Artificial Neural Networks Models Based on ARX and State Space Forms and Adaptive Control PID/LQR of Systems Based on Peltier Cells

Denis F. Sousa de Sá[1] and João Viana da Fonseca Neto[2]

[1] Department of Electrical Engineering-Balsas, and PPGEE, Universidade Federal do Maranhão, Brazil

[2] Department of Electrical Engineering and PPGE, Universidade Federal do Maranhão

Abstract

To improve the performance of a thermal plant based on Peltier cell actuators, an online parametric estimation via artificial neural networks and adaptive controller is presented. The control actions are based on adaptive digital controller and an adaptive quadratic linear regulator approaches. The Artificial neural networks topology is based on ARX and NARX models, and its training algorithms, such as accelerated backpropagation and recursive least square. The Control strategies are design-oriented to adaptive digital PID controller and quadratic linear regulator framework. The proposal is evaluated on temperature control of an object that is inside of a chamber.

Keywords: Adaptive Control, Neural Networks, System Identification, Thermal System, Peltier Cell actuator, Control Design, LQR.

Introduction

The identification of Systems is of extreme importance for any study involving physical systems, [1], and even biological ones such as [2] and [3], which say that identification begins with simple human attitudes and [24], which says that it is the detection of behavioral patterns of physical systems. Thus, identifying is a natural practice of the complex system of the human body and began to be transcribed in a mathematical way to represent physical systems.

The need to represent these physical systems through mathematical models for applications such as control and prediction, for example, opened the door to several researches involving classical models and computational intelligence.

Then, the ARX (Auto-Regressive eXogenous inputs) model is the one that stands out most in this work, where it develops methods of identification and models based on computational intelligence to represent nonlinear systems. Many examples prove the importance of the use of computational intelligence in identification as in [4] that a radial base network is used to identify a nonlinear system with time variable learning, [5] that uses Computational intelligence to identify a nonlinear stationary system and in [6] that used a high order recurrent network for identification in a manipulator robot.

In this context, the optimization methods serve as a tool in the determination of these models, such as least squares (LS), recursive least squares (RLS) and gradient methods. In addition, there are two approaches: a *online*, that the identification is performed during the process evolution and *offline*, which is performed separately and subsequently used.

Together with identification and modeling, the control system is fundamental, and it is possible to use deterministic and adaptive techniques such as [7], where joining adaptive techniques with adaptive control can bring satisfactory results applied to non-linear systems.

Therefore, it is in these techniques that the development of this research is grounded, joining the search for algorithm improvements and the use of computational intelligence for parametric estimation applied to a nonlinear thermal system and using adaptive control as an application.

Artificial Neural Networks and Identification

The ANN's, which according to [8] is a technology that has as reference the biological neuron, is highly complex, non-linear and parallel, involves several applications such as in the identification and modeling of systems, Control [9], data classification [10] and associated with other technologies, such as Fuzzy for example presented in [11].

There are a variety of ways to use ANNs in the identification of systems and one of them is in the form of Fig.1, where θ^* is the parameters estimated by the network for a ARX structure, $U(t)$ the input, $y(t)$ the output e $y^*(t)$ the estimated output [12]. This form was chosen because it was the one that best suited to the system of adaptive control predicted as application.

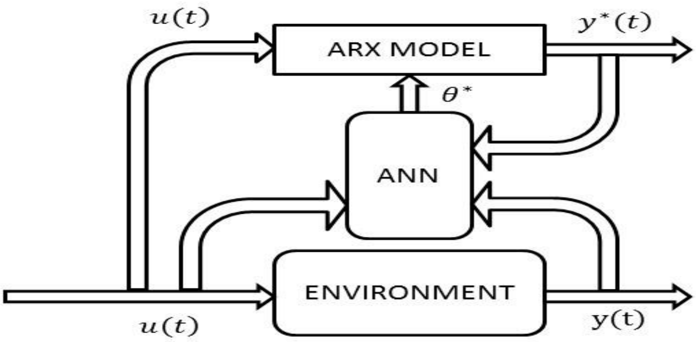

Figure 1 *Example of an ANN as parametric estimator*

There are, according to [8], a set of network structures suggested for systems identification processes, such as [13], [14], [15] and [16]. Among them, the networks are fed with signals delayed in time that seek to follow the functional structure of classic models like ARX for example. The Fig.2 presents a generalist structure of a network based on the ARX model.

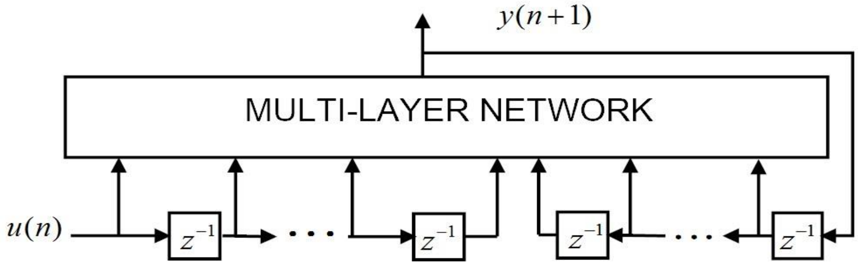

Figure 2-*ANN/ARX model from [8]*

This network represents an input/output model with feedback from output $y(n + 1)$ delayed in time and an input $u(n)$ which is also delayed in time, where φ is the activation function. Therefore, it is a recurrent multilayer network with $I^1(n)$ being the output signal of the first layer of neurons and $I^l(n)$ of the last layer, with $y(n+1) = I^1(n+1)$, the equations that relate the input to the output of the network are given by

$$\begin{aligned} I^1(n + 1) &= \varphi\ I^1(n),u(n)\ , \\ I\ (n + 1) &= \varphi\ I\ (n),I\ (n + 1)\ , \\ I^l(n + 1) &= \varphi(I^l(n),I^{l-1}(n + 1)). \end{aligned} \tag{1}$$

From this relation it is possible to define order models 1 to n for applications in systems identification.

Online Training of ANNs

The training of ANNs can be done in on-line and off-line forms, and the on-line method with some of this section is of interest in this work.

Gradient Method

The gradient method is based on the directional derivatives, where given $f(x,y)$, the directional derivative provides the rate of change of this function in the direction and sense of a unit vector U, given a point $P(x_0,y_0)$ in the xy plane [17]. Thus, is defined as i as the unit vector in the direction of the x-axis and j in the y-axis direction, and given $U = i + j$, the gradient is defined as the rate of change of $f(x,y)$, in the direction of U, where the gradient of $f(.)$ is given by

$$\nabla f(x,y) = f_x(x,y)i + f_y(x,y)j, \tag{2}$$

where $f_x(x,y)$ the derivative of $f(.)$ in relation to x and $f_y(x,y)$ in relation to y. The gradient vector given by Eq.(2) has direction and direction of maximum function growth. Thus, the gradient can be used both to solve problems involving maximization and minimization in the direction of maximum decay $-\nabla f(x,y)$, called descending methods.

This method can be associated to solve neural network training problems. Thus, in the case of the neural network [12], the goal is to minimize the mean square error given by

$$\varepsilon = \frac{1}{2}\sum_{n=0}^{p}\left(d(n)-y(n)\right)^2, \tag{3}$$

where $d(n)$ is the desired value and $y(n) = \varphi(\omega_k(n),x_i(n))$ the network output, with n integer. Thus, the learning, in the case of a neural network, based on the gradient method, deals with obtaining the optimal solution considering the minimization of the quadratic error ϵ, so that the weights vary in the sense of reducing this error, which consists of the update equation given by

$$\omega_k(n+1) = \omega_k(n) - \lambda \nabla \omega(\omega_k), \tag{4}$$

where λ is the learning rate and $g(n) = \nabla\, \omega_k\, \epsilon(\omega_k)$ is the gradient vector in the direction of ω_k.

According [8], this case is defined as an unrestricted optimization problem, where $\epsilon(\omega)$ is differentiable, having as solution ω^*, satisfying the condition $\epsilon(\omega*) \leq \epsilon(\omega)$, being that the gradient vector is given by

$$\nabla \epsilon(\omega) = \left[\frac{\partial \epsilon}{\partial \omega_1}, \quad \frac{\partial \epsilon}{\partial \omega_2}, \quad ,\cdots, \quad \frac{\partial \epsilon}{\partial \omega_n} \right]^T, \tag{5}$$

where the convergence condition is given by the condition $\epsilon(\omega(n+1)) < \epsilon(\omega(n))$. The updating of the weights given by Eq.(4) it's said of steep descent, serving as the basis for the various training methods in neural networks.

