Chaos Control of Lur'e Like Chaotic System using Backstepping Controller Optimized by Chaotic Particle Swarm Optimization

Reza Gholipour[1], Alireza Khosravi[1], Hamed Mojallali[2] and Jalil Addeh[1]

**[1] Faculty of Electrical and Computer Engineering, Babol (Noushirvani) University of Technology
P.O. Box 47135-484, Babol, Iran**

**[2] Electrical Engineering Department, Faculty of Engineering, University of Guilan
P.O. Box 3756, Rasht, Iran**

Abstract

This paper deals with the design of optimal backstepping controller, by using the chaotic particle swarm optimization (CPSO) algorithm to control of chaos in Lur'e like chaotic system. The backstepping method consists of parameters which could have positive values. The parameters are usually chosen optional by trial and error method. The controlled system provides different behaviors for different values of the parameters. It is necessary to select proper parameters to obtain a good response, because the improper selection of the parameters leads to inappropriate responses or even may lead to instability of the system. The proposed optimal backstepping controller without trial and error determines the parameters of backstepping controller automatically and intelligently by minimizing the Integral of Time multiplied Absolute Error (ITAE) and squared controller output. Finally, the efficiency of the proposed optimal backstepping controller (OBSC) is illustrated by implementing the method on the Lur'e like chaotic system.

Keywords: *Lur'e Like System, Backstepping Method, Logistic Map, Particle Swarm Optimization, control of chaos.*

1. Introduction

Chaotic phenomena can be found in many scientific and engineering fields such as biological systems, electronic circuits, power converters, chemical systems, and so on [1]. Chaotic systems have irregular, complex and unpredictable dynamic behavior. Since the pioneering work of Ott, Grebogi, and Yorke [2] proposed the well-known OGY control method, the control of chaotic systems has been widely studied. Recently, quite a few techniques and approaches have been successfully applied to chaotic motion control under different conditions and requirements [3-5].

The backstepping approach is one of the most popular nonlinear techniques of control design. It is capable of generating a globally asymptotically stabilizing control laws to suppress and synchronize chaotic system [6-8]. The idea of backstepping design is to select recursively some appropriate functions of state variables as pseudo-control inputs for lower dimension subsystems of the overall system.

Each backstepping stage results in a new pseudo-control design, expressed in terms of the pseudo-control designs from preceding design stages. When the procedure is terminated, a feedback design for the true control input results, which achieves the original design objective by virtue of a final Lyapunov function, which is formed by summing up the Lyapunov functions associated with each individual design stage.

Evolutionary algorithms with their heuristic and stochastic properties often suffer from getting stuck in local optima. This common characteristic led to the development of evolutionary computation as an increasingly important field. A GA is a stochastic search procedure based on the mechanics of natural selection, genetics and evolution [9]. Since this type of algorithm simultaneously evaluates many points in the search space, it is more likely to find a global solution to a given problem. PSO describes a solution process in which each particle moves through a multidimensional search space [10]. The particle velocity and position are constantly updated according to the best previous performance of the particle or of the particle's neighbors, as well as the best performance of all particles in the entire population. GAs have demonstrated the ability to reach near-optimal solutions for large problems; however, they may require a long processing time to reach a near-optimal solution. Similarly to GAs, BPSO is also a population-based optimizer. BPSO has a memory, so knowledge of good solutions is retained by all the particles and optimal solutions are found by the swarm particles if they follow the best particle. Unlike GAs, BPSO does not contain any crossover and mutation processes [11]. Hybridization of evolutionary algorithms with local search has been investigated in many studies [12, 13]. Such hybrids are often referred to as memetic algorithms (MA). An MA can be treated as a genetic algorithm coupled with a local search procedure [14]. The shuffled frog leaping algorithm (SFL algorithm) combines the benefits of an MA and the social PSO algorithm. Unlike in MAs and PSO, the population consists of a set of solutions (frogs), which is partitioned into subsets referred to as memeplexes. In the search space, each group performs a local search, and then exchanges information with other groups [15]. Ant-colony optimization algorithms (ACO) were developed by Dorigo et al. Similar to PSO, they evolve not based on genetics but on social behavior. Unlike PSO, the ACO uses ants to find the shortest route between their ant hill and a source of food; ants can deposit pheromone trails whenever they travel as a form of indirect communication [16].

Generating an ideal random sequence is of great importance in the fields of numerical analysis, sampling and heuristic optimization. Recently, a technique which employs chaotic sequences via the chaos approach (chaotic maps) has gained a lot of attention and been widely applied in different areas, such as the chaotic neural network (CNN), chaotic optimization algorithms (COA), nonlinear circuits, DNA computing, and image processing. All of the above-mentioned methods rely on the same pivotal operation, namely the adoption of a chaotic sequence instead of a random sequence, and thereby improve the results due to the unpredictability of the chaotic sequence [17].

Chaos can be described as a bounded nonlinear system with deterministic dynamic behavior that has ergodic and stochastic properties [18]. It is very sensitive to the initial conditions and the parameters used. In other word, cause and effect of chaos are not proportional to the small differences in the initial values. In what is called the ‘‘butterfly effect’’, small variations of an initial variable will result in huge differences in the solutions after some iteration. Mathematically, chaos is random and unpredictable, yet it also possesses an element of regularity. PSO shows a promising performance on nonlinear function optimization and has thus received much attention [19].

However, the performance of the traditional PSO greatly depends on its parameters, and it often suffers the problem of being trapped in local optima [20, 21]. In order to avoid these disadvantages, the chaotic particle swarm optimization (CPSO) method based on the logistic equation has been proposed [21, 22]. Such an algorithm which is known as Chaotic Particle Swarm Optimization (CPSO) is used in this paper in order to determine the optimal backstepping controller parameters.

