OPTIMAL TUNING OF PI SPEED CONTROLLER USING NATURE INSPIRED HEURISTICS

Millie Pant[1], Radha Thangaraj[1] and Ajith Abraham[2]
[1]Department. of Paper Technology, IIT Roorkee, India
[2]Center of Excellence for Quantifiable Quality of Service,
Norwegian University of Science and Technology, Norway
millifpt@iitr.ernet.in, t.radha@ieee.org, ajith.abraham@ieee.org

Abstract

This paper presents a comparative study of three popular, Evolutionary Algorithms (EA); Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE) for optimal tuning of Proportional Integral (PI) speed controller in Permanent Magnet Synchronous Motor (PMSM) drives. A brief description of all the three algorithms and the definition of the problem are given.

1. Introduction

Optimization is one of the most discussed topics in engineering and applied research. Many engineering problems can be formulated as optimization problems for example Economic Dispatch problem, Pressure Vessel Design, VLSI design etc. these problems when subjected to a suitable optimization algorithm helps in improving the quality of solution. Due to this reason the Engineering community has shown a significant interest in optimization algorithms. In particular there has been a focus on Evolutionary algorithms 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 (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.
- 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 where as classical optimization techniques are problem specific.

Some common EAs are Genetic Algorithms (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Differential Evolution (DE) etc. these algorithms have been successfully applied for solving numerical bench mark problems and real life problems**.** Several attempts have been made to compare the performance of these algorithms with each other [1] - [3], etc. In this study we investigate the performance of PSO, DE and GA for optimizing the PI speed controller gains of the Permanent Magnet Synchronous Motor (PMSM).

The PMSM is of great concern for researchers and industrialists due to its advantages over other electric motors like induction motor, DC motor etc. The research potential of the drive is especially towards development of speed controller so that performance of the PMSM is optimized. In this paper we have used PSO, GA and DE off-line to determine the controller parameters (optimum value) based on speed error and its derivative of the PMSM.

The remaining of the paper is organized as follows: In Section 2 a brief overview of GA, PSO and DE is presented; Section 3 gives the mathematical model of PMSM, results are given in section 4. Finally the paper concludes with Section 5. Pseudo code of all the three algorithms is given in Appendix A.

2. Evolutionary Algorithms used for comparison

Evolutionary algorithms may be termed as general purpose algorithms for solving optimization problems. Each EA is assisted with special operators that are based on some natural phenomenon. These algorithms are iterative in nature and in each iteration the operators are invoked to reach to optimal (or near

978-0-7695-3382-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISDA.2008.69

optimal) solution. A brief description of the three EA used in this study is given in the following subsections:

2.1. Genetic algorithms

Genetic algorithms are perhaps the most commonly used EA for solving optimization problems. The natural phenomenon which forms the basis of GA is the concept of *survival of the fittest*. GAs were first suggested by John Holland and his colleagues in 1975 [4]. The main operators of GA are *Selection, Reproduction* and *Mutation*. GAs work with a population of solutions called *chromosomes*. The fitness of each chromosome is determined by evaluating it against an objective function. The chromosomes then exchange information through *crossover* or *mutation.* More detail on the working of GAs may be obtained from Goldberg [5] etc.

2.2. Particle Swarm Optimization

Particle swarm optimization (PSO) was first suggested by Kennedy and Eberhart in 1995 [6]. The mechanism of PSO is inspired from the complex social behavior shown by the natural species. For a D-dimensional search space the position of the ith particle is represented as $X_i = (x_{i1},x_{i2},..x_{iD})$. 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 neighborhood, 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 equation (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 of [0, 1].

2.3. Differential Evolution

Differential Evolution was proposed by Storn and Price [7]. It is a population based algorithm like genetic algorithms using the similar operators; crossover, mutation and selection. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operator [8]. DE works as follows: First, all individuals are initialized with uniformly distributed random numbers and evaluated using the fitness function provided. Then the following will be executed until maximum number of generation has been reached or an optimum solution is found.

