COMPARISON BETWEEN PID CONTROLLERS FOR GRYPHON ROBOT OPTIMIZED WITH NEURO-FUZZY SYSTEM AND THREE INTELLIGENT OPTIMIZATION ALGORITHMS

Somayyeh Nalan Ahmadabad[1] and Maryam Kouzehgar[2] and Fatemeh Masoudnia[3]

[1,3]Faculty of Sofian Islamic Azad University Tabriz, Iran
2Control Engineering Department Faculty of Electrical & Computer Engineering Tabriz, Iran

ABSTRACT

In this paper three intelligent evolutionary optimization approaches to design PID controller for a Gryphon Robot are presented and compared to the results of a neuro-fuzzy system applied. The three applied approaches are artificial bee colony, shuffled frog leaping algorithm and particle swarm optimization. The design goal is to minimize the integral absolute error and reduce transient response by minimizing overshoot, settling time and rise time of step response. An Objective function of these indexes is defined and minimized applying the four optimization methods mentioned above. After optimization of the objective function, the optimal parameters for the PID controller are adjusted. Simulation results show that FNN has a remarkable effect on decreasing the amount of settling time and rise-time and eliminating of steady-state error while the SFL algorithm performs better on steady-state error and the ABC algorithm is better on decreasing of overshoot. On the other hand PSO sounds to perform well on steady-state error only. In steady state manner all of the methods react robustly to the disturbance, but FNN shows more stability in transient response.

KEYWORDS

Robotics, PID controller, Gryphon, dynamic equations, shuffled frog leaping, artificial bee colony, particle swarm optimization, neuro-fuzzy system

1. INTRODUCTION

PID controller (proportional-integral-derivative controller) benefiting from major advantages such as simple structure and wide-domain performance, is one of the most frequently used controllers. There are three parameters, on the basis of which, the PID controller is defined, i.e. proportional gainK_p, integral gainK_i, and derivative gain K_d. The traditional procedure to assign these parameters is a sophisticated and time-consuming try and error process. Thus several new methods are recommended to reduce the complexity of tuning PIDs. These methods, having the biological or social inspiration as a common theme, are accomplished within evolutionary

DOI : 10.5121/ijctcm.2013.3604

algorithms such as binary and continuous genetic algorithms [1], ant colony optimization [2], continuous and discrete particle swarm optimization [3], [4] and different types of honey bee colony algorithms [5],[6].

In this approach, working on Gryphon robot which includes 5 joints, we must design five PID controllers; each for one joint. Thus we must first verify the dynamic equations for Gryphon robot. Then we will formulate the PID controller designation problem as an optimization problem with an objective function which is supposed to be minimized by adjusting the four performance indexes, i.e. the maximum overshoot, the rise time, the settling time and the integral absolute error of step response. Finally we will minimize the defined objective function for each joint by applying shuffled frog leaping (SFL) algorithm, artificial bee colony (ABC) algorithm, particle swarm optimization (PSO) and neuro-fuzzy system (FNN).

The paper is organized in 6 sections to illustrate the case. Section 2 includes a brief description of the system under-investigate and presents the verification of the dynamic equations of the robot. Section 3 reviews the utilized intelligent algorithms and FNN and in section 4 the problem formulation is explained. Section 5 presents the simulation results and analyses them. Consequently section 6 concludes the paper.

2. A BRIEF DESCRIPTION OF THE SYSTEM

As shown in Figure 1, Gryphon is a robot with 5 revolute joints. Thus this robot benefits from 5 degrees of freedom. The first three joints which are called shoulder elbow and wrist respectively, are supposed to determine the position of the end-effecter while the last two joints are responsible for the orientation of the end-effecter. Fast and smooth movement while maintaining the high precision is one of the major characteristics of this robot. This robot is controlled by four micro-processors, one of which is to determine the position of the axis, and two are motor controllers and the last one is supposed to support the others and communicate with the host computer. Each joint is moved by a stepper motor while there is feedback on the corresponding encoder to realize the closed-loop control. In this robot the utilized gear ratio is high enough to assume all joints independent from one another [7].

As with [8], we obtained the dynamic equations of all five joints. Table 1 shows transfer functions of all five joints that are obtained from the so-called dynamic equations, adding the process of obtaining which sounds to need a vast space for complicated equations and that is why we have avoided them here. In the transfer functions in Table 1, α stands for the amount of disturbance exerted on the system. For an undisturbed system α is set to zero.

