A NOVEL TECHNIQUE FOR CONTROLLER TUNING

Mohammad Amin Rashidifar[1] and Abdolah abertavi[2]

[1]Department of Mechanical Engineering, Islamic Azad University, SHADEGAN Branch, Iran
[2] Department of Electrical Engineering, Islamic Azad University, SHADEGAN Branch, Iran

Abstract

Neural networks and genetic algorithms have been in the past successfully applied, separately, to controller tuning problems. In this paper we purpose to combine its joint use, by exploiting the nonlinear mapping capabilities of neural networks to model objective functions, and use them to supply their values to a genetic algorithm which performs on-line minimization. Simulation results show that this is a valid approach, offering desired properties for on-line use such as a dramatic reduction in computation time and avoiding the need of perturbing the closed-loop operation.

Keywords

Neural Networks, Genetic Algorithms, PID Autotuning

1. INTRODUCTION

Neural networks (NNs) and genetic algorithms (GAs) have been applied, successfully, for controller tuning, during the last years, in several applications. Until now, however, they have been applied separately, exploiting the nonlinear approximation capabilities of neural networks, on one hand, and employing genetic algorithms for optimization purposes. In this paper, we purpose the joint use of these two tools to obtain, in real-time, the tunings achieved by time-consuming optimization procedures.

The paper is organized as follows: As this work arises from neural network PID autotuning techniques introduced previously, these will be briefly described in Section 2. Section 3, the main core of the paper, justifies the above claim by the mean of examples. This way, Section 3.1 considers controller tuning of a fixed plant. Section 3.1.1 describes a PID autotuning scheme based on two performance indices, while Section 3.1.2 illustrates the tuning of a lag-lead controller based on six control specifications. Preliminary results concerning the update of this technique to the control of time-varying systems will be illustrated in Section 3.2. Final remarks are given in Section 4.

2. EXISTING PID AUTOTUNING TECHNIQUE

The work described in this paper arises from previous work in neural network PID autotuning, and, for this reason, it is briefly reviewed in this section. Two approaches were introduced (Ruano *et al*., 1992), (Mamat *et al*., 1995), differing mainly in the identification measures employed. Basically, the neural networks, previously trained off-line, are used to supply, in real-time, the PID parameters to a standard PID controller.

Concentrating here in the first (chronologically) of the two approaches (Ruano et al., 1992), in this technique three MultiLayer Perceptrons (MLPs) are used, each one of them supplying one of the PID parameters to a PID controller. As the identification measures employed may be obtained in open and in closed loop, this technique is applicable to both, with similar tunings being achieved in both situations.

2.1. Neural network inputs

Let us consider a plant with transfer function:

$$G_p(s) = k_p e^{-d_p s} \frac{\prod_{i=1}^{n_z}\left(1+\tau_{z_i} s\right)}{\prod_{i=1}^{n_p}\left(1+\tau_{p_i} s\right)} \tag{1}$$

The identification measures $F(\alpha)$:

$$F(\alpha) = \frac{G(\alpha)}{k_P}, \alpha \in \Re, \tag{2}$$

for a range of values of α, will be used to identify the plant. In order to serve this purpose, α must be related with the time delay (d_p) and time constants (τ) of the plant. An algebraic sum of these factors, T_T:

$$T_T = d_p + \sum_{i=1}^{n_p} \tau_{P_i} - \sum_{i=1}^{n_z} \tau_{Z_i}, \tag{3}$$

is used as a scaling factor for α:

$$\alpha = \frac{\sigma}{T_T}, \sigma \in [0.1,10] \tag{4}$$

For the sake of the following explanation it is convenient to express the identification measures in terms of σ:

$$F(\sigma) = e^{-\frac{d_p \sigma}{T_T}} \frac{\prod_{i=1}^{n_z}\left(1+\frac{\tau_{Z_i}\sigma}{T_T}\right)}{\prod_{i=1}^{n_p}\left(1+\frac{\tau_{P_i}\sigma}{T_T}\right)} \tag{5}$$

These last identification measures, $F(\sigma)$, are the inputs of the MLPs. The advantage of using them is related to the fact that they can be obtained in real time, both in open and in closed loop, through the use of integral measures of the output and/or the control signal (please see Ruano et al., 1992).

