Online Optimal Controller Design using Evolutionary Algorithm with Convergence Properties

Yousef Alipouri[1], Javad Poshtan[2]
1- Departmen of Electrical Engineering, University of Science and Technology, Tehran, Iran
E-mail: yalipouri@iust.ac.ir
2- Department of Electrical Engineering, University of Science and Technology, Tehran, Iran
E-mail: jposhtan@iust.ac.ir

Received: August 2013 Revised: December 2013 Accepted: February 2014

ABSTRACT
Many real-world applications require minimization of a cost function. This function is the criterion that figures out optimally. In the control engineering, this criterion is used in the design of optimal controllers. Cost function optimization has difficulties including calculating gradient function and lack of information about the system and the control loop. In this article, for the first time, gradient memetic evolutionary programming is proposed for minimization of non-convex cost functions that have been defined in control engineering. Moreover, stability and convergence of the proposed algorithm are proved. Besides, it is modified to be used in online optimization. To achieve this, the sign of the gradient function is utilized. For calculating the sign of the gradient, there is no need to know the cost-function's shape. The gradient functions are estimated by the algorithm. The proposed algorithm is used to design a PI controller for nonlinear benchmark system CSTR (Continuous Stirred Tank Reactor) by online and off-line approaches.

KEYWORD: Nonlinear Optimal Controller, GMEP (Gradient Memetic Evolutionary Programming), Sign of Gradient Function, Online Optimization, Nonlinear Benchmark CSTR.

1. INTRODUCTION

The performance of the most systems (especially in industrial applications) can be upgraded by using the methods of optimization and optimal control. One of the basic problems in this course is complexity and non-linearity of the system, which makes classical control methods that are linear in nature but they're not applicable in nonlinear optimization problems, or it proves them as inaccurately. Normally, it is assumed that the process can be approximated well by a linear model at least around the current operating point. This is appropriate for regulatory control, but maybe it's not the best option for process outputs that showing large amplitude (or frequency) changes. A useful but difficult research direction is to develop methods for nonlinear systems.

Designing of optimal nonlinear controllers is confronted with two main problems: 1) Nonlinear Cost Function Optimization; and 2) Determining the structure and parameters of the controller. The optimal controller for nonlinear systems has normally a nonlinear structure. The optimization of nonlinear-controller parameters by classical methods is difficult and sometimes impossible. Optimality of a controller is measured by a cost function. The most common cost function in optimal control is defined as follows [1]:

$$z(t)=\frac{1}{2}\int_{t=0}^{t}\left(e(\tau)^2+u(\tau)^2\right)d\tau. \quad (1)$$

where e is the output tracking error and u is the control signal.

Assuming that the controller is structured as follows:

$$u(t)=g\left(y(t),\ldots,y(t-d),\theta(t)\right) \quad (2)$$

where $\Theta(t)$ are the controller of unknown parameters, time-dependent in the general case need to be adaptively optimized.

The gradient based method is a usual method for calculating the optimal parameters:

$$\frac{\partial z(t)}{\partial \theta(t)}=0. \quad (3)$$

To calculate this, there are two fundamental problems: 1) There is a need to calculate $\frac{\partial e(t)}{\partial \theta(t)}=\frac{\partial e(t)}{\partial u(t)}\frac{\partial u(t)}{\partial \theta(t)}$, where calculating $\frac{\partial e(t)}{\partial u(t)}$ by considering the system has unknown properties (in this study, the system is considered to be black box), is not applicable. and 2) This function should be adaptively optimized in the

general case and in each instance (or in moments of need). Optimization of nonlinear and non-convex cost functions is very difficult and many classical methods cannot be applied, therefore, in most optimal controller designs, linear models and methods such as LQG are being used. In this case, using intelligence-techniques such as Evolutionary Algorithms to solve this problem can be useful [2]. Since Evolutionary Algorithms (EAs) do not need the gradient function, the problems which exist in the classical algorithms are not seen in EAs. Using EAs to optimize cost functions defined in control engineering is not an easy task. The main disadvantages of such algorithms are: 1) There is no guarantee in the stability and convergence of EAs; and 2) These algorithms cannot be used in online applications [3]. For engineering applications, especially in control engineering, stability and the convergence of the algorithm are very important aspects, because none of these features can lead to the instability of the control system, and hence, they may cause physical and financial harms. Nevertheless, the key issue is the reliability of the algorithm. Since evolutionary algorithms (similar to other intelligent algorithms) are faced with the problem of convergence and stability, they cannot be used in many applications, especially in online applications in which , the reliability is an important factor and few controllers can be tested and evaluated on the system. As EAs are population-based algorithms, the population prevents their online applications [4]. Unfortunately, without population, these algorithms lose their ability in searching global minimum, which is their main advantage over classical algorithms.

If we could somehow fulfill both addressed problems, EAs can be the best option for nonlinear and non-convex optimization problems. This paper deals with the implementation of this goal. In section II, the convergence of the proposed algorithms is analyzed. Then, the online version of the proposed algorithm is studied. Moreover, section III presents the results of the proposed algorithm to design an online and offline optimal controller for a nonlinear CSTR benchmark system.

2. THE PROPOSED ALGORITHM: GRADIENT MEMETIC EVOLUTIONARY PROGRAMMING (GMEP)

For solving the problem of convergence in evolutionary algorithms, some features of classical algorithms can be utilized. Many classical algorithms use the gradient of the cost function. The gradient function acts like a map, and shows the direction to the algorithm. The algorithm starts at an arbitrary point on the cost surface and it minimizes along the direction of the gradient to reach the minimum point, which is usually a local minimum. Therefore, if this feature could be utilized, it gives advantage to have the convergence condition in EAs. However, it must be considered that EAs have a black box view to the cost function. Any information about the gradient of the cost function will not be accepted. Otherwise, the algorithm performance will drop dramatically. Then, it will try to approximate the gradient of the cost function in order to be able to guarantee the convergence. It is useful to firstly become familiar with evolutionary programming.

a) Evolutionary Programming (EP)

An Evolutionary Algorithm is an iterative and stochastic process that operates on a set of individuals (population). Each individual represents a potential solution to the problem that's being solved. This solution is obtained by means of an encoding/decoding mechanism. Initially, the population is randomly generated (perhaps with the help of a construction heuristic). Using a fitness function, every individual in the population is assigned a value as a measure of its goodness with respect to the problem under consideration. This value is the quantitative information that the algorithm uses it to guide the search [5].