The backstepping controller consists of positive parameters which are determined optional and by trial and error. The system response is different depending on the choice of these parameters. Improper choice of the parameters causes improper performance and sometimes instability of the system. In this paper, the CPSO algorithm is utilized for determination of proper values of the parameters. In fact, this algorithm determines the parameters of backstepping controller by minimizing the objective function. The objective function, used for controller tuning has been taken as a weighted sum of the Integral of Time multiplied Absolute Error (ITAE) and squared control signal. In the proposed controller, the backstepping method parameters are chosen such that the time response of system states converges to zero in a short time, i.e. the system chaos is controlled faster. Besides, more limited control signal is needed for stabilization of system states and chaos control.

The organization of this article is as follows. Section 2 describes the backstepping method. PSO and CPSO are described in Section 3. The proposed optimal backstepping control design is given in Section 4. In Section 5, simulation results are provided to validate the effectiveness of the proposed method. The conclusions of this paper are drawn in Section 6.

2. Backstepping Method

Backstepping is a recursive procedure, which allows deriving control law for a nonlinear system, associated with appropriate Lyapunov function, which guaranties stability.

Considering the following n-order system with strict-feedback form:

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\
\dot{x}_2 &= f_2(x_1, x_2) + g_2(x_1, x_2)x_3 \\
\dot{x}_3 &= f_3(x_1, x_2, x_3) + g_3(x_1, x_2, x_3)x_4 \\
&. \\
&. \\
\dot{x}_{n-1} &= f_{n-1}(x_1, x_2, x_3, ..., x_{n-1}) + \\
&g_{n-1}(x_1, x_2, x_3, ..., x_{n-1})x_n \\
\dot{x}_n &= f_n(x_1, x_2, x_3, ..., x_n) + g_n(x_1, x_2, x_3, ..., x_n)u
\end{aligned} \tag{1}$$

Where $x \in R^n, u \in R$. With $f_i(0) = 0$ and $g_i(0) \neq 0$ for $i = 1, ..., n$. f_i and g_i are smooth functions and are differentiable.

Step1: Considering the first subsystem of (1), x_2 is taken as a virtual control input and choose:

$$x_2 = \frac{1}{g_1(x_1)}\left(u_1 - f_1(x_1)\right) \tag{2}$$

The first subsystem is changed to be $\dot{x}_1 = u_1$. Choosing $u_1 = -k_1 x_1$ with $k_1 > 0$, the origin of the first subsystem $x_1 = 0$ is asymptotically stable, and the corresponding Lyapunov function is $V_1(x_1) = x_1^2 / 2$, (2) is changed to:

$$x_2 = \phi_1(x_1) = \frac{1}{g_1(x_1)}\left(-k_1 x_1 - f_1(x_1)\right) \tag{3}$$

Step2: Take x_3 as a virtual control input and the (x_1, x_2) subsystem is changed to (5).

$$x_3 = \frac{1}{g_2(x_1, x_2)}\left(u_2 - f_2(x_1, x_2)\right) \tag{4}$$

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\ \dot{x}_2 &= u_2 \end{aligned} \tag{5}$$

Which is in the form of backstepping method, so the control law u_2 is as follow:

$$\begin{aligned} u_2 = -\frac{\partial V_1}{\partial x_1} g_1(x_1) - k_2\left(x_2 - \phi_1(x_1)\right) + \frac{\partial \phi_1}{\partial x_1}[f_1(x_1) \\ + g_1(x_1)x_2] \end{aligned} \tag{6}$$

Where $k_2 > 0$. This control law asymptotically stabilizes $(x_1, x_2) = (0, 0)$ and Lyapunov function is as (7).

$$V_2(x_1, x_2) = V_1(x_1) + \frac{1}{2}\left(x_2 - \phi_1(x_1)\right)^2 \tag{7}$$

Substituting (6) into (4) gives

$$\begin{aligned} x_3 = \phi_2(x_1, x_2) = \frac{1}{g_2(x_1, x_2)}[-\frac{\partial V_1}{\partial x_1} g_1(x_1) - \\ k_2\left(x_2 - \phi_1(x_1)\right) + \frac{\partial \phi_1}{\partial x_1}\left(f_1(x_1) + g_1(x_1)x_2\right) \\ -f_2(x_1, x_2)] \end{aligned} \tag{8}$$

Step 3:
Take x_4 as a virtual control input and the (x_1, x_2, x_3) subsystem is changed to (10).

$$x_4 = \frac{1}{g_3(x_1, x_2)}\left(u_3 - f_3(x_1, x_2, x_3)\right) \tag{9}$$

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\ \dot{x}_2 &= f_2(x_1, x_2) + g_2(x_1, x_2)x_3 \\ \dot{x}_3 &= u_3 \end{aligned} \tag{10}$$

Which is in the form of backstepping method, so the control law u_3 is as follow:

$$\begin{aligned} u_3 = -\frac{\partial V_2}{\partial x_2} g_2(x_1, x_2) - k_3\left(x_3 - \phi_2(x_1, x_2)\right) \\ + \frac{\partial \phi_2}{\partial x_1}\left(f_1(x_1) + g_1(x_1)x_2\right) \\ + \frac{\partial \phi_2}{\partial x_2}\left(f_2(x_1, x_2) + g_2(x_1, x_2)x_3\right) \end{aligned} \tag{11}$$

Where $k_3 > 0$. This control law asymptotically stabilizes $(x_1, x_2, x_3) = (0, 0, 0)$ and Lyapunov function is as (12).

$$V_3(x_1, x_2, x_3) = V_2(x_1, x_2) + \frac{1}{2}\left(x_3 - \phi_2(x_1, x_2)\right)^2 \tag{12}$$

Substituting (11) into (9) gives

$$\begin{aligned} x_4 = \phi_3(x_1, x_2, x_3) = \frac{1}{g_3(x_1, x_2, x_3)}[-\frac{\partial V_2}{\partial x_2} \times \\ g_2(x_1, x_2) - k_3\left(x_3 - \phi_2(x_1, x_2)\right) + \frac{\partial \phi_2}{\partial x_1}(f_1(x_1) + \\ g_1(x_1)x_2) + \frac{\partial \phi_2}{\partial x_2}\left(f_2(x_1, x_2) + g_2(x_1, x_2)x_3\right) - \\ f_3(x_1, x_2, x_3)] \end{aligned} \tag{13}$$