2.2. Neural network outputs

The outputs of the MLPs $\left(k_c^{'}, T_i^{'}, T_d^{'}\right)$ are the parameters of the PID controller $\left(k_c, T_i, T_d\right)$, conveniently scaled. Specifically:

$$k_c = \frac{1}{k_c^{'} k_p}, T_i = T_i^{'} T_T, T_d = T_d^{'} T_T \tag{6}$$

2.3. Neural network training

Before being used in real-time control, the MLPs are trained off-line. In order to obtain good results from this technique, two conditions must be fulfilled:

- The examples employed for training must cover the whole operational range of the controller. An indirect way to solve this problem is to have an *a priori* knowledge of the types of transfer functions, as well as the range of its time constants and delay times that are exhibited during the lifetime of the process. Discretizing this multidimensional space, suitable examples required for training can be obtained.

- For every transfer function in the training set, it is necessary to determine the corresponding PID values. Although any tuning criterion can be employed, which accounts for the versatility of the proposed technique, a suitable criterion that produces well-damped responses should be selected. Thanks to its selectivity, the Integral of Time multiplied by the Absolute Error (ITAE) is used in our method.

After having obtained the training data, the MLPs are trained using a fast training algorithm (Ruano *et. al*., 1991a).

2.4. Results obtained

This technique has been shown, whether in simulations (Ruano *et al*, 1992) or in real-time (Ruano *et al*, 1991b), to produce good results, as long as the plant under control lies within the boundaries of the training space.

3. CURRENT WORK

In the approaches previously introduced, the neural network parameters remain fixed after the off-line training phase. Excellent results have been obtained with both approaches, as long as two conditions are fulfilled:

- the training data adequately spans the input space of the neural networks, and
- the actual plant to be controlled lies within the training space.

The neural PID autotuners can be somehow envisaged as *blind experts,* who were previously taught how to control plants within a certain range, and subsequently perform their task in a very efficient manner. Unfortunately their performance is not improved as a result of continuously on-line operation.

On the other hand, the controller is tuned based only in one tuning criterion, in this case, reference following. It would be of interest to consider, additionally, other tuning aims, such as disturbance rejection, bandwidth criteria, etc.

Current research tries to address these two problems: incorporation of more than one criterion in the tuning and improving on-line the tuning capabilities of the neural networks. Our final aim is to implement a system as the one shown in Fig. 1.

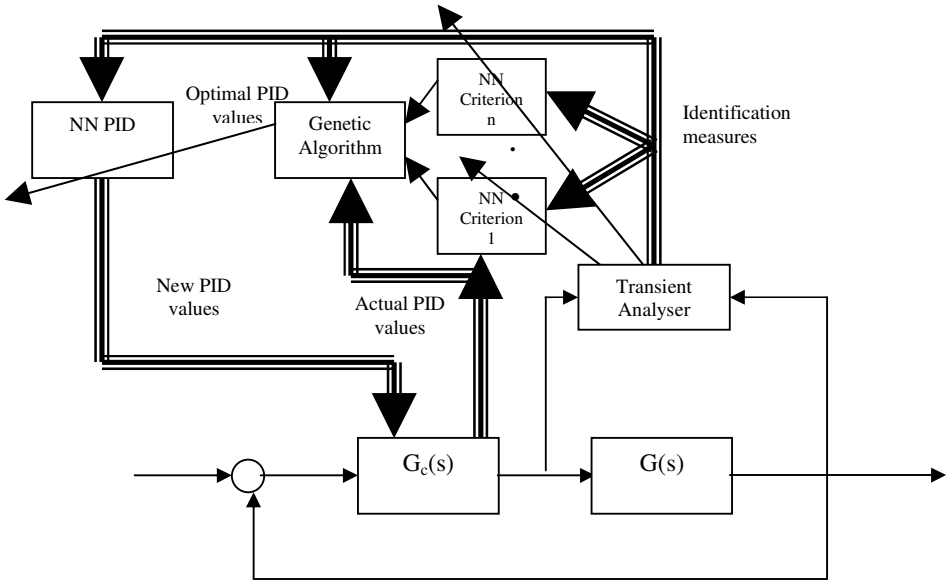

Figure1. PID control using a neuro-genetic auto-tuner

This paper describes current work that is being developed towards that final objective.

