LAMDA*
IAE	5.719	5.185	34.82	2.809
Δ	68.25%	59.44%	170.1%	–

stream $W_1(t)$. As is shown in Eq. (102), the dead time changes depending on W_1. The changes of the hot stream are shown in Fig. 9a, and the dead time variations are presented in Fig. 9b. The variable dead time causes the dynamics of the system also changes, as is presented in Eq. (106), so it is appropriate to use an adaptive method for the control.

The complete control scheme for the mixing tank with variable dynamic, is presented in Fig. 10, which shows the online identification block and the control block. The LAMDA identifier inputs based on Eq. (107) are $[x(k+1); x(k); x(k-1); u(k-1)]$, and in the LAMDA controller the inputs are $[x_{ref}(k); x(k); x(k-1); u(k-1)]$.

The tests are carried out for the controllers Fuzzy-PI, LAMDA-PI, ANFIS and the Adaptive LAMDA. Fig. 11 shows qualitatively the effectiveness of our proposal. Here, Adaptive LAMDA convergence to the reference is faster when W_1 changes abruptly, taking into account that the controller design has not required the plant model or a calibration stage, as in the case of the Fuzzy-PI and LAMDA-PI controllers. It has been observed that our learning algorithm with two classes per descriptor has presented very good results, with the advantage that the computational time is less with respect to the use of more classes. Although techniques without learning respond well to the control of this system, they have an oscillatory response with higher overshoot, especially in the case of the disturbance at time 450 min, which causes the system dead time to change abruptly (see Fig. 11a). At this point, our proposal corrects it in less time without requiring additional calibration, since it adapts to these changes automatically with the LAMDA identifier block that learns online. Disturbances at time 10 min, 120 min and 250 min are quickly corrected with minimal overshoot by our method, with excellent performance. Additionally, it can be observed that the non-learning methods (Fuzzy-PI and LAMDA-PI) degrade their response considerably as the plant changes, which is an important aspect of the performance of the system. Adaptive LAMDA does not degrade, and its control action is smoother than the other controllers, which is an important advantage since in the real system the actuator is not overstressed (see Fig. 11b). In the case of ANFIS, it is observed that it maintains an error in a steady-state, that is, the algorithm is not able to reach the reference. One solution would be to place an additional integration stage to correct it, which would increase the computational time and the complexity of the controller design.

The effectiveness of the obtained results is quantitatively evaluated by the computation of the Integral Absolute Error (IAE), an index used to determine the performance of the controllers. The IAE reflexes the cumulative error, i.e., how far the response is with respect to the applied reference. Therefore, the controller with the minimum index is the best in terms of performance. Additionally, the percentage change Δ from the best IAE value is computed to observe the improvement in performance terms (see Table 3).

The index with the lowest value is Adaptive LAMDA, because it reaches the reference quickly and with lower overshoot. In the presence of disturbances, it can be seen that the adaptive proposal is better at around 60%, with respect to Fuzzy-PI and LAMDA-PI, and in 170% with respect to ANFIS (the most similar approach) since this method is not able to reach the reference. These percentages show the potential of our learning algorithm in these types of systems.

5.2. Case study 2: Regulation of an HVAC system

HVAC systems are complex structures consisting of chillers, heat pumps, heating/cooling coils, boilers, air handling, thermal storage and liquid/air distribution units. It is a MIMO (Multiple-Input Multiple-Output) system with many variables whose modeling and dynamic study is complex due to its nonlinear characteristics [60]. Neuro-fuzzy systems are widely used in complex processes for modeling and control, however, its application in the HVAC systems is very limited [61]. The adaptive approach is simulated in the HVAC system presented by Arguello-Serrano and Velez-Reyes [62]. The main control objective in this simulation is to solve a regulation problem, analyzing and validating the proposed controller to abrupt disturbances in the thermal space variables (Zone 3 in Fig. 12), these are: Temperature ($T_3[°F]$) and Humidity Ratio (W_3 $[lb/lb]$).

The system operation is described as follows, outdoor air flows into the system mixing 25% of it with 75% of the returning air, expelling the rest. The mixed air passes through a filter to the heat exchanger, where it is conditioned to the set point. The conditioned air is propelled to the thermal zone with a fan. The system requires to control the variables T_3 and W_3, simultaneously, based on thermal loads, by varying the fan speed, u_1, to regulate the airflow rate and the cold-water pumping rate, u_2, from the chiller to the heat exchanger [49]. The differential equations of energy and mass balances known from the conventional mathematical model of HVAC systems required for the simulations are:

$$\dot{T}_3 = \frac{f}{V_s}(T_2 - T_3) - \frac{h_{fg}}{C_p V_s}(W_s - W_3) + \frac{1}{0.25 C_p V_s}(Q_0 - h_{fg} M_0) \quad (108)$$

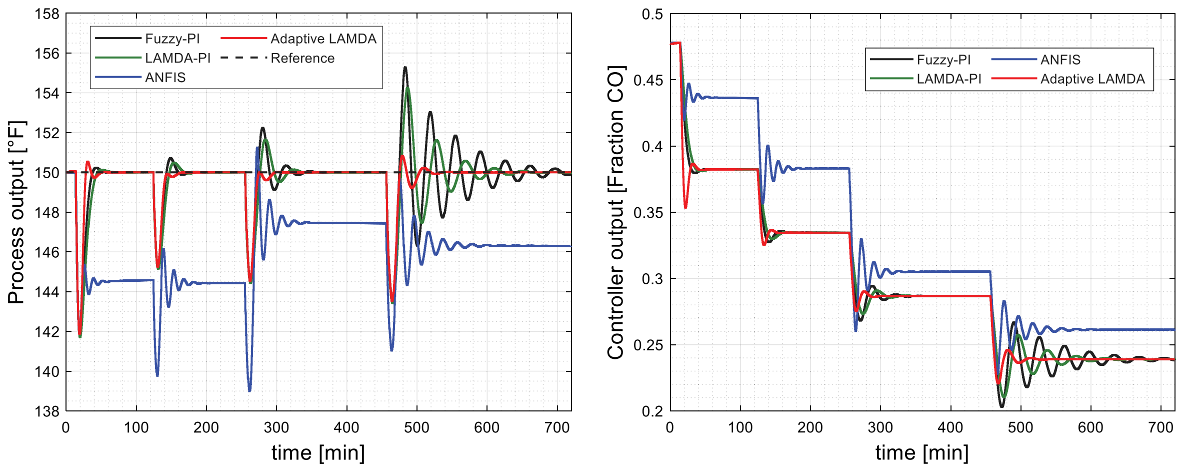


Fig. 11. (a) Comparative response of the system (temperature) (b) Applied control actions.

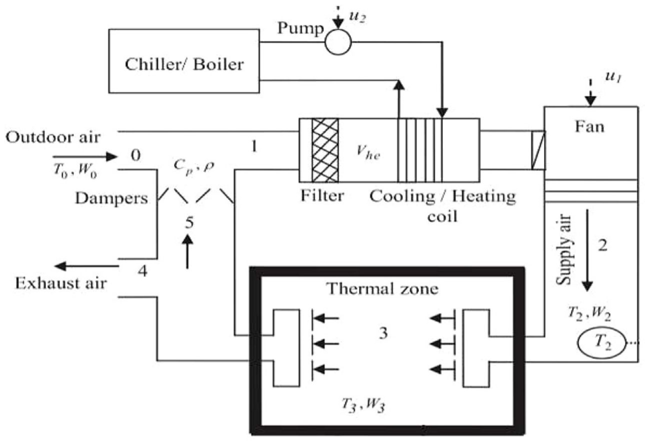


Fig. 12. Block diagram of a simple HVAC system [62].

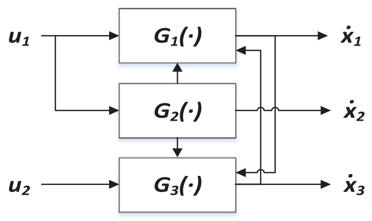


Fig. 13. Block diagram of HVAC model (MIMO system).

$$\dot{W}_3 = \frac{f}{V_s}(W_s - W_3) + \frac{M_0}{\rho V_s} \tag{109}$$

$$\dot{T}_2 = \frac{f}{V_{he}}(T_3 - T_2) - \frac{0.25f}{V_{he}}(T_0 - T_3) - \frac{f h_w}{C_p V_{he}}(0.25W_0 + 0.75W_3 - W_s) - 6000\frac{gpm}{\rho C_p V_{he}} \tag{110}$$

W_0 is the humidity ratio of outdoor air, h_w is the enthalpy of liquid water, h_{fg} is the enthalpy of water vapor, V_{he} is the volume of the heat exchanger, C_p is the specific heat of air, W_s is the humidity ratio of supply air, W_3 is the humidity ratio of Zone 3, T_0 is the temperature of outdoor air, M_0 is the moisture load, Q_0 is the sensible heat load, T_2 is the temperature of supply air, T_3 is the temperature of Zone 3, V_s is the volume of Zone 3, ρ is the air mass density, f is the volumetric flow rate of air (ft^3/min), and gpm is the flow rate of chilled water (gal/min).

To represent the system in state space notation for the design of the control system, let $u_1 = f$, $u_2 = gpm$, $x_1 = T_3$, $x_2 = W_3$, $x_3 = T_2$, $y_1 = T_3$, $y_2 = W_3$. The following parameters are defined to complete the model: $\alpha_1 = 1/V_s$, $\alpha_2 = h_{fg}/C_pV_s$, $\alpha_3 = 1/\rho C_pV_s$, $\alpha_4 = 1/\rho V_s$, $\beta_1 = 1/V_{he}$, $\beta_2 = 1/\rho C_pV_{he}$, $\beta_3 = h_w/C_pV_{he}$. Eqs. (43)–(45) can be re-written as:

$$\dot{x}_1 = u_1\alpha_1 60(x_3 - x_1) - u_1\alpha_2 60(W_s - x_2) + \alpha_3\left(Q_0 - h_{fg}M_0\right) \tag{111}$$

$$\dot{x}_2 = u_1\alpha_1 60(W_s - x_2) + \alpha_4 M_0 \tag{112}$$

Table 4
Numerical values of the system parameters.

$\rho = 0.0074$ $[lb/ft^3]$	$C_p = 0.24$ $[Btu/lb°F]$	$f_{ref} = 17{,}000$ $[ft^3/min]$
$T_{ref} = 55$ $[°F]$	$T_{3ref} = 71$ $[°F]$	$W_s = 0.007$ $[lb/lb]$
$W_{3ref} = 0.0088$ $[lb/lb]$	$V_{he} = 60.75$ $[ft^3]$	$V_s = 58{,}464$ $[ft^3]$

Table 5
Numerical values of the system parameters at the operating point.

$x_1^0 = 71$ $[°F]$	$x_2^0 = 0.0092$ $[lb/lb]$	$x_3^0 = 55$ $[°F]$
$T_0^0 = 85$ $[°F]$	$W_0^0 = 0.0018$ $[lb/lb]$	$M_0^0 = 166.06$ $[lb/hr]$
$u_1^0 = 17{,}000$ $[ft^3/min]$	$u_2^0 = 58$ $[gpm]$	$Q_0^0 = 289{,}897.52$
$W_s^0 = 0.007$ $[lb/lb]$		

$$\dot{x}_3 = u_1\beta_1 60(x_1 - x_3) + u_1\beta_1 15(T_0 - x_1) - u_1\beta_3 60(0.25W_0 + 0.75x_2 - W_s) - 6000u_2\beta_2 \tag{113}$$

Tables 4 and 5 contain the numerical values chosen for the simulation and the system parameters at the operating point, respectively.

u_1 and u_2 are the control actions that modify the target variables T_3 (x_1) and $W_3(x_2)$. Fig. 13 shows the interaction of the inputs and outputs of the MIMO control problem.

