Self-Tuning Neural Controller and its Application to a Non Linear System.

Alfonso Gomez[1], Alfonso Noriega[1], Alberto Aguado[2], and Roberto Salas[3]

[1] Facultad de Ingeniería, Universidad Autónoma de Querétaro, Querétaro, Qro.., México
[2] ICIMAF, La Habana, Cuba
[3] MIDE, Querétaro, Qro., México
{Alfonso Noriega, anoriega@uaq.mx}

Abstract - The performance of an industrial process control system equipped with a conventional controller may be degraded severely by a long system-time delay, dead zone and/or saturation of actuator mechanisms, model and/or parameter uncertainties, and process noises. Recently it has been demonstrated that Artificial Neural Networks can be used to control Non Linear Systems and also it was proposed that a Self-tuning Neural Network implementation is possible with a real time permanent adjustment of the weighting coefficients based on the control error. In the present paper, the Self-tuning Neural Network practical implementation results in a Non Linear Systems are shown and a methodology to adjust the parameters is proposed. As part of the proposal we present a practical method to accelerate net learning by bounding neuron outputs and a time variable learning coefficient.

Keywords: *Neural Networks, Back-propagation, Non Linear Systems, Adaptive Control.*

I. Introduction

The best developed aspect of the mathematical systems theory, deals with the analysis and synthesis of dynamical systems, using well established techniques based on linear algebra, complex variable theory, and the theory of ordinary linear differential equations. Since design techniques for dynamical systems are closely related to their stability properties and since necessary and sufficient conditions for the stability of linear time-invariant systems have been generated, well-known design methods have been established for such systems. In contrast to this, the stability of nonlinear systems can be established for the most part only on a system-by system basis and hence it is not surprising that design procedures that simultaneously meet the requirements of stability, robustness, and good dynamical response are not currently available for large classes of such systems.

Control, Virtual Instrumentation and Digital Systems.
Research in Computing Science 24, 2006, pp. 199-211

Narendra and Parthasarathy (1990), demonstrated that neural networks can be effectively used for nonlinear dynamical systems identification and control. In that paper, direct and indirect control approaches are discussed, explaining that at that moment, methods for directly adjusting the parameters based on the output error (between the plant and the reference model output) were not available. Based on low order nonlinear dynamic systems simulation studies, suggested that identification and control using neural network controllers can find wide application in many areas of applied science.

Later , Cui and Shin (1993) propose a direct adaptive controller and coordinator using neural networks. One of the key problems in designing such a controller/coordinator is to develop an efficient training algorithm. A neural network is usually trained using the output errors of the network, not the controlled plant. However, when a neural network is used to directly control a plant, the output errors of the network are unknown, because the desired control actions are unknown. A simple training algorithm is proposed that enables the neural network to be trained with the output errors of the controlled plant. The only a priori knowledge of the controlled plant is the direction of its output response. However, in that paper it is not discussed the influence of some training parameters, particularly the learning coefficient, over the closed loop dynamics. It is also not remarked that practically with a direct neural control scheme the training stage can be substituted by a permanent and real time adaptation of the weighting coefficients of the neural network.

Recently, Noriega et al (2004), propose a self-tuning neural regulator, inspired in the ideas of Cui and Shin, but with the particular feature that the previous training is substituted by a permanent adjustment of the weighting coefficients based on the control error. After an exhaustive simulation test of the algorithm in several non-linear systems they conclude that a previous training of the network is not generally required; the dynamical performance of the closed loop depends exclusively on the learning coefficient magnitude and they found very convenient to use a variable learning coefficient, using an expression as: $\eta' = \eta + \alpha abs(e_y)$, so the system present a fast response when the errors are big and then to go slowly to the reference value.

In the present paper, the Self-tuning Neural Network practical implementation results, in a Non Linear Systems are shown and a methodology to adjust the parameters is proposed. As part of the proposal we present a practical method to accelerate net learning by bounding neuron outputs and a time variable learning coefficient. In Section II, the self-tuning neural controller structure is presented. The weighting coefficients adaptation algorithm is developed in Section III and some recommendations to accelerate the network learning (by bounding neuron outputs and using a time variable learning coefficient) are given. Section IV presents the experimental results in a real non-linear system. The paper concludes with Section V, where some directions are given for further development.

II. Self Tuning Neural Controller Structure

In Figure 1, it is shown the scheme proposed by Noriega et al (2004), for the self-tuning neural regulator. The neural network which assumes the regulator function, is a 3 neuron layers perceptron (one hidden layer) which weighting coefficients are adjusted by a modified back-propagation algorithm.