Evolutionary algorithm was first proposed between years 1940-1950 with the first generation of modern evolutionary algorithms by Fogel [6], Rechenberg [7], Holand [8] and Koza [9], beginning with evolutionary programming, genetic algorithms and Evolutionary Strategies (ES).

Evolutionary programming (EP) was presented for the first time by Fogel. But, it was not used for about 30 years. Then in 1980, he used the algorithm in the continuous parameter optimization. EP and ES have some similarities. In both algorithms, the key operator is mutation, and the operators such as the selection of parents and the crossover are not done [10]. Moreover, in both EP and ES, the choice of the next generation is performed with a competition, which will be explained below. The mutation operator is the only operator in EP that produces the next generation. Therefore, most methods were proposed to improve the algorithm changing the mutation operator.

This algorithm is as follows [11]:

1) Generate the initial population of μ individuals, and set $k = 1$. Each individual is taken as a pair of real valued vectors, $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$, where the x_i's are the objective variables and the η_i's are the standard deviations for Gaussian mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).
2) Evaluate the fitness score for each individual $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$ of the population, based on the objective function $f(x_i)$.

3) Each parent $(x_i, \eta_i), i = 1,...., \mu$ creates a single offspring (x_i', η_i') by the following: for $j = 1,..., n$

$$x_i'(j) = x_i(j) + \eta_i(j) N_j(0,1)) \quad (4)$$

$$\eta_i'(j) = \eta_i(j) \exp(\tau' N(0,1) + \tau N_j(0,1)) \quad (5)$$

where $x_i(j)$, $x_i'(j)$, $\eta_i(j)$ and $\eta_i'(j)$ denote the j–th component of the vectors x_i, x_i', η_i and η_i', respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with a mean of zero and a standard deviation of one. $N_j(0,1)$ indicates that the random number is generated anew for each value of j. The factors τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(2\sqrt{n})^{-1}$.

4) Calculate the fitness of each offspring $(x_i', \eta_i'), \forall i \in \{1,....,\mu\}$.

5) Conduct pairwise comparisons over the union of parents (x_i, η_i) and the offspring $(x_i', \eta_i'), \forall i \in \{1,....,\mu\}$. For each individual, q opponents are chosen uniformly at random from all of the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."

6) Select the μ individuals out of (x_i, η_i) and $(x_i', \eta_i'), \forall i \in \{1,....,\mu\}$, that have the most wins to be parents of the next generation.

7) Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

Gaussian mutations *N(0,1)* are used in this algorithm. The mutation produces a random vector R = (r_1, r_2, ..., r_n) by the Gaussian probability distribution function. According to Equation (4), the next generation is produced by summing the random vector and the values of the parent variables (6).

$$x' = x + R \quad (6)$$

Usually, the classic choice for the distribution function is the Gaussian type; however, other types such as Cauchy [11], Exponential [12,13], Levy [14] and etc are proposed. For reviewing new methods proposed to improve EP, see reference [15].

The algorithm can be interpreted graphically as:

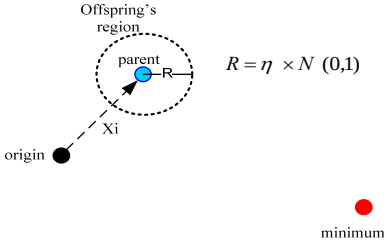

Fig. 1. The region of offspring production

Therefore, multiplying the white noise in the strategy parameter will determine the radius, where the new point is generated within this radius. The location is completely random and it depends on the amounts of white noise and the strategy parameter. It should be also noted that the strategy parameters have a random mode and it depends on the amount of white noise (5). This process of producing new generations is repeated for every new point. Therefore, the searching technique is as follows:

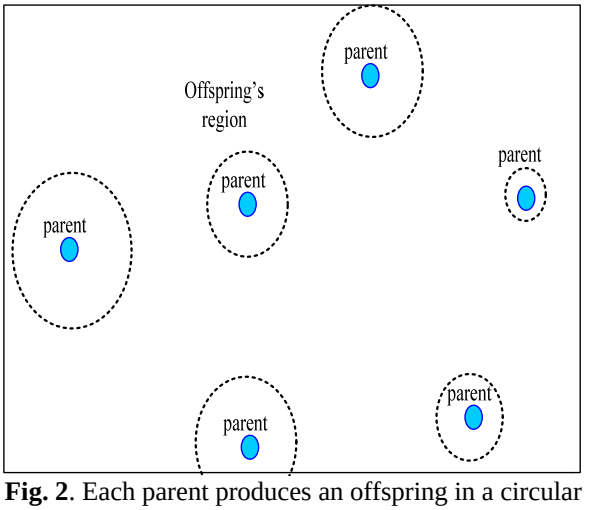

Fig. 2. Each parent produces an offspring in a circular region with random radius

The radius of the circle is very important in achieving the global minimum. If the algorithm is stuck in a local minimum, the value of the radius determines whether or not the algorithm is able to get out of the local minimum, and the magnitude of the radius will determine the speed and stability of the algorithm. The larger radius makes the algorithm search a larger area. Consequently, the speed of the algorithm increases. On the other hand, this reduces the heritage information transferred from the former generation, and hence the stability of the algorithm decreases. Hence, finding the right radius was always one of the challenges of this algorithm. Many of the techniques presented for improving the algorithm have noted this issue and attempted to increase the radius. For more details about how the radius affects the performance of the algorithm, see [16]. The parents and produced offsprings illustrated in Fig. (3) are generated by the EP.

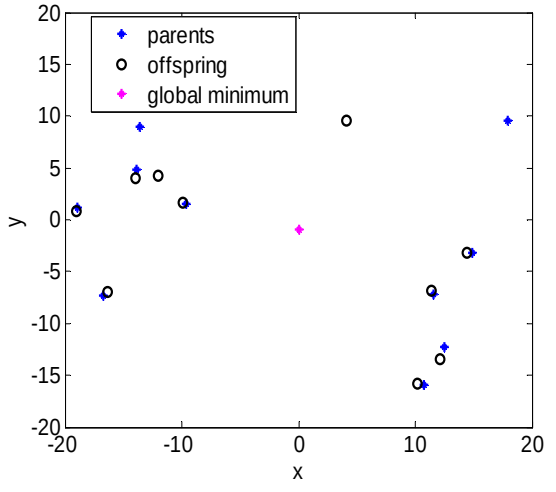

Fig. 3. shows the produced offsprings by their parents [15]

The following solution is proposed to establish the convergence of the algorithm. The circular area in Fig. (1) is modified to a semi-circular area in Fig. (4), where the curve inclines toward the optimal point.