Step n:
Actual control law u where can asymptotically stabilize (1), is as follows:

$$u = \frac{1}{g_n(x_1,...,x_n)}[-\frac{\partial V_{n-1}}{\partial x_{n-1}} g_{n-1}(x_1,...,x_{n-1}) - k_n\left(x_n - \phi_{n-1}(x_1,...,x_{n-1})\right) + \frac{\partial \phi_{n-1}}{\partial x_1}\left(f_1(x_1) + g_1(x_1)x_2\right) + ... + \frac{\partial \phi_{n-1}}{\partial x_{n-1}}\left(f_{n-1}(x_1,...,x_{n-1}) + g_{n-1}(x_1,...,x_{n-1})x_n\right) - f_n(x_1,...,x_n)] \quad (14)$$

Where $k_n > 0$. This control law asymptotically stabilizes $(x_1,...,x_n) = (0,...,0)$ and Lyapunov function is as (15).

$$V_n(x_1,...,x_n) = V_{n-1}(x_1,...,x_{n-1}) + \frac{1}{2}(x_n - \phi_{n-1}(x_1,...,x_{n-1}))^2 \quad (15)$$

3. Optimization Method

3.1 Particle Swarm Optimization (PSO)

In original PSO [10], each particle is analogous to an individual ‘‘fish’’ in a school of fish. It is a population-based optimization technique, where a population is called a swarm. A swarm consists of N particles moving around in a D-dimensional search space. The position of the *i*th particle can be represented by $x_i = (x_{i1}, x_{i2},..., x_{iD})$. The velocity for the *i*th particle can be written as $v_i = (v_{i1}, v_{i2},..., v_{iD})$. Each particle coexists and evolves simultaneously based on knowledge shared with neighboring particles; it makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. The best previously encountered position of the *i*th particle is denoted its individual best position $p_i = (p_{i1}, p_{i2},..., p_{iD})$, a value called $pbest_i$. The best value of the all individual $pbest_i$ values is denoted the global best position $g_i = (g_1, g_2,..., g_D)$ and called $gbest$. The PSO process is initialized with a population of random particles, and the algorithm then executes a search for optimal solutions by continuously updating generations. At each generation, the position and velocity of the *i*th particle are updated by $pbest_i$ and $gbest$ in the swarm. The update equations can be formulated as:

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times r_1 \times (pbest_{id} - x_{id}^{old}) + c_2 \times r_2 \times (gbest_d - x_{id}^{old}) \quad (16)$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \quad (17)$$

r_1 and r_2 are random numbers between (0, 1), and c_1 and c_2 are acceleration constants, which control how far a particle will move in a single generation. Velocities v_{id}^{new} and v_{id}^{old} denote the velocities of the new and old particle, respectively. x_{id}^{old} is the current particle position, and x_{id}^{new} is the new, updated particle position. The inertia weight w controls the impact of the previous velocity of a particle on its current one [23]. In general, the inertia weight is decreased linearly from 0.9 to 0.4 throughout the search process to effectively balance the local and global search abilities of the swarm [24]. The equation for the inertia weight w can be written as:

$$w = (w_{\max} - w_{\min}) \times \frac{Iteration_{\max} - Iteration_i}{Iteration_{\max}} + w_{\min} \quad (18)$$

In Eq. (18), $w_{\max}$ is 0.9, $w_{\min}$ is 0.4 and $Iteration_{\max}$ is the maximum number of allowed iterations.

3.2 Chaotic Particle Swarm Optimization (CPSO)

In the field of engineering, it is well recognized that chaos theory can be applied as a very useful technique in practical application. The chaotic system can be described by a phenomenon, in which a small change in the initial condition will lead to nonlinear change in future behavior, besides that the system exhibits distinct behaviors under different phases, i.e. stable fixed points, periodic oscillations, bifurcations, and ergodicity [25]. Chaos [26] is also a common nonlinear phenomenon with much complexity and is similar to randomness. Chaos is typically highly sensitive to the initial values and thus provides great diversity based on the ergodic property of the chaos phase, which transits every state without repetition in certain ranges. It is generated through a deterministic iteration formula. Due to these characteristics, chaos theory can be applied in optimization.

In PSO, the parameters w, r_1 and r_2 are the key factors affecting the convergence behavior [27, 28]. The inertia weight controls the balance between the global exploration and the local search ability. A large inertia weight favors the global search, while a small inertia weight favors the local search. For this reason, an inertia weight that linearly decreases from 0.9 to 0.4 throughout the search process is usually used [24]. Since logistic maps are frequently used chaotic behavior maps and chaotic sequences can be quickly generated and easily stored, there is no need for storage of long sequences [29]. In CPSO, sequences generated by the logistic map substitute the random parameters r_1 and r_2 in PSO. The parameters r_1 and r_2 are modified by the logistic map based on the following equation.

$$Cr_{(t+1)} = 4 \times Cr_{(t)} \times (1 - Cr_{(t)}) \tag{19}$$

In Eq. (19), $Cr_{(0)}$ is generated randomly for each independent run, with $Cr_{(0)}$ not being equal to {0, 0.25, 0.5, 0.75, 1}. The velocity update equation for CPSO can be formulated as:

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times Cr \times (pbest_{id} - x_{id}^{old}) + c_2 \times (1 - Cr) \times (gbest_d - x_{id}^{old}) \tag{20}$$

In Eq. (20), Cr is a function based on the results of the logistic map with values between 0.0 and 1.0. Fig. 1 shows the chaotic Cr value using a logistic map for 150 iterations where $Cr_{(0)} = 0.0001$. The pseudo-code of CPSO is shown below [17, 22].

