JAST, Vol. 4, No. 2, pp 1–12

JAST
Journal of Aerospace Science and Technology

Near-Optimal Tuning of Linear Controllers Based on Genetic Algorithm and Swarm Intelligence: A Flight Control Example

Ali Reza Mehrabian[1], Jafar Roshanian[2], Caro Lucas[3]

This paper presents a method for the tuning of PID-type controllers for systems with time-varying dynamics based on optimization by genetic algorithms (GA) and swarm intelligence, i.e. a modified version of particle swarm optimization. The powerful abilities of bio-inspired computing in locating the optimal (or near-optimal) solutions to a given optimization problem are utilized for determining the parameters of the controller in order to meet specified performance objectives for a given problem in terms of weighted integral of different signals. The presented study recommends a new modification in PSO algorithm where a novel approach for tuning of the learning factors is employed to improve the speed of convergence and the accuracy of the search in the multi-dimensional space. The proposed technique has the flexibility to be applicable to a wide range of control problems. In order to illustrate the performance of the technique, development and application of this algorithm for an aerospace launch vehicle during atmospheric flight in an experimental setting is presented.

NOMENCLATURE

c_1	Cognitive component learning factor (also called acceleration coefficient)
c_2	Social component learning factor (also called acceleration coefficient)
c_i^{iter}	The ith ($i = 1, 2$) learning factor at the present time step
$c_i^{\max}$	The ith ($i = 1, 2$) maximum (final) learning factor
$c_i^{\min}$	The ith ($i = 1, 2$) minimum (initial) learning factor
$e_{\text{ac}}(t)$	Actuator realization error
$e_{\text{pa}}(t)$	Pitch angle error
$e_{\text{pr}}(t)$	Pitch rate error
F_x	Force along the X axis
F_y	Force along the Y axis
F_z	Force along the Z axis
$\mathbf{g}_{\text{best}}$	Particle's best global position in multi-dimensional space
I_x	Moment of inertia along the X axis
I_y	Moment of inertia along the Y axis
I_z	Moment of inertia along the Z axis
iter	The iteration number at the present time step
$\text{iter}_{\max}$	Maximum number of iterations in a given search
J	Performance index (also called fitness or objective function)
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain
M_x	Angular momentum along the X axis

1. *Ph.D Candidate, Department of Electrical and Computer Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, Quebec, Canada H3G 1M8. email: armehrabian@gmail.com.*
2. *Associate Professor, Department of Aerospace Engineering, K.N. Toosi University of Technology, P.O.Box: 16765–3381, Tehran, Iran. email: roshanian@kntu.ac.ir.*
3. *Professor, Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, P.O.Box: 14395–515, Tehran, Iran; and School of Cognitive Science, IPM, Tehran, Iran. email: lucas@ipm.ir.*

M_y	Angular momentum along the Y axis
M_z	Angular momentum along the Z axis
$M]_{U,q,\delta}$	Dynamic coefficients of the longitudinal momentum
m	System's mass
N	Number of agents
n_i	The ith ($i = 1, 2$) non-linear inflection index
$\mathbf{p}_{\text{best}}$	Particle's best local position vector in multi-dimensional space
p	Roll angle rate of change
q	Pitch angle rate of change
r	Yaw angle rate of change
t_f	Flight time
U	Linear velocity along the X axis
V	Linear velocity along the Y axis
$\mathbf{v}$	Particle velocity vector in multi-dimensional space
$\mathbf{v}_k$	Particle velocity vector in multi-dimensional space at the kth step
$\nu_{\max}$	Maximum allowable particle velocity in multi-dimensional space
$\nu_{\min}$	Minimum allowable particle velocity in multi-dimensional space
W	Linear velocity along the Z axis
w	Momentum factor (also called inertia weight)
$w_{\max}$	Particle's maximum allowable momentum factor
$w_{\min}$	Particle's minimum allowable momentum factor
$\mathbf{x}$	Particle position vector in multi-dimensional space
$\mathbf{x}_k$	Particle position vector in multi-dimensional space at the kth step
$x_{\max}$	Maximum allowable particle position in multi-dimensional space
$x_{\min}$	Minimum allowable particle position in multi-dimensional space
$Z]_{U,q,\theta,\delta}$	Dynamic coefficients of the longitudinal force
δ_c	Commanded thrust vector deflection in the pitch channel (also called control effort)
δ_e	Thrust vector deflection in the pitch channel
θ	Pitch angle
[TF]	Transfer function
$\dot{[\bullet]}$	Newton's notation for derivative with respect to time ($= d[\bullet]/dt$)

INTRODUCTION

Mathematical optimization methods can be classified into two different sets: direct and gradient-based search methods.[1] In direct search methods, only objective (also called "fitness" or "performance") function and constraint values are used to guide the search strategy, whereas gradient-based methods use the first and/or second-order derivatives of the objective function and/or constrains to guide the search process. Since derivative information is not used, the direct search methods are usually slow, requiring many function evaluations for convergence that increases the computational load. For the same reason, they can also be applied to many problems without a major change of the algorithm. On the other hand, gradient-based methods quickly converge to an optimal solution that needs not to be the global optimum, and cannot be used in the case of non-differentiable or discontinues objectives. In addition, there are some common difficulties with most of the traditional direct and gradient-based techniques:

1. Most algorithms tend to get stuck to a suboptimal solution.
2. An algorithm efficient in solving one optimization problem may not be efficient in solving a different optimization problem.
3. Algorithms are not efficient in handling problems having discrete variables.

Because nonlinearities and complex interactions among variables often exist in engineering design problems, the search space may have more than one optimal solutions, of which most are undesired locally optimal solutions having inferior objective function values. When solving these problems, if traditional methods get attracted to any of these locally optimal solutions, there is no solution. To overcome these problems, researchers have proposed evolutionary-based algorithms for searching near-optimum solutions to problems.[2]

Evolutionary algorithms (EA) are stochastic search algorithms that imitate the metaphor of natural biological evolution and/or the social behavior of species. Examples include how birds find their destination during migration and Darwin's theory of evolution, which basically stresses on the fact that the existence of all living things is based on the law of "survival of the fittest". Various researchers have developed computational systems that seek fast and robust solutions to complex optimization problems mimicking the behavior of biological systems. The leading evolutionary-based optimization research introduced in the literature was the genetic algorithms (GA).[3] Basically, GA is a method for finding solutions to optimization or search problems by means of simulated evolution (i.e. concepts from genetics). GA is a set of adaptive (search, learning) methods based on the genetic processes of biological organisms. Based on its demonstrated ability to reach near-optimum solutions

to large problems, the GA method has been used to solve several problems in science and engineering.[4–7] Though GA techniques have been employed successfully to solve complex optimization problems, recent studies have acknowledged some difficulties in GA performance. This degradation in efficiency of GA is noticeable in applications when the objective function should be optimized by highly correlated parameters. Furthermore, the premature convergence of GA degrades its performance and decreases its search capability that directs to a higher probability in the direction of obtaining a local optimum.[8,9] Beside various improvements in GA, recent developments in EA include other techniques inspired by different biological processes, e.g., memetic algorithms (MA),[10] particle swarm optimization (PSO),[11] invasive weed optimization (IWO),[12] and ant-colony system (ACS).[13]