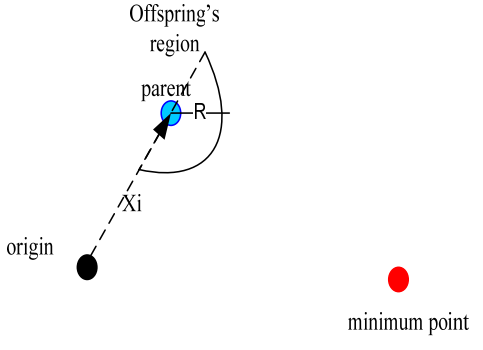

Fig. 4. The circular region is modified to half-circle region by the proposed algorithm

It can be shown that the points generated by this method are getting closer to the minimum. To illustrate this, we prove the convergence of the proposed algorithm.
For implementing this idea, it is proposed to change the circular shaped area (Fig. 1) to the half-circle shape (Fig. 4), the production of new individuals (4) is modified as follows:

$$x_{new} = x_{old} - \eta_{old} |N(0,1)| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x_{old}}\right) \tag{7}$$

In the above, only the sign of the gradient is added. This modification is derived from [5]. Now suppose that the cost function is defined as follows:

$$z(x) = \frac{1}{2} e^2(x) \tag{8}$$

The Lyapunov function can be defined as follows:

$$V(x) = Z(x) = \frac{1}{2} e(x(k))^2 \tag{9}$$

For the stability of the algorithm, the change of Lyapunov function must be negative:

$$V(k+1) - V(k) = \frac{1}{2}\left[e(k+1)^2 - e(k)^2\right] \tag{10}$$

Deviation in the cost function (8) can be written as follows:

$$e(k+1) = e(k) + \nabla e(k) \quad \Rightarrow \quad e(k+1)^2 = e(k)^2 + \nabla^2 e(k) + 2e(k)\nabla e(k) \tag{11}$$

Therefore, we have:

$$\nabla V(k+1) = \frac{1}{2}\left[e(k+1)^2 - e(k)^2\right] = \nabla e(k)\left[e(k) + \frac{1}{2}\nabla e(k)\right] \tag{12}$$

Let's suppose the deviation in the cost function is small, as a result, we can write:

$$\nabla e(k) = \frac{\partial e(k)}{\partial x} \nabla x \tag{13}$$

Considering relation (7), we have:

$$\nabla x = -\eta |N(0,1)| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right) \tag{14}$$

By substituting relations (13) and (14) in (12), we have:

$$\nabla V(k+1) = \frac{\partial e(k)}{\partial x}\left[-\eta |N(0,1)| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right] \cdot \left(e(k) + \frac{1}{2}\frac{\partial e(k)}{\partial x}\left[-\eta |N(0,1)| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right]\right) \tag{15}$$

The above equation can be rewritten as follows:

$$\nabla V(k+1) = -\frac{\partial e(k)}{\partial x}\left[\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right] e(k) + \frac{1}{2}\left[\frac{\partial e(k)}{\partial x}\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right]^2 \quad (16)$$

According to the cost function (8), it can be written as:

$$\nabla V(k+1) = -\frac{1}{2}\frac{\partial z(x)}{\partial x}\left[\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right] + \frac{1}{2}\left[\frac{\partial e(k)}{\partial x}\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right]^2 \quad (17)$$

For stability, the Lyapunov function should be negative:

$$\frac{\partial z(x)}{\partial x}\left[\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right] \geq \left[\frac{\partial e(k)}{\partial x}\eta \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right]^2 \quad (18)$$

We know that the following equation is always true:

$$\frac{\partial z(x)}{\partial x}\left[\left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right)\right] \geq 0 \quad (19)$$

Thus, to establish (18) we have:

$$\eta \geq \left[\frac{\partial e(k)}{\partial x}\eta\right]^2 \left|N(0,1)\right| \operatorname{sgn}\left(\frac{\partial z(x)}{\partial x}\right) \Big/ \frac{\partial z(x)}{\partial x} \quad (20)$$

Simplifying the above equation we have:

$$\eta \geq \left[\frac{\partial e(k)}{\partial x}\eta\right]^2 \left|N(0,1)\right| \Big/ \left|\frac{\partial z(x)}{\partial x}\right| \quad (21)$$

The above equation can be rewritten as:

$$\eta > 0 \quad (22)$$

Therefore, we have:

$$0 < \eta \leq \left|\frac{\partial z(x)}{\partial x}\right| \Big/ \left(\frac{\partial e(k)}{\partial x}\right)^2 \left|N(0,1)\right| \quad (23)$$

Therefore, the algorithm is sufficient to ensure that the strategy parameter has a positive value and it is small enough. However, an important question arises: How can $sign\left(\frac{\partial z(x)}{\partial x}\right)$ be calculated?

At first, it must be noted that the first order derivative function can be approximated with the accuracy of desirable orders, for example $O(h^3)$. The following equations hold for function $f(.)$ by considering the Taylor series expansion.

$$f(t-2h) = f(t) - 2hf'(t) + 2h^2 f''(t) - \frac{4h^3}{3}f'''(t) + \frac{2h^4}{3}f^{(4)}(t) + O(h^5). \quad (24)$$

$$f(t-h) = f(t) - hf'(t) + \frac{h^2}{2}f''(t) - \frac{h^3}{6}f'''(t) + \frac{h^4}{24}f^{(4)}(t) + O(h^5). \quad (25)$$

$$f(t) = f(t). \quad (26)$$

$$f(t+h) = f(t) + hf'(t) + \frac{h^2}{2}f''(t) + \frac{h^3}{6}f'''(t) + \frac{h^4}{24}f^{(4)}(t) + O(h^5). \quad (27)$$

The weighted sum of the above equations produces Eq. (28):

$$af(t-2h) + bf(t-h) + cf(t) + df(t+h) = (a+b+c+d)f(t) + hf'(t)(d-b-2a) + \frac{h^2}{2}f''(t)(d+b+4a) + \frac{h^3}{6}f'''(t)(d-b-8a) + \frac{h^4}{24}f^{(4)}(t)(d+b+16a) + \max(a,b,d)O(h^5). \quad (28)$$

Choosing the weighted coefficients in the following manner results in Eq. (29):

$$a = \frac{1}{6h}, \quad b = -\frac{1}{h}, \quad c = \frac{1}{2h}, \quad and \quad d = \frac{1}{3h}.$$

$$\frac{f(t-2h) - 6f(t-h) + 3f(t) + 2f(t+h)}{6h} = f'(t) + \frac{h^3}{12}f^{(4)}(t) + O(h^4). \quad (29)$$

In Eq. (29), the error is of order $O(h^3)$.

