On maximizing positive Lyapunov exponents in a chaotic oscillator with heuristics

L.G. de la Fraga
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional,
Computer Science Department,
Av. IPN 2508, 07360 México City, México,
e-mail: fraga@cs.cinvestav.mx.

E. Tlelo-Cuautle and V.H. Carbajal-Gómez
Instituto Nacional de Astrofísica, Óptica y Electrónica, Electronics Department,
Luis Enrique Erro 1, Tonantzintla, Puebla, 72840, México.

J.M. Muñoz-Pacheco
Benemérita Universidad Autonóma de Puebla, Facultad de Ciencias de la Electrónica,
Ciudad Universitaria, Av. San Claudio y 18 Sur, 72570 Puebla, México.

Recibido el 1 de febrero de 2012; aceptado el 9 de abril de 2012

A positive Lyapunov exponent indicates the presence of chaos in a dynamical system. In this manner, computing its maximum value guarantees the unpredictability grade of a chaotic system. In this investigation we present the application and comparison of two heuristics: Differential Evolution (DE) and Particle Swarm Optimization (PSO), to maximize the positive Lyapunov exponent in a multi-scroll chaotic oscillator based on saturated nonlinear function series. The computed results show that DE and PSO algorithms are suitable to maximize the positive Lyapunov exponent of truncated coefficients over the continuous spaces. In addition, the phase diagrams show that for a small positive Lyapunov exponent the attractors are well defined, while for its maximum value, the attractors are not well appreciated because the unpredictability grade of the chaotic oscillator is increased.

Keywords: Chaotic oscillator; Multi-scroll attractor; Lyapunov exponent; Saturated function series; PWL function; Evolutionary algorithms.

Un exponente positivo de Lyapunov indica la presencia de caos en un sistema dinámico. De esta manera, el cálculo de un valor máximo garantiza el grado de impredicibilidad de un sistema caótico. En esta investigación presentamos la aplicación y comparación de dos heurísticas: evolución diferencial (DE) y optimización por enjambre de particulas (PSO), para maximizar el exponente positivo de Lyapunov en un oscilador caótico de múltiples enrollamientos basado en series de funciones saturadas. Los resultados calculados muestran que DE y PSO son adecuados para maximizar el exponente positivo de coeficientes truncados sobre espacios continuos. Adicionalmente, los diagramas de fase muestran que para un exponente positivo de Lyapunov pequeño los atractores estan bien definidos, mientras que para su valor máximo, los atractores no se aprecian bien porque el grado de impredicibilidad del oscilador caótico está aumentado.

Descriptores: Oscilador caótico; atractor de múltiples enrollamientos; exponente de Lyapunov; serie de funciones saturadas; función PWL; algoritmos evolutivos.

PACS: 05.45.Pq; 05.45.Pq; 84.30.Ng; 07.50.Ek; 84.30.-r; 01.50.Pa

1. Introduction

Several multi-scroll chaotic oscillators [1-3], can be modeled by piecewise-linear (PWL) approaches [4], so that the nonlinear dynamical problem is transformed to a linear one. However, some of the research challenges in chaotic oscillators are, for example: how to understand when a deterministic dynamical system might exhibit chaotic behavior, the required conditions of this behavior [1-3], the ways available to control it [5], the ways to implement it with electronic devices [6-8], and the practical and theoretical implications that follow. For instance, the Lyapunov exponents provide a means of ascertaining whether the behavior of a system is chaotic. In this manner, the presence of at least one positive Lyapunov exponent in a dynamical system has often been taken as a confirmation of chaotic motion [4].

A large positive Lyapunov exponent indicates a large increase in the degree of unpredictability of a dynamical system. Henceforth, in this investigation we present the application and comparison of two heuristics: Differential Evolution (DE) and Particle Swarm Optimization (PSO) to maximize the positive Lyapunov exponent in a multi-scroll chaotic oscillator based on saturated nonlinear function series. The Lyapunov exponents are computed for the chaotic oscillator from two to six scrolls, and with the same execution conditions for DE and PSO. Drawing the phase diagrams of the multi-scroll chaotic oscillator highlights the results provided by these evolutionary algorithms. From the preliminary work presented in Ref. 4, we also provide a figure to observe the Kaplan-Yorke dimension of the attractors optimized with DE, PSO and with constant coefficients of 0.7. Finally, a short discussion on the behavior of DE and PSO to maximize the positive Lyapunov exponent, is provided.