PSO has recently developed as an important branch of combinatorial meta-heuristic techniques which operate on a population of potential solutions to explore the search space for optimization.[14] PSO is motivated by the simulation of social behavior instead of survival of the fittest. While each member of the swarm bears minimal intelligence, the swarm as a whole shows a good amount of intelligence as if the intelligence of each member is added up. Associated with a velocity, each candidate (also called "particle") "flies" through the search space. The velocity is constantly adjusted according to the corresponding particle's experience and the particle's companions' experiences. It is expected that the particles will move towards better solution areas.[15]

Regardless of vast advances in the field of control systems engineering, PID scheme still remains the most common control algorithm used today in the industry. It is widely used because of its flexibility, high reliability and ease of operation and tuning (see for example Ref. 16). PID controller design methods differ with respect to the knowledge of the process dynamics they require.[16,17] These techniques were developed empirically through the simulation of a large number of process systems to provide a simple rule. The methods operate particularly well for simple systems and those which exhibit a clearly dominant pole-pair, but for more complex systems the PID gains may be strongly coupled in a less predictable way. In many practical situations the system is parameter varying and/or nonlinear with powerful disturbances; thus design methods based on classical PID tuning methods are not reliable. Another possibility for obtaining a compromise between several different criteria is to use optimization methods. Optimization is a powerful tool for controller design. The method is simple in conception. A control scheme with few parameters is specified. Next, system specifications are expressed as inequalities of functions of parameters. Then, the important specifications are chosen as a function (performance index) to be optimized. The method can be applied to design PID parameters easily. Different performance criteria for PID controllers are addressed in Ref. 16. Indeed care must be exercised when defining performance index, since it may have many local minima. Another difficulty is the computations required may easily be excessive. However, optimization based on EA can overcome these difficulties.

In the current study it is sought to employ two evolutionary algorithms, i.e. GA and PSO, for optimizing gains of an Aerospace Launch Vehicle (ALV) PID-like autopilot during atmospheric flight. The problem of designing intelligent adaptive autopilot for the ALV was conducted in a pervious study (Ref. 18). However, the study of optimal constant gain controller for the system is not addressed before. The design procedure and controller performance evaluation is addressed to compare GA and PSO for finding near-optimal gains for control.

GA AND PSO BASED OPTIMIZATION

In general, EAs share a common approach for their application to a given problem. The problem first requires some representation to suit each method. Then, the evolutionary search algorithm is applied iteratively to arrive at a near-optimum solution. A brief description of the two algorithms, GA and PSO is presented in the following subsections.

Genetic algorithms

GA is a search/optimization technique that uses genetics and natural selection as a model for problem solving. In the GA, a population of randomly created individuals (chromosomes) goes through a simulated process of evolution, which is a digital version of the survival of the fittest, law in which individuals in the current population are bred to produce new individuals hopefully better suited to the environment. Each individual represents a potential solution to the problem, a candidate set of parameter values. GA works on the coding of the parameters and not on the exact parameters so that it does not depend on the continuity of the parameters or the existence of derivatives of the functions required in some conventional optimization algorithms. The coding method allows the GA to handle multi-modal (i.e. many-peaked) and multi-parameter type optimization problems easily, which are rather difficult or impossible to be treated with classical optimization methods. The sequential steps for searching optimal solution using GA are shown in Figure 1. Reference 2 gives a general overview of heuristic search methods including GA.

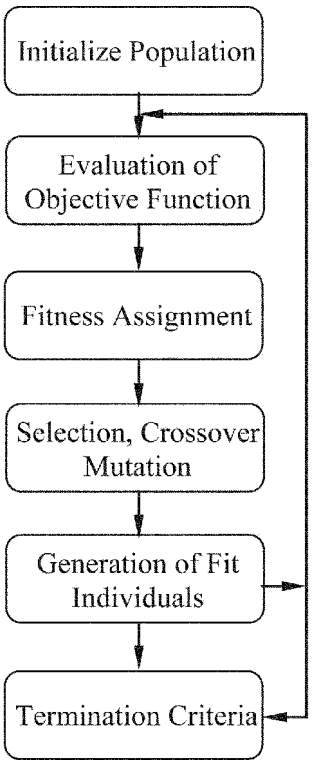

Figure 1. GA computational flow chart.

Particle swarm optimization

The particle swarm optimization (PSO) technique was developed by Kennedy and Eberhart.[11] The PSO is inspired by the social behavior of a flock of migrating birds (or fishes) trying to reach an unknown destination. A particle is analogous to a chromosome (population member) in GA. Nevertheless, the evolutionary process in the PSO does not create new particles from the parents. To a certain extent, the birds in the population only change their social behavior and accordingly their movement towards a destination. Physically, this mimics a flock of birds that communicate together as they fly. Each bird looks in a specific direction, and then when communicating together, they identify the bird that is in the best location. Consequently, each bird speeds towards the best bird using a velocity that depends on its current position. Each bird, then, investigates the search space from its new local position, and the process repeats until the flock reaches a desired destination. It is important to note that the process involves both social interaction and intelligence so that birds learn from their own experience (local search) and also from the experience of others around them (global search).[2]

To formulate this social behavior, Each "particle" is represented by a vector in multi-dimensional space to characterize its position, $\mathbf{x}$, and another vector to characterize its velocity $\mathbf{v}$. To update velocity in each time step $\mathbf{v}_{k+1}$ is a function of three major components: the current velocity of the particle ($\mathbf{v}_k$), the difference of the best position particle ever reached (best local position, $\mathbf{p}_{\text{best}}$) and the current position ($\mathbf{x}_k$), and the difference of the best position found so far in the swarm (global best position, $\mathbf{g}_{\text{best}}$) and the current position ($\mathbf{x}_k$). The second and third components are stochastically weighted and added to first component. The basic PSO algorithm can be described in vector notation as follows:

$$\mathbf{v}_{k+1} = w \times \mathbf{v}_k + \underbrace{\text{rand} \times c_1 \times (\mathbf{p}_{\text{best}} - \mathbf{x}_k)}_{\text{cognitive component}} + \underbrace{\text{Rand} \times c_2 \times (\mathbf{g}_{\text{best}} - \mathbf{x}_k)}_{\text{social component}} \tag{1a}$$

Velocity clamping :

$$\begin{cases} \text{if} \quad \nu_{k+1} > \nu_{max} \quad \text{then} \quad \nu_{k+1} = \nu_{\max} \\ \text{if} \quad \nu_{k+1} < \nu_{min} \quad \text{then} \quad \nu_{k+1} = \nu_{\min} \end{cases} \tag{1b}$$

