PLANT IDENTIFICATION AND CONTROL USING A NEURAL CONTROLLER BASED ON REFERENCE MODEL.

Cosme Rafael Marcano-Gamero
Systems Engineer (Universidad de los Andes, 1986)
***Magíster Scientiarum* in Electronics Engineering (UNEXPO, 2004)**
Professor at Electronics Department of the
Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO)
Puerto Ordaz – Venezuela.
cosmemarcano@gmail.com

Abstract. Plant or process identification in order to put it under control has always been a problem hard to face up, due to the no linearity of a real process. In this work, Neural Control theory is applied to identify and control a plant conformed by two subsystems of second order which alternate their operation on a constant time base. Firstly, a neural network is trained to learn the plant behavior. Once trained, this network is integrated to the rest of the system in order to jointly operate with another neural network which will serve as a controller. Obtained results permit us assure the actual possibility of using neural networks to identify and controlling this kind of plant. However, special interest must be pay in the controller fine adjustment, in order to minimize the steady state error.

Key words: Neural Control, Plant Identification, Neural Controller Training.

1. INTRODUCTION

Present work consists of a neural control application on a non linear plant, based on the reference model technique. Firstly, a neural network is designed to identify the plant, i.e., the neural network 'learns' the plant behavior through some kind of training, and this knowledge is then used to generate an output signal which is compared with the actual plant output. This comparison is fed back and inputted to another neural network which will act as the controller. This neural controller is designed in such a way that makes the plant output to follow the output of a reference model, which dynamics be well known. Plant under consideration is described in Sec 2, and it is conformed by two subsystems which alternate each *50s* [1]. In Sec. 2, the plant will be described, in such a way a proper neural network design and parameterization be possible to do, as explained in Sec. 3. Another neural network will be trained in order to operate as a plant's controller, taking into account the behavior of a reference model, which dynamic is well known. Results are obtained using Matlab 7.1, through numerical simulations. They are presented and analyzed in Sec. 4. Finally, in Sec. 5, some conclusions are presented. It is important to note that the plant definition describe a highly nonlinear system over which neural control strategy looks appropriate to be used for control purposes.

2. PLANT DESCRIPTION.

Plant under consideration has two subsystems which alternate each 50s. Subsystems 1 and 2 are given by the equations systems (1) and (2), respectively:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-1 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}u \qquad (1)$$
$$y = [1 \quad 0][x_1 \quad x_2]^T$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}u \qquad (2)$$
$$y = [1 \quad 1][x_1 \quad x_2]^T$$

where $[x_1 \quad x_2]$ is the so-called *state vector*; *y* designates the output plant; *u* is a scalar control function. Both, subsystem 1 and 2, are of second order we can see the subsystems poles and zeros in order to find out their particular dynamics. As we know from the control theory, poles can be calculated by solving *det(sI-A)=0,* where *A* designates the so-called *system matrix* or the *state-space matrix*. In doing so, we get that the poles of the subsystem 1 are $s_{1,2} = \{-1{,}0\}$ and subsystem 2 has two identical poles in -1. On the other hand, zeros of subsystem 1 is at ∞, whereas the second subsystem has zeroes at $z_{1,2} = \{\infty, \quad -1\}$. Precisely, the zero located in -1 will cancel the pole located at the same value and, in consequence, this subsystem will behave as a first order one.

3. NEURAL CONTROLLER DESIGN BASED ON REFERENCE MODEL.

There are various neural control strategies. *Neural Networks Toolbox* of *Matlab 7.1* offers three of these strategies: Model Reference Controller, NARMA-L2 controller type and Predictive Neural Network (*NN Predictive*).
Here, we will utilize the Reference Model one that consists in choosing a plant model which description and dynamic are well known in order to make the neural networks which will act as a controller learns, through proper training, its behavior, in order to drive the plant to behave as the reference model. A generic neural controller based in model reference simulink diagram is shown in Figure 1.