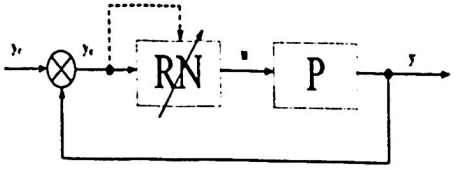

Fig. 1. Self-tuning neural regulator.

In this case, instead of the net output error:

$$e_u(t) = u_d(t) - u(t) \tag{1}$$

it is used the process output error :

$$e_y(t) = y_r(t) - y(t) \tag{2}$$

to adjust the weighting coefficients.

In Fig. 2, it is represented the neural network controller structure. The output layer has only one neuron because by the moment, we are limiting the analysis to one input-one output processes. In the input layer the present and some previous values of the regulation error are introduced, it means that:

$$x(t) = [e_y(t) \quad e_y(t-1) \quad \ldots \quad e_y(t-n)] \tag{3}$$

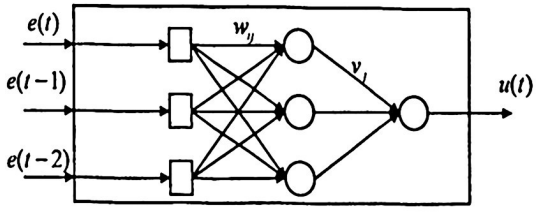

Fig. 2. Neural network controller structure.

For the cases that were simulated in previous works, the value n=2 was sufficient to obtain an adequate closed loop performance. It means that the number of input neurons can be 3. A similar number of hidden layer neurons was equally satisfactory.

III. Weighting Coefficients Adaptation Algorithm

In figure 2, the weighting coefficients w_{ji} and v_j for the hidden layer and output layer input connections respectively, are shown. In what follows, we will detail the adaptation algorithm for those coefficients, which ensures the minimization of a regulation error $e_y(t)$ function.

The output of the j hidden layer neuron may be calculated by means of:

$$h_j = \frac{1}{1+e^{-S_j}}, \quad j = 1,2,3.... \tag{4}$$

Where:

$$S_j = \sum_{i=1}^{3} w_{ji} x_i \tag{5}$$

At the same time, the output layer neuron output value will be:

$$u(t) = \frac{1}{1+e^{-r}} \tag{6}$$

Where:

$$r = \sum_{j=1}^{3} v_j h_j \tag{7}$$

As criteria to be minimized, we defined the function:

$$E(t) = \frac{1}{2} \sum_{k=1}^{t} e_y(k)^2 \tag{8}$$

where it is supposed that the time has been discretized by using an equally-spaced small time interval.

The minimization procedure consists, as it is known, in a movement in the negative gradient direction of the function E(t) with respect to the weighting coefficients v_j and w_{ji}. The E(t) gradient is a multi-dimensional vector whose components are the partial

derivatives $\frac{\partial E(t)}{\partial v_j}$ *and* $\frac{\partial E(t)}{\partial w_{ji}}$, it means that:

$$\nabla E(t) = \begin{bmatrix} \frac{\partial E(t)}{\partial v_j} \\ \frac{\partial E(t)}{\partial w_{ji}} \end{bmatrix} \tag{9}$$

Let us first obtain the partial derivatives with respect to the coefficients of the output neuron. Applying the chain rule, we get:

$$\frac{\partial E(t)}{\partial v_j} = \frac{\partial E(t)}{\partial e_y} * \frac{\partial e_y}{\partial e_u} * \frac{\partial e_u}{\partial u(t)} * \frac{\partial u(t)}{\partial r} * \frac{\partial r}{\partial v_j} \tag{10}$$

$$= e_y . \frac{\partial e_y}{\partial e_u} . (-1) . u(t)(1-u(t)) . h_j \tag{11}$$

In (11) it is used the well known relation:

$$\frac{\partial u(t)}{\partial r} = \frac{\partial\left(\frac{1}{1+e^{-r}}\right)}{\partial r} = \frac{e^{-r}}{\left(1+e^{-r}\right)^2} = \frac{e^{-r}}{1+e^{-r}} \frac{1}{1+e^{-r}} = u(t)(1-u(t)) \tag{12}$$

Let us define:

$$\delta^1 = e_y\, u(t)(1-u(t)) \tag{13}$$

Then:

$$\frac{\partial E(t)}{\partial v_j} = -\delta^1 h_j \frac{\partial e_y}{\partial e_u} \tag{14}$$

In the equation (14), it appears the partial derivative $\frac{\partial e_y}{\partial e_u}$, which can be interpreted as some kind of "equivalent gain" of the process. Further we will make some considerations about that term.

