ANALYTIC PROGRAMMING IN THE TASK OF EVOLUTIONARY SYNTHESIS OF THE ROBUST CONTROLLER FOR SELECTED DISCRETE CHAOTIC SYSTEMS

[1]Roman Senkerik, [1]Zuzana Kominkova Oplatkova, [2]Ivan Zelinka, [1]Michal Pluhacek

[1]Tomas Bata University in Zlin , Faculty of Applied Informatics
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{senkerik , oplatkova , pluhacek}@fai.utb.cz

[2]Department of Computer Science, Faculty of Electrical Engineering and Computer Science
VB-TUO, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
ivan.zelinka@vsb.cz

KEYWORDS
Deterministic chaos, Control, Discrete chaotic maps, Evolutionary computation, Symbolic regression, Analytic programming, SOMA, Differential Evolution

ABSTRACT

This research deals with a utilization of a tool for symbolic regression, which is analytic programming, for the purpose of the synthesis of a new robust control law. This universal synthesized robust chaotic controller secures the fully stabilization of selected set of discrete chaotic systems. The paper consists of the descriptions of analytic programming as well as selected chaotic systems, used heuristic and cost function. For experimentation, Self-Organizing Migrating Algorithm (SOMA) and Differential evolution (DE) were used.

INTRODUCTION

During the recent years, usage of new intelligent systems in engineering, technology, modeling, computing and simulations has attracted the attention of researchers worldwide. The most current methods are mostly based on soft computing, which is a discipline tightly bound to computers, representing a set of methods of special algorithms, belonging to the artificial intelligence paradigm. The most popular of these methods are neural networks, evolutionary algorithms, fuzzy logic and tools for symbolic regression like genetic programming. Currently, evolutionary algorithms are known as a powerful set of tools for almost any difficult and complex optimization problem.
The interest about the interconnection between evolutionary techniques and control of chaotic systems is spread daily. First steps were done in (Liu et al., 2007) representing the utilization of differential evolution algorithm for the synchronization and control of chaotic systems. The papers (Senkerik et al., 2010), (Zelinka et al., 2009) were concerned to tune several parameters inside the original control technique for discrete chaotic systems. The evolutionary tuned control technique was based on Pyragas method: Extended delay feedback control – ETDAS (Pyragas, 1995). Another example of interconnection between deterministic chaos and evolutionary algorithms represents the research focused on the embedding of chaotic dynamics into the evolutionary algorithms (Aydin et al., 2010), (Davendra et al., 2010), (Pluhacek et al., 2013).
This paper shows a possibility how to generate the whole robust control law by means of analytic programming (AP) (not only to optimize several parameters) for the purpose of stabilization of a set of chaotic systems. The synthesis of control is inspired by the Pyragas's delayed feedback control technique (Just, 1999), (Pyragas, 1992).
AP is a superstructure of EAs and is used for synthesis of analytic solution according to the required behaviour. Control law from the proposed system can be viewed as a symbolic structure, which can be synthesized according to the requirements for the stabilization of the chaotic system.
Firstly, AP is explained, and then a problem design is proposed. The next sections are focused on the description of used softcomputing tools and the design of cost function. Results and conclusion follow afterwards.

MOTIVATION

This work is focused on the expansion of AP application for synthesis of a whole robust control law instead of parameters tuning for existing and commonly used control technique to stabilize desired Unstable Periodic Orbits (UPO) of set of selected discrete chaotic systems.
This work represents an extension of previous research (Kominkova Oplatkova et al., 2013), (Senkerik et al., 2013), where the new control laws were evolutionary synthesized only individually for the particular chaotic systems.
In general, this research is concerned to stabilize set of chaotic systems at p-1 UPO, which is a stable state, utilizing the only one universal robust synthesized control law.

SELECTED DISCRETE CHAOTIC SYSTEMS

This section contains the brief description of selected and used discrete chaotic systems within this research.

Logistic Equation

The first described example of discrete chaotic systems is the one-dimensional Logistic equation in form (1):

$$x_{n+1} = rx_n(1 - x_n). \tag{1}$$

The Logistic equation (logistic map) is a one-dimensional discrete-time example of how complex chaotic behaviour can arise from very simple non-linear dynamical equation (Hilborn 2000). This chaotic system was introduced and popularized by the biologist Robert May (May, 2001). It was originally introduced as a demographic model as a typical predator – prey relationship. The chaotic behaviour can be observed by varying the parameter *r*. At *r* = 3.57 is the beginning of chaos, at the end of the period-doubling behaviour. At *r* > 3.57 the system exhibits chaotic behaviour. The example of this behaviour can be clearly seen from bifurcation diagram (Figure. 1).

