SYNTHESIS OF FEEDBACK CONTROLLER FOR CHAOTIC SYSTEMS BY MEANS OF EVOLUTIONARY TECHNIQUES

[1]*Roman Senkerik,* [1]*Zuzana Oplatkova,* [1, 2]*Ivan Zelinka,* [1]*Donald Davendra,* [1]*Roman Jasek*
[1]Tomas Bata University in Zlin , Faculty of Applied Informatics
Nad Stranemi 4511, 762 72 Zlin, Czech Republic
[2]Department of Computer Science, Faculty of Electrical Engineering and Computer Science
VB-TUO, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{senkerik, oplatkova, zelinka, davendra, jasek}@fai.utb.cz , ivan.zelinka@vsb.cz

Abstract
This research deals with a synthesis of control law for three selected discrete chaotic systems by means of analytic programming. The novality of the approach is that a tool for symbolic regression – analytic programming - is used for such kind of difficult problem. The paper consists of the descriptions of analytic programming as well as chaotic systems and used cost function. For experimentation, Self-Organizing Migrating Algorithm (SOMA) with analytic programming was used.

Keywords: Chaos Control, Analytic programming, optimization, evolutionary algorithms.

1. Introduction

The interest about the interconnection between evolutionary techniques and control of chaotic systems is spread daily. First steps were done in [1] - [3] where the control law was based on Pyragas method: Extended delay feedback control – ETDAS [4]. These papers were concerned to tune several parameters inside the control technique for chaotic system. Comparingly, this paper shows a possibility as to how to generate the whole control law (not only to optimize several parameters) for the purpose of stabilization of a chaotic system. The synthesis of control is inspired by the Pyragas's delayed feedback control technique [5], [6]. Unlike the original OGY control method [7], it can be simply considered as a targeting and stabilizing algorithm together in one package [8]. Another big advantage of the Pyragas method for evolutionary computation is the amount of accessible control parameters, which can be easily tuned by means of evolutionary algorithms (EA).
Instead of EA utilization [9], analytic programming (AP) is used in this research. 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. The advantage is that it is not necessary to have some "preliminary" control law and to estimate its parameters only. This system will generate the whole structure of the law even with suitable parameter values.
This work is focused on the expansion of AP application for synthesis of a whole control law instead of parameters tuning for existing and commonly used method control law to stabilize desired Unstable Periodic Orbits (UPO) of chaotic systems.
This work is in general concerned to stabilize p-1 UPO – a fixed point (stable state). Simulations with higher periodic orbits (oscillations between several values) are now in progress.
Firstly, a problem design is proposed. The next sections are focused on the description of AP and evolutionary algorithm. Results and conclusion follow afterwards.

2. Problem design

The brief description of used chaotic systems and original feedback chaos control method, ETDAS is given. The ETDAS control technique was used in this research as an inspiration for synthesizing a new feedback control law by means of evolutionary techniques.

2.1. Logistic Equation

The first of chosen examples of chaotic systems was the one-dimensional Logistic equation in form (1).

$$x_{n+1} = r x_n \left(1 - x_n\right) \tag{1}$$

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. This chaotic system was introduced and popularized by the biologist Robert May [10]. 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 r > 3.57, the system exhibits

Proceedings of the Fourth Global Conference on Power Control and Optimization
AIP Conf. Proc. 1337, 273-279 (2011); doi: 10.1063/1.3592477

chaotic behaviour. The example of this behaviour is depicted in bifurcation diagram – Figure 1.