For a D-dimensional search space, each target vector $x_{i,g}$, a mutant vector is generated by

$$v_{i,g+1} = x_{r_1,g} + F*(x_{r_2,g} - x_{r_3,g}) \quad (3)$$

where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ are randomly chosen integers, must be different from each other and also different from the running index i. F (>0) is a scaling factor which controls the amplification of the differential evolution $(x_{r_2,g} - x_{r_3,g})$. In order to increase the diversity of the perturbed parameter vectors, crossover is introduced [9]. The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,g+1}$,

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1} & if\ (rand_j \le CR)\ \ or\ \ (j = j_{rand}) \\ x_{ji,g} & if\ (rand_j > CR)\ \ and\ \ (j \ne j_{rand}) \end{cases}$$

where j = 1, 2,……, D; $rand_j \in [0,1]$; CR is the crossover constant takes values in the range [0, 1] and $j_{rand} \in (1,2,.....,D)$ is the randomly chosen index.

Selection is the step to choose the vector between the target vector and the trial vector with the aim of creating an individual for the next generation.

3. Permanent Magnet Synchronous Motor (PMSM)

3.1. Mathematical model of PMSM

The PMSM drive model consists of a PWM converter, a PWM inverter, a PMSM speed controller and a voltage controller. The PMSM drive receives power from three-phase AC supply and runs a mechanical load at regulated speed. The drive is developed so that it injects minimum current harmonics into the utility supply system and drives mechanical load with minimum ripples in torque and speed. The developed model of the drive system is used for design in voltage, current and speed controllers.

The mathematical model of PMSM in d-q synchronously rotating reference frame can be obtained from synchronous machine model. The PMSM can be

represented by the following set of nonlinear equations [10]:

$$p\omega_r = \frac{1}{J}(T_e - T_l - B * \omega_r) \quad (4)$$

$$T_e = \frac{3}{2}\frac{P}{2}(\psi_f + (L_d - L_q)i_d^s)i_q^s \quad (5)$$

$$pi_d^s = \frac{1}{L_d}(-Ri_d^s + \left(\frac{P}{2}\right)\omega_r L_d i_q^s + v_d^s) \quad (6)$$

$$pi_q^s = \frac{1}{L_d}(-Ri_q^s - \left(\frac{P}{2}\right)\omega_r L_d i_d^s + v_q^s - \left(\frac{P}{2}\right)\omega_r \psi_f) \quad (7)$$

where v_d^s, v_q^s are the q, d-axis voltages, L_d, L_q are the q, d-axis inductances, i_d^s, i_q^s are the q, d-axis currents, R is the stator resistance per phase, ψ_f is the constant flux linkage due to rotor permanent magnet, ω_r is the rotor speed, J is the moment of inertia, P represents number of poles, p is the differential operator, Te is the electromagnetic torque, T_l is the load torque, B is the viscous coefficient.

3.2. PI speed controller

The PI speed controller is the simplest speed controller in comparison to any other speed controller. The general block diagram of the PI speed controller is shown in Figure 1. The input of the PI controller is given below,

$$e(t) = \omega_{ref} - \omega_{act} \quad (8)$$

where ω_{ref} -- Desired (reference) speed of the motor
ω_{act} – actual speed of the motor

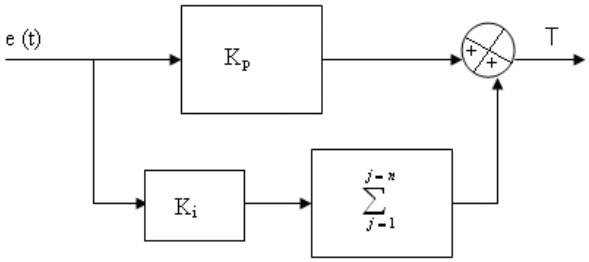

Figure 1 Block diagram of PI speed controller

The output of the PI controller is torque command which limited by a torque limiter with specified values corresponding to the motor's rating. The output of the speed controller at nth instant is expressed as follows:

$$T_{(n)} = T_{(n-1)} + K_p\{e(t)_n - e(t)_{(n-1)}\} + K_i e(t)_n \quad (9)$$

where $T_{(n)}$ is the torque output of the speed controller at the n^{th} instant.
K_p and K_i are proportional and integral gain constants
$e(t)_{(n-1)}$ is the speed error at (n-1)th instant.

The reference current generator decides the pulse width and triggers the inverter circuits so that required voltage will be applied to the motor. The motor speed can be observed by tacho generator and feedback to PI controller.