This approach can be used in any order of derivative functions to achieve any desired accuracy. It has been shown theoretically in Einar Hille theorem [17,18 that the discrete Taylor series expansion can approximate any function with the desired accuracy. Thus, the relation (7) can be rewritten as follows:

$$x^{i+m+1} = x^{i+m} - \eta^{i+m} \left| N(0,1) \right| \operatorname{sgn}\left(\frac{\alpha_1 z^{i+m} + \alpha_2 z^{i+m-1} + \ldots + \alpha_m z^{i}}{\alpha_1 x^{i+m} + \alpha_2 x^{i+m-1} + \ldots + \alpha_m x^{i}} \right) \quad (30)$$

Based on the degree used to approximate the gradient function, the parameters α_i are specified. z_i and x_i are defined in the following algorithm. The proposed algorithm can be written as follows:

1) Generate the initial population of μ individuals, and set $k = 1$. Each individual is taken as a pair of real valued vectors, $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$, where the x_i 's are the objective variables and the η_i 's are the standard deviations for Gaussian mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).
2) Evaluate the fitness score for each individual $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$ of the population, based on the objective function $f(x_i)$.
3) Select each parent and *m-1* individuals randomly for estimating the gradient function, and then each parent $(x_i, \eta_i), i = 1,....,\mu$ creates a single offspring (x_i', η_i') by the following: for $j = 1,...,n$

$$x_j^{i+m+1} = x_j^{i+m} - \eta_j^{i+m} \left| N(0,1) \right| \operatorname{sgn}\left(\frac{\alpha_1 z^{i+m} + \alpha_2 z^{i+m-1} + \ldots + \alpha_m z^{i}}{\alpha_1 x^{i+m} + \alpha_2 x^{i+m-1} + \ldots + \alpha_m x^{i}} \right) \quad (31)$$

$$\eta_j^{i+m+1}(j) = \left| \eta_j^{i+m}(j) \exp(\tau' N(0,1) + \tau N_j(0,1)) \right| \quad (32)$$

where x_j^{i+m} and η_j^{i+m} denote the j–th component of the vectors x^{i+m} and η^{i+m}, respectively. x^{i+m} is the parent and x^{i+r}, $r = 0,\ldots,m-1$, are the points selected for estimating the gradient function, and z^{i+r} are corresponding cost value.

4) Calculate the fitness of each offspring.
5) Conduct pairwise comparisons over the union of parents (x_i, η_i) and the offspring $(x_i', \eta_i'), \forall i \in \{1,....,\mu\}$. For each individual, q opponents are chosen uniformly at random from all of the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win."
6) Select the μ individuals out of parents and offsprings that have the most wins to be the parents of the next generation.
7) Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

It must be noted that, at each stage, the proposed algorithm generates new points, and these points become closer to the original minimum (otherwise it will be removed from the population) So, the algorithm has the convergence condition.

Consider the cost function $z = x_1^2 + x_2^2$. The proposed algorithm is used to optimize this cost function. The convergence of the proposed algorithm can be viewed by the individuals generated for the cost function with two variables illustrated in Figs. 5-8.