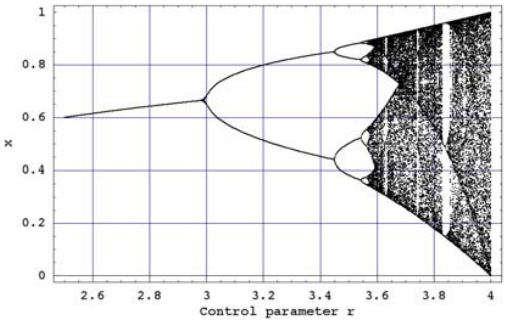

Figure 1. Bifurcation diagram of Logistic equation

2.2. Henon map

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

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

This is a model invented with a mathematical motivation to investigate chaos. The Henon 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 Henon map have values of $a = 1.4$ and $b = 0.3$. For these canonical values the Henon map is chaotic [11].
The example of this chaotic behavior can be clearly seen from bifurcation diagram – Fig.

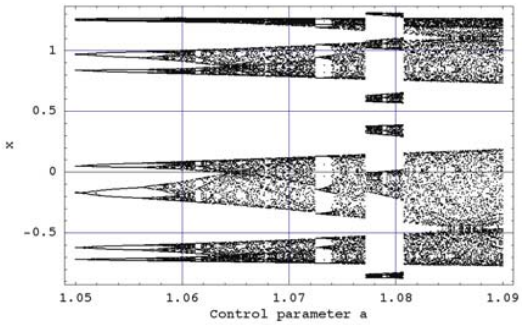

Figure 2. Bifurcation diagram of Henon Map

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

2.3. Synthesized system

The last chosen example of chaotic systems was a synthesized system introduced in [12] and also used in the research related to the investigation of the design of the "blackbox mode" cost functions securing the stabilization to desired UPO (unstable periodic orbit) [13]. The experiments published in [12] have been made for the purpose of synthesizing various chaotic systems by means of Analytic Programming. The above mentioned paper [13] discusses the use of evolutionary algorithms for controlling of selected chaotic systems synthesized by means of AP, but evolutionary algorithms were used for tuning of control rule parameters. Here, a complete synthesis of control rule structure including optimal value of all parameters is presented. This approach suppresses the fact which appeared within previous research, that some of synthesized systems are barely controllable. One of them was chosen for this case (3).

$$x_{n+1} = \frac{A\left(2A - 2x_n^{\,2} - 3x_n\left(A - x_n + Ax_n\right)\right)}{-A + x_n - x_n^{\,2}} \tag{3}$$

This system exhibits chaotic behavior for the control parameter A in the ranges <0.1, 0.13> and <0.8, 1.2> (see Figures 3 and 4).

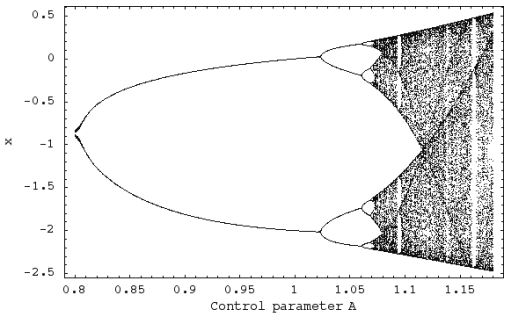

Figure 3. Bifurcation diagram for A = <0.8, 1.2>

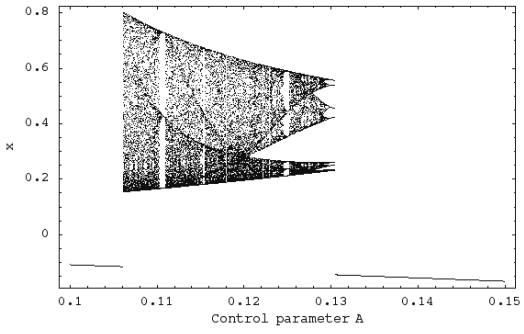

Figure 4. Bifurcation diagram for A = <0.1, 0.15>

2.4. ETDAS control method

Desired UPOs was only p-1 orbit (a fixed point) in this research. Original EDTAS chaos control method was obviously an inspiration for the preparation of sets of basic functions and operators for AP.
The original control method – ETDAS has form (4).