3.1. Use of simultaneous tuning criteria

In this case, we consider that the plant is fixed, and that the controller will be tuned considering more than one objective. A Genetic Algorithm will determine the optimal value of the controller parameters, and for that, the neural networks, previously trained off-line, will deliver the objective functions to the GA. We shall consider, under this framework, two examples.

3.1.1. PID Auto tuning

Considering again the PID autotuning problem, let us assume that our plant has transfer function:

$$G(s) = \frac{e^{-s}}{s+1} \tag{7}$$

And is subject to PID control with the topology shown in Figure 2.

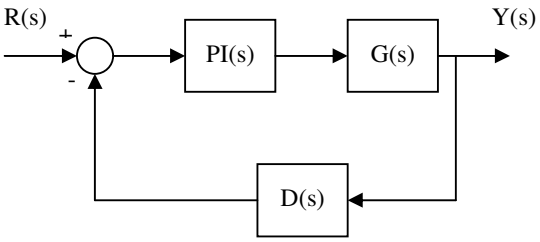

Figure 2. PID Control of plant (7)

Where R(s) is the input reference, Y(s) is the output response, PI(s) is the proportional + integral block with transfer function given by (8):

$$PI(s) = K_c\left(1 + \frac{1}{sT_i}\right) \tag{8}$$

And D(s) is the derivative block with transfer function given by (9):

$$D(s) = \frac{1 + sT_d}{1 + sT_f} \tag{9}$$

Where $T_f = \frac{T_d}{10}$.

In this case, we consider only two objectives for PID tuning, namely, reference tracking and output disturbance rejection. This way, we define the following criteria that we pretend to minimise, (10) and (11). In what concerns reference tracking, the ITAE criterion to a reference step is employed:

$$ITAE = \int t|e(t)|dt \tag{10}$$

Where $e(t) = y(t) - r(t)$.

Relatively to output disturbance rejection, considering a null reference and a unit step added to the G(s) output, the criterion to minimise is (11):

$$ITAY = \int t|y(t)|dt \tag{11}$$

Two MLPs are used to approximate the mappings between the PID parameters and the criteria (ITAE and ITAY) described above.

The NN inputs are PID normalised values defined by (6). In the current application, with the transfer function (7), we have $T_T = 2$ and k_p=1.

The NN outputs are normalised criteria values given by (12):

$$ITAE' = \frac{T_T^{\,2}}{ITAE} \qquad ITAY' = \frac{T_T^{\,2}}{ITAY} \tag{12}$$

After having determined that the best topology for this application was $[3\ \ 9\ \ 5\ \ 1]$, (3 input neurons, 9 neurons in the first nonlinear hidden layer, 5 neurons in the second hidden layer, and one output neuron), the MLPs were trained using a fast training algorithm (Ruano *et al*, 1991b), and used to output the objective functions to the GA algorithm. The GA is employed to minimise a function that takes into account the two objectives defined by (10) and (11). Representing the function to minimise by f(), we have:

$$f(k_c, T_i, T_d) = w_1 ITAE(k_c, T_i, T_d) + w_2 ITAY(k_c, T_i, T_d) \tag{13}$$

In Fig.3 and Fig. 4, we compare the closed-loop responses in three different situations:

a) The optimal PID values are obtained employing a common gradient method for the ITAE minimization. In this case the ITAE values are computed by dynamic simulation, i.e., no neural network is employed;
b) The optimal PID values are obtained by a GA minimizing the ITAE criterion. The ITAE values are obtained using the trained neural network;
c) he PID is tuned by Ziegler & Nichols tuning rules.

In Fig. 3 y(t) represents the closed loop step output response with a null disturbance. Fig. 4 represents the closed-loop output response when the reference is set to zero, and an unit step disturbance is added to the plant output.

In both figures the upper part represents the response y(t), and the lower part represents the evolution of the ITAE and ITAY, respectively. In both figures the dashed-dot line represents situation a), the solid line b), and the crossed (heavy) line situation c).