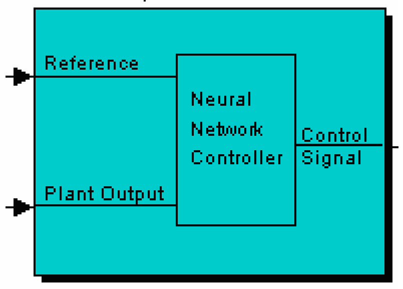


Figure 1.- Reference Model Controller

Looking under the mask, controller internal structure is shown in Figure 2.

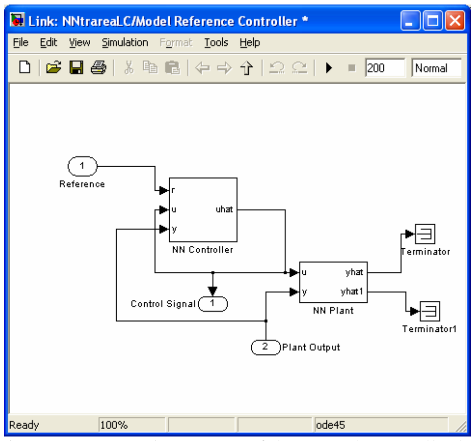


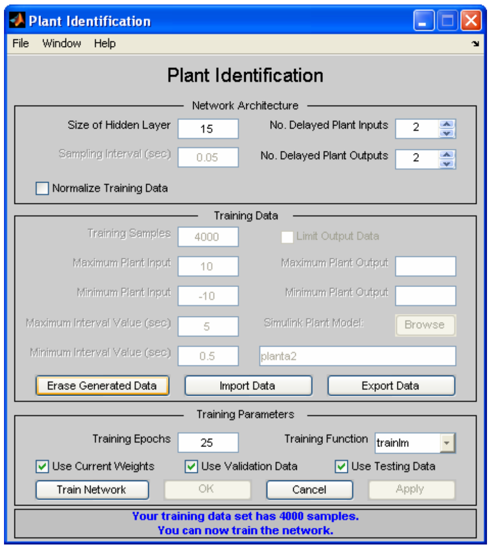
Figure 2.- Internal Structure of the Neural Controller by Reference Model.

Note that, actually, there are two neural networks: the plant identification one and the other, which realize the controller function. It is very important to note the necessity of identifying the plant, first, before proceeding with the neural controller training.

As it can be observed, controller requires the reference model output as an input in order to establish the behavior the plant has to develop. A good description of the plant we want to control is needed as input of the neural controller, in order to it can compare the desired output with the actual one.

A description of the plant to be controlled is also necessary as a controller's input. These inputs and the previous training of the controller allow it to output a control signal which will be inputted to the plant jointly possible external disturbances, if they would exist. In this particular case, disturbances are assumed to be null. Training technique used here is called in the Matlab context as *trainbfgc* [2], due to its promoters, Broyden, Fletcher, Goldfarb y Shannon (BFGS). This technique is of the, so-called, quasi-Newton back-propagation type, which are suitable for training neural controller base in reference models.

Before proceeding to train the controller, it is required to do a previous step, which consists of the plant identification, i.e., the recollection of information about dynamic plant to be controlled.



Plant Identification

Network Architecture

Size of Hidden Layer	15	No. Delayed Plant Inputs	2
Sampling Interval (sec)	0.05	No. Delayed Plant Outputs	2

☐ Normalize Training Data

Training Data

Training Samples	4000	☐ Limit Output Data	
Maximum Plant Input	10	Maximum Plant Output	
Minimum Plant Input	-10	Minimum Plant Output	
Maximum Interval Value (sec)	5	Simulink Plant Model:	
Minimum Interval Value (sec)	0.5		planta2

Training Parameters

Training Epochs	25	Training Function	trainlm

☑ Use Current Weights ☑ Use Validation Data ☑ Use Testing Data

Your training data set has 4000 samples.
You can now train the network.

Figure 3.- Plant Identification Input Data.