$$F_n = K\left[(1-R)S_{n-m} - x_n\right]$$
$$S_n = x_n + RS_{n-m} \quad (4)$$

Where K and R are adjustable constants, F is the perturbation, S is given by a delay equation utilizing previous states of the system and m is the period of m-periodic orbit to be stabilized.
The feedback perturbation F_n in equations (4) may have an arbitrarily large value, which can cause the diverging of the system outside the interval <0, 1.0> for Logistic equation, <-1.5, 1.5> for Henon map and <-2.5, 0.5> for new synthesized chaotic system (in case of control parameter A = <0.8, 1.2>).
Therefore, feedback perturbation F_n should have a value between $-F_{\max}$, $F_{\max}$. In this paper a suitable $F_{\max}$ value was taken from the previous research [3].
To find the optimal value for this threshold and including other parameters into a meta-evolution process is now in progress.

3. Cost Function

Proposal for the cost function comes from the simplest Cost Function (CF) presented in [9]. The core of CF could be used 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.

But another universal cost function had to be used for stabilizing of higher periodic orbit and having the possibility of adding penalization rules. It was synthesized from the simple CF and other terms were added. In this case, it is not possible to use the simple rule of minimizing the area created by the difference between the required and actual state on the whole simulation interval – τ_i, due to many serious reasons, for example: degrading of the possible best solution by phase shift of periodic orbit.

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 (approx. 20 - 50 iterations) from the point, where the first min. value of difference between desired and actual system output is found. Such a design of CF should secure the successful stabilization of either p-1 orbit (stable state) or higher periodic orbit anywise phase shifted.
The CF_{Basic} has the form (5).

$$CF_{Basic} = pen_1 + \sum_{t=\tau 1}^{\tau 2} \left| TS_t - AS_t \right| \quad (5)$$

Where:
TS - target state
AS - actual state
τ_1 - the first minimal 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)

4. Analytic Programming

Basic principles of the AP were developed in 2001 [14]. Until that time only Genetic Programming (GP) and Grammatical Evolution (GE) had existed. GP uses Genetic Algorithms (GA) while AP can be used with any EA, independently on individual representation. To avoid any confusion, based on the nomenclature according to the used algorithm, the name - Analytic Programming was chosen, since AP represents synthesis of analytical solution by means of EA.
The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is a 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 termed the "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 [14].
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) (Figure 5) [14] - [16] 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.

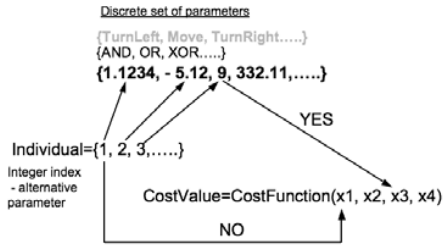

Figure 5 Discrete set handling

AP needs some EA [16] that consists of a 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 creation of the program can be schematically observed in Figure 6. The individual contains numbers which are indices into GFS. The detailed description is represented in [14] - [16].

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 implies metaevolution.

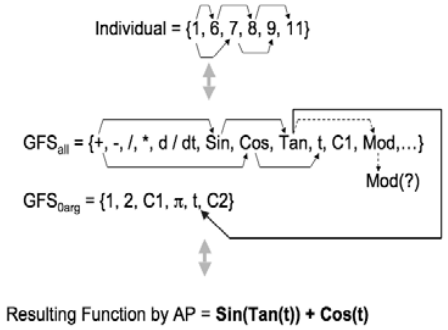

Figure 6 Main principles of AP

5. Used evolutionary algorithm

SOMA is a stochastic optimization algorithm that is modelled on the social behaviour of cooperating individuals [17]. It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum [17]. SOMA works with groups of individuals (population) whose behavior can be described as a competitive – cooperative strategy. The construction of a new population of individuals is not based on evolutionary principles (two parents produce offspring) but on the behavior of social group, e.g. a herd of animals looking for food. This algorithm can be classified as an algorithm of a social environment. To the same group of algorithms, Particle Swarm Optimization (PSO) algorithm can also be classified, sometimes called swarm intelligence. In the case of SOMA, there is no velocity vector as in PSO, only the position of individuals in the search space is changed during one generation, referred to as 'migration loop'.

