Neural Networks for Self-tuning Control Systems

A. Noriega Ponce, A. Aguado Behar, A. Ordaz Hernández, V. Rauch Sitar

In this paper, we presented a self-tuning control algorithm based on a three layers perceptron type neural network. The proposed algorithm is advantageous in the sense that practically a previous training of the net is not required and some changes in the set-point are generally enough to adjust the learning coefficient. Optionally, it is possible to introduce a self-tuning mechanism of the learning coefficient although by the moment it is not possible to give final conclusions about this possibility. The proposed algorithm has the special feature that the regulation error instead of the net output error is retropropagated for the weighting coefficients modifications.

Keywords: neural networks, feedforward, back-propagation, networks, self-tuning control.

1 Introduction

The use of Artificial Neural Nets for systems identification and control, has been the subject of a vast amount of publications in recent years. It is possible to mention for instance, the paper of Narendra and Parthasaraty [5] in which several possible structures of the neural controller that suppose an a priori knowledge of the plant dynamic structure are proposed. In Bhat and McAvoy [3] and in many other papers, a scheme based in an inverse dynamic neural model is used. A similar approach is adopted by Aguado and del Pozo [2] but here the inverse neural regulator is complemented by a self-tuning PID algorithm which guarantees that the stationary error goes to zero. In the above mentioned paper and in many others, it is required an exhaustive previous training of the neural net, which is a serious obstacle for the practical implementation of the proposed solution in the industry. At the same time, the controller structure in some cases, as in the mentioned paper of Narendra and Parthasaraty [5], is very complex, including several nets each one with two hidden layers of many neurons which must be trained using large data samples.

An important contribution to enhance the possibilities of neural nets in the solution of practical problems is the paper by Cui and Shin [4]. In that work a direct neural control scheme is proposed for a wide class of non-linear processes and it is shown that in many cases, the net can be trained directly retropropagating the regulation error instead of the net output error. 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 net.

In the present paper, it is proposed a self-tuning neural regulator, inspired in the ideas of Cui and Shin [4], but with the particular feature that the previous training is substituted by a permanent adjustment of the weighting coefficients based on the control error. At the same time, it is shown the influence of the learning coefficient over the closed loop dynamics and some criteria are given about how to choose that parameter. Finally, some examples are given where the possibilities of the proposed method for difficult non-linear systems control is shown, specially when there exists a considerable pure time delay.

2 Self tuning neural controler structure

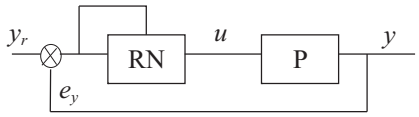

Fig. 1.: Scheme of the self-tuning neural regulator

In Fig. 1, it is shown the proposed scheme of the self--tuning neural regulator. The neural net which assume the regulator function, is a 3 neuron layers perceptron (one hidden layer) which weighting coefficients are adjusted by a modified retropropagation algorithm. 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 net 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:

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

For the cases that has been simulated until now, 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.

3 Weighting coefficients adaptation algorithm

In Fig. 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 algo-

rithm for that coefficients, which ensures the minimisation of a regulation error $e_y(t)$ function.

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

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

where:

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

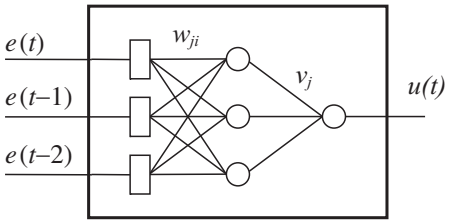

Fig. 2.: Neural net controller structure

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 a criteria to be minimised, 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 minimisation 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 $\frac{\partial E(t)}{\partial v_j}$, $\frac{\partial E(t)}{\partial w_{ji}}$, 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:

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

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

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

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

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

Let us define:

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

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:

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

$$\frac{\partial E(t)}{\partial w_{ji}} = e_y \cdot \frac{\partial e_y}{\partial e_u} \cdot u(t) \cdot (1-u(t)) \cdot v_j h_j \left(1-h_j\right) x_i$$

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

Let us define:

$$\delta_j^2 = \delta^1 v_j h_j \left(1-h_j\right) \tag{17}$$

and then:

$$\frac{\partial E(t)}{\partial v_j} = -\delta_j^2 x_i \frac{\partial e_y}{\partial e_u} \tag{18}$$

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

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

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

where η is the so-called learning coefficient and $\frac{\partial e_y}{\partial e_u}$ is the "equivalent gain" of the plant. 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 [4], 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, by instance, to apply a step function at the process input.