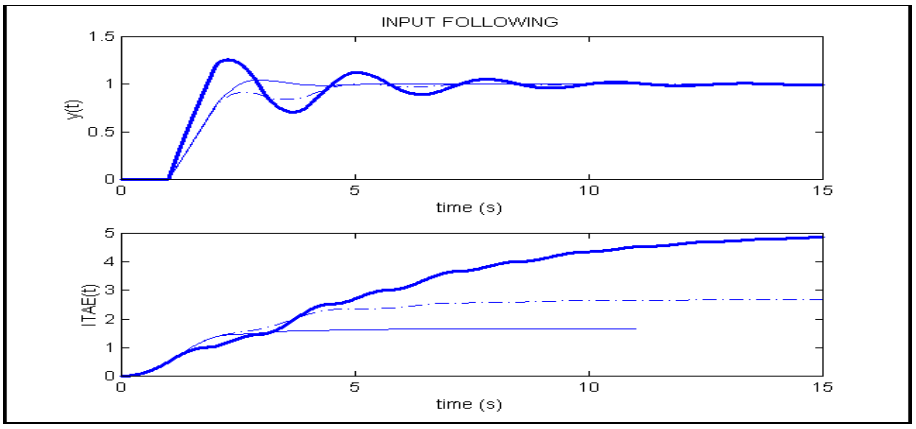

Figure 3. Reference tracking: comparison between situations a), b) and c)

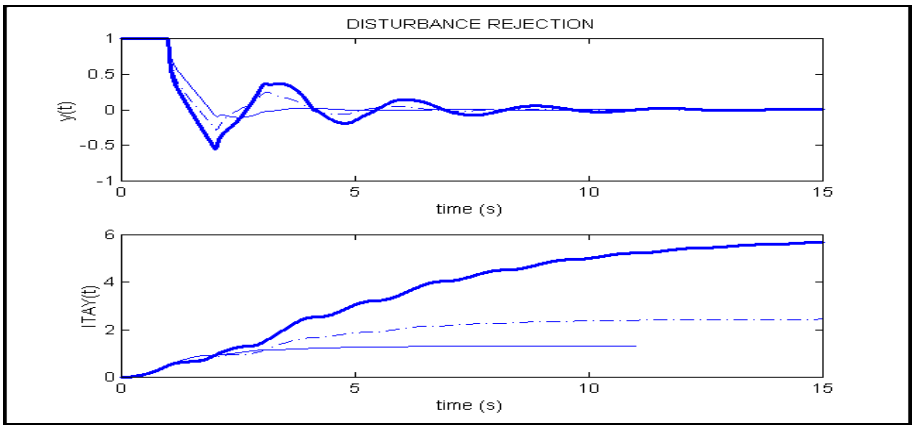

Figure 4. Disturbance rejection: comparison between situations a), b) and c)

Analysing these figures, we can conclude that, by employing the neural network as a model for the ITAE, together with the GA, better results than employing a common gradient minimization technique together with dynamic simulation, and much better than the Ziegler-Nichols results, are

achieved. More important, situation b) takes an insignificant fraction of the time taken by approach a), and, since there is no need to perturb the control loop to compute the ITAE, it can be employed in real-time.

In Fig. 5 and Fig. 6, the optimisation was just performed using the GA, and the neural networks as models of the objective functions. In each figure we can compare the results of three different optimizations, corresponding to different combination of weights w_1 and w_2 in eq. (13). In these figures, the dashed-dot line represents $w_1 = 1,\ w_2 = 1$, the solid line $w_1 = 1,\ w_2 = 0$, and the crossed (heavy) line $w_1 = 0,\ w_2 = 1$

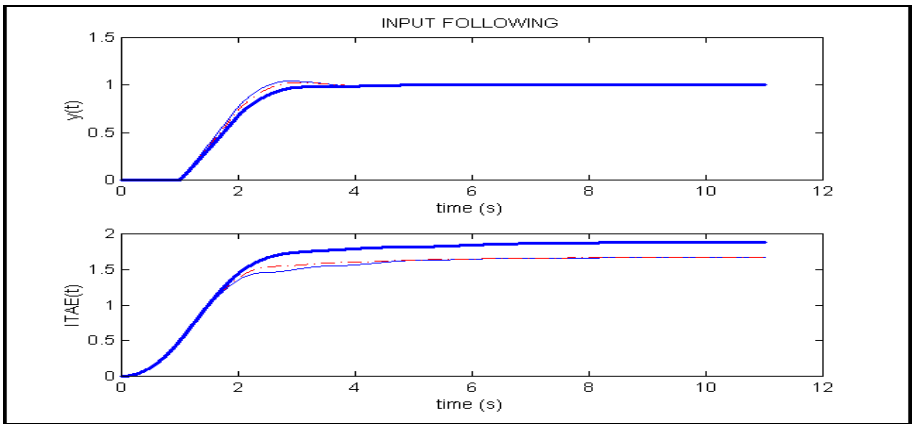

Figure 5. Reference tracking. GA optimisation: comparison between three objective functions.

