International Journal of Soft Computing 9 (6): 408-415, 2014
ISSN: 1816-9503

Step Response Enhancement of Hybrid Stepper Motors Using Soft Computing Techniques

Amged S. El-Wakeel and A.A. Sarhan
Military Technical College, Cairo, Egypt

Abstract: This study presents the use of different soft computing techniques for step response enhancement of hybrid stepper motors. The basic differential equations of hybrid stepper motor are used to build up a model using MATLAB Software package. The implementation of Fuzzy Logic (FL) and Proportional-Integral-Derivative (PID) controllers are used to improve the motor performance. The numerical simulations by a PC-based controller show that the PID controller tuned by Genetic algorithm produces better performance than that tuned by fuzzy controller. They show that the fuzzy PID-like controller produces better performance than the other linear fuzzy controllers. Finally, the comparison between PID controller tuned by Genetic algorithm and the fuzzy PID-like controller shows that the fuzzy PID-like controller produces better performance.

Key words: Hybrid stepper motor, soft computing techniques, fuzzy controller, PID controller, Genetic algorithm

INTRODUCTION

Stepper motors are a type of electromagnetic mechanical devices which can convert discrete electric impulses into angular displacement or linear displacement. Their output angular displacement is directly proportional to the amount of the input pulses and their speed of rotation is directly proportional to the frequency of the input pulses (Wen *et al*., 2006). These motors possess all the advantages of standard Permanent Magnet (PM) synchronous machines while their cost is much lower (Chen *et al*., 2006). Stepping motors are the ideal choice for those applications where power is small (<100 W) while position control must be sharp and fast such as in robotics, aerospace applications, machine tools, printers, scanners and servos (Bellini *et al*., 2004).

Stepper motors have several merits such as great output torque, small inertia and high frequency response. These features contribute to the wide usage in the industry nowadays, especially in measurement and control applications (Wen *et al*., 2006). Stepper motors have also some demerits such as step response with overshoot and relatively long settling time. Besides, loss of synchronism appears when steps of high frequency are given. It is thus necessary to develop control schemes to improve the performance of stepper motors (Alvarez-Gallegos *et al*., 2004).

Recently, the positioning systems are being implemented using stepper motors instead of traditionally DC motors (Zribi and Chiasson, 1991). This is because, for high speed repetitive motion, the brushes in DC motors are subject to excessive mechanical wear and consequently lead to a decrease in performance. Also, since there is a winding on the rotor of DC motors, the rotor copper loss and hence the heat does not have a direct path to the outside environment but instead must be dissipated through the stator. Another important aerospace application of the stepper motors is that in order to achieve uniform performance throughout the life span of >10 years and to avoid unpredictable and intolerable disturbances to the satellite, the stepper motors are required to deliver a constant torque (Rajagopal *et al*., 1997a, b).

MATHEMATICAL MODEL FOR HYBRID STEPPER MOTOR

Basically, the model of the hybrid stepper motors can be expressed in forms of equations as follows (Sarhan *et al*., 2009; Ghafari and Alasty, 2004; Persson and Perriard, 2003; Alvarez-Gallegos *et al*., 2004):

$$L\frac{di_a}{dt} = -Ri_a + K_c\omega\sin(N_r\theta) + v_a \quad (1)$$

$$L\frac{di_b}{dt} = -Ri_b + K_c\omega\cos(N_r\theta) + v_b \quad (2)$$

$$J\frac{d\omega}{dt} = -i_a K_c\sin(N_r\theta) + i_b K_c\cos(N_r\theta) - \tau_L \quad (3)$$

Corresponding Author: Amged S. El-Wakeel, Military Technical College, Cairo, Egypt