2. Multi-Scroll Chaotic Oscillator

A multi-scroll chaotic oscillator can be described by the system of differential equations given in (1) [1,3], where a, b, c, and d_1 are positive constants which can have values in the interval $[0, 1]$. The system is controlled by the PWL approximation, *e.g.* series of a saturated function f,

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -ax - by - cz - d_1 f(x; m) \end{aligned} \tag{1}$$

Now, it will be described in detail how the saturated function f in (1) is obtained. Let f_0 be the saturated function:

$$f_0(x; m) = \begin{cases} 1, & \text{if} \quad x > m \\ \frac{x}{m}, & \text{if} \quad |x| \leq m \\ -1, & \text{if} \quad x < -m, \end{cases} \tag{2}$$

where $1/m$ is the slope of the middle segment and $m > 0$; the upper radial $\{f_0(x; m) = 1 \ | x > m\}$, and the lower radial $\{f_0(x; m) = -1 \ | x < -m\}$ are called *saturated plateaus*, and the segment $\{f_0(x; m) = x/m \mid |x| \leq m\}$ between the two saturated plateaus is called *saturated slope*. Figure 1 shows the phase diagram of the saturated function $f_0(x; m)$.

Lets us consider also the saturated functions f_h and f_{-h} defined as:

$$f_h(x; m, h) = \begin{cases} 2, & \text{if} \quad x > h + m \\ \frac{x-h}{m}, & \text{if} \quad |x - h| \leq m \\ 0, & \text{if} \quad x < h - m, \end{cases} \tag{3}$$

and

$$f_{-h}(x; m, -h) = \begin{cases} 0, & \text{if} \quad x > h + m \\ \frac{x-h}{m}, & \text{if} \quad |x - h| \leq m \\ -2, & \text{if} \quad x < h - m, \end{cases} \tag{4}$$

where h is called the *saturated delay time* and $h > m$. Therefore, a saturated function series for a chaotic oscillator with s scrolls is defined as the function:

$$f(x; m) = \sum_{i=0}^{s-2} f_{2i-s+2}(x; m, 2i - s + 2), \tag{5}$$

where $s > 2$.

For example, using $f = f_0$ in (1) a 2-scrolls chaotic oscillator can be generated; the saturated function series for a 3-scrolls oscillator is

$$f(x; m) = f_{-1}(x; m, -1) + f_1(x; m, 1),$$

and for a 4-scrolls oscillator it will be

$$f(x; m) = f_{-2}(x; m, -2) + f_0(x; m) + f_2(x; m, 2).$$

Both function series are shown in Fig. 2 for $m = 0.2$. Note that the value of h in (3) and (4) represents the center of the saturated slopes.

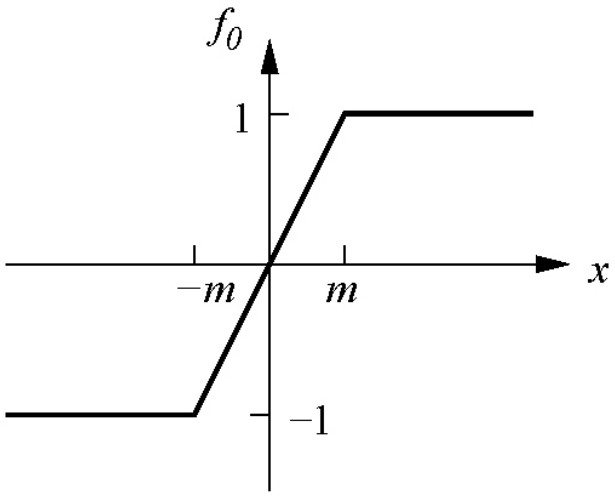
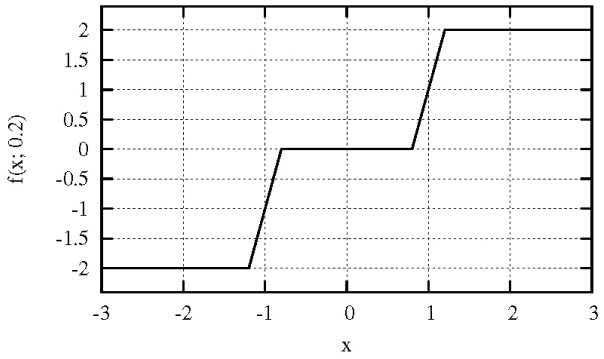
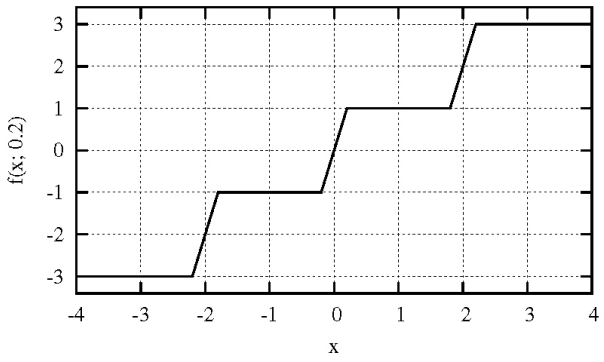


FIGURE 1. Saturated function $f_0(x; m)$.



