Adaptive Control of Nonlinear Systems Using Bacterial Foraging Algorithm

Bharat Bhushan[1] and Madhusudan Singh[2]

***Abstract*—In this paper bacterial foraging algorithm (BFA) have been implemented for indirect adaptive control of two nonlinear systems. The nonlinear systems considered in present analysis are liquid level control of surge tank and armature controlled DC motor speed control system. Simulation has been carried out in MATLAB. Using bacterial foraging algorithm plant nonlinearities component alpha and beta have been estimated and compared with actual plant nonlinearities for both the nonlinear systems. Liquid level estimated by BFA adaptive controller and actual liquid level has been compared in the liquid level nonlinear control system. Angular speed estimated by BFA adaptive controller and actual trajectory has been compared for DC motor speed control system. Fitness for the best member in the population of bacteria has been observed for both the nonlinear systems. In DC motor speed tracking obtained using indirect BFA adaptive control have been obtained and compared with reference one. Also liquid level set by indirect BFA adaptive controller have been obtained and compared with reference trajectory of level. Error plots show that error between desired and actual trajectory reduces after ten seconds of system operation in both nonlinear systems. Simulations studies have been also carried out through genetic algorithm (GA) for both nonlinear systems. A comparison of BFA results with GA shows that BFA based indirect adaptive controller are more effective in tracking the desired trajectory in both the nonlinear systems.**

***Index Terms*—bacterial foraging optimization algorithm, indirect adaptive control, liquid level control and DC motor.**

I. Introduction

To tackle several complex search problems of real world, scientists have been looking into the nature for years- both as model and as metaphor- for inspiration. Optimization is at the heart of many natural processes like group behaviour of social insects, birds and the foraging strategy of other microbial creatures. Natural selection tends to eliminate species with poor foraging strategies and favour the propagation of genes of species with successful foraging behaviour. Since a foraging organism or animal takes necessary action to maximize the energy utilized per unit time spent for foraging, considering all the constraints presented by its own physiology such as sensing and cognitive capabilities, environment, natural foraging strategy can lead to optimization and essentially this idea can be applied to real world optimization problems.

Many bio-inspired computational methodologies such as Genetic Algorithm (GA) [2], Particle Swarm Optimization (PSO) [6], Ant Colony [5] and Artificial Fish Swarm Algorithm (AFSA) [3] have been intensively studied and applied to various optimization problems. In recent years, a new and rapidly growing subject- Bacterial Foraging Algorithm (BFA) [1] proposed by Prof. Passion in 2002 is a novel modern search algorithm based on the behaviour of E. coli bacteria. E. coli is a common type of bacteria with a diameter of 1um and a length of about 2 um. During the lifetime of E. coli, it shows the behaviour of chemotactic action, which makes the E. coli swimming up a nutrient gradient or out of noxious substance. Assuming the foraging time T, one E. coli obtains the energy with the value of E. Then E/T (foraging rate) depicts the chemo-tactic actions which models the foraging of E. coli as an optimization process. When an animal seeks to maximize the energy obtained in per unit time spent foraging. The E. coli bacteria communicate each other and at the same time completes for food. Beside the chemo-tactic action and information communication, other complex stages such as reproduction, elimination and dispersal stages are included in E. coli foraging behaviours. BFA has been tested on many unconstrained global optimization functions like Sphere function, Rosenbrock function, Rastrigin function, Ackley function and Griewank function etc. A Fast Bacterial Swarming Algorithm [7] was tested on High-dimensional Function Optimization. Self-adaptive Bacterial Foraging Optimization [8], employing the adaptive search strategy to significantly improve the performance of the original algorithm was achieved by enabling BFO to adjust the run-lengthunit parameter dynamically during evolution to balance the exploration/exploitation tradeoff. A variable denoting the overall best value [9] is incorporated to guide the bacterial swarm to move to the global optima and replace the role of interaction behavior between bacteria in classic BFO which is complicated and time-consuming. Micro-bacterial foraging algorithm [10] was proposed, which evolves with a very small population compared to its classical version. To accelerate the convergence speed of the bacterial colony near global optima, two cooperative approaches have been applied to BFO [11] that resulted in a significant improvement in the performance of the original algorithm in terms of convergence speed, accuracy and robustness. BFA aiming for optimization in dynamic environments [12] called DBFA, is studied. Analysis of Reproduction Operator [13] was studied in Bacterial Foraging Optimization Algorithm. Also BFA has been applied to many kinds of real world optimization problems [15-16]. Many hybrid algorithms have been developed using BFA [4, 14] In this paper implementation of BFA for two nonlinear systems, one is liquid level control system and other one is DC motor speed control system [18] is carried out and relative improvement in system performance in

Manuscript received November 23, 2010; revised February 12, 2011.

Bharat Bhushan1 and Madhusudan Singh are with the Department of Electrical Engineering, Delhi Technological University, Delhi (email: bharatdce@yahoo.co.in[1], madhusudan_s2@rediffmail.com[2]).

comparison to those of GA is discussed in detail.

This paper is organized as follows: Section II describes the basic principle of BFOA and pseudo code of Bacterial Foraging Optimization Algorithm. Section III introduces the nonlinear sysstems. Section IV describes the indirect BFA adaptive control. The simulation results and discussions are shown in Section V. Conclusions are shown in Section VI followed by references in section VII.