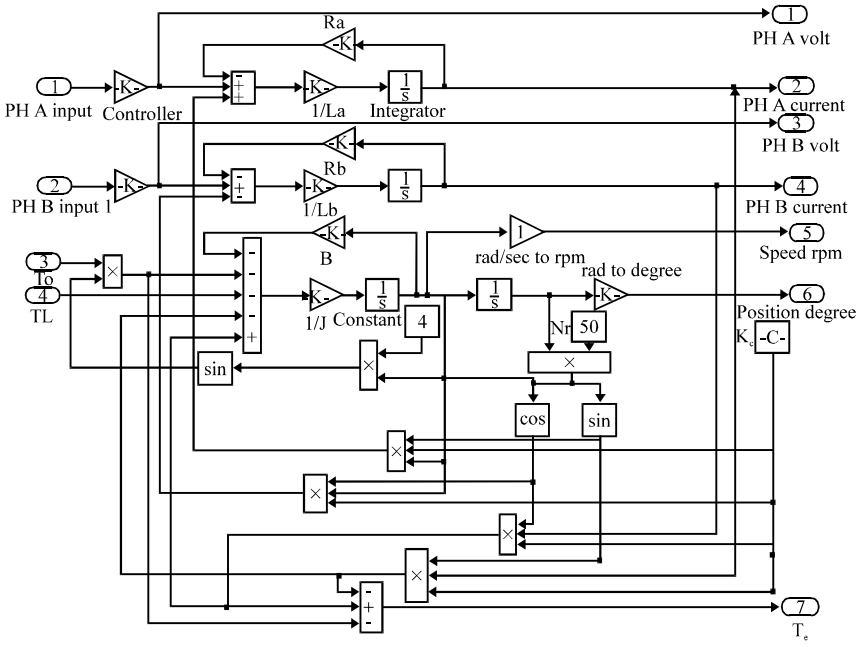

Fig. 1: Hybrid stepper motor model using MATLAB toolbox

$$\omega = \frac{d\theta}{dt} \quad (4)$$

Where:

v_a and i_a = The voltage and current in phase a, respectively
v_b and i_b = The voltage and current in phase b, respectively
ω = The rotor speed
θ = Rotor position angle
R = The resistance of the phase winding
L = The inductance of the phase winding
B = The friction coefficient
J = The rotor inertia
K_c = The motor torque constant
τ_L = The load torque
N_r = The number of rotor teeth

In this study, MATLAB Software is used to solve simulate, study and investigate the motor performance subject to different conditions. The motor model (Sarhan *et al*., 2009) is shown in Fig. 1.

FUZZY LOGIC CONTROLLER (FLC)

The FLC provides an algorithm which converts the linguistics control based on expert knowledge into an automatic control strategy. Therefore, the FLC algorithm is much closer in spirit to human thinking than traditional logical systems (Betin *et al*., 2000). FLC provides a formal methodology for representing, manipulating and implementing a human's heuristic knowledge about how to control a system (Passino and Yurkovich, 1998). The block diagram in Fig. 2 shows a fuzzy control embedded in a closed-loop control system. The plant output is denoted by y(t), its input is denoted by u(t) and the reference input to the fuzzy controller is denoted by r(t). As shown in Fig. 2, the fuzzy control has four main components: the rule-base, the inference mechanism, the fuzzification interface and the de-fuzzification interface.

The "rule-base" holds the knowledge, in the form of a set of rules (if-then rules), of how best to control the system. There are at least four main sources for finding control rules (Lewis, 2007): expert experience and control engineering knowledge, operator's control actions, fuzzy model of the plant or based on learning. The inference mechanism evaluates which control rules are relevant at the current time and then decides what the input to the plant should be. The fuzzification interface simply modifies the inputs so that they can be interpreted and compared to the rules in the rule-base. The de-fuzzification interface converts the conclusions reached by the inference mechanism into inputs to the plant. In brief, FLC is designed using certain steps mentioned as (El-Halim Mohammad, 2006):

- Define the characteristics of the plant from the operator
- Apply conventional control simulation to study the system performance
- Define the control variables such as the position, torque and the speed

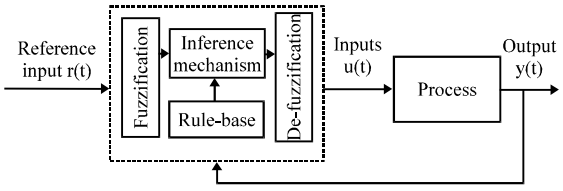

Fig. 2: Fuzzy controller architecture

- Identify the input and output of FL controller
- Define the universe of discourse of each input and output
- Define universe of discourse, fuzzy sets and their inputs and outputs membership functions for inputs and outputs
- Determine the rules-table (look-up table) and the corresponding scaling factors for inputs and outputs
- Simulate the system using FL controller and observe performance
- Iterate the rule table, membership functions (mfs) until the performance is satisfactory
- Implement the FL controller for real time

There are several types of linear FLC (Lewis, 2007): Fuzzy Proportional (FP), Fuzzy Proportional-Derivative (FPD), Fuzzy Proportional-Derivative-plus-Integral (FPD+I), Fuzzy Incremental (FI) and Fuzzy PID-like controller.