The partial derivative of function E(t) with respect to the weighting coefficients w_{ji}, can be obtained applying again the chain rule:

$$\frac{\partial E(t)}{\partial w_{ji}} = \frac{\partial E(t)}{\partial e_y} * \frac{\partial e_y}{\partial e_u} * \frac{\partial e_u}{\partial u(t)} * \frac{\partial u(t)}{\partial r} * \frac{\partial r}{\partial h_j} * \frac{\partial h_j}{\partial S_j} * \frac{\partial S_j}{\partial w_{ji}} \tag{15}$$

$$= -e_y \frac{\partial e_y}{\partial e_u} u(t)(1-u(t)) v_j h_j (1-h_j) x_i$$

$$\frac{\partial E(t)}{\partial w_{ji}} = -\delta^1 v_j h_j (1-h_j) x_i \frac{\partial e_y}{\partial e_u} \tag{16}$$

Let us define: $$\delta^2{}_j = \delta^1 v_j h_j (1-h_j) \tag{17}$$

And then: $$\frac{\partial E(t)}{\partial w_{ji}} = -\delta^2{}_j x_i \frac{\partial e_y}{\partial e_u} \tag{18}$$

Using equations (14) and (18), the adjustments of weighting coefficients v_j and w_{ji} can be made by means of the expressions :

$$v_j(t+1) = v_j(t) + (\eta \frac{\partial e_y}{\partial e_u}) \delta^1 h_j \tag{19}$$

$$w_{ji}(t+1) = w_{ji}(t) + (\eta \frac{\partial e_y}{\partial e_u}) \delta^2{}_j x_i \tag{20}$$

Where η is the so-called "learning coefficient" and $\frac{\partial e_y}{\partial e_u}$ is the "equivalent gain" of the

The main obstacle to apply the adjustment equations (19) and (20) is that in

general the plant equivalent gain $\frac{\partial e_y}{\partial e_u}$ is unknown. However, in the above mentioned paper by Cui and Shin (1993), it is demonstrated that it is only required to know the sign of that term to ensure the convergence of the weighting coefficients, because the magnitude can be incorporated in the learning coefficient η if the non-restrictive condition $\frac{\partial e_y}{\partial e_u} < \infty$ is accomplished. Besides, the sign of $\frac{\partial e_y}{\partial e_u}$ can be easily estimated by means of a very simple auxiliary experiment, for instance, to apply a step function at the process input.

The assumption that the sign of the gain remains constant in a neighborhood of the operation point of the process is not very strong and it is accomplished in most practical cases. Finally, in the worst of cases, real time estimation of the gain sign could be incorporated, without great difficulties, in the control algorithm. Having in mind the above considerations, the equations (19) and (20) could be written as follows:

$$v_j(t+1) = v_j(t) + \eta \; sign(\frac{\partial e_y}{\partial e_u}) \delta^1 h_j \tag{21}$$

$$w_{ji}(t+1) = w_{ji}(t) + \eta \; sign(\frac{\partial e_y}{\partial e_u}) \delta^2{}_j \, x_i \tag{22}$$

The right value of learning coefficient η can be experimentally determined from the observation of closed loop system performance when some changes are made in the controlled variable set-point. The equations (21) and (22) for the proposed neural controller structure, may be interpreted as the regulator adaptation equations instead of training equations as it is normally done.

III.1 Accelerating the Convergence of the Back-Propagation Algorithm

The back-propagation algorithm as described above encounters the following difficulty. From equations (13), (17), (21) and (22), the increment of weighting coefficients Δv_j and Δw_{ji} can be made by means of the expressions:

$$\Delta v_j = \eta \; sign(\frac{\partial e_y}{\partial e_u}) \delta^1 h_j \qquad \text{Where: } \delta^1 = e_y \, u(t)(1-u(t)) \tag{23}$$

$$\Delta w_{ji} = \eta \, sign(\frac{\partial e_y}{\partial e_u}) \delta^2{}_j \, x_i \qquad \text{Where: } \delta^2{}_j = \delta^1 v_j \, h_j (1 - h_j) \qquad (24)$$

When the actual values h_j and $u(t)$ approach either extreme value, the factors $h_j(1-h_j)$ and $u(t)(1-u(t))$ in equations (23) and (24) make the error signal very small. This implies that an output unit can be maximally wrong without producing a strong error signal with which the weighting coefficients could be significantly adjusted. This retards the search for a minimum in the error. This delay of the convergence is caused by the derivative of the activation function. Unfortunately, any saturating response function is bound to have this property: near the saturation point the derivative vanishes.