Accelerated Gradient Method (AG)

The acceleration method is based on Nesterov's studies, [18], whose application is restricted to problems of optimization of convex functions. Later, Saeed Ghadimi in [19], proposes, based on Nesterov that applies to convex and non-convex optimization problems. This method is also applied in predictive control, as in [19], which joins the acceleration of the gradient with the Lagrange duality principle. In this paper is use the method proposed in [20] together with the *Backpropagation* algorithm in an attempt to compensate for a larger number of operations with the reduction in the number of iterations, implying in the reduction of computational effort.

Therefore, it is a generic cost function, continuous and convex, having the following expression:

$$f^*(x) = \min_{x \in R^n} f(x), \tag{6}$$

In this case, is considered the condition of convexity of Lipschitz, given by

$$| f(x) - f(x') | \leq L | x - x' |, \tag{7}$$

with $L > 0$ and $\forall x, x' \in \mathrm{R}^n$. Considering $f(.)$ a convex function, is possible say that the gradient function is continuous and convex on the same criterion if

$$|| \nabla f(x) - \nabla f(x') || \leq L || x - x' ||. \tag{8}$$

Based on this theory, [18] shows that the degree of complexity in an algorithm using acceleration, given a solution $\bar{x}$ and a tolerance $f(\bar{x}) - f^* \leq \epsilon$, can be determined by $O\left(1/\sqrt{\epsilon}\right)$ while for the gradient descent it is given by $O(1/\epsilon)$. However, the Nesterov method does not apply to convex optimization when the number of samples is large.

As a development, [20] demonstrates the convergence and expansion of the accelerated method for any case, convex or not. In this case, a class of problems given by

$$\min_{x\in R^n} y(x)+h(x), \tag{9}$$

where $y(x) := f(x)+j(x)$ and $f(x) \in R^{n}$ is a function possibly not convex, $j(x) \in R^n$ a convex function and $h(x) = L_x(x)$ a convex function with limited domain, where $L_x(x)$ is an indicator function of a compact convex assembly $X \subset R^n$. This way, a limiting factor can be defined for each function by a Lipschitz constant L_y to $y(x)$, L_f to $f(x)$ and L_j to $j(x)$ where by the energy function is possible define $L_y = L_j + L_f$ as generalized limiting factor. Being $h(x)$ non-differentiable, [20] defines a new criterion based on the gradient mapping $G(.,.,.) = \nabla f(x)$ to analyze the complexity of the AG and shows that the policy, even if aggressive, presents a good convergence rate when finding the solution $\bar{x} \in R^n$ with degree of complexity given by

$$O\left\{\left(\frac{L_f^2}{\varepsilon}\right)^{1/3} + \frac{L_y L_f}{\varepsilon}\right\}. \tag{10}$$

Therefore, the AG method is presented to solve a class of non-linear optimization problems, convex and non-convex, where $y(x)$ is differentiable and limited inferiorly, presenting a greater comprehension with respect to the method of the gradient descent, as it presents the Algorithm 1.

Algorithm 1: Accelerated Gradient Algorithm (AG)

Data: $x_o \in R^n$, $\alpha \in (0,1)$ to any $k \geq 2$, $\beta > 0$ and $\lambda > 0$

initialization $x^{ag}_o = x_o$, $x^{md}_k = x_o$ e $k = 0$;

for $k \longleftarrow k + 1$ **do**

$$x_k^{md} = (1-\alpha)x_{k-1}^{ag} + \alpha x_{k-1}$$
$$x_k = x_{k-1} - \lambda \nabla y(x_k^{md})$$
$$x_k^{ag} = x_k^{md} - \beta \nabla y(x_k^{md})$$

end

This algorithm can be applied together with the learning method *backpropagation* doing $\beta = \lambda$, these being the adjustment parameters of the training and x_k^{md}, x_k, x_k^{ag} the weighted, normal and accelerated parameters, respectively.

Backpropagation Training

The *Backpropagation* is one the most used types of training in Artificial Neural Networks (ANN) that have several layers of neurons, as show in Fig.3.

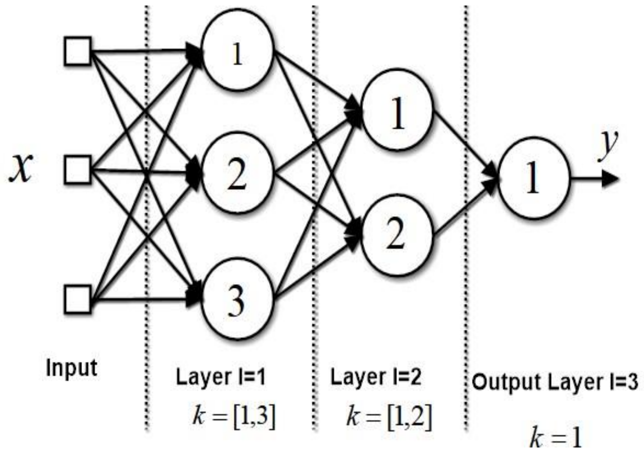

Figure 3- *Architecture of a multilayer network*

Therefore, let k be the neuron of layer l, with i inputs, having a synaptic weights defined by w_{ki}^{l} where I_k^l is the output this neuron, that represent a presynaptic activity of one neuron, is given by

$$I_k^1 = \sum_{i=1}^{n_x} w_{ki}^1 x_i \tag{11}$$

as being the pre-synapitc activity to the first layer of neurons

$$I_k^2 = \sum_{i=1}^{n_{c1}} w_{ki}^2 y_i^1, \tag{12}$$

as being the pre-synapitic activity of the hidden layer and,

$$I_k^3 = \sum_{i=1}^{n_{c2}} w_{ki}^3 y_i^2, \tag{13}$$

as being the pre-synapitic activity of the output layer, where n_x is the number of inputs, n_{c1} and n_{c2} the neurons number in the first and second layers with the neuron output given by $y_k^l = \varphi(I_k^l)$, where $\varphi(.)$ is the activation function. The weights are updated in the output layer towards input layer. For the output layer, the update of the i^{th} weight of the k neuron, referring the output y_k^2 of the hidden layer is given by

$$w_{ki}^3(n+1) = w_{ki}^3(n) + \eta \delta_k^3(n) y_k^2(n), \tag{14}$$

being η the learning factor, $w_{ki}^3(n)$, $w_{ki}^3(n+1)$ the past and current weights and the gradient, $\delta_k^3(n)$ given by

$$\delta_k^3(n) = (d(n) - y^3(n))\varphi'(I_k^3(n)), \tag{15}$$

with $d(n)$ being the target value and $y^3(n)$ the output of a ANN. Now, it's possible to determine the update of the weights for output layer and do the same for the other layers, determining

$$\delta_k^2(n) = \left(\sum_{j=1}^{n_{c3}} \delta_j^3(n) w_{ji}^3(n) \right) \varphi'(I_k^2(n)), \quad (16)$$

that is based on output layer gradient and in the postsynaptic activity of the layer itself, where the update on this layer is given by

$$w_{ki}^2(n+1) = w_{ki}^2(n) + \eta \delta_k^2(n) y_k^1(n). \quad (17)$$

To finish, on the input layer, where the gradient is based on gradient of the second layer

$$\delta_k^1(n) = \left(\sum_{j=1}^{n_{c3}} \delta_j^2(n) w_{ji}^2(n) \right) \varphi'(I_k^1(n)), \quad (18)$$

$$w_{ki}^1(n+1) = w_{ki}^1(n) + \eta \delta_k^1(n) x_i(n). \quad (19)$$

This method of training has the purpose the online learning of a ANN to a parametric identification applied to the thermal system and apply the gradient acceleration method.

Backpropagation Training

The adaptation of the classic backpropagation to its accelerated form, deals with the attempt to improve the speed of adaptation of the algorithm. The acceleration process has been used a lot in predictive control, as in [19]. In this work, the gradient acceleration method for the online type estimation process, applied in a thermal system based on the Peltier effect, being proposed the Accelerated Backpropagation (BPAG) training.