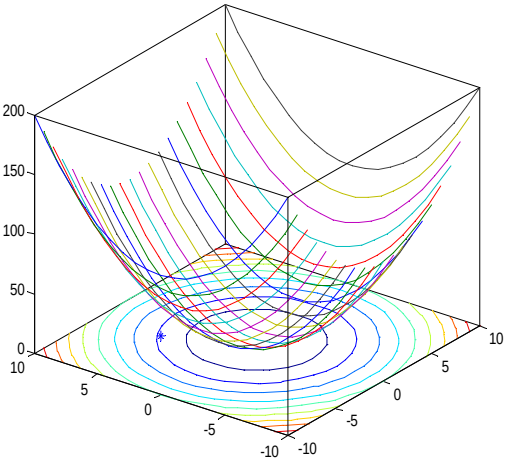

Fig. 5. Offspring is produced in the first iteration

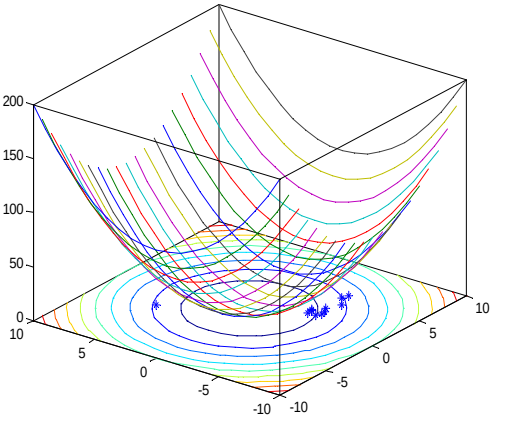

Fig. 6. Offsprings are produced until iteration 10

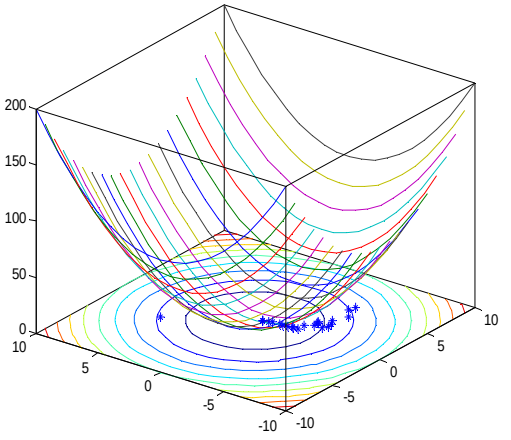

Fig. 7. Offsprings are produced until iteration 25

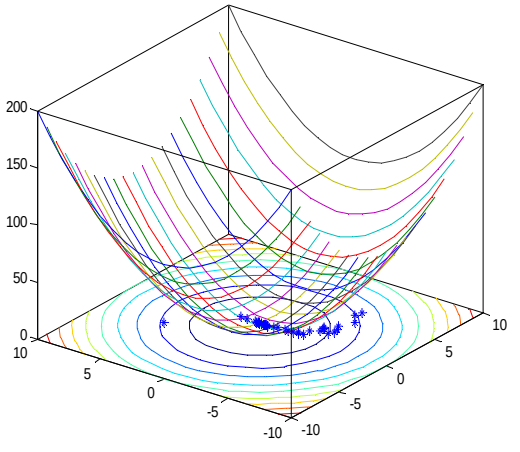

Fig. 8. Offsprings are produced until iteration 50

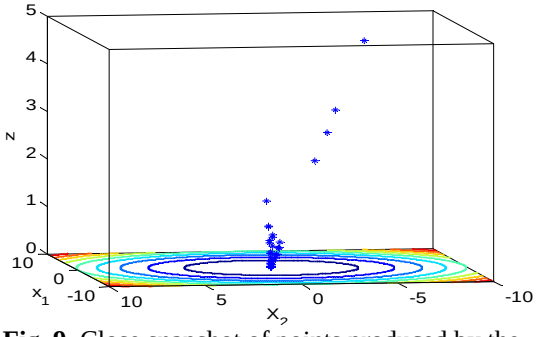

Fig. 9. Close snapshot of points produced by the proposed algorithm

It can be seen that the points move toward the center of the cost function, which is the global minimum. Step sizes are random values determined by the strategy parameter values. The curve cost versus iteration is drawn in Fig. (10).

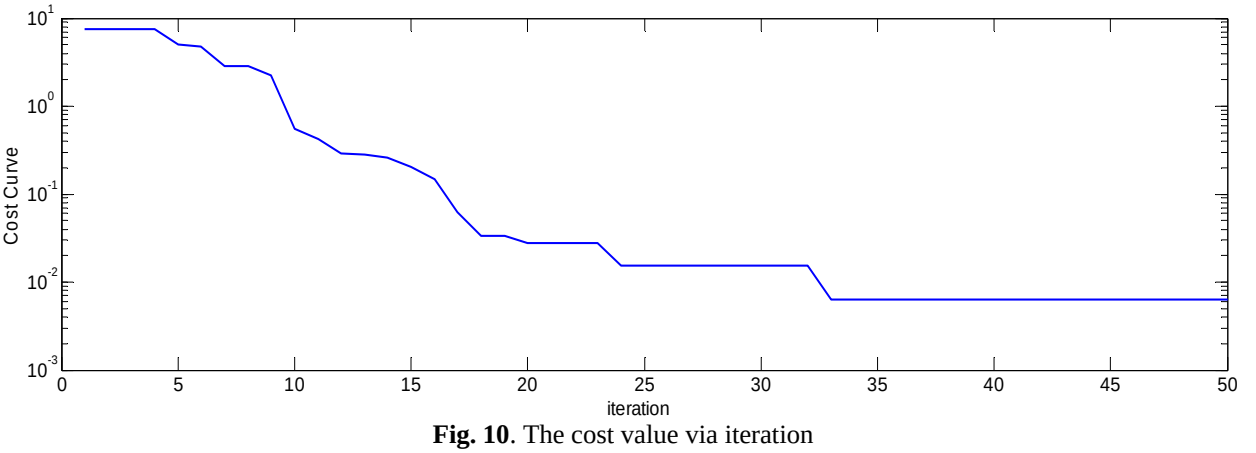

Fig. 10. The cost value via iteration

This cost function is a convert cost function which has a single minimum. This single minimum is also the global Minimum. In the following, the proposed algorithm for optimizing non-convex cost function is used in practical applications.

2.1. Offline PI Controller Design

In this section, the proposed algorithm is used to design a PI controller for the CSTR benchmark system by minimizing the following cost function:

$$z(N)=\sum_{k=1}^{N}\left[(r-y(k))^2+\nabla u(k)^2\right] \quad (33)$$

The study example is a CSTR with a first-order exothermic reaction provided in [20]. It is a typical chemical engineering process which is intensively studied in the control and system identification areas. The dynamic behaviour for this CSTR can be described by using the following nondimensional normalized equations [19]:

$$\dot{x}_1=-x_1+D_a(1-x_1)e^{\frac{x_2}{1+x_2/\lambda}}$$
$$\dot{x}_2=-x_2+BD_a(1-x_1)e^{\frac{x_2}{1+x_2/\lambda}}-\beta(x_2-x_c) \quad (34)$$

x_1 and x_2 are the reactor's dimensionless concentration and its temperature, respectively. The case study under consideration is a regulation of temperature x_2. Coolant temperature x_c is the manipulated variable. One set of parameter values B = 1.0, β = 0.3, λ = 20.0 and D_a = 0.072, which yields an open-loop system with a single stable steady state for all fixed values of the input, is selected in [0 23] (ref. [20]). The detailed nomenclature for this exothermic CSTR can be found in [21].

The input real value range is [0 23]. The starting situation is a stable steady situation with the initial states x_1 = 0.6219 and x_2 = 3.7092 and input u_t = 14. The Output of the CSTR system is the reactor temperature $y(t) = x_2(t)$ and the control goal is to regulate the output in reference value $r = 3$.

The simulation parameters for GMEP algorithms are shown in table 1. The parameters of the GMEP are the same as [15].

Table. 1 . Parameters of GMEP

Population size	20
Tournament size	5
Number of repetitions	20
Number of iterations	1000
Range bound of variables	[-100,100]

The proposed algorithm is used to tune two variables of the PI controller. Fig. 11 shows the cost curve corresponding to the best function value found in the last generation. This figure shows that the GMEP could find the acceptable network weights in a few number of iterations. It admits the capability of this method in fast training of the PI controller. Fig. 12 shows the points generated by the proposed algorithm, and corresponding cost values in the range [-10, 10].