II. Bacterial Foraging Optimization Algorithm

The optimization in BFA comprises the following processes: chemotaxis, swarming, reproduction, elimination and dispersal. Chemotaxis is the activity that bacteria gathering to nutrient- rich areas spontaneously. A cell- to – cell communication mechanism is established to simulate the biological behaviour of bacteria swarming. Reproduction comes from the concept of natural selection and only the bacteria best adapted to their environment tend to survive and transmit their genetic characters to succeeding generations while those less adapted tend to be eliminated. Elimination-dispersal event selects parts of the bacteria to diminish and disperse into random positions in the environment, which ensures the diversity of the species. Each of these processes has been described and finally pseudo-code of the BFOA is presented.

a) **Chemotaxis**: This process simulates the movements of an E.coli cell through swimming and tumbling via flagella. Biologically an E.coli bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$ is i-th bacterium at j-th chemo-tactic, k-th reproductive and l-th elimination dispersal step. $C(i)$ is the size of the step taken in the random direction specified by the tumble. Then in computational chemotaxis the movement of the bacterium may be represented by:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

where Δ indicates a unit length vector in the random direction.

b) **Swarming**: Interesting group behaviour has been observed for several motile species of bacteria including E.coli and S.typhimurium, where stable spatio-temporal patterns (swarms) are formed in semisolid nutrient medium. A group of E.coli cells arrange themselves in a travelling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The cells when stimulated by high level of succinate release an attractant aspartate, which helps them to aggregate into groups and thus move as concentric patterns of swarms of high bacterial density. The cell to cell, signalling in E.coli swarm may be represented with the following function.

c) **Reproduction**: After chemo-tactic steps, the fitness values for the i-th bacterium in the chemotaxis loop are accumulated and calculated by: $J^i_{healt\,h} = \sum_{j=1}^{N_C+1} J^i(j,k,l)$

where $J^i_{healt\,h}$ represents the health of the i-th bacterium. The smaller the $J^i_{healt\,h}$ is the healthier the bacterium is. To simulate the reproduction character in nature and to accelerate the swarming speed, all the bacteria are sorted according to their health values in an ascending order and each of the first bacteria splits into two bacteria. The characters including location and step length of the mother bacterium are reproduced to the children bacteria. Through this selection process the remaining unhealthier bacteria are eliminated and discarded. To simplify the algorithm, the number of the bacteria keeps constant in the whole process.

d) **Elimination- dispersal**: For the purpose of improving the global search ability, elimination-dispersal event is defined after reproductive steps. The bacteria are eliminated and dispersed to random positions in the optimization domain according to the elimination-dispersal probability. This elimination-dispersal event helps the bacterium avoid being trapped into local optima.

e) Pseudo Code of the BFOA Algorithm

Parameters:

[Step1] Initialize parameters $n, N, N_C, N_S, N_{re}, N_{ed}, P_{ed}, C(i)(i = 1,2,3,4 \ldots\ldots N)$.

Where,

n: Dimension of the search space,

N: The number of bacteria in the population,

N_C: Chemotactic steps,

N_S : Swim length,

N_{re}: The number of reproduction steps,

N_{ed}: The number of elimination-dispersal events,

P_{ed} : Elimination-dispersal with probability,

$C(i)$: The size of the step taken in the random direction specified by the tumble.

Algorithm:

[Step 2] Elimination-dispersal loop: $l = l + 1$

[Step 3] Reproduction loop: $k = k + 1$

[Step 4] Chemotaxis loop: $j = j + 1$

[1] For $i = 1,2,3,4 \ldots N$, take a chemotactic step for bacterium i as follows.

[2] Compute fitness function, $J(i,j,k,l)$. Let, $J(i,j,k,l) = J(i,j,kl) + J_{cc}(\theta^i(j,k,l), P(j,k,l))$ (i.e. add on the cell to cell attractant-repellant profile to simulate the swarming behaviour)

[3] Let $J_{last} = J(i,j,k,l)$ to save this value to find a better cost via a run.

[4] Tumble: Generate a random vector $\Delta(i) \in \Re^n$ with each element $\Delta_m(i), m = 1,2,3,4 \ldots p,$ a random number on [-1, 1].

[5] Move: Let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

this result in a step of size $C(i)$ in the direction of the tumble for bacterium i .

[6] Compute $J(i,j,k,l)$ and let $J(i,j,k,l) = J(i,j,kl) + J_{cc}\left(\theta^i(j,k,l), P(j,k,l)\right)$

[7] Swim

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

ii) While m < N_S (if have not climbed down too long).

- Let m=m+1
- If $J(i,j+1,k,l) < J_{last}$ (if doing better), let $J_{last} = J(i,k+1,k,l)$ and $\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$ and use this $\theta(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$ as in step [6].
- Else, let m= N_S . This is the end of the while statement.

[8] Go to next bacterium $(i,+1)$ if $i \neq N$ (i.e., go to [2] to process the next bacterium).

[Step 5] If $j < N_C$, go to step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.

[Step 6] Reproduction:

- For the given k and l , and for each i =1,2,3,4.... N , let

$$J^i_{healt\,h} = \sum_{j=1}^{N_C+1} J(i,j,k,l)$$

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 and chemotactic parameters $C(i)$ in order of ascending cost $J_{healt\,h}$ (higher cost means lower health).

- The S_r bacteria with the highest $J_{healt\,h}$ values die and the other S_r bacteria with the best values split (and the copies that are placed at the same location as their parent).

[Step 7] If $k < N_{re}$, go to step 2. In this case, we have not reached the number of specified reproduction steps, so we start the next generation in the chemotactic loop.

