Paper: ASAT-15-219-CT

15th International Conference on
***AEROSPACE SCIENCES & AVIATION TECHNOLOGY***,
***ASAT - 15*** – May 28 - 30, 2013, Email: asat@mtc.edu.eg ,
Military Technical College, Kobry Elkobbah, Cairo, Egypt,
Tel: +(202) 24025292 –24036138, Fax: +(202) 22621908

# Speed Regulation of Brushless DC Drives Using Optimized Fuzzy Logic Controller

F. N. Hassan*, A. Kamel†, W. M. Fayek‡, H. M. Seoudy‡

**Abstract:** Brushless Direct Current (BLDC) motor becomes at the top of motors in high performance drive systems such as Electric vehicle (EV). This paper presents a modern approach of speed control for BLDC using particle swarm optimization (PSO) algorithm. to optimize the scaling factors of fuzzy logic controller (FLC). The overall system is simulated under various operating conditions and simulation results have been examined at different control techniques. At the same operating conditions, these simulation results are compared with those obtained using fuzzy logic. The investigated results illustrated that the use of PSO as an optimization algorithm makes the drive robust, with faster dynamic response, higher accuracy and insensitive to load variation. The system is tested for a step change in load and the simulation results showed good dynamic response with fast recovery time.

## 1. Introduction

The brushless DC (BLDC) motor is a permanent magnet synchronous machine supplied from a six-transistor inverter whose on/off switching is determined by the rotor position. Because of no brushes or commutator, the system is becoming increasingly attractive in high performance variable-speed drives. It can produce torque-speed characteristic similar to that of a permanent-magnet conventional DC motor while avoiding the problems of failure of brushes and mechanical commutation. In addition to the reduction in maintenance needs, the BLDC motor has low inertia, large power to volume ratio, significant reduction in friction and very small noise as compared with the permanent-magnet conventional DC servomotor at the same rating. However, all the above advantages are purchased at the price of high cost and more complex controller than that of the conventional motor.

Many control techniques have been developed to improve the performance of motor drives. These techniques include PID control, nonlinear feedback control, fuzzy control, adaptive control, sliding mode control, etc. The most widely used control method is Proportional Integral Derivative (PID) controller.

BLDC motor speed and position control have been a topic of interest for the last few years. FLC for speed control plus integral proportional (IP) for a robust position control is discussed in [1]. The speed FLC has two rule tables one for common control during transient and another for fine control at steady state [1]. Yu et al. [2] have presented a Linear Quadratic Response (LQR) method to optimally tune the PID gains. In this method, the response of the system is near optimal but it requires mathematical calculations and solving equations. A

* Dept. of El. Power &Machine - Faculty of Engineering - Helwan University, Cairo, Egypt.

† Dep. of El. Power &Machine - High Institute of Engineering - Shorouk Academy, Cairo, Egypt; aellissy54@yahoo.ca .

‡ Dep. of El. Power &Machine - High Institute of Engineering - Shorouk Academy, Cairo, Egypt.

15th International Conference on
***AEROSPACE SCIENCES & AVIATION TECHNOLOGY***,
***ASAT - 15*** – May 28 - 30, 2013, Email: asat@mtc.edu.eg ,
Military Technical College, Kobry Elkobbah, Cairo, Egypt,
Tel: +(202) 24025292 –24036138, Fax: +(202) 22621908

fuzzy based sliding mode control scheme for BLDC motor was proposed in [3]. It combines the best features of both the FLC and sliding mode control to achieve rapid and accurate

control. Rubaai A, et al [4] gave the design and experimental verification of hybrid fuzzy and PI controllers for BLDC motor drive. The idea is to use PI controller while keeping in the background a fuzzy controller, which is ready to take over PI controller when severe perturbations occur. Thus, the PI and fuzzy controllers can be managed to take advantage of their positive attributes. For adaptive control, complex adaptive laws are often used to compensate the effect of parameters variations while the transient response may not be satisfactory in practical applications [5].