The FP controller is derived from the PID controller with the I-action set to zero ($(1/T_i) = 0$) and the D-action set to zero ($T_d = 0$) where, T_i and T_d are the integral and derivative time constants, respectively. As shown in Fig. 3, FP controller acts on the error (e) and its control signal is (U). The FP controller has two tuning gains: Error Gain (GE) and Output Gain (GU) where the crisp proportional controller has just one, K_P. The control signal U(n), at the time instant (n) is generally a nonlinear function of the input e(n):

$$U(n)=f(GE\cdot e(n))\cdot GU \tag{5}$$

The FPD type control generates control output from error and change of error as shown in Fig. 4. Derivative action helps to predict the future error and the PD controller uses the derivative action to improve closed-loop stability.

The control signal U(n) at the time instant (n) is a nonlinear function of error and change in error is:

$$U(n)=f(GE.e(n), GCE.\dot{e}(n)).GU \tag{6}$$

where, GCE is the change of error gain.

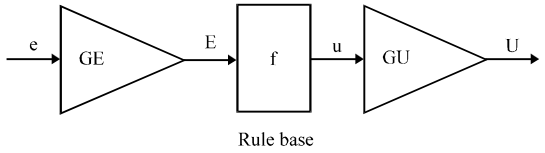

Fig. 3: FP controller

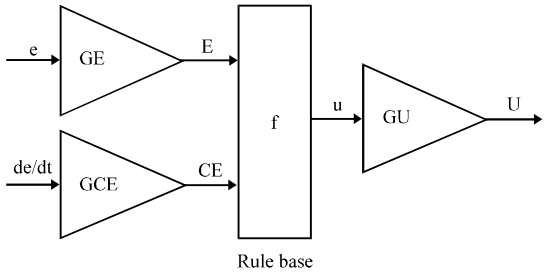

Fig. 4: FPD controller

If the Closed-Loop System exhibits a sustained error in steady state, integral action is necessary. The integral action will increase (or decrease) the control signal if there is a positive (or negative) error, even for small magnitudes of the error. Thus, a controller with integral action will always return to the reference in steady state. A fuzzy PID controller acts on three inputs: error, integral error and change in error. A rule base with three premise inputs can be a problem. To overcome this problem, a simple design is to combine crisp integral action and a fuzzy PD rule base in the fuzzy PD+I (FPD+I) controller as in Fig. 5.

The control signal U(n) at the time instant (n) is a nonlinear function of error, change in error and integral error as:

$$U(n)=\left[f(GE.e(n), GCE.\dot{e}(n))+GIE\sum_{j=1}^{n}e(j)T_s\right].GU \tag{7}$$

Where:

T_S = The sampling time
GIE = The integral error gain

The fuzzy incremental controller in Fig. 6 is of almost the same configuration as the FPD controller except for the added integrator. The conclusion in the rule base is now called change in output (cu) and the gain on the output is, accordingly, GCU.

The control signal U(n) at time instant (n) is the sum of all previous increments and can be expressed as:

$$\begin{aligned}U(n)&=\sum_{j=1}^{n}(cu(j)\cdot GCU\cdot T_s)\\&=\sum_{j=1}^{n}(f(GE\cdot e(j),GCE\cdot\dot{e}(j))\cdot GCU\cdot T_s)\end{aligned} \tag{8}$$