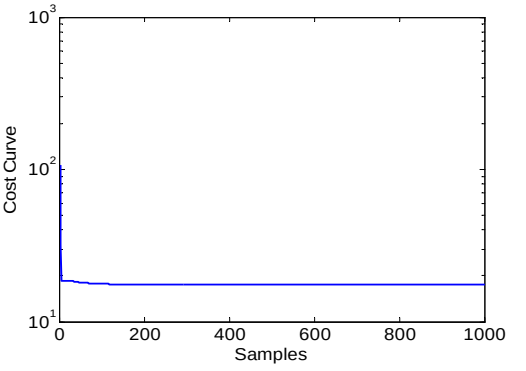

Fig. 11. The cost curve corresponded to the best function value found in the last generation

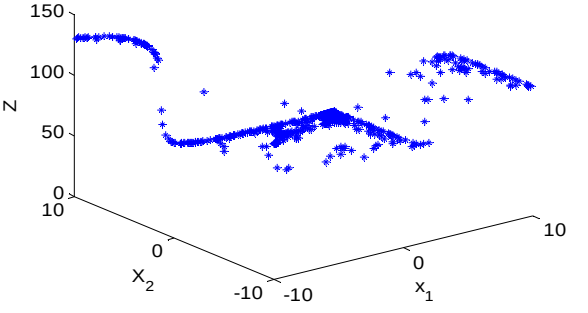

Fig. 12. Points generated by the proposed algorithm

The best found variables of the controller are as follows:

$$[K_p\ K_I]=[12.4518\ \ 19.9107] \quad (35)$$

Step response and control signal for the controlled system with the optimal values (35) can be seen in Fig. (13).

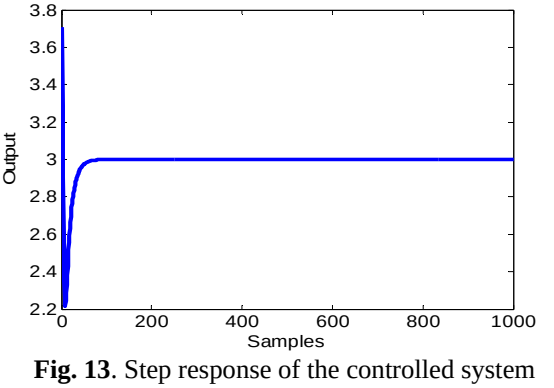

Fig. 13. Step response of the controlled system

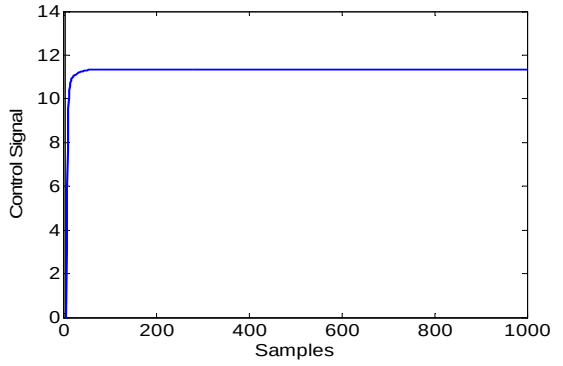

Fig. 14. Control signal of the controlled system

Above the controller was tuned offline within assuming that the cost function and system model are unknown. In the remaining sections, the new strategy for designing an online controller for the nonlinear system described above will be introduced.

3. ONLINE APPROACH FOR DESIGNIG A CONTROLLER USING THE PROPOSED ALGORITHM

The main challenge for the use of intelligent algorithms in online applications is the definition of population. At each step of the algorithm, new members should be produced equal to the number of members in the generation (Fig. 2), and each new point should be tested on the system to reveal its cost. In online optimization, such a procedure may not be available. To fulfill this problem, the online version of the proposed algorithm is as follows:

1) Generate the initial population of μ individuals, and set $k = 1$. Each individual is taken as a pair of real valued vectors, $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$, where the x_i 's are the objective variables and the η_i 's are the standard deviations for Gaussian mutations (also known as strategy parameters in self-adaptive evolutionary algorithms).
2) Evaluate the fitness score for each individual $(x_i, \eta_i), \forall i \in \{1,....,\mu\}$ of the population, based on the objective function $f(x_i)$.
3) Select the best parent and *m-1* individuals randomly for estimating the gradient function, and then the best parent creates a single offspring by the following: for $j = 1,...,n$

$$x_j^{i+m+1} = x_j^{i+m} - \eta_j^{i+m} \left| N(0,1) \right| \operatorname{sgn}\left(\frac{\alpha_1 z^{i+m} + \alpha_2 z^{i+m-1} + \ldots + \alpha_m z^{i}}{\alpha_1 x^{i+m} + \alpha_2 x^{i+m-1} + \ldots + \alpha_m x^{i}} \right) \quad (31)$$

$$\eta_j^{i+m+1}(j) = \left| \eta_j^{i+m}(j) \exp(\tau' N(0,1) + \tau N_j(0,1)) \right| \quad (32)$$

where x_j^{i+m} and η_j^{i+m} denote the j–th component of the vectors x^{i+m} and η^{i+m}, respectively. x^{i+m} is the parent and x^{i+r}, $r = 0,\ldots,m-1$, are the points are selected for estimating the gradient function, z^{i+r} are corresponding cost value.

4) Calculate the fitness of the offspring.
5) Select the worst parent in the population and compare its cost with the cost of the new offspring. If the cost of the new offspring is better than the parent, then substitute it in place of the worst parent in the population.
6) Stop if the halting criterion is satisfied; otherwise, $k = k + 1$ and go to Step 3.

In the proposed algorithm, just a new point is generated in each iteration instead of the population. Therefore, it can be used in online applications. Nevertheless, the required population for the algorithm is obtained by collecting optimum points during algorithm runtime. Therefore, the algorithm is a population-based search algorithm. For the online applications, the following flowchart can be followed.