Applying the acceleration to update the synaptic weights for the output layer is present by

$$\begin{aligned} w_{k,i}^{md,3}(n+1) &= (1-\alpha) w_{k,i}^{ag,3}(n) + \alpha w_{k,i}^3(n), \\ w_{k,i}^3(n+1) &= w_{k,i}^3(n) - \lambda \delta_k^3(n) y_k^2(n), \\ w_{k,i}^{ag,3}(n+1) &= w_{k,i}^{md,3}(n+1) - \beta \delta_k^3(n) y_k^2(n). \end{aligned} \quad (20)$$

To the hidden layer:

$$\begin{aligned} w_{k,i}^{md,2}(n+1) &= (1-\alpha) w_{k,i}^{ag,2}(n) + \alpha w_{k,i}^2(n), \\ w_{k,i}^2(n+1) &= w_{k,i}^2(n) - \lambda \delta_k^2(n) y_k^1(n), \\ w_{k,i}^{ag,2}(n+1) &= w_{k,i}^{md,2}(n+1) - \beta \delta_k^2(n) y_k^1(n). \end{aligned} \quad (21)$$

Finally, to input layer:

$$\begin{aligned} w_{k,i}^{md,1}(n+1) &= (1-\alpha) w_{k,i}^{ag,1}(n) + \alpha w_{k,i}^1(n), \\ w_{k,i}^1(n+1) &= w_{k,i}^1(n) - \lambda \delta_k^1(n) x_i(n), \\ w_{k,i}^{ag,1}(n+1) &= w_{k,i}^{md,1}(n+1) - \beta \delta_k^1(n) x_i(n). \end{aligned} \quad (22)$$

This training is applied in the same way of traditional Backpropagation. Another method that be presented it's the Recursive Least Square (RLS).

Recursive Least Square (RLS)

This method is based on Least Squares (LS) [21], which has the objective of parametric estimation for models, minimizing the quadratic error between real and estimated value. The method is recursive with adaptive type approaches and with temporal references in t and $t-1$.

Based on this context, it's presented tows forms of RLS, like present [21]. First is presented the Standard RLS, where the update synaptic weight is given by:

$$\begin{aligned}&\hat{\omega}(n)=\hat{\omega}(n-1)-P(n)\varphi(n)\varphi^{T}(n)\hat{\omega}(n-1)+\\&+P(n)\varphi(n)y(n)\end{aligned} \tag{23}$$

being $P(n)$ the covariance matrix, given by $\varphi^T(n)\varphi(n)$, with $\varphi(n)$ the data matrix and $y(n)$ the real output where

$$\begin{aligned}&\hat{\omega}(n)=\hat{\omega}(n-1)+P(n)\varphi(n)\\&\left(y(n)-\varphi^{T}(n)\hat{\omega}(n-1)\right),\end{aligned} \tag{24}$$

with $\varepsilon(n)=y(n)-\varphi^{T}(n)\hat{\theta}(n-1)$ the error between the output and the target and $K(n) = P(n)\varphi(n)$, $P(n) = (\varphi^T(n)\varphi(n))^{-1}$ and $\varphi(n)$ the data vectors. The recursive equation e given by

$$\hat{\omega}(n)=\hat{\omega}(n-1)+K(n)\varepsilon(n). \tag{25}$$

The second RLS form, cited by [21], applies drastic variations on parameters with a few frequency in reason of a big step $K(n)$. For this case the average quadratic error is given by

$$V(\omega,n)=\frac{1}{2}\sum_{i=1}^{n}\lambda^{n-i}\left(y(n)-\varphi^{T}(n)\omega(n)\right)^{2} \tag{26}$$

being λ call forgetting factor. The RLS formulation is given by:

$$\begin{aligned}&\omega(n)=\omega(n-1)+K(n)\left(y(n)-\varphi^{T}(n)\omega(n-1)\right),\\&K(n)=P(n)\varphi(n)\\&=P(n-1)\varphi(n)\left(\lambda+\varphi^{T}(t)P(n-1)\varphi(n)\right)^{-1},\\&P(n)=\left(I-K(n)\varphi^{T}(n)\right)P(n-1)/\lambda,\end{aligned} \tag{27}$$

where the first formulation on Eq. (27) is the update of the synaptic weights, the second the update of $K(n)$, and the last the covariance matrix.

Thermal System

The thermal system in use has the capacity to heat ans cool small objects or environments, which serves as the basis for the lifting of models and temperature control.

On the Figure 4 is possible see the system compose to a thermal box of Styrofoam, temperature sensors, where the sensor 1 measure the internal temperature of the box and the second on object. The Peltier cell is the thermal actuator with non-linear characteristics, Eq. (28) [22], where P_h is the power supplied, I the current, R_m the internal resistance, $\alpha_{p,n}$ the Seeback coefficient and T_h the temperature on the hot side.

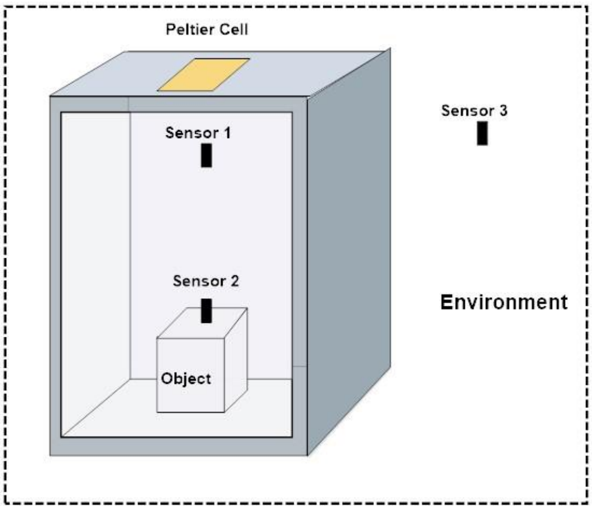

Figure 4- *Thermal Chamber*

$$P_h = \frac{I^2 R_m}{2} + \alpha_{p,n} T_h I. \tag{28}$$

System characteristics: Model cell TEC1-1209, Thermal chamber of 17*cm*x25*cm and a* Cubic Aluminum object of edge 2,5*cm*.

The differential equations based on thermal transfer lows are given by

$$m_o c_o \frac{dT_o}{dt} = K_{in,o}(T_{in} - T_o), \tag{29}$$

$$m_{in} c_{in} \frac{dT_{in}}{dt} = K_{in,o}(T_o - T_{in}) \tag{30}$$

where m_o and m_{in} are the mass the object and the internal air, c_o and c_{in} the specific heat, $K_{in,o}$ the thermal conductivity of internal air to object, $K_{in,ext}$ the lost to external environment, T_o, T_{in} and T_{ext} the temperatures of the object, internal air and external air, u_{celula} the amount of heat supplied by the actuator related with Eq.(28).

Intelligent Online Estimation

It's presented two structures on this section of ANN's based on State Space and ARX Modeling to the thermal system and use the Backpropagation, Accelerated Backpropagation and the RLS to online training the ANN's.

Recursive Least Square (RLS)

The first structure presented is the based on ARX modeling, called here of ANN/ARX, how is represented in Fig.5.

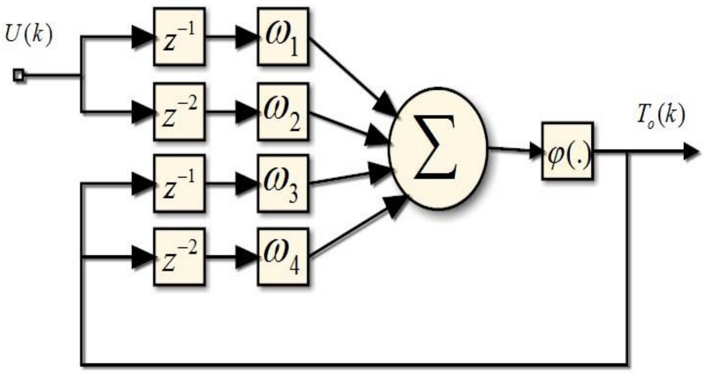

Figure 5- *ANN/ARX Model Structure*

The second structure is based on stat space modeling, represented by Fig.6.