At this point, it is necessary to set up parameters like hidden neurons number of the neural network that will identify the plant. In addition, the sampling time *T*, minimum and maximum time for taking samples during training, and the minimum and maximum plant input signal values in order to establish the inputs range for the controller. Figure 3 shows the input data used to identify the plant.

Once this data values are inputted, a training data generation is required. Note that in this dialog box, the file name of the *Matlab simulink* diagram which contains the plant description to be identified has to be entered (in this case, it is called *planta2.mdl*).

After generating the training data, we have to continue the identification neural network training, by pressing the *network training* button. Afterwards, input and output data are displayed on the screen. Process performance is measured through the MSE index, which behavior is shown in Figure 4. To identify the plant, a Levenberg-Marquandt (*trainlm*) technique is used.

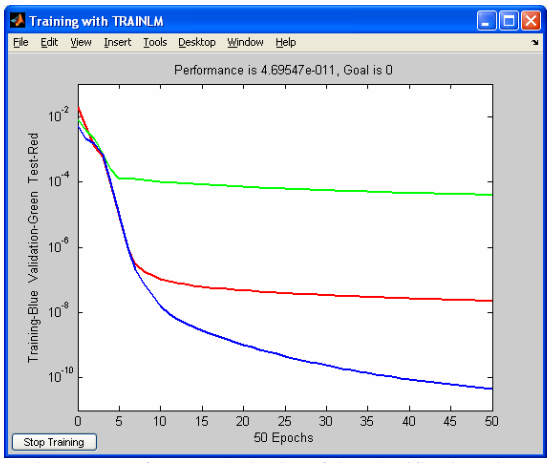
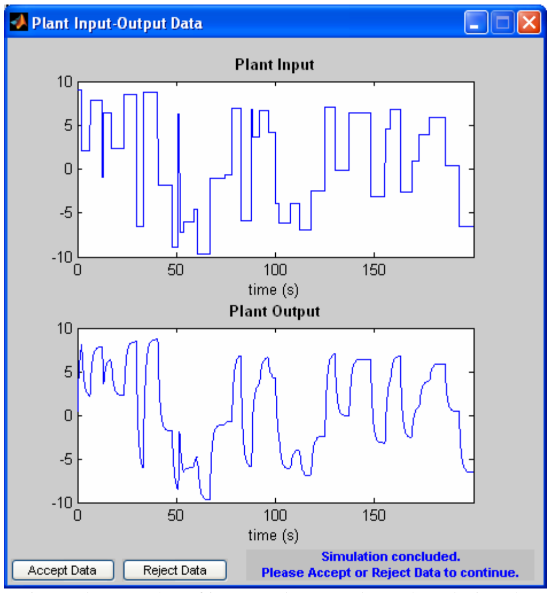


Figure 4.- Performance behavior of the Identification Plant process.

Note, also, that the Matlab stipulated goal for the performance index is zero. In this case, the reported value for that index was 4.69547-011, which is good enough for all practical purposes.

Identification process gives us the results shown in Figure 5.

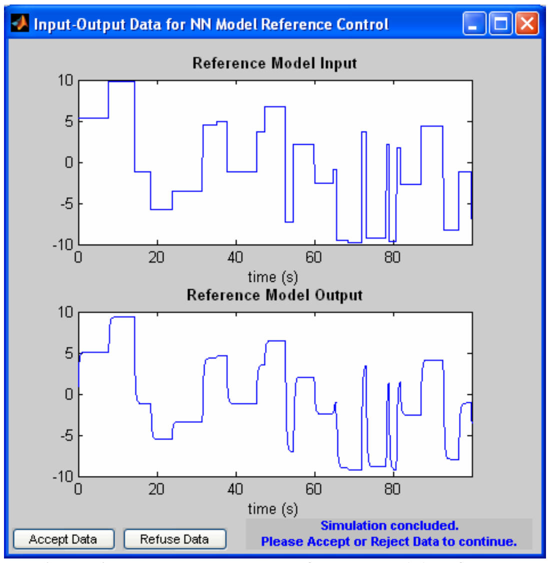
Furthermore, Matlab realize two steps: firstly, it trains the neural network with the input data shown before. Secondly, it tests the consistency of the results, using random data, to assure it is different from the known data used before.