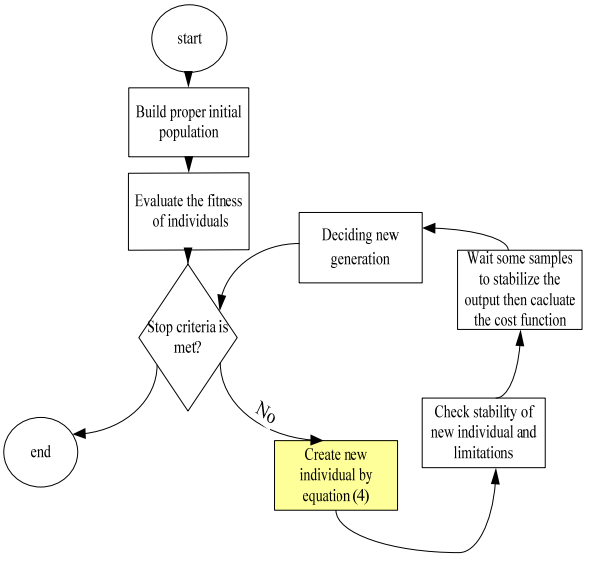

Fig. 15. Flowchart for online application of the proposed algorithm

3.1. Online PI Controller Design

The PI controller for the nonlinear system that was presented in the previous section is designed using the introduced online approach. To do this, the control loop is considered as follows:

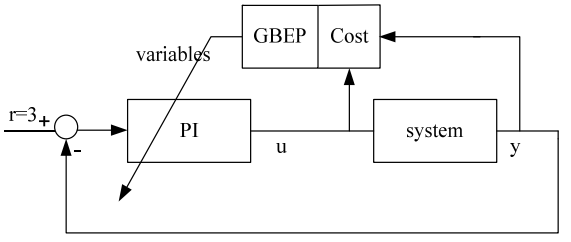

Fig. 16. The control loop diagram for online control of a CSTR system

The controller is updated every 10 seconds (100 samples), and the cost of the new controller is calculated for the 100 samples. Since the system response time is less than 10 seconds, this time is an enough time to calculate the cost. in Fig. 17, the cost curve corresponds to the best cost value found in each iteration and it is drawn for 1000 iterations. The control signal and the output of the system can be seen in Fig. 18.