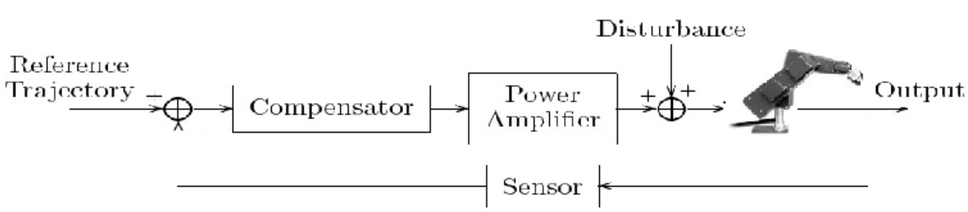

Figure 1: Joint model for Gryphon

3. THE UTILIZED INTELLIGENT ALGOTITHMS

3.1 The Shuffled Frog Leaping Algorithm (SFL)

In the SFL, the population consists of a set of frogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process [9]. The local search and the shuffling processes continue until defined convergence criteria are satisfied [10].

Tble 1. Transfer Functions of Gryphon Joints

Joint Number	Transfer Function
Joint 1	$\frac{-.0005s+.1012}{.0448s^2+.5581s-.0126+\alpha}$
Joint 2	$\frac{-.0031s+.1012}{.0221s^2+.6761s-.0125+\alpha}$
Joint 3	$\frac{.0005s+.1013}{.0210s^2+.3968s-.0130+\alpha}$
Joint 4	$\frac{-.0022s+.1005}{.0096s^2+1.4940s-.0051+\alpha}$
Joint 5	$\frac{.0276s+.1005}{.0539s^2+1.4210s-.0059+\alpha}$

An initial population of P frogs is created randomly. For S-dimensional problems (S variables), a frog i is represented as $X_i = (x_{i1}, x_{i2}, \dots, x_{is})$. Afterwards, the frogs are sorted in a scending order according to their fitness. Then, the entire population is divided into m memeplexes, each containing n frogs (i.e. $P = m \times n$). In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog (m) goes to the mthmemeplex, and frog (m+1) goes back to the first memeplex, etc.

Within each memeplex, the frogs with the best and the worst fitnesses are identified as X_b and X_w, respectively. Also, the frog with the global best fitness is identified as X_g. Then, a process is applied to improve only the frog with the worst fitness (not all frogs) in each cycle. Accordingly, the position of the frog with the worst fitness is adjusted as follows:

$$\text{Change in frog position } (D_i) = \text{rand} \times (X_b - X_w) \quad (1)$$

$$\text{New position } X_w = \text{current position } X_w + D_i\,;$$
$$D_{max} \geq D_i \geq -D_{max} \qquad (2)$$

Where rand is a random number between 0 and 1; and D_{max} is the maximum allowed change in a frog's position.

If this process produces a better solution, it replaces the worst frog. Otherwise, the calculations in Eqs (1) and (2) are repeated but with respect to the global best frog (i.e. X_g replaces X_b). If no improvement becomes possible in this case, then a new solution is randomly generated to replace that frog. The calculations then continue for a specific number of iterations [8]. Accordingly, the main parameters of SFL are: number of frogs P; number of memeplexes; number of generation for each memeplex before shuffling; number of shuffling iterations; and maximum step size.

3.2 The Artificial Bee Colony Algorithm (ABC)

Karaboga analyzes the foraging behavior of honey bee swarm and proposes a new algorithm simulating this behavior for solving multi-dimensional and multi-modal optimization problems, called Artificial Bee Colony (ABC) [5]. The main steps of the algorithm are:

1) send the employed bees onto the food sources and determine their nectar amounts.
2) calculate the probability value of the sources with which they are preferred by the onlooker bees.
3) stop the exploitation process of the sources abandoned by the bees.
4) send the scouts into the search area for discovering new food sources, randomly.
5) memorize the best food source found so far.

In the algorithm, an artificial bee colony consists of three groups of bees: employed bees, onlookers and scouts. Employed bees are associated with a particular food source which they are currently exploiting. They carry the information about this particular source and share this information with a certain probability by waggle dance. Unemployed bees seek a food source to exploit. There are two types of unemployed bees: scouts and onlookers. Scouts search the environment for new food sources without any guidance. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. On the other hand onlookers observe the waggle dance and so are placed on the food sources by using a probability based selection process. As the nectar amount of a food source increases, the probability value with which the food source is preferred by onlookers increases, too. In the ABC algorithm the first half of the colony consists of the employed bees and the second half includes the onlookers. For every food source, there is only one employed bee. Another issue that is considered in the algorithm is that the employed bee whose food source has been exhausted by the bees becomes a scout. In other words, if a solution representing a food source is not improved by a predetermined number of trials, then the food source is abandoned by its employed bee and the employed bee is converted to a scout.