In [6] speed regulation scheme of a small brushless DC motor (BLDC motor) with trapezoidal back-emf consideration has been presented. Its proposed control strategy uses the proportional controller in which the proportional gain, kp, is appropriately adjusted by using genetic algorithms. Sliding mode controllers SMCs have been widely used for speed and position control of Permanent Magnet Synchronous Motors PMSMs. They provide a fast dynamic response, insensitivity to parameter variations and external load disturbance [7; 8]. Fuzzy logic controllers have been used in many areas after fuzzy set theory was first introduced by Zadeh. A FLC can be easily used in the control of systems for which an exact mathematical model of the system cannot be obtained. The essential part of a FLC is a set of the linguistic control rules related by the dual concepts of fuzzy implication and the compositional rule of inference. Classically, fuzzy variables have been adjusted by expert knowledge and trial and error. A neuro-fuzzy control (NFC) for speed control of a PMSM drive system is proposed in [9]. A neural network (NN) is used to adjust input and output parameters of membership functions in FLC. The back propagation-learning algorithm is used for training this network. Simulation and experimental results indicate that the proposed controller is reliable and effective in the speed control of the PMSM. Cetin Elmas, and Oguz Ustun in [10] introduces a hybrid controller (HC) for speed control of a PMSM drive. In the drive, a SMC and a NFC are connected in parallel which used to obtain a controller that eliminates the chattering phenomenon and provides a fast and smooth dynamic response for the speed control of PMSM drive. By using an error band method, the system is controlled by the SMC to get a fast dynamic response in transient mode, and is also controlled by the NFC to get a smooth dynamic response in steady state mode. In [11] a novel PSO-based approach to optimally design a PID controller for a brushless DC motor is proposed. Alberto A. in [12] demonstrates the ability to optimize PID parameters for a BLDC motor by using a particle swarm optimizer.

In this paper, a PSO-based approach to optimally design a Fuzzy logic controller for a brushless DC motor subjected to wide loading variations is proposed. Simulation results with real motor parameters were processed to yield estimate the dynamic response and measurements.

## 2. Mathematical Model of the BLDC Motor Drive

It is assumed that the BLDC motor is connected to the output of the inverter, while the inverter input terminals are connected to a constant supply voltage. The equivalent circuit model that refers to this circuit diagram is shown in Figure 1. Another assumption is that there are no power losses in the inverter and the 3-phase motor winding is connected in star.

The equations that govern this model are as follows.

$$v_A = v_N + v_{sA}$$
$$v_B = v_N + v_{sB} \quad (1)$$
$$v_C = v_N + v_{sC}$$

where:

$v_{sA}$, $v_{sB}$, $v_{sC}$ are the inverter output voltages that supply the 3 – phase winding.

$v_A$, $v_B$, $v_C$ are the voltages across the motor armature winding.

$v_N$ – voltage at the neutral point.

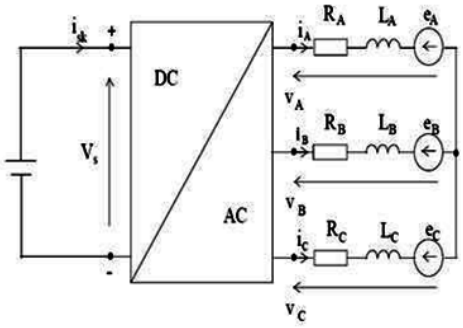


**Figure 1: Equivalent circuit of 3-phase PM BLDC motor.**

For a symmetrical winding and balanced system, the voltage equation across the motor winding is as follows:

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = \begin{bmatrix} R_A & 0 & 0 \\ 0 & R_B & 0 \\ 0 & 0 & R_C \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} L_A & L_{AB} & L_{AC} \\ L_{BA} & L_B & L_{BC} \\ L_{CA} & L_{CB} & L_C \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \quad (2)$$

Since $R_A = R_B = R_C = R$ , and for the inductances, since the self and mutual inductances are constant for surface mounted permanent magnets on the cylindrical rotor, and the winding is symmetrical:

$$L_A = L_B = L_C = L; \text{ and } L_{AB} = L_{BC} = L_{CA} = L_{BA} = L_{AC} = L_{CB} = M \quad (3)$$

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} L & M & M \\ M & L & M \\ M & M & L \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \quad (4)$$

For a Y-connected stator winding,

$$i_A + i_B + i_c = 0 \quad (5)$$

Therefore, the voltage takes the following form:

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} L_s & 0 & 0 \\ 0 & L_s & 0 \\ 0 & 0 & L_s \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \quad (6)$$

where the synchronous inductance, Ls =L−M
To link the input voltages and currents of the inverter with those of the output, the power equality equation, Pin = Pout is assumed at both sides.
From this, the inverter input current is:

$$i_{sk} = \frac{1}{v_s}(i_A v_{sA} + i_B v_{sB} + i_C v_{sC}) \quad (7)$$

where $v_{sA}$, $v_{sB}$ and $v_{sC}$ are the phase voltages that supply the motor.
The produced torque is defined by the following equations:

$$T_{em} = J_{eq} \frac{d\omega_m}{dt} + B \cdot \omega_m + T_L \quad (8)$$

where $J_{eq} = J_M + J_L$ is the equivalent moment of inertia, and $J_M$, $J_L$ are the moments of inertia of the motor and load respectively, B - friction coefficient and $T_L$ - load torque.
The electromagnetic torque for this 3-phase motor is dependent on the current (i), speed ($\omega_m$) and electromotive force (e). The equation is:

$$T_{em} = \frac{e_A i_A}{\omega_m} + \frac{e_B i_B}{\omega_m} + \frac{e_C i_C}{\omega_m} = K_E(f_a(\phi_e)\cdot i_A + f_b(\phi_e)\cdot i_B + f_C(\phi_e)\cdot i_C) \quad (9)$$

where,

$$\begin{aligned} f_a(\phi_e) &= \sin(\theta_e) \\ f_b(\phi_e) &= \sin\left(\theta_e - \frac{2\cdot\pi}{3}\right) \\ f_c(\phi_e) &= \sin\left(\theta_e - \frac{4\cdot\pi}{3}\right) \end{aligned} \quad (10)$$

Combining all the above equations, the system in state-space form is;

$$\dot{x} = Ax + Bu \quad (11)$$

$$x = [i_A \;\; i_B \;\; i_c \;\; \omega_m \;\; \theta_e]^t \quad (12)$$

$$A = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & 0 & -\frac{K_E(f_a(\phi_e))}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{K_E(f_b(\phi_e))}{L_s} & 0 \\ 0 & 0 & -\frac{R_s}{L_s} & -\frac{K_E(f_c(\phi_e))}{L_s} & 0 \\ \frac{K_E(f_a(\phi_e))}{J} & \frac{K_E(f_b(\phi_e))}{J} & \frac{K_E(f_c(\phi_e))}{J} & -\frac{D}{J} & 0 \\ 0 & 0 & 0 & \frac{P}{2} & 0 \end{bmatrix} \quad (13)$$

$$B = \begin{bmatrix} \frac{1}{L_s} & 0 & 0 & 0 \\ 0 & \frac{1}{L_s} & 0 & 0 \\ 0 & 0 & \frac{1}{L_s} & 0 \\ 0 & 0 & 0 & -\frac{1}{J} \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (14)$$

$$u = [v_A \;\; v_B \;\; v_c \;\; T_L]^t \quad (15)$$

# 3. Controller Design

## 3.1. Fuzzy Logic Controller (FLC)

Increasing interest has been, seen in applying fuzzy set theory to BLDC drive systems. The power of fuzzy logic approach stems from the ability to implement linguistic description of control rule for difficult to model system.
In general, FLC design consists of the following steps:

1. Identification of input and output variables.
2. Construction of control rules.
3. Establishing the approach for describing system state in terms of fuzzy sets (establishing fuzzification method and fuzzy membership functions).
4. Selection of the compositional rule of inference.
5. Defuzzification method, i.e, transformations of fuzzy control statement into specific control actions.

The block diagram of this controller is shown in Figure 2 the controller has two inputs, error (e) and the change of error (ce), and a single output (u). All the three variables must lie in the interval [-1,1]. This is achieved by scaling the physical system variables by appropriate constant and if necessary, limiting the resulting value. The controller inputs of error (e) and change of error (ce) are scaled by the constant e(K) and ce(K) respectively, and the control output (u) by K.

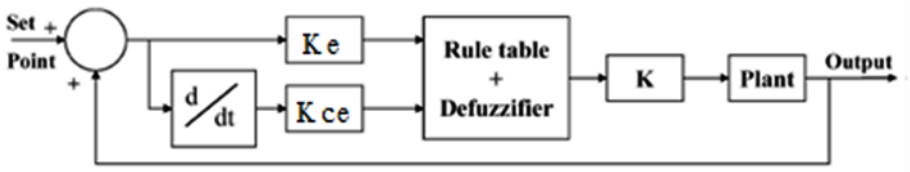


**Figure 2: Block diagram of fuzzy logic controller.**

The Fuzzy rules are in the form:

$R_i$ : IF E is $A_i$ and CE is $B_i$ THEN u is $C_i$

where:
E = Ke * e
CE = Kce * ce
E : the normalized value of the error.
CE : the normalized value of the change of error.
$A_i$, $B_i$ and $C_i$ : Fuzzy subsets in their universes of discourse.