The rules are as follows: In every migration loop the best individual is chosen, i.e. individual with the minimum cost value, which is called the Leader. An active individual from the population moves in the direction towards the Leader in the search space. At the end of the crossover, the position of the individual with minimum cost value is chosen. If the cost value of the new position is better than the cost value of an individual from the old population, the new one appears in new population. Otherwise the old one remains there. The main principle is depicted in Figures 6 - 8.

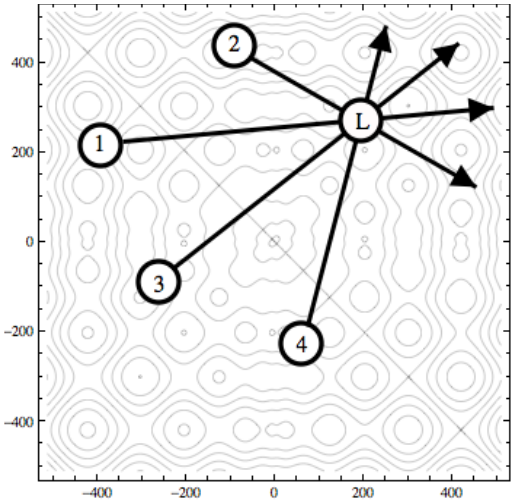

Figure 6. Principle of SOMA, movement in the direction towards the Leader

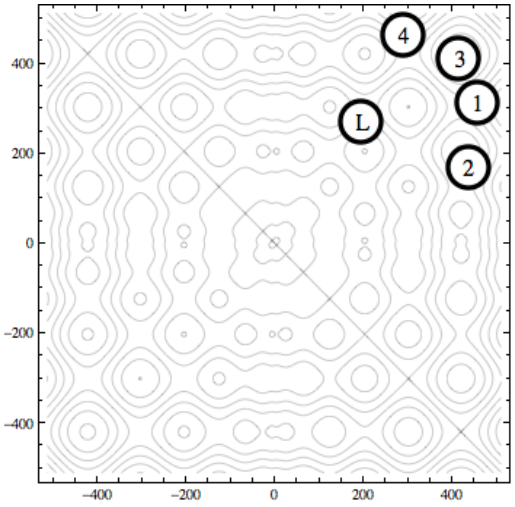

Figure 7. Principle of SOMA, the end of one migration loop

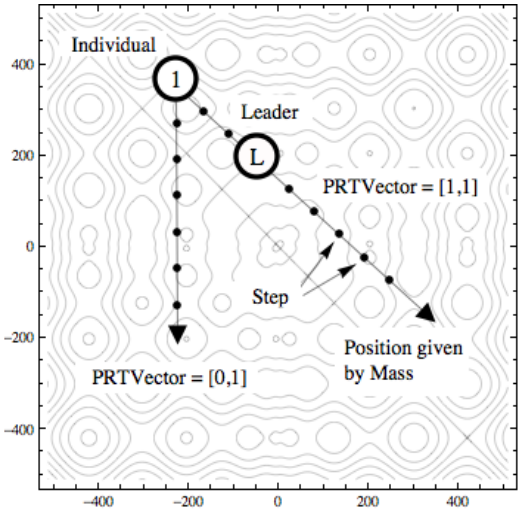

Figure 8. Basic principle of crossover in SOMA, PathLength is replaced here by Mass

6. Experimental results

As described in the last paragraph of section 4, AP requires an EA for its run. In this paper AP_{meta} version was used. SOMA ATO strategy [17] was used for both optimization tasks – to find a suitable form of the control law and in meta-evolution process, thus to find optimal values of constants in the evolutionary synthesized control law. Settings of EA parameters for both processes were similar and based on performed numerous experiments and simulations with AP_{meta} (Table 1 and Table 2).

