Journal of Computer Science, 9 (2): 183-197, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.183.197 Published Online 9 (2) 2013 (http://www.thescipub.com/jcs.toc)

OPTIMAL CONTROL ALGORITHMS FOR SECOND ORDER SYSTEMS

Danilo Pelusi and Raffaele Mascella

Department of Communications Science, University of Teramo, Italy

Received 2012-10-13, Revised 2012-12-14; Accepted 2013-04-02

ABSTRACT

Proportional Integral Derivative (PID) controllers are widely used in industrial processes for their simplicity and robustness. The main application problems are the tuning of PID parameters to obtain good settling time, rise time and overshoot. The challenge is to improve the timing parameters to achieve optimal control performances. Remarkable findings are obtained through the use of Artificial Intelligence techniques as Fuzzy Logic, Genetic Algorithms and Neural Networks. The combination of these theories can give good results in terms of settling time, rise time and overshoot. In this study, suitable controllers able of improving timing performance of second order plants are proposed. The results show that the PID controller has good overshoot values and shows optimal robustness. The genetic-fuzzy controller gives a good value of settling time and a very good overshoot value. The neural-fuzzy controller gives the best timing parameters improving the control performances of the others two approaches. Further improvements are achieved designing a real-time optimization algorithm which works on a genetic-neuro-fuzzy controller.

Keywords: PID Controllers, Fuzzy Logic, Genetic Algorithms, Second Order Plants, Neural Networks

1. INTRODUCTION

The quality of control in a system depends on settling time, rise time and overshoot values. The main problem is to optimally reduce such timing parameters, avoiding undesirable overshoot, longer settling times and vibrations. To solve this problem, many authors have proposed different approaches. A first approach is the Proportional Integral Derivative (PID) controllers application. They are extensively used in industrial process control application. Vaishnav and Khan (2007) designed a Ziegler-Nichols PID controller higher order systems. A tuning method which uses PID controller has been developed (Shamusuzzoha and Skogestad, 2010). Such method requires one closed-loop step setpoint response experiment similar to the classical Ziegler-Nichols experiment. However, in complex systems characterized by nonlinearity, large delay and time-variance, the PID's are of no effect (Cao *et al*., 2008). The design of a PID controller is generally based on the assumption of exact knowledge about the system. Because the knowledge is not available for the majority of systems, many advanced control methods have been introduced. Some of these methods make use of the fuzzy logic which simplifies the control designing for complex models. As an example Kumar and Garg (2004) designed a fuzzy controller to control a single link manipulator robot. Moreover, a gain tuning fuzzy controller has been designed to monitor the track seeking in optical disks (Huang and Su, 2007). In order to improve the control precision of a ball mill circuit, a fuzzy interpolation algorithm is presented (Cao *et al*., 2008). Moreover, PID fuzzy controllers can be designed as power system stabilizer (Corcau and Stoenescu, 2007).

The design of a fuzzy controller depends on the choice of membership functions. A natural choice through trial and errors procedures is impossible to obtain, overall for complex systems. In these situations, a huge computational time is necessary. In order to overcome such difficulty, Genetic Algorithms (GA) are applied to fuzzy controllers with good results (Khan *et al*., 2008; Kumar and Garg, 2004; Chegeni *et al*., 2007; Pelusi, 2011c). Such genetic methods are useful approaches for problems that require efficient searching.

Corresponding Author: Danilo Pelusi, Department of Communications Science, University of Teramo, Italy

Khan *et al*. (2008) the membership functions and the fuzzy logic rules were optimized through Genetic Algorithms methods for a temperature control system. Thanks to GA, Leng *et al*. (2006) eliminated the limitation on symmetric membership functions and symmetric fuzzy rules. To achieve better control performances of complex systems, neuro-fuzzy techniques are developed. These two techniques match the capability of modelling a problem using the knowledge with the capability to learn from data. A neural-fuzzy network can self-adjust the parameters of rule base using neural-network-based learning algorithms. In the literature, a datadriven adaptive neuro-fuzzy controller has been designed for the water-level control of U-tube steam generators in nuclear power plants (Munasinghe *et al*., 2005). Moreover, in (Allaoua *et al*., 2009) a neuro-fuzzy controller has been designed to control the DC motor speed.

Many authors have proposed suitable combinations of fuzzy, genetic and neural techniques for different applications (Leng *et al*., 2006; Saridakis *et al*., 2006; Cho, 2002). In this way, hybrid intelligent algorithms have been developed. For example, a hybrid algorithm based on a genetic algorithm to design a neuro-fuzzy network is proposed in (Leng *et al*., 2006). In such work, the model has been built for a system without a priori knowledge about the partitions of input space and the number of fuzzy rules. Akbarzadeh *et al*. (2000) hybrid paradigms are successfully implemented to solve three prominent robot control issues. Handwritten digit recognition can be solved through combining methods of neural networks (Cho, 2002). The proposed hybrid method uses some fuzzy concepts to combine the outputs of separate networks which relevance is assigned by GA. Good solution for real-time crack identification systems is described in (Saridakis *et al*., 2006). In this work, the analytical model is approximated with a neural network which is used to solve the inverse problem of the crack identification. A genetic search method produces values for the crack attributes as input arguments to the neural network and the genetic algorithm objective function relies on a fuzzy logic representation. Recent studies (Pelusi, 2011a; 2011b; 2012) have proposed genetic-neuro-fuzzy techniques able to improve the timing performances of second order control systems.

The aim of this study is to achieve an optimal control performance of industrial actuators designing suitable controllers. Four research guidelines are considered. The first one regards the design of a PID controller based on Ziegler-Nichols tuning formula (Xue *et al*., 2007). Ziegler and Nichols presented two methods: the step response method and the frequency response method (Astrom and Hagglund, 2004). The first method is presented in this work and it is applied to three different plants. Our second approach attempts of improving the PID timing results designing a fuzzy controller optimized through GA techniques. The optimization is initially made on membership functions only, subsequently, with the same genetic procedure, is made on fuzzy rules. The design of a suitable neuro-fuzzy controller which improves the performances of genetic-fuzzy controller is the third approach of our model. To further improve the settling time and rise time values, a suitable real-time optimization algorithm is designed. Such algorithm works on a suitable genetic-neuro-fuzzy controller. The target of these different approaches is also to improve the simulation results shown in (Khan *et al*., 2008).

1.1. Tuning Parameters of PID Controller

The PID controllers have a wide range of applications in industrial control because of their simple control structure. The PID controllers need of less plant information than a complete mathematical model. In this way, the controller parameters can be adjusted with a minimum of effort. One survey of Desborough and Miller (2002) indicates that more than 97% of regulatory controllers utilize the PID algorithm.

