A Hybrid Bacterial Foraging - PSO Algorithm Based Tuning of Optimal FOPI Speed Controller

***Rajasekhar Anguluri*[1], *Ajith Abraham*[2] *and Vaclav Snasel*[3]**

Bacterial Foraging Optimization Algorithm (BFOA) has recently emerged as a very powerful technique for real parameter optimization. In order to overcome the delay in optimization and to further enhance the performance of BFO, this paper proposed a new hybrid algorithm combining the features of BFOA and Particle Swarm Optimization (PSO) for tuning a Fractional order speed controller in a Permanent Magnet Synchronous Motor (PMSM) Drive. Computer simulations illustrate the effectiveness of the proposed approach compared to that of basic versions of PSO and BFO.

Keywords: *Permanent Magnet Synchronous Motor; Fractional order Proportional Integral (FO-PI) controller; chemotaxis; PSO; Integral Time Absolute Error (ITAE).*

Introduction

With the increase in production of ore and coal, more and more mining dump trucks are required as they carry over 70 percent of ore and coal in surface mine. The transportation cost only accounts for over 50 percent of all costs. Hence heavy-duty, high-efficiency and energy-saving mining dump truck is the key transportation equipment in the surface mine or quarry. Off-highway mining dump trucks are divided into two groups based on transmission system – mechanical drive truck and electric drive truck. The latter performs better, runs faster, hauls more and operates safely than the former. Due to the arrangement of structure and electromagnetic interference, the majority of electric drive trucks are droven by AC drive system instead of DC drive system i.e., propulsion is delivered to the wheels by two parallel AC wheel motors on rear axle.

AC drive system in mining dump truck is designed robustly to withstand a higher vibration and altitude spectrum and to offer a temperature of range from -40 to 60 degrees. The air or liquid cooled AC drive system also helps users in achieving high productivity while lowering operation costs through its exceptional performance, efficiency and reliability. This AC drive system incorporates alternator, wheel motor with planetary gear reduction unit, auxiliary electric subsystems and control devices. IGBT inverter converts the power from the alternator working in conjunction with a diesel engine, first to DC and then to variable frequency AC for the wheel motors. According to recent studies, with the advancement of control theories, power electronics, microelectronics in connection with new motor design and magnetic materials since 1980's electrical (A.C) drives are making tremendous impact in the area of variable speed control systems [1-2]. Permanent Magnet Synchronous Motor are common AC motors which are being used in industrial environment. PMSM with high efficiency, variable speed, high torque and small bulk out performs induction motor which has poor overload capability and hard controllability. Permanent Magnet Synchronous Motors with high energy permanent magnet materials like "Neodymium Iron Boron" ("Nd-Fe-B") provide fast dynamics and computability with the applications if controlled properly. Electrically excited field windings are replaced by Permanent Magnets (PM) because of their advantages which include elimination of brushes, slip-rings, rotor copper losses; which yields high efficiency [3].

PMSM's are commonly used for applications like actuators, machine tools and robotics. This is due to some of its advantages such as high power-density, efficiency, reduced volume and weight, low noise and robustness [3]. Now-a-days vector control technique has made it possible to apply the PMSM's in high-performance industrial applications where only D.C motor drives were previously available. In order to make the most of a motor performance a very effective control system is needed. Although many possible solutions are available eg., non-linear, adaptive, intelligent control [4-5] the market of electrical drives doesn't justify the expense needed to implement such sophisticated solution in industrial drives and Proportional-Integral (PI) based control system scheme still remains the more widely adopted solution. Such a propensity is supported by a fact that, although simple, a PI based control allows achieving of very high performances when optimally designed [6]. PI controllers have been widely used for decades in industries for process control applications. The reason for their wide popularity lies in the simplicity of design and performances including low percent overshoot, less maintenance cost [6].

[1] *Ing. Rajasekhar Anguluri*, National Institute of Technology-Warangal, INDIA, rajasekhar.anguluri@ieee.org

[2] *Prof. Abraham Ajith, PhD.*, VSB - Technical University of Ostrava, Ostrava - Poruba, Czech Republic and Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, WA, USA, ajith.abraham@ieee.org

[3] *Prof. Snasel Vaclav, PhD.*, VSB - Technical University of Ostrava,Ostrava - Poruba, Czech Republic, vaclav.snasel@vsb.cz

(Review and revised version 15. 09. 2011)

An elegant way of enhancing the performance of PI controllers is to use *fractional-order controllers*. Dynamic systems based on fractional order calculus have been a subject of extensive research in recent years since the proposition of concept of the *fractional-order controllers* and the demonstration of their effectiveness in actuating desired fractional order system responses by Podlubny [7]. Fractional Order Proportional Integral (FOPI) controller is one of a convenient fractional order structure that has been employed for control purposes. In an FOPI controller (in general PI^{λ}) I-operations are usually fractional order; therefore besides setting K_P, K_I we have another parameter i.e., order of fractional integration. If $\lambda = 1$ it is an integer PI controller and if $\lambda = 0$ it is proportional gain. Finding appropriate set of values for these three parameters to achieve optimum performance of PMSM drive in three dimensional hyper-space calls for real parameter optimization. Our tuning method focuses on minimizing the Integral Time Absolute Error (ITAE) criterion. This is done by using Nature inspired heuristic methods for optimal design of the controller.