Table 1. SOMA settings for AP

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

Table 2. SOMA settings for meta-evolution

Parameter	**Value**
PathLength	3
Step	0.11
PRT	0.1
PopSize	40
Migrations	5
Max. CF Evaluations (CFE)	5318

For all 3 cases – Logistic equation, Henon map and synthesized chaotic system, the total number of 50 simulations were carried out. All 150 simulations were successful and have given new synthesized control law, which was able to stabilize the system at required value within the short stabilizing interval of 150 iterations. Table 3 shows the number of cost function evaluations (CFE) required for obtaining the control law i.e. the main process of AP including the meta approach of second SOMA algorithm.

Table 3. CFE for AP including meta approach

Case study/ CFE	**Logistic equation**	**Henon map**	**Synthesized system**
Min	170176	154222	239310
Max	2.6*10^7	1.7*10^7	2.5*10^7

In addition to total number of CFE required for meta-evolution process, another decisive factor describing the quality of solution is a speed of stabilization of chaotic system measured in iterations. Generally there are two approaches. The first one is evaluated as a number of required iterations to achieve a selected accepted error value (10^{-6} in this research) between desired UPO and system output (Table 4). But in some rare cases, this can represent only entering of system into very close neighbourhood of desired UPO and not full stabilization. The second approach was evaluated as a first minimal difference between actual and required system output within whole simulation interval – i.e. the beginning of full stabilization at desired UPO (Table 5).

Table 4. Iterations required for stabilization – based on accepted error value

Case study/ Iterations	**Logistic equation**	**Henon map**	**Synthesized system**
Min	78	20	65
Max	151	67	77
Average	95	35	71

Table 5. Iterations required for stabilization – based on min. diff. between required and actual output

Case study/ Iterations	**Logistic equation**	**Henon map**	**Synthesized system**
Min	89	70	68
Max	131	130	81
Average	113	103	80

6.1. Example of control law for Logistic equation

The example of a new synthesized feedback control law F_n (perturbation) for the controlled Logistic equation (6) inspired by original ETDAS control method (4) has the form (7).

$$x_{n+1} = rx_n(1-x_n) + F_n \tag{6}$$

$$F_n = 0.455252x_n(x_n - x_{n-1})\left(x_n x_{n-1} + \frac{x_{n-1}}{x_n} + x_{n-1}\right) \tag{7}$$

Simulation output is depicted in Figure 9.

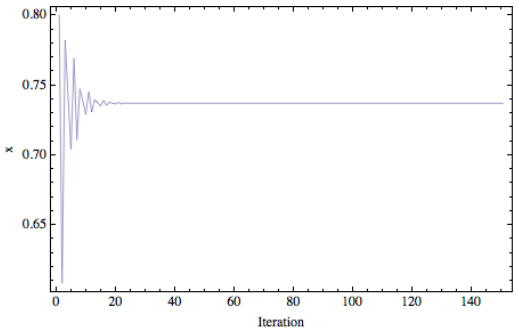

Figure 9. Stabilization of Logistic equation

6.2. Example of control law for Henon map

The example of a new synthesized feedback control law F_n (perturbation) for the controlled Henon map (8) inspired by original ETDAS control method (4) has the form (9).

$$x_{n+1} = a - x_n^2 + by_n + F_n \tag{8}$$

$$F_n = \frac{x_n - x_{n-1}}{x_n - x_{n-1} - 2x_{n-1} + x_n + 1.45831} \tag{9}$$

Simulation output representing successful and quick stabilization of Henon map is depicted in Figure 10.