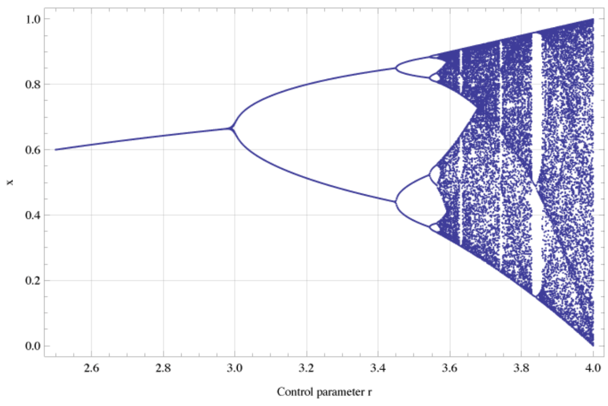

Figure 1: Bifurcation diagram of Logistic Equation

Hénon map

The second chosen example of chaotic system was the two dimensional Hénon map in form (2):

$$\begin{aligned} x_{n+1} &= a - x_n^2 + by_n \\ y_{n+1} &= x_n \end{aligned} \tag{2}$$

This is a model invented with a mathematical motivation to investigate chaos. The Hénon map is a discrete-time dynamical system, which was introduced as a simplified model of the Poincaré map for the Lorenz system. It is one of the most studied examples of dynamical systems that exhibit chaotic behavior. The map depends on two parameters, *a* and *b*, which for the canonical Hénon map have values of $a = 1.4$ and $b = 0.3$. For these canonical values the Hénon map is chaotic (Hilborn 2000). The example of this chaotic behavior can be clearly seen from bifurcation diagram – Figure 2.

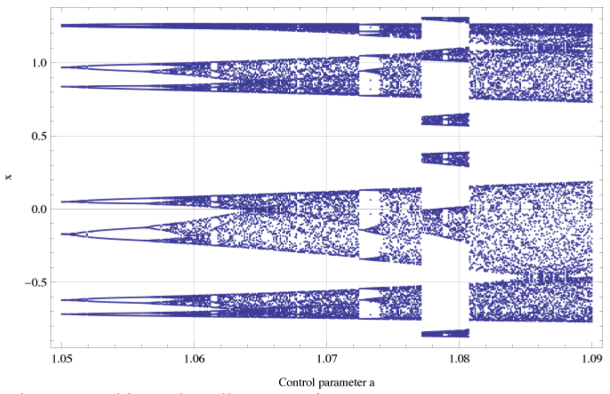

Figure 2: Bifurcation diagram of Hénon Map

Figure 2 shows the bifurcation diagram for the Hénon map created by plotting of a variable *x* as a function of the one control parameter for the fixed second parameter.

Lozi Map

The last example of discrete chaotic systems is the Lozi map, which represents the simple discrete two-dimensional chaotic map. The *x*, *y* plot of the Lozi map is depicted in Figure 3. The map equations are given in Eq. 3. The parameters are: *a* = 1.7 and *b* = 0.5 as suggested in (Sprott, 2003).

$$\begin{aligned} X_{n+1} &= 1 - a|X_n| + bY_n \\ Y_{n+1} &= X_n \end{aligned} \tag{3}$$