$G_1(\cdot)$, $G_2(\cdot)$, and $G_3(\cdot)$ are the expressions (111)–(113), respectively.

To implement the LAMDA controllers in each of the control variables, it is necessary to analyze if a decoupling stage is required so that the design consists of independent controllers for each variable. The process to identify the correlation between inputs and outputs is based on the procedure of reaction curves to obtain the transfer functions, applying a step at one of the inputs, and monitoring the response at the outputs, obtaining the numerical values in the form of FOPDT system. The detailed procedure to obtain the transfer functions is presented in [49]. The linearized model can be represented by the $G(s)$ matrix:

$$X(s) = G(s)U(s) \tag{114}$$

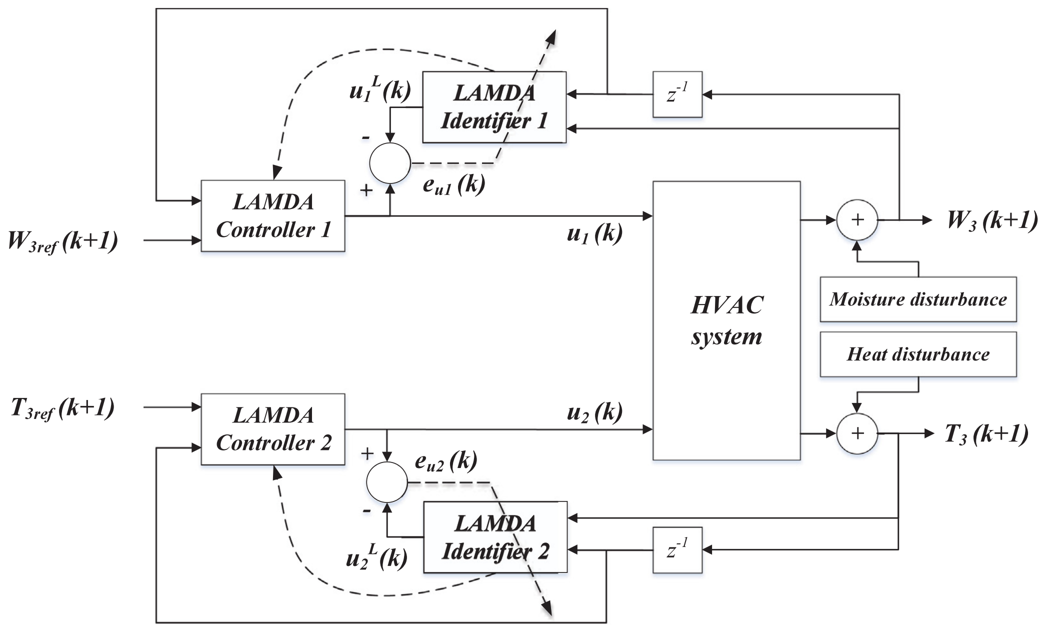


Fig. 14. Adaptive control structure for the HVAC system.

$$G(s) = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} = \begin{bmatrix} \dfrac{9.8164\times10^{-4}e^{-0.0016}}{0.2137s+1} & \dfrac{-1.3223e^{-0.0012s}}{0.2301s+1} \\ \dfrac{-1.1764\times10^{-7}e^{-0.0011s}}{0.0527s+1} & 0 \end{bmatrix} \quad (115)$$

From (115), the gains of the transfer function are obtained to form the gain matrix K.

$$K = \begin{bmatrix} 9.8164\times10^{-4} & -1.3223 \\ -1.1764\times10^{-7} & 0 \end{bmatrix} \quad (116)$$

The relative gain array RGA [63] (Bristol's matrix) is used to measure the interaction between the inputs and outputs in a multivariate process, and it is defined as:

$$RGA\,(K) = \Lambda\,(K) \triangleq K \times \left(K^{-1}\right)^T \quad (117)$$

The operator $\times$ denotes the element-by-element multiplication:

$$\Lambda\,(K) = \begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (118)$$

$\Lambda\,(K)$ shows the dependence between the inputs and outputs. Based on these terms, the decoupling stage is not necessary for the control. Due to the HVAC system characteristics and the resulting parameters of $\Lambda\,(K)$, the control design with two independent LAMDA controllers, one for the temperature x_1 and another for the humidity ratio x_2, is feasible.

$$u_2 \rightarrow x_1 \quad and \quad u_1 \rightarrow x_2 \quad (119)$$

Finally, to proceed with the simulation of the proposed control, the HVAC system is discretized by applying the Euler method to Eqs. (111)–(113), considering the sample time T_s:

$$\begin{aligned} x_1\,(k+1) &= T_s[u_1\,(k)\,\alpha_1 60\,(x_3\,(k) - x_1\,(k)) \\ &\quad - u_1\,(k)\,\alpha_2 60\,(W_s - x_2\,(k)) \\ &\quad + \alpha_3\left(Q_0 - h_{fg}M_0\right)] + x_1\,(k) \quad (120) \end{aligned}$$

$$x_2\,(k+1) = T_s\,[u_1(k)\alpha_1 60\,(W_s - x_2(k)) + \alpha_4 M_0] + x_2\,(k) \quad (121)$$

$$\begin{aligned} x_3\,(k+1) &= T_s[u_1(k)\beta_1 60\,(x_1\,(k) - x_3\,(k)) \\ &\quad + u_1(k)\beta_1 15\,(T_0 - x_1\,(k)) \\ &\quad - u_1\,(k)\,\beta_3 60\,(0.25W_0 + 0.75x_2\,(k) - W_s) \\ &\quad - 6000u_2(k)\beta_2] + x_3\,(k) \quad (122) \end{aligned}$$

Fig. 14 shows the operational scheme of the control system with two separated control loops in the application stage, to regulate the two variables in the thermal space of the system.

For the training stage, the LAMDA Identifier 1 uses two inputs $[W_3\,(k+1)\,;W_3\,(k)]$ (because we consider a first-order plant based on [49]), each with two classes, the design parameters are $\eta_1 = 0.02$, $\beta_1 = 0.01$, $\lambda_1 = 0.997$. LAMDA Identifier 2 uses two inputs $[T_3\,(k+1)\,;T_3\,(k)]$, each with two classes, and the design parameters are $\eta_2 = 0.95$, $\beta_2 = 0.01$, $\lambda_2 = 0.997$. The sampling period of the simulation is $T_s = 1.2$ min. Similar to case study 1, a random input is generated for the plant that consists of 60different random values of 54 min duration.

In Fig. 14, the online learning block (*LAMDA Identifier 1*) is placed between the control action u_1 and the output Humidity Ratio W_3, to learn the inverse model of the system using current and past information. The system output W_3 and its previous state are used as inputs for the identifier, in order to minimize the error $e_{u1}(k) = u_1(k) - u_1^L(k)$, where u_1^L is the output of the LAMDA identifier. The minimization of $e_{u1}(k)$ allows adjusting the parameters of the LAMDA model, which are updated in the controller at every sample time. The procedure described above is similarly applied to control the temperature T_3 of the Thermal Zone, considering the minimization of the error$e_{u2}(k) = u_2(k) - u_2^L(k)$, where $u_2^L(k)$ is the output of the *LAMDA Identifier 2*.

The performance of Adaptive LAMDA controller is analyzed by evaluating its response in the presence of abrupt disturbances, to test the robustness. The IAE is compared with the controllers Fuzzy-PI, LAMDA-PI and ANFIS. Fuzzy-PI and LAMDA-PI were designed based on the expertise of the plant. ANFIS and LAMDA based on online learning were set with the same values for the parameters applied to the learning stage. Two types of disturbances are applied to the HVAC system separately, to observe the behavior of each control variable in the Thermal Zone 3: Heat and Humidity Ratio (see Figs. 15a and 15b, respectively).

First, only the temperature disturbance is applied to the plant. Fig. 16a shows that the control action u_1 stays at 17000[*cfm*], while the Humidity Ratio stays at 0.0092[*lb*/*lb*]. Thus, the temperature disturbance does not affect W_3, see (120)–(122).

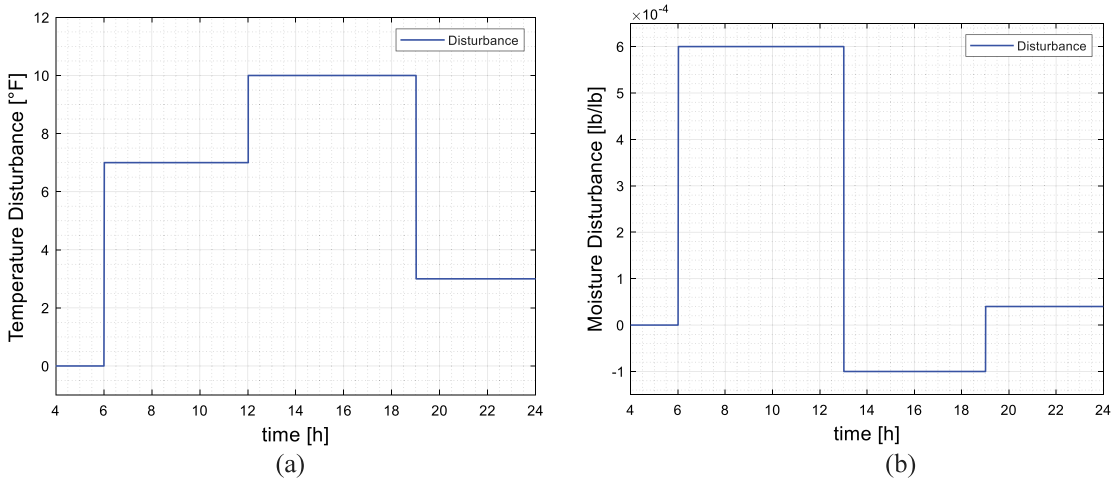


Fig. 15. (a) Heat disturbance signal, (b) Moisture disturbance signal applied to robustness analysis.