(a)



(b)

FIGURE 2. Examples of two saturated function series to generate, (a) 3-scrolls and (b) 4-scrolls.

3. Computing Lyapunov Exponents

The deterministic, still unpredictable behavior of nonlinear dissipative dynamical systems is an important subject in more and more fields of science, from mathematics to biology; even in engineering.

The Lyapunov exponents (LEs) give the most characteristic description of the presence of a deterministic nonperiodic flow. Therefore, LEs are asymptotic measures characterizing the average rate of growth (or shrinking) of small perturbations to the solutions of a dynamical system. LEs provide quantitative measures of response sensitivity of a dynamical system to small changes in initial conditions [9]. The number of LE equals the number of state variables, and if at least one is positive, this is an indication of chaos [4,9,10]. That way, we show that DE and PSO algorithms are suitable to maximize the positive LE to guarantee chaotic regime.

In order to measure the LE, the initial state is set to

$$\mathbf{v}_0 \in \mathbb{R}^{12}$$
$$\mathbf{v}_0 = [\mathbf{u}_0^{\mathrm{T}}, \mathbf{e}_1^{\mathrm{T}}, \mathbf{e}_2^{\mathrm{T}}, \mathbf{e}_3^{\mathrm{T}}]^{\mathrm{T}} \tag{6}$$

where $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] = I$, and I is the identity matrix of size 3×3. Thus, $\mathbf{e}_i$, for $i = 1, 2, 3$, are each unitary column vectors of the identity matrix I.

The original dynamical system in (1) is observed by expanding it with other three variational systems. Each one of these added systems will give us the changes in the original dynamical system with respect to three different directions for each of the three state variables. For instance, if $\mathbf{u} = [x, y, z]^{\mathrm{T}}$ represents one state of the system in (1) at any $t > 0$, the state of the new observed system will be $\mathbf{v} = [\mathbf{u}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]^{\mathrm{T}}$.

The observational system is integrated over several steps until a period T_O is reached [4]. After this step, the state of the variational system is orthonormalized by using the standard Gram-Schmidt method [11]. Using the new orthonormalized vectors as initial conditions carries out the next integration.

The exponents measure the long time sensitivity of the flow in $\dot{\mathbf{u}}$ with respect to the initial data $\mathbf{u}_0$ at the directions of every orthonormalized vector. This measure is taken when the variational system is orthonormalized. If $\mathbf{v}$=$[\mathbf{u}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]^{\mathrm{T}}$ is the state after the matrix $[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]$ is orthonormalized, the LE λ_i, for $i = 1, 2, 3$, is calculated by

$$\lambda_i \approx \frac{1}{T} \sum_{j=T_O}^{T} \ln \left\| \mathbf{p}_i \right\|, \tag{7}$$

where T_O is a time step greater than the integration time step used to solve (1).

Since LEs measure the long-term sensitivity, it is not required to calculate them at every integration step. The selection of T_O is made by using the inverse of the minimum absolute value of all the eigenvalues of the system [4], that is:

$$T_O = \frac{1}{\min(\lambda_i)}, \quad \text{for} \quad i = 1, 2, 3. \tag{8}$$

4. DE and PSO

DE and PSO are two evolutionary algorithms, both work with a population of tentative solutions to the problem, and new solutions are generated by combining the old ones and by surviving the ones with better fitness. Both algorithms are used as solvers for global optimization problems, more commonly in problems with a continuous representation.

A global optimization problem can be formulated as to minimize the function

$$f: \mathbb{R}^D \to \mathbb{R}$$
$$f(\mathbf{x}), \;\; s.t. \;\; x_j \in [l_j, u_j], \;\; j = 1, \ldots, D \tag{9}$$

where f is the objective function, and $\mathbf{x}$ is a continuous variable vector of D dimensions. The feasible domain of variable x is defined by specifying upper (u_j) and lower (l_j) limits of each component j.

The usefulness of DE and PSO relies in the fact that they need only the value of function f to work, or in other words, it is not necessary that f be continuous or to get any information about how the derivative of function f is.