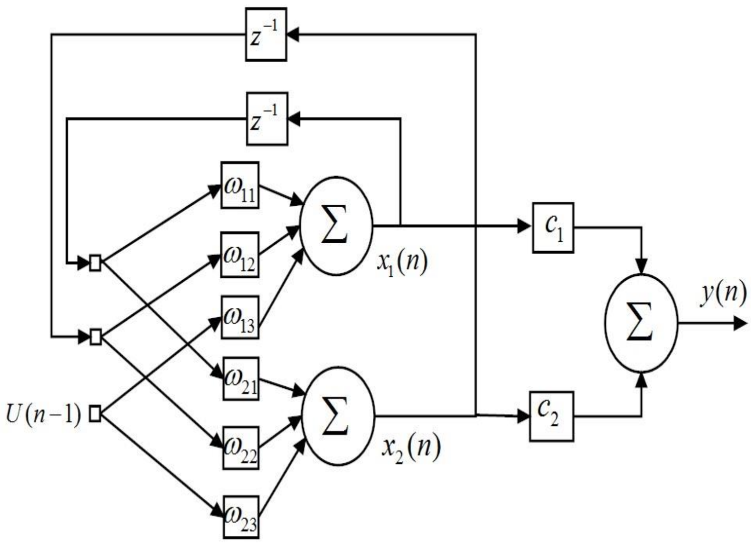

Figure 6- *ANN/State Space structure*

Estimation with Backpropagation

Is presented here the parameter estimations using Backpropagation, with a stopping criterion of ±0.01.

- ANN/ARX case

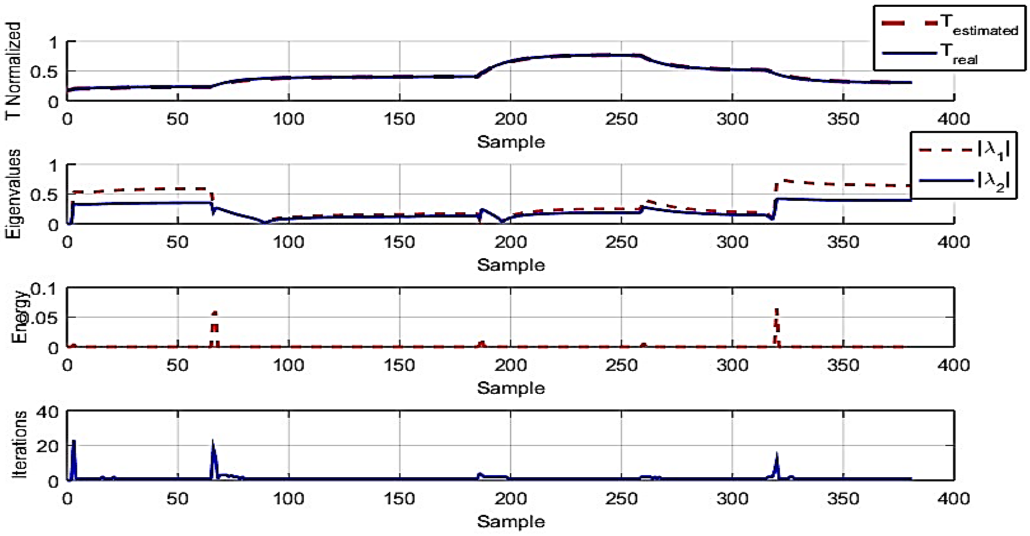

Figure 7- *Object temperature, Training BP - ANN/ARX*

Based on Fig. 7, the algorithm presented a good precision, with synaptic weights unsaturated, eigenvalues stables and more approximate of zero related the others cases studied here ans a low energy.

- ANNState Space case

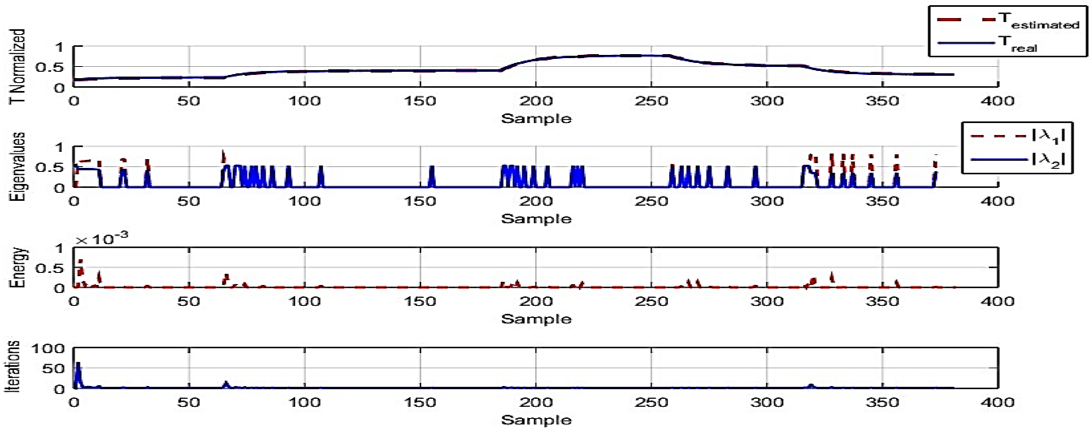

Figure 8- *Object temperature, Training BP - ANN/State Space*

In observation to Fig.8, the algorithm present a good answer, with error inside of determined margin, but present many parameters oscillation along all the process, even presenting a synaptic weight unsaturated.

BP-AG Estimation

To Accelerated Backpropagation, the same criterion of stopping error was used.

- ANN/ARX case

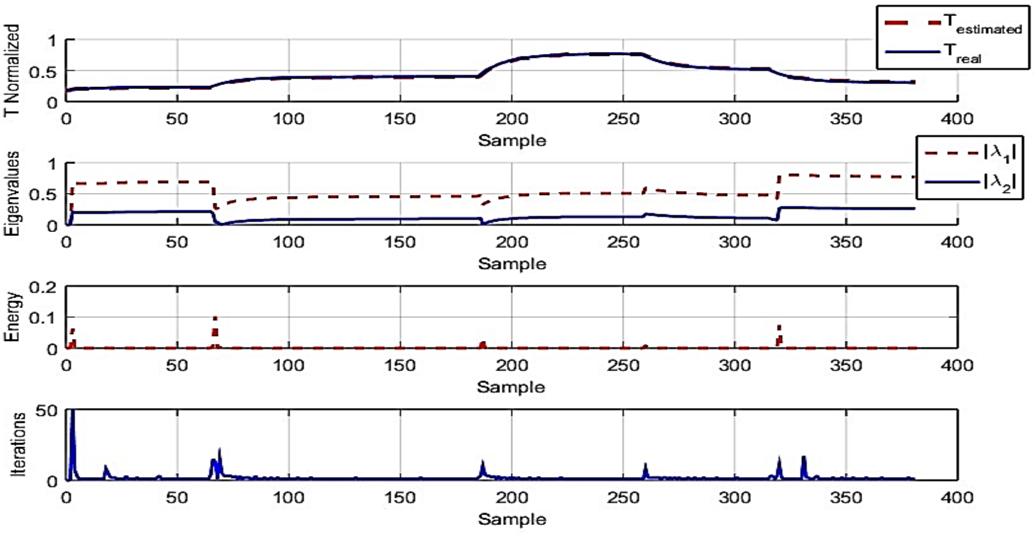

Figure 9- *Object temperature, Training BP-AG - ANN/ARX*

To the case of Fig.9, the response was considered good, don't hasn't many deformations related to the target of temperature, eigenvalues stables and error in the admissible range.