There are many versions of a PID controller. In this study, we consider a controller described by Equation (1):

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \quad (1)$$

where, u(t) is the input signal sent to the plant model, e(t) = r(t)-y(t) the error, y(t) the output and r(t) is the reference input signal. The parameters K_p, T_i and T_d are the tuning parameters. There are more ways to obtain the tuning values of K_p, T_i and T_d: our PID controller uses the Ziegler-Nichols tuning formula. The tuning formula is obtained when the plant model is given by a first-order plus dead time which can be expressed by Equation 2:

$$G(s) = \frac{k}{1+sT}^{-sL} \quad (2)$$

A huge variety of plants can be approximately modeled by (2). If the system model cannot be physically derived, experiments can be performed to extract the parameters for the approximate model (2). For instance, if the step response of the plant model can be measured through an experiment, the output signal can be recorded and the parameters k, L and T (or a, where a = kL= T) can be extracted (Xue *et al*., 2007). The proposed PID controller is designed to control some second order control systems. In order to verify the robustness of the model, we consider three plants with different transfer functions. The first one is typically used to approximate the working of DC motors (Khan *et al*., 2008) and has the form Equation 3:

$$G_1(s) = \frac{2}{s^2 + 12s + 24} \tag{3}$$

The second transfer function (Equation 2) is used for processes with first order dynamics with time delay Equation 4 (Amini, 2008):

$$G_2(s) = \frac{1}{(1+s)(1+5s)} \tag{4}$$

The third transfer function (Equation 5) is joined to the attempts of many researches of improving the tuning parameters through intelligent techniques (Meza *et al*., 2009):

$$G_3(s) = \frac{400}{(s^2 + 50s)} \tag{5}$$

The **Fig. 1** shows the block diagram of PID controller. The difference between the step and the output feedback is passed as input into PID controller block. Such block contains MATLAB functions which implement the Ziegler-Nichols tuning formulas (Xue *et al*., 2007).

The output of the PID controller block serves as an input to the transfer function block. We consider the PID controller behavioral for different plants defined by (3), (4) and (5) whereas the intelligent controllers are designed only for second order plants with transfer function (3).

1.2. Design of Genetic-Fuzzy Controller

In order to improve the timing performances of designed PID controller, suitable genetic procedures are used.

Generally, the first step to design a fuzzy system is the choice of the number of input/output membership functions. Assuming all possible rules are used (which is often the case), if the membership functions number increasing, then the number of rules grows exponentially. It needs to avoid this situation because it is very important trying to minimize the time to compute the fuzzy controller outputs given some inputs. Some studies (Chopra *et al*., 2005) deal with the design of fuzzy logic controllers with less number of rules leading to a smaller amount of computational time. The designed fuzzy controller has two inputs: the error e, that is the difference between the reference value and the output of controller and the change in error de, that is the difference between the error at time t and that one at t-1. These inputs have seven membership functions: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero Error (ZE), Positive Small (PS), Positive Medium (PM), Positive Big (PB). The fuzzy output has the same membership functions of fuzzy inputs. Analyzing the findings of (Chopra *et al*., 2005), we define the rules of **Table 1**. During the rules designing process, we have discovered that increasing the fuzzy rules beyond 49 rules is useless. In fact, this procedure increases the complexity of fuzzy logic controller and has no positive effects on output response of the system.

The **Fig. 2** shows the block diagram of fuzzy controller. The difference *e* between the step and the output feedback is passed as an input into fuzzy logic controller together to the change in error *de*. The output of the fuzzy logic controller serves as an input to the transfer function block. The membership functions parameters are optimized through a search algorithm based on GA. This technique assures that at least a good local optimum can be discovered. Because GA are based on the survival principle of the fittest, it is necessary to establish a fitness function which provides a performance measure of tuning parameters. Such function can be expressed through the Equation 6 and 7:

$$f(x) = \exp(-x) \tag{6}$$

Where:

$$x = \sum_{i=1}^{n} e(i)^2 \tag{7}$$

and n is the number of iterations. In this way, the error e is reduced at minimum. The variables to optimize are four for the first and seventh membership function (trapezoidal functions) and three for the others five membership functions (triangular functions). Because there are two fuzzy inputs and one fuzzy output with seven membership functions, the number of variables to optimize is 69.

The optimization algorithm works as follows:

Step 1: Initialize the variables to optimize.
Step 2: Compute randomly the slope parameters and establish the termination criteria.
Step 3: If it is achieved the termination criteria, the genetic procedure is stopped and go to Step 6.
Step 4: Implement the genetic operations as crossover, mutation and selection (Ivakpour, 2006).
Step 5: Repeat the steps 3-4.
Step 6: Print the optimal values of slope parameters.

After 20 generations, the optimal fuzzy sets of **Fig. 3-5** are obtained. The optimized fuzzy controller uses the Mamdani inference method and the centroid defuzzification technique.