DE is one of the most recent population-based stochastic evolutionary optimization techniques [12]. We use the most common version of DE, the rand/1/bin. The pseudocode of DE is shown in Algorithm 1. Each individual is represented

ALGORITHM 1. Differential evolution algorithm (rand/1/bin version).

1:	N is the number of individuals
2:	G is the number of iterations (generations)
3:	Variable bounds $x_i \in [l_i, u_i]$, for $i = 1, 2, \ldots D$
4:	**Procedure** DE (N, G, $\{l_i, u_i\}$)
5:	**for** $i = 1 : N$ **do** ▷ Initialize the population
6:	**for** $d = 1 : D$ **do**
7:	$x_i[d] = l_d + (u_d - l_d) \cdot$ rand()
8:	x_i.fit ← evaluate (x_i)
9:	**for** $i = 1 : G$ **do**
10:	Let j_1, j_2 and j_3 be three random numbers in $\{1, N\}$ without replacement and also different to i.
11:	jrand ← $\lfloor \text{rand}() \cdot D \rfloor + 1$
12:	**for** $d = 1 : D$ **do**
13:	**if** rand()< R OR d = jrand **then**
14:	$y[d] = x_{i2}[d] + F(x_{i0}[d] - x_{i1}[d])$
15:	**if** $y[d] < l_d$ OR $y[d] > u_d$ **then**
16:	$y[d] = l_d + (u_d - l_d) \cdot$ rand()
17:	**else**
18:	$y[d] = x_i[d]$
19:	y.fit = evaluate(y)
20:	**if** y.fit < x_i.fit **then**
21:	x_i ← y; x_i.fit ← y.fit
22:	**search q** = $\mathrm{x}_k \mid \min(\mathrm{x}_k\text{.fit}),$ for $k = 1, 2, \ldots, N.$
23:	**q** is the solution at iteration i

by a vector $\mathbf{x} \in \mathbb{R}^D$, and its fitness value is represented as $\mathbf{x}$.fit. The location j of individual i is represented as $x_i[j]$. rand() is a function that returns a random number greater or equal to zero and less than one. evaluate( ) is a function that calculates the fitness function (function to be optimized).

The core of DE is in the loop on lines 13-18: a new individual is generated from three different individuals randomly chosen; each value of the new vector (a new individual) is calculated from the first father, plus the difference of the other two fathers multiplied by F, the difference constant; the new vector value is calculated if a random real number (between zero and one) is less than R, the DE's recombination constant. To prevent the case when the new individual is equal to the first father, at least one vector's component is forced to be calculated from their fathers values, it is in line 13 of the pseudocode, when $d =$ jrand, and jrand is an integer random number between 1 and D. Then the new individual is evaluated. If it is better than the father (in line 20), then the child replaces its father (line 21).

The pseudocode for PSO is shown in Algorithm 2. Each particle p_i has three associated values: position p_i.pos, velocity p_i.vel, and the value of the fitness function p_i.fit. Particle pbest has only position and fitness function value. gbest[ ] is a vector to store indexes to reference pbest particles. rand() is a function that returns a random number greater or equal to zero and less than one. evaluate( ) is a function that calculates

ALGORITHM 2. Particle swarm optimization algorithm.

1:	N is the number of particles
2:	G is the number of iterations (generations)
3:	Variable bounds $x_i \in [l_i, u_i]$, for $i = 1, 2, \ldots D$
4:	**Procedure PSO** (N, G, $\{l_i\}$, $\{u_i\}$)
5:	**for** $i = 1 : N$ **do** $\triangleright$ Initialize particle's positions
6:	**for** $d = 1 : D$ **do**
7:	p_i.pos$_d$ = l_d + $(u_d - l_d) \cdot$ rand()
8:	pbest$_i$.pos$_d$ $\leftarrow$ p_i.pos$_d$
9:	p_i.fit $\leftarrow$ evaluate(p_i.pos)
10:	pbest$_i$.fit $\leftarrow$ p_i.fit
11:	**for** $i = 1 : N$ **do** $\triangleright$ Initialize particles' velocities
12:	**for** $d = 1 : D$ **do**
13:	vmin = $l_d - p_i$.pos$_d$
14:	vmax = $u_d - p_i$.pos$_d$
15:	p_i.vel$_d$ = vmin + (vmax-vmin)$\cdot$ rand()
16:	**for** $g = 1 : G$ **do** $\triangleright$ Iterate G generations
17:	**for** $i = 1 : N$ **do** $\triangleright$ For each particle
18:	Let j_1, j_2 and j_3 be three random numbers in $\{1, N\}$
19:	gbest$[i] = k \mid \min$(pbest$_k$.fit), for $k \in \{i, j_1, j_2, j_3\}$
20:	**for** $i = 1 : N$ **do** $\triangleright$ For each particle
21:	**for** $d = 1 : D$ $\triangleright$ For each dimension
22:	p_i.pos$_d$ $\leftarrow$ w p_i.vel$_d$ + w$_1$(pbest$_i$.pos$_d$ - p_i.pos$_d$) + w_2 (pbest$_{\text{gbest}[i]}$.pos$_d$ - p_i.pos$_d$)
23:	p_i.vel$_d$ $\leftarrow$ p_i.pos$_d$ + p_i.vel$_d$
24:	**if** p_i.pos$_d < l_d$
25:	p_i.pos$_d$ = l_d; p_i. vel$_d$ = 0
26:	**if** p_i.pos$_d > L_d$ **then**
27:	p_i.pos$_d$ = L_d; p_i.vel$_d$ = 0
28:	f = evaluate(p.pos$_i$)
29:	**if** $f <$ pbest.fit$_i$ **then**
30:	pbest$_i$.pos $\leftarrow$ p.pos$_i$
31:	pbest$_i$.fit $\leftarrow$ f
32:	**search q** = pbest$_k$.pos — $\min$(pbest$_k$.fit), for $k = 1, 2, \ldots, N$.
33:	**q** is the solution at iteration g