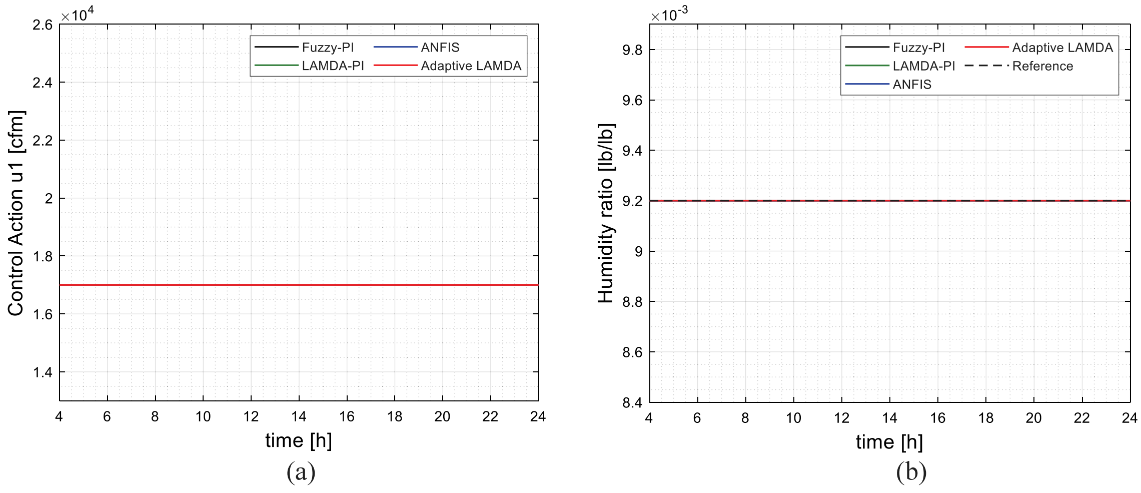


Fig. 16. Comparative results with temperature disturbance: (a) control action u_1, (b) Humidity Ratio W_3.

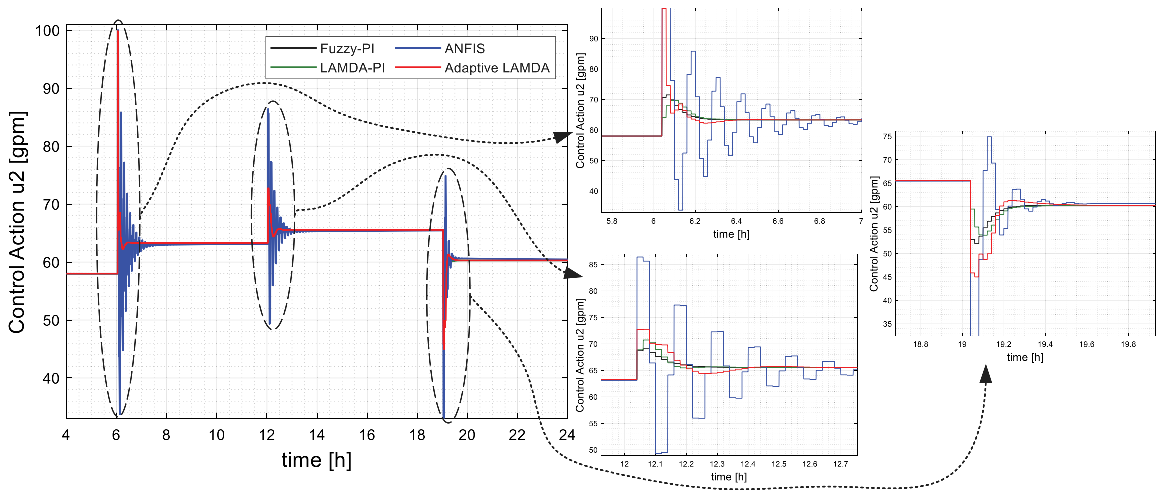
Fig. 17a shows the behavior of the control action u_2 and Fig. 17b shows the variation of the temperature T_3 when the temperature disturbance is applied.

The calibration of the non-adaptive methods in this system has been complex, and it takes a lot of time for this process because there are two controllers and several parameters to set, such as gains, the classes and their values. These methods present very good results in the plant when abrupt disturbances (the worst conditions) are applied to the output temperature, as is shown in Fig. 17. Our proposal also presents satisfactory results in the control of this process, avoiding the problems exposed in the design of the non-adaptive controllers. The control signal of our method is abrupt with respect to the Fuzzy-PI and LAMDA-PI controllers, due to the learning parameters selected for this experiment. See the zoom in Fig. 17, where the effectiveness of our proposal is evaluated qualitatively, highlighting the transient in the response of the controllers, and observing that our method responds quickly without error in steady-state. The ANFIS control being the most similar to our proposal, has a fairly abrupt and oscillatory response, which consequently leads to greater overshoot. In addition, this method presents an error in a steady-state of around $\pm 0.5[°F]$.

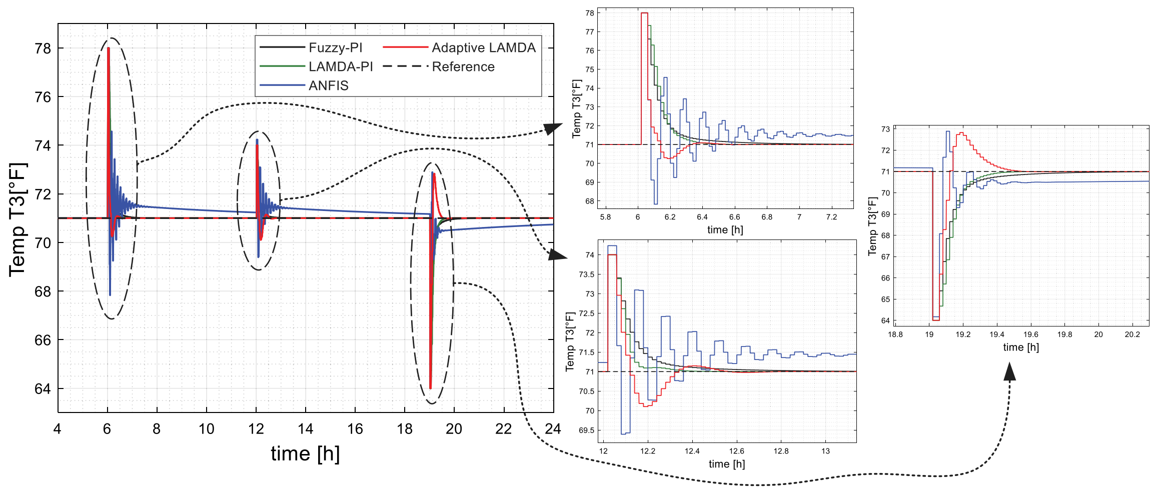
In the next experiment, the moisture disturbance is applied to the plant in order to analyze how the variables T_3 and W_3 are affected, and how the controllers are able to regulate them. Fig. 18a shows the control action u_1, and Fig. 18b shows the behavior of Humidity Ratio $W_3$for all the analyzed controllers. Fig. 19a shows the control action u_2 and Fig. 19b shows the behavior of the Temperature $T_3$for all the analyzed controllers.

Figs. 18, 19 show that the abrupt moisture disturbance applied to the Humidity Ratio, affects the variables W_3 and T_3. For the Humidity Ratio, it can be seen that the output of the Adaptive LAMDA control is very good, with few oscillations in the transient response, and quick to reach the reference (fast convergence), without overshoot because the control action is not abrupt. This demonstrates that the online learning performed by the algorithm when the system is subjected to disturbances is adequate (see the zoom in Figs. 18 and 19). The convergence of our approach is better than the non-adaptive methods and the transient response is faster. The calibration of the non-adaptive methods in this case has an extra complexity degree due to the interaction between the input and output variables, which is avoided with our adaptive method. It is also observed that our approach is much better than the ANFIS controller, which is oscillatory and not able to reach the reference, presenting an error in a steady-state of around $\pm 0.2 \times 10^{-3}[°lb/lb]$, especially with the disturbances at time $6h$ and $13h$. The selected learning parameters of the algorithm are adequate for the HVAC system, and the algorithm works very well with only two classes per descriptor, reducing the computational time.

Fig. 19 shows that our approach presents a very good response regulating the Temperature T_3. Control actions of Adaptive



(a)



(b)

Fig. 17. Comparative results with temperature disturbance: (a) control action u_2, (b) Temperature T_3.

LAMDA are less abrupt than Fuzzy-PI controller (black line), and stabilize the system in a very short time as desired in control systems, which allows us to conclude that the selection of learning parameters is adequate for rapid convergence without overshoot in response, such as LAMDA-PI (method that requires a complex calibration). ANFIS, as in the previous cases, is the controller with the most oscillatory response. In this case, the steady-state error is around $\pm 0.25°[F]$ for the two initial disturbances, and for a disturbance at time $19h$ the error is zero, but reaching this value in a longer time compared to the other proposals, which is not useful in these systems.

A quantitative analysis, computing the IAEs after applying the temperature and moisture disturbances to the plant, is shown in Table 6. For the temperature disturbance, the Adaptive LAMDA controller is the best (minimum value). For the moisture disturbance, our approach is the best to control this variable, and the second-best to control the temperature, with the advantage that our proposal does not require a tuning method for the parameters and a previous knowledge of the plant. The LAMDA-PI controller has very good results due to the fact that an exhaustive knowledge engineering has been used to establish the classes and rules on which the control actions are defined, which is a process that requires time and must be properly calibrated. It is not required by Adaptive LAMDA, being this its main advantage.

In the case of the temperature disturbance, since the humidity ratio variable is not affected, there is no reference change in that variable, as is shown in Fig. 16b. Because of this, the IAE in controller 1 is zero in all cases.

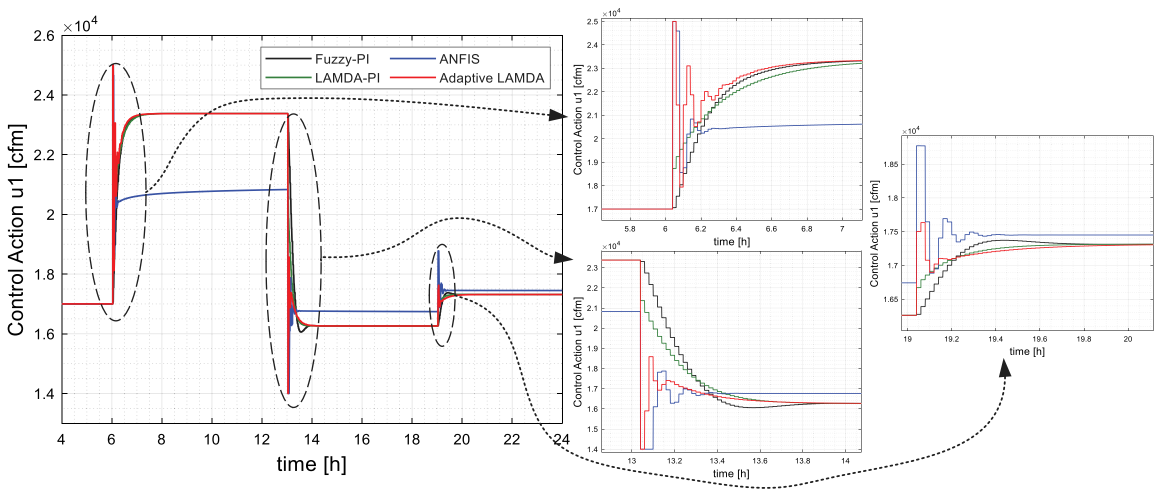
Table 6
Numerical values for IAE for the HVAC experiments.