[Step 8] Elimination-dispersal: For $i = 1,2,3,4 \ldots N$, with Probability P_{ed} , eliminate and disperse each bacterium, and this keeps the 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.

[Step 9] If $l < N_{ed}$, then go to step 1; otherwise end.

III. Nonlinear Systems

a) Trajectory Control of Armature Controlled DC motor

The discrete time model of an armature controlled DC motor [17] is given by

$$w(k+1) = k1w(k) + k2w(k-1) + k3\left[sgn\left(w(k)\right)\right]w^2(k) + k4\left[sgn\left(w(k)\right)\right]w^2(k-1) + k5u(k) + k6 \quad (1)$$

$$\alpha\left(w(k)\right) = k1w(k) + k2w(k-1) + k3\left[sgn\left(w(k)\right)\right]w^2(k) + k4\left[sgn\left(w(k)\right)\right]w^2(k-1) + k6 \quad (2)$$

$$\beta\left(w(k)\right) = k5 \quad (3)$$

Now, since $w(k+1) = \alpha\left(w(k)\right) + \beta\left(w(k)\right)u(k)$

Where

$$k1 = 2L_aJ + T(R_aJ + L_aB) - T^2\frac{R_aB + K_bK_T}{K} \quad (4)$$

$$K = L_aJ + T(R_aJ + L_aB) \quad (5)$$

$$K_2 = L_a\frac{J}{K} \quad (6)$$

$$K_3 = -T(\mu L_a + \mu R_aT)/K \quad (7)$$

$$K_4 = T\mu L_a/K \quad (8)$$

$$K_5 = K_T\frac{T^2}{K} \quad (9)$$

$$K_6 = -T_FR_aT^2/K \quad (10)$$

Where R_a is armature winding resistance, L_a is armature winding inductance, w is angular velocity of motor, B is viscous friction coefficient of motor rotor with attached mechanical load, T_F frictional torque, T sampling time, J moment of inertia of the motor rotor with attached mechanical load, K_T torque constant, K_b back emf constant and μ is a constant.

b) Liquid Level Control of Surge Tank

Equation (11) shows the nonlinear model of a surge tank for liquid level control [1] , *h(t)* is liquid level (saturated), *u(t)* is the liquid level control, c and d are constants and *A(h(t))*= $|ah(t)+b|$ is the unknown cross-sectional area in which a and b are also constants. Identifier model is an affine mapping that matches plant nonlinearities.

$$\frac{dh(t)}{dt} = \frac{-d\sqrt{2gh(t)}}{A\left(h(t)\right)} + \frac{c}{A\left(h(t)\right)}u(t) \quad (11)$$

Discrete time approach has been used. We ignore the saturation at the actuator input and the fact that the liquid never goes negative, and view the dynamics as

$$h(k+1) = \left(h(k) + T\frac{-d\sqrt{19.6\,h(k)}}{|ah(k)+b|}\right) + \left(\frac{cT}{|ah(k)+b|}\right)u(k) \quad (12)$$

$$\alpha\left(h(k)\right) = \left(h(k) + T\frac{-d\sqrt{19.6\,h(k)}}{|ah(k)+b|}\right)$$

And

$$\beta(h(k)) = \left(\frac{cT}{|ah(k)+b|}\right)$$

We have $h(k+1) = \alpha(h(k)) + \beta(h(k))u(k)$ (13)

IV. Bacterial Foraging Adaptive Control

Optimization methods such as gradient algorithms and least squares methods are used to implement estimation methods, which are used to estimate models or controllers in adaptive control. Here, foraging algorithm has been used as the basis for adaptive control. In indirect adaptive control one seeks to learn a plant model during the operation of a system. Learning is viewed as foraging for good model information (i.e. information that is truthful and useful for meeting goals). An identifier model is used which is a parameterized model of the plant, and consider foraging algorithm searching in the parameter space that corresponds to finding nutrients. Multiple identifier models and social foraging (i.e. multiple models are tuned simultaneously, with foragers possibly sharing information to try to improve foraging success). The FBA search location in the parameter space , which corresponds to getting low identification errors between the model and the plant. Then, according to the sum of the squared identifier errors, at each time instant the model that is the best and uses it in a standard certainty equivalence approach to specify a controller is chosen. Each identifier model is an affine mapping to match plant nonlinearities. The identifier model parameters represent the forager's position. The cost function for each forager, which defines the nutrient profile, is defined to be the sum of squares of past identifier error values for each identifier model. For parameter adjustment, a foraging algorithm is used that is based on E. coli chemotactic behaviour. Here a plant model is tuned in order to specify the controller parameters. A set (population) of approximators is used to tune and the optimization method used to tune the set is bacterial foraging optimization algorithm. Fig.1 shows the adaptive control using BFA.