TABLE I. Calculated positive Lyapunov exponent and coefficients values from 2 to 6 scrolls.

Scrolls	Lyap. exponent obtained with $a = b = c = d_1 = 0.7$	Maximum Lyap. exp. and (a, b, c, d_1) coef. values for PSO	Maximum Lyap. exp. and (a, b, c, d_1) values for DE
2	0.1257	0.3859	0.3856
		(0.3578, 1.0000, 0.0902, 0.7828)	(0.2336, 0.7564, 0.0746, 0.6363)
3	0.1564	0.4226	0.4063
		(0.4680, 1.0000, 0.0813 0.5206)	(0.3972, 0.8154, 0.0768, 0.4043)
4	0.1748	0.4328	0.4468
		(0.6823, 0.9388, 0.1055, 0.6676)	(0.3779, 0.8635, 0.0669, 0.3722)
5	0.1867	0.4428	0.4421
		(0.9560, 0.7921, 0.1601, 0.9614)	(0.9146, 0.9522, 0.1106, 0.8829)
6	0.1810	0.4548	0.4554
		(0.9978, 0.7254, 0.1297, 0.9999)	(0.7012, 0.9506, 0.0862, 0.6885)

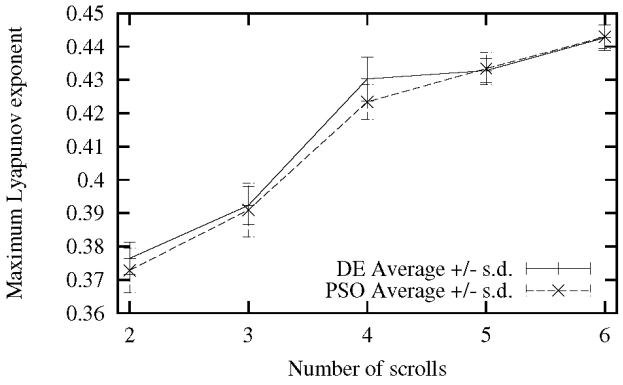


FIGURE 3. Comparison of results obtained by DE and PSO. The average and ± standard deviation of 30 runs of each algorithm are shown.

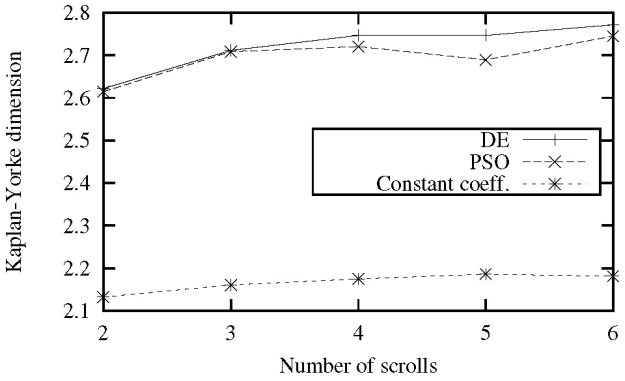


FIGURE 4. Kaplan-Yorke dimension of the attractors optimized with DE, PSO and with constant coefficients of 0.7.

the value function that calculates the value of the fitness for the problem to solve. This PSO version was inspired from [13] and [14,15]. The main advantage of this PSO algorithm (not using extra parameters), consists on having only the essential parameters as the number of individuals and the number of iterations (generations). Particles positions p_i are initialized randomly and also their velocities (in lines 5-10 and 11-15 in Algorithm 2, respectively). Each particle is evaluated and pbest$_i$ particles are initialized equal to the p_i ones.