3.3 Particle Swarm Optimization

PSO is an evolutionary computation technique developed by Kennedy and Eberhart in 1995[11]. In this algorithm each solution is regarded as a particle which is defined by its position and the fitness calculated based on the position. Also

There is a speed vector which specifies the direction in which the particle is moving. Other parameters which are determined during the run, are as follows:

- P_{best}-the personal best: Each particle remembers the best position that it has visited so far. This best position is known as P_{best}.

- G_{best}- the global best: The best of all positions explored by all particles.
- N_{ibest}-the neighborhood best: For each particle $N_{ibest}$is the best position of the particles in the neighborhood of i^{th} particle.

To apply the algorithm, first the particles are distributed randomly in the search space. Then the cost function is evaluated for each particle, afterwards P_{best} , N_{ibest} , $G_{best}$are updated. At the end by applying (3) the positions and speeds are updated. Eventually the algorithm checks the stopping criteria and loops until they are satisfied.

$$\begin{gathered} V_i = wV_i + c_1 \text{rand}(N_{ibest} - X_i) + c_2 \text{rand}(P_{best} - X_i) \\ X_i = V_i + X_i \qquad (3) \end{gathered}$$

Where $V_i$is the velocity of i^{th} particle, X_i shows the position of the i^{th}particle, w is the inertia weight, utilized to avoid premature convergence and usually is set to 0.5. Separate random numbers are generated to accelerate through $P_{best}$and N_{ibest} . $c_1$andc_2 are acceleration constants both equal to 2; these parameters change the amount of tension in the system, i.e. weighting the stochastic acceleration terms that pull the particle towards P_{best} or N_{ibest} . In some iterations N_{ibest} may be substituted by G_{best}.

Particle velocities are clamped to a maximum value of V_{max}, thus serve a constraint on the global exploration ability. $V_{max}$is routinely adjusted at about 10-20% of the dynamic range of the variable on each dimension [3].

3.4 The Nero-Fuzzy System (FNN)

Neural Networks, using a system based on human brain structure, have the ability to learn how to face new challenges. These types of systems are able to adapt themselves to learn how to become compatible with new conditions which have not formerly experienced. Fuzzy logic, based on rules that includes a knowledgebase, is formed byfuzzy rules of "if - then". These rules are made of simple and understandable words. The experts who know the system behavior define these rules by means of natural language. Through combination of Neural Networks and Fuzzy logic Neuro-Fuzzy systems are accessible. Any Neuro-Fuzzy system is a Neural Network which learns how to classify data using Fuzzy rules and Fuzzy sets.

A Neuro-Fuzzy network is a network with feedback comprising five layers. These layers are:

1) Input layer.
2) Input fuzzymembership functions.
3) Fuzzy rules.
4) Output fuzzymembership functions.
5) Output layer.

Figure 2 shows a general view of this network. In this figure the network has two deterministic inputs, T and H and it produces a deterministic output, Q. In a system, number of inputs and outputs are changeable according to our requirement. In this paper there are four inputs(e,ė) and three outputs(K_p , K_i , K_d)

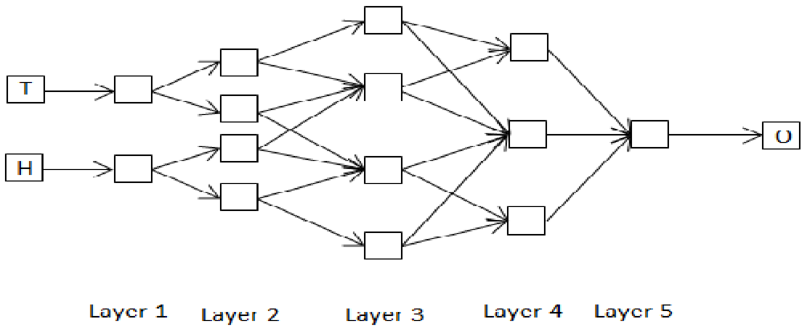

Figure 2. A general view of a nero-fuzzy network with five layers.

First layer is only supposed totransmitthe deterministic inputs to the second layer of network. In the second layer, deterministic inputs turn tofuzzy values (in this paper, we used triangular membership functions for fuzzification). Third layer includes aset of fuzzy rules. In this layer, each neuron is a fuzzy rule. System can be set with some initial fuzzy rules and the network adjusts the weights in a way which gives the best response. In some cases it is possible to start the system without initial rules. In such cases system learns the rules by itself and adjusts the weights based on them [12].

Forth layer in the network includes the neurons presenting the membership functions for different and possible outputs of fuzzy rules. Fifth and the last layer combines and defuzzyfiesdifferent tuned outputs.

FNN method hereissimulated by ANFIS, thetrain data for ANFIS was obtained from several hand-tunings of PID parameters[13]. Train data loaded to ANFIS and was use for FNN learning. After learning, by loading ANFIS into simulation environment, the PID parameters will be set.