Displacement clamping :

$$\begin{cases} \text{if} \quad x_{k+1} > x_{max} \quad \text{then} \quad x_{k+1} = x_{\max} \\ \text{if} \quad x_{k+1} < x_{min} \quad \text{then} \quad x_{k+1} = x_{\min} \end{cases} \tag{1c}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1} \tag{1d}$$

At iteration k the velocity vector $\mathbf{v}_k$ is updated based on its current value affected by a momentum factor, or inertia weight (w),[19] and on a term which attracts the particle towards previously found best positions. The strength of attraction to the best local position is given by the coefficient , and to the global best position by the coefficient c_2 (c_1 and c_2 are also called "acceleration coefficients" or "learning factors"). The particle position, $\mathbf{x}$, is updated ($\mathbf{x}_{k+1}$) using its current value and the newly computed velocity $\mathbf{v}_{k+1}$. Randomness necessary for good space exploration is introduced via the vectors of random numbers rand and Rand, usually selected as uniform random numbers in the range $[0, 1]$.

Since the first publication of PSO algorithm, a large body of research has been done to study either the performance of PSO, or to improve PSO performance. Showing very fast convergence, much effort has been invested to obtain a better understanding of the convergence properties of PSO. These studies concentrated mostly on the influence of the basic PSO control parameters (the learning factors, momentum factor, velocity and displacement clamping, and swarm size). From these empirical studies it can be concluded that the PSO is sensitive to control parameter choices, specifically the momentum factor, learning factors and velocity/displacement clamping. Wrong initialization of these parameters may lead to divergent or cyclic behavior.[20,21] In a well tuned PSO the particles show a logical behavior as follows:

1. The velocity of particles increases; thus, the particles are spread over the search space.
2. The particles can locate optimal (global and/or local) solutions to the optimization problem while moving randomly over the search space.
3. The particles try to move towards global optimal solution while slowing down; but some of them are being trapped in locally optimal solutions.
4. The particles being trapped in founded local or global solution slow down and search for better solutions in their local search space, until they can find the best possible answer.

Nevertheless, a search algorithm that suffers from bad tuning may show any one of the following behaviors:[22]

1. The velocity of particles increases rapidly, and particles go out of the search space.
2. The velocity of particles decreases rapidly, and particles nearly stop.
3. Particles are trapped in locally optimal solutions and cannot get out.

In order to avoid these undesirable conditions an analysis between the relation of the parameters and the behavior of particles can be carried out.[21,22] However, there is not any rigid formulation/procedure reported in the literatures for determining PSO parameter; thus, the parameters are often selected by trial and error. It is suggested to choose learning factors within 0.2 to 2 and select a momentum factor that decreases linearly from 1.4 ($w_{\max}$) to 0.5 ($w_{\max}$).[2,19] In this fashion, global search starts with a large weight and then decreases with time to favor local search over global search.[2,23]

Particle swarm optimization with dynamic learning factor (MPSO)

Introducing the concept of momentum factor (inertia weight), Shi and Eberhart[19] suggested that a better performance would be obtained if the momentum factor were chosen as a linear-time-varying dynamically decreasing quantity, rather than being a constant value.[23] This idea was followed by Chatterjee and Siarry,[14] where they suggested a non-linear dynamic decrease in the momentum factor. A higher value of the momentum factor implies larger incremental changes in velocity per time step which mean exploration of the new search areas in pursuit of a better solution. However, a smaller value of the momentum factor implies less variation in velocity to provide slower updating for fine tuning of a local search. It is supposed that the PSO algorithm should start with a high momentum factor for crude global search and the momentum factor should decrease (linearly/non-linearly) to facilitate finer local explorations in later iterations.[14,19]

The learning factors, c_1 and c_2, are parameters that characterize ardency of the particles in moving towards best local/global positions. It is clear that at the beginning of the search, when particles are being spread over the search space by an initially large momentum factor, the swarm is not intended to converge to optimal solutions; but in return, it is desired that the swift swarm keeps on locating the possible solutions for the problem. This idea can be achieved by selecting relatively smaller values for learning factors in the beginning of the search that means the swarm has lesser tendency in gathering around local solutions, which are found in the beginning of the search. As a consequence, the swarm keeps on exploring new spaces for possible potential solutions to the problem that may contain even better answers for the optimization problem. The value of the learning factors, therefore, can be increased when the exploration of the search space is performed successfully, in order to absorb the particles towards locally/globally optimal solutions for finer tuning in later iterations. The idea is determination of the learning factors as a non-linear function of the present iteration number (iter) at each time step. The proposed formulation for learning factors is given as:

$$c_i^{\text{iter}} = \frac{(\text{iter}_{\max} - \text{iter})^{n_i}}{(\text{iter}_{max})_{n_i}}(c_i^{\min} - c_i^{\max}) + c_i^{\max} \quad \text{where} \quad i = 1, 2 \tag{2}$$

where $c_i^{\min}$ is the ith ($i = 1,2$) minimum (initial) learning factor at the commence of a given search, and $c_i^{\max}$ is the ith ($i = 1,2$) maximum (final) learning factor at the end of a given search, when $\text{iter}_{\max}$ is the maximum number of iterations in a given search, iter is the iteration number at the present time step, c_i^{iter} is the ith ($i = 1,2$) learning factor at the present time step, and n_i is the ith ($i = 1,2$) non-linear inflection index. Figure 2 shows typical learning factor with iterations for different values of n_i ($i = 1,2$) on either side of unity. With $n_i = 0$ ($i = 1,2$), the system becomes a special case of constant learning factor with time, as proposed by Shi and Eberhart.

The search algorithm starts with low initial learning factors, which should allow it to explore new search areas aggressively without much interest in assembling around local/global solutions of the problem, and then increases it slowly according to equation (2) following different paths for different values of n_i to reach $c_i^{\max}$ at $\text{iter}_{\max}$. The objective is to arrive at an attractive solution for any given problem by decreasing the chance of being trapped in potential local answers that are found in commerce by the immature swarm for a given search problem.