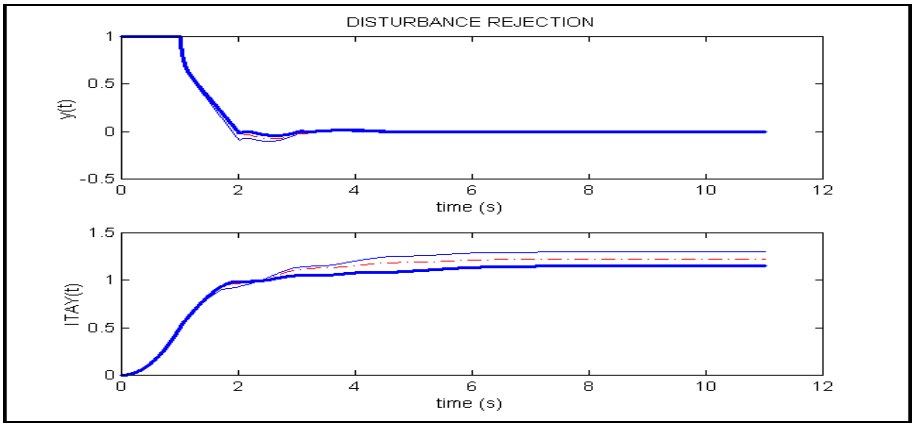

Figure 6. Disturbance rejection. GA optimisation: comparison between three objective functions.

We conclude that the neural networks proved to be effective as ITAE models, and, for this particular plant, the results obtained for tracking and disturbance rejection do not differ very much.

3.1.2. Controller tuning

We consider here the case of controlling a plant with transfer function:

$$G(s) = \frac{1}{s\left(1+\frac{s}{2}\right)} \tag{14}$$

The objective is to design a controller:

$$G_c(s) = k\frac{1+\tau_1 s}{1+\tau_2 s} \tag{15}$$

That satisfies the specifications:

Table 1. Design specifications

	E_{ss}	T_s	T_p	% Os	BW –3 dBs	BW – 40 dBs
Specifications	≤ 0.02	≤ 0.3 s	≤ 0.1s	≤ 20 %	≥ 25 rad/s ≤ 60 rad/s	≤ 300 rad/s

E_{ss} is the steady-state error, T_s is the settling time, T_p is the peak time, %Os denotes the percentage of overshoot, and BW the bandwidth.

Six Radial-Basis-Function networks (RBFs) were trained off-line to approximate the mappings between the controller parameters (k, τ_1 and τ_2) and the corresponding control specifications. An additional RBF was trained to classify between acceptable and non-acceptable (unstable or near unstable) parameter values. After that, the neural networks were used to supply the objective values to a GA minimizing the single objective:

$$f() = \left(\frac{E_{ss}}{0.02}\right)^2 + \left(\frac{O_s}{10}\right)^2 + \left(\frac{T_p}{0.1}\right)^2 + \left(\frac{T_s}{0.3}\right)^2 + \left(\frac{BW-3dBs}{60}\right)^2 + \left(\frac{BW-40dBs}{280}\right)^2 \tag{16}$$

An initial population of 75 random individuals was created, and after 30 iterations of the GA, the closed loop response obtained with the optimal parameters found is shown in Fig. 7.