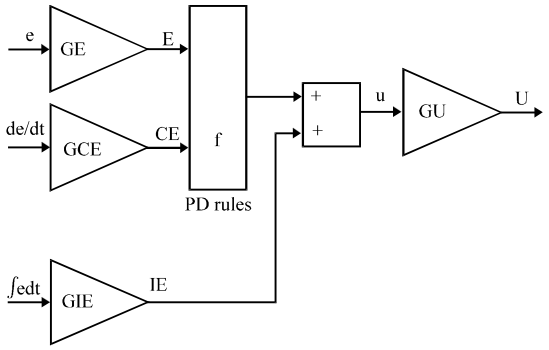

Fig. 5: FPD+I controller

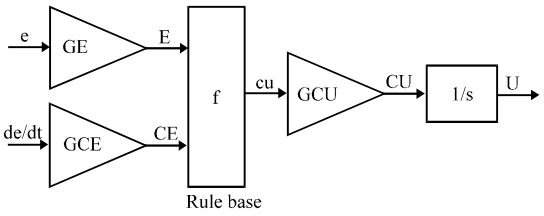

Fig. 6: Fuzzy incremental controller

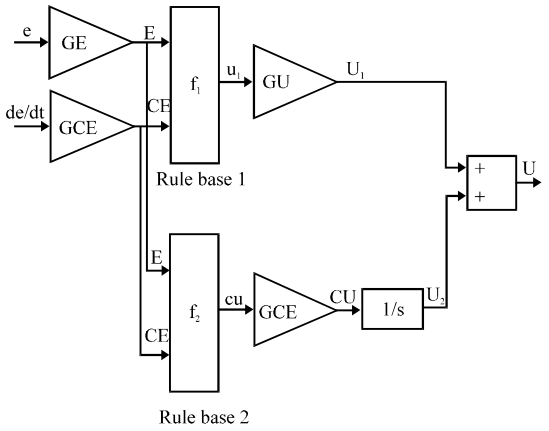

Fig. 7: Fuzzy PID-like controller fuzzy

A fuzzy PID-like controller is a fuzzified PID controller. It acts on the same input signals but the control strategy is formulated as fuzzy rules. A Fuzzy PID-like controller structure is simply formed by connecting the FPD type and FIC together in parallel as shown in Fig. 7.

The control signal U(n) at the time instant (n) is the sum of U_1 and U_2.

$$\begin{aligned} U(n) &= U_1 + U_2 \\ &= f_1(GE \cdot e(n), GCE \cdot \dot{e}(n)) \cdot GU + \\ &\quad \sum (f_2(GE \cdot e(j), GCE.\dot{e}(j)).GCU.T_s) \end{aligned} \tag{9}$$

GENETIC ALGORITHM (GA)

GA is simple, powerful, general-purpose, derivative-free, Stochastic Global Optimization Methods inspired by the laws of natural selection and genetics. These algorithms are derivative-free which means that they do not need functional derivative information to search for a set of parameters that minimize (or maximize) a given objective function. Instead, they exclusively rely on repeated evaluations of the objective function and the subsequent search direction after each evaluation follows certain heuristic guidelines. In particular, the optimum solution is obtained by investigating new solutions which incorporate three genetic operations: reproduction, crossover and mutation in a selective environment where the fittest survive (Werner Leonhard, 2001). There are three main steps for Genetic algorithm, random initialization of population, evaluation of fitness function and finally generation of new population.

In random initialization of population, the initial population is created randomly with even Number (N) of individuals. An individual is characterized by a fixed-length binary bit string which is called a chromosome. In evaluation of fitness function all the individuals of the initially created population are evaluated by means of a fitness function (f). The fitness function is then used in the next step, to create a genetic pool. After evaluating the fitness of the individuals of the initial population, a new population is created. The creation of a new generation is performed basically in three stages, reproduction, crossover and mutation. The overall goal of this step is to obtain a new population with individuals which have high fitness values.

In reproduction stage, the individuals are selected among the population with probabilities proportional to their fitness values. In this way, individuals with lower fitness values are eliminated whilst others are copied to the next generation one or more times and some are not copied at all. The population after reproduction stage is called mating pool.