4. PROBLEM FORMULATION TO DESIGN PID CONTROLLERS

The transfer function of the PID controller is considered as follows.

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s \qquad (4)$$

Where K_p, $K_i$and K_d are proportional, integral and derivative gains respectively. In the time domain the characteristics of step response such as rise timet_r, overshootM_p, settling time t_s and steady state errorE_{ss} are considered as a performance criterion for PID controller. We use a function of these indexes as the objective function which is illustrated by Equation (5)[14].

$$f(k) = \left(1 - e^{-\beta}\right)\left(M_p + E_{ss}\right) + e^{-\beta}\left(t_s - t_r\right) \qquad (5)$$

Where K is a vector whose elements are the PID controller gains. With which the step response of the system is simulated and the values of Mp, Ess, ts and tr are obtained to evaluate f(k).

$$K = [K_d K_p K_i] \qquad (6)$$

And β is a weighting factor whose value effects the four indexes. This value, if set to be smaller than 0.7, results in reduction of t_r and t_s. Otherwise, if set to values larger than 0.7, end in a decrease in E_{ss} and M_p while setting β=0.7 will cause the four parameters to effect similarly on objective function.

On both of the compared algorithms, the vector K is defined as a member of population. The algorithms start from a random population. During each iteration of the algorithm, the elements of each member of population are substituted in the controller transfer function $G_c(s)$ and the step response is calculated for each member of the population, on the basis of which, t_r, t_s, E_{ss} and M_p are calculated and substituted in the defined objective function. The goal of all the algorithms is to minimize the objective function through the strategies illustrated above.

5. SIMULATION RESUTLS

5.1. Parameter Evaluation on the Algorithms

On both of the algorithms, all of the controller parameters are restricted to the interval [0, 30].
For both of the algorithms, 30 runs , with 5000 function evaluations included in each run, have been carried out to have the average answer which is more reliable. To determine the robustness of the system, all simulations have been run under the effect of a disturbance factor. The parameter evaluation for each utilized algorithm is as follows.

5.1.1. SFL Algorithm

Population size=50, The number of memeplexes=10, Memetic evolution steps before shuffling=10, maximum number of function evaluation=5000.

5.1.2 ABC Algorithm

Population size=20, The number of employee bees=0.3 population size, The number of on-looker bees=0.3 population size, The number of scout bees=0.4 population size, maximum number of function evaluation=5000.

5.1.2 PSO Algorithm

Population size=30, neighbourhood size=6, w=0.8, c1=c2=2, Maximum number of function evaluation=5000.

5.2. Parameter Evaluation on FNN

FNN was generated by grid partition with 3 membership functions and was trained by hybrid method [13] .

Number of MFs=3, Input MF type=gbellmf, Output MF type=constant

5.3. The results

Tables 2, 3 and 4 respectively present the simulation results of SFL, ABC and PSO algorithm for all of the five joints with β=0.7. Table 5 indicates the same values for FNN. The step responses for each of the five motors are shown on figures 3 to 22. In these pictures, the graph shown with direct line indicates the response of the undisturbed system while the other two show the disturbed systems with different disturbances (α=0.1 and α= -0.1).

Applying the kp, ki, kd values obtained from the intelligent algorithms, simulation results show that FNN has a remarkable effect on decreasing the amount of settling time and rise-time and eliminating of steady-state error because considering the average value of the 5 joints according to Tables 2 to 5, it is obviously seen in Table 6 that the average settling-time, the average rise-time and the average steady-state error is far less in FNN in comparison to ABC and SFL while the SFL algorithm compared to ABC performs slightly better on steady-state error (0.0006 vs. 0.0009) and the ABC algorithm is better on decreasing of overshoot . On the other hand PSO to some extent performs well only on steady-state error. In steady state manner all of the methods react robustly to the disturbance, but FNN shows more stability in transient response.

Table 2. Simulation Results applying SFL algorithm

Joints	PID Parameters			Step Response Parameters			
	K_d	K_p	K_i	M_p	E_{ss}	t_r	t_s
Joint 1	9.8832	23.4645	6.0537	.0417	.0000	1.3371	6.7864
Joint 2	2.0628	26.3914	14.1972	.0861	.0000	.5354	4.0012
Joint 3	12.2736	24.8578	3.3813	.0253	.0001	1.7084	7.8000
Joint 4	2.2865	22.9050	14.4540	.1038	.0000	.5998	3.8516
Joint 5	1.8861	21.5737	.6692	.0130	.0029	1.1362	5.3366