- ANN/State Space case

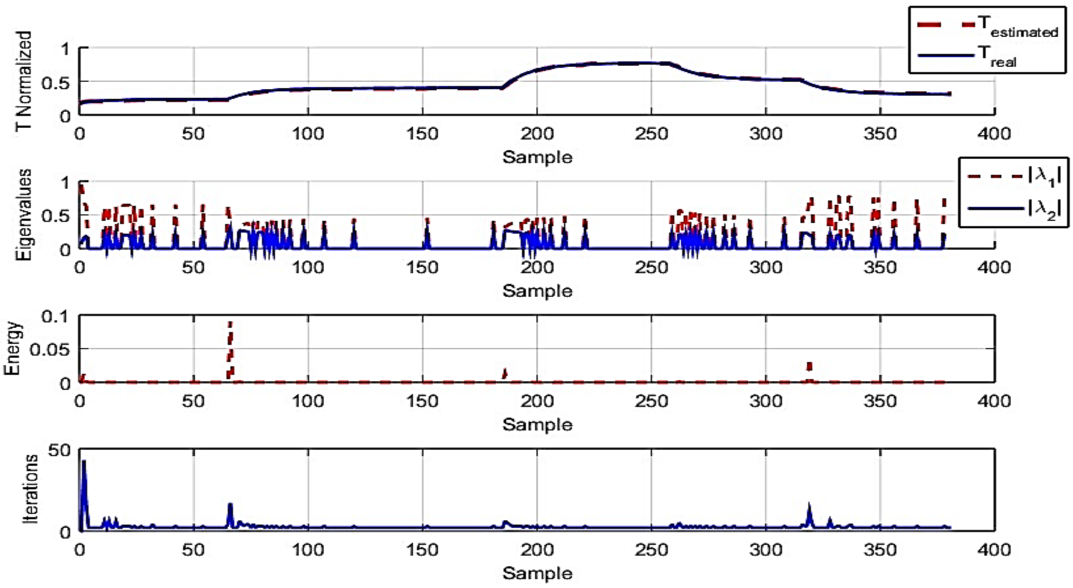

Figure 10 - *Object temperature, Training BP-AG - ANN/State Space*

In case of Fig.10, the behavior was the same that the others and the ARX model showing a not saturation of eigenvalues but with great oscillations and the energy has few peaks with amplitudes similar to the cases already studied.

RLS Estimation

Using the same criterion related the others cases, in this section is presented the RLS studies.

- ANN/ARX case

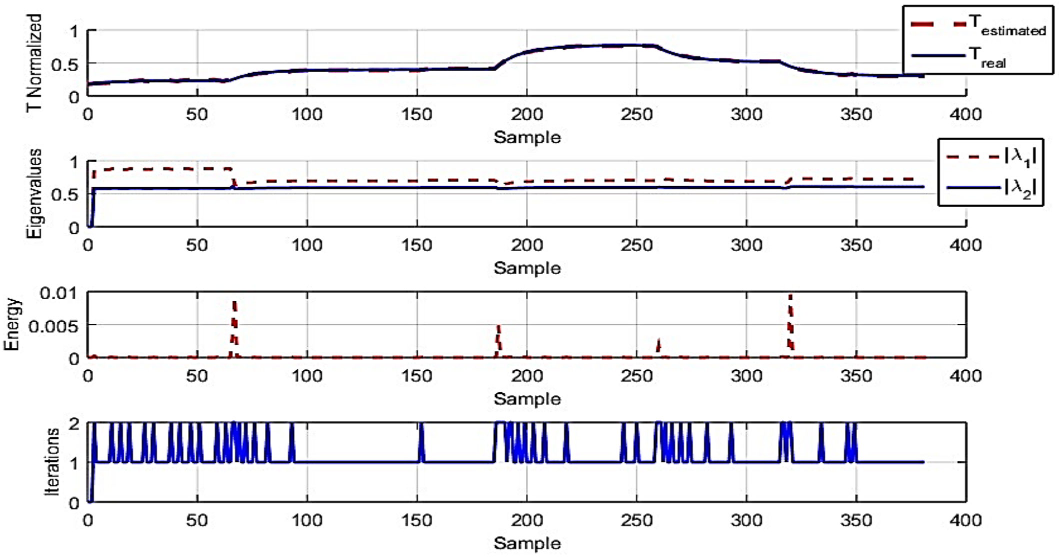

Figure 11- *Object temperature, RLS training - ANN/ARX*

In the case of Fig.11, the answer was considered good, with eigenvalues not saturation and stables for evaluated process. The energy is in expected range and there were no large peaks of iterations per sample, being a fast estimate.

- ANN/State Space case

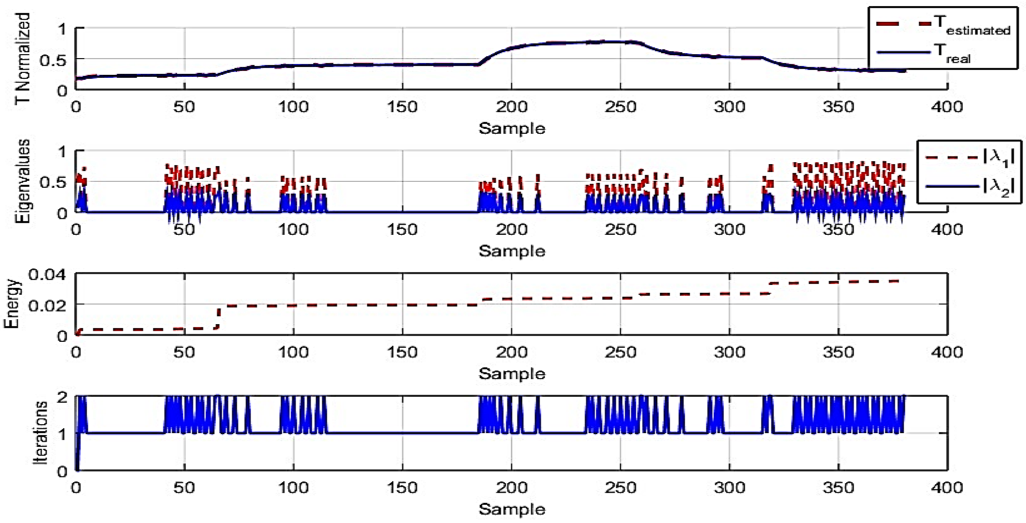

Figure 12- *Object temperature, RLS training - ANN/ARX*

The case of Fig.12 is the last, presenting great oscillation of the synaptic weights, eigenvalues and increase of the energy.

Methods evaluation

Is presented here an explanation of the methods used, to ANN/ARX and ANN/State Space. five tests were performed for each algorithm, analyzing the total energy accumulated in process learning, the maximum energy peaks, the total energy means, the mean of iterations peaks for each data of training, the mean iteration far all process learning, mean time execution in each test and the maximum time the ANN used to learn one simple data.

Are presented two Tables I and V-E presenting the best case, the worst case and the means of all five case tested.

Table I- *Algorithm analysis to case ANN/ARX*

	Iteration mean	Iteration max	Time mean (μs)	Time max (ms)	Energ. max	Total Energ.
Algorithm			**BP**			
worst case	1	21	63	5.1	0.2851	0.3528
best case	1	64	63.5	6.2	0.0757	0.1570
Mean	1	48,6	63.3	5.8	0.1397	0.2648
Algorithm			**BP-AG**			
worst case	1	60	32.7	8.5	0.2096	0.4647
Best case	1	19	32.2	2.9	0.028	0.094
Mean	1	45	32.5	6.7	0.1096	0.2922
Algorithm			**RLS**			
Worst case	1	2	33.7	4.8	0.0406	0.1486
Best case	1	2	32.7	2.8	0.002	0.0104
Mean	1	2	33.1	3.8	0.0194	0.0684

Observing the Table I is possible verify a homogeneity of iterations mean in each test and differences between the maximum of iterations on each test, where the BP presented the best results for initial parts of training. The BP-AG, presented a smaller peaks of iterations to each data training and mean of iterations. But, in this requirement the RLS was the best.

Analyzing the mean time to ANN learn one sample data, the BP has a great vantage related to BP-AG and RLS with the BP-AG better than RLS. Regarding the worst case in convergence time for all test, the algorithms had close performances.

To finish, in energy factor or square error, the great peaks of energy was from BP with the BP-AG showing a better results and the RLS was considered the best for this analyze.

To case of ANN in Stat Space form, analyzing the Table II, the algorithms RLS and BP presented the best results related to iterations mean. Related to the peak iterations to learn one sample data the BP-AG was the worse with similar results in comparison to ANN/ARX and the RLS presented a more satisfactory result.

In temporal terms, to the mean time of learn, all algorithms were very close. Related to maximum time to learn a sample data, the RLS was the most satisfactory.

To finish, analyzing the energy, the BP and BP-AG were the best for all training related to RLS.