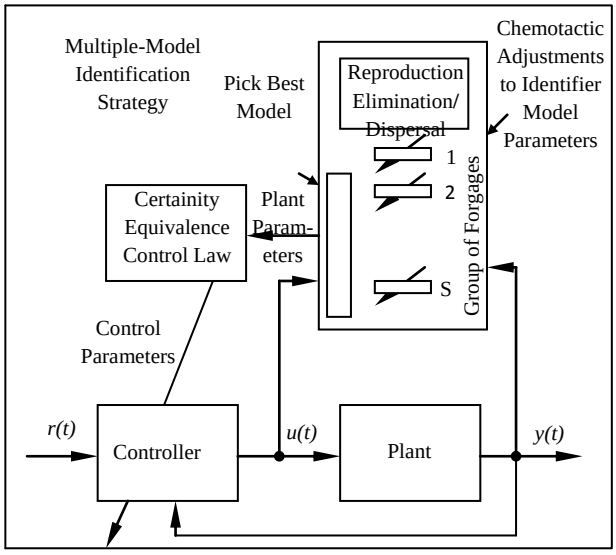

Fig.1 Adaptive control using BFA

Class of plant used is represented by

$$y(k+d) = \alpha(x(k)) + \beta(x(k))u(k) \quad (14)$$

$\alpha(x(k))$ *and* $\beta(x(k))$ are unknown smooth functions of the state $x(k)$ while $y(k+d)$ is a nonlinear function of past values of u . $\beta(x(k)$ is requiring to be bounded away from zero. $d \geq 1$ is the delay between the input and output. For

d =1, $y(k+1) = \alpha(x(k)) + \beta(x(k))u(k)$

$$y(k+d) = \alpha(x(k)) + \beta(x(k))u(k) \quad (15)$$

$$y(k+d) = \alpha_u(x(k)) + \alpha_k(k) + (\beta_u(x(k)) + \beta_k(x(k))u(k) \quad (16)$$

Functions $\alpha_u(x(k))$ and $\beta_u(x(k))$ represent the unknown nonlinear dynamics of the plant. It is these functions which require to be estimated so that a controller can be specified. $\alpha_k(k)$ and $\beta_k x(k)$ are defined to be as known parts of the plant dynamics, these can be set to zero. $\beta(x(k))$ is assumed to satisfy

$0 < \beta_0 < \beta(x(k))$ for some known $\beta_0 > 0$ for all $x(k)$.

Estimation of an unknown ideal controller: An ideal controller is given by

$$u^*(k) = \frac{-\alpha(x(k))+r(k+d)}{\beta(x(k))} \quad (17)$$

This linearizes the dynamics of equation (15) such that $y(k) \to r(k)$. substituting $u(k) = u^*(k)$ in equation (15) we obtain $y(k+d) = r(k+d)$ so that tracking of reference input have been achieved within d steps.Since $\alpha(x(k))$ *or* $\beta(x(k))$ are unknown, an estimator is developed for these plant nonlinearities and used them to form an approximation to $u^*(k)$. Using a "Certainty equivalence controller", the control input can be defined as

$$u(k) = \frac{-\tilde{\alpha}(x(k))+r(k+d)}{\tilde{\beta}(x(k))} \quad (18)$$

$\tilde{\alpha}(x(k))$ and $\tilde{\beta}(x(k))$ are estimates of $\alpha(x(k))$ and $\beta(x(k))$ respectively.

The certainty equivalence controller can be defined with the following estimates

$$\tilde{\alpha}(x(k)) = F_\alpha(x(k), \theta_\alpha(k)) \quad (19)$$

and $$\tilde{\beta}(x(k)) = F_\beta\left(x(k), \theta_\beta(k)\right) \quad (20)$$

Error $e(k) = r(k) - y(k)$ is not a linear function of the parameters. Also $e(k) = \tilde{y}(k) - y(k)$, where $\tilde{y}(k)$ is estimate of $y(k)$. A set of approximators for α and β where the i^{th} ones are denoted by

$F_\alpha(x, \theta_\alpha^i)$ *and* $F_\beta(x, \theta_\beta^i)$ for i=1,2,....,S. From a foraging perspective, θ^i is viewed as the location of the i^{th} foragers in the environment. In foraging method position of the

forager θ^i is used to minimize the fitness function $J(\theta^i)$. Let the i^{th} estimate of the output and identification error be

$$\tilde{y}^i(k+1) = F_\alpha\left(x(k), {\theta^i}_\alpha(k)\right) + F_\beta\left(x(k), {\theta^i}_\beta(k)\right)u(k)$$

and $e^i(k) = \tilde{y}^i(k) - y(k)$ for i=1,2..., S. i^{th} Individual (bacteria) at time k can be given by

$$\theta^i(k) = \left[{\theta^i}_\alpha^T(k), {\theta^i}_\beta^T(k)\right], \text{ i=1,2,..., S.} \qquad (21)$$

Fitness function can be defined as

$$J\left(\theta^i(k-1)\right) = \left(e^i(k)\right)^2$$

$$= \left(\tilde{y}^i(k) - y(k)\right)^2$$

$$= \left(F_\alpha\left(x(k-1), \theta_\alpha^i(k-1)\right) + F_\beta\left(x(k-1), \theta_\beta^i(k-1)\right)u(k-1) - y(k)\right)^2 \qquad (22)$$

which measures the size of the estimation error for the i^{th} estimate. It is required to minimize $J\left(\theta^i(k-1)\right)$.

Forager's position in one dimension is given by θ_α and in the other dimension by θ_β so that forager's position is $\theta^i = [\theta_\alpha^i, \theta_\beta^i]^T, i = 1,2..,S$. S is the population size of the bacteria. Foraging strategy is based on E.coli chemotaxis, but without swarming, elimination-dispersal, and reproduction. Here chemotactic hill-climbing strategy is used to adjust the parameters. At each time step, one foraging step is used, that means either one tumble-tumble step. Foraging occurs while the control system operates with foraging (searching) for parameters occurring at each time step. For instance, if over one time step the cost did not decrease for an individual, then there is a tumble, and by this, a random direction is generated which update the parameters (location of the forager) in that direction. If, cost is improved from the last step, then another step in the same direction taken last time is made. In such case, forager is on a run in a good direction, down the cost function.