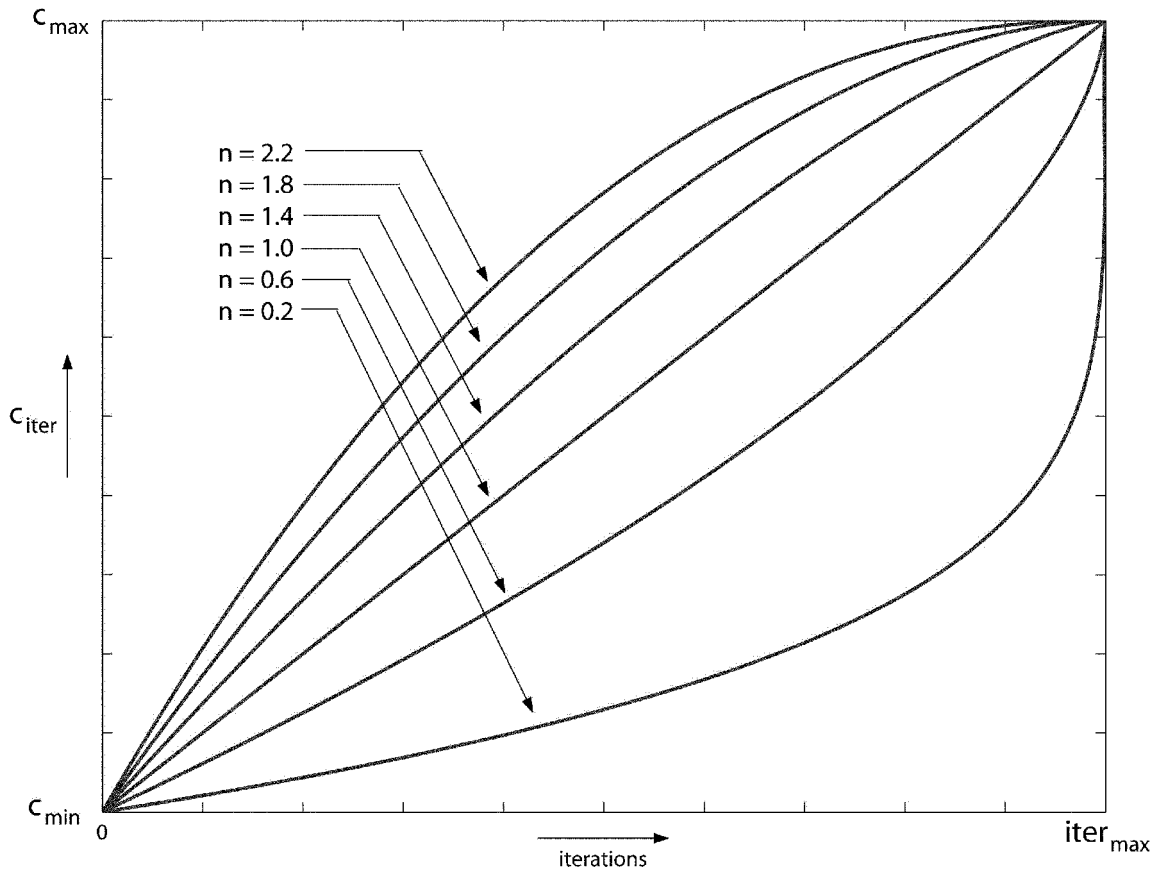

Figure 2. Variations of learning factor for different values of non-linear inflection index.

A DESIGN EXAMPLE

The model used in this analysis is based on the aerospace launch vehicle model employed as a base line in previous studies for non-linear/adaptive control (see Ref. 18).

ALV model

The equation of motion for an ALV can be derived from Newton's second law of motion. Considering rigid airframe for an ALV, six degree of freedom (6-DOF) equations of motions can be obtained as follows:[24]

$$\begin{aligned} F_x &= m(\dot{U} + qW - rV) \\ F_y &= m(\dot{V} + qU - rW) \\ F_z &= m(\dot{W} + qV - rU) \\ M_x &= I_x\dot{p} \\ M_y &= I_y\dot{q} + (I_x - I_y)pr \\ M_z &= I_y\dot{r} + (I_y - I_x)pq \end{aligned} \tag{3}$$

The control design for a linear ALV is being considered in this paper, due to the fact that the ALV attitude control systems are usually designed using linearized equations of motion. This mainly is because the nominal trajectory of the system is planned to maintain the ALV at near-zero angle of attack. This is normally attempted by programming the pitch attitude or pitch rate to yield a zero-g trajectory. [24] Therefore, the assumption of near-zero angle of attack for the equilibrium condition is quite valid, and any changes in angle of attack can be considered perturbations from the equilibrium conditions. Thus, considering small perturbations and rigid airframe, linearized longitudinal equations of motion of an ALV can be obtained as follows:

$$\dot{U} = Z_U U + Z_q q + Z_\theta \theta + Z_\delta \delta \tag{4}$$

$$\dot{q} = M_U U + M_q q + M\delta\delta \tag{5}$$

where Z, M represent dynamic coefficients, and the control force is provided by the deflection of thrust vector shown by δ. Variations of longitudinal dynamic coefficients and ALV parameters during the atmospheric part of powered flight using 6-DOF implementation are obtained as shown in Figure 3. (For confidentiality reasons, given flight time and parameters are unit-less.)

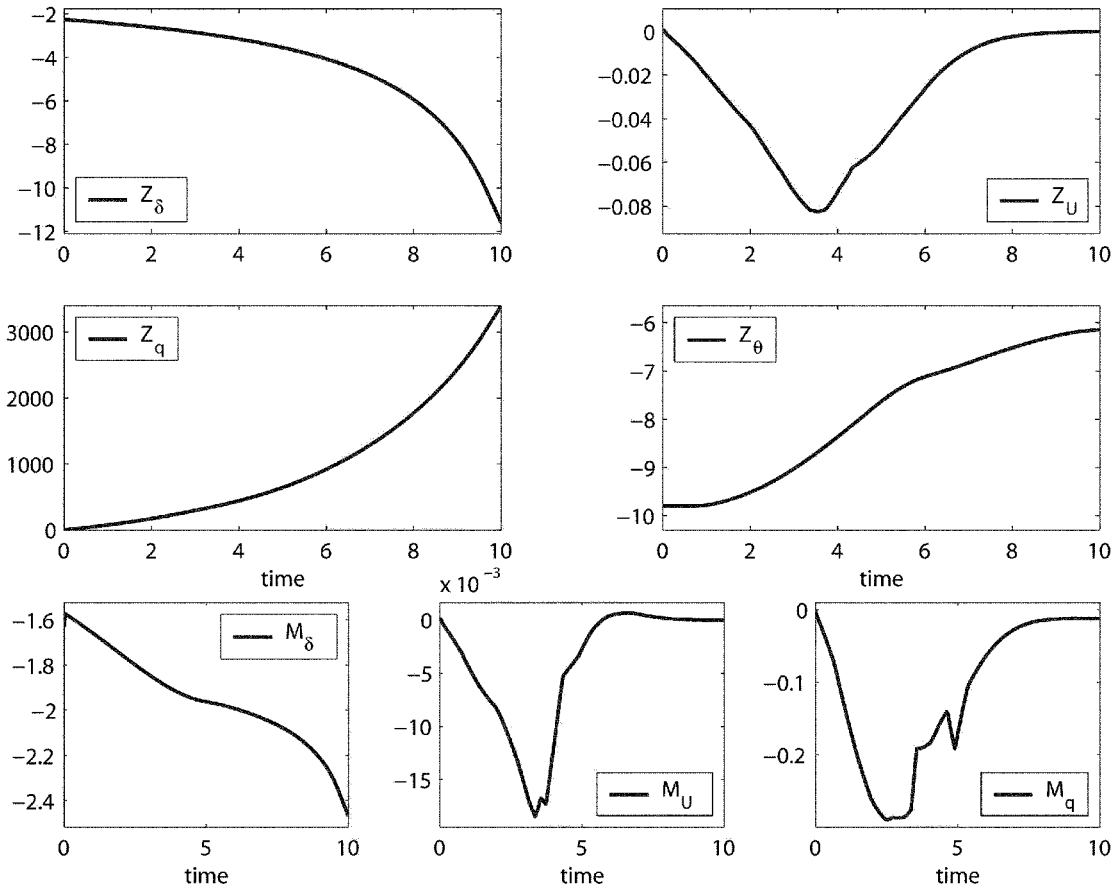

Figure 3. Figure 3 Longitudinal dynamic coefficients.