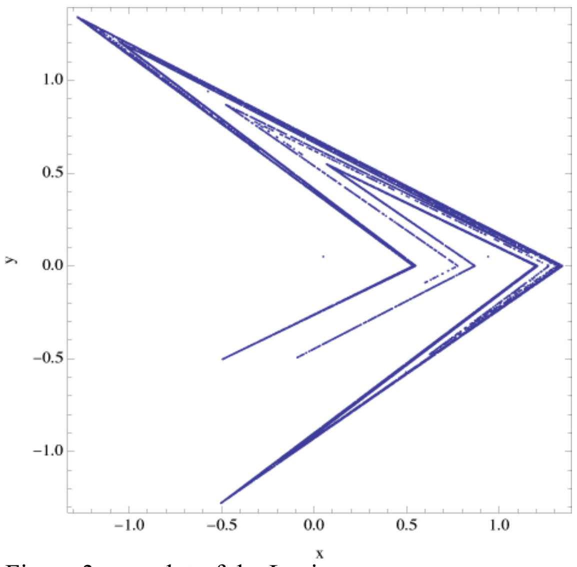

Figure 3: *x*, *y* plot of the Lozi map

ORIGINAL TDAS CHAOS CONTROL METHOD

This work is focused on explanation of application of AP for synthesis of a whole control law instead of demanding tuning of any original method control law to stabilize desired Unstable Periodic Orbits (UPO). In this research desired UPO is only p-1 (the fixed point, which represents the stable state). Original TDAS delayed feedback control method was used in this research as an

inspiration for synthesizing a new feedback control law by means of evolutionary techniques and for preparation of sets of basic functions and operators for AP.
The original control method – TDAS has form (4) and its discrete form is given in (5).

$$F(t) = K[x(t-\tau) - x(t)] \tag{4}$$

$$F_n = K(x_{n-m} - x_n) \tag{5}$$

Where: K is adjustable constant, F is the perturbation, τ_d is a time delay; and m is the period of m-periodic orbit to be stabilized. The perturbation F_n in equation (5) may have arbitrarily large value, which can cause diverging of the system. Therefore, F_n should have a value between $-F_{\max}$, $F_{\max}$. In this work a suitable $F_{\max}$ value was taken from the previous research.

USED SOFTCOMPUTING TOOLS

This section gives the brief overview and the description of used softcomputing tools. This research utilized the symbolic regression tool, which is analytic programming and two evolutionary algorithms: Self-Organizing Migrating Algorithm (Zelinka, 2004); and Differential Evolution (Price, 2005).
Future simulations expect a usage of soft computing GAHC algorithm (modification of HC12) (Matousek and Zampachova, 2011) and a CUDA implementation of HC12 algorithm (Matousek, 2010).

Analytic Programming

Basic principles of the AP were developed in 2001. Until that time only genetic programming (GP) and grammatical evolution (GE) had existed. GP uses genetic algorithms while AP can be used with any evolutionary algorithm, independently on individual representation. AP represents synthesis of analytical solution by means of evolutionary algorithms. Various applications of AP are described in (Zelinka et al., 2008), (Oplatkova et al., 2009), (Zelinka et al., 2011), (Chramcov and Varacha, 2013).
The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is set of functions, operators and so-called terminals (as well as in GP), which are usually constants or independent variables. This set of variables is usually mixed together and consists of functions with different number of arguments. Because of a variability of the content of this set, it is called here "general functional set" – GFS. The structure of GFS is created by subsets of functions according to the number of their arguments. For example GFS_{all} is a set of all functions, operators and terminals, GFS_{3arg} is a subset containing functions with only three arguments, GFS_{0arg} represents only terminals, etc. The subset structure presence in GFS is vitally important for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments, etc. The content of GFS is dependent only on the user. Various functions and terminals can be mixed together (Zelinka et al., 2005), (Oplatkova et al., 2009).
The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is a mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is called discrete set handling (DSH) (Zelinka et al. 2005) and the second one stands for security procedures which do not allow synthesizing pathological programs. The method of DSH, when used, allows handling arbitrary objects including nonnumeric objects like linguistic terms {hot, cold, dark…}, logic terms (True, False) or other user defined functions. In the AP DSH is used to map an individual into GFS and together with security procedures creates the above mentioned mapping which transforms arbitrary individual into a program.
AP needs some evolutionary algorithm that consists of population of individuals for its run. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS. The individual contains numbers which are indices into GFS. The detailed description is represented in (Zelinka et al., 2011).
AP exists in 3 versions – basic without constant estimation, AP_{nf} – estimation by means of nonlinear fitting package in Mathematica environment and AP_{meta} – constant estimation by means of another evolutionary algorithms; meta means metaevolution.