Table II- *Algorithm analysis to case ANN/State Space*

	Iteration mean	**Iteration max**	**Time mean (μs)**	**Time max (ms)**	**Energ. max**	**Total energy.**
Algorithm			BP			
worse case	1	68	8.31	1.7	0.2073	0.2131
Best case	1	27	7.82	0.69	0.00046	0.0053
Mean	1	48.4	8.11	6.2	0.0460	0.0054
Algorithm			BP-AG			
Worse case	2	124	9.53	2	0.1162	0.2711
Best case	2	36	8.31	0.68	0.0287	0.0492
Mean	2	45	8.75	1.3	0.0756	0.1542
Algorithm			RLS			
Worse case	1	2	8.31	0.14	0.0462	11.44
Best case	1	2	7.82	0.082	0.007	1.505
Mean	1	2	81.6	0.52	0.0217	5.684

Adaptive Control

On this section is presented two strategies of control using the ANN-ARX (PID control) and ANN-State Space (DLQR).

Digital Adaptive PID

The first structure [7] proposal is presented in Figure 13.

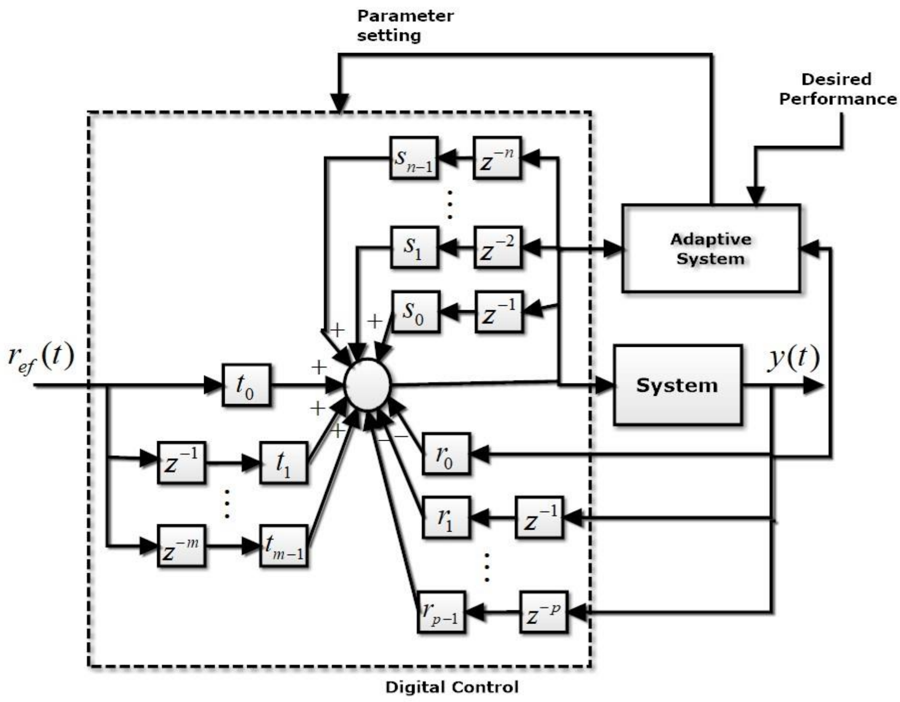

Figure 13 *General structure to digital control, based on [7].*

The equation of discrete control proposed by [7] and associated with Figure 13 is given by

$$H(z^{-1}) = K\left[1 + \frac{T_s}{T_i}\frac{1}{1-z^{-1}} + \frac{\frac{NT_s}{T_d + NT_s}\left(1-z^{-1}\right)}{1 - \frac{T_d}{T_d + NT_s}z^{-1}}\right], \tag{31}$$

being T_s, T_i, T_d e N, the time sample, timing integration, to derivative, and derivative filter. The form presented by Fig. 13 is call RST controller and the PID is obtained by association:

$$s_1 = -\frac{T_d}{T_d + NT_s}, \tag{32}$$

$$r_0 = K\left(1 + \frac{T_s}{T_i} - N\frac{T_s}{T_d}s_1\right), \tag{33}$$

$$r_1 = K\left[s_1\left(1 + \frac{T_s}{T_i} + 2N\frac{T_s}{T_d}\right) - 1\right], \tag{34}$$

$$r_2 = -Ks_1\left(1 + N\frac{T_s}{T_d}\right), \tag{35}$$

where the controller is given by

$$R(z^{-1}) = r_0 + r_1 z^{-1} + r_2 z^{-2}, \tag{36}$$

$$S(z^{-1}) = \left(1 - z^{-1}\right)\left(1 + s_1 z^{-1}\right). \tag{37}$$

That structure is associated to a target equation given by Eq.(38) and can be applied to ANN-ARX.

$$H(s)=\frac{b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}=\frac{B(z^{-1})}{A(z^{-1})}. \tag{38}$$

Take like reference a dynamic target given by

$$\begin{aligned}P(z^{-1})&=1+p_1z^{-1}+p_2z^{-2}+p_3z^{-3}+p_4z^{-4}\\&=\left(1+p_1{}'z^{-1}+p_2{}'z^{-2}\right)\left(1+\alpha_1z^{-1}\right)\left(1+\alpha_2z^{-1}\right),\end{aligned} \tag{39}$$

being p_1' and p_2' the basic parameters of second order of system and α_1, α_2 the auxiliary poles where to have no influence on dynamic, [7] define $\alpha_1 = \alpha_2$ or $\alpha_2 = 0$ in a range $-0{,}05 \le \alpha_1,\alpha_2 \le -0{,}5$. Now, is possible determinate the equations given by

$$x^T=\left[1,s_1,r_0,r_1,r_2\right], \tag{40}$$

$$p^T=\left[1,p_1,p_2,p_3,p_4\right], \tag{41}$$

$$M=\begin{bmatrix}1&0&0&0&0\\a_1'&1&b_1&0&0\\a_2'&a_1'&b_2&b_1&0\\a_3'&a_2'&0&b_2&b_1\\0&a_3'&0&0&b_2\end{bmatrix}, \tag{42}$$

being $a_1'=a_1-1$, $a_2'=a_2-a_1$ and $a_3'=-a_2$, where the solution is $x=M^{-1}p$, that associate with the ANNARX the synaptic weights are $w(0) = -a_1$, $w(1) = -a2$, $w(2) = b_1$ and $w(3) = b_2$. The algorithm is given by

Algorithm 2: PID estimator

Input: Synaptic weights w_k, Target dynamic $gd(n)$, to $k = [0,3]$ with $n = [0,4]$ and $gd(0) = 1$

$$a_1=-w(0)-1$$
$$a_2=w(0)-w(0)$$

Determinant of M: $det=w^3(4)-(a_1w^2(3)+w(1)w^2(2)-a_2w(3)w(2))w(2);$

Calculate:

$$\begin{aligned}&s_1=(gd(1)w^3(3)-w(2)gd(2)w^2(3)+w(3*gd(3)w^2(2)-gd(4)w^3(2))/det;\\&r_2=(-w(1)s_1+gd(4))/w(3);\\&r_1=(-a_2s_1-w(2)r_2+gd(3))/w(3);\\&r_0=(-a_1s_1-w(2)r_1+gd(2))/w(3);\end{aligned}$$

Like application is presented the Fig.14, that show the Control Effort, ANN answer and the target dynamic. The reference used was 70oC to a dynamic given by τ = 300s and ζ = 0.8.

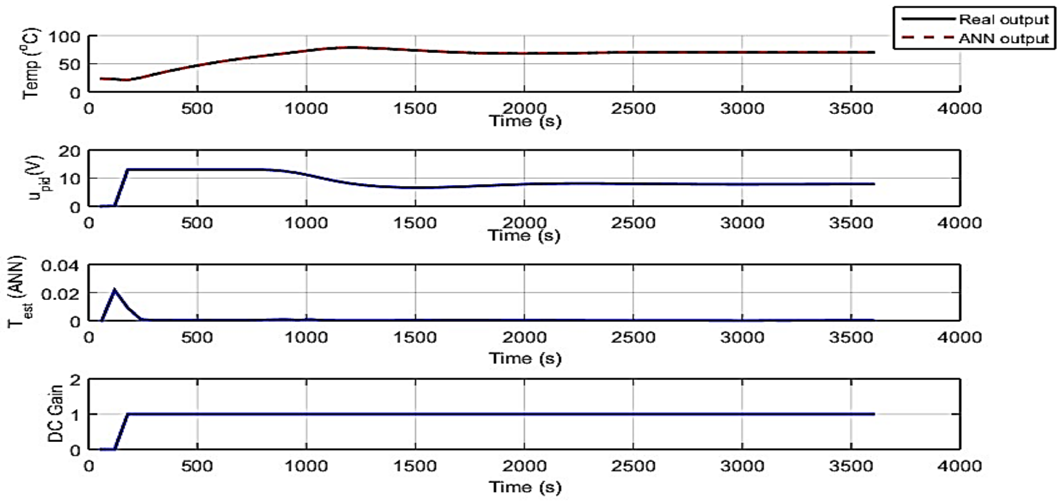

Figure 14 - Digital PID with τ = 300, ζ = 0.8 and Tref = 70

It's possible observe that the controller start actuate in T = 180s. From this point the time constant obtained was τ = 360s, considered a good answer related to target. The performance is presented in Table III, where τ is the constant time, ζ the damping factor, Tref, Toutput, the target of temperature and output of system.