V. Simulation Results and Discussion

Simulations have been performed in Matlab 7.4 and scratches have been developed using Duo CPU T 6400@ 2.00GHz 1.20GHz, 1.99 GB of RAM. Eighteen independent runs of BFOA were carried out. In the case of BFOA algorithm, the best suited sets of parameters are chosen after a series of hand tuning experiments. The step size $C(i,k)$ is set to be 0.05 for all bacteria for all times. The maximum number of steps along a good direction is N_S =4, and ${\theta^i}_\alpha(0)$=2, ${\theta^i}_\beta(0)$=0.5, i=1,2,...,S.

Parameter settings for BFA algorithms

- DC Motor Speed Control: $p = 2, S = 1000, N_S = 4, N_C = 100$, runlengthunit=0.05, beta0=0.05, beta1=0.5, input control -200 < $u(k)$ <200.
- Liquid Level Control: $p = 2, S = 9000, N_S = 4, N_C = 1000$, runlengthunit=0.05, beta0=0.05, beta1=0.5, input control -50 < $u(k)$ <50.

Parameter settings for GA algorithm

- DC Motor Speed Control: $p = 2, S = 1000$, probability of crossover P_c=0.9, mutation probability Pm=0.05, Initial population element ${\theta^i}_\alpha(0)$=2, Initial population element ${\theta^i}_\beta(0)$=0.5, input control -200 < $u(k)$ <200.
- Liquid level Control: $p = 2, S = 1000$, probability of crossover Pm=0.05, mutation probability Pm=0.05, Initial population element ${\theta^i}_\alpha(0)$ =2, Initial population element ${\theta^i}_\beta(0)$ =0.5, input control -50 < $u(k)$ <50.

a) Simulation results of DC motor speed control using indirect adaptive control with GA and BFA are shown in figures (2a) and (2b). The tracking performance of DC motor with reference trajectory shows that after few initialisation indirect adaptive controller is able to track the reference trajectory.

Figures 3a and 3b show the plant nonlinearities alpha and beta estimated by indirect GA and BFA adaptive controller respectively. Estimated plant nonlinearities are almost same as the actual plant nonlinearities. Also the estimated trajectory of DC motor is nearby same to actual trajectory after 10 sec.

The best costs of the foragers are shown in figures 4a and 4b for GA and BFA based adaptive control for DC motor. It takes time before the controller adapts, but that as the cost index decreases over time, the fitness function value decreases hence healthier bacteria are available for reproduction, elimination and dispersal.

Figure 5a shows the error between actual speed and reference trajectory of speed for DC motor using GA adaptive control. Error magnitude is more up to 10 sec, and then it reduces significantly.

Figure 5b shows the error between actual speed and reference trajectory of speed for DC motor using BFA adaptive control. Error is significant in magnitude up to 10 sec, after this error reduces. A comparison of error for GA and BFA adaptive control indicates that there are slightly high fluctuation in speed in GA based control. Hence here performance of BFA based controller is better than GA based controller.