Self Organizing Migrating Algorithm - SOMA

Self Organizing Migrating Algorithm (SOMA) is a stochastic optimization algorithm that is modelled on the social behaviour of cooperating individuals (Zelinka, 2004). It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum (Zelinka, 2004). SOMA works on a population of candidate solutions in loops called *migration loops*. The population is initialized randomly distributed over the search space at the beginning of the search. In each loop, the population is evaluated and the solution with the highest fitness becomes the leader *L*. Apart from the leader, in one migration loop, all individuals will traverse the input space in the direction of the leader. Mutation, the random perturbation of individuals, is an important operation for evolutionary strategies (ES). It ensures the diversity amongst the individuals and it also provides the means to restore lost information in a population. Mutation is different in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. This parameter has the same effect for SOMA as mutation has for genetic algorithms.
The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.

The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension.
An individual will travel a certain distance (called the PathLength) towards the leader in *n* steps of defined length. If the PathLength is chosen to be greater than one, then the individual will overshoot the leader. This path is perturbed randomly. The main principle is depicted in Figures 4 and 5.

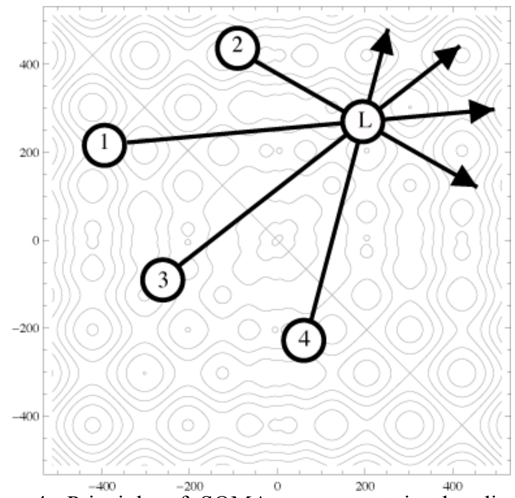

Figure 4: Principle of SOMA, movement in the direction towards the Leader

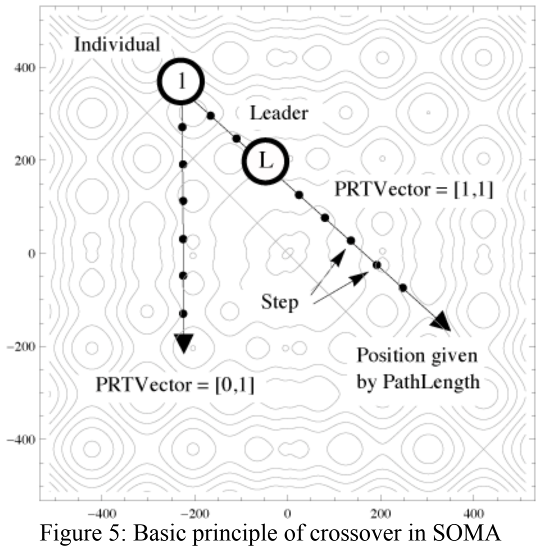

Figure 5: Basic principle of crossover in SOMA

Differential evolution

DE is a population-based optimization method that works on real-number-coded individuals (Lampinen and Zelinka, 1999), (Price, 2005). DE is quite robust, fast, and effective, with global optimization ability. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. Description of used DERand1Bin strategy is presented in (6). Please refer to (Price and Storn 2001, Price 2005) for the description of all other strategies.

$$u_{i,G+1} = x_{r1,G} + F \cdot \left(x_{r2,G} - x_{r3,G}\right) \qquad (6)$$

COST FUNCTION DESIGN

The proposal of the basic cost function (CF) is in general based on the simplest CF, which could be used problem-free only for the stabilization of p-1 orbit. The idea was to minimize the area created by the difference between the required state and the real system output on the whole simulation interval – τ_i. This CF design is very convenient for the evolutionary searching process due to the relatively favorable CF surface. Nevertheless, this simple approach has one big disadvantage, which is the including of initial chaotic transient behavior of not stabilized system into the cost function value. As a result of this, the very tiny change of control method setting for extremely sensitive chaotic system causing very small change of CF value, can be suppressed by the above-mentioned including of initial chaotic transient behavior
But another universal cost function had to be used for stabilizing of extremely sensitive chaotic system and having the possibility of adding penalization rules. It was synthesized from the simple CF and other terms were added.
This CF is in general based on searching for desired stabilized periodic orbit and thereafter calculation of the difference between desired and found actual periodic orbit on the short time interval - τ_s (40 iterations for higher order UPO) from the point, where the first minimal value of difference between desired and actual system output is found (i.e. floating window for minimization – see Figure 6.).