On the other hand, when a data acquisition card is used to carry the control signal $u(t)$, the actuator will be saturated for any value of $u(t)$ outside a range determined by the card resolution. For instance, if the card resolution is 12 bits, the actuator will be saturated for $u(t)$ values outside the range [0.00024, 0.99976].

To overcome those difficulties, in the experimental part of this work we normalized the error signal and bounded the weighting coefficients in the range [-4,4], so the algorithm never gets stuck until the local minimum is reached and almost each change in the control signal reflect in the plant behavior.

IV. Experimental Results

The experiment was carried out in a 166MHz Pentium processor computer, a "PCL-818HG" data acquisition card (sample frequency ≤ 100kHz), and a "412" DC servomotor amplifier (24-90 VCD, 10 A). Figure 3 shows a general view of the experiment implementation.

IV.1 System Description

In this system, a pole is connected to the potentiometer shaft, which is used to know the angular position of the pole. A motor with 1600:1 gearbox is connected to the pole with an "L" shaped extension ended in a ring. The ring diameter doubles the pole diameter so there is a backslash in the connection. Depending on the angular position of the motor, for the same motor velocity the pole velocity will change showing a faster response when the ring is connected near the pole base. Figures 4a and 4b show the system at two different positions.

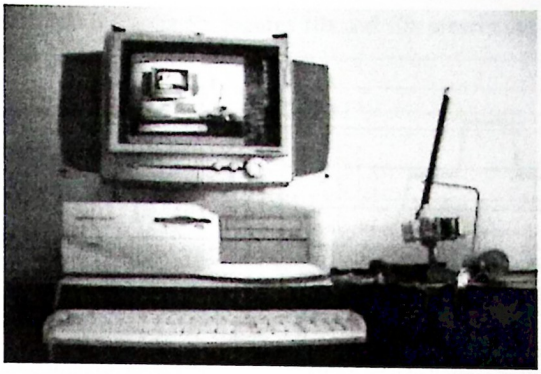

Fig. 3. General view of the experiment implementation.

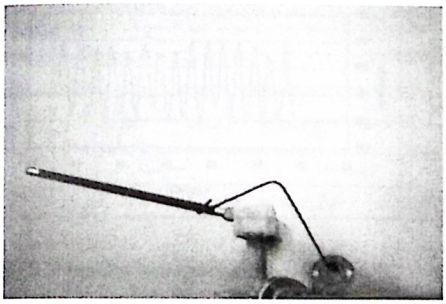

Fig. 4a. System at 160°.

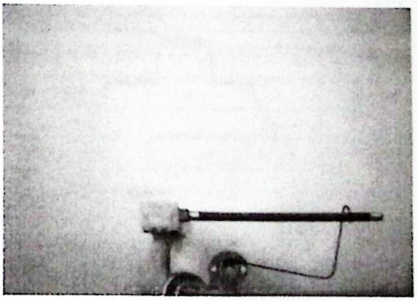

Fig. 4b. System at 0°.

For later reference, a PID controller was tuned to achieve the fastest response of the system without oscillation at step changes of the reference. The controller parameters were adjusted to K = 100, KTτd = 9.2 ms. and (K / Tτi) = (1 / 4.6 s). For those settings, in Figures 5a and 5b the closed loop behavior of the controlled system and the control signal are presented. The required time for the system, to follow the step change of the reference from 23° to 90° is 7 seconds. Any further reduction of this time will lead to oscillation for the step change of the reference from 90° to 157°.

Figure 6a presents the system behavior when the differential control action is eliminated and the figure 6b shows the oscillating response around 157° set point.

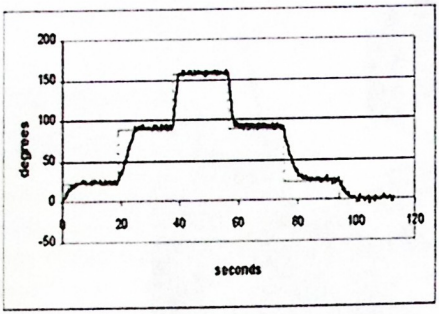

Fig 5a. System controlled whit a PID.