Natural selection tends to eliminate animals with poor foraging strategies and favor the propagation of genes of those animals that have successful foraging strategies since they are more likely to enjoy reproductive success. After many generations poor foraging strategies are either eliminated or shaped into good ones. Based on the researches on the foraging behavior of *E-coli* bacteria K.M. Passino proposed a new Evolutionary computation technique known as Bacterial Foraging Optimization Algorithm (BFOA) [8], briefly explained in the following sections. Until date BFOA has found its successful implementation real world problems such as PI/PID controller design, stock market prediction, and power systems. However, during the process of chemotaxis, the BFOA depends on random search directions which may lead to delay in reaching global solution.

In order to speed the convergence of Bacterial Foraging Optimization W. Karoni had proposed an improved BFOA namely BF-PSO [9]. The BF-PSO algorithm borrowed the ideas of velocity updating from particle swarm optimization (PSO), the search directions specified by the tumble of bacteria are oriented by the individual best location and global best locations concurrently. The proposed method by W.Korani had shown remarkable results in PID controller tuning. To reduce the time of optimization and to accelerate the convergence speed of group of bacteria near global optima for this BFO-PSO we propose a new hybrid algorithm "ABF-PSO" in which the chemotactic step had been made adaptive. The proposed method is applied in parameter optimization of FOPI speed controller for a PMSM drive. We also compared the obtained results with that of traditional methods and state of art optimization techniques.

Permanent Magnet Synchronous Motor Drive

In general PMSM with approximately sinusoidal back electromotive force (i.e., back EMF) can be broadly categorized into two types 1) Interior (or buried) Permanent Magnet Synchronous Motors (IPMSM) 2) Surface mounted Permanent Magnet Synchronous Motors (SPMSM). In this paper, we considered the SPMSM. In this type of motor,

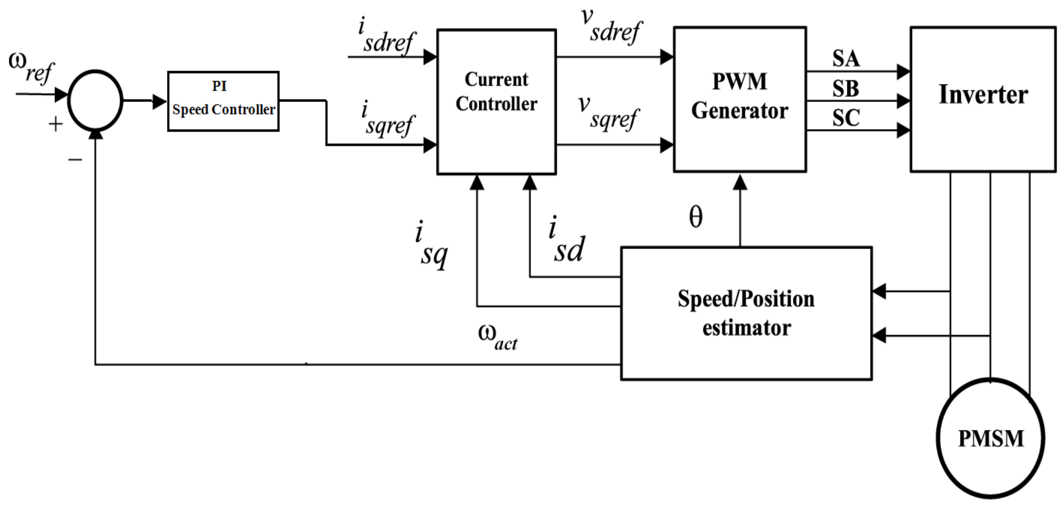

Fig. 1: Block diagram of PMSM drive.

the magnets are mounted on the surface of the motor. Since the incremental permeability of these magnets is between 1.02-1.20 relative to external fields, the magnets have high reluctance and accordingly the SPMSM can be considered to have large and effective uniform air-gap. This property makes the saliency effect negligible. Thus quadrature axis synchronous inductance of SPMSM is equal to its direct axis inductance. As a result magnetic torque can only be produced by SPMSM, which arises from the interaction of magnet flux and quadrature axis current. The stator carries a three-phase winding which produces a near sinusoidal distribution of magneto motive force based on the value of stator current. They have the same role as the field winding in a synchronous machine except their magnetic field is constant and there is no control on it [10].