CPSO pseudo-code

```
01: begin
02:   Randomly initialize particles swarm
03:   Randomly generate $Cr_{(0)}$
04:   while (number of iterations, or the stopping criterion
      is not met)
05:     Evaluate fitness of particle swarm
06:     for n = 1 to number of particles
07:       Find $pbest$
08:       Find $gbest$
09:       for d = 1 to number of dimension of particle
10:         update the Chaotic Cr value by Eq. (19)
11:         update the position of particles by Eq. (20) and
            Eq. (17)
12:       next d
13:     next n
14:     update the inertia weight value by Eq. (18)
15:   next generation until stopping criterion
16: end
```

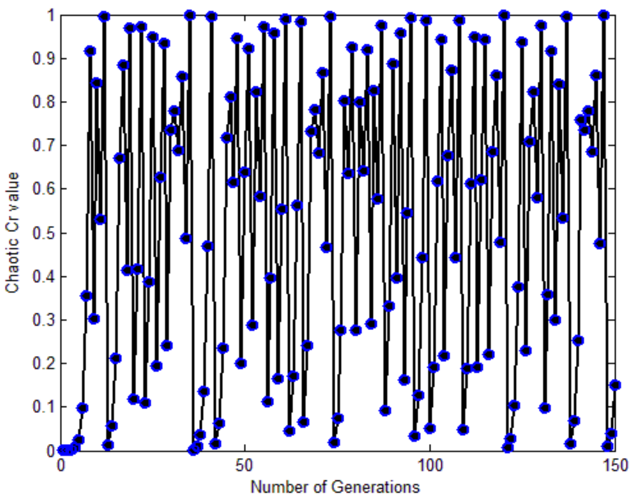

Fig. 1. Chaotic Cr value using a logistic map for 150 generations; Cr(0)=0.0001

In fact, In CPSO, a chaotic map was embedded to determine the PSO parameters r_1 and r_2 . The PSO parameters r_1 and r_2 cannot ensure optimal ergodicity in the search space because they are absolutely random i.e. the r_1 and r_2 are generated by a linear congruential generator (LCG) with a random seed. The generated sequence of LCG consists of pseudo-random numbers that have periodic characteristics. Furthermore, the generated sequence of a logistic map also consists of pseudo-random numbers, but there are no fixed points, periodic orbits, or quasi-periodic orbits in the behavior of the chaos system [17]. As a result, the system can avoid being entrapment in local optima [21]. So we use CPSO, in this paper.

4. Proposed Optimal Backstepping Controller

By adopting the chaotic searching to improve the global searching performance of the particle swarm optimization (PSO), and using the improved PSO (CPSO) to optimize the key parameters of the backstepping controller, an optimal backstepping controller (OBSC) is formed. The structure of the proposed controller is depicted in Fig. 2. The backstepping controller consists of positive parameters which are determined optional and by trial and error. The system response is different depending on the choice of these parameters. Improper choice of the parameters causes improper performance and sometimes instability of the system. In proposed method, the CPSO algorithm is utilized for determination of proper values of

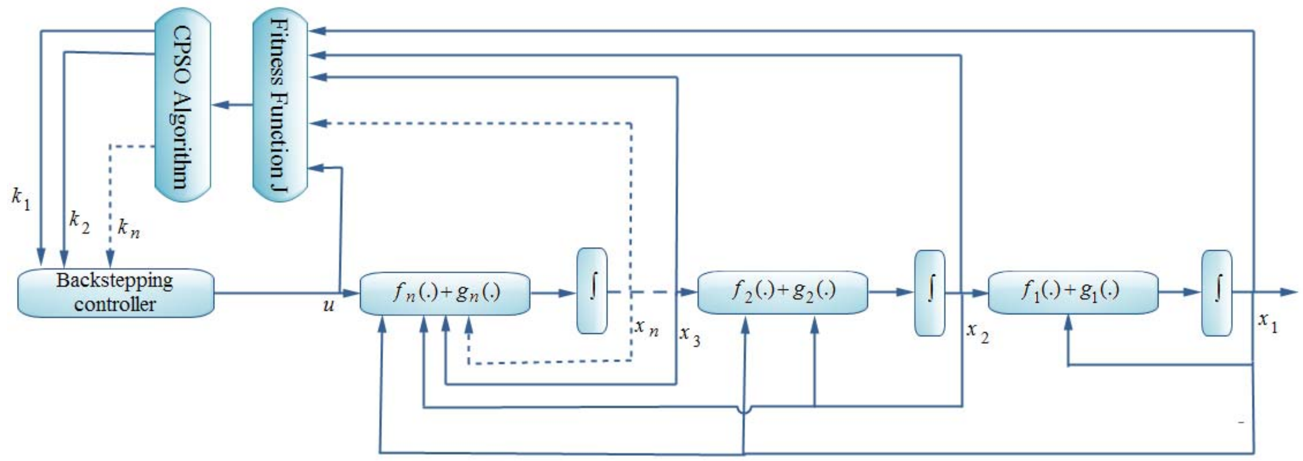

Fig. 2. The proposed optimal backstepping controller structure

the parameters. In fact, this algorithm determines the parameters of backstepping controller by minimizing the objective function. The objective function, used for controller tuning has been taken as a weighted sum of the Integral of Time multiplied Absolute Error (ITAE) and squared controller output similar to that of [30], i.e.

$$\begin{cases} J = \int_0^{t_f} [w_1 t \left|E(t)\right| + w_2 u^2(t)]dt \\ \left|E(t)\right| = \sum_{i=1}^{n} \left|e_i(t)\right| \\ e_i(t) = x_i(t) - x_{di} \end{cases} \tag{21}$$

Where t_f is the final time, in seconds and t is the time, in seconds. x_i is the system state and x_{di} is the favorite mood for x_i. Based on the purpose of system for placing the state at zero value; x_{di} is equal to zero. Also, n represents the system degree, and u is the control signal.

It is worth mentioning that the weights w_1 and w_2 have been introduced in the objective function (21) with a provision of balancing the impact of the error and control signal. In the present simulation study we have considered equal weights for the two objectives to be met by the controller as such the minimization of the error index is as equally important as the control signal is. The objective function J in (21) is now minimized to find out the optimal set of controller parameters which simultaneously reduces the ITAE and control signal $u(t)$. In fact, in the proposed controller, the backstepping method parameters are chosen such that the time response of system states converges to zero in a short time, i.e. the system chaos is controlled faster. Besides, more limited control signal is needed for stabilization of system states and chaos control.