For a given number of iterations the following process is applied: (1) three random numbers are calculated in $[1, N]$ (N= population size) with replacement; gbest$[i]$ points to the best particle inside this cluster of three particles. (2) A new particle is calculated (line 22 on Algorithm 2) and its velocity is updated (line 23). If this new particle is better than its associated pbest, then pbest particle takes the values of the new particle. The solution after one iteration is searched in the set of pbest particles (lines 32-33 in Algorithm 2).

TABLE II. Average execution time ± standard deviation time (in seconds) of 30 executions of each heuristic against number of scrolls.

No. scrolls	PSO (sec)	DE (sec)
2	178 ± 15	218 ± 13
3	114 ± 16	117 ± 7
4	148 ± 13	170 ± 13
5	102 ± 16	97 ± 8
6	158 ± 14	162 ± 9

5. Maximizing the Positive Lyapunov Exponent

The calculation of the Lyapunov exponents for the saturated nonlinear function series based chaotic oscillator described by (1), can be performed by simply setting: $a = b = c = d_1 = 0.7$ and $m = 0.1$ [2,3].

In most of the work reported using saturated nonlinear function series based chaotic oscillator [2-4], the coefficients

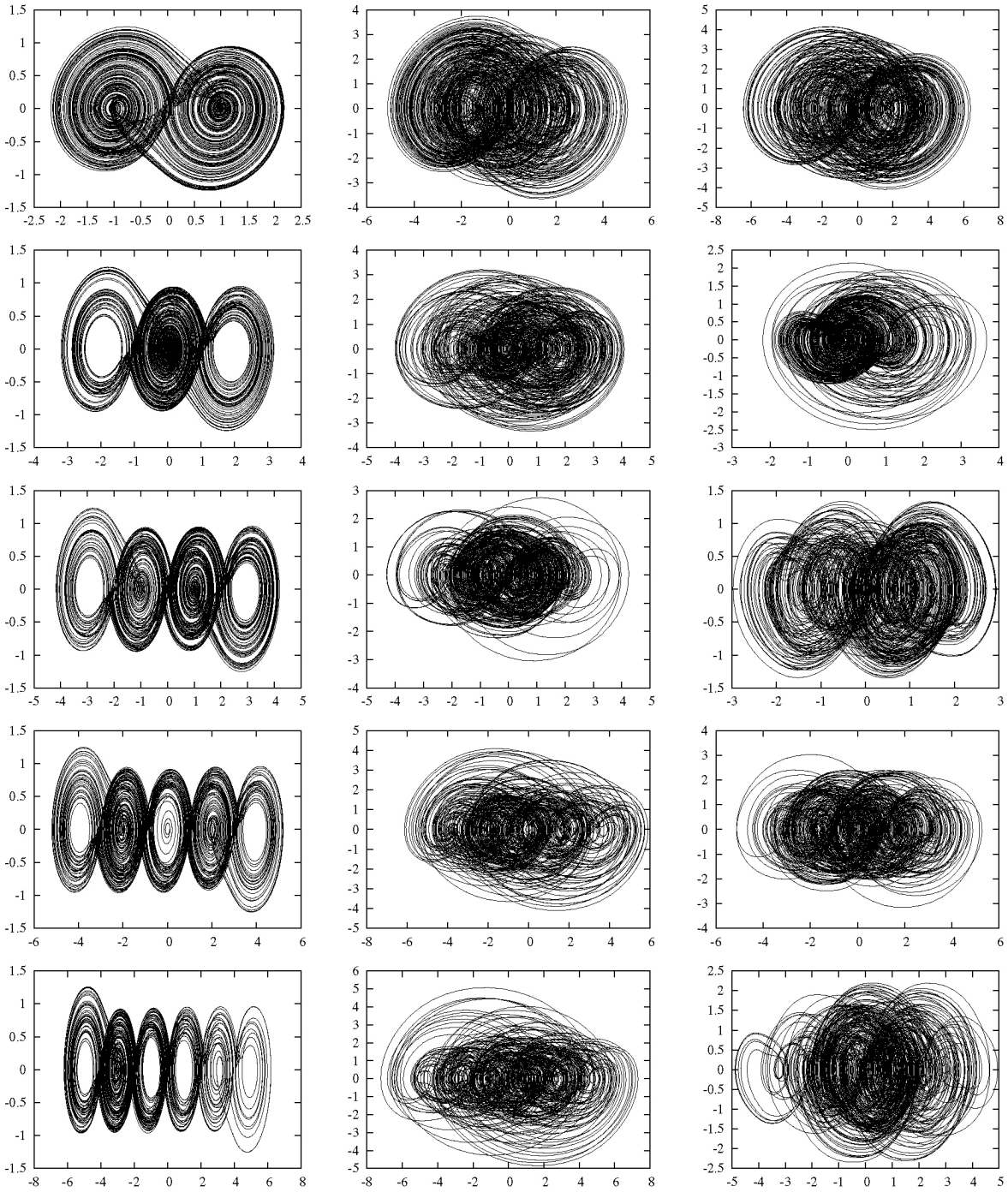


FIGURE 5. Attractors for the maximized Lyapunov exponent: First column with constant coefficients of 0.7, second column is for PSO results, and third column for DE results; row 1 to 5 show the results for 2, 3, 4, 5 and 6 scrolls, respectively.

of the system are fixed to 0.7, but the positive Lyapunov exponent is relatively small. Furthermore, in this investigation we compare this value for the fixed and equal constants of 0.7, with the values obtained by DE and PSO, as shown in Table I.

DE was executed with the value for the recombination constant $R = 8$, and difference constant of $F = 0.6$. For both DE and PSO, a population of 40 individuals was used, and the number of generations was set arbitrarily to 100. The comparison of statistics for 30 executions of both algorithms is provided in Fig. 3.

In Fig. 4 is shown the Kaplan-Yorke dimension (Lyapunov dimension) calculated as [4]:

$$d_L = j + \frac{\lambda_1 + \lambda_2 + \cdots \lambda_j}{|\lambda_{j+1}|}$$

where for our case, $j = 2$ and λ_i are the Lyapunov coefficients ordered in decreasing order. The largest dimension of the optimized results indicates less contraction over the trajectories. The average execution time for 30 executions of DE and PSO are shown in Table II[i]. The execution times showed in Table II are for one run to obtain one solution, *i.e.* the execution of the heuristic with 40 individuals and 100 generations.

Figure 5 shows the phase diagram for the cases listed in Table I. As one sees, the dynamic behavior of the chaotic system is more complex as the positive Lyapunov exponent increases, because it achieves greater unpredictability.

