R-chaosoptimiser: an optimiser for unconstrained global nonlinear optimisation written in R language for statistical computing

R-chaosoptimiser: un optimizador global no lineal sin restricciones escrito en lenguaje R para el calculo estadístico

Juan David Velásquez H.[1]

ABSTRACT

This paper discusses using R-chaosoptimiser, an R language package for nonlinear optimisation based on gradient techniques and chaos optimisation algorithms. Its implementation was based on three building blocks which could be executed alone or un combination: the first carrier wave algorithm, the chaos-based cyclical coordinate search method and the second wave carrier algorithm. Using chaos optimisation algorithms allows the tool to break away from local optimal points and converge towards an overall optimum inside a predefined search domain. Within the previous components, a user would be specifying the BFGS algorithm for refining the current best solution. Using the BFGS algorithm is not mandatory, so that its implementation was able to optimise problems having objective function discontinuities. However, the BFGS algorithm is a powerful local search method, meaning that it is used to exploit current knowledge about an objective function for improving a current solution; an explanatory example is presented.

Keywords: optimisation, R language, gradient-based method, chaos, algorithm.

RESUMEN

En este artículo se discute la implementación de rchaosoptimizer, un paquete de R para la optimización no lineal basada en técnicas de gradiente y algoritmos de optimización caóticos. La implementación está basada en tres bloques constructivos que pueden ser ejecutados solos o combinados: 1) el algoritmo de primera onda; 2) el método de búsqueda por coordenadas cíclicas basado en caos; y, 3) el algoritmo de segunda onda. El uso de algoritmos de optimización caóticos permite a la herramienta implementada escapar de puntos óptimos locales y converger al óptimo global dentro del domino predefinido de búsqueda. Dentro de los componentes previos, el usuario podría especificar una llamada al algoritmo BFGS para refinar la solución actual. El uso del algoritmo BFGS no es obligatorio, tal que la implementación es capaz de optimizar problemas con discontinuidades en la función objetivo. Sin embargo, el algoritmo BFGS es un método poderoso de búsqueda local, tal que, él es usado para explotar el conocimiento sobre la función objetivo para mejorar la solución actual. Finalmente, en ejemplo exploratorio es presentado.

Palabras clave: optimización, Lenguaje R, métodos basados en gradiente, caos, algoritmos.

Received: January 25th 2010
Accepted: October 6th 2011

Introduction

Solving nonlinear optimisation complex problems represents an important practical and research field. Developing algorithms and practical implementation have led to applications in mathematics, engineering, economics and computer sciences. Optimisation is usually difficult due to (Pardalos and Resende, 2002):

- Objective function complexity: analytical calculation of derivates is difficult if not impossible; moreover, even though derivates may be available, there are no mathematical formulas for estimating the overall optimum;
- Restrictions imposed on the problem: these are related to a feasible solution's complexity. Optimisation methods may be able to search inside such region and to discard unfeasible solutions;
- The presence of so-called multiple local minima: this is related to the complexity of the surface generated by the objective function, so that it is very difficult to evade local optimal points; and
- The limitations of many optimisation methodologies: or, in other words, how the methods are affected by the previous aspects and in which particular conditions a specific method is able to find good solutions within the search space.

The development, testing and implementation of heuristic-based optimisation algorithms is an important research field, especially for those based on stochastic optimisation (Pardalos and Resende, 2002). Well-known examples would include classical paradigms such as simulated annealing (Kirkpatrick, Gelatt and Vecchi, 1983), random search (Matyas, 1965; Solis and Wets, 1981) and genetic algorithms (Goldberg, 1989). However, new methodologies have emerged such as artificial immune systems (Farmer, Packard and Perelson, 1986; Bersiniand Varela, 1991), the fast clonal algorithm (Khilwani, Prakash and Shankar, 2008) and harmony search (Lee and Geem, 2005).

[1] PhD. in Engineering, Energy Systems Area, Universidad Nacional de Colombia, Medellín, Colombia (2009); Masters in Systems Engineering , Universidad Nacional de Colombia, Medellín, Colombia (1997); Associate professor, Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia.. Director of the Applied Computing Research Group, Facultad de Minas, Universidad Nacional de Colombia. jdvelasq@ unal.edu.co

New optimisation methodologies based on using numerical sequences generated by means of a chaotic map (Li and Jiang, 1997; Choi and Lee, 1998; Tavazoei and Haeri, 2007; Yang, Li and Cheng, 2007; Velásquez, 2010) have emerged recently instead of random number generators. They are called chaos optimisation algorithms (COA).