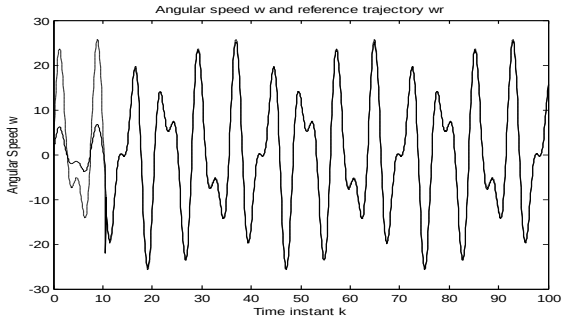

Fig.2a Angular speed of DC motor and Reference speed with GA

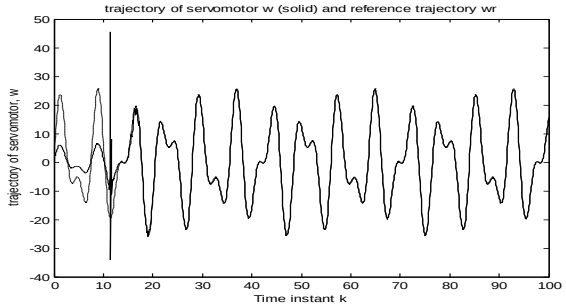

Fig.2b Angular speed of DC motor and Reference speed with BFA

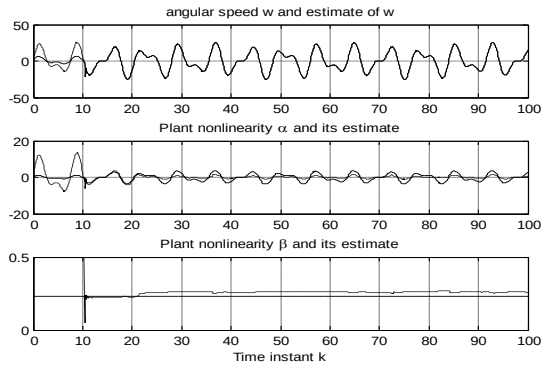

Fig.3a Plant nonlinearities alpha, beta and speed with GA

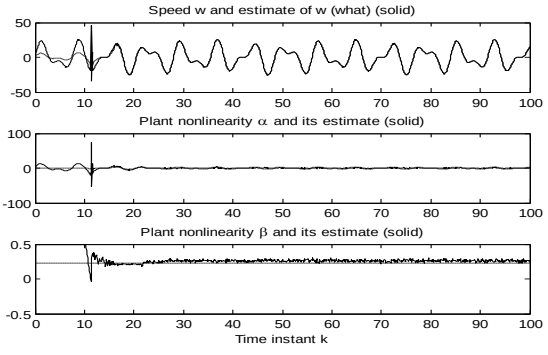

Fig.3b Plant nonlinearities alpha, beta and speed with BFA

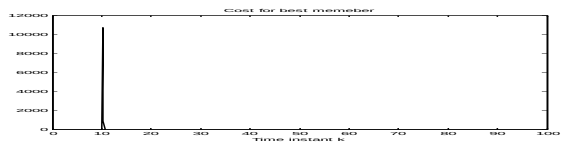

Fig.4a Fitness values of best member in DC motor speed control system with GA

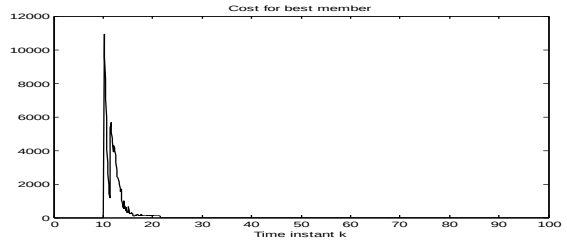

Fig.4b Fitness values of best member in DC motor speed control system with BFA

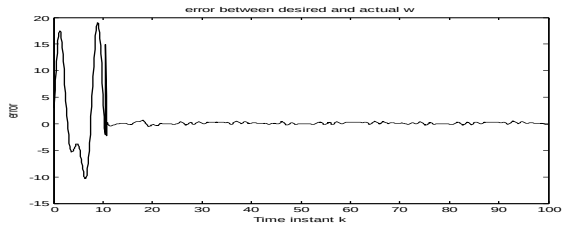

Fig.5a Error between actual speed and estimated speed with GA

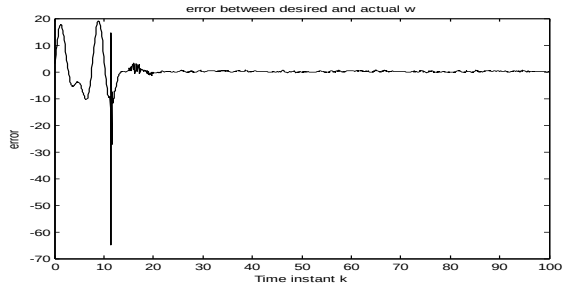

Fig.5b Error between actual speed and estimated speed with BFA

b) The tracking performance of indirect adaptive controller based on GA and BFA with respect to reference square trajectory for liquid level control system are shown in figures 6a and 6b. Both the adaptive controller follows the reference trajectory after 10 sec.

Figures 7a and 7b shows the actual value of liquid level and estimated liquid level obtained from GA and BFA adaptive controller. Also the estimated plant nonlinearities alpha and beta have been shown with respect to actual plant nonlinearities. From figures 7a and 7b it has been observed that nonlinearities estimated by GA and FBA adaptive controller differs slightly as compared to actual plant nonlinearities. Plant nonlinearity alpha is almost matching to actual alpha in liquid level system, while there is slightly more difference between estimated one and actual one for beta nonlinearity.

The costs of the best individuals are shown in figures 8a and 8b for both GA and FBA based adaptive controller. It is observed in the simulation that cost is zero initially, then at 10 sec., the cost jumps to a relatively high value; this represents that a poor initialization initial for the population of bacteria. After some time, however, the foraging strategy is somewhat successful at adjusting the

population members so that the estimation error decreases and hence, the best cost decreases. Also, however, cost does not always decrease over time, it increase and one cause of this is due to change in the reference input. A reduction in fitness value of bacteria indicates presence of healthier bacteria.

Figures 9a and 9b show the error between actual liquid level and reference value of liquid level for GA and BFA based adaptive controller respectively. Error is high from 8 to 10 sec, after this error reduces.