IAE computed with temperature disturbance

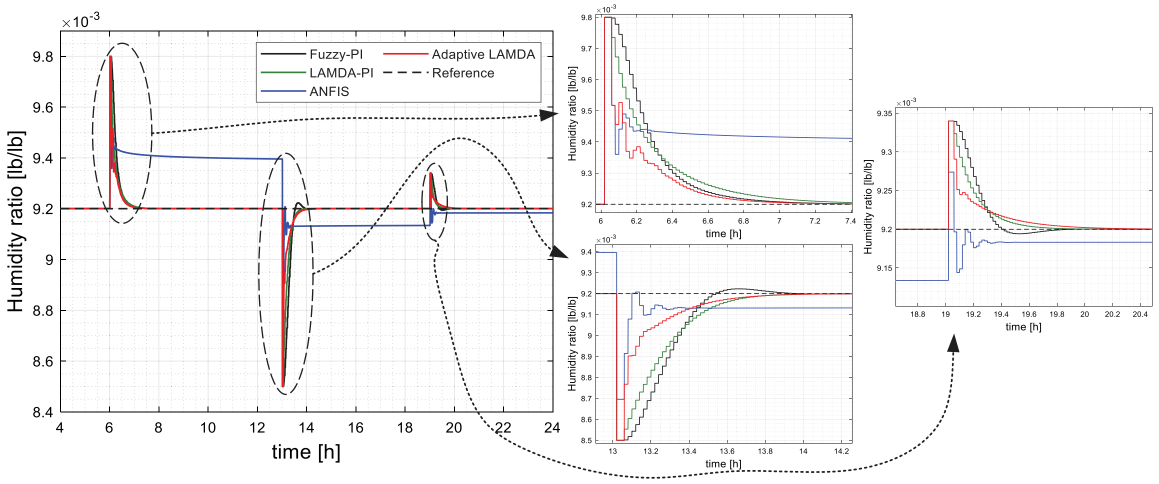
Controller	*Fuzzy-PI*	*LAMDA-PI*	*ANFIS*	*Adaptive LAMDA*
Controller 1	0	0	0	0
Controller 2	2.04	1.911	7.689	1.529

IAE computed with moisture disturbance

Controller	*Fuzzy-PI*	*LAMDA-PI*	*ANFIS*	*Adaptive LAMDA*
Controller 1	3.71×10^{-4}	3.49×10^{-4}	1.96×10^{-3}	$2.25. \times 10^{-4}$
Controller 2	1.879	0.377	2.912	0.492



(a)



(b)

Fig. 18. (a) Comparative results with moisture disturbance: (a) control action u_1, (b) Humidity Ratio W_3.

5.3. *Case study 3: Tracking the trajectory of a mobile robot*

Finally, to validate the proposed controller in tracking trajectory tasks, we present its application in a mobile robot. Trajectory control of a mobile robot is one of the objectives to be achieved in the field of autonomous robotics, due to the large number of associated applications as: risky or hazardous tasks for the humans, defense, medical, automation of industries and processes, among others [64]. Because the dynamic models of these systems are complex to obtain, or could present errors, it is necessary to design robust controllers that can compensate for these problems. For this reason, our proposal is applied to these systems, in which the algorithm will learn from the dynamics of the system (which is completely unknown) for the development of the controller based on the inverse model, without requiring previously the dynamic robot model, in order to apply it to the task of tracking different trajectories in a robot simulation environment.

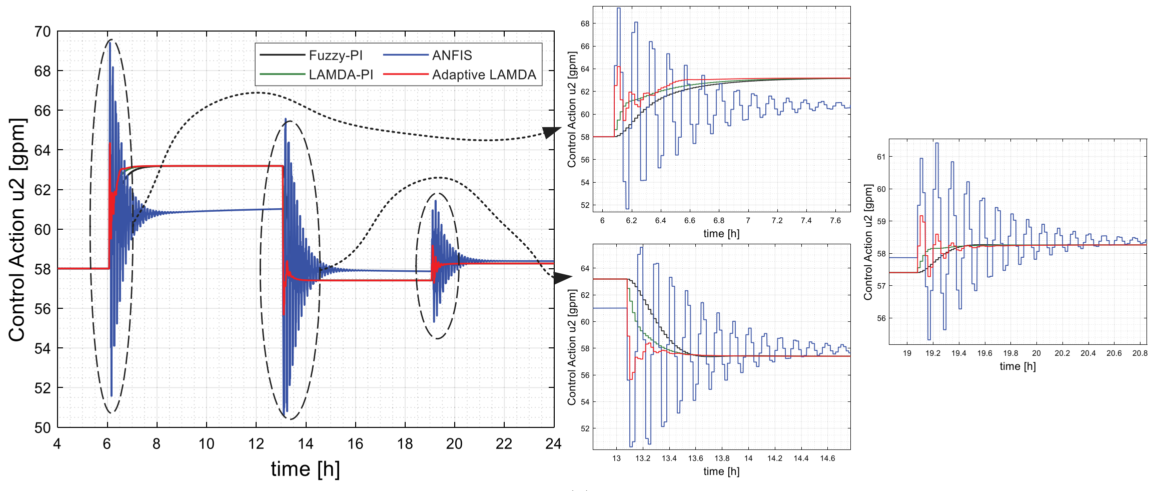
5.3.1. *Robot model*

The unicycle type robot is widely used in the field of automatic control due to its fast and nonlinear dynamics. Fig. 20 shows a representation of the robot, where v and ω are the linear and angular velocities, respectively, h is the point of interest with x, y coordinates in the XY plane, ψ is the orientation of the robot, a is the distance between h and the central point of the virtual axis B that connects the wheels, and r_1 is the radius of the wheels.

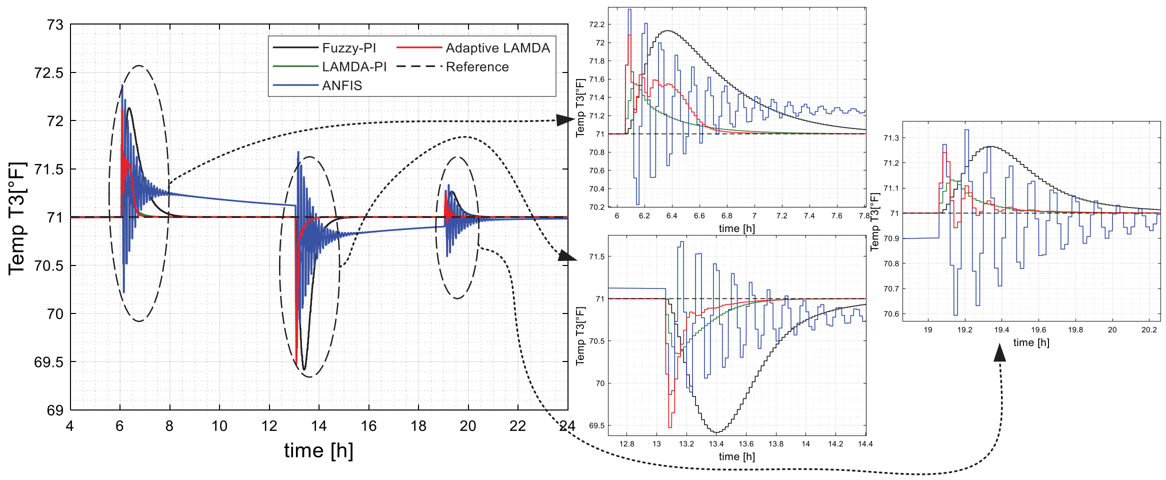
The complete mathematical representation of the mobile robot consists in the kinematic and the dynamic model. The general discretized kinematic model, assuming that the disturbance term is a zero vector and considering that T_s is the sample time [19], is:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \psi(k+1) \end{bmatrix} = T_s \begin{bmatrix} \cos\psi(k) & -\mathrm{a}\sin\psi(k) \\ \sin\psi(k) & \mathrm{a}\cos\psi(k) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ \omega(k) \end{bmatrix} + \begin{bmatrix} x(k) \\ y(k) \\ \psi(k) \end{bmatrix} \tag{123}$$

In [19] is proposed the application of two controllers, one of them based on feedback linearization for the robot kinematics, and the other one based on the dynamics. In our case, we consider the dynamic model as unknown (black box), thus, its identification and control is done with the Adaptive LAMDA, which is the main contribution in this experiment.



(a)



(b)

Fig. 19. Comparative results with relative humidity disturbance: (a) control action u_2, (b) Temperature T_3.

5.3.2. *Kinematic controller*

The kinematic controller (see Eq. (124)) is based on the kinematic model of the robot given by (123), considering the coordinates of the point of interest $[x, y]^T$. The control law is:

$$\begin{bmatrix} v_{ref}^c(k) \\ \omega_{ref}^c(k) \end{bmatrix} = \begin{bmatrix} \dfrac{\cos\psi\,(k)}{T_s} & \dfrac{\sin\psi\,(k)}{T_s} \\ -\dfrac{1}{a}\dfrac{\sin\psi\,(k)}{T_s} & \dfrac{1}{a}\dfrac{\cos\psi\,(k)}{T_s} \end{bmatrix} \times \begin{bmatrix} x_{ref}\,(k+1) + l_x \tanh\left(\dfrac{k_x}{l_x} e_x\,(k)\right) - x\,(k) \\ y_{ref}\,(k+1) + l_y \tanh\left(\dfrac{k_y}{l_y} e_y\,(k)\right) - y\,(k) \end{bmatrix} \tag{124}$$

Where $a > 0$, $\begin{bmatrix} v_{ref}^c(k) & \omega_{ref}^c(k) \end{bmatrix}^T$ is the output of the kinematic controller, $e_x\,(k) = x_{ref}\,(k) - x(k)$, and $e_y\,(k) = y_{ref}\,(k) - y(k)$ are the position errors in the X and Y axis respectively, $k_x > 0$, $k_y > 0$ are the gains of the controller, $l_x, l_y \in \mathbb{R}$ are saturation constants. The tanh $(\cdot)$ function is added to avoid a saturation of the control actions in the case of large position errors [65]. In the stability analysis, perfect velocity tracking is considered, $v_{ref}^c(k) \equiv v\,(k)$ and $\omega_{ref}^c(k) \equiv \omega\,(k)$. By replacing (124) in (123), the closed-loop equation is:

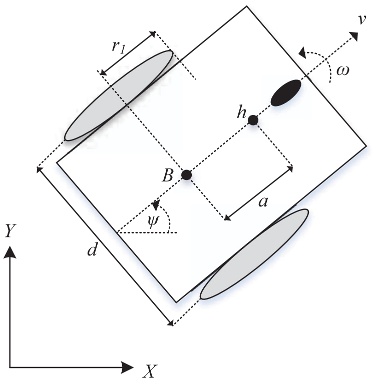


Fig. 20. Parameters of the unicycle-like mobile robot.

$$\begin{bmatrix} e_x\,(k+1) \\ e_y\,(k+1) \end{bmatrix} + \begin{bmatrix} l_x \tanh\left(\dfrac{k_x}{l_x} e_x\,(k)\right) \\ l_y \tanh\left(\dfrac{k_y}{l_y} e_y\,(k)\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{125}$$

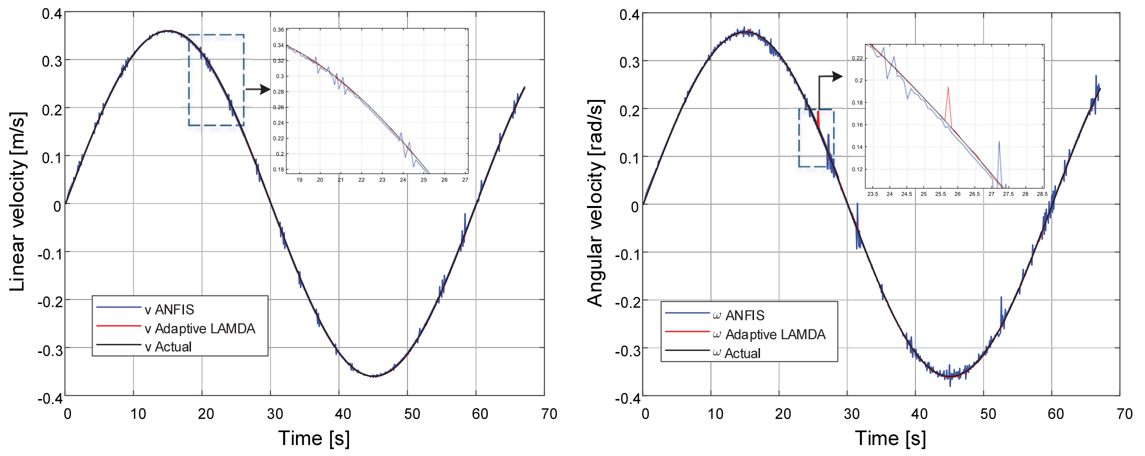


Fig. 21. Comparison of learning algorithms in mobile robot (a) linear velocity, (b) angular velocity.

