Logic Journal of IGPL Advance Access published July 8, 2010

Optimal gain tuning of PI speed controller in induction motor drives using particle swarm optimization

Radha Thangaraj[1,*], Thanga Raj Chelliah[1], Millie Pant[1], Ajith Abraham[2] and Crina Grosan[3]

[1]*Indian Institute of Technology Roorkee, India 247001,* [2]*Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Washington, USA and* [3]*Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania*

Abstract

This article presents particle swarm optimization (PSO)-based optimal gain tuning of proportional integral (PI) speed controller in an induction motor (IM) drive (30 hp) with mine hoist load diagram. Optimization considers the load and speed variations, and provides appropriate gains to the speed controller to obtain good dynamic performance of the motor. IM performance is checked with the optimal gains through the simulation studies in MATLAB/SIMULINK environment. Results are compared with hand tuning (fixed gains) and fuzzy logic (FL) speed controller. Hybrid of FL and PSO-based PI controller for the speed control of given motor is also performed to eliminate the drawbacks of PI controller (overshoot and undershoot) and FL controller (steady-state error). From the simulation studies, hybrid controller produces better performance in terms of rise time, overshoot and settling time.

Keywords: Fuzzy logic, gain tuning, induction motor drives, particle swarm optimization, speed controller.

1 Introduction

Optimization is one of the most discussed topics in engineering and applied research. Many engineering problems can be formulated as optimization problems, e.g. economic dispatch problem, pressure vessel design, VLSI design, communication system, etc. These problems when subjected to a suitable optimization algorithm help in improving the quality of solution. Due to this reason the Engineering community has shown a significant interest in soft computing techniques. In particular, there has been a focus on evolutionary algorithms (EAs) for obtaining the global optimum solution to the problem, because in many cases it is not only desirable but also necessary to obtain the global optimal solution. Evolutionary algorithms have also become popular because of their advantages over the traditional optimization techniques such as decent method, quadratic programming approach, etc.

Some important differences of EAs over classical optimization techniques are as follows:

- Evolutionary algorithms start with a population of points, whereas the classical optimization techniques start with a single point.

*E-mail: t.radha@ieee.org

doi:10.1093/jigpal/jzq031

- No initial guess is needed for EAs; however, a suitable initial guess is needed in most of the classical optimization techniques.
- EAs do not require an auxiliary knowledge like differentiability or continuity of the problem, on the other hand classical optimization techniques depend on the auxiliary knowledge of the problem.
- The generic nature of EAs makes them applicable to a wider variety of problems, whereas classical optimization techniques are problem specific.

Some common EAs are genetic algorithms, evolutionary programming, particle swarm optimization (PSO), differential evolution, bacterial foraging, etc. These algorithms have been successfully applied for solving numerical benchmark problems and real-life problems. Several attempts have been made to compare the performance of these algorithms with each other [1, 3, 7, 11, 12, 17, 18]. How the artificial intelligence, particularly neural network, provides interesting solutions in the computer security problems are discussed in [4, 5]. In the present study, we investigate the performance of PSO for optimizing the PI speed controller gains of the induction motor (IM).

IM can be considered as one of the largest consumers of electrical energy due to its well-known advantages including robustness, reliability, low price and maintenance free operation. The IMs are used in both industrial and commercial sectors in a wide range of applications, such as fans, compressors, pumps, conveyors, winders, mills, transports, elevators, home appliances and office equipments. In some applications of vector-controlled IM like mine hoist and high dynamic performances (torque and speed control) are required. Hence, the research potential of the drive is especially towards development of speed controller, so that performance of the motor is optimized. In this article, PI gains (optimal values) for various operating regions of mine hoist load are obtained off-line by PSO based on the speed error and its derivative of IM. The optimized gain values are arranged in a look-up table and are fed to the controller to simulate the drive.

The remaining of the article is organized as follows: in Section 2, a brief overview of PSO is presented; Section 3 gives the mathematical models of IM and speed controllers, results are given in Section 4. Finally, the article concludes with Section 5. Pseudo-code of PSO algorithm is given in Appendix A.

2 PSO

PSO was first suggested by Kennedy and Eberhart in 1995 [6]. The mechanism of PSO is inspired from the complex social behaviour shown by the natural species. For a D-dimensional search space the position of the i-th particle is represented as $X_i=(x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle maintains a memory of its previous best position $P_i=(p_{i1}, p_{i2}, \ldots, p_{iD})$ and a velocity $V_i=(v_{i1}, v_{i2}, \ldots, v_{iD})$ along each dimension. At each iteration, the P vector of the particle with best fitness in the local neighbourhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector are given by

$$v_{id}=\omega v_{id}+c_1 r_1(p_{id}-x_{id})+c_2 r_2(p_{gd}-x_{id}) \quad (1)$$

$$x_{id}=x_{id}+v_{id} \quad (2)$$

The first part of (1) represents the inertia of the previous velocity, the second part is tells us about the personal thinking of the particle and the third part represents the cooperation among particles and is, therefore, named as the social component. Acceleration constants c_1, c_2 and inertia weight ω are predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range $[0, 1]$. Pseudo-code of PSO algorithm used in this study is available in Appendix A.