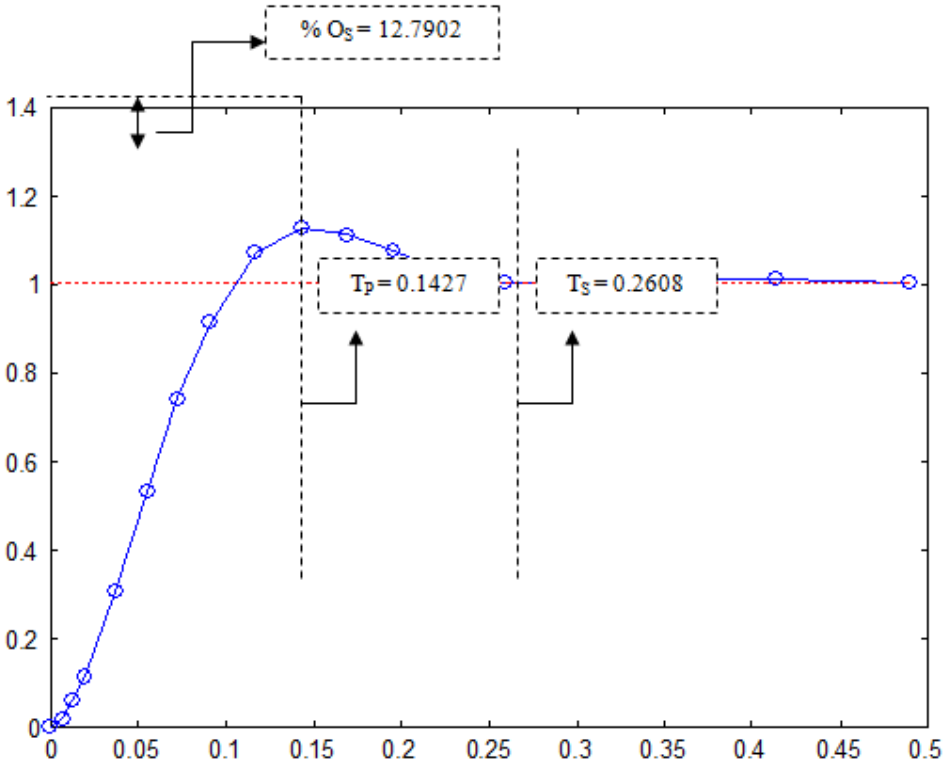

Figure 7 - Closed-loop output obtained with the optimal parameters

The closed loop frequency response can be found in Fig. 8. The value obtained for E_{ss} was 0.036. We can therefore conclude that all the specifications were fulfilled, with the exception of the error in steady state, and the peak time, which were slighted bypassed. Comparing the number of flops spent in the GA optimization process using the neural networks to deliver the objective functions, and the GA employing a Simulink model to obtain the objectives, a saving of more than 14 times was obtained. As the results illustrate, the neural networks proved again their capability to model objective functions.

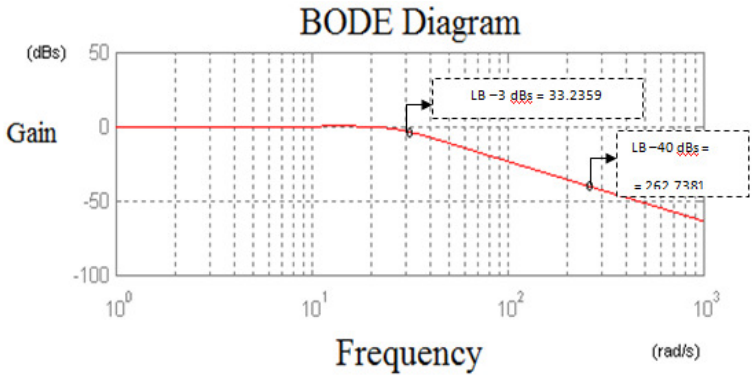

Figure 8. Closed-loop frequency response obtained with the optimal parameters

3.2. Time-varying plant

We consider, at this stage, only one objective, and that the plant is time-varying. If MLPs or, in less extent, RBFs are employed, as these are memoriless networks, on-line training within one region of the input space can modify the mapping achieved in different regions of the input space, which is an undesirable feature. For this reason, the neural networks used here, and in our final objective, are B-spline neural-networks (Brown and Harris, 1994). B-spline networks are a kind of Associative Memory Networks which differs from the others because they store information locally ("learning" about one part of the input space minimally affects the rest of it). To overcome the problem known as *curse of dimensionality* and in order to obtain, during off-line training, a good network structure for later on-line training, we employ the ASMOD construction algorithm (Weyer, 1995).

Focusing our attention here in just the neural network responsible for delivering the PID values (NN PID in Fig. 1), the plant considered has a transfer function

$$G(s) = \frac{e^{-Ls}}{s+1}, \text{ with } L \in \{0.1 : 0.1 : 3\}, \tag{17}$$

From which we obtain the identification measures $F(\sigma)$, given by (5), considering only $\sigma = 1$. Some of the identification measures from this set (6 out of the 30 examples) are used to off-line train three neural networks. As in this case we are considering PID control, tuned by the ITAE criterion, their outputs are optimal scaled PID values given by (6). In order to determine the order of the B-splines to be employed, networks of order 2, 3 and 4 were constructed and trained off-line. It was verified that the best results were obtained with networks of order 3. For this case, after off-line training, we have verified that the neural networks have "learned" exactly the supplied training set.