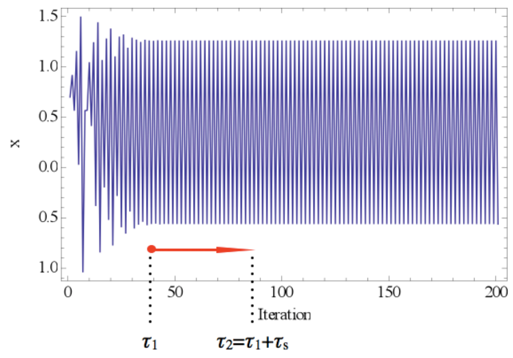

Figure 6: “Floating window” for minimization

Such a design of universal CF should secure the successful stabilization of either p-1 orbit (stable state) or higher periodic orbits anywise phase shifted. Furthermore, due to CF values converging towards zero, this CF also allows the using of decision rules, avoiding very time demanding simulations. This rule stops EA immediately, when the first individual with good parameter structure is reached, thus the value of CF is lower then the acceptable (CF_{acc}) one. Based on the numerous experiments, typically $CF_{acc} = 0.001$ at time interval $\tau_s = 20$ iterations, thus the difference between desired and actual output has the value of 0.0005 per iteration – i.e. successful stabilization for the used control technique. The CF_{Basic} has the form (7):

$$CF_{Basic} = pen_1 + \sum_{t=\tau 1}^{\tau 2} \left| TS_t - AS_t \right|, \tag{7}$$

where:
TS - target state, AS - actual state
τ_1 - the first min value of difference between TS and AS
τ_2 – the end of optimization interval ($\tau_1 + \tau_s$)
$pen_1 = 0$ if $\tau_i - \tau_2 \geq \tau_s$;
$pen_1 = 10*(\tau_i - \tau_2)$ if $\tau_i - \tau_2 < \tau_s$ (i.e. late stabilization).

Finally, the CF used within the evolutionary synthesis of a new robust control law is given in (8). It was evaluated as a sum of the three partial cost functions values each for one of the selected chaotic system.

$$CF_{Final} = CF_{Basic\_Logistic} + CF_{Basic\_Hénon} + CF_{Basic\_Lozi} \tag{8}$$

RESULTS

As described in the section about Analytic Programming, AP requires some EA for its run. In this paper AP_{meta} version was used. Meta-evolutionary approach means usage of one main evolutionary algorithm for AP process and the second algorithm for coefficient estimation, thus to find optimal values of constants in the evolutionary synthesized control law.
SOMA algorithm was used for the main AP process and DE was used in the second evolutionary process. Settings of EA parameters for both processes given in Table 1 and Table 2 were based on performed numerous experiments with chaotic systems and simulations with AP_{meta}.

Table 1: SOMA settings for AP

SOMA Parameter	Value
PathLength	3
Step	0.11
PRT	0.1
PopSize	50
Migrations	4
Max. CF Evaluations (CFE)	5345

Table 2: DE settings for meta-evolution

DE Parameter	Value
PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

The data set for AP required only constants, operators like plus, minus, power and output values x_n and x_{n-1}.

Basic set of elementary functions for AP:
GFS2arg= +, -, /, *, ^
GFS0arg= $data_{n-1}$ to $data_n$, K

Total number of cost function evaluations for AP was 5345, for the second EA it was 6000, together 32.07 millions per each simulation. See Table 3 for simple final CF values statistic.

Table 3: Cost Function values

Statistical parameter	Value
Min	$4.2186 \cdot 10^{-15}$
Max	$7.3693 \cdot 10^{-7}$
Average	$8.1544 \cdot 10^{-8}$
Median	$2.0881 \cdot 10^{-10}$
Std. Deviation	$1.8478 \cdot 10^{-7}$

Following description of the two selected experiments results contains illustrative examples of direct output from AP – synthesized control law without coefficients estimated (9) and (11); further the notation with simplification after estimation by means of second algorithm DE (10) and (12), Tables 4 and 5 with corresponding CF statistics and the average error value between actual and required system output, and finally Figures 7 – 12 with simulation results.