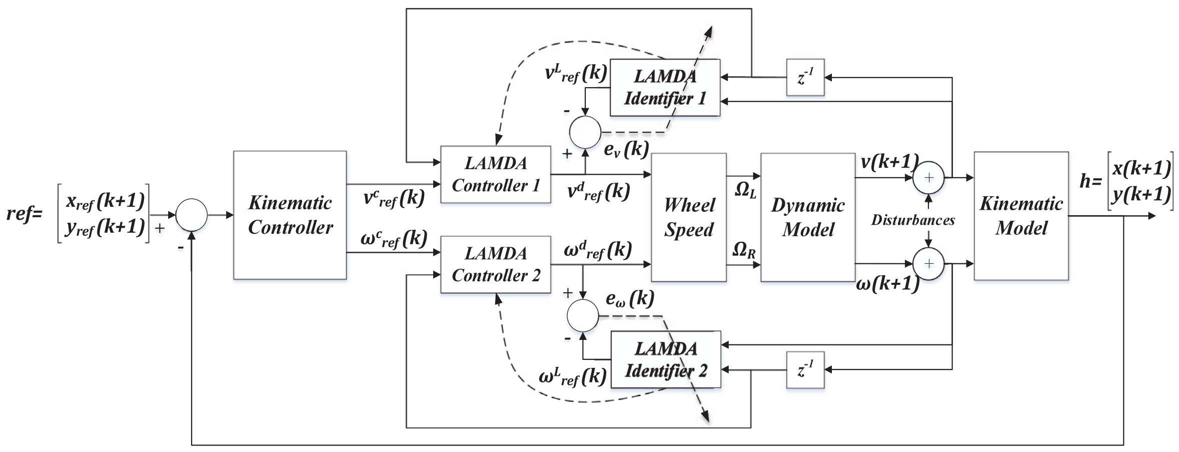


Fig. 22. Adaptive control structure for a mobile robot.

Defining the output error vector as $\tilde{h}(k) = \begin{bmatrix} e_x(k) & e_y(k) \end{bmatrix}^T$, then (125) can be written as:

$$\tilde{h}(k+1) = -\begin{bmatrix} l_x \tanh\left(\frac{k_x}{l_x} e_x(k)\right) \\ l_y \tanh\left(\frac{k_y}{l_y} e_y(k)\right) \end{bmatrix} \tag{126}$$

In [19] has been selected the Lyapunov's candidate function for the kinematic control law as $V(k) = \frac{1}{2}\tilde{h}^T(k)\,\tilde{h}(k)$. In the cited paper is demonstrated the stability of the kinematic controller for tracking trajectories if the parameters are set as $k_x < 1$, $k_y < 1$, $\frac{k_x}{l_x} < 1$ and , $k_y/l_y < 1$, then $\tilde{h} \to \infty$ for $k \to \infty$.

5.3.3. Dynamic controller

The design of the dynamic controller is complex because a large number of parameters corresponding to the actuation mechanisms and physical variables of the robot must be considered in real-time. For this reason, non-adaptive controllers (Fuzzy-PI and LAMDA-PI) are not tested in this experiment since their calibration is complex and time-consuming, therefore, the adaptive methods are appropriate in this system. The following results are for ANFIS and Adaptive LAMDA since they are proposals that can learn about the dynamic of the system and do not require parametric calibration for the design.

In this case study, the benefits of the Adaptive LAMDA are clearly appreciated since the algorithm learns the dynamics of the system, which is considered as unknown and variable. Our method is used to model it as follows: in the training stage, LAMDA is applied to learn the inverse dynamic model based on the scheme of Fig. 4a. The controller takes as input information the computed reference values in the output of the kinematic controller $\begin{bmatrix} v^c_{ref} & \omega^c_{ref} \end{bmatrix}^T$ and the measured variables of the robot $\begin{bmatrix} v & \omega \end{bmatrix}^T$. With this information, the identifier updates the internal parameters of the LAMDA model in the controller (obtaining the inverse model) in each sample time. For the training stage, the LAMDA Identifier 1 uses two inputs $[v(k+1); v(k)]$, each with two classes, $\eta_1 = 0.08$, $\beta_1 = 0.01$, $\lambda_1 = 0.999$. LAMDA Identifier 2 uses two inputs $[\omega(k+1); \omega(k)]$, each with two classes, $\eta_2 = 0.08$, $\beta_2 = 0.01$, $\lambda_2 = 0.999$. The sampling period of the simulation is $T_s = 0.1seg$. A sinusoidal input of 670 samples is generated for the system. Fig. 21 shows the comparison of learning algorithms in the robot, which shows a better fit to the real values of linear and angular velocity of the adaptive LAMDA with respect to ANFIS.

In the application, the controller computes the output $\begin{bmatrix} v^d_{ref}(k) & \omega^d_{ref}(k) \end{bmatrix}^T$ necessary to bring the system to the reference. The proposed structure, with an external kinematic controller and an internal dynamic controller based on the adaptive methods (LAMDA or ANFIS), is the cascade scheme shown in the block diagram of Fig. 22.

In the scheme, the online learning block (*LAMDA Identifier 1*) is placed between the control action computed by the dynamic controller $v^d_{ref}(k)$ and the measured variable v, to learn the inverse dynamics of the system using current and past information. The linear velocity v and its previous state are used as inputs for the identifier, in order to minimize the error $e_v(k) = v^d_{ref}(k) - v^L_{ref}(k)$,

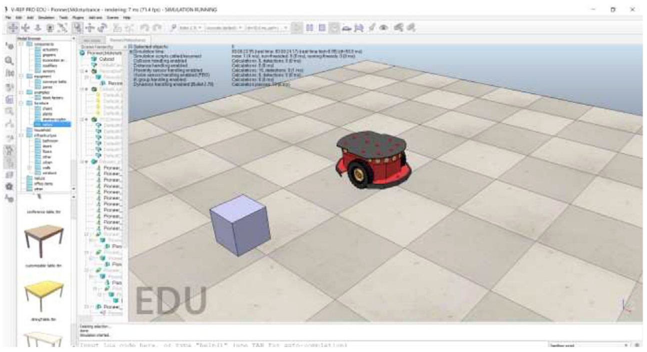


Fig. 23. V-REP simulation scene showing the Pioneer 3dx robot.

where $v^L_{ref}(k)$ is the output of the identifier. The minimization of e_v allows adjusting the parameters of the LAMDA model that are updated in the controller, at every sample time. The procedure described above is similarly applied to control the angular velocity ω, considering the minimization of the error $e_\omega(k) = \omega^d_{ref}(k) - \omega^L_{ref}(k)$, where $\omega^L_{ref}(k)$ is the output of the *LAMDA Identifier 2*. The control variables are the motor velocities, then we need to find the speed for the left and right wheels Ω_L and Ω_R, respectively, based on the values of $\left[v^d_{ref}(k) \quad \omega^d_{ref}(k)\right]^T$. These relations are given by:

$$\Omega_L = \frac{2v^d_{ref}(k) - d\omega^d_{ref}(k)}{2r_1} \quad \text{and} \quad \Omega_R = \frac{2v^d_{ref}(k) + d\omega^d_{ref}(k)}{2r_1} \tag{127}$$

The proposed controllers are tested on a Pioneer 3DX robot [66] inside a virtual 3d robot environment. A Virtual Robot Experimentation Platform (V-REP) allows simulating robotic systems considering their kinematics, dynamics and the physic of the environment [67]. The versatility of this software is linked to the availability of plug-ins to connect with other computational tools, such as Matlab, where our algorithms have been programmed. The main user interface of V-REP with the Pioneer 3DX robot is shown in Fig. 23.

The performance of the Adaptive LAMDA controller is tested against the ANFIS, such that the operating modes are similar to make a fair comparison. In this case study, the aim is to perform the trajectory control of the Pioneer 3DX applied in three different paths, applying a load to the robot (as a disturbance) to modify its dynamics and analyze the performance of the controllers. Graphical and numerical comparisons are performed to test the performance and effectiveness of the algorithms in this control task (see IAE in Table 7).

Three trajectories are tested: Circular (see Eq. (128)), Lemniscate curve (see Eq. (129)) and Square (see Eq. (130)). The starting point of the robot in all cases is in the coordinate $(x, y) = (0, 0)m$.

$$\begin{cases} x_{ref}(k) = 2\cos(0.033\pi kT_0) \\ y_{ref}(k) = 2\sin(0.033\pi kT_0) \end{cases} \tag{128}$$