Each input universe and output universe of discourse may be divided into seven Fuzzy subsets. These are:

1. Positive Big : “PB”
2. Positive Medium : “PM”
3. Positive Small : “PS”
4. Zero : “ZE”
5. Negative Small : “NS”
6. Negative Medium : “NM”
7. Negative Big : “NB”

The partition of Fuzzy subsets and the shape of the membership functions are shown in Figure 3. The triangular shape of the membership functions of this arrangement presumes that for any particular input there is only one dominant Fuzzy subset.

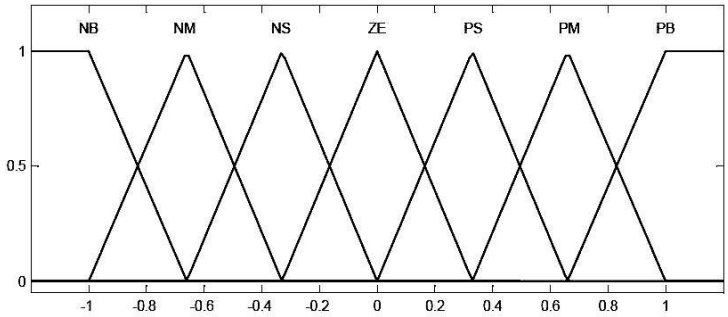


**Figure 3: The membership functions for input signals E, CE and output u.**

The above IF-THEN rule is a fuzzy description of control logic representing the human experts qualitative knowledge. For different (e) and (ce) values, a set of fuzzy control rules can be obtained as shown in Table 1.
The control output is calculated using the minimum operation for fuzzy implication and center of gravity defuzification.

**Table 1: Fuzzy rule table**

| E / ce | NB | NM | NS | ZE | PS | PM | PB |
|---|---|---|---|---|---|---|---|
| **NB** | NB | NB | NB | NB | NM | NS | ZE |
| **NM** | NB | NB | NB | NM | NS | ZE | PS |
| **NS** | NB | NB | NM | NS | ZE | PS | PM |
| **ZE** | NB | NM | NS | ZE | PS | PM | PB |
| **PS** | NM | NS | ZE | PS | PM | PB | PB |
| **PM** | NS | ZE | PS | PM | PB | PB | PB |
| **PB** | ZE | PS | PM | PB | PB | PB | PB |

For n rules:

$$U(e, ce) = \frac{\sum_{i=1}^{n} c_i . \min\left(u_{Ai}(e), u_{Bi}(ce)\right)}{\sum_{i=1}^{n} \min\left(u_{Ai}(e), u_{Bi}(ce)\right)}$$

$c_i$ is defined as the value of u where the fuzzy set $C_i$ obtains a maximum (or median of the interval where maximum is reached).

### 3.2. Effect of Controller Parameter

Finding optimum adjustment of a controller gains for a given process is not trivial.
The influence of the Fuzzy controller parameters Ke, Kce and Kc are studied. Therefore, a computer simulation is done, for studying:

a. Effect of error gain Ke:
   The error gain sets the controllable range from $V_r - (\frac{1}{K_e})$ to $V_r + (\frac{1}{K_e})$ of the output voltage. Outside this range, the error of volt e is assumed to be saturated and is classified as one of the two extreme Fuzzy subsets, PB or NB.
b. Effect of error change gain Kce:
   The effect of changing the error gain sets the damping factor of the transient response. Small Kce results in oscillatory response, while a large Kce produces a damped response. Also, large Kce have a weak effect on increasing the rise time.

c. Effect of error output gain Kc:
The parameter Kc is simply the maximum change of control output of the Fuzzy controller, large Kc have a strong effect on decreasing the rise time which voltage up the response and vise versa. This parameter has a weak effect on damping.

### 3.3. Tuning of Fuzzy Logic Controllers

Tuning the Fuzzy controllers is a difficult task as it has many parameters to be tuned as follows:

a. Rule tuning: Changes of rules may affect the performance. However, it is not so easy to tune the rule base.
b. Membership function (MF) tuning: Changes of MF's may not affect the performance very much. It is also not very convenient to tune MF's.
c. Gain tuning: Changes of gain affect the performance greatly. As it is easier to tune the gain than the rule base and MF's. So, the gain tuning is the most common way for tuning Fuzzy controllers. Based on the above reasons, the general rule base shown in Table 1 and standard MF's shown in Figure 3 can be used for different applications, leaving the optimum tuning to the scaling gains.