In crossover stage, a genetic crossover operator is applied to the mating pool to generate new individuals. Thus individuals of the mating pool are paired randomly and N/2 genetic couples are obtained. There are many crossover operators can be used but the most basic crossover operator is the one-point crossover operator in this case a crossover point in the string bits of the selected pair is randomly chosen and the bits of the two parents are interchanged at this point as shown in Fig. 8. In two-point crossover operation, the two crossover points are selected in the binary strings of the pair under consideration and between these points the bits are swapped as shown in Fig. 9. This crossover process is similar to the mating process in a biological system where

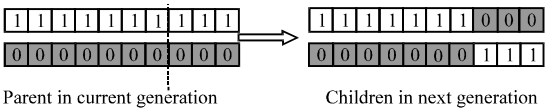

Fig. 8: One-point crossover

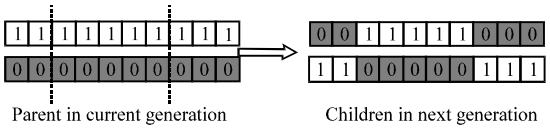

Fig. 9: Two-point crossover

parents pass segments of chromosomes to their offspring and thus offspring can outperform their parents if they get 'good' genes from both parents.

The crossover process is followed by a mutation process which introduces further changes to a bit string. This is required, since if the population does not contain all the encoded information required to solve a specific problem, no amount of gene mixing can provide a satisfactory solution. By applying the mutation operator, it is possible to produce new chromosomes. This can be implemented in various ways and the most common technique is to change a randomly chosen bit in the bit string of the individual to be mutated. Thus, certain bit is changed from 1 into 0 or from 0 into 1. Finally, it can be said that the crossover and mutation together give Genetic algorithms most of their searching power (El-Wakeel *et al*., 2013; Mohmoud, 2009).

SIMULATION RESULTS

The hybrid stepper motor with data given in Table 1 is used in the simulation (Sarhan, 2011). While, control rules for FLC are shown in Table 2. Table 3-5 show the control rules for self tuning of the PID controller.

First of all, the model is verified by comparing its response with the built-in hybrid stepper motor model obtained in MATLAB simulink toolbox. The comparison results are shown in Fig. 10. It is clear from the comparison that the error in the rotor position, motor speed and electromagnetic torque is too small and can be neglected.

The hybrid stepper motor is investigated on open loop control. This investigation includes the study of the motor position, rotor speed and electromagnetic generated torque on the shaft as shown in Fig. 11.

The motor performance using PID controller with different tuning methods is compared. The tuning of the PID controller by GA is carried out using a MATLAB

Table 1: Hybrid stepper motor parameters

Parameters	Values
Motor phase A resistance (Ω)	$R_a = 10$
Motor phase A inductance (H)	$L_a = 0.001$
Motor phase B resistance (Ω)	$R_b = 10$
Motor phase B inductance (H)	$L_b = 0.001$
Machine torque constant (Nm/A)	$K_C = 0.113$
Applied DC phase voltage (V)	$V_s = 5$
Friction coefficient (Nm S/rad)	B = 0.00
Inertia constant (kg m^2)	J = 0.00005
Number of rotor teeth	$N_r = 50$

Table 2: Control rules of FLC

	ė						
e	NB	NM	NS	ZR	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZR
NM	NB	NB	NB	NM	NS	ZR	PS
NS	NB	NB	NM	NS	ZR	PS	PM
ZR	NB	NM	NS	ZR	PS	PM	PB
PS	NM	NS	ZR	PS	PM	PB	PB
PM	NS	ZR	PS	PM	PB	PB	PB
PB	ZR	PS	PM	PB	PB	PB	VB

Table 3: Rule base for determining the gain K_P

	ė				
e	NB	NS	ZE	PS	PB
NB	VB	VB	VB	VB	VB
NS	B	B	B	MB	VB
ZE	ZE	ZE	MS	S	S
PS	B	B	B	MB	VB
PB	VB	VB	VB	VB	VB

Table 4: Rule base for determining the gain K_i

	ė				
e	NB	NS	ZE	PS	PB
NB	M	M	M	M	M
NS	S	S	S	S	S
ZE	MS	MS	ZE	MS	MS
PS	S	S	S	S	S
PB	M	M	M	M	M

Table 5: Rule base for determining the gain K_d

	ė				
e	NB	NS	ZE	PS	PB
NB	ZE	S	M	MB	VB
NS	S	B	MB	VB	VB
ZE	M	MB	MB	VB	VB
PS	B	VB	VB	VB	VB
PB	VB	VB	VB	VB	VB

built-in routine so called Simulink Response Optimization (SRO) toolbox. The SRO, automatically, formulates an optimization problem and calls a GA and direct search toolbox as an optimization routine to solve the problem (Sarhan, 2011). It is clear from Fig. 12 that PID controller with GA provides better performance. The three types of PID controller have no overshoot and no steady state error. PID controller with GA has the least rise time and settling time.