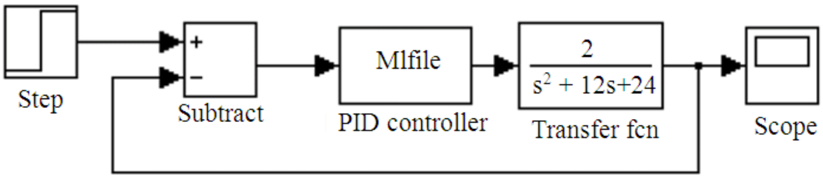

Fig. 1. PID controller block diagram

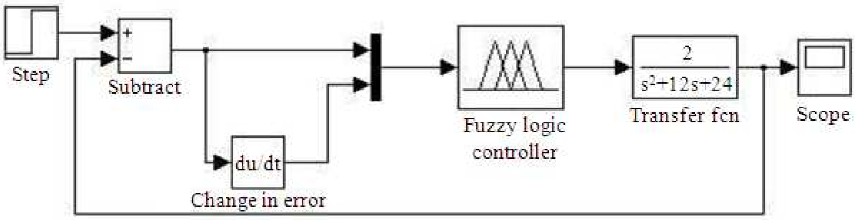

Fig. 2. Fuzzy controller blocks diagram

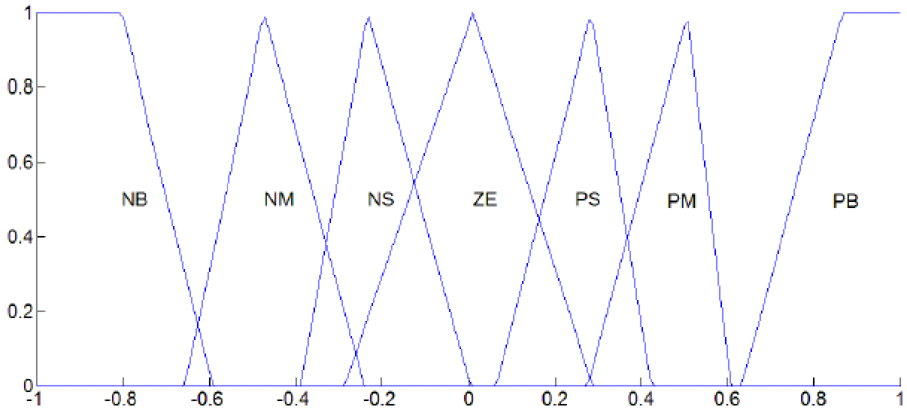

Fig. 3. Optimized membership functions of input *e*

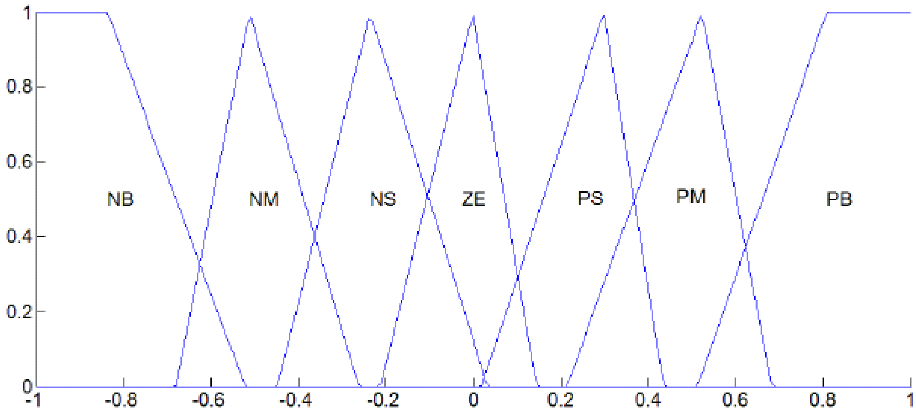

Fig. 4. Optimized membership functions of input *de*

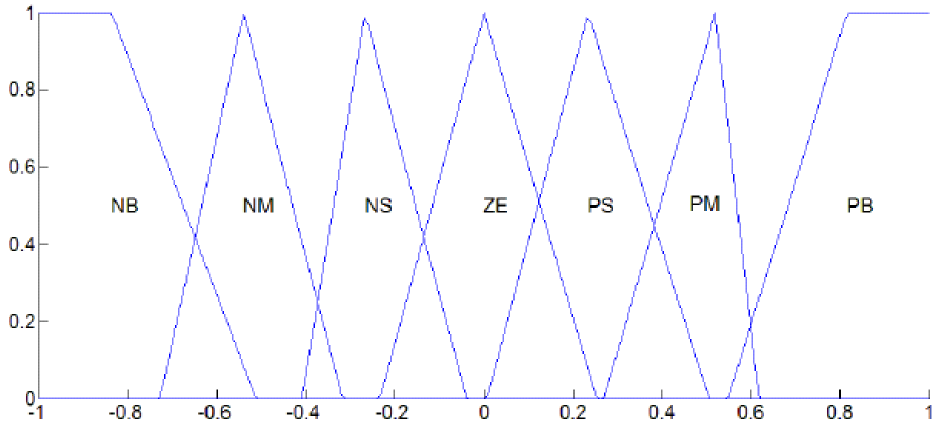

Fig. 5. Optimized membership functions of output

Table 1. Fuzzy rules

e\de	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

Many authors (Khan *et al*., 2008; Kumar and Garg, 2004; Chegeni *et al*., 2007; Pelusi, 2011b) have proposed GA techniques to achieve optimal fuzzy rules. Therefore, the above optimization algorithm is also used to find the fuzzy rules with the higher weight. In fact, to improve the control, it is very important discovering the rules which give the smallest timing control parameters. For this task, we consider the (7) as fitness function and apply the described optimization algorithm.

1.3. Neural Networks Application

In order to improve the control performances of genetic-fuzzy controller, a suitable optimization of fuzzy rule is proposed. For this task, we consider a data-driven intelligent controller based on adaptive features. A neural-fuzzy network can self-adjust the parameters of the fuzzy rules using neural-based learning algorithms. Our idea is to tune the rules weights considering the rules that give good timing performances. The fuzzy rules weights are tuned with the constraint of achieving small values of settling and rise time. Our control system has self-tuning capabilities and requires an initial rule base (**Table 1**) to be specified prior to training.

Generally, the design of neural networks for specific applications is a test and error process. This process sometime depends mainly on previous experience in similar applications. Moreover, the performances and the cost of a neural network are joined to neurons number, net architecture and learning algorithms. Some works (Fiszelew *et al*., 2007; Chad, 2005) are focused in the development of methods for the evolutionary design of architectures to search optimal configurations of neural networks.

Among the main training techniques there is the back-propagation algorithm. This training procedure is used in many applications (Amini, 2008; Chang and Shih, 2002). Back-propagation involves minimization of an error function which is accomplished by performing gradient descent search on the error surface.

In order to define the layers number and the neurons number for each layer, trial and error procedures are used. The designed neural network has three layers: the first one has 2 neurons (equal to inputs number), the hidden layer has 7 neurons and the output layer has 49 neurons (**Fig. 6**). The training technique used is back-propagation. The network has been designed through Neural Network Toolbox of MATLAB.

Difficult task is the definition of a suitable training set. The training sample of our neural network is characterized by the inputs *e* and *de* and 49 rules weights values. The training set is obtained as follows. The error *e*, the change in error *de* and the weights are randomly generated and sent to the genetic-fuzzy controller. The weights values with settling and rise time less than the best timing values of genetic-fuzzy controller are extracted. Formally Equation 8 and 9:

$$t_s < t_{sbest} \quad (8)$$

and:

$$t_r < t_{rbest} \tag{9}$$

where, t_s is the settling time of neural-fuzzy controller and t_{sbest} is the best settling time of genetic-fuzzy controller, whereas t_r is the rise time of neural-fuzzy controller and t_{rbest} is the best rise time of genetic-fuzzy controller. The obtained training sample of 1050 patterns is applied to the neural network.

A single presentation of all input vectors to the network is defined as training epoch. The network is then updated according to the results of all the presentations. Training occurs until a maximum number of epochs occurs or the performance goal is met. After 150 epochs and with a goal of 0.05, the performance of neural network is 0.0820087 (**Fig. 8**).

The block diagram of neuro-fuzzy controller is showed in **Fig. 7**. The inputs e and de are sent to the trained neural network which gives the optimal weights for the 49 fuzzy rules. Such tuning parameters are passed to the fuzzy controller together with the error signal e and the change in error de. The output of fuzzy controller tunes the second order plant $G_1(s)$. The difference between the signal reference and the output feedback is passed as an input to the neural network and fuzzy controller. The process restarts with the calculation of new values of the error and change in error.

1.4. Real-Time Optimization Algorithm

The genetic fuzzy controller works on a run time optimization algorithm described before. The new idea is to design a real-time optimization algorithm taking into account time computer problems. To accomplish such task, genetic techniques are again used.

The intelligent procedure performs a stochastic search via iteratively processing populations of solutions in according to fitness. In control applications, the fitness is usually depending on performance measures as settling time and rise time. To design our real-time algorithm, we define the fitness function f as expressed in equation 10:

$$f(x) = \frac{1}{1+x} \tag{10}$$

with $x = \sum_{t=1}^{n} e(t)^2$, where n is the number of iterations. The goal is to reduce the quantity x at minimum, where x is the sum of square errors. The **Fig. 9** shows the block diagram of control system. In order to evaluate the timing performance of structure, we consider the step response of system. The difference between the step and the output feedback is sent to the input of genetic-neuro-fuzzy controller together with the change in error computed by change in error block. Such inputs serve as inputs to the trained neural network which gives the optimal weights of fuzzy rules. At the same time, the GA block gives the optimal MF scaling parameters for given inputs. The output of fuzzy logic controller drives the second order plant defined by (3).

The novelty of this approach is the real-time optimization of MF and weights using respectively GA and NN. The steps of our real-time optimization algorithm are the following:

- Step 1: Initialize the MF scaling parameters. The number of parameters is 69. The population number is 100 and the number of generation is 20. A population of problem solutions is expressed in the form of chromosomes, i.e., strings encoding problem solutions.
- Step 2: Define the range of each MF scaling parameter. This is a delicate phase because there could be undesirable overlapping. Subsequently, compute randomly the scaling parameters and establish the termination criteria.
- Step 3: When it is achieved the termination criteria, the intelligent procedure is stopped and go to Step 9.
- Step 4: Compute the fitness function to select good strings. The fitness function also defines the optimal weights via the trained neural network. The task is to achieve the maximum of $f(x)$ (equation 10).
- Step 5: Implement the selection. The selection process copies parent chromosomes into a tentative new population. The number of copies reproduced for the next generation by an individual is expected to be directly proportional to its fitness value.
- Step 6: Compute the crossover. Such genetic procedure recombines genetic material of parent chromosomes to produce offspring on the basis of crossover probability. Let y, z be two chromosomes of length 5. As an example, considering y = 01001 and z = 11010 and onepoint crossover at the fourth point, two new chromosomes y0 = 01010 and z0 = 11001 are produced.
- Step 7: Implement the mutation. The mutation selects a random position of a random string and complements the bit value. For example, if mutation is applied to the third bit of string *y0*, the transformed string becomes 01110.
- Step 8: Repeat the steps 3-7.
- Step 9: Print the optimal values of MF scaling parameters and the weights of fuzzy rules.