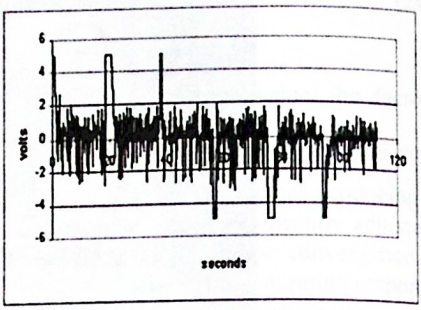

Fig 5b. Control signal for a PID

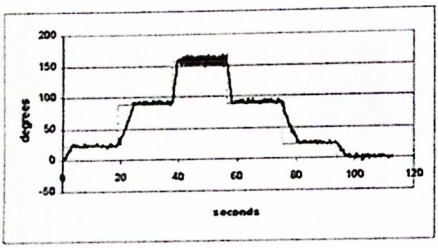

Fig. 6a. System controlled whit a PI.

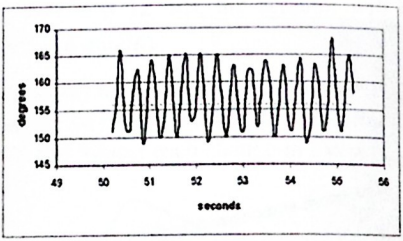

Fig. 6b. Oscillating response around 157°.

The following charts present the experiments whit the self-tuning neural controller. For this experiments the sample and control period were 0.47 ms.

Figure 7 shows the results of the experiments when the learning coefficient is "τ = 0.01". The adaptation time of the weighting coefficients is 50 second and some overshoots show at each reference step. Changing the learning coefficient to "$\tau = 0.01 + 0.1\text{abs}(e_y)$", the adaptation time is reduced to 26 seconds (figure 8).

To reduce the adaptation time it is advisable to use a big learning coefficient at the beginning, and then reduce it to achieve higher stability. Figure 9a shows the results when the learning coefficient $\eta' = \eta + \alpha\text{abs}(e_y)$ is used, and the parameters η and α are adjusted for the first three changes of the reference as follows: (η =0.5 and α=5.0), (η =0.05 and α=0.5) and (η =0.001 and α=0.01). The adaptation time is reduced to 12 seconds and the required time for the system to follow the step change of the reference from 23° to 90° is 3 seconds (43% of the time required for the PID controller in the Figure 5a). The control signal of the self-tuning neural controller, in Figure 9b presents a smoother behavior than

the one shown for the PID in Figure 5b. Figures 10a and 10b present the evolution of two weighting coefficients.

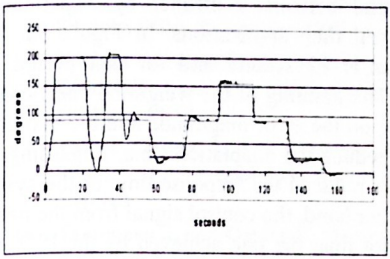

Fig. 7. system controlled whit $\eta = 0.01$.

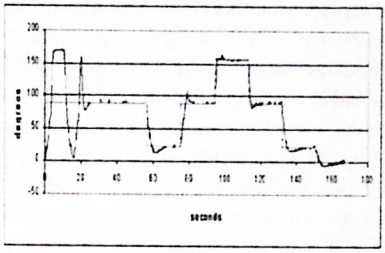

Fig. 8. system controlled whit $\eta = 0.01$ and $\alpha = 0.1$.

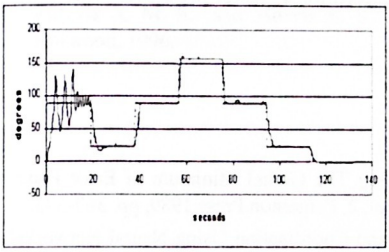

Fig. 9a. η and α adjusted.

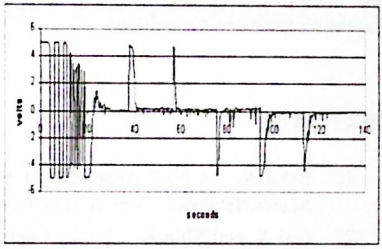

Fig. 9b. Control signal.

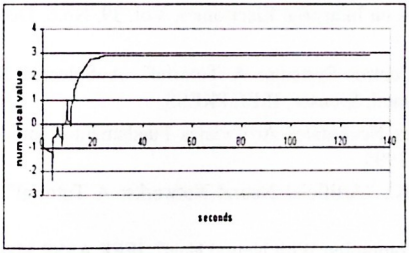

Fig. 10a. Weighting coefficient W[1,2].