3 IM drive system

Figure 1 shows the basic configuration of speed control of IM drive. The drive is controlled with two control loops, i.e. inner pulse width modulation (PWM) current control loop and outer speed control loop. Reference or command speed is compared with actual speed of the drive and speed error is processed through the speed controller. The output of the speed controller is torque command for the drive. The electrical torque of the drive is directly proportional to the q-axis current component (i_{qs}) of the IM. Dividing the torque command by torque constant, the q-axis current command is obtained. Gain tuning of PI speed controller is performed by the PSO algorithm.

3.1 Mathematical model of IM

The squirrel cage IM is modelled using direct and quadrature axes (dq) theory in the stationary reference frame, which needs fewer variables and hence analysis becomes easy. The voltage–current relationship in the stationary reference frame of the IM in terms of the dq

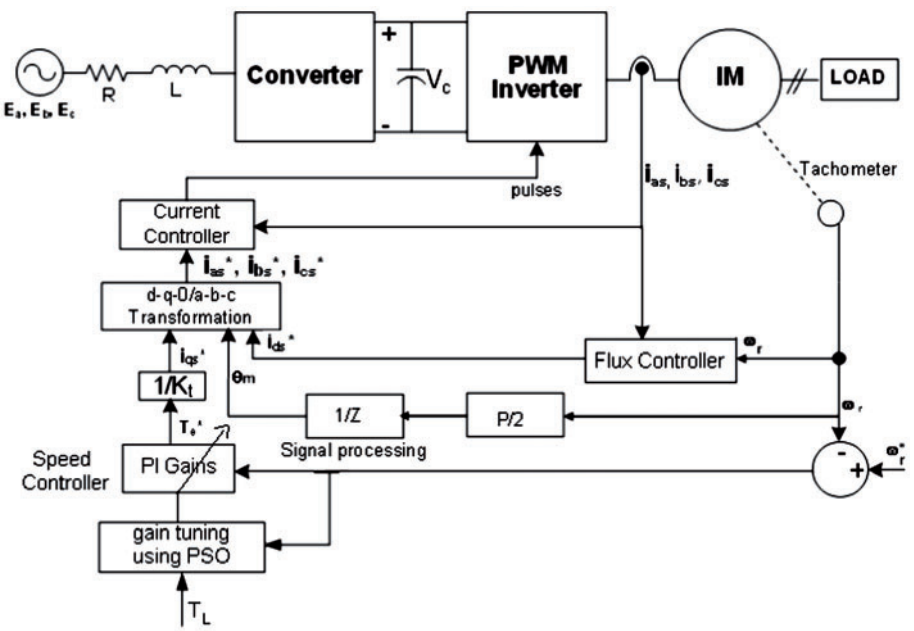

FIG. 1. Configuration of vector control of IM drive with speed control loop.

variable is expressed as [9]

$$V = [R]i + [L]pi + [G]\omega_r i + [F]\omega_c i \tag{3}$$

In the above equation, $[R]$ matrix consists of resistive elements, $[L]$ matrix consists of the coefficients of the derivative operator p, $[G]$ matrix has elements that are the coefficients of the electrical rotor speed ω_r and $[F]$ matrix is the frame matrix, which has the coefficients of the reference frame speed ω_c. In the stationary reference frame, the term $[F]\omega_c i$ is found to be identically zero, hence (3) can be rewritten as

$$[v] = [R][i] + [L]p[i] + \omega_r[G][i] \tag{4}$$

Rearranging (4), the current derivative vector can be expressed as follows:

$$p[i] = [L]^{-1}\left\{[v] - [R][i] - \omega_r[G][i]\right\} \tag{5}$$

In the above equation, 'p' is the differential operator ($\mathrm{d}/\mathrm{d}t$) and 'ω_r' is the rotor speed in electrical 'rad/s'. Three-phase IM is assumed to have balanced windings and connected with balance supply voltages, thus the zero sequence components are zero.

The electromagnetic torque is obtained by

$$T_e = \frac{3}{2}\frac{P}{2}L_m(i_{qs}i_{dr} - i_{ds}i_{qr}) \tag{6}$$

At the steady-state condition of the motor, (6) can be rewritten as

$$T_e = K_t i_{qs} \tag{7}$$

where $K_t = \frac{3}{2}\frac{P}{2}L_m^2 i_{ds}$ is a torque constant, which depends on air-gap flux. In the present study, air-gap flux is also adjusted to run the motor at optimal efficiency. Hence, this constant is valid only for steady-state operation. P is the number of poles in the motor.

3.2 Speed controller

Proportional integral (PI) controller can be used to control the speed of IM. The PI and differential (PID) controller is normally avoided because differentiation can be problematic when input command is a step. Generally, the speed error, which is the difference of reference speed ($\omega_{r(n)}{}^*$) and actual speed ($\omega_{r(n)}$), is given as input to the controllers. These speed controllers process the speed error and give torque value as an input. Then the torque value is fed to the limiter, which gives the final value of reference torque. The speed error and change in speed error at n-th instant of time are given as

$$\omega_{re(n)} = \omega^*_{r(n)} - \omega_{r(n)} \tag{8}$$

$$\Delta\omega_{re(n)} = \omega_{re(n)} - \omega_{re(n-1)} \tag{9}$$

This article considers three types of speed control methods for simulation study: PI controller with PSO and hand tuning, fuzzy speed controller and hybrid controller [hybridization of fuzzy logic (FL) and PI].