The servo dynamics describing the thrust vector deflection is:

$$[\mathrm{TF}]_{\mathrm{servo}} = \frac{\delta_e}{\delta_c} = \frac{1}{0.1s+1} \tag{6}$$

with rate limit of $\left|\frac{d}{dt}\delta_e\right| < 2.5$ deg/sec. Reference signal of the control system is pitch rate, so that a rate gyro is used for measuring obtained pitch rate which has dynamics described as follows:

$$[\mathrm{TF}]_{\mathrm{gyro}} = \frac{(80\pi)^2}{s_2 + 40\pi s + (80\pi)^2} \tag{7}$$

ALV control circuit

Pitch program of an ALV is provided by guidance systems. Some guidance systems provide only a pitch program whereas some others also require that the control system be capable of accepting a commanded pitch rate. Considering the second case, block diagram of a typical ALV attitude control system is shown in Figure 4. It is illustrated in the mentioned figure that the controller is provided by the pitch rate program and measured pitch rate signal. Constructing pitch rate and pitch angle error (by integrating pitch rate error) the controller allocates constant gains K_d, and K_p to each signal respectively. There is also a K_i gain associated with the integral of the pitch angle error. Thus, the control command, δ_c, is generated by multiplication of the pitch rate, the pitch angle, and the integral of pitch angle error signals with assigned constant gains. The controller is similar to the conventional PID controller where the differential signal is available by measurement. Gains of the control law are generally selected by automatic control theory techniques, such as pole-zero placement method of the closed-loop transfer function. Solution to this problem in application to the launch vehicle attitude control system has the following particular features: 1) Vehicle dynamic coefficients are time-variant along the trajectory. Therefore, to satisfy the response and accuracy requirements, the feedback loop gains can be chosen as a time variable. These gains can be selected as programmed versus time or the vehicle net acceleration along the nominal path, or can be calculated during a flight by some adaptive optimal or robust algorithms considering external disturbances

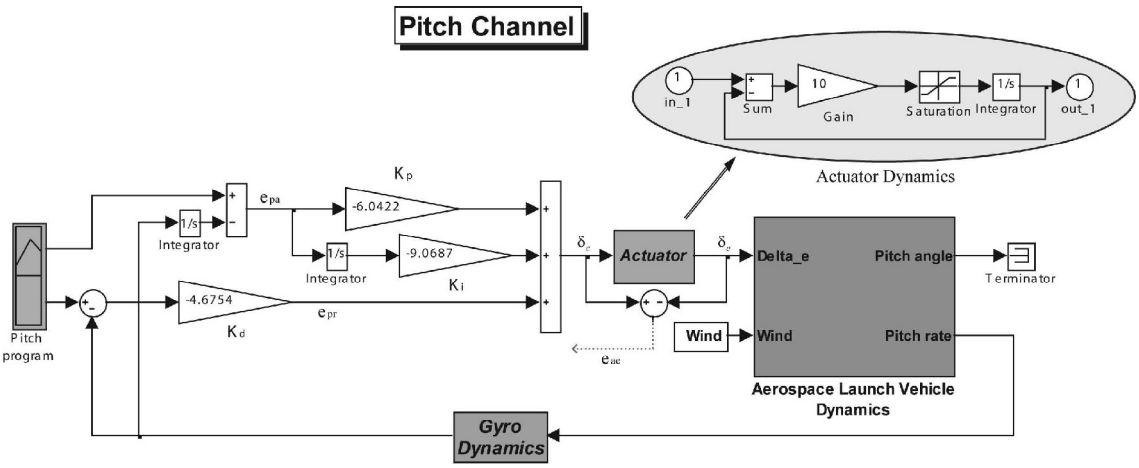

Figure 4. ALV control circuit.

acting on the vehicle. 2) Attitude control system is influenced by different disturbances, such as stage motor thrust deviations from nominal, stage separation disturbances, atmospheric density deviations from reference, wind in atmosphere, vehicle aerodynamic coefficient uncertainties, etc.

To take into account the above mentioned features and to satisfy design requirements of modern autopilots many methodologies have been developed during the last few years.[25−29] The procedure for tuning controller gains based on non-linear optimization technique is specified in the following section.

ALV AUTOPILOT DEVELOPMENT BASED ON EA

To reach an adequate controller with good performance and low sensitivity to wind disturbances it is proposed to tune the parameters of the controller throughout atmospheric flight. It is clear that classical dynamics and steady-state measures of performance, e.g. over shoot, rise time, and settling time are inconveniencing in multi-step reference trajectory evaluation. This is particularly true when the final scalar objective function is to be a weighted sum of the different performance measures. The problem here arises in attaching weighting coefficients to each of these measures so that they correspond directly to the relative importance of the objectives or allow trade-offs between the objectives to be expressed. Therefore, use of integral criteria as a measure of performance for control systems was considered in this work. The comprehensive performance index (fitness function) employed in this study is given in equation (8):

$$\begin{aligned}\underset{K_i,K_p,K_d}{\text{Min:}} \quad J &= 10\ln\left\{\int_0^f \left[10|e_{\text{pr}}(t)| + 10\int_0^f |e_{\text{pr}}(t)|dt\right.\right.\\ &\left.+0.1|\delta_c(t)| + 100|e_{\text{ac}}(t)|\right] dt\} + 1\\ &= 10\ln\left\{\int_0^f [10|e_{\text{pr}}(t)| + 10|e_{\text{pa}}(t)|\right.\\ &\left.+0.1|\delta_c(t)| + 100|e_{\text{ac}}(t)|\right] dt\} + 1\end{aligned} \tag{8}$$

where $e_{\text{pr}}(t)$ indicates pitch rate error, $e_{\text{pa}}(t)$ stands for pitch angle error, and $e_{\text{ac}}(t)$ for actuator realization error as illustrated in Figure 4. The mentioned fitness function minimized errors in tracking nominal pitch rate and pitch angle (shown in Figure 5) as well as the control effort, $|\delta_c(t)|$ and the error caused by the difference between actuator input and output signals. The latter term is included to prevent difficulties