5. Simulation Results

5.1 Lur’e like System

The nonlinear function of Lur’e like system is discontinuous. The dynamic equations of Lur’e like system are as follows [31]

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= ax_1 + bx_2 + cx_3 + d\,sign(x_1) = f(x_1, x_2, x_3) \end{aligned} \tag{22}$$

Where x_1, x_2 and x_3 are the state variables and $a = -1.2$, $b = -0.6$, $c = -1$ and $d = 2$ .For the initial condition $(x_1, x_2, x_3) = (1.5, 1, 0)$, the chaotic motion of the system is illustrated in Figs. 3 and 4.

5.2 Controlling Lur’e like Chaotic System

As shown in Figs. 3 and 4, the system has chaotic behavior when the control input does not apply. In this section, the backstepping method is utilized for the control of chaos of the Lur’e like system. For this purpose, a control signal u is added to the equation (1). The system (1) is rewritten, as following:

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= x_3\\ \dot{x}_3 &= ax_1 + bx_2 + cx_3 + d\,sign(x_1) + u\end{aligned} \tag{23}$$

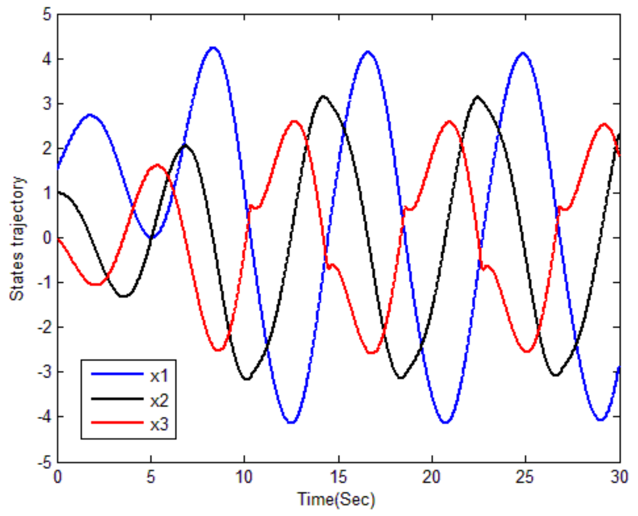

Fig. 3. Chaotic motion of states

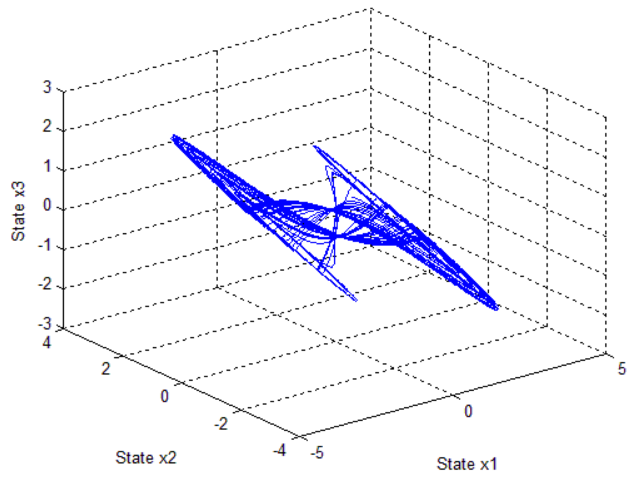

Fig. 4. Chaotic attractor (0-300sec)

Backstepping method is used to set states x_1, x_2, x_3 to the origin point (0, 0, 0) via the control signal u calculated with three steps. According to section (2), the design procedure is as follows:

Step1: x_2 is taken as (26) to construct the Lyapunov function (25) for (24).

$$\dot{x}_1 = x_2 \tag{24}$$

$$V_1(x_1) = x_1^2/2 \tag{25}$$

$$x_2 = \phi_1(x_1) = -k_1 x_1 \tag{26}$$

Step2: Take virtual control input (28) and Lyapunov function (29) for (x_1, x_2) of (27).

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= x_3\end{aligned} \tag{27}$$

$$x_3 = \phi_2(x_1, x_2) = -k_2(x_2 + k_1 x_1) - k_1 x_2 - x_1 \tag{28}$$

$$V_2(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{1}{2}(x_2 + k_1 x_1)^2 \tag{29}$$

Step 3: Final control input and Lyapunov function are given in (31) and (32) for (30).

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= x_3\\ \dot{x}_3 &= ax_1 + bx_2 + cx_3 + d\,sign(x_1) + u\end{aligned} \tag{30}$$

$$\begin{aligned}u =\ &1.2x_1 + 0.6x_2 + x_3 - 2sign(x_1) - k_3\big(x_3 +\\ &(1 + k_1k_2)x_1 + (k_1 + k_2)x_2\big) - (1 + k_1k_2)x_2 -\\ &(k_1 + k_2)x_3 - (x_2 + k_1 x_1)\end{aligned} \tag{31}$$

$$\begin{aligned}V_3(x_1, x_2, x_3) = \frac{1}{2}x_1^2 + \frac{1}{2}(x_2 + k_1x_1)^2 + \frac{1}{2}[x_3 +\\ k_2(x_2 + k_1x_1) + k_1x_2 + x_1]^2\end{aligned} \tag{32}$$

According to equation (31), it is observed that the control signal consists of the parameters which are positive. These parameters have to be chosen properly. Improper selection of the parameters causes improper performance and even instability of the system. Besides, finding the parameters through trial and error is so time-consuming. The CPSO algorithm obtains the proper values of the parameters via minimizing the fitness function. The parameters of the CPSO Algorithm are set as shown in Table 1. The sampling time in this simulation is 0.02.

In proposed controller, The searching ranges for the backstepping parameters k_1, k_2, and k_3 are limited to [0, 10]. The backstepping parameters are obtained for 20 iterations. In this example t_f is equal to 10 seconds.