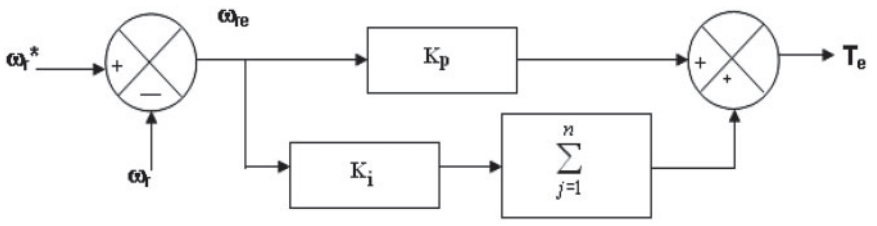

FIG. 2. Block diagram of PI speed controller.

3.2.1 PI controller

The general block diagram of the PI speed controller is shown in Figure 2 [14]. The output of the speed controller (torque command) at n-th instant is expressed as follows:

$$T_{e(n)} = T_{e(n-1)} + K_p \Delta\omega_{re(n)} + K_i \omega_{re(n)} \tag{10}$$

where $T_{e(n)}$ is the torque output of the controller at the n-th instant, and K_p and K_i the proportional and integral gain constants, respectively.

A limit of the torque command is imposed as

$$T_{e(n+1)} = \begin{cases} T_{e\max} & \text{for} \quad T_{e(n+1)} \geq T_{e\max} \\ -T_{e\max} & \text{for} \quad T_{e(n+1)} \leq -T_{e\max} \end{cases} \tag{11}$$

The gains of PI controller shown in (10) can be selected by many methods such as trial and error method, Ziegler–Nichols method and evolutionary techniques-based searching. The numerical values of these controller gains depend on the ratings of the motor.

3.2.2 FL speed controller

The PI speed controller, which has been discussed in the previous section, is simple in operation and has zero steady-state error when operating on load. But the disadvantages of this PI controller is the occurrence of overshoot while starting, undershoot while load application and overshoot again while load removal. Furthermore, it requires motor model to determine its gains and is more sensitive to parameter variations, load disturbances and suffer from poor performance when applied directly to systems with significant non-linearities [9, 14]. These disadvantages of PI controller can be eliminated with the help of a FL controller, which need not require model of the drive and can handle non-linearity of arbitrary complexity.

Fuzzy rules of this controller are shown in Figure 3 [8, 15] where the fuzzy variables: NB stands for negative-big, NM for negative-medium, NS for negative-small, ZE for zero, PB for positive-big, PM for positive-medium and PS for positive-small.

3.3 Hybrid speed controller

To take over the advantages present in both PI (zero steady-state error) and FL (negligible overshoot and undershoot) controllers, a hybridization of PI and FL controllers, called fuzzy pre-compensated PI (FPPI) controller, is done and is used as a single controller. In this

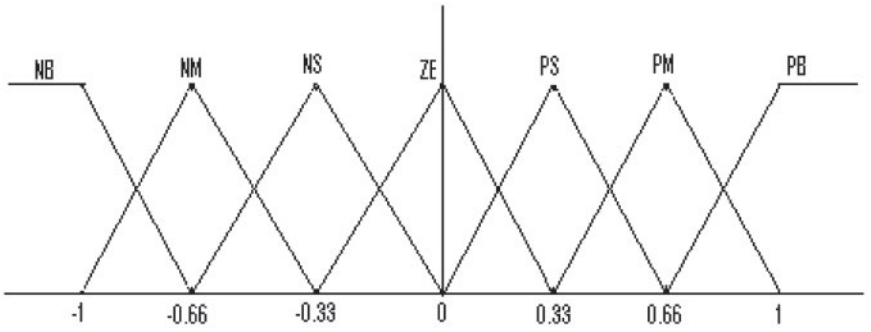

FIG. 3. Fuzzy sets considered for speed control.

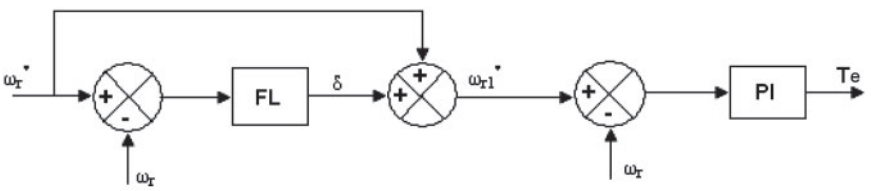

FIG. 4. Block diagram of hybrid (FPPI) speed controller.

controller, FL is used for pre-compensation [8, 10, 13, 15, 16] of reference speed, which means that the reference speed signal (ω_r^*) is altered in advance in accordance with the rotor speed (ω_r), so that a new reference speed signal (ω_{r1}^*) is obtained and the main control action is performed by PI controller. Some specific features such as overshoot and undershoot occurring in the speed response, which are obtained with PI controller can be eliminated [15] and this controller is much useful to mine hoist load where torque/speed of the motor varies time to time.