Mathematical model of PMSM

The PMSM drive model consists of a Pulse Width Modulation (PWM) inverter, a PWM generator, a current controller followed by speed controller and is also embedded with speed/position estimator. The schematic representations of these components are shown in Fig. 1. The PMSM drive receives power from three-phase AC supply and runs mechanical load at desired speed. The developed model of the drive system is used for design in current and speed controllers. The mathematical model of PMSM in *d-q* synchronously rotating frame of reference can be obtained from synchronous machine model. The PMSM can be represented by the set of following nonlinear [11] differential equations.

$$v_{sd} = r_s i_{sd} + p\lambda_{sd} - \omega_e \lambda_{sq} \tag{1}$$

$$v_{sq} = r_s i_{sq} + p\lambda_{sq} + \omega_e \lambda_{sd} \tag{2}$$

$$\lambda_{sd} = L_d i_{sd} + \lambda_m \tag{3}$$

$$\lambda_{sq} = L_q i_{sq} \tag{4}$$

$$T_e = \frac{3}{2}\frac{P}{2}[\lambda_m i_{sq} + (L_d - L_q) i_{sd} i_{sq}] \tag{5}$$

$$T_e = J\frac{2}{p}\frac{d\omega_e}{dt} + B\frac{2}{P}\omega_e + T_l \tag{6}$$

$$\omega_e = P\omega_r/2 \tag{7}$$

$$p\theta_r = \frac{2}{P}\omega_e \tag{8}$$

where v_{sq}, v_{sd}, i_{sq}, i_{sd} are *d-q* axis voltages and currents respectively. L_d, L_q are *d-q* axis inductances; λ_{sq}, λ_{sd} are *d-q* axis flux linkages; and ω_e, r_s are electrical speed of motor, and stator resistance respectively. λ_m is the constant flux linkage due to rotor permanent magnet; T_e is the electromagnetic torque; T_l is the load torque; P represents number of poles; p is the differential operator; B is the damping coefficient; θ_r is the rotor position; ω_r is the rotor speed; and J is the moment of inertia.

For constant flux operation when $i_{sd(ref)}$ equals zero [11], in vector-control technique or Field Oriented Control (FOC) the equations of PMSM are modified as

$$pi_{sd} = (v_{sd} - r_s i_{sd} + \omega_e L_q i_{sq})/L_d \tag{9}$$

$$pi_{sq} = (v_{sq} - r_s i_{sq} - \omega_e L_d i_{sd} - \omega_e \lambda_m)/L_q \tag{10}$$

$$\frac{d\omega_e}{dt} = \frac{1}{J}[\frac{P}{2}(T_e - T_l) - B\omega_e] \tag{11}$$

$$T_e = \frac{3}{2}\frac{P}{2}[\lambda_m i_{sq}] = K_t i_{sq} \tag{12}$$

Problem Formulation

Fractional calculus: a brief-overview

Fractional Calculus (FC) is the branch of Mathematics, having 300 years of history. Recently this theory was applied to many fields of science and engineering [12, 13, 14, 15, 16]. FC is a generalization of ordinary differential calculus which considers the possibility of taking real number power of differential and integration operator. There are many ways to describe fractional-order integrals and derivatives. The main concept of FC lies in developing a functioning operator D which is known as differ-integrator operator, according to Riemann-Liouville definition it is mathematically defined as

$${}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(m-\alpha)}\left(\frac{d}{dt}\right)^m \int_a^t \frac{f(\tau)}{(t-\tau)^{1-(m-\alpha)}} d\tau \tag{13}$$

Where $\Gamma(\cdot)$ is the Euler's `gamma` function in the range of $m-1<\alpha<m$. Another alternative method of defining this D is by using the concept of fractional differentiation by Grünwald-Letnikov, which is given by

$$_aD_t^{\alpha} f(t) = \lim_{h\to 0} \frac{1}{\Gamma(\alpha)h^{\alpha}} \sum_{k=0}^{(t-a)/h} \frac{\Gamma(\alpha+k)}{\Gamma(k+1)} f(t-kh). \tag{14}$$

The generalized form of fractional order PI controller is the PI^{λ} controller, which involves an fractional integrator of order λ (can be any real number). The controller signal $u(t)$ can then expressed in time domain as

$$u(t) = K_P e(t) + K_I D^{-\lambda} e(t) \tag{15}$$

In the frequency domain the transfer function of this controller is given as

$$G_c(s) = K_P + \frac{K_I}{s^{\lambda}} \tag{16}$$

The interpretation of s^{λ} is that, on a semi-log plane, there is a line having slope of -20λdB/dec.

Where $\lambda=+1$ implies normal PI controller and $\lambda=0$ implies proportional gain. All these different forms of PI controller are the special cases of fractional PI^{λ} controller. It is very interesting to note that FOPI controller generalizes the integer-order PI controller and expands it from point to plane. This will add more flexibility to controller design and the controlling of real word process is more precise and accurate.