The real-time algorithm gives the optimized membership functions shown in **Fig. 10-12**.

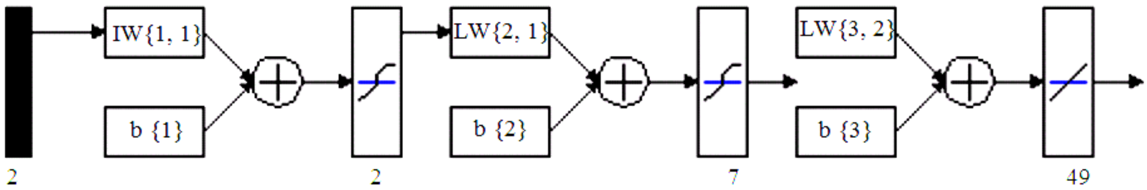

Fig. 6. Neural network architecture

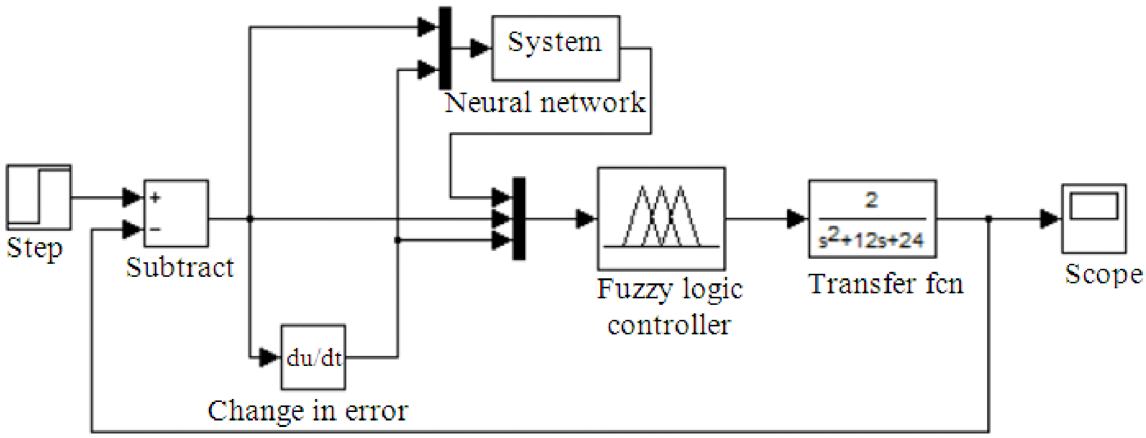

Fig. 7. Neuro-fuzzy controller block diagram

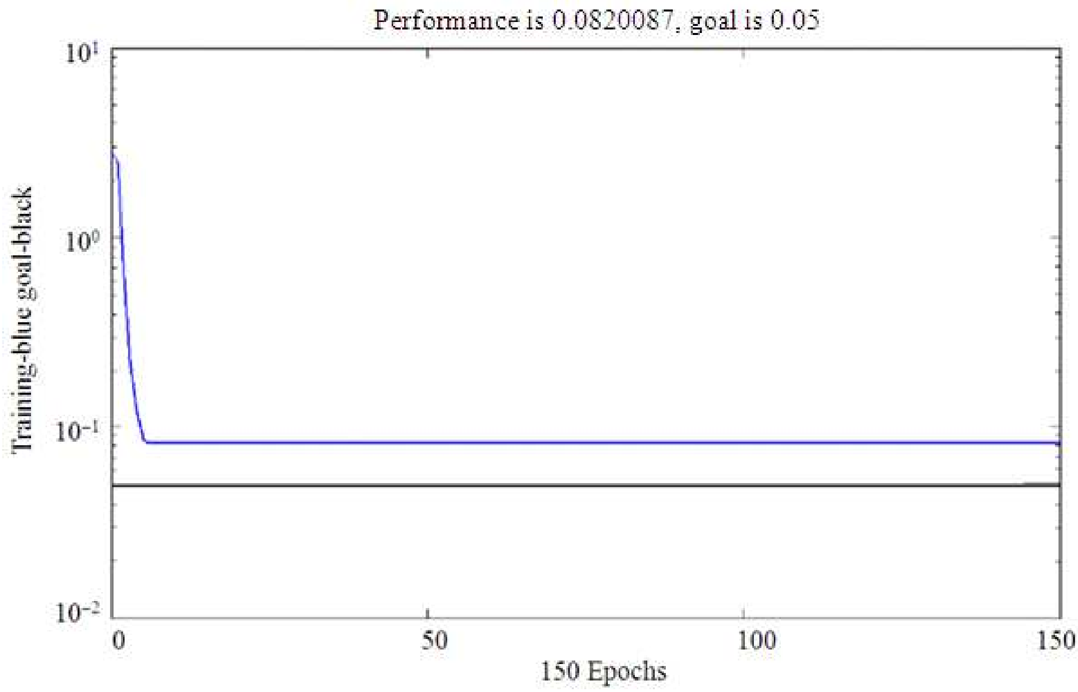

Fig. 8. Neural network training plot

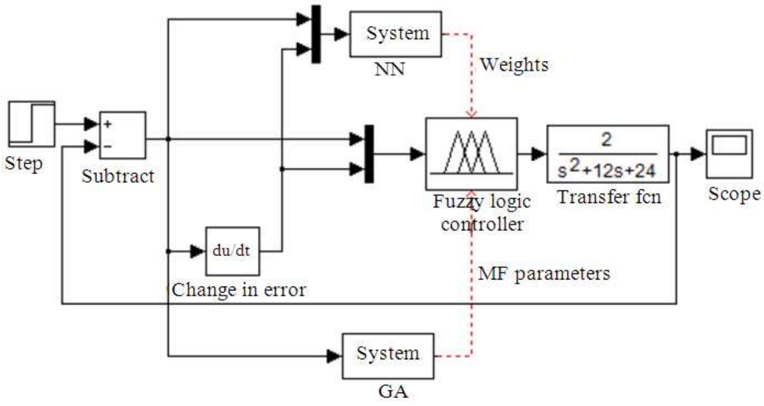

Fig. 9. Genetic-Neuro-Fuzzy controller block diagram

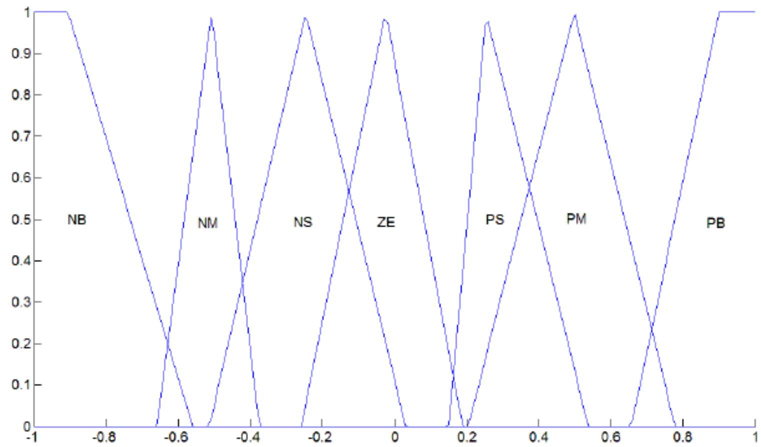

Fig. 10. Optimal membership functions of error obtained with real-time optimization algorithm