The motor performance using different fuzzy controllers is compared. It is clear from Fig. 13 that the

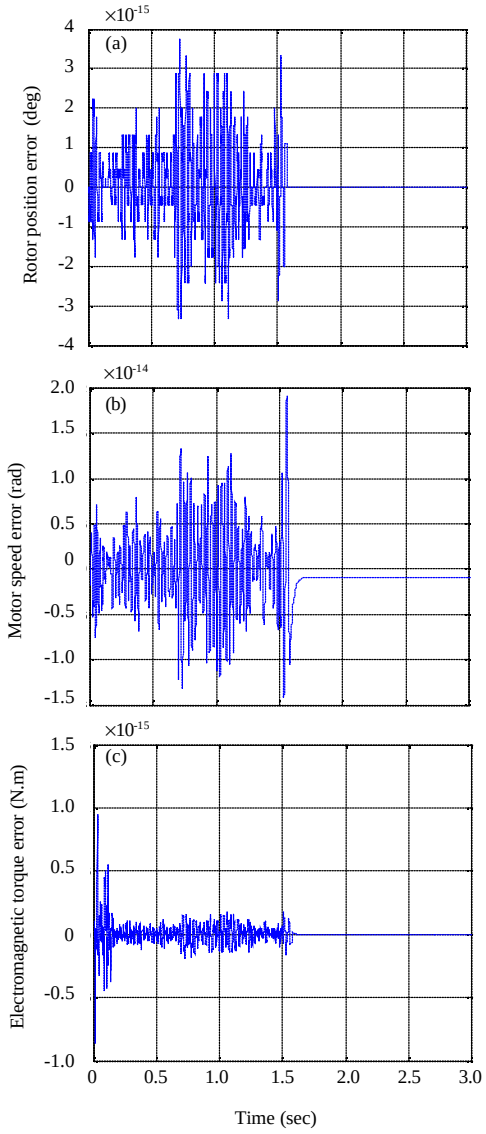

Fig. 10: Results for new model verification; a) rotor position error; b) motor speed error and c) electromagnetic torque error

best performance is obtained from fuzzy PID-like controller. The different types eliminate the oscillation around final position, overshoot and the steady state error. The least rise time and settling time obtained from fuzzy PID-like controller.

Finally, the motor performance using PID with GA and fuzzy PID-like controller is compared. It is clear from Fig. 14 that the fuzzy PID-like controller produces better performance.

The model with fuzzy PID-like controller is tested to verify the controller capability to follow a reference position and the results are shown in Fig. 15.