Digital Realization of FOPI Controller

The usual way of using transfer functions including fractional orders of s is to approximate it to integer order transfer functions. To actually approximate a fractional transfer function, an integer transfer function would have to involve an infinite number of poles and zeros (which is very difficult). But it is possible to obtain logical approximations with a finite number of zeros and poles.

Out of many real approximation methods, one of the well-known approximations is by Oustaloup method [17] which uses recursive distribution of poles and zeros. According to Oustaloup the transfer function is given by

$$s^{\alpha} \approx k \prod_{n=1}^{N} \frac{1+(s/\omega_{z,n})}{1+(s/\omega_{p,n})} \tag{17}$$

The approximation is valid in the frequency range of $[\omega_l, \omega_h]$. The value of poles and zeros (N) is fixed, The required performance of the approximation mainly depends on: truncated values causing simpler approximations, but may lead to ripples in both phase and gain regions. These ripples may be functionally removed by increasing poles and zeros count, ultimately leading to heavier computation. Required frequencies of poles and zeros in Eqn (17) is given according to

$$\omega_{z,1} = \omega_l \sqrt{\eta} \tag{18}$$

$$\omega_{p,n} = \omega_{z,n}\varepsilon,\ n = 1,\dots,N \tag{19}$$

$$\omega_{z,n+1} = \omega_{p,n}\eta,\ n = 1,\dots,N-1 \tag{20}$$

$$\varepsilon = (\omega_h/\omega_l)^{v/N} \tag{21}$$

$$\eta = (\omega_h/\omega_l)^{(1-v)/N} \tag{22}$$

In the case of $\alpha<0$, this can be handled by reversing Eqn (17).

Formulation of the objective function

There are various performance criteria for design of controllers, of which some are integral absolute error (*IAE*), integral squared error (*ISE*). The main drawback of using these *ISE* and *IAE* criteria is a dynamic response with a relatively less overshoot but a heavy settling time because they will calculate the errors uniformly over time [18]. If integral-time-weighted-squared-error (*ITSE*) is used the drawback of prior mentioned methods can be eliminated, but it cannot ensure to have a desirable stability (and is also computationally complex). So, we employed Integral-time-weighted-absolute-error (*ITAE*) which has an advantage of producing lesser oscillations and overshoot along with less settling time, which is Mathematically defined as

$$ITAE = \int_0^\infty t(|\omega_{ref} - \omega_{act}|) = \int_0^\infty t(|e(t)|) \tag{23}$$

The performance of the drive depends on the fractional controller parameters values which indeed depend on the objective function to be minimized. So, to get a optimal set of parametric values for K_P, K_I, and λ to meet the user defined specifications for a given process call for real parameter optimization in three-dimensional hyperspace.

Bacterial Foraging Optimization Algorithm

Bacterial Foraging optimization is based on foraging behavior of Escherichia coli (*E.coli*) bacteria present in the human intestine and been already implemented to real world problems [19]. In this foraging theory, the objective of the animal is to search for and obtain nutrients in a fashion that energy intake per unit time (E/T) is minimized [8]. A group of bacteria move in search of food and away from noxious elements known as Foraging. BFO algorithm draws its inspiration from this foraging behavior. Bacteria have a tendency to gather to the nutrient-rich areas by activity called Chemotaxis. Its movement and behavior is characterized by the spinning flagella which acts as a Biological motor and helps bacteria to swim.

E.coli bacteria has $8-10$ flagella placed randomly on its body with a speed of $100-200$ rps. An *E.coli* bacteria alternates through running and tumbling. Running speed is $10-20\mu/s$ and they cannot swim straight. The flagella can rotate either clockwise or counterclockwise. When all the flagella rotate counterclockwise, they form a compact, helically propelling the cell along a trajectory, which is called "run". When the flagella rotate in clockwise direction they enable the bacterium to move in different directions and cause bacteria to "tumble". The bacterial foraging process consists mainly of four sequential mechanisms namely chemotaxis, swarming and reproduction and elimination-dispersal. Brief overviews of these are given in the following section.

1. *Chemotaxis:-* An *E.coli* bacterium can move in two different ways: it can run (swim for a period of time) or tumble and alternates between these movements throughout its travel in search of food. In BFO, a unit walk with random direction represents a "tumble" and a unit walk with the same direction in the last step indicates "run". In the computational chemotaxis, the movement of i^{th} bacterium after one step can be represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\varphi(j) \tag{24}$$

Here $\theta^i(j,k,l)$ denotes the location of i^{th} bacterium at j^{th} chemotactic k^{th} reproductive and l^{th} elimination and dispersal step. $C(i)$ is the length of unit walk, which is constant in basic BFO and $\varphi(j)$ is the direction angle of the j^{th} step. When the bacterium is in run mode $\varphi(j)$ is same as $\varphi(j-1)$, otherwise $\varphi(j)$ is a random angle directed within a range of [0,2Π]. If the cost at $\theta^i(j+1,k,l)$ is better than the cost at $\theta^i(j,k,l)$, then the bacterium takes another step of size $C(i)$ in that direction otherwise it is allowed to tumble. This process is repeated until the number of steps taken is greater than the number of iterations in chemotactic loop, N_c