Simulation concluded.
Please Accept or Reject Data to continue.

Figure 5.- Samples of input and output data taken during the identification process

Figure 7 shows the first step results and Figure 8 presents the second step ones.



Simulation concluded.
Please Accept or Reject Data to continue.

Figure 6.a.- Input Output Data for NN Model Reference Controller.

Note that, as we can expect, the output does not follow the input exactly, due, between other factors, to the inertia and time constants of the plant. In fact, very short input variations are not "seen" by the system at all.

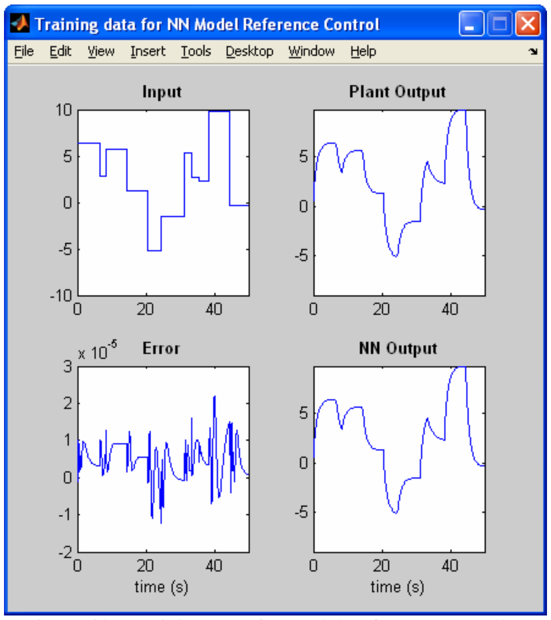


Figure 6.b.- Training Data for Model Reference controller.

If we consider that it is a good enough approximation to the plant behavior, we accept it and proceed with the rest of the controller training process.

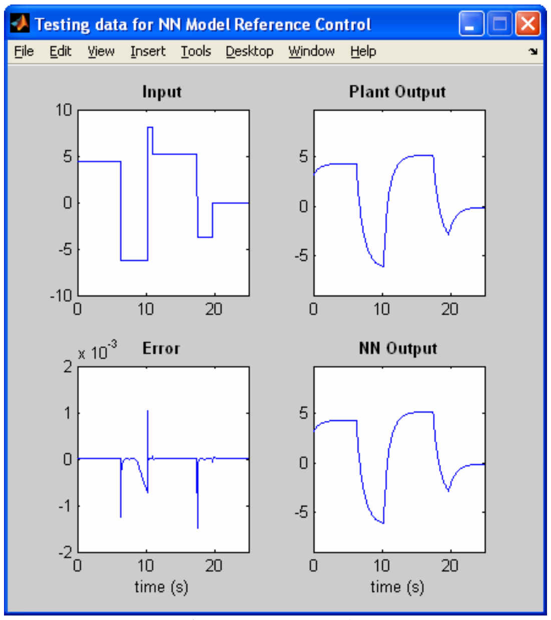


Figure 7.- Test results.

Next, we proceed with the training controller. As we mentioned before, Matlab uses a training technique named *trinbfgc*, which reported the performance shown in Figure 9

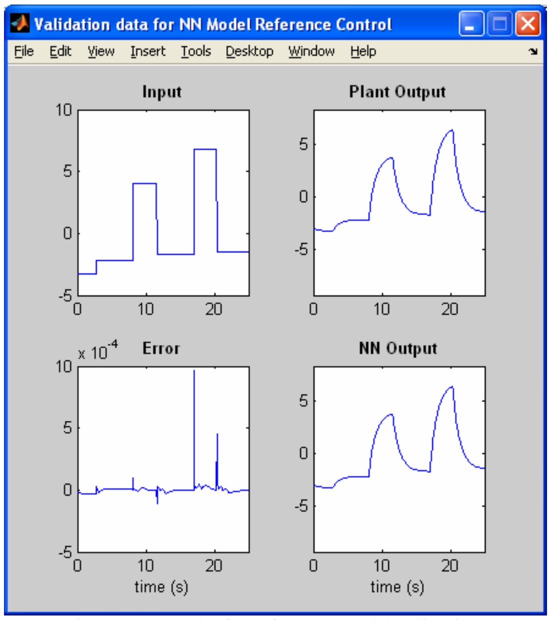


Figure 8.- Results for reference model validation.