The assumption that the sign of the gain remains constant in a neighbourhood 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:

$$v_j(t+1) = v_j(t) + \eta \operatorname{sign}\left(\frac{\partial e_y}{\partial e_u}\right)\delta^1 h_j \qquad (21)$$

$$w_{ji}(t+1) = w_{ji}(t) + \eta \operatorname{sign}\left(\frac{\partial e_y}{\partial e_u}\right)\delta_j^2\, x_i \qquad (22)$$

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. Indeed, the simulation study done until now using the scheme shown in Fig. 1 and permanently adjusting the weighting coefficients v_j and w_{ji} by means of (21) and (22) allowed us to arrive to the next conclusions:

- It is not in general required a previous training of the net and once the control loop is closed, the weighting coefficients self-adjust in a few control periods, carrying the regulation error $e_y(t)$ to zero.
- The dynamical performance of the closed loop depends exclusively on the learning coefficient magnitude, corresponding to higher values of η, a faster response that can even present a considerable over-shoot. Diminishing the value of η, the system is damped up to the point in which the desired response is obtained. The system, however, keeps the stability for a wide range of η values.
- It is very convenient to use a variable learning coefficient, using an expression as:

$$\eta' = \eta + \alpha \operatorname{abs}\left(e_y\right) \qquad (23)$$

In this way a small basic value of η can be used, for instance $\eta = 0.1$, and the effective value η' is incremented depending of the regulation error magnitude. The right value of α can be tuned experimentally without troubles. The use of equation (23) gives the system the possibility to present a fast response when the errors are big and then to go slowly to the reference value. That behaviour is, indeed, very convenient in practice.

4 Some simulation results

Although the above described algorithm has been tested in many examples, here we will only show the results obtained in two simulated cases corresponding to non-linear processes which can be described by means of the equation:

$$y(t) = \left(\frac{K\, e^{-T_d s}}{(T_1 s + 1)(T_2 s + 1)}\right)^2 u(t)^2 \qquad (24)$$

The simulation was carried out on a real time environment provided by the CPG System (Aguado, 1992) so that the obtained results are very close to those that could be expected in a real process.

In Fig. 3 it is represented the closed loop behavior corresponding to the next process and regulator parameters:

$$T_d = 5\,\text{s}, \quad T_1 = 3\,\text{s}, \quad T_2 = 5\,\text{s}, \qquad (25)$$

$$\eta = 0.6, \quad \alpha = 0.4\,. \qquad (26)$$

As can be seen, a practically perfect response is obtained when positive and negative step changes in the reference output value are applied. Given that the time constants are relatively small, the values of η and α can be chosen relatively big without producing positive or negative over-shoots. This type of performance is observed in general, it means that for fast dynamics processes, it is possible and convenient a fast learning of the net. We have observed that for time constants in the order of milliseconds, values of η of 5 and more can be used.

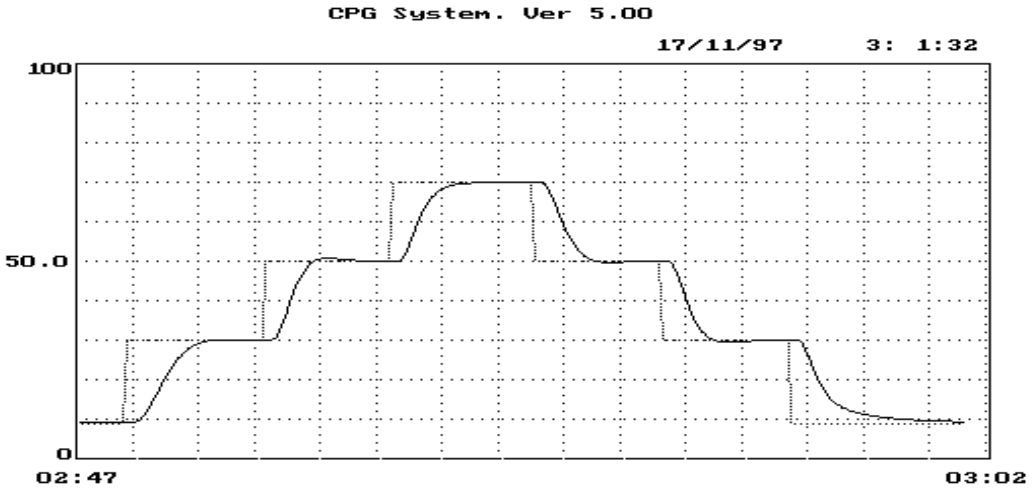

Fig 3.: Closed loop behaviour for process in equations (25)