2. *Swarming:-* A bacterium in times of stresses releases attractants to signal the bacteria to swarm together. Each bacterium also releases repellant to signal the others to be at a minimum distance from it. Thus all of them will have a cell to cell attraction via attractant and cell to cell repulsion via repellant. The cell to cell signaling in *E.coli* swarm may be mathematically represented as

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l) =$$

$$\sum_{i=1}^{S}[-d_{attractant} exp(-\delta_{attractant} \sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] +$$

$$\sum_{i=1}^{S}[h_{repellant} exp(-\delta_{repellant} \sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] \tag{25}$$

Here $J_{cc}(\theta, P(j,k,l))$ represents objective function value to be added to actual objective function, S is the total number of bacteria, p is the number of parameters to be optimized and $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^T$ is a point in p-dimensional search domain. $d_{attractant}$ is the depth of attractant released by the cell and $\delta_{attractant}$ is the measure of width of the attractant signal. $h_{repellant} = d_{attractant}$ is the height of repellant effect magnitude, $\delta_{repellant}$ is a measure of width of repellant. These coefficients are to be taken judiciously.

3. *Reproduction:-* After the completion of all N_c chemotactic steps a reproduction step takes place. Fitness value of the bacteria is stored in ascending order. The lower half of bacteria having a higher fitness die and remaining $S_r = S/2$ bacteria are allowed to split into two identical one. Thus the population after reproduction remains constant.

4. *Elimination and Dispersal:-* There is a probability that bacteria may be stuck around the initial or local optima positions, it is required to diversify the bacteria either gradually or suddenly so that the possibility of being trapped in to local minima is eliminated and global optima is obtained. The dispersion operation takes place after a certain number of reproduction processes. A bacterium is chosen, according to a present probability p_{ed}, to be dispersed and moved to another position within the environment. This may disturb optimization process but prevent the local minima trapping.

Bacterial foraging oriented with particle swarm optimization (BF-PSO)

BF-PSO algorithm combines both BFO and PSO. The aim is to make PSO ability to exchange social information and BF ability in finding new solution by elimination and dispersal, a unit length direction of tumble behavior is randomly generated. Random direction may lead to delay in reaching the global solution. In "BF-PSO" algorithm the unit length random direction of tumble behavior can be decided by the global best position and the best position of each bacterium. During the chemotaxis loop tumble direction is updated by:

$$\varphi(j+1) = \omega * \varphi(j) + C_1 * rand * (pbest - pcurrent) + C_2 * rand * (gbest - pcurrent) \tag{26}$$

Where *pbest* is the best position of each bacterium and *gbest* is the global best bacterium. The brief pseudo-code of BF-PSO has been provided below. Algorithm to find optimal parameters using BFO for the objective function ITAE is described below

[Step 1] Initialize the parameters $p, S, N_c, N_s, N_{re}, N_{ed}, p_{ed}, C(i)(i = 1,2,3,\dots,S), \theta^i$
where

p –Dimension of the search space;

S –Number of bacteria in the population;

N_s –Swimming length after which tumbling of bacteria will be undertaken in chemotactic loop;

N_c –The number of iterations to be undertaken in chemotactic loop, always $N_c > N_s$;

N_{re} –Maximum no. of reproduction steps;

N_{ed} –the maximum no. of Elimination and dispersal events to be imposed over bacteria;

p_{ed} –Probability with which elimination and dispersal will continue;

θ^i –Location of the i^{th} $(i = 1,2,3,\dots,S)$ bacterium;

$C(i)$ –Step size of the i^{th} bacterium taken in random direction, specified by tumble. Generate a random vector $\varphi(j)$ in the range $[-11]$

C_1, C_2, ω: PSO parameters

[Step 2] Elimination and dispersal loop: $l = l + 1$
[Step 3] Reproduction loop: $k = k + 1$
[Step 4] Chemotaxis loop: $j = j + 1$

[substep 4.1] For $i = 1,2,3,\dots,S$ take a chemotactic step for bacterium i as follows

[substep 4.2] Compute fitness function, $ITAE(i, j, k, l)$.

[substep 4.3] $ITAE_{last} = ITAE(i, j, k, l)$ save this value, since we may get better value via a run.