3.3.1 Design of hybrid speed controller

As usual, the inputs to the FL are speed error ($\omega_{re(n)}^*$) and the change in speed error ($\Delta\omega_{e(n)}$), the output of the FL controller is added to the reference speed to generate a pre-compensated reference speed (δ), which is to be used as a reference speed signal by the PI controller shown in Figure 4. The fuzzy pre-compensator can be mathematically modelled as follows [8]:

Referring (8) and (9) for speed error and change in speed error, pre-compensated speed reference (δ) and updated new reference speed (ω_{r1}^*) can be calculated as

$$\delta_{(n)} = F\left[\omega_{re(n)}, \Delta\omega_{re(n)}\right] \tag{12}$$

$$\omega_{r1}^* = \delta_{(n)} + \omega_{r(n)}^* \tag{13}$$

where F is FL mapping

3.4 Parameter settings

3.4.1 Objective function

The performance of the IM varies according to PI controller gains and is judged by the value of Integral Time Absolute Error (ITAE). The performance index ITAE is chosen as objective function. The purpose of PSO algorithms is to minimize the objective function or maximize the fitness function, where fitness function is $1/(ITAE+1)$. If >5% overshoot occurs in starting speed response a 75% of the penalty is imposed to the fitness value. All particles of the population are decoded for K_p and K_i.

3.4.2 IM parameters

Torque constant	4.1 Nm/A
Phase Stator resistance	0.251 Ω
Phase rotor resistance	0.249 Ω
Phase inductance	1.4 mH
Mutual inductance	41.6 mH
Number of poles	4
Moment of inertia of motor	0.305 kg m^2
Rated speed	314 rad/s (electrical)

3.4.3 Parameter settings for the PSO algorithm

Swarm size: 20.
Inertia weight (w): linearly decreasing (0.9–0.4)
Acceleration constants: $c_1 = c_2 = 2.0$.
Since PSO algorithm is stochastic in nature, more than one execution is needed to reach to a solution. A maximum of 25 iterations were fixed for the optimization algorithm.

The algorithm was implemented using Turbo C++ on a PC compatible with Pentium IV, a 3.2 GHz processor and 2 GB of RAM.

4 Experimental results and discussions

To illustrate the importance of efficient speed controllers in the industrial drives, we considered the load diagram of hoist in a mineral industry (Figure 5) [2, 16]. A motor, normally high rated (in MW), is employed with mine hoist and is operated with variable load and speed as shown in Figure 5. Region 't_1' of this load diagram offers 1.5 times rated load and half-rated speed of the motor. This article considers a 30 hp motor and focuses all regions and particularly the instant at which step changes occur in the torque and speed. Table 1 shows the results of PSO algorithm in terms of control parameters K_p and K_i. These gains are used as a look-up table in the simulation study of IM. The motor is accelerated to the step speed command of 0.5, 1.0 and 0.5 pu corresponding to the regions of Figure 5 from start. Similar to speed command, torque command is also changed in accordance with the load diagram. The given load diagram is initiated at 1.0 s in the simulation study. Figures 6–9 show the simulation results for the motor operated with optimal PI gain obtained from PSO, hand

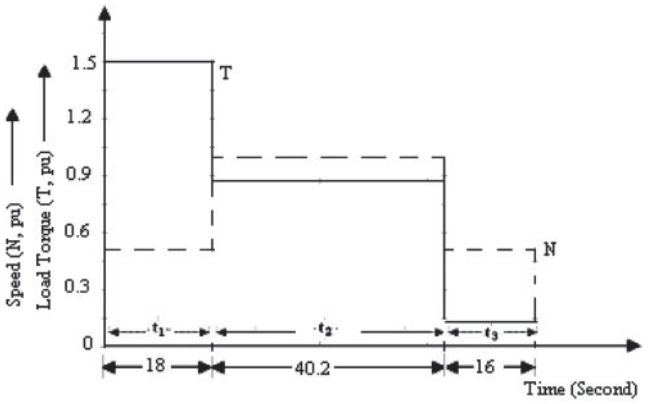

FIG. 5. Mine hoist load diagram.

TABLE 1. Optimal gain (K_p and K_i) values obtained from PSO Algorithm

Region	Torque (Nm)	Speed (rad/s)	Optimal gains	
			K_p	K_i
Region 0 (initial)	0	250	25.2545	0.203491
Region 1(t_1)	210	125	26.031	0.439
Region 2 (t_2)	125	250	25	0.4868
Region 3 (t_3)	20	125	26.381	0.3222

tuning, FL speed controller and hybrid controller. The figures show speed and developed torque from top to bottom order.

4.1 Results of PI controller with hand tuning gains

The PI speed controller gain parameters of (10) are selected by trial and error basis by observing their effects on the response of the drive. The values of K_p and K_i obtained from the hand tuning are 25 and 0.4, respectively. The dynamic performances of the motor with hand gain tuning of PI controller is shown in Figure 6.

At the starting point of simulation (0 s), motor speed is reached to 127.5 rad/s, whereas the commanded speed is 125 rad/s. The torque response in this instant is raised up to 600 Nm with small oscillation before it settles. At 1.0 s, where the given load diagram is applied, motor speed undershoots by nearly 5 rad/s due to the presence of heavy load. The torque overshoots by 65 Nm (31%) and settles with small oscillation. At 2.0 s, where Region 2 of the load diagram is applied, motor speed overshoots up to 252.75 rad/s due to the load removal and also due to the increase in command speed. Torque at this instant overshoots up to 350 Nm, which is significantly higher than the commanded torque. The torque oscillates with the value 5 Nm, but the oscillation in the speed response is almost negligible. At Region 3 (3.5 s), both speed and torque responses undershoot due to the reductions of their desired commands.