Table III *Performance of Digital PID adaptive*

Dynamic target			**Dynamic result**		
τ	*ζ*	*Tref*	*τ*	*Mp*	*Toutput*
300	0,8	70	360	78	70,38
400	0,8	70	390	78	70,29
500 a 700	0,8	70	390	76	70,01
1000	0,8	70	420	73	69,6 a 71,6
300 a 1200	0,8	95	720	99	94,7
1300	0,8	95	1110	98	95,21

Analyzing the Table III, was verified there is a range that the system not obtained the dynamic target and for *τ* between 500 and 700 seconds had a best result in terms of precision related result in *τ* = 400*s*. When the time constant is large the composting of the controller is better.

Optimal Control DLQR

The Discrete Linear Quadratic Regulator (DLQR) it's a classic optimal control presented in [23] and is based on optimal theory. Based on this context the Fig. 15 present the schematic diagram to adaptive DLQR like application to ANN in state space format.

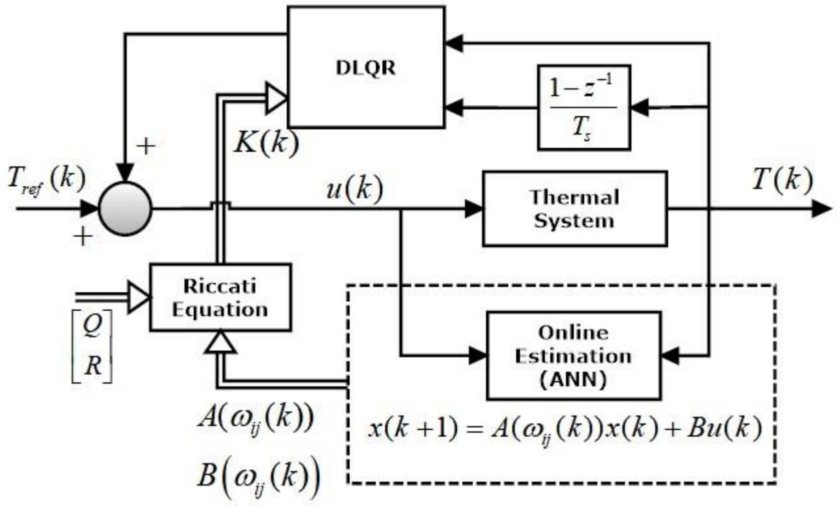

Figure 15- *Adaptive DLQR structure*

On this case, the online estimation provide the matrix $A(k)$ and $B(k)$ composed by the ANN weights. The matrix parameters of DLQR, Q and R provide by user, given the control low by Eq. (43), where the iterative method of Riccati given by Equations (46) and (46) determine the $K(k)$ weights of the DLQR. This structure represents the control law given by

$$u(k) = T_{ref}(k) - kx(k), \tag{43}$$

that optimize the cost equation and solution, given by

$$J = \frac{1}{2}\sum_{k=1}^{N} x_k^T Q_k x_k + u_k^T R_k u_k. \tag{44}$$

$$-A_k^T P_{k+1} B_k (R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k + A_k^T P_{k+1} A_k - P_k + Q_k = 0. \tag{45}$$

$$K(k) = (R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k. \tag{46}$$

Since the DLQR is used as parameters $R = 1$ and $Q = I_{2,2}$ with a temperature target of $T_{ref} = 0^oC$, since DLQR is a regulator, Fig.16.

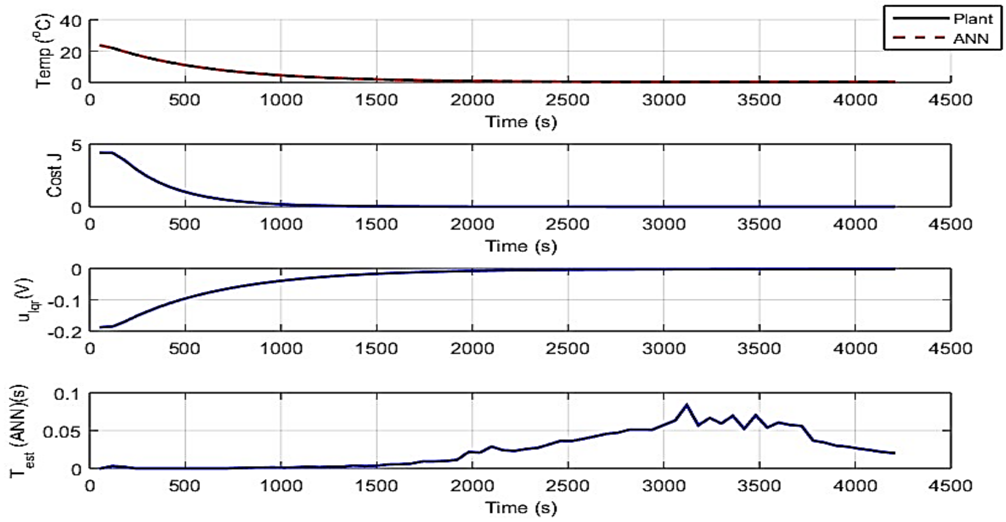

Figure 16- *Adaptive DLQR applied to the system*

The DLQR can be transformed in a controller if associate a PI structure, Fig. 17. Therefore, using trial and error method, was determined kp = 0,1 and ki = 0,15. Now, it's possible verify the Q and R dynamic effect. Being the temperature reference Tref = 80oC, was obtained τ = 545s and Tsaida = 77,2oC like result. The cost J was bigger related to DLRQ pure and the final result was considered good.

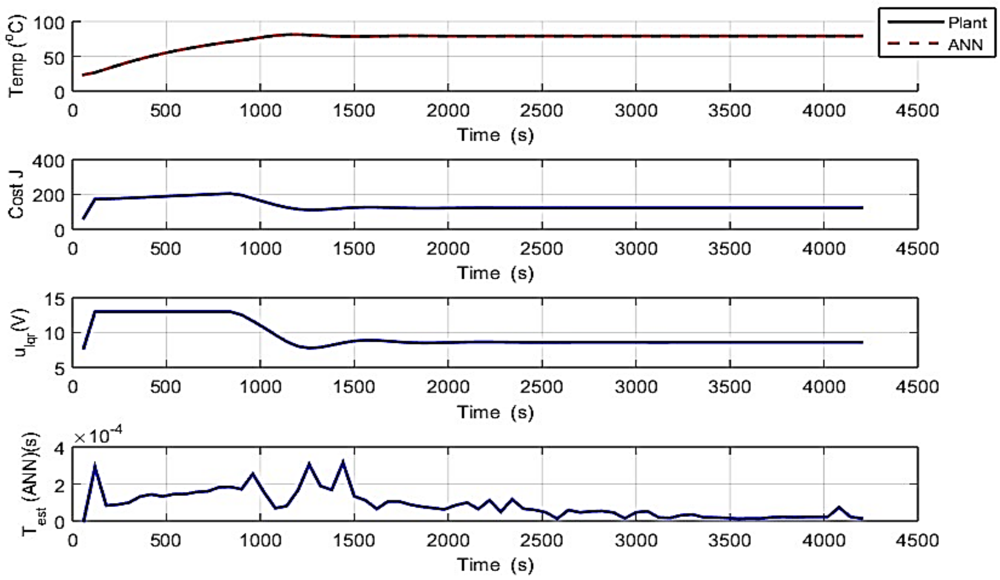

Figure 17- *Adaptive DLQR with PI controller*

To evaluate the DLQR in both situations is presented the Table considering to DLQR pure a convergence temperature value of $0,3^oC$:

Inputs Q and R		Performance DLQR		Performance DLQR/PI			
$Q = aI$ a	$R = bI$ b	Mean J	Convergence (s)	$Treg\ ^0C$	$Mp\ ^0C$	τ (s)	Mean J
3	3	0,835	3060	77,78	79,83	560	382
1	3	0,360	3660	76,53	78,55	560	292
0,1	3	0,035	2940	77,31	79,28	560	259
3	1	0,823	3060	77,75	79,79	560	213
1	1	0.304	3900	77,78	79,82	560	128
0,1	1	0,033	4500	76,43	78,56	560	87
3	0,1	0,933	4080	76,63	78,39	560	132
1	0,1	0,306	3960	75,35	76,99	560	48
0,1	0,1	0,029	3360	77,75	79,78	560	12

Analyzing the Table IV is verified variations in convergence time of DLQR pure. When R is big and Q reduce, the cost J reduce and the convergence time change with a peak and after with a reduction. To intermediary values to R and Q, J reduce and the convergence increase.