Experiment result 1

$$F_n = \left(x_n - x_{n-1}\right)\left(K_1 x_{n-1} + \frac{1}{K_2\left(K_3 + 2x_n\right)}\right) \tag{9}$$

$$F_n = \left(x_n - x_{n-1}\right)\left(1.12677 x_{n-1} + \frac{0.0667783}{2x_n - 1.25027}\right) \tag{10}$$

Table 4: Experiment 1 results attributes

Results attributes	Value
Final CF value	$4.2186 \cdot 10^{-15}$
$CF_{Logistic}$ value	$1.3323 \cdot 10^{-15}$
$CF_{Hénon}$ value	$1.7764 \cdot 10^{-15}$
CF_{Lozi} value	$1.1102 \cdot 10^{-15}$
Average error per iteration	$8.44 \cdot 10^{-17}$

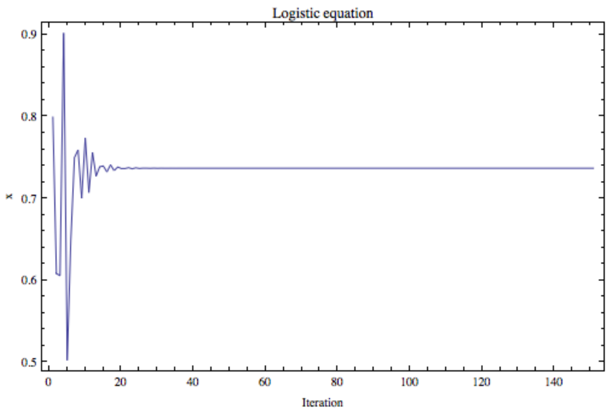

Figure 7: Experiment 1 – Simulation for Logistic equation

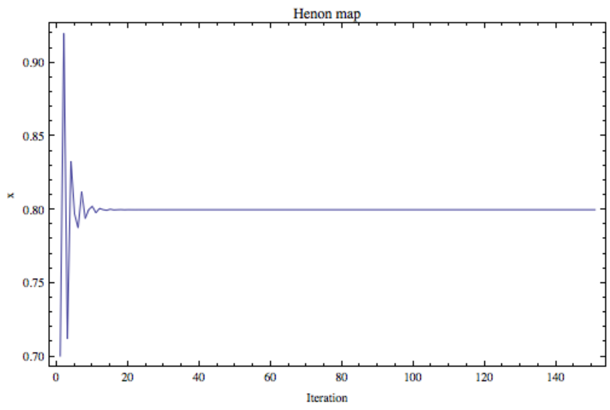

Figure 8: Experiment 1 – Simulation for Hénon map

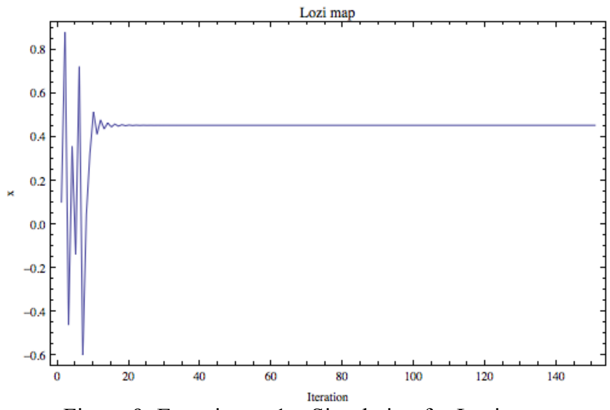

Figure 9: Experiment 1 – Simulation for Lozi map

Experiment result 2

$$F_n = \frac{K_1 x_{n-1} x_n (x_n - x_{n-1})}{2x_{n-1} + x_n - 1} \tag{11}$$

$$F_n = \frac{1.30511 x_{n-1} x_n (x_n - x_{n-1})}{2x_{n-1} + x_n - 1} \tag{12}$$