[substep 4.4] Tumble: Let,

$$\varphi(j+1) = \omega * \varphi(j) + C_1 * rand * (pbest - pcurrent) + C_2 * rand * (gbest - pcurrent)$$

[substep 4.5] Move: Let,

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\varphi(j)$$

This results in a step of size C(i) in the direction of the tumble for bacterium *i*

[substep 4.6] Compute $ITAE(i, j+1,k,l)$

[substep 4.7] Swim:

(i) Let $m = 0$ (counter for swim length)

(ii) while $m < N_s$ (if the bacteria have not climbed too long)

- Let $m = m+1$
- If $ITAE(i, j+1,k,l) < ITAE_{last}$(if doing better), Let $ITAE_{last} = ITAE(i, j+1,k,l)$ and let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\varphi(j)$$

use this $\theta^i(j+1,k,l)$ to compute new cost function $ITAE(i, j+1,k,l)$

- else, let $m = N_s$. This is the end of while statement

[substep 4.8] go to next bacterium $(i+1)$ if $i \neq S$ (i.e., go to [substep 4.2]) to process next bacterium

[Step 5] If $j < N_c$, go to **[Step 4]**. In this case, continue chemotaxis Since the life of the bacteria is not over.
[Step 6] Reproduction:

[substep 6.1] For the given *k* and *l* and for each $i = 1,2,3,...,S$ let

$$ITAE^i_{health} = \sum_{j=1}^{N_c+1} ITAE(i, j,k,l)$$

be the health of the bacterium *i* (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria in order of ascending cost $ITAE_{health}$ (higher cost means lower health).

[substep 6.2] The $S_r = S/2$ bacteria with the highest $ITAE_{health}$ values die and remaining S_r with the best values split and this process is performed by the copies that are made are placed at same location as their parent.

[Step 7] If $k < N_{re}$, go to the **[Step 3]**. Since in this case the specified reproduction steps are not reached, start the next generation of the chemotactic loop.
[Step 8] Elimination-dispersal: For $i = 1,2,...,S$ with the probability p_{ed}, eliminate and disperse each bacterium, which results in keeping number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If $l < N_{ed}$ then go to **[Step 2]**, otherwise end;

Adaptive bacterial foraging oriented particle swarm optimization (ABF-PSO)

To enhance the performance of BF-PSO we had proposed Adaptive Bacterial Foraging Oriented Particle Swarm Optimization (ABF-PSO). Chemotaxis is a foraging strategy that implements a type of local optimization where the bacteria try to climb up the nutrient concentration, avoid noxious substance and search for ways out of neutral media. A chemotactic step size varying as the function of the current fitness value is expected to provide better convergence behavior as compared to a fixed step size. A simple adaption scheme for the step size for i_{th} bacterium given in following equation is employed to get better optimal controller parameters for PMSM drive.

$$C(i) = \frac{|j^i(\theta)|}{|j^i(\theta) + \psi|} = \frac{1}{1 + \frac{\psi}{j^i(\theta)}} \tag{27}$$

where ψ is positive constant.

$j^i(\theta) =$ cost function of the i_{th} bacterium.
$C(i) =$ variable run (step) length of i_{th} bacterium.

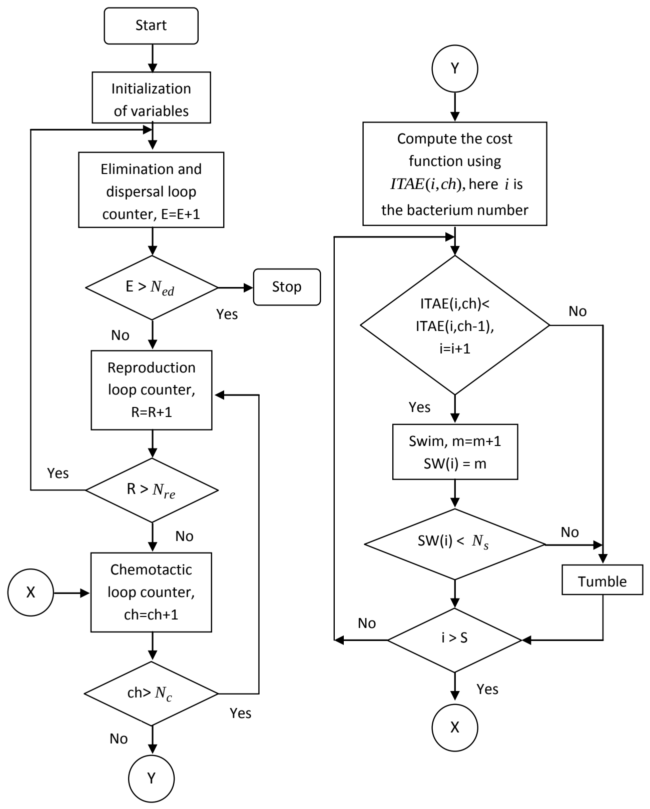
Start
Initialization of variables
Elimination and dispersal loop counter, E=E+1
E > N_{ed}
Yes
Stop
No
Reproduction loop counter, R=R+1
Yes
R > N_{re}
No
X
Chemotactic loop counter, ch=ch+1
ch> N_c
Yes
No
Y
Y
Compute the cost function using $ITAE(i,ch)$, here i is the bacterium number
ITAE(i,ch)< ITAE(i,ch-1), i=i+1
No
Yes
Swim, m=m+1 SW(i) = m
SW(i) < N_s
No
Tumble
No
i > S
Yes
X

Fig. 2: Flow chart of ABF-PSO algorithm.