Table 1: Parameters used in the CPSO

Population size	20
Acceleration constant c_1	2
Acceleration constant c_2	2
Inertia weight w	started from 0.9 and decreased linearly to 0.4
Number of iterations	20

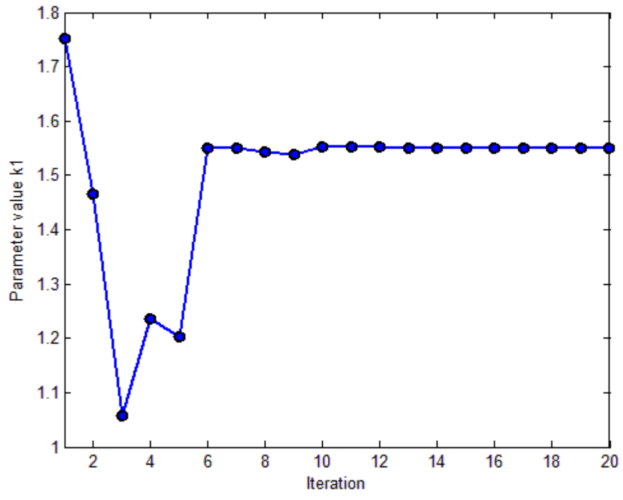
Fig. 5. The parameter value trajectory k1

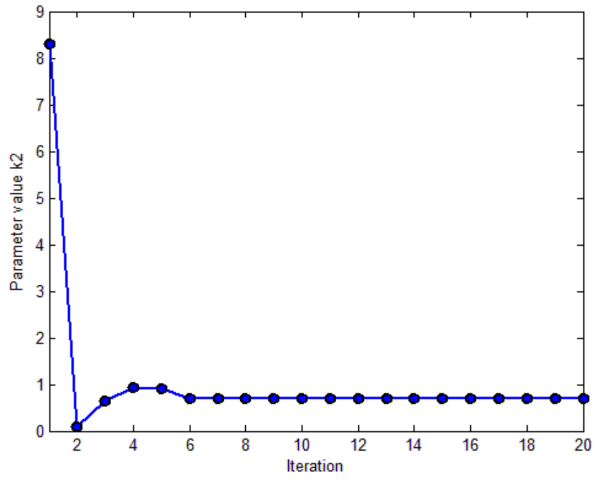
Fig. 6. The parameter value trajectory k2

Besides, the weights w_1 and w_2 of fitness function are chosen as 0.5. n represents the system degree and is equal to 3 in this example.

The parameters of backstepping controller are obtained by using CPSO algorithm, as follows: $k_1 = 1.5512$, $k_2 = 0.6958$, $k_3 = 0.1$. The search process of CPSO algorithm for finding the parameters is shown in Figs. 5-7.

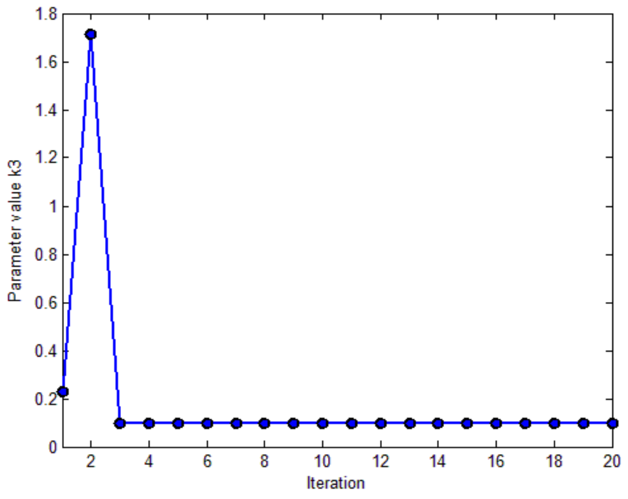
Fig. 7. The parameter value trajectory k3

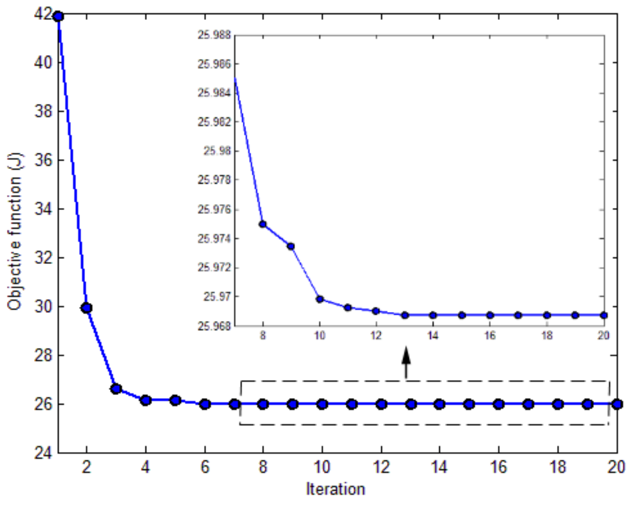
Fig. 8. The Objective function trajectory

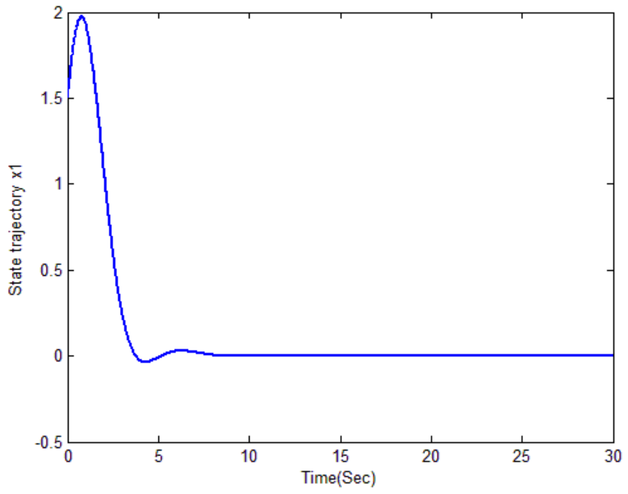
Fig. 9. Controlled time response of the state x_1

Besides, the fitness value obtained by the algorithm is 25.9687. The trajectory of fitness variations with respect to algorithm iteration is shown in Fig. 8.