Related to DLQR with PI, the steady state for all situations were very close. The peaks values and τ were constant. The only change was when the a and b values reduced make $Q = 0,1 * I$ and $R = 0,1$ that the cost decreased sharply.

Conclusion

On this work was presented the study and development of algorithms and online estimations using ANN's and some training methods.

A study on neural networks was carried out, where some training methods and structures were presented for the identification and modeling of systems, together with two methodologies of system identification, where in one the network represented the model itself and in another it played the role of parametric estimator.

Later, a study of the optimal methods for neural network training was presented, where the gradient method, Backpropagation and nonlinear systems solution methods were highlighted. It was also presented the proposal of gradient acceleration that combined with the classic BP was obtained the BP with gradient acceleration denominated of BP-AG and more the RLS, that were used in the intelligent estimation.

Given the optimal algorithms, the intelligent online estimation was performed on two proposed network structures, one in the ARX form and the other in the state space, where the algorithms were analyzed through the time of convergence and iteration by point of operation, and the BP -AG presented no relative advantages over classical BP and RLS.

In order to evaluate the algorithms more clearly, these were implemented in C language and embedded in a microcontroller and along with this were presented two approaches of adaptive control. One using a generalized digital adaptive PID with filtering in the derivative factor and another using LQR with Riccati recurrence. The adaptive digital PID was emulated in the microcontroller and the performance obtained in a closed loop was evaluated and compared to the desired one, obtaining good results. In this context, the three algorithms were evaluated in terms of the amount of operations expended where classical BP had the greatest advantage. It was also presented the programming structure implemented in the chip where all process was simulated using a circuit simulator that integrates electronic devices and systems models.

Therefore, the results were satisfactory, where the work presented all the proposed steps, presented from the methods with computational intelligence, evaluating the system and applying the methods in classic adaptive and optimal control.

Acknowledgment

We thanked to the Federal University of Maranhão (UFMA) and the Graduate Program in Electrical Engineering (PPGEE/UFMA) to support the development of this scientific and technical research. We are especially grateful to FAPEMA for encouraging high-level research in the State of Maranhão. Finally, we thank CAPES and CNPq for always promoting and supporting advanced research that contributed to this work. References

References

[1] F. Pasqualetti, F. Dorfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 11, pp. 2715–2729, 2013.

[2] L.-Z. Liu, F.-X. Wu, L.-L. Han, and W. Zhang, "Structure identification and parameter estimation of biological s-systems," in *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*, 2010, pp. 329–334.

[3] P. Z. Marmarelis and K.-i. Naka, "Identification of multi-input biological systems," *Biomedical Engineering, IEEE Transactions on*, vol. BME-21, no. 2, pp. 88–101, 1974.

[4] C.-N. Ko,"Identification of non-linear systems using radial basis function neural networks with time-varying learning algorithm," *Signal Processing, IET*, vol. 6, no. 2, pp. 91–98, 2012.

[5] C. Turchetti, F. Gianfelici, G. Biagetti, and P. Crippa, "A computational intelligence technique for the identification of non-linear non-stationary systems," in Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, 2008, pp. 3034–3038.

[6] F. Jurado, M. Flores, V. Santibanez, M. Llama, and C. Castaneda,"Continuous-time neural identification for a 2 dof vertical robot manipulator,"in *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE*, Nov 2011, pp.
77–82.

[7] I. D. Landau, Digital Control Systems, A. J. E.D.Sontag, M.Thoma, Ed. Springer, 1938.

[8] S. Haykin, *Redes Neurais - 2ed.* BOOKMAN COMPANHIA ED, 2001.

[9] Xiao, F. Long, and Y. Zhao,“Based on elm forged neural control for a class of strict feedback stochastic nonlinear switched system with time varying delay,”in *Control Conference (CCC), 2013 32nd Chinese*, 2013, pp. 2168–2173.

[10] S. Nirkhi, “Potential use of artificial neural network in data mining,” in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, vol. 2, 2010, pp. 339–343.

[11] I. Baruch, S. Hernandez, S. Echeverria, and O. Castillo,“Decentralized direct and indirect i-term adaptive fuzzy-neural control of a bioprocess plant,” in *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, 2012, pp. 1–8.

[12] S. Haykin, *Neural Networks, A Comprehensive Foundation*, 2nd ed. Prentice Hall International, 1999.

[13] J. R. Raol and H. Madhuranath,“Neural network architectures for parameter estimation of dynamical systems,”*Control Theory and Applications, IEE Proceedings* -, vol. 143, no. 4, pp. 387– 394, Jul 1996.

[14] R. Bharadwaj, A. Parlos, and H. Toliyat, “Adaptive neural network-based state filter for induction motor speed estimation,” in *Industrial Electronics Society, 1999. IECON ’99 Proceedings. The 25th Annual Conference of the IEEE*, vol. 3, 1999, pp. 1283– 1288 vol.3.

[15] J. V. G. Pezzotti and Londono, “Discrete deadbeat control of a plant pressure through identification by neural networks,”*IEEE Laton America Transactions*, 2012.

[16] A. Janczak, *Identification of Nonlinear Systems Using Neural Networks and Polynomial Models: A Block-Oriented Approach*, ser. Lecture Notes in Control and Information Sciences. Springer, 2004.

[17] L. Leithold, *O ca´lculo com geometria anal´ıtica*, 3rd ed. Harbra, 1994, no. v. 2.

[18] Y. Nesterov, “A method of solving a convex programming problem with convergence rate o (1/k2),” *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[19] P. Patrinos and A. Bemporad, “An accelerated dual gradientprojection algorithm for embedded linear model predictive control,” *Automatic Control, IEEE Transactions on*, vol. 59, no. 1, pp. 18–33, Jan 2014.

[20] S. Ghadimi and G. Lan,“Accelerated gradient methods for nonconvex nonlinear and stochastic programming,” *arXiv preprint arXiv:1310.3787*, 2013.

[21] K. Astrom and B. Wittenmark, *Adaptive Control: Second Edition*, ser. Dover Books on Electrical Engineering. Dover Publications, 2008.

[22] L. d. A. H. N. A.M.N. Lima, G.S. Deep and M. Fontana, “A gain-scheduling pid-like controller for peltier-based thermal hysteresis characterization platform,” *IEEE Instrumentation and Measurement Technology Conference*, vol. may, 2001.

[23] F. L. Lewis, *Applied Optimal Control & Estimation - Digital Design & Implementation*, 1st ed., ser. Digital signal Processing.New Jersey: Prentice Hall, 1992, vol. 17.

[24] M. Norgaard at all, Neural Networks for modelling and Control of Dynamic Systems, Springer London, 2000.

[25] Avery, R. J., Bryant, W. K., Mathios, A., Kang, H., & Bell, D. (2006). Electronic course evaluations: Does an online delivery system influence student evaluations? *The Journal of Economic Education*, *37*(1), 21–37. https://doi.org/10.3200/JECE.37.1.21-37

[26] Berk, R. A. (2012). Top 20 strategies to increase the online response rates of student rating scales. *International Journal of Technology in Teaching and Learning*, *8*(2), 98–107.