Table 3. Simulation Results applying ABC algorithm

Joints	PID Parameters			Step Response Parameters			
	K_d	K_p	K_i	M_p	E_{ss}	t_r	t_s
Joint 1	3.1838	22.8034	1.2721	.0172	.0012	1.1903	5.2351
Joint 2	5.5132	29.9019	3.0487	.0196	.0001	1.0494	5.0728
Joint 3	.8966	20.3226	1.1469	.0201	.0013	.9203	7.1673
Joint 4	1.4114	27.2506	1.0234	.0118	.0019	.7417	2.7938
Joint 5	11.7983	24.4511	3.9375	.0287	.0000	1.6203	7.7243

Table 4. Simulation Results applying PSO algorithm

Joints	PID Parameters			Step Response Parameters			
	K_d	K_p	K_i	M_p	E_{ss}	t_r	t_s
Joint 1	7.4025	23.8861	4.4924	.0351	.0000	1.2592	7.2114
Joint 2	3.9689	26.8548	12.0161	.0744	.0000	0.7927	4.9169
Joint 3	10.6848	23.9077	2.9857	.0194	.0001	1.5611	5.8308
Joint 4	1.8521	23.5006	10.1128	.1426	.0000	1.2944	6.8835
Joint 5	1.7355	22.7348	1.9102	.0393	.0008	2.5943	16.9618

Table 5. Simulation Results applying FNN algorithm

Joints	PID Parameters			Step Response Parameters			
	K_d	K_p	K_i	M_p	E_{ss}	t_r	t_s
Joint 1	2.1666	52.5094	8.2394	.0421	.0001	.2822	1.7952
Joint 2	2.1665	52.5157	8.2425	.0422	.0001	.2822	1.7954
Joint 3	2.5001	52.5080	12.7716	.0421	.0000	.2876	2.8411
Joint 4	1.9811	52.5017	15.1483	.0416	.0000	.2665	2.9366
Joint 5	1.3549	38.9996	19.6747	.0839	.0000	.3273	3.3340

Table 6. The Average Values For Comparison

	Average settling-time	Average rise-time	Average steady-state error	Average overshoot
FNN	2.54046 sec	0.28916 sec	0.00004	0.05038
ABC	4.59866 sec	1.1044 sec	0.00090	0.01948
SFL	5.55516 sec	1.06328 sec	0.00060	0.05398
PSO	8.36088 sec	1.50034 sec	0.00018	0.06216