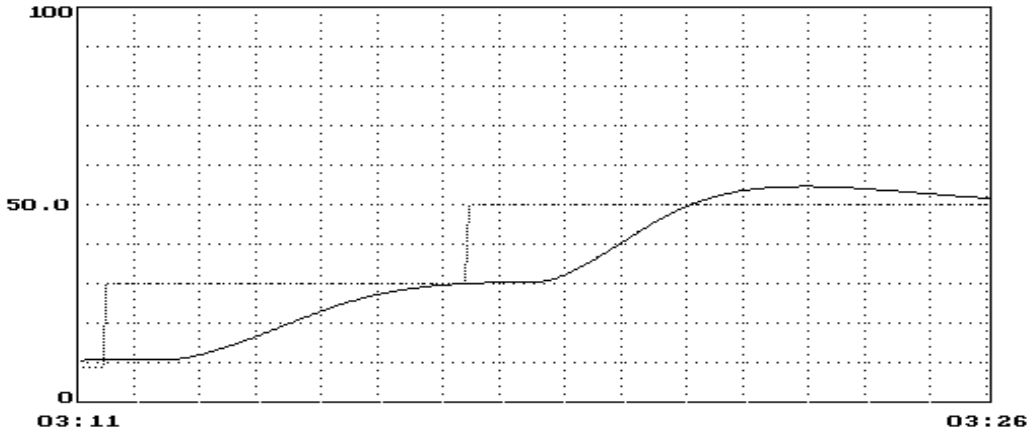

Fig. 4.: Closed loop behaviour for process in equations (27)

In Fig. 4, it is represented the close loop performance for the next constants values:

$$T_d = 60\,\text{s}, \quad T_1 = 10\,\text{s}, \quad T_2 = 20\,\text{s}, \tag{27}$$

$$\eta = 0.05, \quad \alpha = 0.20\,. \tag{28}$$

As can be observed, we have a large time-delay non linear system which can be hardly controlled by a conventional adaptive algorithm, for instance a self-tuning PID. However, with the adaptive neural regulator, it is obtained a response that can be considered as very good, given the process characteristics. Notice that in this case, the values of η and α are considerably smaller, as expected, given that the process dynamics is much slower.

5 Conclusions

The self-tuning control algorithm based on a neural net presented in this paper, promise to be a very interesting option for the control of processes with a difficult dynamics that could not be adequately controlled with PID regulators, even in their self-tuning versions, as it was shown in the above presented simulation cases. The class of processes in which the algorithm could be applied is very wide and it includes most of the cases that can appear in practice. In the near future, we plan to apply the algorithm to some real laboratory processes and to extend the obtained results to the case of multivariable and multiconnected systems.

References

[1] Aguado A.: *Controlador de Propósito General*. Memorias del V Congreso Latinoamericano de Control Automático, La Habana, Cuba, 1992.

[2] Aguado A., del Pozo A.: *Esquema de Control Combinado usando Redes Neuronales*. Memorias de CIMAF 97, La Habana, Cuba, 1997

[3] Bhat N., McAvoy T. J.: "Use of Neural Nets for Dynamic Modelling and Control of Chemical Process Systems." *Computers on Chem. Engineering*, Vol. **14** (1990), No. 4/5, p. 573–583.

[4] Cui X., Shin K. G.: "Direct Control and Coordination Using Neural Networks." *IEEE Transactions on Systems, Man and Cybernetics*, Vol. **23** (1992), No. 3, p. 686–697.

[5] Narendra K. S., Parthasaraty K.: "Identification and Control of Dynamical Systems using Neural Networks." *IEEE Transactions on Neural Networks*, Vol. **1** (1990), No. 1.

M C. Alfonso Noriega Ponce
phone: 01 (442) 1921264
fax: 01 (442) 1921264 ext. 103
e-mail: anoriega@uaq.mx

Dr. Alberto Aguado Behar
M. C. Antonio Ordaz Hernández
Dr. Vladimir Rauch Sitar

Universidad Autónoma de Querétaro
Centro Universitario
Santiago de Querétaro
Qro. C. P. 76010, México