Afterwards, six passes of on-line learning over the whole 30 examples were performed. The following table, where the unit employed is 10^{-3}, summarizes the results obtained.

For each neural network, three measures of the quality of the approximation obtained for the whole 30 examples are presented: the commonly used sum of the square of the errors, this quantity, but scaled by the sum of the squares of the target patterns, and the sum of the square of the relative errors. It can be observed that there is a steady decrease in all the refereed measures, starting from the situation where just off-line training had been performed (Initial row). Figures 9a) to c) illustrate the evolution of the relative errors, as on-line learning is performed.

It should be noted that a special care must be taken in the selection of the on-line learning rate. The previous results were obtained with a learning rate of 0.1, as attempts to adapt the networks with learning rates of 0.2 and 0.5 shown that the neural networks performance was worse. So, the learning rate chosen must be as small as necessary to guarantee an improving of performance with the on-line adaptation.

Table 2. Best results obtained in the off-line training of the ITAE network using different training strategies.

Presentantion order of inputs	Training Rule	Spline order	Best Structure	M.S.E	Max. $\lvert$rel. err.%$\rvert$	Mean $\lvert$rel.err.%$\rvert$
$F(1), \overline{k_{inv}}, \overline{T_i}, \overline{T_d}$	N.L.M.S	3rd	$F(1)+\overline{k_{inv}}+\overline{T_i}+\overline{T_d}$	2.49	282	34.6
$F(1), \overline{k_{inv}}, \overline{T_i}, \overline{T_d}$	New L.M.S	3rd	$F(1)*\overline{T_d}+\overline{k_{inv}}*\overline{T_i}$	0.2676	274	23
$\overline{T_d}, \overline{T_i}, \overline{k_{inv}}, F(1)$	New L.M.S	3rd	$F(1)*\overline{T_d}+\overline{k_{inv}}*\overline{T_i}$	0.2251	218	18.4
$\overline{T_d}, \overline{T_i}, \overline{k_{inv}}, F(1)$	New L.M.S	2nd	$F(1)*\overline{k_{inv}}+\overline{T_i}*\overline{T_d}$	1.286	114	29.8
All together	New L.M.S	3rd	$F(1)*\overline{k_{inv}}+\overline{T_i}*\overline{T_d}$	1.29	537	30.8
All together	New L.M.S	2nd	$F(1)*\overline{T_d}+\overline{k_{inv}}*\overline{T_i}$	0.3547	312	20.1

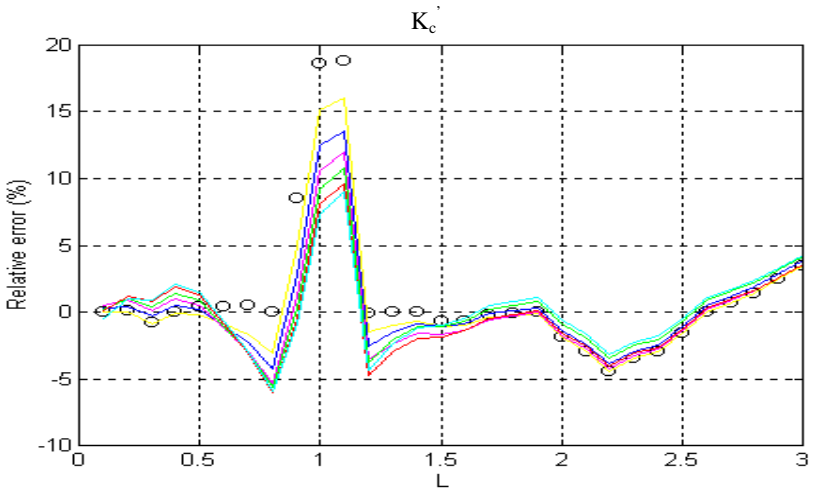

Figure 9a) – Evolution of the relative errors for the $\overline{k_{inv}}$ neural network.