If $j^i(\theta)$ tends to zero then $C(i) \longrightarrow 0$ and when $j^i(\theta) \longrightarrow large, C(i) \longrightarrow 1$. This implies that the bacterium which is in the vicinity of noxious substance associates with higher cost function. Hence it takes larger steps to migrate to a place with higher nutrient concentration. Use of Eqn (27) in Eqn (24) is expected to give improved convergence performance compared to fixed step size due to the above phenomenon.

[Step 1] Same as that of BF-PSO based optimization
[Step 2-3] Same as that of BF-PSO, but only difference is that while updating location in Eqn (24) (and also in swim) the adaptive run length unit, $C(i)$ defined in Eqn (27) is used instead of fixed run length unit.
[Step 5-8] Same as that of BF-PSO based technique.

Experimental Results

Settings

This section describes about the parametric set up of the PMSM drive and also the algorithmic parameters. Design Specifications of Drive are

Tab. 1: Parameter Setting of PMSM Drive

Variable	Actual implication	Value	Units
r_s	Stator resistance	2.0	Ω
L_{sd}	*d-axis* inductance	2.419	mH
L_{sq}	*q-axis* inductance	2.419	mH
J	Moment of inertia	0.00344638	kg-m^2
λ_m	Magnet mutual flux	0.27645	V/rad/sec
B	Damping coefficient	0.0027715	Nm/rad/sec
P	Number of pole	4	–

Simulation is done for a time $T = 1sec$ under a load torque 5 Nm with a reference speed of 1300 rpm. The range for K_P is taken from 0 to 1, for that of K_I is 0 to 10 and $\lambda$is taken between 0 and 1. Various parameters employed in the simulation study for BFO, PSO, ABF-PSO are given below

$S = 10$, $N_c = 5$, $N_s = 4$, $N_{re} = 2$, $N_{ed} = 2$, $P_{ed} = 0.25$, $C(i) = 0.075$, $\psi = 180$, $C_1 = C_2 = 2.05$, $\omega = 0.9$. Number of particles fiexd for PSO are 25. As the optimization process cannot be obtained in a single iteration, we used a total of 100 iterations for each algorithm for comparisons to be fare enough.

Step responses of FOPI controller

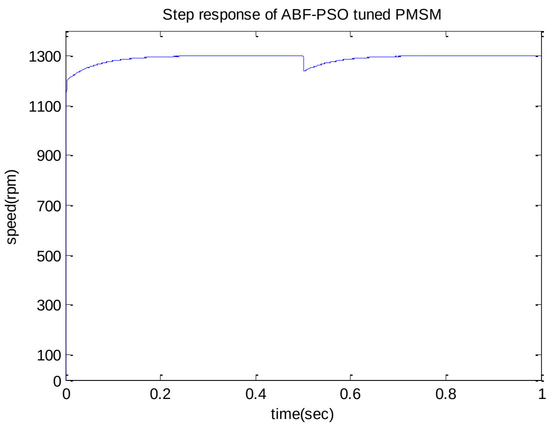

Fig. 3: Speed response of ABF-PSO tuned FOPI controller for PMSM drive.

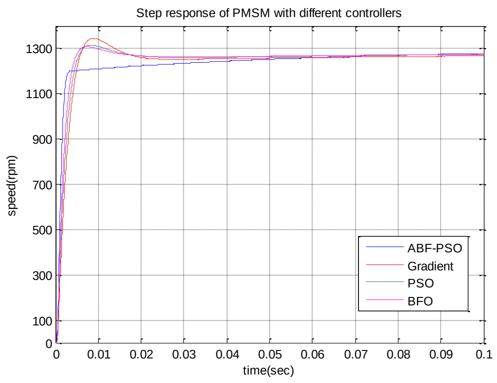

Fig. 4: Speed responses of PMSM using FOPI controller before load.

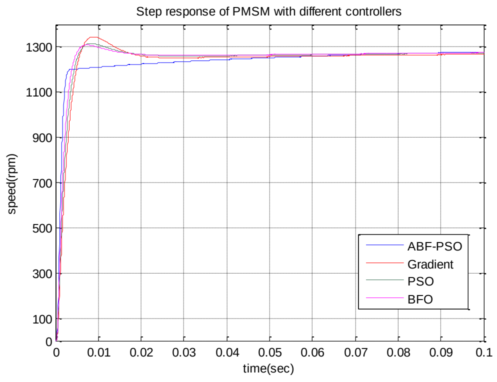

Fig. 5: Speed response of PMSM using FOPI controller after load.