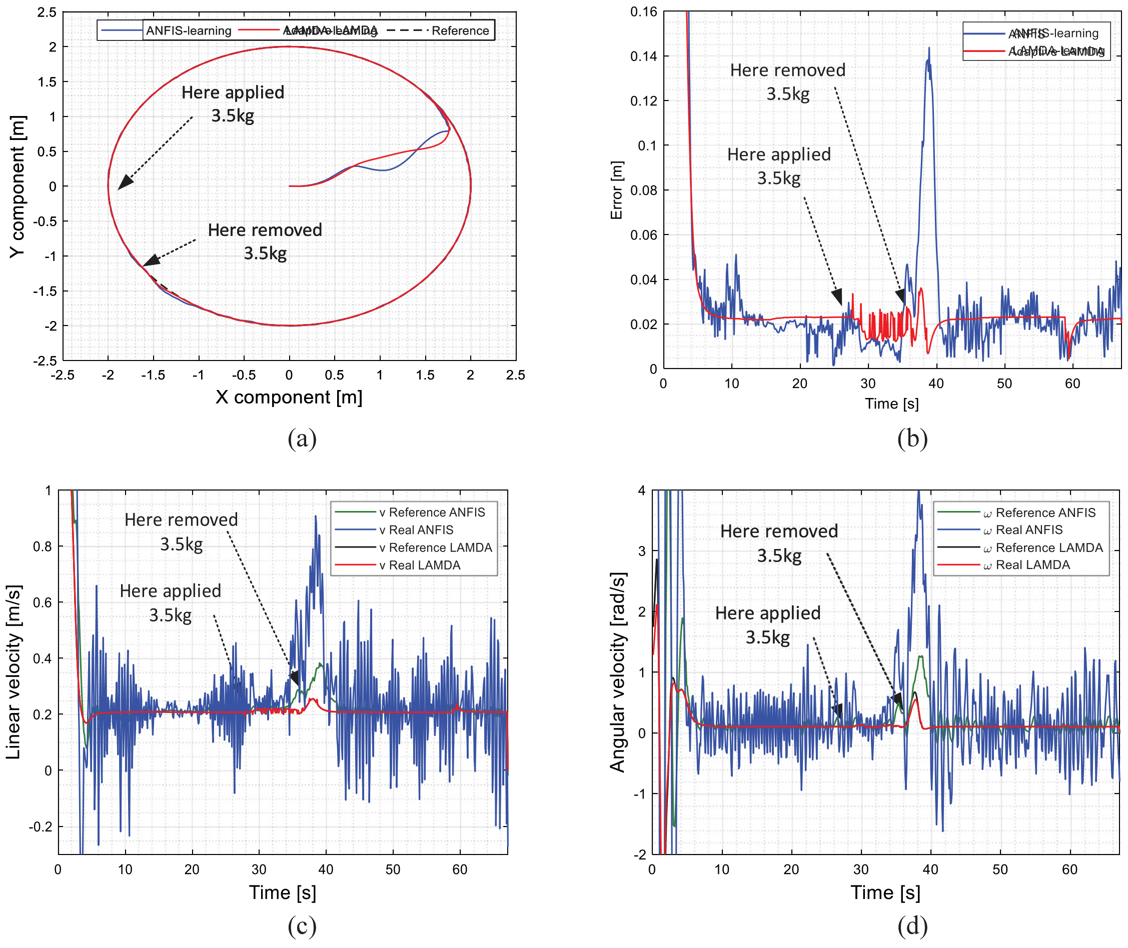


Fig. 24. (a) Circular trajectory followed by the robot, (b) instantaneous quadratic error of the robot position, speeds of the robot and control actions (c) linear velocity and (d) angular velocity.

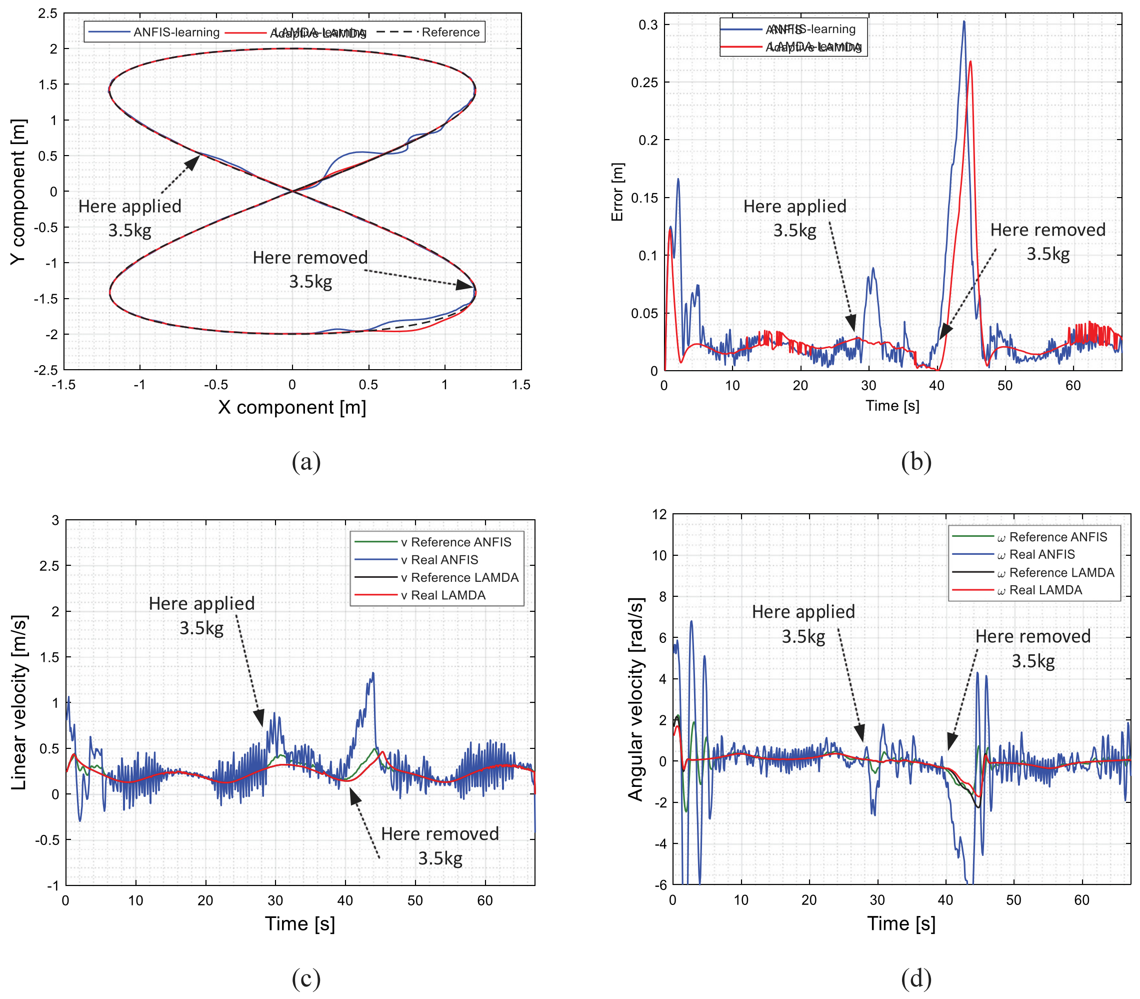


Fig. 25. (a) Lemniscate trajectory followed by the robot, (b) instantaneous quadratic error of the robot position, speeds of the robot and control actions (c) linear velocity and (d) angular velocity.

$$\begin{cases} x_{ref}\,(k) = 1.2\sin\left(0.063\pi kT_0\right) \\ y_{ref}\,(k) = 2\sin\left(0.0315\pi kT_0\right) \end{cases} \tag{129}$$

$$\begin{cases} x_{ref}\,(k) = 1.5\forall kT_o \in [0, 15];\ (4.5 - 0.2kT_o)\,\forall kT_o \in [15, 30]; \\ -1.5\forall kT_o \in [30, 45]\,;\ (-10.5 + 0.2kT_o)\,\forall kT_o \in [45, 60] \\ y_{ref}\,(k) = (-1.5 + 0.2kT_o)\,\forall kT_o \in [0, 15];\ 1.5\forall kT_o \in [15, 30]; \\ (7.5 - 0.2kT_o)\,\forall kT_o \in [30, 45]\,;\ -1.5\forall kT_o \in [45, 60] \end{cases} \tag{130}$$

Figs. 24–26 show the comparative results for the Circular, Lenmiscate and Square trajectories, respectively, in which the response of the ANFIS and Adaptive LAMDA controllers are shown, as well as the position error in each trajectory. Additionally, the linear and angular speed references are shown, with the respective real values reached by the robot. During the simulation of this experiment, a 3.5 kg load is added and removed on the robot at different time instants, affecting its dynamics to analyze the controllers' response.

Figs. 24a, 25a and 26a show the paths followed by the mobile robot. The results show qualitatively that the Adaptive LAMDA provides finer and more efficient control with respect to ANFIS (the two methods designed with the same learning parameters), getting a smaller distance error with respect to the references, and especially, under disturbances. The instantaneous quadratic error of the robot position controlled by the Adaptive LAMDA has an average of 2 cm (see Figs. 24b, 25b and 26b), a value considered acceptable taking into account that the dynamic of the controller is not based on the mathematical model of the system for its design. In the case of the square trajectory, it is observed that the errors in the corners reach values of 9 cm due to the abrupt changes in the orientation of the robot, but as is observed, they are quickly corrected by the LAMDA controller.

The linear speed in all the tested trajectories is around 0.2m/s (see Figs. 24c, 25c and 26c), references reached by the Adaptive LAMDA controller with very good performance and fast convergence, showing to be more efficient qualitatively than ANFIS due to the softer response and better in quantitative terms if we analyze the IAE values in Table 7. Then, it is clear that the learning parameters set for the linear speed controller are adequate because our method does not show oscillations in the control actions, while ANFIS presents a large number of oscillations, with amplitudes around ±0.4 m/s, which is excessive if compared to established references that can damage the actuators by the applied energy variations.

In the case of angular velocity, the tests show that the Adaptive LAMDA presents smooth control actions again. The ANFIS proposals have oscillations around the reference of ±5 rad/s. This behavior can be seen in the Figs. 24d, 25d and 26d, where it is observed that our proposal converges faster to the references than ANFIS. Also, the learning parameters set for the angular speed controller are adequate because our method does not show excessive oscillations in the control actions. LAMDA is much

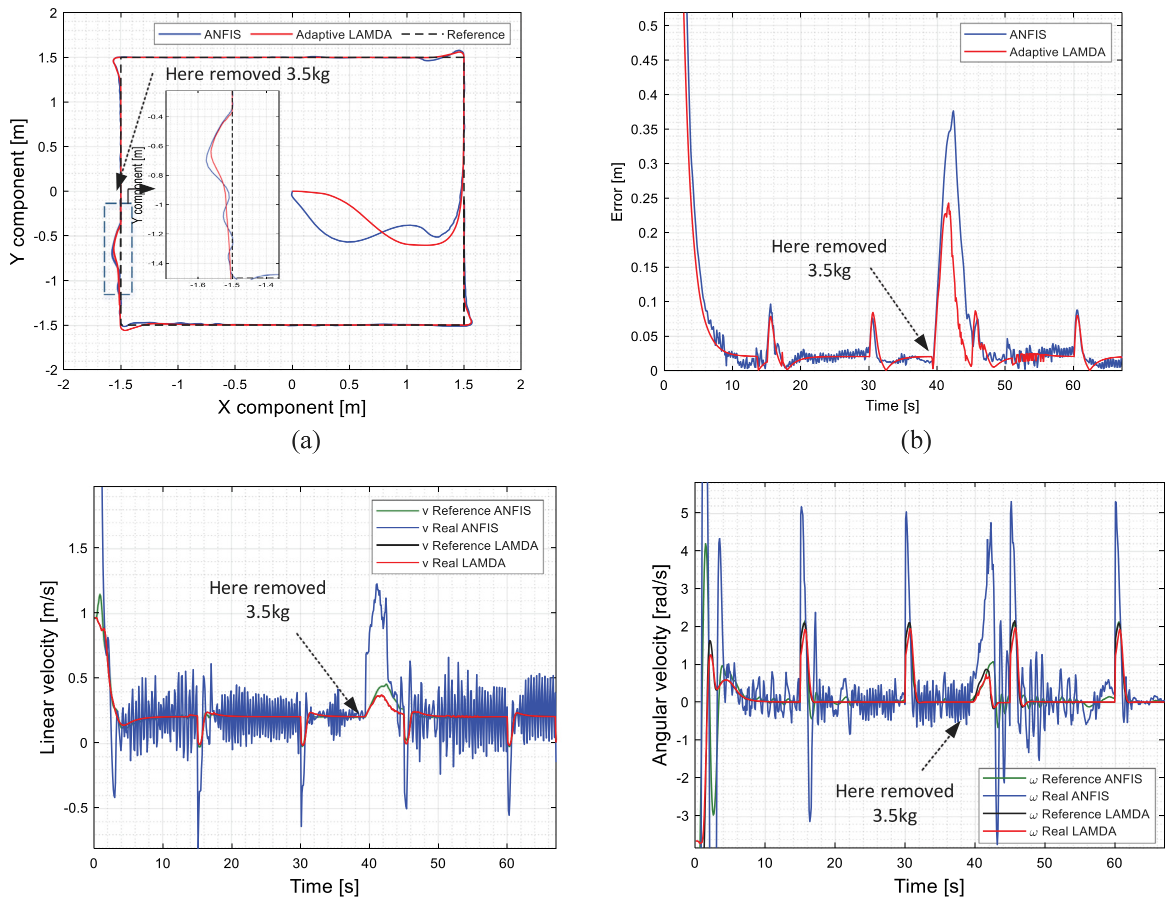


Fig. 26. (a) Square curve trajectory followed by the mobile robot, (b) instantaneous quadratic error of the robot position, speeds of the robot and control actions (c) linear velocity and (d) angular velocity.