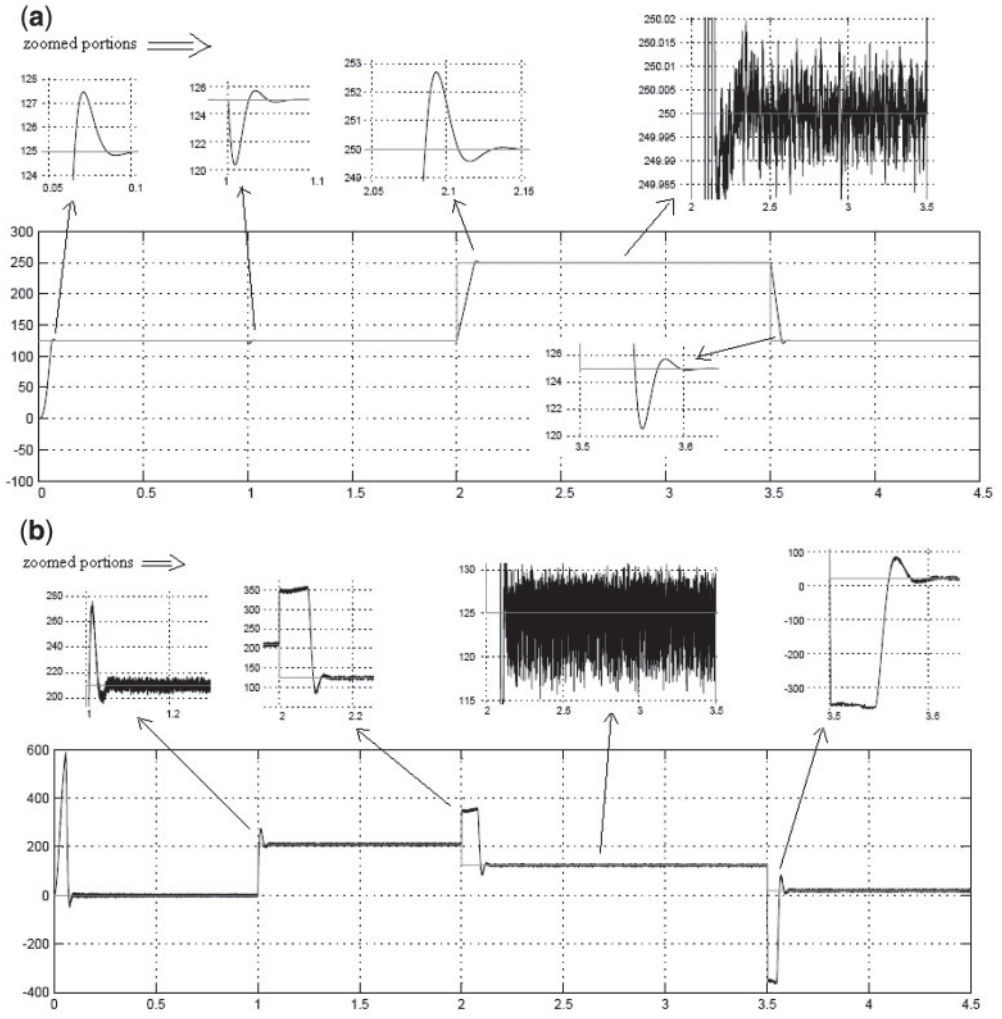

FIG. 6. Results for PI controller with hand tuning: (**a**) speed and (**b**) torque.

4.2 Results of PI controller with optimal gains

As mentioned earlier, optimal gains of PI speed controller are obtained from the PSO algorithm, which considers the variations in the load and speed requirements but do not considers the parameter variations due to external disturbances such as temperature variation, supply voltage, etc. in the motor and in the controllers. The results of present case are shown in Figure 7. At starting (0 s), motor speed is nearly equal to commanded value and is settled at 0.2 s. Improved torque response is also obtained with optimal gain than hand tuning at this instant. Starting torque overshoots up to 550 Nm at this case, whereas it reached to 600 Nm at hand tuning. At 1 s, motor responses are more or less same as hand tuning. At Region 2, the oscillation in the speed response is lower than hand tuning. At Region 3, speed response slightly undershoots by 3 rad/s. It is noted that overall dynamic performances of the motor