4. Optimization of PI gains using GA, PSO and DE

GA, PSO and DE are utilized off-line to determine the controller parameters (K_p and K_i) based on speed error and its derivative of the PMSM shown Figure 2.
The performance of the PMSM varies according to PI controller gains and is judged by the value of ITAE (Integral Time Absolute Error). The performance index ITAE is chosen as objective function. The purpose of stochastic algorithms is to minimize the objective function or maximize the fitness function, where fitness function is $1/(ITAE+1)$. As in [11], if more than 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.

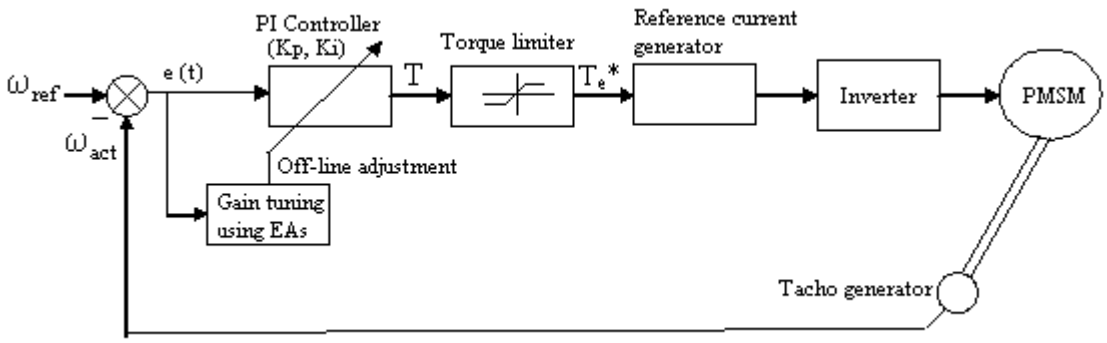

Figure 2 Speed controller of PMSM using PI with evolutionary tuning

4.1. Experimental Settings

PMSM parameters [11]:

Continuous stall current	3.40 A
Torque constant	0.75 Nm/A
Phase resistance	2.55 Ω
Phase inductance	5.15 mH
Number of poles	8
Moment of inertia of motor	0.062 x 10^{-3} kg-m^2
Total load inertia	1.914 x 10^{-3} kg-m^2
Damping friction	0.0041 N-m/rad.

GA settings [11]	PSO settings	DE settings
Population size: 20 CR: 0.7 MR: 0.005	Swarm size: 20. Inertia weight (w): linearly decreasing (0.9-0.4) Acceleration constants: $c_1 = c_2 = 2.0$.	Population size: 20 crossover constant: CR= 0.5 Scaling parameter : F = 0.5.

Parameter setting for the optimization algorithms:
As mentioned earlier, each EA requires certain sets of parameters which are defined in the beginning of the algorithm. Since all the three algorithms are stochastic in nature, more than one execution is needed to reach to a solution. A maximum of 25 iterations was fixed for all the three algorithms.

Computer Settings
All the three algorithms were implemented using Turbo C++ on a PC compatible with Pentium IV, a 3.2 GHz processor and 2 GB of RAM.

4.2. Simulation results

Table 1 shows the results of DE and PSO algorithms in terms of control parameters K_p, K_i and the fitness function value. The fitness values for 1st and 25th generation are shown in Figure 3 and 4 respectively.

Table 1 Comparison Results PSO, DE and GA

Algo rithm	Kp	Ki	Fitness value	Run time (sec)
GA [11]	0.0360	0.0032	0.0179	----
PSO	0.075816	0.004873	0.22494	35
DE	0.075543	0.002725	0.39032	36