Usually, in gain tuning, Ke and Kce are chosen first by trying to spread e all over the controlled range for best resolution, and do not let e out of the range too much in order to keep the control of it. The second step is tuning Kc until a desirable rise time is reached. As Fuzzy gives a velocity type control, so Kc is usually smaller.
Practically, it may be hard to get an optimum performance by the above tuning strategy because transient state and steady state need different control resolutions.
In transient state, large errors need a coarse control which employs coarser MF's, while in steady state, small errors need a fine control which employs finer MF's. The relationship between coarser and MF's is actually a constant and can be adjusted by the scaling gains.

## 4. Particle SWARM Optimization

In order to optimize the fuzzy logic controller, the Particle Swarm Optimization (PSO) *technique* has been used to optimize the scaling factors to convert the input variables from its real value to a normalized. Using PSO to optimize a fuzzy logic controller has been done before [13; 14], and is an interesting way to give better performance to a fuzzy logic system.
The original PSO algorithm is discovered through simplified social model simulation. It is related to the bird flocking, fishing schooling, and swarm theory. The PSO was first designed to simulate birds seeking food, which is defined as a "cornfield vector." The bird would find food through social cooperation with other birds around it (within its neighborhood). It was then expanded to multidimensional search. The topological rather than Euclidean neighborhood was utilized. The original PSO algorithm is described as below.

$$vid = vid + c1rand()(pid-xid) + c2Rand()(pgd-xid) \quad (16)$$

$$xid = xid + vid \quad (17)$$

where:
*c1* and *c2* are positive constants,
*rand*() *and Rand*() are two random functions in the range [0,1];
*Xi* = (*xi1, xi2, … , xiD*) represents the ith particle;

*Pi* = (*pi1, pi2, … , piD*) represents the best previous position (the position giving the best fitness value) of the ith particle; the symbol g represents the index of the best particle among all the particles in the population;
*Vi* = (*vi1, vi2, … , viD*) represents the rate of the position change (velocity) for particle *i*.

Equations (16, 17) are the equation describing the flying trajectory of a population of particles. Equation (16) describes how the velocity is dynamically updated and Equation (17) the position update of the "flying" particles. Equation (16) consists of three parts. The first part is the momentum part. The velocity can't be changed abruptly. It is changed from the current velocity. The second part is the "cognitive" part which represents private thinking of itself - learning from its own flying experience. The third part is the "social" part which represents the collaboration among particles - learning from group flying experience [15].
In Equation (16), if the sum of the three parts on the right side exceeds a constant value specified by user, then the velocity on that dimension is assigned to be ±*Vmax*. Where, particles' velocities on each dimension are clamped to a maximum velocity V*max*, which is an important parameter. This originally is the only parameter required to be adjusted by users. Big V*max* has particles have the potential to fly far past good solution areas while a small *Vmax* has particles have the potential to be trapped into local minima, therefore unable to fly into better solution areas. Usually a fixed constant value is used as the *Vmax*, but a welldesigned dynamically changing *Vmax* might improve the PSO's performance.
The PSO algorithm is simple in concept, easy to implement and computational efficient. The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on D dimensions in the problem space.
2. For each particle, evaluate the desired optimization fitness function in D variables.
3. Compare particle's fitness evaluation with its *pbest*. If current value is better than *pbest*, then set *pbest* equal to the current value, and *Pi* equals to the current location *Xi* in *D*-dimensional space.
4. Identify the particle in the neighborhood with the best success so far, and assign its index to the variable g.
5. Change the velocity and position of the particle according to Equation (16) and (17).
6. Loop to step 2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