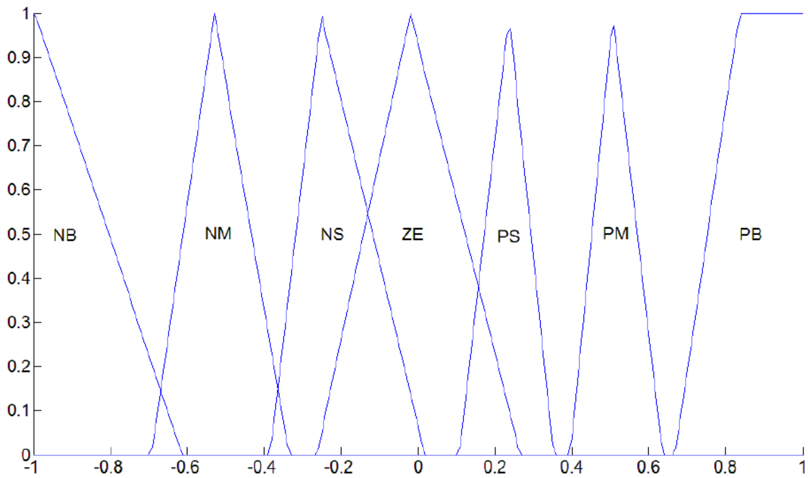

Fig. 11. Optimal membership functions of change-in-error obtained with real-time optimization algorithm

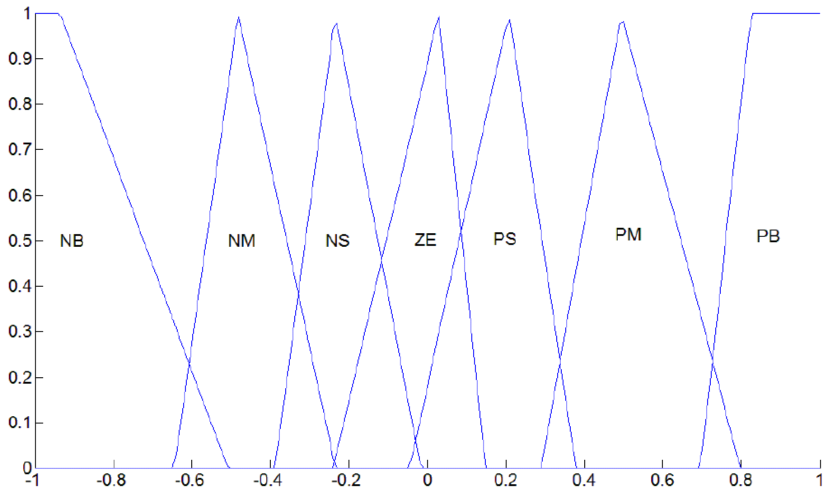

Fig. 12. Optimal membership functions of fuzzy output obtained with real-time optimization algorithm

1.5. Simulation Results

The designed controllers are simulated through the MATLAB. The simulation results for different plants of PID controller are shown in **Table 3**. Our PID controller improves the rise time and the overshoot of (Khan *et al*., 2008). In fact, the rise time of conventional PID controller in (Khan *et al*., 2008) is 0.371s, whereas the PID controller here designed has rise time equal to 0.118s. Moreover, the overshoot of (Khan *et al*., 2008) is 0.6748, whereas our PID controller gives a value of 0.223. The step response of PID controller with plant defined by (3) is shown in **Fig. 13**. Moreover, the controller has shown good robustness performances changing the plant parameters. In **Table 3** are shown the settling time, rise time and the overshoot values of control system for the three different plants $G_1(s)$, $G_2(s)$ and $G_3(s)$. The step response is respectively shown in **Fig. 13-15**.

The genetic-fuzzy controller with optimized MF has better t_s and t_r than fuzzy logic PD controller in (Khan *et al*., 2008). In our work, the settling time is 0.699s versus a value of 0.8735s of fuzzy logic PD controller designed in (Khan *et al*., 2008). Comparing the results, we can note that the rise time has an improvement of above 45% percent respect to (Khan *et al*., 2008). The genetic-fuzzy controller also gives a settling time better than our PID controller (considering $G1(s)$ as transfer function) and zero overshoot. The improvements can be deduced observing PID and genetic-fuzzy controllers step response (**Fig.13-16**). We remind that the genetic fuzzy, neuro-fuzzy and genetic-neuro-fuzzy controllers are applied on plants with transfer function defined by (3).

Better timing results are achieved optimizing the fuzzy rules through the run-time optimization algorithm.. The optimization of rules weights shows that there are five more relevant rules (**Table 4**). These rules have weight greater than 0.9. We can note that with error *e* equal to NB and NM and for all *de* values, the fuzzy rules are characterized by weights less equal than 0.9. Moreover, with e = ZE there are not relevant rules. The genetic fuzzy controller with optimal rules improves the timing values of fuzzy controller with optimal membership functions. In fact, the settling time is 0.436s, versus the value of 0.699s of genetic fuzzy controller with only optimal membership functions. Moreover, also the value of t_r is improved from 0.385s to 0.241s.

Thanks to the constraints (8) and (9) defined in the neuro-fuzzy controller with $t_{sbest} = 0.699s$ and $t_{rbest} = 0.385s$, the genetic-fuzzy controller results are improved. In fact, the adaptive neuro fuzzy approach gives a settling time of 0.423s and a rise time of 0.234s. The step response of neuro-fuzzy controller is shown in **Fig. 17**.

Further improvements are achieved defining a real-time optimization procedure. The designed genetic-neuro-fuzzy controller works in according with the real-time optimization algorithm described before. Because such algorithm adopts Fuzzy Logic, Genetic Algorithms and Neural Networks techniques which run at the same time, computational cost must be considered.