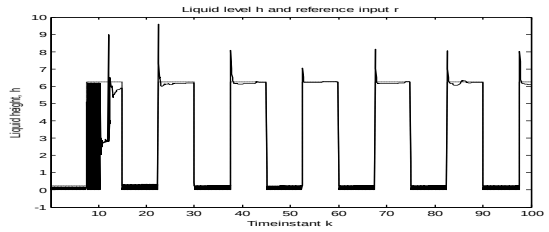

Fig.6a Liquid level and Reference level with GA

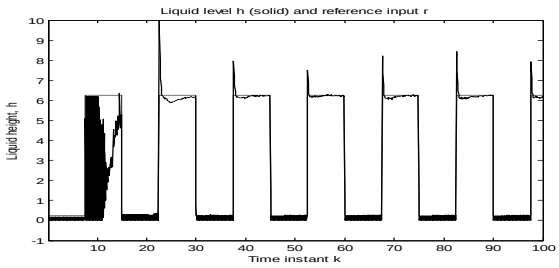

Fig.6b Liquid level and Reference level with BFA

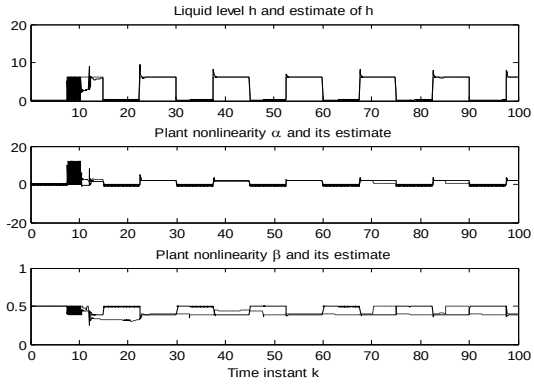

Fig.7a plant nonlinearities alpha, beta and liquid level with GA

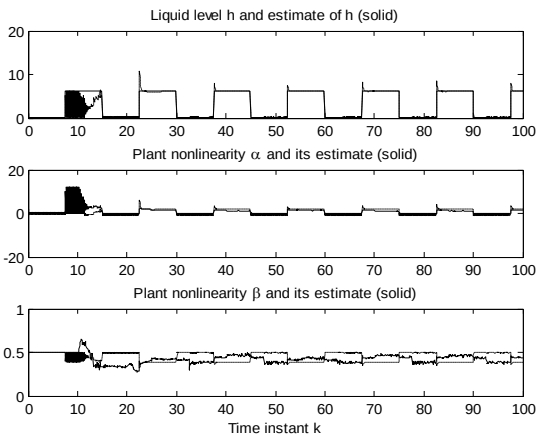

Fig.7b plant nonlinearities alpha, beta and liquid level with BFA

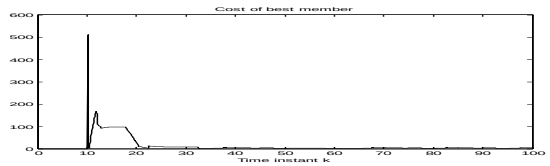

Fig.8a Fitness value of best member in liquid level control system with GA

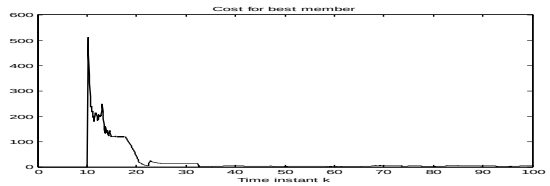

Fig.8b Fitness value of best member in liquid level control system with BFA

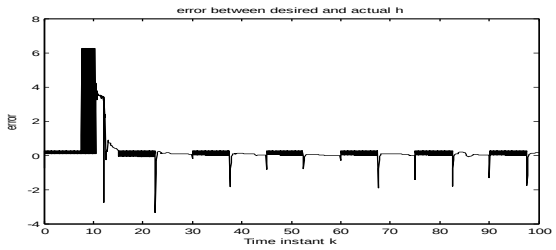

Fig.9a Error between actual liquid level and estimated value of level with GA

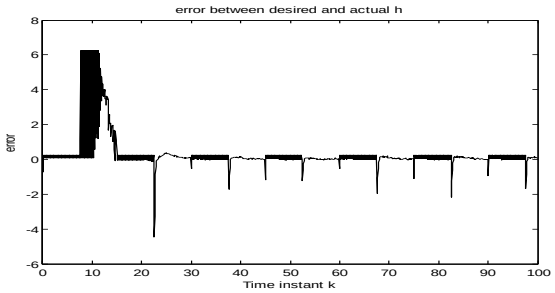

Fig.9b Error between actual liquid level and estimated value of level with BFA

From Table 1 it have been observed that for both the nonlinear systems time taken by BFA based adaptive controller is less than the time taken by GA. The BFA based adaptive controllers improve the dynamic performances of the both nonlinear system.

TABLE 1 ELAPSED TIME IN EACH SIMULATION

Nonlinear system	GA	BFA
DC motor speed control system	127.866 sec	66.073 sec.
Liquid level control of surge tank	129.920 sec	66.065 sec

VI. Conclusion

Bacterial foraging algorithm has been implemented on DC motor speed control system and liquid level control of surge tank system. Indirect adaptive controller based on BFA is used for control of DC motor speed trajectory and liquid level square trajectory. Simulations have been performed for both the systems in MATLAB. It has been observed that BFA based adaptive controller tracks trajectory of both the nonlinear system after ten seconds.

From the responses of both the nonlinear system it has been observed that after an initial transient period that results in part due to the poor initialization of estimators and the controller start up method, a reasonably good tracking of the reference input is obtained. It have been observed that estimates of the angular speed in DC motor and tank liquid level is quite good, even though at times the individual estimates of the nonlinearities are not. Also the error between actual value and estimated value for both the systems reduces nearby to zero. Simulations for both the nonlinear systems have been performed by GA also. In DC motor error in speed trajectory is less fluctuating in BFA adaptive control compare to GA adaptive control. While performance of BFA is nearby same as GA based adaptive control in liquid level system. Elapsed time of simulation is less in BFA adaptive control as compare to GA adaptive control for both the nonlinear systems. Hence BFA based indirect adaptive controller performs well for both the nonlinear systems.

REFERENCES

[1] K. M. Passino, "Biomimicry of Bacterial Foraging for Distributed Optimization and Control," IEEE Control System Magazine, 52-67, 2002.

[2] D.E.Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley Publishing Corporation, Inc., 1989.

[3] J.W. Ma, G.L. Zhang and H. Xie, "The Optimization of Feed-Forward Neural Networks based on Artificial Fish-Swarm Algorithm," Computer Applications, vol. 24, pp. 21-23, Pct. 2004.

[4] W.M. Korani, H.T. Dorrah and H.M. Emara, "Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning" , in proc. of the 10th Annual conference on Genetic and Evolutionary Computation, USA,pp.1823-1826, July 2008.