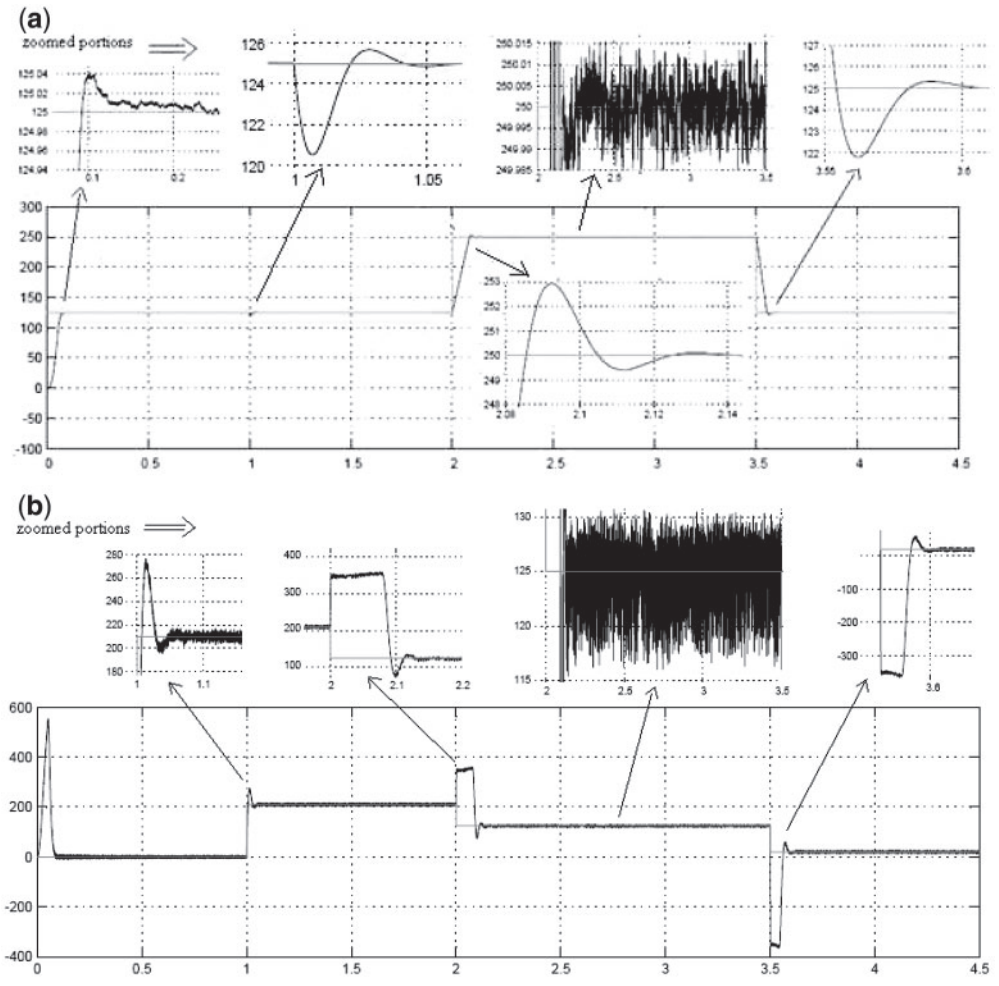

FIG. 7. Results of PI controller with PSO-based optimal gain tuning: (**a**) speed and (**b**) torque.

when operated at PI controller with optimal gains is better than the hand tuning and the steady-state error of speed response is zero.

4.3 Results of FL speed controller

The simulation results of speed and torque responses of the motor, which operate with FL speed controller are shown in Figure 8. For all the regions, there is no speed overshoot and ripples are negligible (main advantageous of FL controller), but it offers more settling time and steady-state speed error (disadvantageous of this controller), shown in Figure 8a. Steady-state speed errors are 1.25, 0.75 and 1.25 rad/s at the regions 1, 2 and 3, respectively.