Tab. 2: Comparisons of step response of FO-PI controller using different methods

Method	K_P	K_I	λ	$po(\%)$	t_r (sec)	t_s (sec)	e_{ss}
Gradient	0.2541	4.5352	0.5	3.5069	0.0042	0.6384	1.2517
PSO	0.3248	6.1264	0.5	1.2274	0.0034	0.5704	0.9087
BFO	0.4375	7.2467	0.7	0.7036	0.0025	0.5589	0.8479
ABF-PSO	0.5851	9.9531	0.9	0.0065	0.0019	0.5575	0.00412

Discussions

From the illustrated graphical and empirical results, the proposed method outperformed all the other considered in almost all the performance indices and also percentage overshoot is greatly reduced. Although the settling time remained to be same for all the methods settling time error has also improved to great extent

Conclusion

In this paper, a novel ABF-PSO algorithm is used for tuning a FOPI Speed controller for PMSM in electric drive mining truck. From the simulations and empirical results, it is easy to conclude that the proposed method performed well when compared to that of PSO, BFO and the gradient descent method. The design specifications like percent overshoot, steady state error are very much improved in ABF-PSO tuned FOPI controller. Our further research will be focused on implementing FOPI controller for the sensor less PMSM drive.

References

[1] Bose, B.K., 1993. Power electronics and motion control-Technology status and recent trends, IEEE Trans. Ind. Applications, vol. 29, pp. 902-909.

[2] Lipo, T.A., 1988. Recent progress in the development of solid state ac motor drives, IEEE Trans. Power Electron., vol. 3, pp.105-117.

[3] Vas P., 1998 Sensorless Vector and Direct Torque Control, 1st Edition, Oxford University Press.

[4] Slotine, J.J., W, Li., 1991. Applied Nonlinear Control. Englewood Cliffs, NJ: Pearson Education.

[5] Jain, L.C., De Silva, C.W., and Jain, L.C., 1998. Intelligent Adaptive Control: Industrial Applications. Boca Raton, FL: CRC.

[6] Astrom, K.J., Hugglund, T., 2001. The Future of PID Control. Control Eng. Pract., Vol. 9, pp. 1163-1175.

[7] Podlubny, I., 1999. Fractional-order systems and $PI^{\lambda}D^{\mu}$ controllers, IEEE Trans on Automatic Control, Vol. 44, no. 1, pp. 208-213.

[8] Passino, K.M., 2002. Bio mimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine, pp. 52-67.

[9] Karoni, W., 2008. Bacterial foraging oriented by particle swarm optimization strategy for PID tuning. In GECCO 2008: Proceedings of the Genetic and Evolutionary computation conf, pp. 1823-1826, ACM.

[10] D. Karaboga, B. Basturk, “A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) algorithm”. Journal of Global Optimization, Springer Netherlands, 39, 459-471, 2007.

[11] Song Chi, M.S.E.E., 2001. Position-Sensorless Control of Permanent Magnet Synchronous Machine Over Wide Range Speed, PH.D Thesis, Ohio State University.

[12] Pillay, P., and Krishnan, R., 1989. Modeling, Simulation, and Analysis of Permanent-Magnet Motor Drives, Part I: The Permanent-Magnet Synchronous Motor Drive. IEEE Trans. on Ind Applications, 25(2), pp. 265-273.

[13] Zamani, M., Ghartemani, M.K., Sadati, N., and Parniani, M., 2009. Design of a fractional order PID controller for an AVR using particle swarm optimization, Control Eng. Pract., (17), pp.1380-138.

[14] Oldham, K.B., and Spanier, J., 1974. The Fractional Calculus. Academic Press, NewYork.

[15] Petras, I., 1999. The fractional order controllers: methods for their synthesis and application. Journal of Electrical Engineering, vol. 50, no. 9–10, pp. 284–288.

[16] Chen, Y.Q., Xue, D., and Dou, H., 2004. Fractional calculus and biomimetic control. In Proc. of the First IEEE Int. Conf. on Robotics and Biomimetics (RoBio04), pages robio2004 347, Shengyang, China, August 2004. IEEE.

[17] Oustaloup, A., Levron, F., Mathieu, B., and Nano, F.M., 2000. Frequency band complex noninteger differentiator: characterization and synthesis. IEEE Trans. Circuits Syst. I, vol. 47, 2000, pp. 25-39.

[18] Krohling, R.A., and Rey, J.P., 2001. Design of Optimal Disturbance Rejection PID Controller Using Genetic Algorithms, IEEE Trans. on Evolutionary Computation, 5, pp. 78-82.

[19] Dasgupta, S., Das, S., Abraham, A., Biswas, A., 2009. Adaptive Computational Chemotaxis in Bacterial Forgaing Optimization: An Analysis. IEEE Trans on Evolutionary Comp., vol. 13, pp. 919-941.