The time response of the states of Lur'e like system after applying the controller is shown in Figs. 9-11. The controlled chaos of the system is demonstrated in Fig. 12. Also, the control signal is illustrated in Fig. 13.

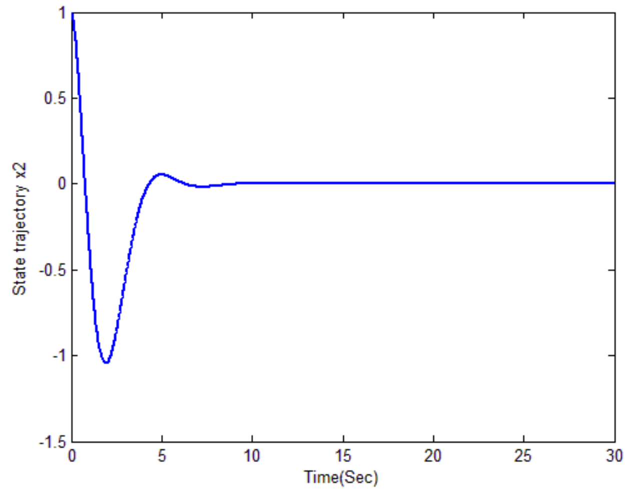

Fig. 10. Controlled time response of the state x_2

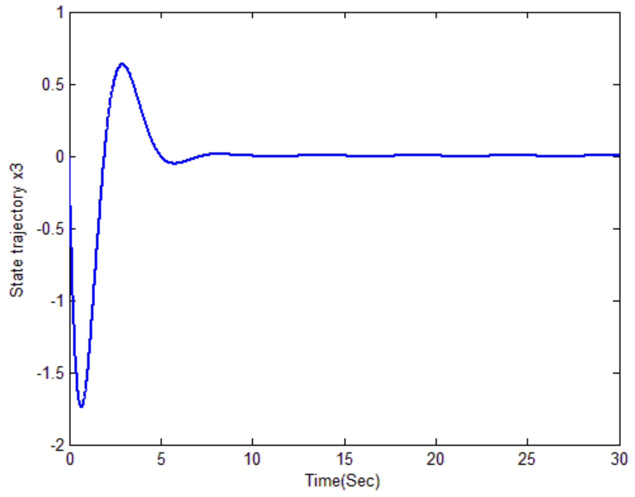

Fig. 11. Controlled time response of the state x_3

As illustrated in Figs. 9-12, the CPSO algorithm causes the time response of the states of the system converge to zero in a short time by minimizing the fitness function. Whereas, if the values of the parameters were chosen by trial and error, the time response of the system states converged to zero much slowly. In fact, the optimal backstepping controller causes the system states become stable in a shorter time and in consequence, the system chaos is controlled in much shorter time. In addition, according to Fig. 13, it is observed that the proposed controller has created a limited control signal to chaos control of Lur'e like system and it is because, in the proposed objective function, the control effort is applied. And by minimizing the objective function, the saturation of control signal is avoided. In fact, the CPSO algorithm via the minimization of the objective function, causes the system states become stable in a shorter time, i.e. the system chaos is controlled in a very short time and also, a lower control signal is needed to control of chaos. And in fact, the saturation of control signal is avoided. Fast control of chaos in a very short time and having more limited control signal for this purpose, are the great advantages of the proposed controller.

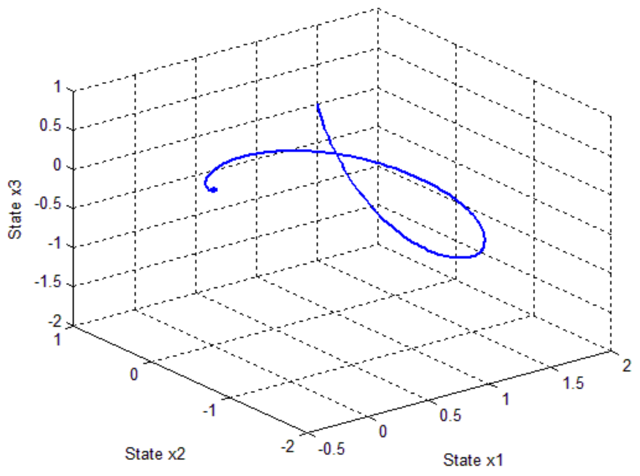

Fig. 12. Controlled Chaotic attractor (0-300sec)

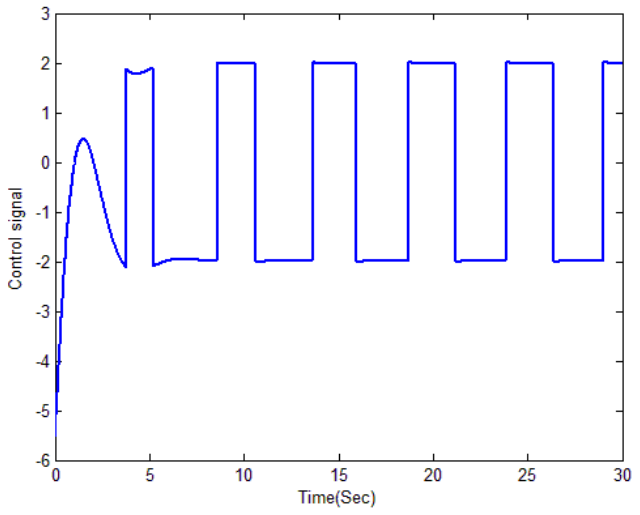

Fig. 13. The control law u

6. Conclusion

This paper has proposed an optimal backstepping controller (OBSC) for chaos control of the Lur'e like chaotic system. The backstepping controller consists of parameters with positive values. The system responds differently for each value of the parameters. The improper choice of the parameters causes an improper behavior which may cause serious problems such as instability of system. In the optimal backstepping controller, the parameters of backstepping controller are determined automatically and intelligently by CPSO algorithm without trial and error. A weighted sum of the Integral of Time multiplied Absolute Error (ITAE) and squared control signal is the minimized fitness function via the CPSO algorithm. In fact, the CPSO optimizes the controller to obtain optimal and proper values for the parameters. In the proposed controller, the backstepping method parameters are chosen such that the time response of system states converges to zero faster, i.e. the system chaos is controlled in a short time. Besides, more limited control signal is needed for stabilization of system states and chaos control. Numerical simulations are presented to show the effectiveness of the proposed scheme.