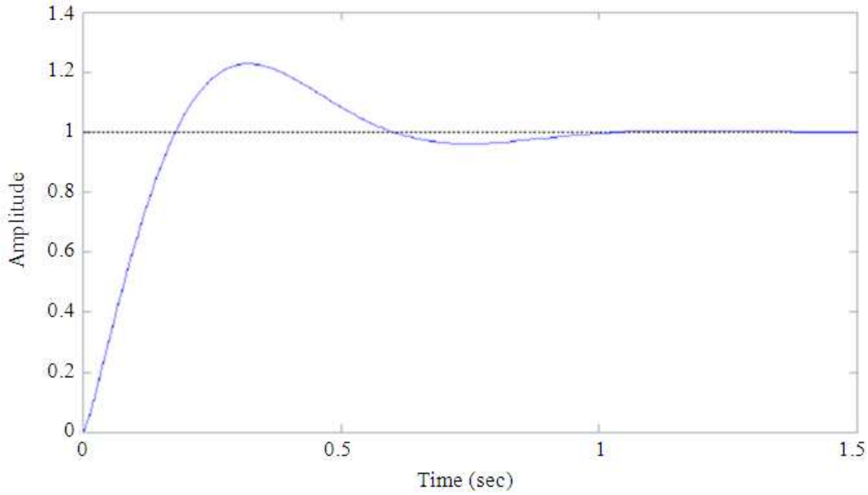

Fig. 13. Step response of PID controller for $G_1(s)$ plant

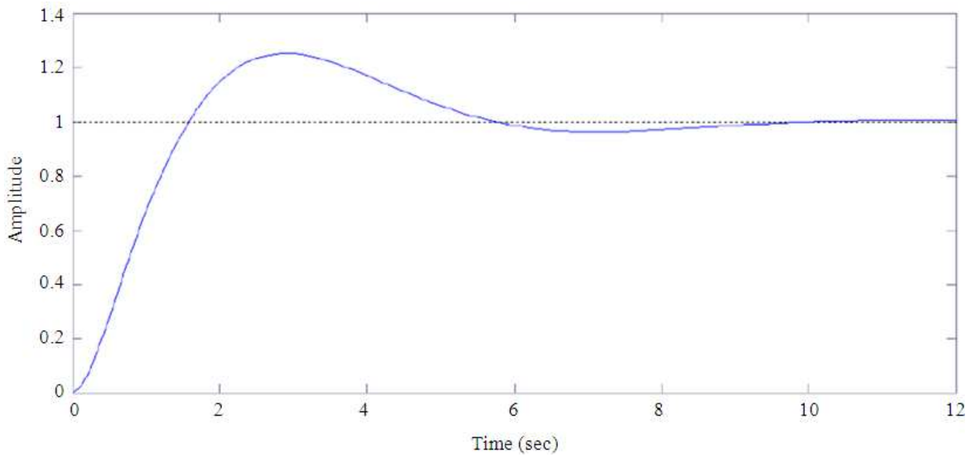

Fig. 14. Step response of PID controller for $G_2(s)$ plant

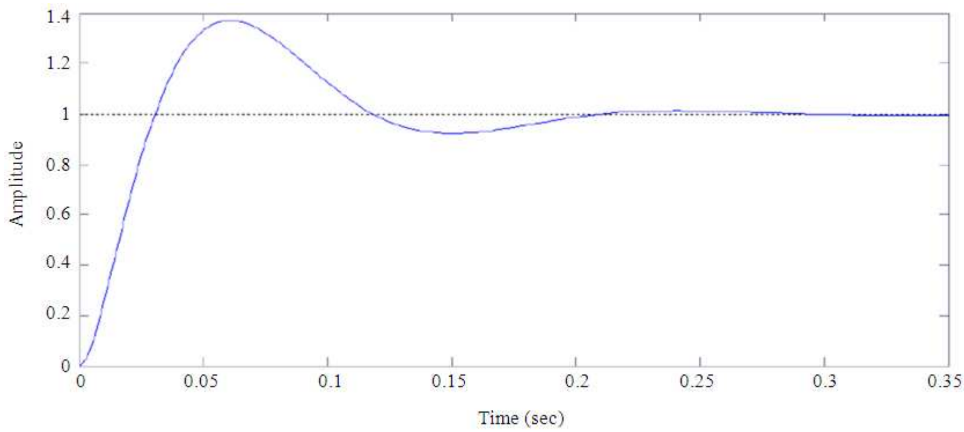

Fig. 15. Step response of PID controller for G3(s) plant

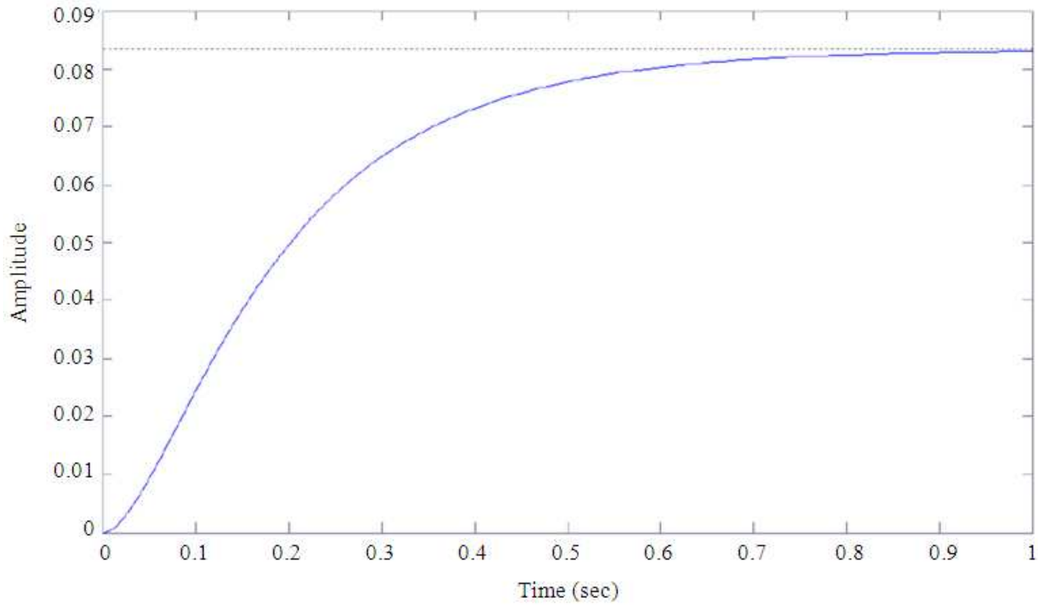

Fig. 16. Step response of genetic-fuzzy controller

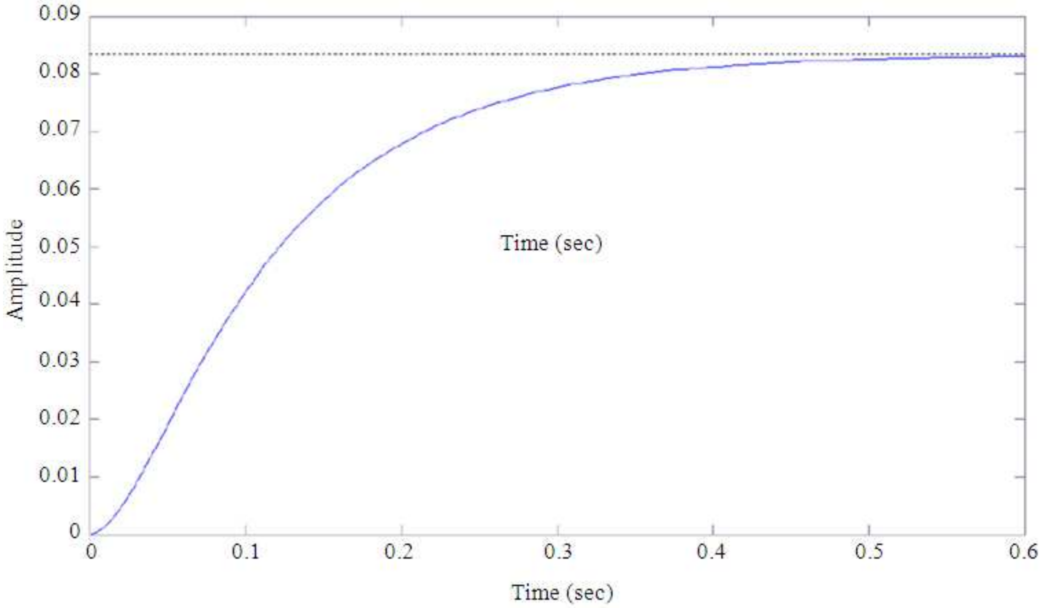

Fig. 17. Step response of neuro-fuzzy controller

We simulate our controller through the MATLAB software and run the optimization algorithm on a 2.5*GHz* CPU speed computer. The algorithm yields the optimal parameters after about 10 h time computer. This is due to the fact that the variables to optimize are 69 and that GA and NN work together. Moreover, the convergence of the algorithm depends on the amplitude of optimization parameters ranges.