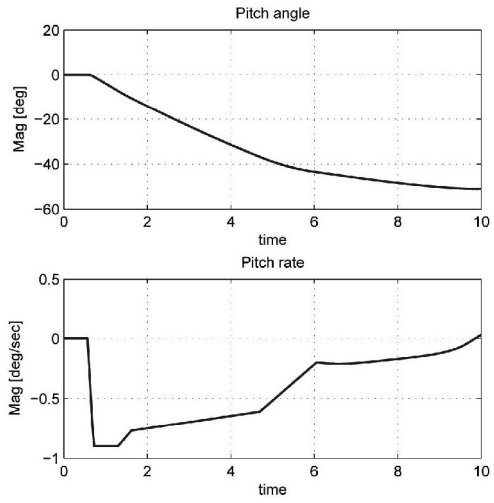

Figure 5. Nominal pitch program.

caused by improper control signal, e.g. noise and saturation in the command signal δ_c. Clearly, there is no analytical solution to minimizing the mentioned performance index; thus, the performance index is minimized using GA and PSO by tuning the parameters of the controller.

AIV autopilot development based on GA

Gains of the controller were tuned by GA in order to minimize performance criteria given in equation (8). Parameters of the GA algorithm used in this research are given in Table 1. In order to reach near-optimal solution, in every iteration the best population is found by maturing best individuals of the previous step. The size of the population must be related to the size of the search space, ensuring a sufficient number of points for the evolutionary algorithm prospect. In the present case, a population of size $N = 10$ was selected in order to reduce the computational time, with the search space limited to $K]_{i,p,d} \in [-1818]$. The maximum population iteration was used as a stop criterion; i.e. the search algorithm stops after 100 iteration. The controller parameter evaluation process by GA is shown in Figure 6. The best fitness function reached after 100 iterations is given in Table 3.

AIV autopilot development based on PSO

The controller design procedure by PSO is very similar to the design by GA method. Like GA, agents (in PSO "particles") are randomly spread over the design space. There is only one difference in the way they converge to the optimal position as addressed in Section 2. PSO parameters in this study, obtained by trial and error, are given in Table 2 for both original PSO and modified PSO (MPSO).

Results addressed in Ref. 14 indicate that a higher choice of $w_{\max}$ consumes more iterations to give the required solution; hence, experiment with different values of $w_{\max}$ and number of iterations is avoided here. The values for the learning factors in PSO are set equal to average values of the respective parameters in MPSO, i.e. $(c_i^{\max} + c_i^{\min})/2$ (where $i = 1, 2$). The best

Table 1. GA settings

Population size	10
Initial range for particle position	[-15, 15]
Iterations	100
Mutation	Gaussian
Crossover probability	0.8
Selection strategy	stochastic uniform
Coding scheme	double vector

Table 2. PSO and MPSO parameters

	PSO	**MPSO**
Population of particles	10	10
Initial range for particle position	[-15, 15]	[-15, 15]
Initial range for particle velocity	[-1, 1]	[-1, 1]
Iterations	100	100
$w_{\max}$	2	2
$w_{\min}$	0.3	0.3
$\nu_{\max}$	5	5
$\nu_{\min}$	-5	-5
$x_{\max}$	15	15
$x_{\min}$	-15	-15
c_1	1.1	–
c_2	0.6	–
$c_1^{\max}$	–	1.4
$c_1^{\min}$	–	0.8
$c_2^{\max}$	–	1
$c_2^{\min}$	–	0.2
n_1	–	2
n_2	–	2

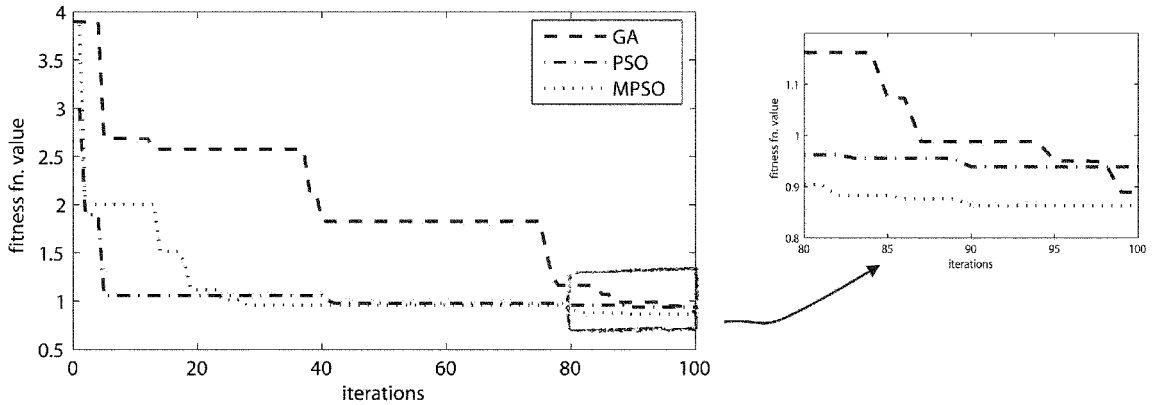

Figure 6. Fitness function values plotted against iterations for GA, PSO, and MPSO algorithms.

Table 3. Controller parameters obtained by GA, PSO, and MPSO

	GA	PSO	MPSO
K_d	1.9578	1.6022	1.8558
K_p	2.7258	3.1111	2.5333
K_i	18.0000	12.9201	18.0000
Fitness function value	0.8885	0.9386	0.8631

fitness function found after 100 iterations is given in Table 3.

SIMULATION RESULTS AND ANALYSES

The evaluation process of controller parameters for GA, PSO, and MPSO algorithm is illustrated in Figure 6. One can see that in each case the algorithm does converge to a solution. Results indicate that the GA algorithm improves gradually, and could successfully reach fairly near-optimal solution at the end of the run; however, it is slower than PSO algorithm.[2] PSO algorithm was able to locate an optimal answer quickly and converge to the solution in a short time. As a matter of fact, it was trapped in a local solution (premature convergence) as it can be seen from achieved parameters and the objective function mentioned in Table 3. MPSO, on the other hand, was able to locate potential solutions to the problem, whilst the swarm had a high speed at the beginning of the search. In the middle of the course for finding the optimal solution while GA and PSO were kept on locating fitter answers, MPSO was gathering the agents in order to look for fitter particles later, the fact that was established at the end of the search.

Performance of the designed controllers is studied by simulating them on the system for nominal conditions as shown in Figure 7. The MPSO parameterized controller is intended to have lesser fitness function during atmospheric flight; however, the results indicate satisfactory outcomes with small differences in tracking commands provided by guidance section for all three controllers. This, in fact, acknowledges the decency of the selected performance index. In Figure 8 the performance of the controller that was parameterized by MPSO algorithm is demonstrated in presence of a powerful gust to illustrate the robustness of the design. The gust profile is given in Figure 9.