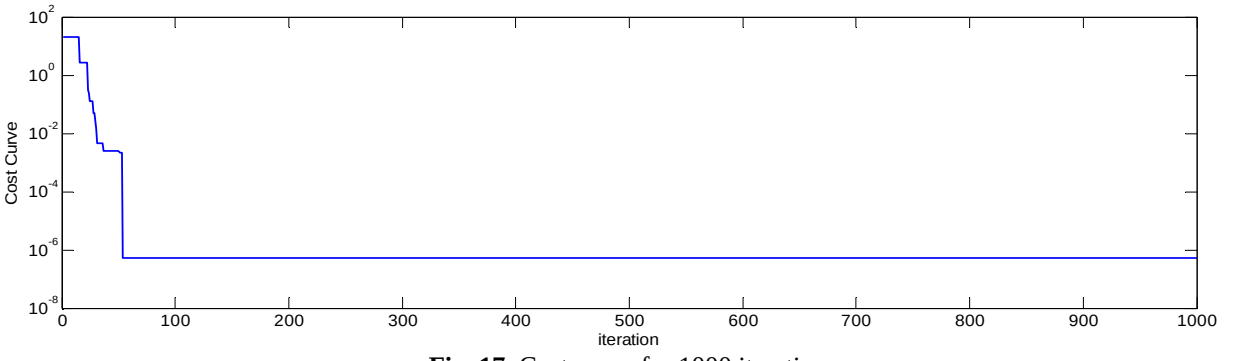

Fig. 17. Cost curve for 1000 iterations

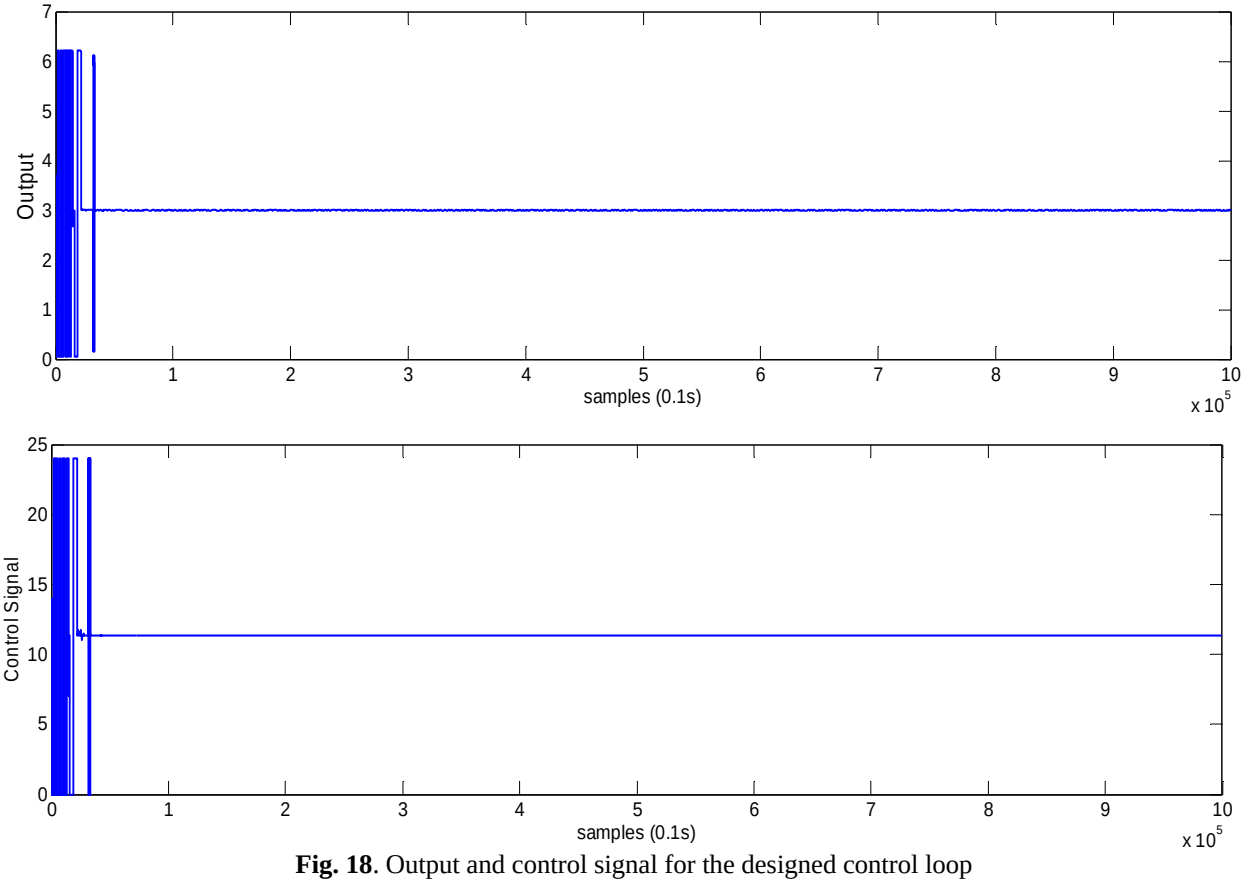

Fig. 18. Output and control signal for the designed control loop

In Fig. (18), it is seen that over time, the output is converging to the reference value. However, it is possible that changes in the parameters of the controller create jumps. These jumps can be observed in many online methods. Besides, the transient (initial) response of the system has some variations. To resolve this problem, the initial population can be chosen to be reasonable (instead of being random). Using the system model (if available) is a good option for modifying the transient situation. However, when there is no information about the system, there is no other choice except to tolerate transient response. It is necessary to note that online design using the proposed algorithm with its convergence condition is not comparable with other intelligent algorithms; besides, there is no classical method that can design the controller without requiring any information about the system. The proposed method does not have any of the above problems. in addition , the cost value (Fig. 17) establishes the optimality of designed controllers.

4. CONCLUSION

In this paper, nonlinear optimization and controller design challenges for nonlinear systems are discussed. This requires an algorithm having the ability to the global search, in addition to satisfying the condition of convergence and stability. Intelligent and classical methods have only one of the above requirements. Therefore, in this paper, the gradient of the cost function is used to obtain the convergence condition. For this purpose, the sign of the gradient function is approximated by the algorithm itself, and does not require any additional information about the cost function. On the other hand, because evolutionary algorithms use the population in searching, they cannot be employed in online applications. In this paper, the proposed algorithm generates only one point in each iteration, so this problem is eliminated. During the design process, the proposed algorithm does not have any information about the system and the cost function. The proposed algorithm is used in online and offline PI controller design for the nonlinear CSTR system.

REFERENCES

[1] B., D. Anderson, J.B., Moore, **"Linear Optimal Control"**, *Prentice-Hall Inc, New Jersey*, 1971.

[2] M.S., Arumugam, M.V., Rao and R., Palaniappan, **"New Hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems",** *Applied Soft Computing* 6, pp.38-52, 2005.

[3] K., Nikolos, **"EVolutionary algorithm based offline/online path planner for UAV navigation"**, *IEEE Transactions on systems, man, and Cybernetics*, Vol. 33, No. 6, pp.898-912, 2003.

[4] P.J., Fleming, R.C., Purshouse, **"Evolutionary algorithm in control systems engineering: a survey"**, *Control Engineering Practice* 10, pp.1223-1241, 2002.

[5] R.L., Haupt, S.E., Haupt, **"Practical Genetic Algorithms"**, *A John Wiley & Sons Inc., second edition,* 2004.

[6] L.J., Fogel**," *Artificial intelligence through simulated evolution",*** *Wiley, New York*, 1966.

[7] I., Rechenberg, **"Evolutionsstrategie: optimierung technischer systeme nach PrinzISien der biologischen evolution"**, *Frommann- Holzboog, Stuttgar*, 1973.

[8] J. H., Holland, **"Adaptation in natural and artificial systems"**, *University of Michigan Press, Ann Harbor*, 1975.

[9] J., Koza, **"Genetic programming: on the programming of computers by means of natural selection"**, *MIT Press, Cambridge*, 1992.

[10] H., Narihisa, K., Kohmoto, T., Taniguchi, M., Ohta and K., Katayama, **"Evolutionary Programming With Only Using Exponential Mutation",** *IEEE Congress on Evolutionary Computations, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada* July 16-21, 2006.

[11] X., Yao, Y., Liu and G., Lin, **"Evolutionary Programming Made Faster"**, *IEEE Trans. on Evolutionary Computation*, Vol. 3,No. 2, 1999.

[12] H., Narihisa, K., Kohmoto and K,. Katayama, **"Evolutionary Programming with Double Exponential Probability Distribution",** *Proc. of The Second International Association of Science and Technology for Development (IASTED) International Conference on Artificial Intelligence and Applications (AIA2002)*, pp.358-363, 2002.

[13] K., Kohmoto, H., Narihisa and K., Katayama, **"Evolutionary Programming Using Exponential Mutation"**, *Proc. of the 6th World Multi conference on Systematics, Cybernetics and Informatics, vol.11, Computer Science* 2, July 14-18, USA, pp.405-410, 2002.

[14] C.Y., Lee, Y., Song, **"Evolutionary Programming using the Levy probability Distribution",** *Proc. of Genetic and Evolutionary Computation Conference (GECCO'99), Morgan Kaufman*, pp.886-893, 1999.

[15] Yo., Alipouri, J., Poshtan, Ya., Alipouri and M.R., Alipour, **"Momentum coefficient for promoting accuracy and convergence speed of evolutionary programming**". *Applied Soft Computing* 12, pp.1765–1786, 2012.

[16] Y., Alipouri, J., Poshtan and Y., Alipouri, **"A modification to classical evolutionary programming by shifting strategy parameters"**, *applied Intelligence*, DOI 10.1007/s10489-012-0364-x, 2012.

[17] H., Einar, R.S., Phillips**," Functional analysis and semi-groups"** *AMS Colloquium Publications, 31, American Mathematical Society*, p.300–327, 1957.

[18] W., Feller**,"An introduction to probability theory and its applications"***, wiley (3rd ed.),* Vol.2, pp. 230–232, 1971.

[19] F.J., Doyle, A., Packard and M., Morari, **"Robust controller design for a nonlinear CSTR"**,

Chemical Engineering Science 44, pp.1929-1947, 1989.

[20] T.D., Knapp, H.M., Budman, **"Robust control design of non-linear processes using empirical state affine models"**, *Int. J. Control* 73 (17), pp. 1525-1535, 2000.

[21] W., Yu, **"Variance Analysis for Nonlinear Systems",** *PHD thesis, Queen's University Kingston, Ontario, Canada* October, 2007.