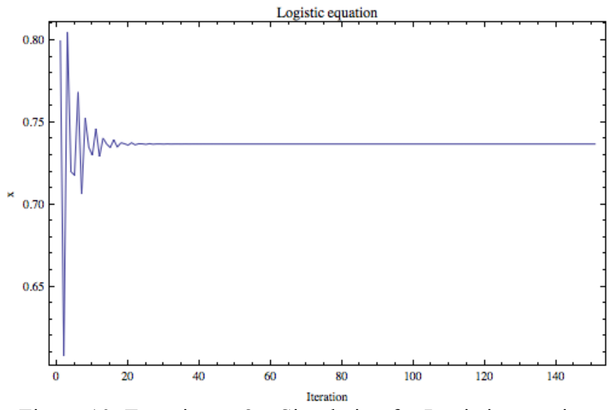

Figure 10: Experiment 2 – Simulation for Logistic equation

Table 5: Experiment 2 results attributes

Results attributes	Value
Final CF value	$4.6629 \cdot 10^{-15}$
$CF_{Logistic}$ value	$6.6613 \cdot 10^{-16}$
$CF_{Hénon}$ value	$2.9976 \cdot 10^{-15}$
CF_{Lozi} value	$9.9920 \cdot 10^{-16}$
Average error per iteration	$9.33 \cdot 10^{-17}$

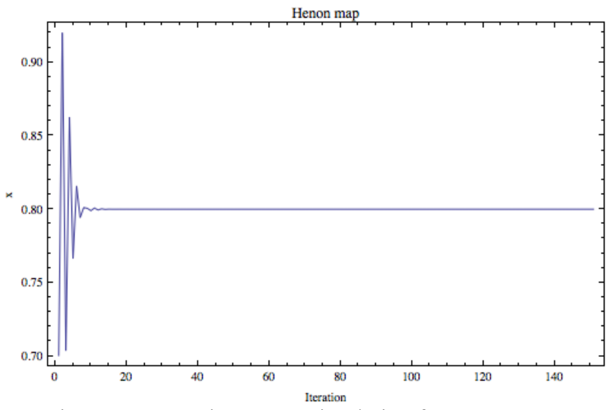

Figure 11: Experiment 2 – Simulation for Hénon map

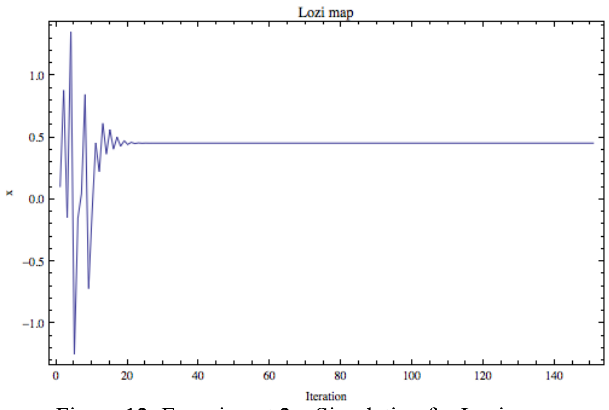

Figure 12: Experiment 2 – Simulation for Lozi map

CONCLUSION

This paper deals with a synthesis of a new universal robust control law by means of AP for stabilization of selected discrete chaotic systems at fixed point. One-dimensional Logistic equation, Hénon map and Lozi map as the examples of two-dimensional discrete chaotic systems were used in this research.

Obtained results reinforce the argument that AP is able to solve this kind of difficult problems and to produce a new robust synthesized control law in a symbolic way securing desired behaviour and precise stabilization of the selected set of chaotic systems.

The future research will include the development of better cost functions, testing of different AP data sets, and performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

ACKNOWLEDGEMENT

This work was supported by Grant Agency of the Czech Republic GACR P103/13/08195S; by the project Development of human resources in research and development of latest soft computing methods and their application in practice, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Program Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic; by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2013/012.

REFERENCES

Aydin I., Karakose M., Akin E. 2010, "Chaotic-based hybrid negative selection algorithm and its applications in fault and anomaly detection", *Expert Systems with Applications*, Vol. 37, No. 7, pp. 5285–5294.

Davendra, D., Zelinka, I., Senkerik, R. 2010, "Chaos driven evolutionary algorithms for the task of PID control", *Computers & Mathematics with Applications*, Vol. 60, No. 4, pp. 1088-1104, ISSN 0898-1221.

Hilborn R.C., 2000. Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers, Oxford University Press, ISBN: 0-19-850723-2.

Just W., 1999, "Principles of Time Delayed Feedback Control", In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8.

Kominkova Oplatkova Z., Senkerik R., Zelinka I., Pluhacek M., 2013, "Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic systems", *Computers & Mathematics with Applications*, ISSN 0898-1221, DOI 10.1016/j.camwa.2013.02.008. (article in press)

Lampinen J., Zelinka I., 1999, "New Ideas in Optimization – Mechanical Engineering Design Optimization by Differential Evolution", Volume 1, London: McGraw-hill, 20 p., ISBN 007-709506-5.