Some time computer problems come from fuzzy rules number of the controller. Such number depends on the membership functions number of each fuzzy input. In order to increase the precision of system, a MF number greater than 7 has been considered. However, the results show that increasing the MF number do not improve the controller performances. The optimal membership functions of inputs and output are shown in **Fig. 10-12**. We can note that there are different slopes for each membership function. We underline that the first membership function and the seventh one are characterized by trapezoidal shape. The first membership function of change in error fuzzy input tends to assume a triangular shape rather than trapezoidal one (**Fig. 11**). From the observation of **Fig. 12**, we deduce that the middle membership functions of output (NS, ZE, PS) are narrow. This means that near the zero, the output value must be evaluated more exactly than others output values.

Let <e: de; o> be the fuzzy rule with *e* and *de* as inputs and *o* as output. The optimization results of the fuzzy rules via Neural Networks show that the more relevant rule, i.e., the rule with the greatest weight, is <PB: NS; PS>. Such rule shows a weight value equal to 1. Viceversa, the rule <PB: ZE; PM> assumes zero value. Three rules have weight less than 0.504, whereas the other ones have weights that lie between 0.504 and 0.777 (**Table 2**).

The **Fig. 18** shows the step response of the genetic-neuro-fuzzy controller. We can note that the overshoot is equal to zero, therefore some results of (Khan *et al*., 2008) are improved. In fact, in (Khan *et al*., 2008) the optimized fuzzy logic PD controller yields a not zero overshoot value. However, the main designing problem of the control systems is to reduce the rise time. Sometime, the huge reduction of the rise time causes high overshoot values (Khan *et al*., 2008). The timing results obtained using the real-time optimization algorithm are shown in **Table 5**, where GFC is Genetic Fuzzy Controller, NFC is Neuro Fuzzy Controller and GNFC is Genetic Neuro Fuzzy Controller. Comparing our results with the previous ones, we can note that there are improvements. In fact, our controller yields a settling time equal to 0.276s versus a value of 0.423s obtained with a run-time algorithm. Moreover, the rise time is reduced from 0.234s to 0.153s. Finally, the real-time optimization algorithm assures an improvement of 35% of control performances.

The good values of settling and rise times given by the optimized controller in (Khan *et al*., 2008), are obtained at the expense of overshoot value. The optimized fuzzy logic PD controller shows a settling time equal to 0.2526*s* and a rise time equal to 0.1559*s* with a not equal zero overshoot. Therefore, the other significant result of our research is that with zero overshoot, also settling time and rise time of optimized controller in (Khan *et al*., 2008) are improved (**Table 5**).

Finally, by comparing the results of control systems performances, we conclude that the genetic-neuro-fuzzy and neuro-fuzzy controllers produces a more desirable performance when compared with PID and genetic-fuzzy controllers.

Table 2. Optimized fuzzy rules weights

e\de	NB	NM	NS	ZE	PS	PM	PB
NB	0.572	0.473	0.446	0.553	0.588	0.693	0.505
NM	0.693	0.744	0.661	0.650	0.550	0.714	0.670
NS	0.706	0.777	0.591	0.577	0.634	0.678	0.668
ZE	0.686	0.640	0.652	0.608	0.535	0.561	0.407
PS	0.728	0.572	0.653	0.627	0.666	0.563	0.653
PM	0.612	0.726	0.588	0.634	0.635	0.518	0.653
PB	0.504	0.617	1.000	0.000	0.676	0.573	0.694

Table 3. Settling time and rise time of PID controller for different plants

	G_1(sec)	G_2(sec)	G_3 (sec)
t_s (sec)	0.846	8.520	0.194
t_r (sec)	0.118	1.110	0.022
Overshoot	0.223	0.251	0.369

Table 4. Fuzzy rules with weight greater than 0.9

e/de	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

Table 5. Timing performance of PID controller, genetic-fuzzy controller and neuro-fuzzy controller

	GFC MF-opt	GFC MF,W-opt	NFC Run-time	GNFC Real-time
t_s (sec)	0.699	0.436	0.423	0.276
t_r (sec)	0.385	0.241	0.234	0.153

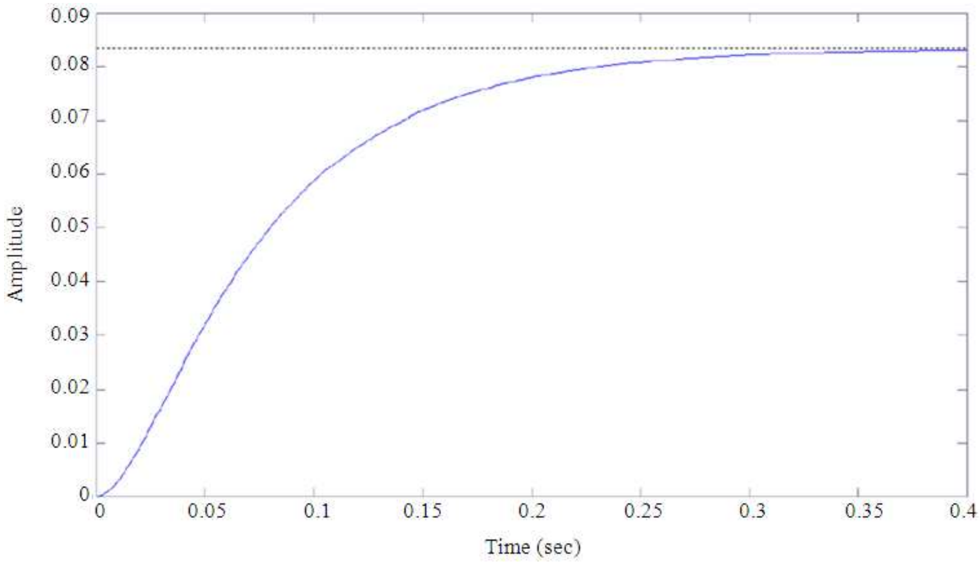

Fig. 18. Step response of Genetic-Neuro-Fuzzy controller

2. CONCLUSION

Different approaches have been presented for the control of second order plants. The first approach employs a PID controller based on Ziegler-Nichols tuning technique to control second order systems. The designed PID controller gives good results in terms of rise time. In fact, the simulations shown a value of t_r = 0.118*s*. Moreover the controller has shown a good robustness considering the plant parameters changing. In order to improve the overshoot and the settling time, a fuzzy controller with optimized membership functions is designed. This optimization is accomplished through the application of genetic procedures. The genetic-fuzzy controller gives a good value of settling time and a very good overshoot value. Better results are obtained applying the optimization algorithm also to optimize the fuzzy rules. Another approach is based on the construction of data-driven intelligent controllers able to adjust the weights of fuzzy rules. This task is accomplished through the neural networks application. The neural-fuzzy controller gives good timing parameters improving the control performances of the others approaches. Taking into account the real-time optimization of membership functions with the best weights given by trained neural network, the best settling time and rise time values are achieved. The goodness of such results is obvious if settling time t_s = 0.276s and rise time t_r = 0.153s values are compared with those ones in (Khan *et al*., 2008). In other words, the simulation results show that the proposed approaches improve the control performance of conventional PID and fuzzy logic PD controllers.