Table 7
Numerical values for IAE for the mobile robot experiments.

IAE for trajectory tracking of a mobile robot

Trajectory	*ANFIS*	*Adaptive LAMDA*	*Δ%*
Circle	5.725	5.038	12.77
Lemniscate	2.495	2.101	17.15
Square	7.632	6.510	15.87

better than ANFIS because the control action is smooth, implying that the actuators are not abruptly actuated to reach the reference in steady-state. The reduction of the oscillations is considerable, which is one of the strong points to be highlighted by LAMDA. Finally, from the results obtained through this experiment, it has been possible to observe a very good performance of our proposed tracking controller applied to the dynamic model of the mobile robot, demonstrating its ability to follow the established speed references, and therefore, the desired trajectories. From the quantitative point of view, in this experiment, it is observed the benefits in performance terms of the Adaptive LAMDA with respect to the ANFIS controller, as can be observed in the results of IAE of Table 7 for all the paths. We consider that all these performance improvements are the result of two important factors, the use of aggregation operators in the GAD computation and the adjustment of the exigency parameter, which adapts to system variations online.

In all cases, the performance of LAMDA is better in a percentage greater than 12% over the ANFIS controller, when they were tested in the different trajectories. Under disturbances that affect the dynamic of the system, our controller is the least affected and the one that converges more quickly towards the reference, which allows us to validate our method in fast dynamic robotic systems.

6. Conclusion

In this paper, an Adaptive LAMDA approach based on the online learning criteria for system modeling and control has been presented. The main contribution has focused on providing LAMDA with the capability to control systems without the need to know its mathematical model. The proposed method can be implemented on any system in which its inverse model can be identified, and offers a great advantage over non-adaptive methods as LAMDA-PI or Fuzzy-PI. These methods require the knowledge of an expert about the plant for the design and calibration, which can be complex and time-consuming depending on the system dynamics.

From a qualitative point of view, it has been observed that the Adaptive LAMDA is capable to control systems better than the other proposals in all case studies. In the mixing tank with variable dead time system, it is observed that the algorithm is capable of adapting to changes in the dynamics of the system produced by the variation of the dead time, calculating a less aggressive control action that is capable to take the system to the reference in less time. In the case of the HVAC system, the two variables T_3 and W_3 are adequately regulated, even when moisture and temperature disturbances are added to the system; the quality of the control action of our proposal is quite good if we consider that the algorithm learns directly from the behavior of the plant and has not required calibration, which is indispensable in the non-adaptive methods. Also, it has been observed that the response of our method is better, especially with respect to ANFIS. In the robot tracking trajectory, it is evident that Adaptive LAMDA is much better than ANFIS, it is enough to observe that the control action of our proposal is less abrupt and oscillatory, which is an advantage since the actuators may not respond to the control action computed by ANFIS.

In quantitative terms, the simulations have shown that the proposed method has very good results compared to the other intelligent proposals. In the case of the mixing tank, the results have shown that our proposal is better in terms of performance by 60% over non-adaptive methods (LAMDA-PI and Fuzzy-PI), and by 170% over the more similar approach (ANFIS), which maintains steady-state error without reaching the reference. In the case of regulation of the HVAC system, it has been possible to observe an excellent performance of our proposal in the control stage, in the presence of temperature and moisture disturbances, considerably improving performance with respect to ANFIS, as is shown in the related results with the IAE (see Table 6). Our proposal presents a smoother and less oscillatory control action that eliminates the steady-state error. On the other hand, in the application of trajectory tracking of the mobile robot, it has been observed that LAMDA achieves the control objective with excellent performance, but the most important characteristic to note is the shape of the control actions produced by our method, in which it is clearly observed that the oscillations decrease considerably with respect to ANFIS, with errors in the trajectory of smaller magnitude, that is, better IAE (see Table 7). The three case studies have shown that the main advantage of the learning-based controllers is that they do not depend on the mathematical model of the plants, which are often complex to obtain and may have modeling errors.

The implemented control system is stable as experiments show fulfilling the control objective, both in the learning stage and in the operation stage, but theoretically, it must be demonstrated its global stability properties in future works.

CRediT authorship contribution statement

Luis Morales: Conceptualization, Writing - original draft, Software. **Jose Aguilar:** Conceptualization, Writing - original draft and editing. **Andrés Rosales:** Writing - reviewing and editing. **Danilo Chávez:** Writing - reviewing. **Paulo Leica:** Writing - reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Q. Yang, S. Jagannathan, Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators, IEEE Trans. Syst. Man, Cybern. Part B 42 (2012) 377–390, http://dx.doi.org/10.1109/TSMCB.2011.2166384.

[2] D. Romeres, M. Zorzi, R. Camoriano, S. Traversaro, A. Chiuso, Derivative-free online learning of inverse dynamics models, IEEE Trans. Control Syst. Technol. (2019) 1–15, http://dx.doi.org/10.1109/TCST.2019.2891222.

[3] J. Vieira, F.M. Dias, A. Mota, Artificial neural networks and neuro-fuzzy systems for modeling and controlling real systems: a comparative study, Eng. Appl. Artif. Intell. 17 (2004) 265–273, http://dx.doi.org/10.1016/j.engappai.2004.03.001.

[4] J. Si, Y.T. Wang, On-line learning control by association and reinforcement, IEEE Trans. Neural Netw. 12 (2001) 264–276, http://dx.doi.org/10.1109/72.914523.

[5] M. Hagan, H. Demuth, O. De Jesús, An introduction to the use of neural networks in control systems, Int. J. Robust Nonlinear Control. 12 (2002) 959–985, http://dx.doi.org/10.1002/rnc.727.

[6] T. Waegeman, F. Wyffels, B. Schrauwen, Feedback control by online learning an inverse model, IEEE Trans. Neural Netw. Learn. Syst. 23 (2012) 1637–1648, http://dx.doi.org/10.1109/TNNLS.2012.2208655.

[7] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man. Cybern. SMC-15 (1985) 116–132, http://dx.doi.org/10.1109/TSMC.1985.6313399.

[8] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man. Cybern. 23 (1993) 665–685, http://dx.doi.org/10.1109/21.256541.

[9] M.A. Denaï, F. Palis, A. Zeghbib, Modeling and control of non-linear systems using soft computing techniques, Appl. Soft Comput. 7 (2007) 728–738, http://dx.doi.org/10.1016/j.asoc.2005.12.005.

[10] J.S.R. Jang, Chuen-Tsai Sun, Neuro-fuzzy modeling and control, Proc. IEEE 83 (1995) 378–406, http://dx.doi.org/10.1109/5.364486.

[11] H. Arpaci, O.F. Özgüven, ANFIS & PIλ Dμ Controller design and comparison for overhead cranes, Indian J. Eng. Mater. Sci. 18 (2011) 191–203.

[12] S.R. Khuntia, S. Panda, ANFIS approach for SSSC controller design for the improvement of transient stability performance, Math. Comput. Model. 57 (2013) 289–300, http://dx.doi.org/10.1016/j.mcm.2011.06.052.

[13] M.A. Denai, F. Palis, A. Zeghbib, ANFIS based modeling and control of non-linear systems: a tutorial, in: 2004 IEEE Int. Conf. Syst. Man Cybern., IEEE, 2004, pp. 3433–3438, http://dx.doi.org/10.1109/ICSMC.2004.1400873.

[14] A. Niasar, A. Vahedi, H. Moghbelli, ANFIS-Based controller with fuzzy supervisory learning for speed control of 4-switch inverter brushless DC motor drive, in: 37th IEEE Power Electron. Spec. Conf., IEEE, 2006, pp. 1–5, http://dx.doi.org/10.1109/PESC.2006.1711957.

[15] F. Baghli, Y. Lakhal, L. El Bakkali, O. Hamdoun, Design and simulation of adaptive neuro fuzzy inference system (ANFIS) controller for a robot manipulator, in: 2014 Second World Conf. Complex Syst., IEEE, 2014, pp. 298–303, http://dx.doi.org/10.1109/ICoCS.2014.7060879.

[16] M. Shafiq, M.A. Shafiq, H.A. Yousef, Stability and convergence analysis of direct adaptive inverse control, Complexity 2017 (2017) 1–12, http://dx.doi.org/10.1155/2017/7834358.

[17] J.L. Calvo-Rolle, O. Fontenla-Romero, B. Pérez-Sánchez, B. Guijarro-Berdiñas, Adaptive inverse control using an online learning algorithm for neural networks, Inform. 25 (2014) 401–414, http://dx.doi.org/10.15388/Informatica.2014.20.

[18] G.L. Plett, Adaptive inverse control of linear and nonlinear systems using dynamic neural networks, IEEE Trans. Neural Netw. 14 (2003) 360–376, http://dx.doi.org/10.1109/TNN.2003.809412.

[19] F.G. Rossomando, C. Soria, R. Carelli, Autonomous mobile robots navigation using RBF neural compensator, Control Eng. Pract. 19 (2011) 215–222, http://dx.doi.org/10.1016/j.conengprac.2010.11.011.

[20] R. Valverde Gil, D. Gachet Páez, Identificación de sistemas dinámicos utilizando redes neuronales RBF, Rev. Iberoam. Autom. Inform. Ind. RIAI 4 (2007) 32–42, http://dx.doi.org/10.1016/S1697-7912(07)70207-8.

[21] W. He, Y. Chen, Z. Yin, Adaptive neural network control of an uncertain robot with full-state constraints, IEEE Trans. Cybern. 46 (2016) 620–629, http://dx.doi.org/10.1109/TCYB.2015.2411285.

[22] V.C. Chen, Y.-H. Pao, Learning control with neural networks, in: Proceedings, 1989 Int. Conf. Robot. Autom, IEEE Comput. Soc. Press, 1989, pp. 1448–1453, http://dx.doi.org/10.1109/ROBOT.1989.100183.

[23] Y. Bao, H. Wang, J. Zhang, Adaptive inverse control of variable speed wind turbine, Nonlinear Dyn. 61 (2010) 819–827, http://dx.doi.org/10.1007/s11071-010-9689-3.