Like the other evolutionary algorithms, PSO algorithms is a population based search algorithm with random initialization, and there is interactions among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to generation. Furthermore, compared with the other evolutionary algorithms, e.g. evolutionary programming, the original version of PSO is faster in initial convergence while slower in fine-tuning.
Velocity changes of a PSO consist of three parts, the "social" part, the "cognitive" part, and the momentum part. The balance among these parts determines the balance of the global and local search ability, therefore the performance of a PSO.
The first new parameter added into the original PSO algorithm is the inertia weight. The dynamic equation of PSO with inertia weight is modified to be:

vid = wvid + c1rand()(pid-xid) + c2Rand()(pgd-xid) 18

$$xid = xid + vid \quad (19)$$

Equations (18, 19) are the same as the Equations (16, 17) except a new parameter, inertia weight w. The inertia weight is introduced to balance between the global and local search abilities. The large inertia weight facilitates global search while the small inertia weight facilitates local search. The introduction of the inertia weight also eliminates the requirement of carefully setting the maximum velocity Vmax each time the PSO algorithm is used. The Vmax can be simply set to the value of the dynamic range of each variable and the PSO algorithm still performs well enough if not better. Figure 4 shows the flowchart of the proposed PSO algorithm [16; 17; 18].

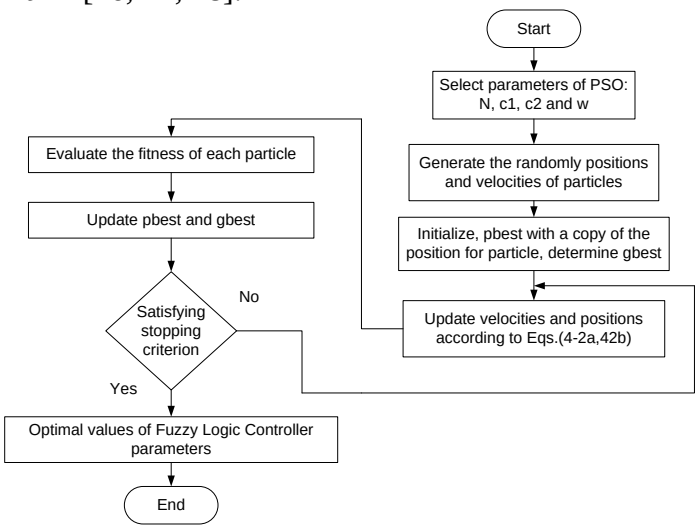


**Figure 4 : Flowchart of the proposed PSO technique.**

## 5. Simulation Results

The simulation results of the drive system under study are investigated when the system is equipped with three proposed controllers. These are:

a. Fuzzy logic controller, and
b. Fuzzy-PSO controller.

For each control system, the following case studies are considered:

a. 10% loading.
b. 50% loading.
c. 100% loading.

In the previous case studies, the proposed controllers are introduced when the system is operating at no load. The load is applied after 0.1 second with a value according to each case study.

### 5.1. Fuzzy Logic Controller

The response of the BLDC motor with FLC is represented by speed following the application of 10%, 50% and 100% step increase in load is shown in Figure 5. It can be seen from the investigation of these curves that the system follows the command reference. Figure 5 illustrates that the speed is first decreased due to sudden increase in load, then its recovered by returning to steady state (reference voltage), with the effect of fuzzy logic controller, within 10 m.sec to 25m.sec as a recovery time.

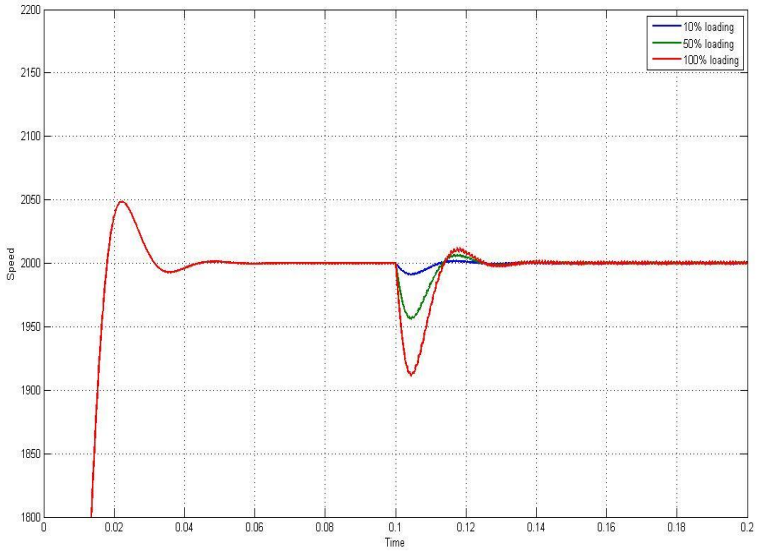


**Figure 5: Speed / time response with fuzzy logic controller at different loading conditions.**

## 5.2. Fuzzy-PSO Controller

The response of the BLDC motor is represented by speed with Fuzzy logic controller optimized with PSO following the application 10%, 50% and 100% step increase in load is shown in Figure 6. It can be seen from examination of these curves that the system follows the command reference. It is obvious that the damping characteristic is improved with Fuzzy-PSO controller, within 6 m.sec to 16m.sec as a recovery time.