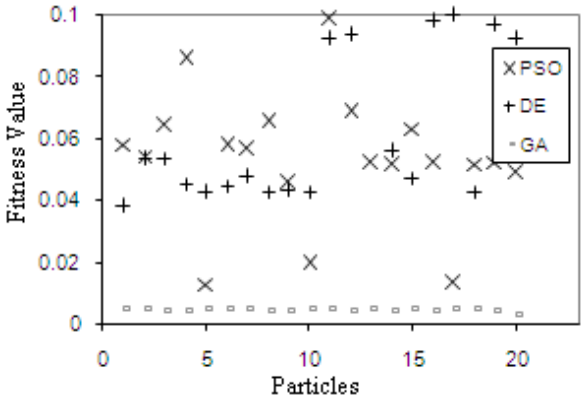

Figure 3 Fitness values corresponding to population at 1st generation

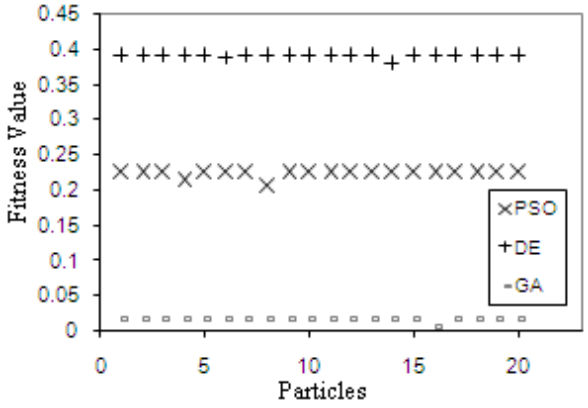

Figure 4 Fitness values corresponding to population at 25th generation

5. Discussion and Conclusion

The present study considers off-line adjustment of PI gains which is not considering the parameter variations in the motor and other controllers due to external disturbance like temperature variation, supply voltage, etc. we investigated the performance of GA, PSO and DE for the off-line tuning of PI gains on the basis of average fitness function value obtained and the CPU time taken during the execution of these programs. From the numerical results presented in Table 1, DE comes out as a clear winner in terms of both fitness function value (which is of maximization type) and average CPU time when considered all the algorithms individually. The second place goes to PSO and finally on the third place is GA. Thus as a concluding remark we would like to add that while solving real life problems using EA, it would be better to make a comparison of two or more techniques in order to reach to final solution.

Appendix A Pseudo codes of algorithms used in this study

(1) Pseudo code for Genetic Algorithm

Begin
Initialize the population
For each individual calculate the fitness value.
For i = 1 to maximum number of generations
Do Selection, Crossover, Mutation
End for
End.

(2) Pseudo code for Particle Swarm optimization

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 Pbest_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 equations (1) and (2).
Step2.2: Evaluate fitness[i].
Step2.3: If fitness[i] < Pbest_fitness[i] then $P_{best}[i]$ =X[i], Pbest_fitness[i] =fitness[i].
Step2.4: Update P_{gbest} by the particle with current least fitness among the population.

(3) Pseudo code for Differential Evolution

Initialize the population
Calculate the fitness value for each particle
Do
For i = 1 to number of particles
Do mutation, Crossover and Selection
End for.
Until stopping criteria is reached.

References

[1] J. Vesterstrom, R. Thomsen, "A Comparative study of Differential Evolution, Particle Swarm optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," In *Proc. IEEE Congr. Evolutionary Computation*, Portland, OR, Jun. 20 – 23, (2004), pg. 1980 – 1987.

[2] 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", Annual Meeting of the North American Fuzzy Information Processing Society, pp. 245 – 250, 2007.

[3] E. Elbeltagi, T. Hegazy and D. Grierson, "Comparison among five evolutionary-based optimization algorithms", Anvanced Engg. Informatics, Vol. 19, pp. 43 – 53, 2005.

[4] Holland, J. H., "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," Ann Arbor, MI: University of Michigan Press.

[5] Goldberg, D., "Genetic Algorithms in Search Optimization and Machine Learning," Addison Wesley Publishing Company, Reading, Massachutes.

[6] Kennedy, J. and Eberhart, R., "Particle Swarm Optimization," IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, 1995, pg. IV: 1942-1948.

[7] R. Storn and K. Price, "Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report, International Computer Science Institute, Berkley, 1995.

[8] D. Karaboga and S. Okdem, "A simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm", Turk J. Elec. Engin. 12(1), 2004, pp. 53 – 60.

[9] R. Storn and K. Price, "Differential Evolution – a simple and efficient Heuristic for global optimization over continuous spaces", Journal Global Optimization. 11, 1997, pp. 341 – 359.

[10] Rajesh Kumar, R. A Gupta and Bhim Singh, "Intelligent Tuned PID Controllers for PMSM Drive - A Critical Analysis," *IEEE Int. conference, PEDES*, 2006, pp. 2055-2060.

[11] A. N. Tiwari, "Investigations on Permanent Magnet Synchronous Motor Drive", Ph.D thesis, IIT Roorkee, 2003.