6. Discussion

The search space in our problem can be calculated from the size of a variable: each coefficient in (1) has one significant digit that can be 0 or 1 and four decimal places can have values in $\{0, 9\}$ (each one). Then it is $2 \times 10 \times 10 \times 10 \times 10 = 2 \times 10^4$. For the whole problem, the search space will be $(2 \times 10^4)^4$, it is 16×10^{16}. This huge search space justifies the use of heuristics to solve the problem of computing Lyapunov exponents.

Basically, both algorithms, PSO and DE, give the same result in this application to maximize the positive Lyapunov exponent of the multi-scroll chaotic oscillator with PWL-functions. From the values listed in Table I, the third coefficient has the smaller value. PSO trend to find higher values for the coefficients (second coefficient is 1.0 for the oscillator with 2 and 3 scrolls). PSO produced a greater value for the maximum Lyapunov exponent with 3 scrolls. DE has a better value for the maximum Lyapunov exponent with 4 and 6 scrolls. For 2 and 5 scrolls, the values of the maximum Lyapunov exponent are slightly different (less than 0.0004) but the obtained coefficient values are totally different.

Our implementation was coded in C programming language. The Runge-Kutta method of four order was used to solve (1) and also to calculate the Lyapunov exponents. The integration step was fixed to 0.01. For all simulations the initial condition was set to $\mathbf{x}_0 = [0.1, 0.0, 0.0]^{\mathrm{T}}$.

The execution time is important because PSO and DE heuristics work only with the value of the objective function, they do not require the derivative of the objective function or the objective function be continuous on the search space, but they need to perform lots of evaluations of the objective function. In our problem, we used 40 individuals and 100 generations, this means that $40 \cdot 100 = 4000$ evaluations of the objective function were performed. It is important to mention that the original implementation in MatLab [4], took more than one day for only one execution. According to the values in Tab. II our implementation gives one result in less than 4 minutes.

From the point of view of memory consumption, PSO takes around twice memory than DE, this is because PSO uses normal and pbest particles. From the point of view of execution time, we can consider from times on Table II, that both evolutionary algorithms have the same execution time. Oscillators with 3 and 5 scrolls take less simulation time because there are lots of solutions that are discharged if their output range in x is not in $[-2.1, 2.1]$ for 3 scrolls, and in $[-4.1, 4.1]$ for 5 scrolls.