For performance comparisons, the results are given against a GS controller presented in Ref. 18 and 30 (proposed in Ref. 26 for first time). Showing very good robustness, the proposed controller tracks

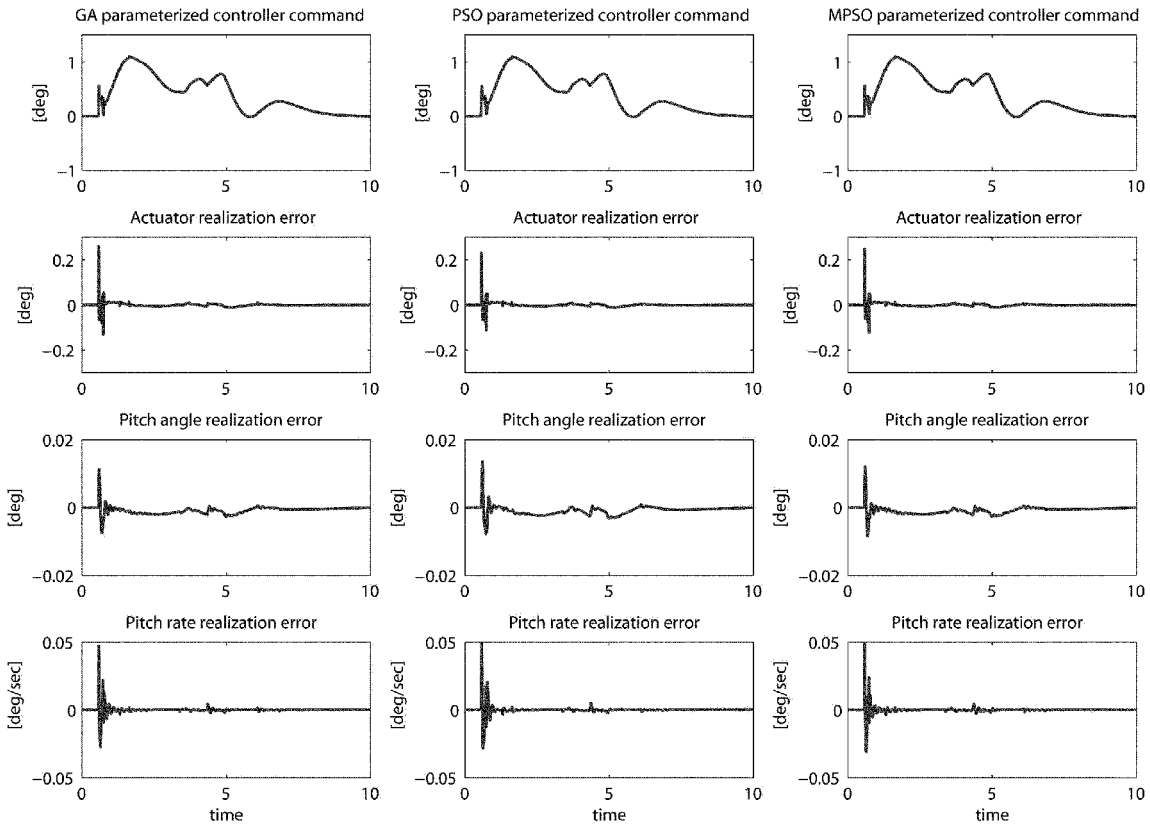

Figure 7. Time history of the closed-loop system for nominal conditions.

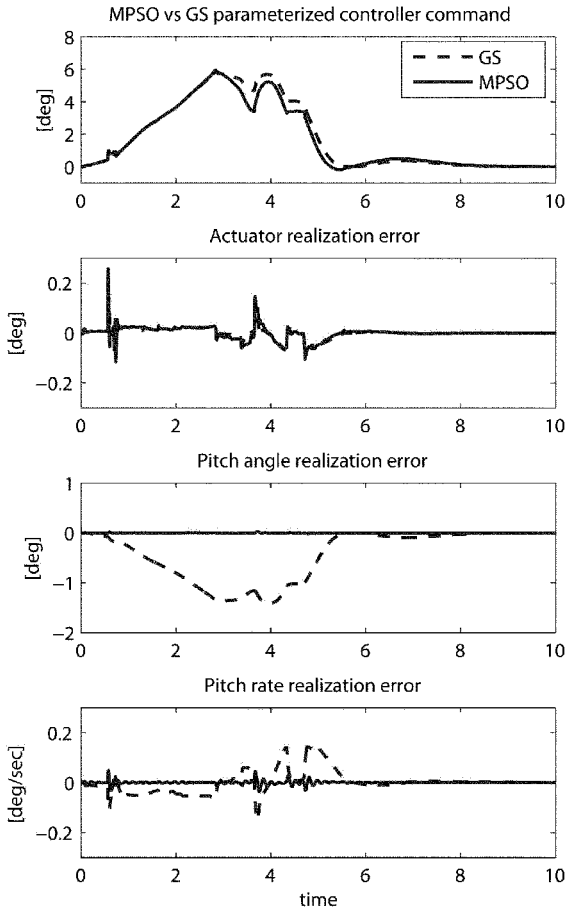

Figure 8. Time history of the closed-loop system for perturbed conditions.

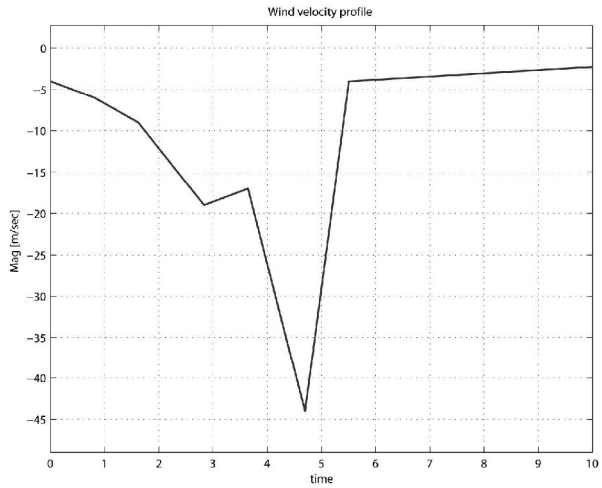

Figure 9. Sample wind velocity profile.

the commanded pitch angle and pitch rate very closely in presence of such a powerful disturbance.

CONCLUSIONS

Socio-biologically inspired algorithms have, in recent years, been widely applied for finding good solutions to complex tasks. This paper sought to develop a new-modification in PSO algorithm and use it in a demanding aerospace engineering task, i.e., the problem of controller development for time-varying systems. In this paper, two evolutionary algorithms, GA and PSO, are employed for non-linear problems of tuning the gains of a linear controller. The controller parameters are found by optimizing a fitness function. Both GA and PSO have shown very good convergence powers for finding proper controller gains.