[24] X.F. Yuan, Y.N. Wang, L.H. Wu, Adaptive inverse control of excitation system with actuator uncertainty, Neural Process. Lett. 27 (2008) 125–136, http://dx.doi.org/10.1007/s11063-007-9064-7.

[25] T. Waegeman, B. Schrauwen, Towards learning inverse kinematics with a neural network based tracking controller, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: LNCS, vol. 7064, 2011, pp. 441–448, http://dx.doi.org/10.1007/978-3-642-24965-5_50.

[26] R.E. Precup, M.L. Tomescu, Stable fuzzy logic control of a general class of chaotic systems, Neural Comput. Appl. 26 (2015) 541–550, http://dx.doi.org/10.1007/s00521-014-1644-7.

[27] R.-E. Precup, H. Hellendoorn, A survey on industrial applications of fuzzy control, Comput. Ind. 62 (2011) 213–226, http://dx.doi.org/10.1016/j.compind.2010.10.001.

[28] H. Deng, J. Luo, X. Duan, G. Zhong, Adaptive inverse control for gripper rotating system in heavy-duty manipulators with unknown dead zones, IEEE Trans. Ind. Electron. 64 (2017) 7952–7961, http://dx.doi.org/10.1109/TIE.2017.2696514.

[29] G. Lai, Z. Liu, Y. Zhang, C.L.P. Chen, S. Xie, Y. Liu, Fuzzy adaptive inverse compensation method to tracking control of uncertain nonlinear systems with generalized actuator dead zone, IEEE Trans. Fuzzy Syst. 25 (2017) 191–204, http://dx.doi.org/10.1109/TFUZZ.2016.2554152.

[30] J.-J. Wang, Adaptive inverse position control of switched reluctance motor, Appl. Soft Comput. 60 (2017) 48–59, http://dx.doi.org/10.1016/j.asoc.2017.06.014.
[31] M. Turki, S. Bouzaida, A. Sakly, F. M'Sahli, Adaptive control of nonlinear system using neuro-fuzzy learning by PSO algorithm, in: 2012 16th IEEE Mediterr. Electrotech. Conf., IEEE, 2012, pp. 519–523, http://dx.doi.org/10.1109/MELCON.2012.6196486.
[32] Jia Li, Yu Jinshou, Nonlinear hybrid adaptive inverse control using neural fuzzy system and its application to CSTR systems, in: Proc. 4th World Congr. Intell. Control Autom. (Cat. No.02EX527), IEEE, 2002, pp. 1896–1900, http://dx.doi.org/10.1109/WCICA.2002.1021413.
[33] T. Kumbasar, I. Eksin, M. Guzelkaya, E. Yesil, Adaptive fuzzy model based inverse controller design using BB-BC optimization algorithm, Expert Syst. Appl. 38 (2011) 12356–12364, http://dx.doi.org/10.1016/j.eswa.2011.04.015.
[34] Y. Chen, G. Mei, G. Ma, S. Lin, J. Gao, Robust adaptive inverse dynamics control for uncertain robot manipulator, Int. J. Innov. Comput. Inf. Control. 10 (2014) 575–587.
[35] S. Ravi, M. Sudha, P.A. Balakrishnan, Design of intelligent self-tuning GA ANFIS temperature controller for plastic extrusion system, Model. Simul. Eng. 2011 (2011) 1–8, http://dx.doi.org/10.1155/2011/101437.
[36] Z. Du, X. Li, Q. Mao, A new online hybrid learning algorithm of adaptive neural fuzzy inference system for fault prediction, Int. J. Model. Identif. Control. 23 (2015) 68, http://dx.doi.org/10.1504/IJMIC.2015.067716.
[37] C. Pramod, M. Tomar, G. Pillai, A modified extreme learning ANFIS for higher dimensional regression problems, in: Comput. Intell. Theor. Appl. Futur. Dir. I, Springer Singapore, 2019, pp. 279–292, http://dx.doi.org/10.1007/978-981-13-1135-2_22.
[38] J. Aguilar-Martín, R. López De Mantaras, The process of classification and learning the meaning of linguistic descriptors of concepts, in: Approx. Reason. Decis. Anal, North-Holland Publishing Company, 1982, pp. 165–175.
[39] L. Morales, J. Aguilar, D. Chávez, C. Isaza, LAMDA-HAD, an extension to the LAMDA classifier in the context of supervised learning, Int. J. Inf. Technol. Decis. Mak. 19 (2020) 283–316, http://dx.doi.org/10.1142/S0219622019500457.
[40] J.F. Botía Valderrama, D.J.L. Botía Valderrama, On LAMDA clustering method based on typicality degree and intuitionistic fuzzy sets, Expert Syst. Appl. 107 (2018) 196–221, http://dx.doi.org/10.1016/j.eswa.2018.04.022.
[41] L. Morales, H. Lozada, J. Aguilar, E. Camargo, Applicability of LAMDA as classification model in the oil production, Artif. Intell. Rev. 53 (2020) 2207–2236, http://dx.doi.org/10.1007/s10462-019-09731-6.
[42] A.R.C. Isaza, J. Aguilar-Martin, M. LeLann, J. Aguilar, A. Rios-Bolivar, An optimization method for the data space partition obtained by classification techniques for the monitoring of dynamic processes, Artif. Intell. Res. Dev. (2006) 80–86, http://www.booksonline.iospress.nl/Content/View.aspx?piid=2087.
[43] G.C. Caicedo, Una Aplicación de la Técnica LAMDA a los Índices de Continuidad del Suministro de Energía Eléctrica, 2006.
[44] C. Allayous, S. Regis, A. Bruel, D. Schoevaert, R. Emilion, T. Marianne-Pepin, Velocity allowed red blood cell classification, IFAC Proc. 40 (2007) 375–380, http://dx.doi.org/10.3182/20070604-3-MX-2914.00064.
[45] I. Rish, An empirical study of the naive bayes classifier, in: IJCAI 2001 Work. Empir. Methods Artif. Intell. vol. 3, New York IBM, 2001, pp. 41–46, https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf.
[46] J.F. Botía, A.M. Cárdenas, C.M. Sierra, Fuzzy cellular automata and intuitionistic fuzzy sets applied to an optical frequency comb spectral shape, Eng. Appl. Artif. Intell. 62 (2017) 181–194, http://dx.doi.org/10.1016/j.engappai.2017.04.001.
[47] J.F. Botía, C. Isaza, T. Kempowsky, M.V. Le Lann, J. Aguilar-Martín, Automaton based on fuzzy clustering methods for monitoring industrial processes, Eng. Appl. Artif. Intell. 26 (2013) 1211–1220, http://dx.doi.org/10.1016/j.engappai.2012.11.003.
[48] L. Morales, J. Aguilar, A. Rosales, J.A.G. de Mesa, D. Chavez, An intelligent controller based on LAMDA, in: 2019 IEEE 4th Colomb. Conf. Autom. Control, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/CCAC.2019.8921299.
[49] L. Morales Escobar, J. Aguilar, A. Garces-Jimenez, J.A. Gutierrez De Mesa, J.M. Gomez-Pulido, Advanced fuzzy-logic-based context-driven control for HVAC management systems in buildings, IEEE Access. 8 (2020) 16111–16126, http://dx.doi.org/10.1109/ACCESS.2020.2966545.
[50] Jang, Neuro fuzzy model control.pdf, Proc. IEEE. 83 (1995) 378–406.
[51] S.J. Yoo, J.B. Park, Y.H. Choi, Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates, Inf. Sci. (NY) 177 (2007) 3074–3098, http://dx.doi.org/10.1016/j.ins.2007.02.009.
[52] J.R. Jang, ANFIS : Adap Tive-Ne Twork-Based Fuzzy Inference System, 1993, p. 23.
[53] J. Aguilar-Martín, N. López De Mantaras, The Process of Classification and Learning the Meaning of Linguistic Descriptors of Concepts, North-Holland Publishing Company, 1982.
[54] M.A. Shoorehdeli, M. Teshnehlab, A.K. Sedigh, M.A. Khanesar, Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods, Appl. Soft Comput. 9 (2009) 833–850, http://dx.doi.org/10.1016/j.asoc.2008.11.001.
[55] M.A. Shoorehdeli, M. Teshnehlab, A.K. Sedigh, Training ANFIS as an identifier with intelligent hybrid stable learning algorithm based on particle swarm optimization and extended Kalman filter, Fuzzy Sets and Systems 160 (2009) 922–948, http://dx.doi.org/10.1016/j.fss.2008.09.011.
[56] C. Paleologu, J. Benesty, S. Ciochiňa, A robust variable forgetting factor recursive least-squares algorithm for system identification, IEEE Signal Process. Lett. 15 (2008) 597–600, http://dx.doi.org/10.1109/LSP.2008.2001559.
[57] Y.H. Kim, S.C. Ahn, W.H. Kwon, Computational complexity of general fuzzy logic control and its simplification for a loop controller, Fuzzy Sets and Systems 111 (2000) 215–224, http://dx.doi.org/10.1016/S0165-0114(97)00409-0.
[58] C. Smith, A. Corripio, Principles and Practice of Automatic Process Control, second ed., John Wiley & Sons, Inc., 1997.
[59] O. Camacho, C.A. Smith, Sliding mode control: an approach to regulate nonlinear chemical processes, ISA Trans. 39 (2000) 205–218, http://dx.doi.org/10.1016/S0019-0578(99)00043-9.
[60] S. Soyguder, M. Karakose, H. Alli, Design and simulation of self-tuning PID-type fuzzy adaptive control for an expert HVAC system, Expert Syst. Appl. 36 (2009) 4566–4573, http://dx.doi.org/10.1016/j.eswa.2008.05.031.
[61] S. Soyguder, H. Alli, An expert system for the humidity and temperature control in HVAC systems using ANFIS and optimization with fuzzy modeling approach, Energy Build. 41 (2009) 814–822, http://dx.doi.org/10.1016/j.enbuild.2009.03.003.
[62] B. Arguello-Serrano, M. Velez-Reyes, Nonlinear control of a heating, ventilating, and air conditioning system with thermal load estimation, IEEE Trans. Control Syst. Technol. 7 (1999) 56–63, http://dx.doi.org/10.1109/87.736752.
[63] E. Bristol, On a new measure of interaction for multivariable process control, IEEE Trans. Automat. Control 11 (1966) 133–134, http://dx.doi.org/10.1109/TAC.1966.1098266.
[64] J.K. Pothal, D.R. Parhi, Navigation of multiple mobile robots in a highly clutter terrains using adaptive neuro-fuzzy inference system, Robot. Auton. Syst. 72 (2015) 48–58, http://dx.doi.org/10.1016/j.robot.2015.04.007.
[65] F.G. Rossomando, C.M. Soria, Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural PID, Neural Comput. Appl. 26 (2015) 1179–1191, http://dx.doi.org/10.1007/s00521-014-1805-8.
[66] Adept Technology Inc. Pioneer 3-DX, 2011, p. 2.
[67] E. Rohmer, S.P.N. Singh, M. Freese, V-REP: A versatile and scalable robot simulation framework, in: 2013 IEEE/RSJ Int. Conf. Intell. Robot. Syst., IEEE, 2013, pp. 1321–1326, http://dx.doi.org/10.1109/IROS.2013.6696520.