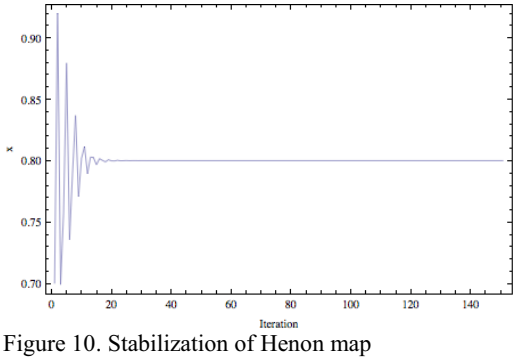

Figure 10. Stabilization of Henon map

6.3. Example of control law for the evolutionary synthesized system

The example of a new synthesized feedback control law F_n (perturbation) for the controlled evolutionary synthesized system (10) inspired by original ETDAS control method (4) has the form (11).

$$x_{n+1} = \frac{A\left(2A - 2x_n^{\,2} - 3x_n\left(A - x_n + Ax_n\right)\right)}{-A + x_n - x_n^{\,2}} + F_n \tag{10}$$

$$F_n = \frac{27.1155}{x_n\left(\left(x_{n-1}^2\right)^{x_n} + x_n + \dfrac{63.367}{x_{n-1} - x_n}\right)} \tag{11}$$

Simulation depicted in Figure 11 lends weight to the argument, that AP is able to synthesize a new control law securing very quick and precise stabilization even for artificial and complicated evolutionary synthesized chaotic system.

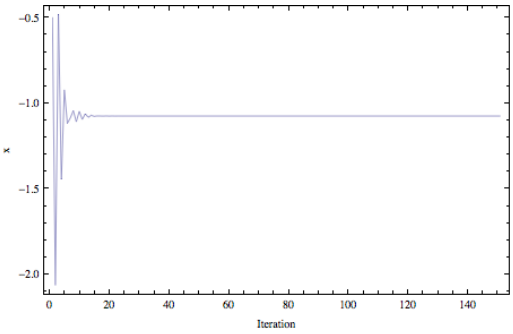

Figure 11. Stabilization of evolutionary synthesized system

7. Conclusion

This paper deals with a synthesis of a control law by means of AP for stabilization of three selected chaotic systems - Logistic equation, Henon map and

evolutionary synthesized system. Within this presented approach, the analytic programming was used instead of tuning of parameters for existing control technique by means of EA's as in the previous research.

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

All repeated 50 simulations for each case study were successful and the control law has been found.

Future plans are concerned to higher periodic orbits stabilization together with performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

8. Acknowledgement

This work was supported by the grant NO. MSM 7088352101 of the Ministry of Education of the Czech Republic and by grants of Grant Agency of Czech Republic GACR 102/09/1680.

9. References

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

[2] Senkerik R., Zelinka I., Navratil E., "Optimization of feedback control of chaos by evolutionary algorithms", in proc 1st IFAC Conference on analysis and control of chaotic systems, Reims, France, 2006, pp. 97 - 102.

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

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

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

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

[7] Ott E., C. Greboki, J.A. Yorke, "Controlling Chaos", Phys. Rev. Lett. vol. 64, 1990, pp. 1196-1199.

[8] Kwon O. J., "Targeting and Stabilizing Chaotic Trajectories in the Standard Map", Physics Letters A. vol. 258, 1999, pp. 229-236.

[9] Senkerik R., Zelinka I., Oplatkova Z., "Evolutionary Techniques for Deterministic Chaos Control", CISSE'08, In Proc. IETA 2008, International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering, 5-13 December 2008, ISBN 978-90-481-3655-1

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

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

[12] Zelinka I., Chen G., Celikovsky S., "Chaos Synthesis by Means of Evolutionary Algorithms," International Journal of Bifurcation and Chaos, Vol 18, No 4, 2008, pp. 911 – 942.

[13] Senkerik, R., Zelinka, I., Oplatkova, Z.: "Optimal Control of Evolutionary Synthesized Chaotic System", VUT Brno, 15th International Conference on Soft Computing MENDEL 2009, Brno, 2009, 220 - 227, ISBN-ISSN 978-80-214-3884-2

[14] Zelinka I.,Oplatkova Z, Nolle L., 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.

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

[16] Oplatková, Z., Zelinka, I.: "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.

[17] Zelinka I., "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