References

[1] G. Chen, "Controlling chaos and bifurcations in engineering systems," Boca Raton, FL: CRC Press; 1999.

[2] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Phys Rev Lett*, vol. 64, pp. 1196–9, 1990.

[3] J. Wang, G. D. Qiao, and B. Deng , "H1 variable universe adaptive fuzzy control for chaotic system," *Chaos, Solitons & Fractals*, vol. 24, pp. 1075–86, 2005.

[4] C. Hua, X. Guan,and P. Shi, "Adaptive feedback control for a class of chaotic systems,"*Chaos, Solitons & Fractals*, vol. 23, pp. 757–65, 2005.

[5] H. Guo, S. Lin, and J. Liu, "A radial basis function sliding mode controller for chaotic Lorenz system," *Phys Lett A*, vol. 351, pp. 257- 61, 2006.

[6] J. H. Park, "Synchronization of Genesio chaotic system via backstepping approach," *Chaos, Solitons & Fractals*, vol. 27, pp. 1369–75, 2006.

[7] S. Bowong, F. M. Moukam Kakmeni, "Chaos control of uncertain chaotic systems via backstepping approach," *ASME J Vibrat Acoust*, vol. 128, pp. 21–7, 2006.

[8] M. T. Yassen, "Chaos control of chaotic dynamical systems using backstepping design," *Chaos, Solitons & Fractals*, vol. 27, pp. 537–48, 2006.

[9] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[10] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neural Networks 4, Perth, Australia*, pp. 1942–1948, 1995.

[11] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms, " *Advanced Engineering Informatics*, vol. 19, pp. 43–53, 2005.

[12] Y-T. Kao, and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing*, vol. 8, pp. 849–857, 2008.

[13] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, pp. 997–1006. 2004.

[14] K. Sorensen, and M. Sevaux, "MAPM: memetic algorithms with population management," *Computers & Operations Research*, vol. 33, pp. 1214–1225, 2006.

[15] M. M. Eusuff, and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Plan Manage*, vol. 129, pp. 210–225, 2003.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, pp. 29–41, 2002.

[17] L.-Y. Chuang, S.-W. Tsai, and C.-H. Yang, "Chaotic catfish particle swarm optimization for solving global numerical optimization problems," *applied mathematics and computation*, vol. 217, pp. 6900-6916, 2011.

[18] H. G. Schuster, Deterministic chaos an introduction, Second revised ed., Physick-Verlag GmnH, Weinheim, Federal Republic of Germany, 1988.

[19] Y. Liu, Z. Qin, Z. Shi, and J. Lu, "Center particle swarm optimization, *Neurocomputing*, vol. 70, pp. 672–679, 2007.

[20] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," *Evolutionary programming*, vol. VII. Springer, pp. 601–10, 1998.

[21] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons and Fractals*, vol. 25, pp. 1261–1271, 2005.

[22] L.-Y. Chuang, C.-J. Hsiao, and C.-H. Yang, "Chaotic particle swarm optimization for data clustering," Expert systems with Applications, vol. 38, pp. 14555-14563, 2011.

[23] Y. Shi, and R.C. Eberhart, "A modified particle swarm optimizer," in: *Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, AK*, pp. 69–73, 2002.

[24] Y. Shi, and R. C. Eberhart, "Empirical study of particle swarm optimization," in: *Proceedings of Congress on Evolutionary Computation, Washington, DC*, pp. 1945-1949, 2002.

[25] L. dos Santos Coelho, and B. M. Herrera, "Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system," *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 3234–3245, 2007.

[26] H. Lu, H. M. Zhang, and L. H. Ma, "A new optimization algorithm based on chaos," *Journal of Zhejiang University Science A*, vol. 7, pp. 539–542, 2006.

[27] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, pp. 317– 325, 2003.

[28] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Transactions on Power Systems*, vol. 18, pp. 60–68, 2003.

[29] H. Gao, Y. Zhang, S. Liang, and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solitons and Fractals*, vol. 29, pp. 393–399, 2006.

[30] I. Pan, S. Das, and A. Gupta, "Tuning of an optimal fuzzy PID controller with stochastic algorithms for networked control systems with random time delay," *ISA Transactions*, vol. 50, pp. 28-36, 2011.

[31] M. Feki, "Observer-based exact synchronization of ideal and mismatched chaotic systems," *Physics Letters A*, vol. 309, pp. 53–60, 2003.

Reza Gholipour is a M.S. student of control engineering at the Babol (Noushirvani) University of Technology, Iran. His research interests are in the fields of nonlinear control, intelligent control, system identification, soft computing, chaos and control of chaotic systems.

Alireza Khosravi received the Ph.D. degree in Control Engineering from Iran University of Science and Technology (IUST), Iran, in 2008. He is currently assistant professor at Electrical Engineering Department, Babol (Noushirvani) University of Technology, Babol, Iran. His research interests include robust and optimal control, modeling and system identification and intelligent systems.

Hamed Mojallali was born in Fouman, Iran, in 1974. He Received the Ph.D. degree in Control Engineering from Iran University of Science and Technology (IUST), Iran, in 2006. He is currently assistant professor at Electrical Engineering Department, University of Guilan, Rasht, Iran. His current research activities include modeling and system identification, evolutionary algorithms and hybrid systems.

Jalil Addeh is a M.S. student of control engineering at the Babol (Noushirvani) University of Technology, Iran. His research interests are in the fields of nonlinear control, soft computing and pattern recognition.