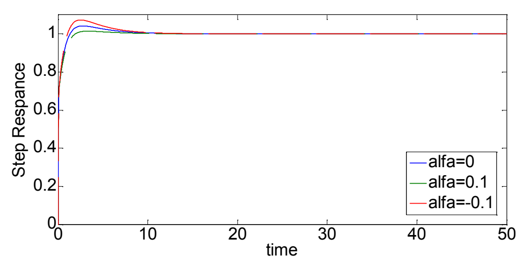

Fig 3. Step responses with SFL applied on motor 1

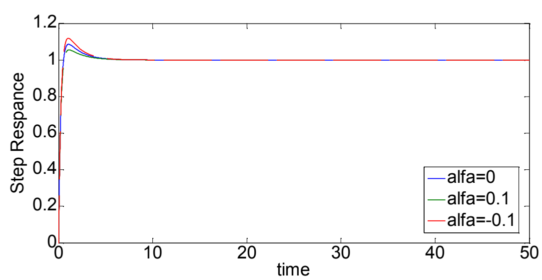

Fig 4. Step responses with SFL applied on motor 2

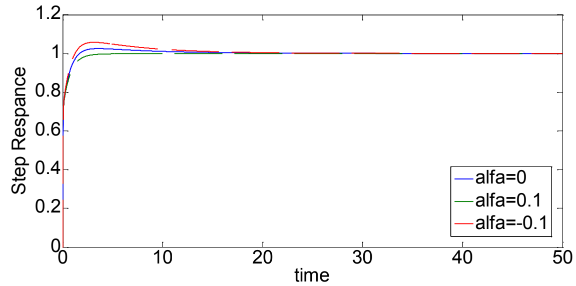

Fig 5. Step responses with SFL applied on motor 3

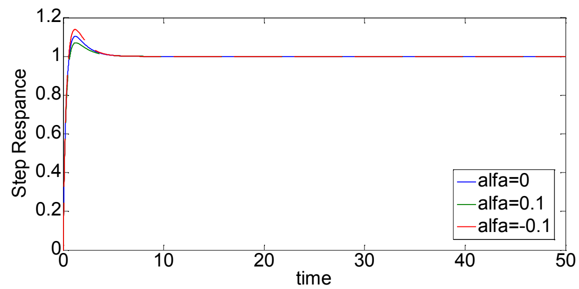

Fig 6. Step responses with SFL applied on motor 4

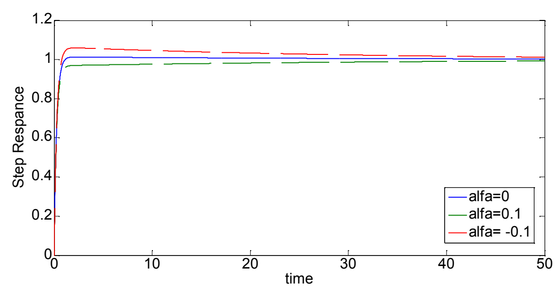

Fig 7. Step responses with SFL applied on motor 5

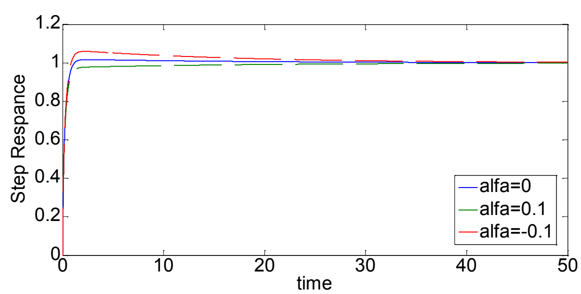

Fig 8. Step responses with ABS applied on motor 1

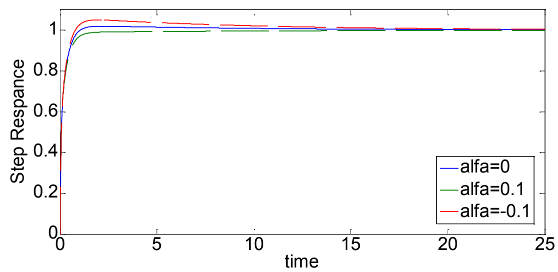

Fig 9. Step responses with ABS applied on motor 2

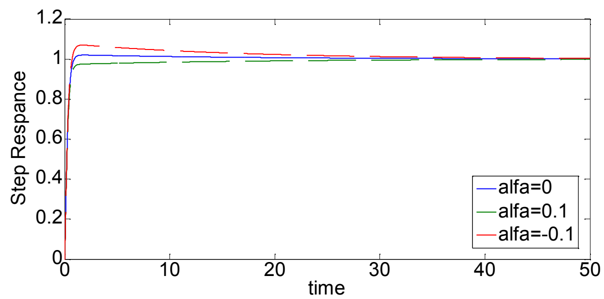

Fig 10. Step responses with ABS applied on motor 3

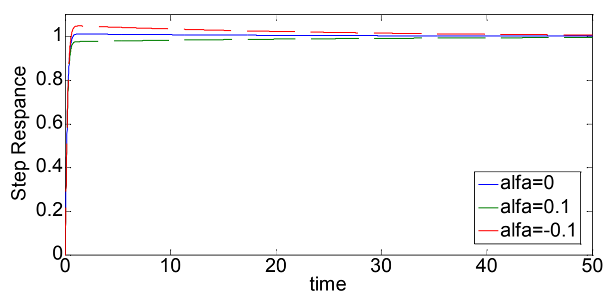

Fig 11. Step responses with ABS applied on motor 4

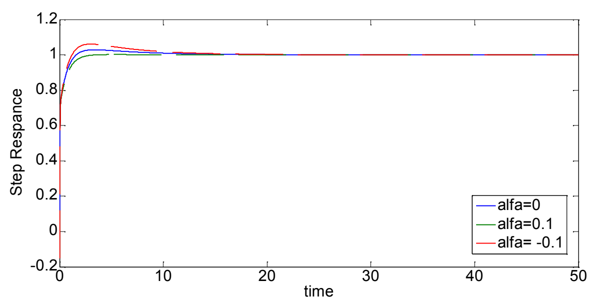

Fig 12. Step responses with ABS applied on motor 5

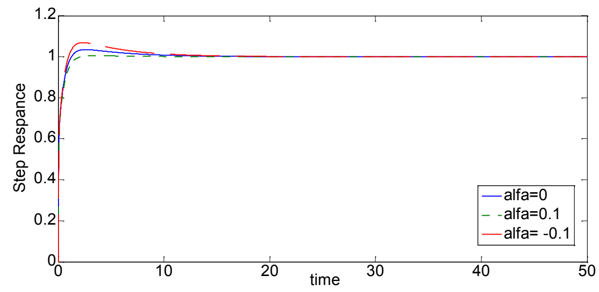

Fig 13. Step responses with PSO applied on motor 1

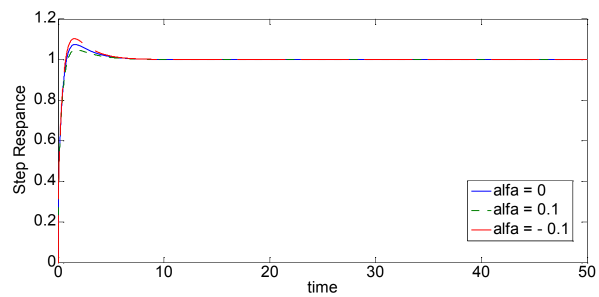

Fig 14. Step responses with PSO applied on motor 2

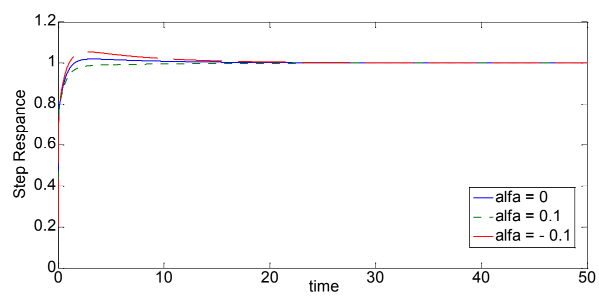

Fig 15. Step responses with PSO applied on motor 3

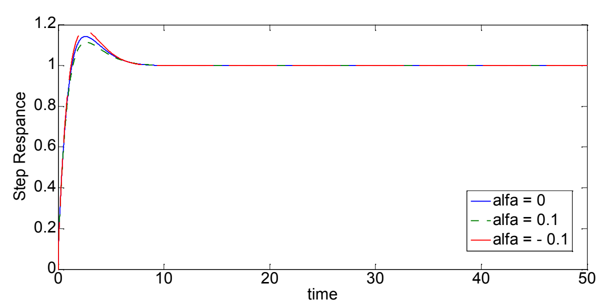

Fig 16. Step responses with PSO applied on motor 4

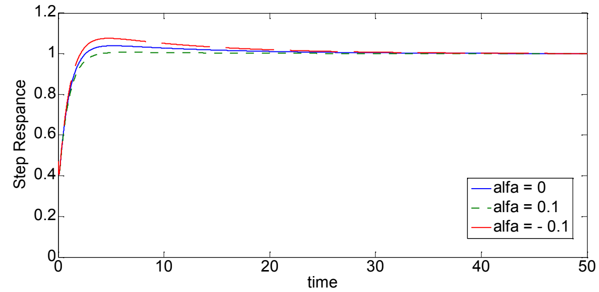

Fig 17. Step responses with PSO applied on motor 5

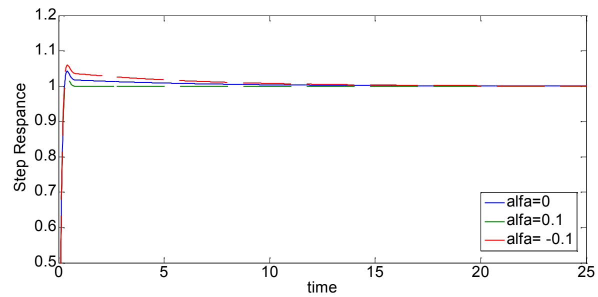

Fig 18. Step responses with FNN applied on motor 1

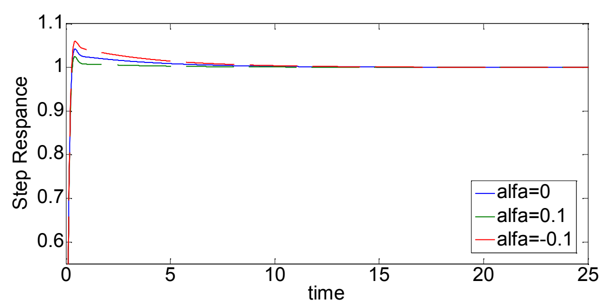

Fig 19. Step responses with FNN applied on motor 2

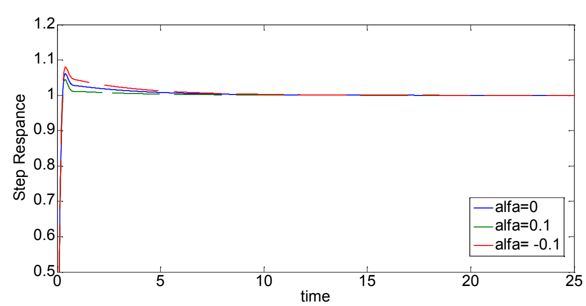

Fig 20. Step responses with FNN applied on motor 3

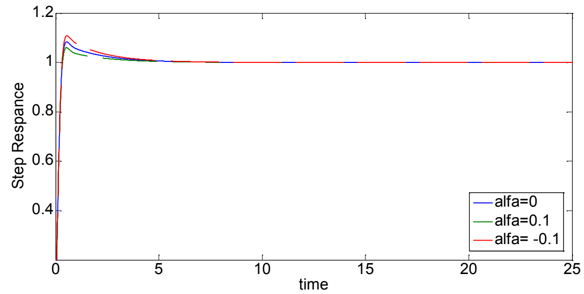