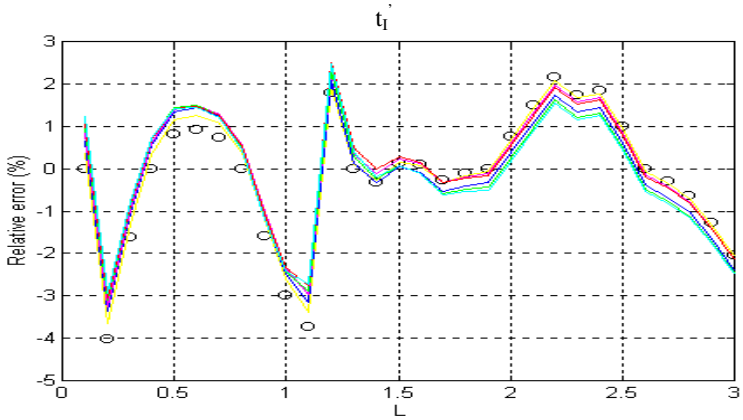

Figure 9b. Evolution of the relative errors for the $\overline{t_i}$ neural network.

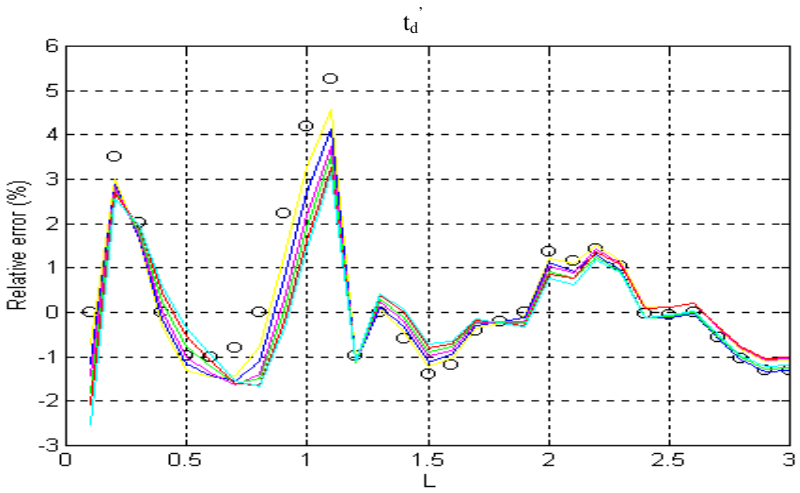

Figure 9c. Evolution of the relative errors for the $\overline{t_d}$ neural network.
o – Initial; iterations: 1st, 2nd, 3rd, 4th, 5th and 6th

4. CONCLUSIONS

In this paper the joint use of genetic algorithms and neural networks for controller tuning was discussed. Simulation results show that neural networks can approximate well the tuning objective functions to deliver, on-line, these values to a genetic algorithm responsible for the minimization of the tuning criteria. The controller optimal values obtained by this procedure deliver good closed-loop tunings. More important, these tunings can be obtained in a fraction of the time spent if neural networks would not be employed, and there is no need to submit the closed loop to perturbations, properties that make this approach suitable for real-time control.
Future work will address the off-line and on-line training of B-spline networks responsible for the modelling of the objectives, and will look at the incorporation of multiobjective optimization tecnhiques in the proposed approach.

REFERENCES

[1] Ruano, A.E., D.I. Jones and P.J. Fleming, (1991a), A New Formulation of the Learning Problem for a Neural Network Controller, Proc. IEEE Conference on Decision and Control, Brighton, U.K., pp. 865-866
[2] Ruano, A.E., D. I. Jones and P.J. Fleming, (1991b), A Neural Network Controller, Proc. IFAC Workshop on Algorithms and Architectures for Real-Time Control, Bangor, U.K., pp. 27-32
[3] Ruano, A.E., P.J. Fleming and D. I. Jones, (1992), A Connectionist Approach to PID Autotuning, IEE Proceedings, Part D., 139(3), pp. 279-285
[4] Brown, M. and C. Harris, (1994), Neurofuzzy Adaptive Modelling and Control, Prentice Hall, London Mamat, R., P.J. Fleming, and A.E. Ruano, (1995), Neural Networks assisted PID autotuning, 2nd Int.
[5] Conf. on Industrial Automation, Nancy, France, pp. 849-854
[6] Weyer, E. and T. Kavli, (1995), The ASMOD algorithm. Some new theoretical and experimental results, SINTEF report STF31 A95024, Oslo