Controller training process stops after 40 epochs and, on the Matlab console, appears the following information:

```
TRAINBFGC-srchbacxc, Epoch 50/50, MSE 0.0601074/0, Gradient 3.77025e+007/1e-006, dperf -1.42148e+015 tol 0.0005 delta 0.01 a 0
TRAINBFGC, Maximum epoch reached, performance goal was not met.
```

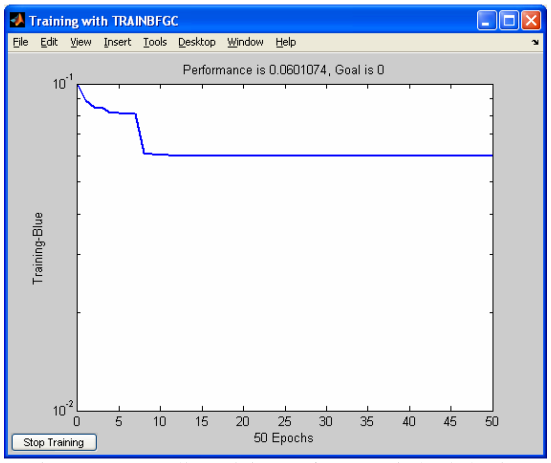


Figure 9.- Controller training performance index behavior using *trainbfgc*.

Performance is measured through the *Médium Square Error, MSE*, which is calculated by:

$$MSE = \frac{1}{Q}\Sigma_{i=1}^{Q} e^2(t) = \frac{1}{Q}\Sigma_{k=1}^{Q}(t(k) - a(k))^2 \qquad (3)$$

where Q is the number of input/output pairs used for training purposes.

$$\{p_1,t_1\},\{p_2,t_2\},\ldots,\{p_Q,t_Q\} \qquad (4)$$

t(k) is the *k-th* plant output value for a given input value $p(k)$, and $a(k)$ is the *k-th* output expected value. After the *MSE* is calculated, a minimum square algorithm (LMS), is used to adjust weights and biases of the neural network associated with the controller.

Because of the stipulated zero goals was not reached in 40 training epochs, we can opt to continue the training process from this point by selecting the *Use Cumulative Training* option, to try to reach a performance index value closer to the goal.

Figures 10, 11 and 12 show the results reported for the plant behavior after be submitted to the controller action, during controller training phase.

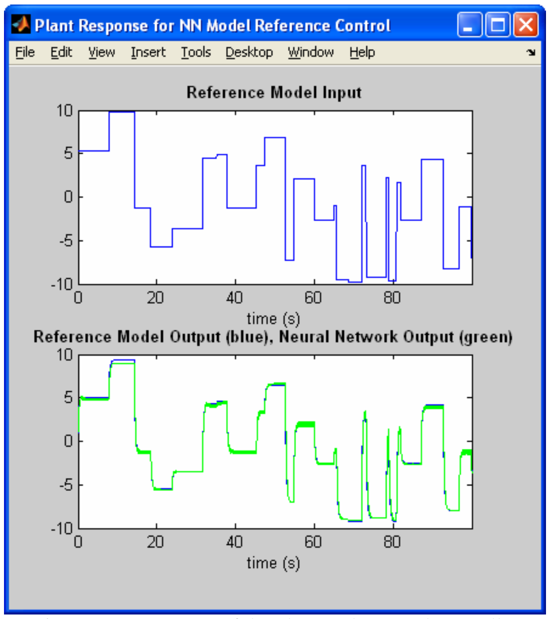


Figure 10.- Response of the plant to the neural controller action.