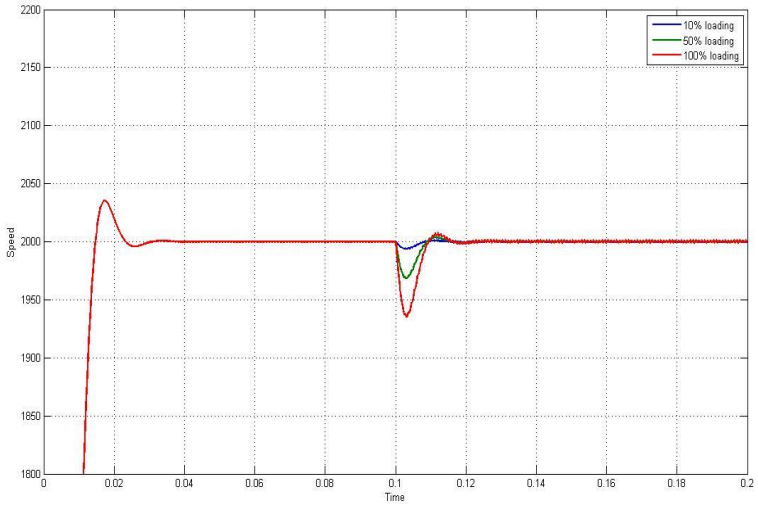


**Figure 6 Speed / time response with Fuzzy-PSO controller at different loading conditions.**

## 5.3. Comparative Study

In this section, a comparative study between the performance characteristic of the BLDC motor under the influence of the control techniques mentioned before presented and discussed. This comparative study is done between Fuzzy logic, and Fuzzy-PSO controllers. In Figure 7 a comparison between the BLDC motor performance as presented by its speed/

time response is demonstrated when equipped with Fuzzy logic and Fuzzy-PSO controllers. In This case study the machine is assumed to loaded as the previous cases. It can be seen from this figure the transient performance of the system when equipped with the different control strategies mentioned before is acceptable. However, a better transient response and damping characteristic is obtained with Fuzzy-PSO controller. The tuning difficulties associated with Fuzzy logic controllers to the variations of the system operating conditions, makes the application of Fuzzy-PSO controller is preferable.

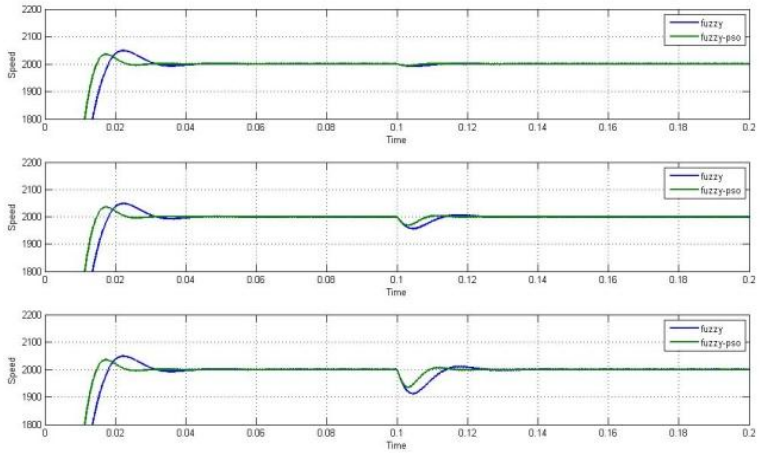


**Figure 7: Speed / time response with Fuzzy logic & Fuzzy-PSO controller at different loading conditions.**

## 6. Conclusions

This paper has presented investigation of two different control strategies on BLDC motor drive. These are Fuzzy logic and Fuzzy-PSO controllers. The results have been accomplished using computer simulation. Fuzzy logic and Fuzzy-PSO controllers are developed. This paper has also shown that it is possible to optimize the scaling factors of Fuzzy logic controller using particle swarm optimization. The influence of these control techniques on the performance characteristics of the system under consideration of various operating condition has been examined. The obtained results show that the transient response with Fuzzy-PSO controller is better than that of other controller. Moreover, a good damping performance is also obtained.