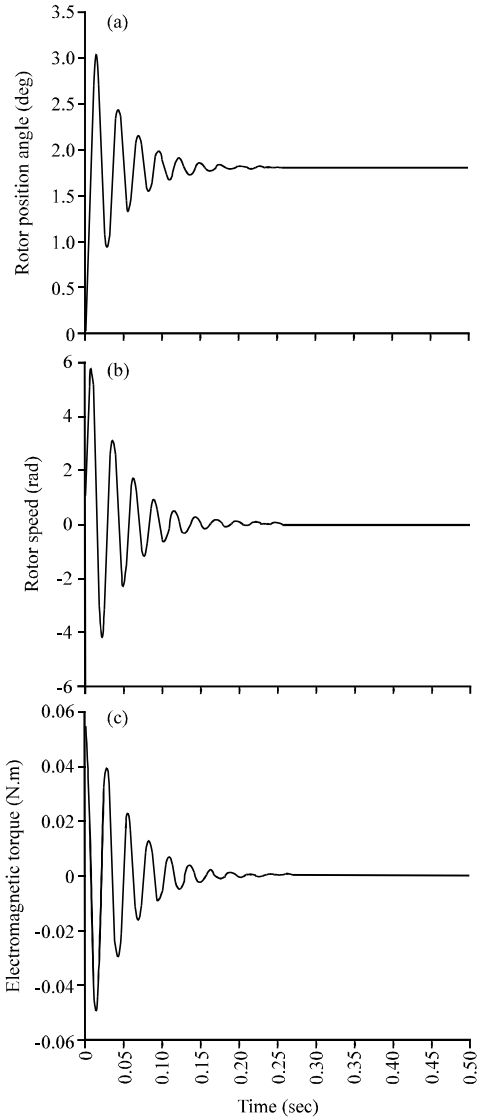

Fig. 11: Full-step motor performance; a) Rotor position angle; b) Rotor speed; c) Electromagnetic torque

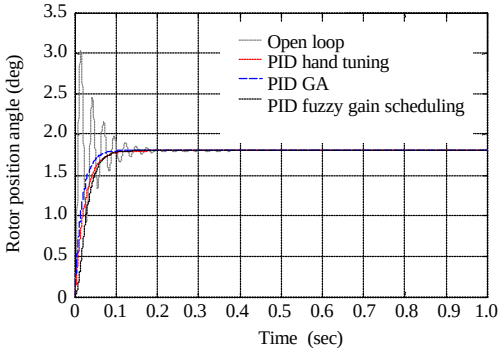

Fig. 12: Comparison between PID controllers with different tuning methods

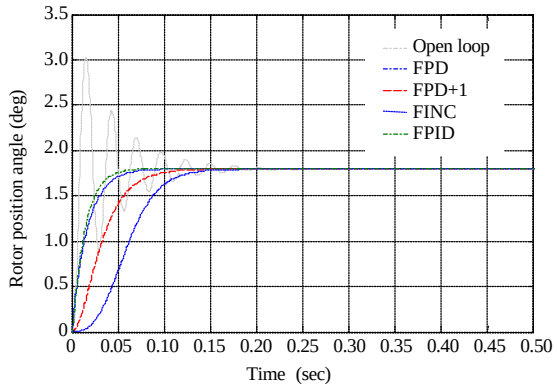

Fig. 13: Comparison between different fuzzy controllers

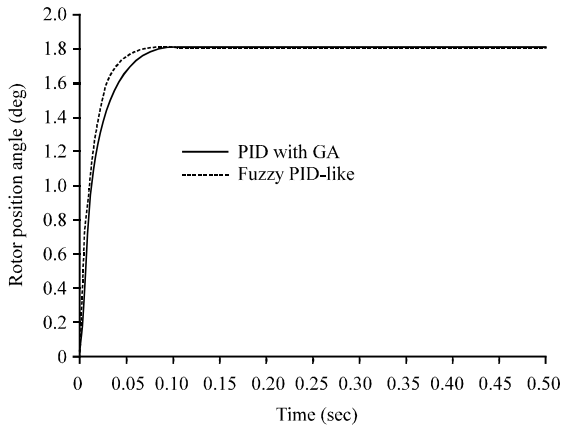

Fig. 14: Comparison between PID with GA and fuzzy PID-like controller

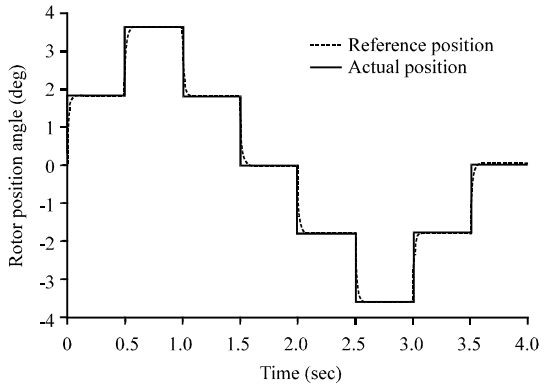

Fig. 15: Motor performance with fuzzy PID-like controller following position trajectory

CONCLUSION

Different soft computing techniques including fuzzy logic and Genetic algorithm have been applied to improve the step response of hybrid stepper motors. The Linear Fuzzy Controller algorithms for closed-loop control of hybrid stepper motors have been discussed. The numerical simulations by a PC-based controller show that the fuzzy PID-like controller produces better performance than the other linear fuzzy controllers.