Liu B., Wang L., Jin Y.H., Huang D.X., Tang F., 2007, "Control and synchronization of chaotic systems by differential evolution algorithm", *Chaos, Solitons & Fractals*, Volume 34, Issue 2, pp. 412-419, ISSN 0960-0779

Matousek R. and Zampachova E., 2011, „Promising GAHC and HC12 algorithms in global optimization tasks", *Optimization Methods & Software*, Vol. 26, No. 3, pp. 405-419. ISSN 1055-6788.

Matousek R., 2010, „HC12: The Principle of CUDA Implementation". *In Proceedings of 16th International Conference On Soft Computing Mendel 2010*, pp. 303-308. ISBN 978-80-214-4120- 0.

May R.M., 2001, *Stability and Complexity in Model Ecosystems*, Princeton University Press, ISBN: 0-691-08861-6.

Oplatková, Z., Zelinka, I.: 2009. Investigation on Evolutionary Synthesis of Movement Commands, Modelling and Simulation in Engineering, Volume 2009 (2009), Article ID 845080, 12 pages, Hindawi Publishing Corporation, ISSN: 1687-559.

Pluhacek M., Senkerik, R., Davendra, D., Kominkova Oplatkova, Z., Zelinka, I., 2013, On the behavior and performance of chaos driven PSO algorithm with inertia weight, *Computers & Mathematics with Applications*, ISSN 0898-1221, DOI 10.1016/j.camwa.2013.01.016, (article in press).

Price, K. and Storn, R. (2001), *Differential evolution homepage*, [Accessed 28/02/2013]: http://www.icsi.berkeley.edu/~storn/code.html

Price K., Storn R. M., Lampinen J. A., 2005, "Differential Evolution: A Practical Approach to Global Optimization", (Natural Computing Series), Springer.

Pyragas K., 1992, "Continuous control of chaos by self-controlling feedback", *Physics Letters A*, 170, 421-428.

Pyragas K., 1995. "Control of chaos via extended delay feedback", *Physics Letters A*, vol. 206, pp. 323-330.

Senkerik R., Zelinka I., Davendra D., Oplatkova Z., 2010, "Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation", *Computers & Mathematics with Applications*, Volume 60, Issue 4, pp. 1026-1037.

Senkerik, R., Oplatkova, Z., Zelinka, I., Davendra, D. 2013, "Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming", *Mathematical and Computer Modelling*, Vol. 57, No. 1 - 2, pp. 57 – 67, ISSN 0895-7177.

Sprott J.C., 2003, *Chaos and Time-Series Analysis*, Oxford University Press.

Chramcov B., Varacha, P., 2013, Usage of the Evolutionary Designed Neural Network for Heat Demand Forecast. *In:Proceedings of Nostradamus 2012: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems*, p. 103-122. ISBN 978-3-642-33226-5.

Zelinka I., 2004. "SOMA – Self Organizing Migrating Algorithm", In: *New Optimization Techniques in Engineering*, (B.V. Babu, G. Onwubolu (eds)), chapter 7, 33, Springer-Verlag, 2004, ISBN 3-540-20167X.

Zelinka I.,Oplatkova Z, Nolle L., 2005. *Boolean Symmetry Function Synthesis by Means of Arbitrary Evolutionary Algorithms-Comparative Study*, International Journal of Simulation Systems, Science and Technology, Volume 6, Number 9, August 2005, pages 44 - 56, ISSN: 1473-8031.

Zelinka, I., Guanrong Ch., Celikovsky S., 2008. Chaos Synthesis by Means of Evolutionary algorithms, *International Journal of Bifurcation and Chaos*, Vol. 18, No. 4, pp. 911–942.

Zelinka I., Senkerik R., Navratil E., 2009, "Investigation on evolutionary optimization of chaos control", *Chaos, Solitons & Fractals*, Volume 40, Issue 1, pp. 111-129.

Zelinka, I., Davendra, D., Senkerik, R., Jasek, R., Oplatkova, Z., 2011, "Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures", In: *Evolutionary Algorithms*, Eisuke Kita (Ed.), InTech.