A new variation of PSO, i.e. MPSO, which is introduced in this study, employs non-linear variation of the learning factors of the swarm. Using proposed algorithm, the swarm is not intended to converge to optimal solution at the beginning of the search (premature convergence), so that the swarm can do aggressive search during initial iterations to achieve better search of the solution space, with little interest in early founded near-optimal solutions. Thus, the swarm can arrive at the global optimal solution at the end of the search, and is not trapped in locally optimal solutions. Fine tuning of the answer is left for later iterations when the slower particles are brought together by increasing the learning factor so that the optimal solution can be reached with higher accuracy. Application results, i.e., the problem of optimization of a fitness function presented by a weighted integral of different signals, e.g., tracking error (for pitch angle and pitch rate), control effort, and actuator realization error are reported to acknowledge applicability of this thesis. Notice that the fitness function has various numbers of local minima with no noticeable difference in their fitness function value. For instance, the minima that were found by PSO, however were local, but are slightly different in value with the minima found by MPSO, while the values of the parameters (K_i, K_p, K_d), were far-away. This fact certainly makes the problem of finding the global minimum more difficult for any search algorithm.

REFERENCES

1. Back, T., Fogel, D., and Michalewicz, Z. (eds.), *Handbook of Evolutionary Computation*, Institute of Physics Pub. and Oxford Univ. Press, (1997).
2. Elbeltagi, E., Hegazy, T., and Grierson, D., "Comparison among five evolutionary-based optimization algorithms", *Advanced Engineering Informatics*, **19**, PP 43-53(2005).
3. Holland, J. H., *Adoption in Natural and Artificial Systems*, University of Michigan, (1975).
4. Jamshidi, M., Krohling, R. A., and Coelho, L. S., *Robust Control Systems with Genetic Algorithms*, CRC Press, (2002).

5. Wang, P., and Kwok, D. P., "Optimal design of PID process controllers based on genetic algorithms", *Control Engineering Practice*, **2(4)**, PP 641-648(1994).
6. Yokoyama, N., and Suzuki, S., "Modified Genetic Algorithm for Constrained Trajectory Optimization", *Journal of Guidance, Control, and Dynamics*, **28(1)**, PP 139-144(2005).
7. Mashadi, H. R., Modir Shanechi, H., and Lucas, C., "A new genetic algorithm with Lamarckian learning for generation scheduling", *IEEE Trans. on Power Systems*, **18(3)**, PP 1181-1186(2003).
8. Angeline, P. J., "Using selection to improve particle swarm optimization", *Proc. IEEE Int. Conf. Evolutionary Computing*, Anchorange, Alaska, PP 84-89(1998).
9. Gaing, Z.-L., "Particle swarm optimization to solving the economic dispatch considering the generator constraints", *IEEE Trans. on Power Systems*, **18(3)**, PP 1187-1195(2003).
10. Moscato, P., "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms", *Technical Report Caltech Concurrent Computation Program, Report 826*, California Institute of Technology, Pasadena, CA, (1989).
11. Kennedy, J., and Eberhart, R., "Particle swarm optimization", *Proc. of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, PP 1942-1948(1995).
12. Mehrabian, A. R., and Lucas, C., "A Novel Numerical Optimization Algorithm Inspired from Weed Colonization", *Ecological Informatics*, **1(4)**, PP 355-366 doi: 10.1016/j.ecoinf(2006.07.003).
13. Dorigo, M., Maniezzo, V., and Colorni, A., "Ant system: optimization by a colony of cooperating agents", *IEEE Trans. Syst. Man and Cybern*, **26(1)**, PP 29-41(1996).
14. Chatterjee, A., and Siarry, P., "Nonlinear inertia weight variation for dynamics adaptation in particle swarm optimization", *Computers and Operation Research*, **33**, PP 859-871(2006).
15. Ghoshal, S. P., "Optimizations of PID gains by particle swarm optimizations in fuzzy based automatic generation control", *Electric Power Systems Research*, **72**, PP 203–212(2004).
16. Astrom, K. J., and Hagglund, T., *PID Controllers: Theory, Design, and Tuning*, ISA, Research Triangle Park, NC, (1995).
17. Astrom, K. J., and Hagglund, T., "Revisiting the Ziegler–Nichols step response method for PID control", *Journal of Process Control*, **14**, PP 635–650(2004).
18. Mehrabian, A. R., Lucas, C., and Roshanian, J., "Aerospace Launch Vehicle Control: An Intelligent Adaptive Approach", *Aerospace Science and Technology*, **10(2)**, PP 149-155 doi: 10.1016/j.ast(2005.11.002).
19. Shi, Y., and Eberhart, R., "A modified particle swarm optimizer", *Proceedings of the IEEE international conference on evolutionary computation*, Piscataway, NJ, (1998).
20. Van den Bergh, F., and Engelbrecht, A. P., "A study of particle swarm optimization particle trajectories", *Information Sciences*, **176(8)**, PP 937-971 doi: 10.1016/j.ins(2005.02.003).
21. Trelea, I. C., "The particle swarm optimization algorithm: convergence analysis and parameter selection", *Information Processing Letters*, **85**, PP 317–325(2003).
22. Yasuda, K., Iwasaki, N., "Adaptive particle swarm optimization using velocity information of swarm", *IEEE Int. Conf. on Systems, Man and Cybernetics*, **4**, PP 3475–3481(2004).
23. Eberhart, R., and Shi, Y., "Comparison between genetic algorithms and particle swarm optimization", *Proceedings of the 7th annual conference on evolutionary programming*, Springer, Berlin, PP 611–618(1998).
24. Blakelock, J. H., *Automatic Control of Aircraft and Missiles*, Wiley, (1991).
25. Filho, W. C. L., "Application of Adaptive Control-Sounding Rocket Attitude", *Proceeding of Int. Conf. on Intelligent Autonomous Control in Aerospace*, Beiging, China, (1995).
26. Filho, W. C. L., and Hsu, L., "Adaptive Control of Missile Attitude", *IFAC Conf. on Adaptive Systems in Control and Signal Processing*, Lund, Sweden, (1986).
27. Malyshev, V. V., and Krasilshikov, M. N., *Aerospace Vehicle Navigation and Control*, FAPESP Publication, (1996).
28. Clementa, B., Ducb, G., and Mauffrey, S., "Aerospace launch vehicle control: a gain scheduling approach", *Control Engineering Practice*, **13**, PP 333-347(2005).
29. Voinot, O., Alazard, D., and Piquereau, A., "A robust multiobjective synthesis applied to launcher attitude control", *15th IFAC symposium on automatic control in aerospace*, Bologna/Forlì, Italy, (2001).
30. Saleh, A. R., Roshanian, J., "Design of an adaptive autopilot for a launch vehicle based on gain scheduling theory", *5th Conference of Iranian Aerospace Society*, Esfehan, Iran, (in Persian), (2004).