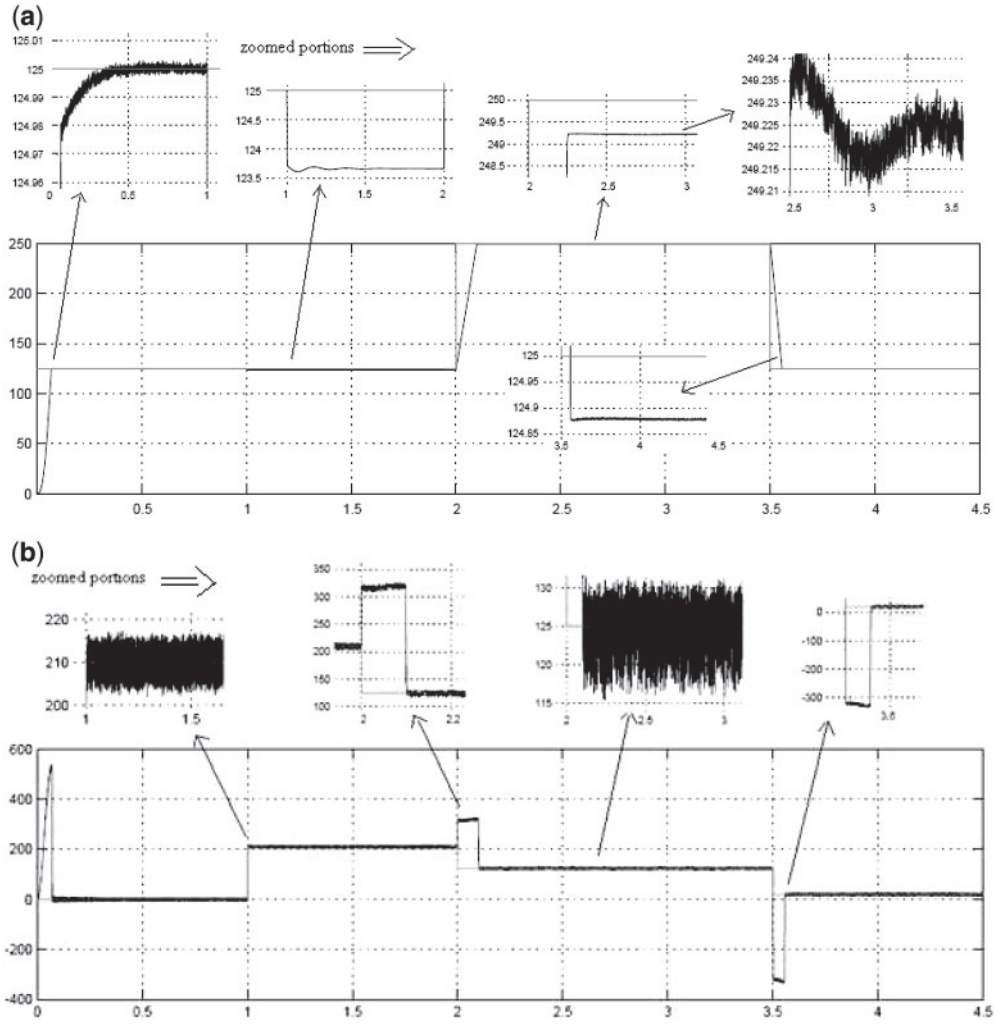

FIG. 8. Results of Fuzzy speed controller: (**a**) speed and (**b**) torque.

Overshoot and undershoot occurred in the torque response but are still better than PI controller with and without optimal tuning of gains.

4.4 Results of hybrid speed controller based on fuzzy pre-compensation

The results of hybrid speed controller (HC) are shown in Figure 9. At starting of motor, speed response has no overshoot and settles faster in comparison with FL controller. It is also noted that there is no steady-state error in the speed response throughout the operation when hybrid controller is activated. Furthermore, no oscillation in the torque response before it finally settles (shown in Figure 9a), whereas oscillation occurred at PI controller with hand tuning. At 1 s, speed undershoots just by 1 rad/s whereas, for PI controller this value was

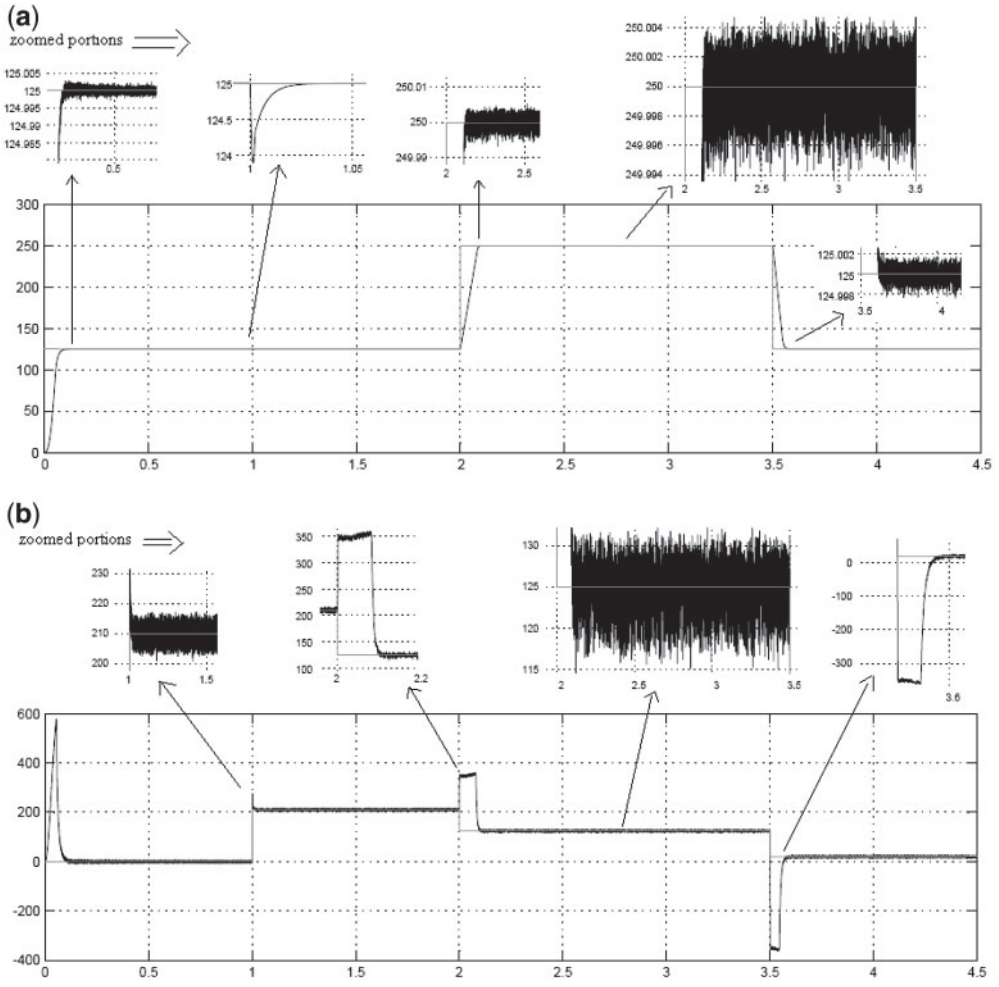

FIG. 9. Results of HC: (**a**) speed and (**b**) torque.

5 rad/s. Good Torque response is obtained with HC controller at this instant. In regions 2 and 3, speed response is better than PI and FL controllers. There is a negligible ripple in speed response at HC in comparison with PI and FL controllers

5 Conclusions

This article presented PSO-based optimal gain tuning of PI speed controller in an IM drive (30 hp) for a mine hoist load diagram. IM performance was checked with the optimal gains through the simulation studies in MATLAB/SIMULINK environment. Results were compared with hand tuning (fixed gains) and FL speed controller. Hybridization of PI and FL controllers was done and used as a single controller by extracting the advantages present in PI (zero steady-state error) and FL (negligible overshoot and undershoot) controllers.