Testing and validation data and respective output of the plant are shown in these figures. Note that, although the general behavior of the reference model is followed by the plant operating under control of the neural controller, some 'chattering' appears on the ridges of the response signal. It will also appear on the 'real' plant response.

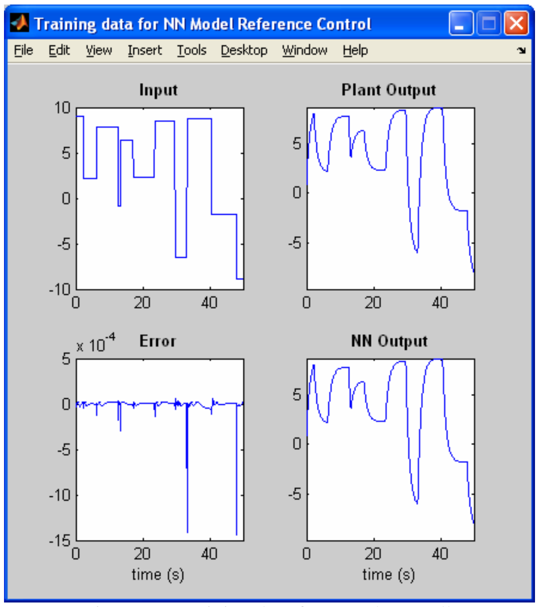


Figure 11. Training data for neural controller

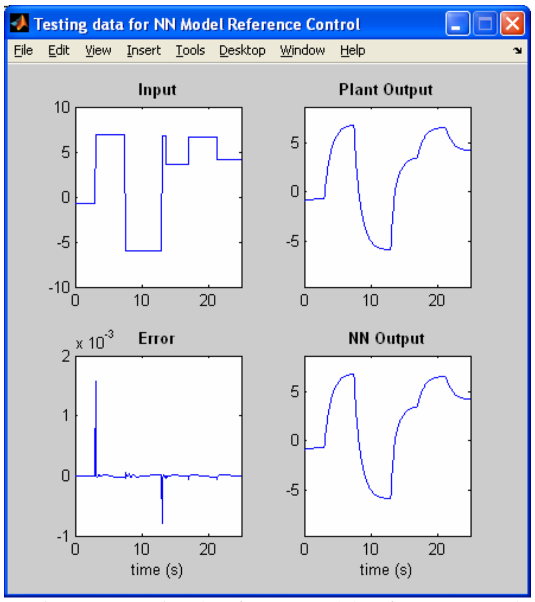
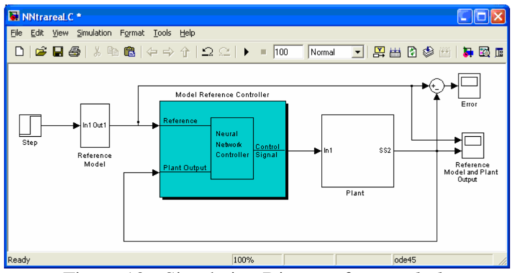


Figure 12. Testing data for NN Model Reference Control

After the end of the controller training process, results should be saved, and the controller block is applied to the 'real' plant as shown in Figure 13.



NNtrareal.C *
File Edit View Simulation Format Tools Help
Step
In1 Out1
Reference Model
Model Reference Controller
Reference
Neural Network Controller
Control Signal
Plant Output
In1
SS2
Plant
Error
Reference Model and Plant Output
Ready
100%
ode45

Figure 13.- Simulation Diagram for *simulink*.

Figure 14 shows the internal structure of the model that is being used as reference.

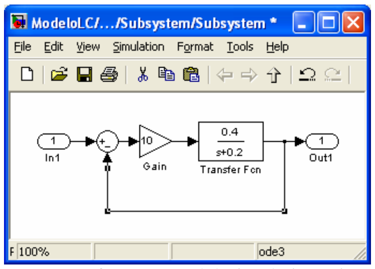

ModeloLC/.../Subsystem/Subsystem *
File Edit View Simulation Format Tools Help
1
In1
10
Gain
0.4
s+0.2
Transfer Fcn
1
Out1
100%
ode3

Figure 14.- Reference Model Simulation Diagram.