This paper was aimed at introducing the R package R-chaosoptimiser, which is a programme for solving nonlinear optimisation problems stated as $f\ (x)$ subject to $L \leq x \leq U$, where $x,\ L,\ U$ are vectors of $n\ x\ 1$, and $f\ ()$ is a nonlinear function, so that $f\!: R^n \rightarrow R$. Our programme consisted of four building blocks:

1. The first wave carrier or basic chaos optimising algorithm (Li and Jiang (1997);
2. The enhanced cyclical coordinate search method (Velásquez, 2010);
3. Li and Jiang's second wave carrier algorithm (1997); and
4. The BFGS gradient-based optimisation algorithm, which is used within any of the previous blocks for enhancing the current best solution (Yang, Li and Cheng, 2007; Velásquez, 2010).

Using chaos optimisation algorithms (Blocks 1, 2 and 3) allows a tool to avoid local optimal points and converge towards an overall optimum within a predefined search domain. Using the BFGS algorithm is not mandatory, so that its implementation was able to optimise problems involving objective function discontinuities. However, the BFGS algorithm is a powerful local search method, meaning that it is used to exploit current knowledge about an objective function, thereby improving a current solution.

Velásquez (2011) has analysed basic chaos optimising algorithm and first and second wave carrier patterns, with the optional use of gradient-based techniques, when four well-known nonlinear benchmark functions were optimised. The author concluded that the first wave carrier was unnecessary and the successful algorithm was due to a combination of second wave carrier methodology and gradient-based optimisation.

Velásquez (2010) has presented a new chaos optimisation algorithm for which the sampling mechanism was based on coordinate search methods and classical chaos optimisation algorithms; a gradient-based technique was used for refining the final solution. Evidence reported by Velásquez (2010) has indicated that the methodology so proposed was a strong alternative to other heuristic methods.

Chaotic maps

Chaos is understood to be complex, bounded and unstable behaviour caused by a simple deterministic nonlinear system or chaotic map so that generated sequences are quasi-random, irregular, ergodic, semi-stochastic and very sensitive to an initial value (Strogatz, 2000). Using chaotic sequences instead of quasi-random number generators seems to be a powerful strategy for improving many traditional heuristic algorithms and their main use is to avoid local minima points (Caponetto, Fortuna, Fazzino and Xibilia, 2003). The following chaotic maps were used in the R-chaosoptimiser programme:

1. A logistic map:

$$\gamma_{n+1} = \lambda\gamma_n(1-\gamma_n), \ \ \text{for } 0 < \lambda \leq 4 \qquad (1)$$

2. A new chaotic map:

$$\gamma_{n+1} = \sin\,(2/\gamma_n) \qquad (2)$$

3. A sine map:

$$\gamma_{n+1} = \sin\,(\pi\,\gamma_n) \qquad (3)$$

4. A tent map:

$$\gamma_{n+1} = 2\min(\gamma_n, 1-\gamma_n) \qquad (4)$$

Algorithms used in this work

This section describes optimisation algorithms used in the R-chaosoptimiser programme.