[5] M. Dorigo, G.D. Caro, and L.M. Gambardella, "Ant Algorithms for Discrete Optimization", Artificial Life, vol. 3, pp. 137-172,1999

[6] J.Kennedy and R.C. Eberhart, "Particle Swarm Optimization," in proc. of the IEEE Int. Conf. on Neural Networks, pp. 1942-1948, 1995.

[7] Y. Chu, H. Mi, H. Liao, Z. Ji and Q.H. Wu, "A Fast Bacterial Swarming Algorithm for High-dimensional Function Optimization," in proc. of IEEE World Congress on Computational Intelligence, pp. 3135-3140,2008.

[8] H. Chen, Y. Zhu and K. Hu, "Self-Adaptation in Bacterial Foraging Optimization Algorithm," in proc. of IEEE Int. Conf. on Intelligent System and Knowledge Engineering, pp.1026-1031, 2008.

[9] Y. Chen and W. Lin, "An Improved Bacterial Foraging Optimization," in Proc. of IEEE Int. Conf. on Robotics and Biomimetics , pp.2057-2062, Dec. 19-23, 2009

[10] S. Dasgupta, A. Biswas, S. Das, B. K. Panigrahi andA. Abrahm, "A Micro-Bacterial Foraging Algorithm for High Dimensional Optimization," in proc. of IEEE Int. Congress on Evolutionary Computation, pp: 785-792, 2009.

[11] H. Chen, Y. Zhu and K. Hu, "Cooperative Bacterial Foraging Algorithm for Global Optimization," in proc. of IEEE Int. Conf. on ,pp. 3896-3901, 2009

[12] W.J. Tang, Q.H. Wu and J.R. Saunders, "Bacterial Foraging Algorithm for Dynamic Environments," in proc. of IEEE Int. Conf. on Evolution Computation, pp. 1324-1330, 2006.

[13] A. Abrahm, A. Biswas, S. Dasgupta and S. Das, "Analysis of Reproduction Operator in Bacterial Foraging Optimization Algorithm," in proc. of IEEE Int. Conf. on, pp.14761483, 2008.

[14] D.G. Jadhav, S.S. Pattnaik, S. Devi, M.R. Lohokare and K.M. Backwad, "Approximate Memtic Algorithm for Consistent Convergence," in proc. of National Conference on Computational Instrumentation, pp. 118-122, 2010.

[15] U. Fang, Y. Liu and J. Liu, "A Novel Simplified Foraging Optimization Algorithm for Parameter Identification of Nonlinear System Model," in proc. of IEEE Int. Conf. on Automation and Logistics, pp. 798-802, 2007.

[16] T. Datta and I. S. Misra, "Improved Adaptive Bacteria Foraging Algorithm in Optimization of Antenna Arry for Faster Convergence," Progress in Electromagnetic Research C, vol.1, pp. 143-157, 2008.

[17] M.Gopal, "Digital Control and State Variable Methods," Tata McGraw-Hill Publishing Company Limited New Delhi, 2003.

[18] S. Nakamura, "Numerical Analysis and Graphic Visualization with MATLAB," Prentice Hall PTR, Upper Saddle River, NJ, 1996.

Madhusudan was born in Ghazipur (U.P.) India, in 1968. He received B.Sc. (Electrical Engineering) degree from the Faculty of Technology, Dayalbagh Educational Institute, Dayalbagh Agra, India, in 1990 and the M.E. degree from the University of Allahabad, Allahabad, India, in 1992. He received his Ph.D. degrees in Department of Electrical Engineering, University of Delhi, India in 2006.

In 1992, he joined the Department of Electrical Engineering, NERIST-Nirjuli, Arunachal Pradesh, as lecturer. In June 1996, he joined Electrical Engineering Department, IET Lucknow, India as a lecturer. In March 1999, he joined Department of Electrical Engineering, Delhi College of Engineering, Delhi India as an Assistant Professor. In October 2007 he became the professor of Electrical engineering in Delhi College of Engineering, Delhi India.

His research interests are in area of modeling and analysis, of Electrical Machines, voltage control aspects of Self–Excited Induction Generator, Power electronics, and drives.

He is a member of the Institution of Engineers (IE), India, Fellow of Institution of Electronics and Telecommunication Engineers, New Delhi, India. He is a Life Member of the Indian Society for Technical Education, New Delhi, India. He is also a member of IEEE (USA).

Bharat Bhushan was born in Delhi in 1971. He received B.E. (Electrical Engineering) degree from the Delhi College of Engineering, University of Delhi, Delhi, in 1992 and the M.E. (Electrical Engineering) degree from the Delhi College of Engineering, University of Delhi, Delhi, in 1996.

In June 1997, he joined the Department of Instrumentation Engineering, Sant Longowal Institute of Engineering & Technology, Longowal, Punjab as Lecturer. In January 1998, he joined Instrumentation & Control Engineering Department, Ambedkar Polytechnic, New Delhi as Lecturer. In May 1999, he joined Department of Electrical Engineering, Delhi College of Engineering, Delhi as Lecturer. In December 2008, he joined as Assistant Professor in Department of Electrical Engineering, Delhi College of Engineering, and Delhi, India. Presently he is Associate Professor in Department of Electrical Engineering, Delhi Technological University (Formerly Delhi College of Engineering), Delhi, India.

His research interests are in the area of Control Systems, Fuzzy Logic, Neural Networks, Genetic Algorithm, Bacterial Foraging Algorithm, Computational Intelligence and Nonlinear Systems.