From the simulation studies, hybrid controller produced better performances in terms of rise time, overshoot, undershoot and settling time. A brief description of PSO algorithm and the definition of the problem were also given.

For practical implementation, the values of PI gains obtained from PSO at different speed and torque commands can be stored in the memory of a digital signal processor and used to operate the motor with optimal gains according to desired speed and torque. Unlike conventional fixed gain PI controller, tuning of PI gains using PSO is insensitive to step change of speed command and is preferred for the normal operation of the drive. Furthermore, online tuning is highly recommended to maintain good stability of the drive during parameter variations in the motor and in the controllers of the drive.

References

[1] A. Biswas, S. Dasgupta, S. Das, and A. Abraham. A synergy of differential evolution and bacterial foraging algorithm for global optimization. *Neural Network World*, **17**, 607–626, 2007.

[2] M. Chilikin. *Electric Drives*, MIR Publishers, 1976.

[3] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, **19**, 43–53, 2005.

[4] Á. Herrero, E. Corchado, M. A. Pellicer, and A. Abraham. MOVIH-IDS: a mobile visualization hybrid intrusion detection system. *Neurocomputing*, **72**, 2775–2784, 2009.

[5] A. Herrero, E. Corchado, L. Saiz, and A. Abraham. *DIPIP: a Neural Knowledge Management Model Framework for Decision Support, Computational Intelligence*, vol. 26, pp. 26–56, Blackwell Publishing, 2010.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV, pp. 1942–1948. IEEE Service Center, 1995.

[7] A. Khosla, S. Kumar, and K. R. Ghosh. A Comparison of Computational Efforts between Particle Swarm Optimization and Genetic Algorithm for Identification of Fuzzy Models. In *Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 245–250, 2007.

[8] J.-H. Kim, K.-C. Kim, and E. K. P. Chong. Fuzzy pre-compensated PID controllers', *IEEE Transactions on Control System*, **2**, 406–411, 1994.

[9] R. Krishnan. *Electric motor drives - Modeling, Analysis, and Control*. Prentice Hall of India publication, 2003.

[10] S.-W. Lee, M.-J. Jeong, B.-I. Jang, C.-H. Yoo, S.-G. Kim, and Y.-S. Park. Fuzzy precompensated PI controller for a variable capacity heat pump. In *Proceedings of the IEEE Conference on Control Applications*, pp. 953–957, 1998.

[11] H. Liu, A. Abraham, and M. Clerc. Chaotic dynamic characteristics in swarm intelligence. *Applied Soft Computing Journal*, **7**, 1019–1026, 2007.

[12] H. Liu, A. Abraham, and W. Zhang. A fuzzy adaptive turbulent particle swarm optimization. *International Journal of Innovative Computing and Applications*, **1**, 39–47, 2007.

[13] K. V. Naresh. *Investigation of SVM-PWM based Induction Motor Drives*. M Tech Dissertation, Indian Institute of Technology Roorkee, 2007.

[14] M. Pant, R. Thangaraj, and A. Abraham. Optimal tuning of pi speed controller using nature inspired heuristics. In *Proceeedings of the Eighth International Conference on*

Intelligent Systems Design and Applications, pp. 420–425. IEEE Computer Society Press, 2008.

[15] B. Singh and G. Choudhuri. Fuzzy logic based speed controllers for vector controlled induction motor drive. *IETE Journal of Research*, **48**, 441–447, 2002.

[16] C. Thanga Raj, S. P. Stivastava, and P. Agarwal. Particle swarm and fuzzy logic based optimal energy control of induction motor for a mine hoist load diagram. *IAENG International Journal of Computer Science*, **36**, 17–25, 2009.

[17] O. Uysal1 and S. Bulkan. Comparison of genetic algorithm and particle swarm optimization for bicriteria permutation flowshop scheduling problem. *International Journal of Computational Intelligence Research*, **4**, 159–175, 2008.

[18] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1980–1987, 2004.

Received 19 July 2009

Appendix A

A.1 Pseudo code of PSO algorithm used in this study

Step1: Initialization.
For each particle i in the population:
Step1.1: Initialize $X[i]$ with uniform distribution.
Step1.2: Initialize $V[i]$ randomly.
Step1.3: Evaluate the objective function of $X[i]$, and assigned the value to fitness$[i]$.
Step1.4: Initialize $P_{best}[i]$ with a copy of $X[i]$.
Step1.5: Initialize P_{best}_fitness$[i]$ with a copy of fitness$[i]$.
Step1.6: Initialize P_{gbest} with the index of the particle with the least fitness.
Step2: Repeat until stopping criterion is reached:
For each particle i:
Step 2.1: Update $V[i]$ and $X[i]$ according to (1) and (2).
Step2.2: Evaluate fitness$[i]$.
Step2.3: If fitness$[i]$ < P_{best}_fitness$[i]$ then $P_{best}[i] = X[i]$, P_{best}_fitness$[i]$ = fitness$[i]$.
Step2.4: Update P_{gbest} by the particle with current least fitness among the population.