First carrier wave (fcw) algorithm

The so-called first carrier wave algorithm is the most elementary procedure for generating candidate points x_c inside a feasible region; optimum, x_l, is the candidate point having the lowest value for $f\ (x_c)$. The process is schematised in Figure 1. Candidate points x_c (line 05) were generated in domain $[L,\ U]$ by means of chaotic sequence vector γ; each component of γ, $\gamma(i)$ was thus mapped linearly to interval $[L(i),\ U(i)]$. It has been assumed in Figure 1 that the components of γ were restricted to interval $[0,\ 1]$ as occurs for the logistic map. A new chaotic sequence vector was generated at each iteration using chaotic map $H\ ()$ (line 12); for example, each component was generated using $\gamma(i) = 4\gamma(i)\ [1-\gamma(i)]$, if $H\ ()$ were the logistic map. The current local optimum x_l was saved in line 09. In addition, every $f\ cw.C_2$ iterations the current local optimum would be refined by means of a gradient-based optimisation algorithm (line 14). In this case, $g()$ was the implementation of the BFGS algorithm in R language *optim* function. The complete process was repeated $f cw.\ C_1$ times (from line 02 to line 15).

```
01  initialize $\boldsymbol{\gamma}$, $fcw.C_1$, $fcw.C_2$, $fcw.BFGS$
02  for($c_1 = 1, \ldots, fcw.C_1$) {
03      let $new.min.found$ = FALSE
04      for($c_2 = 1, \ldots, fcw.C_2$) {
05          let $\mathbf{x}_c = \mathbf{L} + \boldsymbol{\gamma}(\mathbf{U} - \mathbf{L})$
06          if($c_1 == 1$) let $\mathbf{x}_l = \mathbf{x}_c$
07          let $\Delta f = f(x_c) - f(x_l)$
08          if($\Delta f < 0$){
09              let $\mathbf{x}_l = \mathbf{x}_c$
10              let $new.min.found$ = TRUE
11          }
12          let $\boldsymbol{\gamma} = H(\boldsymbol{\gamma})$
13      }
14      if ($fcw.BFGS$ == TRUE && $new.min.found$ == TRUE), let $\mathbf{x}_l = g(\mathbf{x}_l)$
15  }# end of algorithm
```

Figure 1. First carrier wave algorithm

Enhanced cyclical coordinate search (CCS) algorithm

Objective function $f\ (x)$ was minimised along a unitary vector coinciding with one of the coordinates axes in the enhanced cyclical coordinate search algorithm (Figure 2) proposed by Velásquez (2010). A new candidate point was obtained from the current best solution (line 08) by changing the i-th component for a random value inside the interval centred on the current best value (for the i-th component) having radius r (line

09). >This sampling was repeated *ccs.K₂* times (line 06), each time obtaining a better point and thus updating the best current solution (from lines 11 to 13). The complete cycle was repeated *ccs.K₁* times (line 02).

Each time a cycle was completed on the whole axis, the current best solution was optionally refined using the BFGS gradient-based optimisation algorithm (line 17). The following methods were provided for calculating r:

– Straight:

$$r = ccs.Rinitial + (ccs.Rfinal - ccs.Rinitial)/(ccs.K_1 - 1) * (k_1 - 1) \quad (5)$$

– Geometric:

$$r = ccs.Rinitial * \left(\frac{ccs.Rfinal}{ccs.Rinitial}\right)^{\frac{(k_1-1)}{ccs.K_1}-1} \quad (6)$$

– Reciprocal:

$$r = \frac{ccs.Rinitial * ccs.Rfinal * (ccs.K_1 - 1)}{ccs.Rfinal * ccs.K_1 - ccs.Rinitial + (ccs.Rinitial - ccs.Rfinal) * k_1} \quad (7)$$

– Logarithmic:

$$r = \frac{ccs.Rinitial * ccs.Rfinal * (\log(ccs.K_1 + 1) - log(2))}{ccs.Rfinal * \log(ccs.K_1 + 1) - ccs.Rinitial * \log 2 + (ccs.Rinitial - ccs.Rfinal) * \log(k_1 + 1)} \quad (8)$$

– None: a constant radius was used.

```
01 initialize γ, $ccs.K_1$, $ccs.K_2$, $ccs.BFGS$, $ccs.Rinitial$, $ccs.Rfinal$, $ccs.Method$
02 for($k_1 = 1, \dots, ccs.K_1$) {
03     let $new.min.found$ = FALSE
04     let $r = Q(k_1, ccs.Rinitial, ccs.Rfinal, ccs.Method)$
05     for ($i = 1, \dots, n$) {
06         for ($k_2 = 1, \dots, ccs.K_2$) {
07             let $\gamma(i) = H(\gamma(i))$
08             let $\mathbf{x}_c = \mathbf{x}_l$
09             let $x_c(i) = x_c(i) + r\,[2\,\gamma(i) - 1]$
10             let $\Delta f = f(\mathbf{x}_c) - f(\mathbf{x}_l)$
11             if($\Delta f < 0$) {
12                 let $\mathbf{x}_l = \mathbf{x}_c$
13                 let $new.min.found$ = TRUE
14             }
15         }
16     }
17     if($ccs.BFGS$ == TRUE && $new.min.found$ == TRUE), let $\mathbf{x}_l = g(\mathbf{x}_l)$
18 } # end of algorithm
```