## 7. References

[1] Ko-Jong-Sun, Lee-Jung-Hoon, Chung-Se-Kyo, Youn-Myung-Joong. Robust digital position control of brushless DC motor with dead beat load torque observer. s.l. : IEEE Transaction on Industerial Electronics, Oct. 1993. 10.1109/41.238020.

[2] G. Yu, and R. Hwang,. Optimal PID speed control of brush less DC motors using LQR approach. Proc. IEEE Int. Conf. Systems, Man and Cybernetics. pp. 473-478, 2004.

[3] Swamy-CL-Putta, Sinh-Bhim and Singh-BP. swamy-cl-pufuzzy based sliding mode control of permanent magnet brushless dc motor. s.l. : Jornal of the Institution of eletronics and Telecommunication Engineers, Jul-Aug.1995. p 245-252.

[4] Rubaai, A., Ricketts, D. and Kankam, M.D. Experimental verification of hybrid fuzzy control strategy for a high performance brushless DC drive system. s.l. : IEEE Transaction on Industry Applications, March-April 2001. p 503-512.

[5] J. Zhou, Y. Wang. Real-time nonlinear adaptive backstepping speed control for a PM synchronous motor. Singapore : Elsevier Ltd. Control Engineering Practice, October 2005. P. 1259-1269.

[6] S. Poonsawat, and T. Kulworawanichpong. Speed Regulation of a Small BLDC Motor using Genetic-Based Proportional Control. s.l. : PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, JULY 2008. 1307-6884.

[7] Karunadasa, J.P. and Renfrew, A.C. Design and implementation of microprocessor based sliding modecontroller for brushless servomotor. s.l. : Electric Power Applications, IEE Proceedings B, Nov 1991. P. 345 - 363.

[8] Nandam, P. K., & Sen, P. C. A comparative study of a Lunberger observer and adaptive observer-based variable structure speed control system using a self-controlled synchronous motor. s.l. : IEEE transactions on industrial electronics, 1990. pp. 127-132.

[9] Cetin Elmas, Oguz Ustun , Hasan H. Sayan. A neuro-fuzzy controller for speed control of a permanent magnet synchronous motor drive. Turkey : Elsevier Ltd., 2006. 10.1016.

[10] Cetin Elmas, Oguz Ustun. A hybrid controller for the speed control of a permanent magnet synchronous motor drive. Ankara, Turkey : Elsevier Ltd., 2007. doi:10.1016/j.

[11] Mehdi Nasri, Hossein Nezamabadi-pour, and Malihe Maghfoori. A PSO-Based Optimum Design of PID Controller for a Linear Brushless DC Motor. s.l. : PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, APRIL 2007. 1307-6884.

[12] Portillo, Alberto A. Using the particle swarm optimizer to solve for PID values for a BLDC motor. s.l. : UNIVERSITY OF TEXAS SAN ANTONIO, May 2008.

[13] Venayagamoorthy, G.K. and Doctor, S. Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller. s.l. : IEEE Industry Applications Conference, Oct. 2004. 10.1109/IAS.2004.1348565.

[14] Welch, R. and Venayagamoorthy, G.K. A Fuzzy-PSO Based Controller for a Grid Independent Photovoltaic System. s.l. : Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, April 2007. 10.1109/SIS.2007.367942.

[15] Eberhart, Shi Y. and R. A. Parameter Selection in Particle Swarm Optimization. s.l. : Proceedings of Evolutionary Programming VII, IEEE, 1998. pp. 591-600.

[16] H. Shayeghi, A. Pirayeshnegab , A. Jalili , H.A. Shayanfar. Application of PSO technique for GEP in restructured power systems. s.l. : Energy Conversion and Management (Elsevier Ltd.), 2009. 50 (2009) 2127–2135.

[17] H. Shayeghi, H.A. Shayanfar, S. Jalilzadeh , A. Safari. A PSO based unified power flow controller for damping of power system oscillations. s.l. : Energy Conversion and Management (Elsevier Ltd.), 2009. 50 (2009) 2583–2592.

[18] H. Shayeghi, H.A. Shayanfar , A. Safari , R. Aghmasheh. A robust PSSs design using PSO in a multi-machine environment. s.l. : Energy Conversion and Management (Elsevier Ltd.), 2010. 51 (2010) 696–702.