In addition, both Genetic algorithm and fuzzy logic are used to tune the PID controller gains. The numerical simulations show that the PID controller tuned by Genetic algorithm produces better performance than that tuned by fuzzy logic.

The comparison between PID controller tuned by Genetic algorithm and the fuzzy PID-like controller shows that the fuzzy PID-like controller produces better performance.

REFERENCES

Alvarez-Gallegos, J., E. Alvarez-Sanchez and R. Castro Linares, 2004. Experimental setup for sensorless rotor position control of a permanent magnet stepper motor. Proceedings of the 9th IEEE International Power Electronics Congress, October 17-22, 2004, Mexico, pp: 123-127.

Bellini, A., C. Concari, G. Franceschini and A. Toscani, 2004. Mixed mode PWM for high performance stepping motors. Proceedings of the 30th Annual Conference of IEEE Industrial Electronics Society, Volume 2, November 2-6, 2004, Italy, pp: 1212-1217.

Betin, F., D. Pinchon and G.A. Capolino, 2000. Fuzzy logic applied to speed control of a stepping motor drive. IEEE Trans. Industrial Electr., 47: 610-622.

Chen, W.D., K.L. Yung and K.W. Cheng, 2006. A learning scheme for low-speed precision tracking control of hybrid stepping motors. IEEE/ASME Trans. Mechatronics, 11: 362-365.

El-Halim Mohammad, H.A., 2006. Analysis and position control of stepper motor using artificial intellegence. Al-Azhar University, Cairo.

El-Wakeel, A.S., A.E. Elawa and Y. El-Koteshy, 2013. Position control of a single arm manipulator using ga-pid controller. Int. J. Electrical Eng. Technol., 4: 120-135.

Ghafari, A.S. and A. Alasty, 2004. Design and real-time experimental implementation of gain scheduling PID fuzzy controller for hybrid stepper motor in micro-step operation. Proceedings of the IEEE International Conference on Mechatronics, June 3-5, 2004, Tehran, Iran, pp: 421-426.

Lewis, H.W., 2007. The Foundations Of Fuzzy Control. Springer, USA., pp: 324.

Mohmoud, Y.S., 2009. Modern control strategies of electric machines. Military Technical College, Cairo.

Passino, K.M. and S. Yurkovich, 1998. Fuzzy Control. Addison-Wesley, California, ISBN: 9780201180749, Pages: 475.

Persson, J. and Y. Perriard, 2003. Steady state Kalman filtering for sensorless control of hybrid stepper motors. Proceedings of the IEEE International Electric Machines and Drives Conference, Volume 2, June 1-4, 2003, Switzerland, pp: 1174-1177.

Rajagopal, K.R., N. Kannan, B. Singh and B.P. Singh, 1997a. An optimized module-type hybrid stepper motor for spacecraft solar array drive. Proceedings of the International Conference on Power Electronics and Drive Systems, Volume 1, May 26-29, 1997, India, pp: 462-468.

Rajagopal, K.R., M. Krishnaswamy, B. Singh and B.P. Singh, 1997b. An improved high resolution hybrid stepper motor for solar array drive of Indian Remote Sensing satellite. Proceedings of the International Conference on Power Electronics and Drive Systems, February 21-24, 1995, Trivandrum, India, pp: 610-615.

Sarhan, A., A. El-Wakeel and A.B. Kotb, 2009. Analysis and control of hybrid stepper motor for automatic suntracking system. Proceedings of the 13th International conference on Aerospace Sciences and Aviation Technology, May 26-29, 2009, Cairo, Egypt, pp: 1-11.

Sarhan, A.A., 2011. Design analysis and simulation of satellite solar cell tracking system. M.Sc. Thesis, Military Technical College, Cairo.

Wen, Z., W. Chen, Z. Xu and J. Wang, 2006. Analysis of two-phase stepper motor driver based on FPGA. Proceedings of the IEEE International Conference on Industrial Informatics, August 16-18, 2006, Singapore, pp: 821-826.

Zribi, M. and J. Chiasson, 1991. Position control of a PM stepper motor by exact linearization. IEEE Trans. Automatic Control, 36: 620-625.