Fig 21. Step responses with FNN applied on motor 4

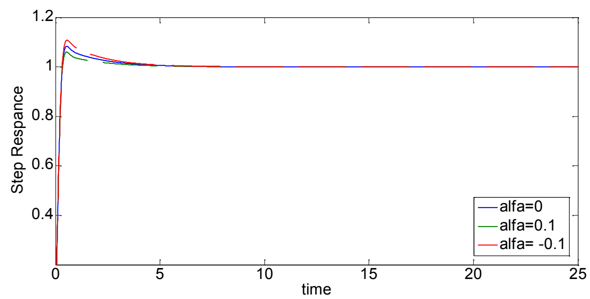

Fig 22. Step responses with FNN applied on motor 5

5. CONCLUSION, DISCUSSION AND FUTURE WORKS

This research begins by introducing the Grypon robot as case-study. Then four methods were applied to design an intelligent PID controller (SFL, ABC, PSO and FNN)which, unfortunately, are not fast enough to be utilized in on-line applications. The results of applied simulations are reported in the included tables and figures.

Comparing the contents of tables and figures leads to introducing the FNN strong enough in elimination of the steady state error and reduction of the settling time and rise time. On the other hand the ABC algorithm acts better on decreasing the overshoot. Also the act of SFL and PSOalgorithm is good in elimination of the steady state error.

From the robustness point of view, while applying the disturbance factor, the SFL algorithm and FNN perform stronger in the transient part, however all of the methods detect the main graph in the steady-state manner.

Designing PID controllers with other intelligent algorithms and changing the present controller to an on-line one may be our future field ofstudy.

REFERENCES

[1] R.L. Haupt, S.E. Haupt, Practical Genetic Algorithms, John Wiley, 2004.

[2] M. Dorigo and T. Stützle, “Ant Colony Optimization”, MIT Press, Cambridge, 2004.

[3] R.C. Eberhart and Y. Shi, “Particle Swarm Optimization: Developments, Applications and Resources”, IEEE Trans, 2001.

[4] J. Kennedy, and R.C. Eberhart, “A Discrete Binary Version ofthe Particle Swarm Algorithm”, IEEE International Conference on Systems, Man and Cybernetics, 'Computational Cybernetics and Simulation', vol.5,pp.4104–4108, 1997 .