The next task will be the application of genetic-fuzzy, neuro-fuzzy and genetic-neuro-fuzzy controllers to plants defined by (4) and (5). To reduce the computational time for the real-time optimization, fuzzy rules with low weight will be identified and removed to make fuzzy controller more compact and transparent. Another task will be design a suitable training sample to improve the training phase of the neural network. A further improvement will consist of optimizing the weights of the neural networks through Genetic Algorithms. Our attempt will be improving the training phase to achieve optimal neural networks.

A future challenge will be the application of the proposed skills on simulators for drum boiler (Prego and Seisdedos, 2011).

3. REFERENCES

Akbarzadeh, M.R.T., K. Kumbla, E. Tunstel and M. Jamshidi, 2000. Soft computing for autonomous robotic systems. Comput. Elect. Eng., 26: 5-32. DOI: 10.1016/S0045-7906(99)00027-0

Allaoua, B., A Laoufi, B. Gasbaoui and A. Abderrahmani, 2009. Neuro-fuzzy DC motor speed control using particle swarm optimization. Department of Electrical Engineering, Bechar University.

Amini, J., 2008. Optimum learning rate in back-propagation n eural network for classification of satellite images (IRS-ID). Scientia Iranica, 15: 558-567.

Astrom, K.J. and T. Hagglund, 2004. Revisiting the Ziegler-Nichols step response method for PID control. J. Process Control, 14: 635-650. DOI: 10.1016/j.jprocont.2004.01.002

Cao, H., G. Si, Y. Zhang, X. Ma and J. Wang, 2008. Load control of ball mill by a high precision sampling fuzzy logic controller with self-optimizing. Asian J. Control, 10: 621-631. DOI: 10.1002/asjc.63

Chad, A.W., 2005. Genetically evolving optimal neural networks. Pennsylvania State University.

Chang, P. and J.S. Shih, 2002. The application of back propagation neural network of multi-channel piezoelectric quartz crystal sensor for mixed organic vapours. Tamkang J. Sci. Eng., 5: 209-217.

Chegeni, A., A. Khoei and K. Hadidi, 2007. Improved genetic algorithm-based optimization of fuzzy logic controllers. Proceedings of the 1st Joint Congress on Fuzzy and Intelligent Systems, Aug. 29-31, Ferdowsi University of Mashhad, Iran.

Cho, S.B., 2002. Fusion of neural networks with fuzzy logic and genetic algorithm. Integr. Comput. Aided Eng., 9: 363-372.

Chopra, S., R. Mitra and V. Kumar, 2005. Fuzzy controller: Choosing an appropriate and smallest rule set. Int. J. Comput. Cognition, 3: 73-79.

Corcau, J.I. and E. Stoenescu, 2007. Fuzzy logic controller as a power system stabilizer. Int. J. Circ. Syst. Signal Proc., 1: 266-273.

Desborough, L.D. and R.M. Miller, 2002. Increasing customer value of industrial control performance monitoring-Honeywell's experience. Chemical Process Control.

Fiszelew, A., P. Britos, A. Ochoa, H. Merlino and E. Fernandez *et al*., 2007. Finding optimal neural network architecture using genetic algorithms. Res. Comput. Sci., 15: 24-27.

Huang, S.J. and M.T. Su, 2007. Gain tuning fuzzy controller for an optical disk drive. Int. J. Elect. Comput. Eng., 2: 440-445.

Ivakpour, J., 2006. Genetic algorithm performance.

Khan, S., S.F. Abdulazeez, L.W. Adetunji, A.H.M.Z. Alam and M.J.E. Salami, 2008. Design and implementation of an optimal fuzzy logic controller using genetic algorithm. J. Comput. Sci., 4: 799-806. DOI: 10.3844/jcssp.2008.799.806

Kumar, M. and D.P. Garg, 2004. Intelligent learning of fuzzy logic controllers via neural network and genetic algorithm. Proceedings of the Japan-USA Symposium on Flexible Automation, Jul. 19-21, Colorado, Denver, pp: 1-8.

Leng, G., T.M. McGinnity and G. Prasad, 2006. Design for self-organizing fuzzy neural networks based on genetic algorithms. IEEE Trans. Fuzzy Syst., 14: 755-766. DOI: 10.1109/TFUZZ.2006.877361

Meza, J.L., R. Soto and J. Arriaga, 2009. An optimal fuzzy self-tuning PID controller for robot manipulators via genetic algorithm. Proceedings of the 8th Mexican International Conference on Artificial Intelligence, Nov. 9-13, IEEE Xplore Press, Guanajuato, pp: 21-26. DOI: 10.1109/MICAI.2009.34

Munasinghe, S.R., M.S. Kim and J.J. Lee, 2005. Adaptive neurofuzzy controller to regulate UTSG water level in nuclear power plants. IEEE Trans. Nuclear Sci., 52: 421-429. DOI: 10.1109/TNS.2004.842723

Pelusi, D., 2011a. Genetic-neuro-fuzzy controllers for second order control systems. Proceedings of the UKSim 5th European Symposium on Computer Modeling and Simulation, Nov. 16-18, Madrid, Spain, pp: 12-17.

Pelusi, D., 2011b. On designing optimal control systems through genetic and neuro-fuzzy techniques. Proceedings of the IEEE International Symposium on Signal Processing and Information Technology, Dec. 14-17, IEEE Xplore Press, Bilbao. pp: 134-139. DOI: 10.1109/ISSPIT.2011.6151547

Pelusi, D., 2011c. Optimization of a fuzzy logic controller using genetic algorithms. IEEE International Conference on Intelligent Human-Machine Systems and Cybernetics, Aug. 26-27, IEEE Xplore Press, Zhejiang, pp: 143-146. DOI: 10.1109/IHMSC.2011.105

Pelusi, D., 2012. Improving settling and rise times of controllers via intelligent algorithms. Proceedings of the 14th International Conference on Computer Modelling and Simulation, Mar. 28-30, IEEE Comprter Society, Cambridge, pp: 187-192.

Prego, J.J.G. and L.V. Seisdedos, 2011. Tailor-made small simulator for a drum boiler control based on linear techniques. Proceedings of the IEEE 16th Conference on Emerging Technologies and Factory Automation, Sept. 5-9, IEEE Xplore Press, Toulouse, pp: 1-4. DOI: 10.1109/ETFA.2011.6059211

Saridakis, K.M., A.C. Chasalevris, A.J. Dentsoras and C.A. Papadopulos, 2006. Fusing neural networks, genetic algorithms and fuzzy logic for diagnosis of cracks in shafts. Proceedings of the 2nd Intelligent Production Machines and Systems Virtual International Conference, Jul. 3-14, Elsevier, pp: 332-337. DOI: 10.1016/B978-008045157-2/50061-4

Shamusuzzoha, M. and S. Skogestad, 2010. The setpoint overshoot method: A simple and fast closed-loop approach for PID tuning. J. Process Control, 20: 1220-1234. DOI: 10.1016/j.jprocont.2010.08.003

Vaishnav, S.R. and Z.J. Khan, 2007. Design and performance of PID and fuzzy logic controller with smaller rule set for higher order system. Proceedings of the Word Congress on Engineering and Computer Science, Oct. 24-26, San Francisco, USA.

Xue, D., D.P. Atherton and Y. Chen, 2007. Linear Feedback Control: Analysis and Design with MATLAB. 1st Edn., SIAM, Philadelphia, ISBN-10: 0898718627, pp: 354.