From the values in Table I we can deduce that the chaotic behavior is multimodal, with several maxima, with respect to the coefficient values: very different coefficients values produce very small changes in the maximum Lyapunov exponent (MLE) (*i.e.*, see result for 2 scrolls in Table I). Also, DE and PSO only search different regions of the multimodal chaotic behavior: for a given set of values for (a, b, c, d_1), the value of the MLE is the same independently of the used algorithm.

Maximizing the positive Lyapunov exponent lead us to implement better chaotic oscillators [6-8], and improved secure communication systems [5,16], for instance.

7. Conclusion

Applying DE and PSO heuristics have maximized the positive Lyapunov exponent of a PWL-function-based multi-scroll chaotic oscillator. The usefulness of DE and PSO relies in the fact that they need only the value of function f to work, *i.e.* it is not necessary any information about how the derivative of function f is.

In this investigation both heuristics give basically the same results, but with very different coefficient values, showing that chaotic behavior is highly multimodal with respect of the coefficient values. The proposed PSO version has the advantage of avoiding the use of any external control parameter. From the computed results, it was observed that coefficient c in (1) is the most sensitive in the dynamical system. As a conclusion, by selecting small values for c and keeping a, b, d_1 with large values, one obtains large positive Lyapunov exponents.

Acknowledgments

This work is partially supported by CONACyT-México under grant 131839-Y.

i. The program was compiled with gcc and -O2 flags, over a SUNz20v machine with two AMD Opteron 248 microprocessors.

1. J. Lu and G. Chen, *International Journal of Bifurcation and Chaos* **16** (2006) 775-858.

2. J. Lu, S. Yu, H. Leung, and G. Chen, *IEEE Transactions on Circuits and Systems I* **56** (2006) 149-165.

3. J.M. Muñoz-Pacheco and E. Tlelo-Cuautle, *Electronic Design Automation of Multi-scroll Chaos Generators*. (Bentham Science Publishers Ltd, USA, 2010).

4. R. Trejo-Guerra, E. Tlelo-Cuautle, J.M. Muñoz-Pacheco, C. Sánchez-López, and C. Cruz-Hernández, *International Journal of Nonlinear Sciences & Numerical Simulation* **11** (2010) 903-910.
5. V.H. Carbajal-Gómez, E. Tlelo-Cuautle, R. Trejo-Guerra, C. Sánchez-López, and J.M. Muñoz-Pacheco, *Nonlinear Science Letters B: Chaos, Fractal and Synchronization* **1** (2011) 37-42.
6. J.M. Muñoz-Pacheco, W. Campos-Lopez, E. Tlelo-Cuautle, and C. Sánchez-López, *Trends in Applied Sciences Research* **7** (2012) 168-174.
7. R. Trejo-Guerra, E. Tlelo-Cuautle, C. Sánchez-López, J.M. Muñoz-Pacheco, and C. Cruz-Hernández, *Rev. Mex. Fis.* **54** (2010) 268-274.
8. R. Trejo-Guerra, E. Tlelo-Cuautle, J.M. Jiménez-Fuentes, J.M. Muñoz-Pacheco, and C. Sánchez-López, *Multiscroll floating gate based integrated chaotic oscillator* (International Journal of Circuit Theory and Applications, 2011).
9. L. Dieci, *Journal of Dynamics and Differential Equations* **14** (2002) 697-717.
10. T.S. Parker and L.O. Chua, *Practical Numerical Algorithms for Chaotic Systems* (Springer-Verlag, NY, 1989).
11. G.H. Golub and C.V. Loan, *Matrix Computations* (The Johns Hopkins University Press, 3rd ed, 1996).
12. R. Storn and K. Price, *Journal of Global Optimization* **11** (1997) 341-359.
13. M. Clerc, *Particle Swarm Optimization* (ISTE Ltd. UK, 2006).
14. M. Clerc, *Standard particle swarm optimisation* (2011-07-13 version). p. 1-14. Available at clerc.maurice.free.fr/pso/SPSO descriptions.pdf.
15. D. Bratton and J. Kennedy. *Defining a standard for particle swarm optimization* (In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, pages 120 -127, april 2007).
16. L. Gamez-Guzman, C. Cruz-Hernández, R.M. Lopez- Gutierrez, and E.E. Garcia-Guerrero, *Rev. Mex. Fis.* **54** (2008) 299-305.