[5] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization:artificial bee colony, Journal of Global Optimization, Volume 39, Issue 3, PP. 459 – 471, 2007

[6] D. Karoboga, B. Basturk, On the Performance of Artificial Bee Colony (ABC) Algorithm, Applied Soft Computing, pp:687- 697, Vol. 8, Issue 1, 2008.

[7] I. Hassanzadeh, H. Jabbari, “Open Architecture of Gryphon Robot for Visual Servo and Tele-operation Tasks”, Proceeding of the 4th t International Symposium on Mechatronics and its Applications (ISMA07), Sharjah, U.A.E. March 26-29, 2007

[8] Mark W. Spong, Seth Hutchinson, M. Vidyasagar, "Robot Modeling and Control". ISBN: 978-0-471-64990-8.

[9] Liong S-Y, Atiquzzaman Md. Optimal design of water distribution network using shuffled complex evolution. J Inst Eng, Singapore 2004;44(1):93–107.

[10] Eusuff MM, Lansey KE. Optimization of water distribution network design using the shuffled frog leaping algorithm J Water Resour Plan Manage 2003;129(3):210–25.

[11] R.C. Eberhart, and J. Kennedy, “A New Optimizer Using Particle Swarm Theory”, Proceeding of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp.39-43, 1995

[12] Ben Coppin, "Artificial Intelligence Illuminated", Jones & Bartlett Publishers Inc., 2004

[13] “Neuoro-fuzzy Modeling and Control”, J.S.R. Jang and C.-T. Sun, Proceeding of the IEEE, 83(3):387-406

[14] L. Gaing, A Paticle Swarm Optimization Approch For Optimum Design of PID Controller in AVR System, IEEE International Conference on Energy conversion, pp. 384-391, Vol.19, Issue2, 2004

[15] E. Elbeltagi, T. Hegazy, and D. Grierson, “Comparison among Five Evolutionary-Based Optimization Algorithms,” Advanced Engineering Informatics, vol. 19, no. 1, pp. 43–53, 2005.