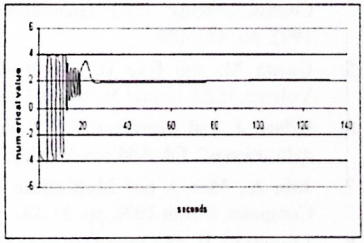

Fig. 10b. Weighting coefficient V[1].

V. Conclusions

According to the experimental results we arise to the following conclusions: It is possible to replace the previous training of the neural network controller by a permanent adjustment of the weighting coefficients in real time applications. In Figure 9a. The adaptation time of the weighting coefficients is 12 second and for that period the algorithm performs 25,000 adaptation cycles. The bonding of the weighting coefficients, and the use of a learning coefficient depending on the error magnitude and the evolution of weighting coefficient played a key role to reduce the adaptation time. Regarding the PID performance, the neural network controller reduced the response time of the system without causing instability problems. On the other hand, the control signal from the neural network controller presents a smoother behavior than the one achieved by the PID, this increases the expected life of the motor and the amplifier by reducing extra heating a vibrations.

To continue with this work it is important to apply the self-tuning neural network controller to other none linear systems including the case of multivariable systems. At the same time it is necessary to implement the algorithm in faster processors like DSP so the adaptation time can be reduced.

References

1. **Baba N.**: "A New Approach for Finding The Global Minimum of Error Function* of Neural Networks". Neural Networks, Vol. 2, Pergamon Press 1989, pp. 367-373.
2. **Cui X.** and **Shin K.**: "Direct Control and Coordination Using Neural Networks". IEEE Transactions on Systems , Man and Cybernetics, Vol. 23, No. 3, May 1993, pp. 686-697.
3. **Freeman J** and **Skapura D.**: "Redes Neuronales Algoritmos, aplicaciones y técnicas de programación". Ed. Addison-Wesley, 1993.
4. **Fukuda T.** and **Shibat T.**: "Theory and Applications for Neural Networks for Industrial Control Systems". IEEE Transactions on Industrial Electronics, Vol. 39, No.6, December 1992, pp. 472-489.
5. **Gupta M.** and **Rao D.**: "Neuro-Control Systems: A Tutorial". A Selected Reprint Volume. IEEE Neural Networks Council, *Sponsor*, IEEE PRESS.
6. **Hilera J.** and **Martínez V.**: "Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones". Ed. Addison-Wesley, 1995.
7. **Jain A., Mao J.** and **Mohiuddin K.**: "Artificial Neural Networks: A Tutorial". IEEE Computer. March 1996, pp. 31-44.
8. **Lippmann R.** "An Introduction to Computing Whit Neural Nets". IEEE ASSP Magazine, April 1987, pp. 4-22.
9. **Moscinski J.** and **Ogonowski Z.**: "Advanced Control with Matlab & Simulink". Ed. Ellis Horwood, 1995.

10. **Narendra K. and Parthasarathy K.**: "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990, pp. 5-27.
11. **Ng G. W.**: "Application of Neural Networks to Adaptive Control of Nonlinear Systems". Ed. John Wiley & Sons Inc, 1997.
12. **Noriega A., Aguado A., Ordaz A.**, and **Rauch V.**: Neural Networks for Self-Tuning Control Systems. Acta Polytechnica Journal of Advanced Engineering. Vol. 44. No 1, 2004, pp. 49-52.
13. **Tai H., Wang J.** and **Ashenayi K.**,: "A Neural Network-Based Tracking Control System". IEEE Transactions on Industrial Electronics, Vol. 39. No.6. December 1992. pp. 504-510.
14. **Tanomaru J.** and **Omatu S.** "Process Control by On-Line Trained Neural Controllers". IEEE Transactions on Industrial Electronics, Vol. 39. No.6. December 1992. pp. 511-521.
15. **Van Ooyen A** and **Nienhuis B.**: "Improving The Convergence of the Back-Propagation Algorithm". Neural Networks, vol. 5, Pergamon Press, 1992. pp. 465-471.
16. **Werbos P.**: "Backpropagation Through Time: What It Does and How to Do It". Proceedings of the IEEE, Vol. 78, no.10, October 1990. pp. 1550-1560.
17. **Zalzala A. M. S.** and **Morris A. S.**: Neural Networks for Robotic Control. Ed. Ellis Horwood, 1996.