Figure 2. Enhanced cyclical coordinate search

Second carrier wave (scw) algorithm

The second wave carrier algorithm (Figure 3) consisted of a local search around X_l. γ is a chaotic sequence vector and r is a scalar parameter related to the search radius around X_l. r is calculated in each iteration by means of function *Q()* (line 04). Each candidate point was generated inside hypercube $[x_l - r, x_l + r]$, since each component of γ (with domain [0, 1]) was mapped to interval *[-r, r]*. The local optima was updated each time a better point was found (lines 08 to 11) so that the procedure continued around the new optimum. The search procedure was similar to the simulated annealing technique where ascending movements were not allowed. The following methods were provided for calculating r :

– Straight:

$$r = scw.Rinitial + (scw.Rfinal - scw.Rinitial)/(scw.M_1 - 1) * (m_1 - 1) \quad (9)$$

– Geometric:

$$r = scw.Rinitial * \left(\frac{scw.Rfinal}{scw.Rinitial}\right)^{\frac{(m_1-1)}{scw.M_1}-1} \quad (10)$$

– Reciprocal:

$$r = \frac{ccs.Rinitial * ccs.Rfinal * (ccs.K_1 - 1)}{ccs.Rfinal * ccs.K_1 - ccs.Rinitial + (ccs.Rinitial - ccs.Rfinal) * k_1} \tag{11}$$

– Logarithmic:

$$r = \frac{scw.Rinitial * scw.Rfinal * (\log(scw.M_1 + 1) - log(2))}{scw.Rfinal * \log(scw.M_1 + 1) - scw.Rinitial * \log 2 + (scw.Rinitial - scw.Rfinal) * \log(m_1 + 1)} \tag{12}$$

– None: a constant radius was used.

```
initialize $scw.M_1,\ scw.M_2,\ scw.Rinitial,\ scw.Rfinal,\ scw.Method, \boldsymbol{\gamma}$
**for**($m_1 = 1, \dots, scw.M_1$) {
    **let**$new.min.found =$ **FALSE**
    **let**$r = Q(m_1,\ scw.Rinitial, scw.Rfinal, scw.Method)$
    **for**($m_2 = 1, \dots, scw.M_2$) {
        **let**$\mathbf{x}_c = \mathbf{x}_l + r(2\boldsymbol{\gamma} - 1)$
        **let**$\Delta f = f(\mathbf{x}_c) - f(\mathbf{x}_l)$
        **if**($\Delta f < 0$) {
            **let**$\mathbf{x}_l = \mathbf{x}_c$
            **let**$new.min.found =$ **TRUE**
        }
    }
    **let**$\boldsymbol{\gamma} = H(\boldsymbol{\gamma})$
    **if**($scw.BFGS ==$ **TRUE**&&
      ( $new.min.found ==$ **TRUE** || $(c_1 == 1\ \&\&c_2 == 1)$)), **let**$\mathbf{x}_l =\ g(\mathbf{x}_l)$
}# end of algorithm
```

Figure 3. Second wave carrier algorithm

Example

An example of using the R-chaosoptimiser programme with Rosenbrock function minimisation is given below:

$$f(x, y) = 100\,(y^2 - x)^2 + (1 - x)^2 \tag{13}$$

This has been plotted in Figure 4. The above function would be generalised to n-dimensions by means of the following transformation:

$$g(\mathbf{z}) = \sum_{i=1}^{n-1} f(\mathbf{z}[i], \mathbf{z}[i+1]) \tag{14}$$

A function was first created for optimising (14):

```
> Rosenbrock.fcn <-
+ function(z) {
+   s = 0
+   for (k in 1:(length(z)-1)) {
+     x = z[k]
+     y = z[k+1]
+     s = s + 100 * (y^2 - x)^2 + (1 - x)^2
+   }
+   return(s)
+ }
```

The second carrier wave algorithm was just applied for optimising this function; R-chaosoptimiser was then used using this function:

```
> r = rchaosoptimiser(fn = Rosenbrock.fcn, N = 2, LB =
-10, UB = 10,
+   chaos.map = 2, fcw.C1 = 0, fcw.C2 = 0, ccs.K1 = 0,
ccs.K2 = 0,
+    scw.M1 = 10, scw.M2 = 100, scw.Rinitial    = 0.2,
scw.Rfinal = 0.001,
+   scw.Rmethod = "straight", scw.BFGS = FALSE, quiet =
FALSE)
```