4. RESULTS ANALYSIS.

Inputting a step of amplitude 1 to the overall system, i.e., the plant plus the neural controller and the reference model, as appears in Figure 11, we obtain the results displayed on Figure 15.

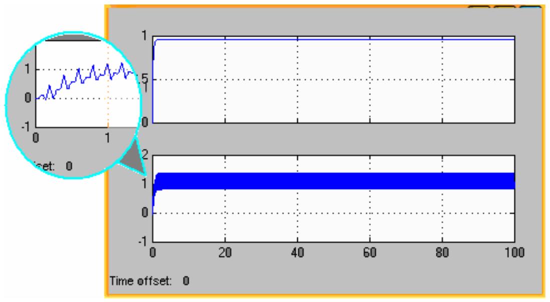


Figure 15.- Response of the Plant (down) and of the Reference Model (up)

As we can see, the dynamic of both plant and reference model are very close, but there exist a steady state error, which can be explained by remembering that first order systems controlled by a proportional control law only, as the one used as reference model, presents an avoidable steady state error: In fact, as we can see on Figure 15, the output value of the reference model is 0.9524, instead of 1

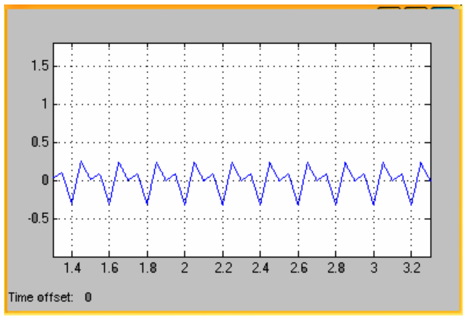


Figure 16.- Difference between plant output and model output.

. On the other hand, the performance index reported by the training process did not reach the goal, as seen in Figure 9.

As we mentioned before, more number of training epochs can report a performance index closer to the stipulated goal, which surely lead us to get a plant output closer to the model one.

5. CONCLUSSION.

This work presents an application example of the neural controller based on reference model, as has been treated for researchers such as Prof. Marios Polycarpou [3], of the Southern California University,

From the results reported by simulations, we can extract some conclusions.

1. It is really possible to synthesize and implement a control law based on the use of neural networks which, properly trained, may reproduce a model output from a plant for which we don't dispose of an adequate mathematical description, as a whole, or no description at all is available, using characteristic input and output data points, which can be used to train a neural network.
2. The adequate joint behavior of the overall system will depend on factors like model selection, which dynamic should be well known by the designer and its dynamics should be similar to the plant which we want to control.
3. Previous comment implies the necessity for the designer to know the plant dynamic as closely as possible, if not from a mathematical point of view, at least by his/her own experience operating that plant.
4. To get a better plant response, the designer should also choose a good reference model that, for example, doesn't present steady state error.

Other considerations can also be taken into account when treating with neural networks, such as the training technique that will be used for both the controller and the plant identification process. However, other techniques could need much more hardware and software resources than the ones used here.

6. REFERENCES

[1] *M. Goire, J. Martín Flores, Á. M. and RoDilla, I.S. Raruch*. Neural Control by Reference Model @2003, CJC- JPN. *ISSN 1405-5546* Computación y Sistemas Vol. 6 No. 4 pp. 284 – 292. México, 2003.

[2] *H. Demuth, M. Beale and M. Hagan*. Neural Networks Toolbox 5. User's Guide. MathWorks, Inc. Revised in September 2006.

[3] *M. Polycarpou y P. Ioannou*. Identification and Control of Nonlinear Systems using Neural Networks: Design and Stability Analysis. Téchnical Report published in September 1991. Souther Californi University, Los Angeles, USA. 44 pages. Available at http://citeseer.ist.psu.edu/524649.html. Visited on 06/22/2007.