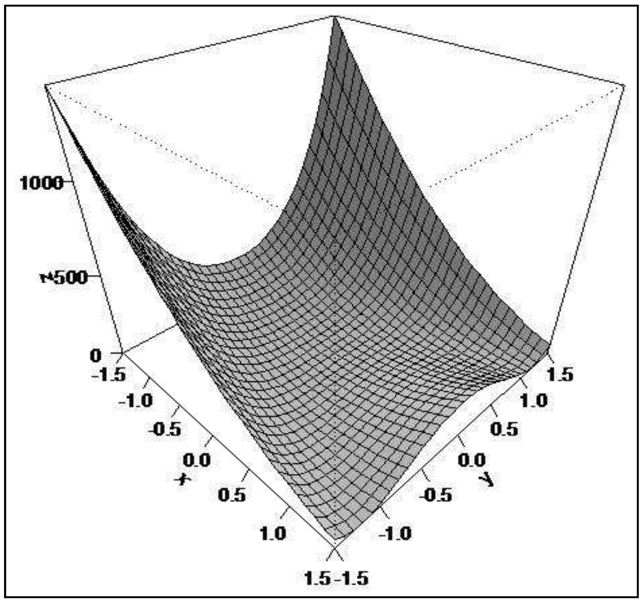

Figure 4. Rosenbrock's function in equation (13)

The first argument in this example was the function to be optimised, the second argument was the number of objective function dimensions, the third and fourth arguments were the lower and upper limits for independent variables, the fifth argument was the logistic map and the remaining arguments were the values for the constants described in the algorithms shown in Figures 1, 2 and 3. The last argument produced a verbose output for the current run.

The output produced by the above was as follows:

```
11/25/09 14:34:18
```

```
Limitsfor variables

-1.00000000E+01  <= x[  1 ] <=  1.00000000E+01
-1.00000000E+01  <= x[  2 ] <=  1.00000000E+01

Chaos map        =  logistic-map
------------------------- First carrier wave parameters
-------------------------
 fcw.C1         =  0
 fcw.C2         =  0
fcw.BFGS=  FALSE
--------------------------- Cyclical  coordinate  search
-------------------------
 ccs.K1         =  0
 ccs.K2         =  0
ccs.Rinitial=  0.2
ccs.Rfinal=  1e-04
ccs.Rmethod=  straight
ccs.BFGS=  FALSE
------------------------- Second  carrier  wave  parame-
ters -----------------------
 scw.M1         =  10
 scw.M2         =  100
scw.Rinitial=  0.2
scw.Rfinal=  0.001
scw.Rmethod=  straight
scw.BFGS=  FALSE


Optimal value for objective function:  0.01066103

Optimal values for variables

x[  1 ] :  +9.057025e-01
      x[ 2 ]: +9.538912e-01
```

The printed output consisted of the following blocks: run date and time, the variables' limits, the chaos map used, the algorithm's parameters' values, the objective function optimal value and the variables' optimal values.

The search procedure was refined by using the BFGS algorithm inside the second carrier wave algorithm, as follows:

```
> r = rchaosoptimiser(fn = Rosenbrock.fcn, N = 2, LB =
-10, UB = 10,
+   chaos.map = 2, fcw.C1 = 0, fcw.C2 = 0, ccs.K1 = 0,
ccs.K2 = 0,
+    scw.M1 = 10, scw.M2 = 100, scw.Rinitial    = 0.2,
scw.Rfinal = 0.001,
+    scw.Rmethod = "straight", scw.BFGS = TRUE, quiet =
FALSE)
```

The output generated by rchaosoptimiser was:

```
11/25/09 21:32:43

Limits for variables

-1.00000000E+01  <= x[  1 ] <=  1.00000000E+01
-1.00000000E+01  <= x[  2 ] <=  1.00000000E+01

Chaos map        =  logistic-map
------------ First carrier wave parameters ------------
 fcw.C1         =  0
 fcw.C2         =  0
fcw.BFGS=  FALSE
------------ Cyclical coordinate search --------------
 ccs.K1         =  0
 ccs.K2         =  0
ccs.Rinitial=  0.2
ccs.Rfinal=  1e-04
ccs.Rmethod=  straight
ccs.BFGS=  FALSE
------------ Second carrier wave parameters -----------
 scw.M1         =  10
 scw.M2         =  100
scw.Rinitial=  0.2
scw.Rfinal=  0.001
scw.Rmethod=  straight
scw.BFGS=  TRUE

Optimal value for objective function:  1.008543e-08

Optimal values for variables

x[  1 ] :  +9.999000e-01
      x[ 2 ]: +9.999495e-01
```

Conclusions

This paper has discussed R-chaosoptimiser, an R language package for nonlinear optimisation based on gradient techniques and chaos optimisation algorithms. Our implementation was based on three building blocks which could be executed alone or combined: the first carrier wave algorithm, the chaos-based cyclical coordinate search method and the second wave carrier algorithm. Chaos algorithms are used for defining solution space sampling mechanism and allow an algorithm to avoid using local optimal points. Gradient-based techniques were used to refine the current solution. A user would specify the BFGS algorithm for refining the current best solution inside the previous building blocks. Using the BFGS algorithm is not mandatory, so that our implementation was able to optimise problems involving discontinuities in the objective function.

Bibliography

Bersini, H., Varela, F.J.,Hints for adaptive problem solving gleaned from immune networks.,Lecture Notes in Computer Science, Vol. 496, 1991, pp. 343-354.

Caponetto, R., Fortuna, L.,Fazzino, S.,Xibilia, M. G., Chaotic sequences to improve the performance of evolutionary algorithms., IEEE Transactions on Evolutionary Computation, Vol. 7, No. 3, 2003, pp. 289-304.

Choi, C., Lee, J.J.,Chaotic local search algorithm., Artificial Life and Robotics, Vol. 2, No. 1, 1998, pp. 41-47.

Farmer, J.D., Packard, N.,Perelson, A., The immune system, adaptation and machine learning.,Physica D, Vol. 2, 1986,pp. 187-204.

Lee, K.S.,Geem, Z.W., A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice., Computer Methods in Applied Mechanics and Engineering, Vol. 194, No. 36-38, 2005,pp. 3902-3933.

Li, B., Jiang, W.-S., Chaos optimization method and its application., Control Theory and Application, Vol. 14, No. 4, 1997, pp. 613-615.

Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning., Boston, MA, Kluwer Academic Publishers, 1989.

Khilwani, N.,Prakash, A., Shankar, R.,Tiwari, M.K., Fast clonal algorithm., Engineering Applications of Artificial Intelligence, Vol. 21, No. 1, 2008, pp. 106-128.

Kirkpatrick, S.,Gelatt, C. D.,Vecchi, M. P., Optimization by simulated annealing., Science, Vol. 220, No. 4598, 1983, pp. 671-680.

Matyas, J., Random optimization., Automation and Remote Control, Vol. 26, 1965, pp. 246-253.

Pardalos, P. M.,Resende, M. G. C.,(ed.),Handbook of Applied Optimization., New York, Oxford University Press, 2002.

Solis, F.J., Wets, J.B., Minimization by random search techniques., Mathematics of Operations Research, Vol. 6, No. 1, 1981, pp. 19-30.

Strogatz, S. H., Nonlinear dynamics and chaos., Massachussetts, Perseus Publishing, 2000.

Tavazoei, M. S.,Haeri, M., An optimization algorithm based on chaotic behavior and fractal nature., Journal of Computational and Applied Mathematics, Vol. 206, No. 2, 2007, pp. 1070-1081.

Velásquez, J. D., An Enhanced Hybrid Chaotic Algorithmusing Cyclic Coordinate Search and Gradient Techniques., Revista de Ingeniería, Vol. 32, 2010, pp. 45-53.

Velásquez, J. D., Una introducción a los algoritmos basados en caos para optimización numérica., Revista Avances en Sistemas e Informática, Vol. 8, No. 1, 2011, pp. 51-60.

Yang, D., Li, G., Cheng, G., On the efficiency of chaos optimization algorithms for global optimization., Chaos, Solitons and Fractals, Vol. 34, 2007, pp. 1366-1375.