Evolutionary Chaos Controller Synthesis for Stabilizing Chaotic Hénon Maps

Roman Senkerik
Zuzana Oplatkova
Tomas Bata University in Zlin
Faculty of Applied Informatics
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic

Ivan Zelinka
Donald Davendra
Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic

This paper deals with synthesizing control laws by means of analytic programming (AP) for the Hénon map, which is a discrete chaotic system. The tool for symbolic regression is used for stabilizing the stable state and higher periodic orbits, which represent oscillations between several values of a chaotic system. For experimentation, the self-organizing migrating algorithm (SOMA) is used with AP and differential evolution (DE) is used as the second algorithm for meta-evolution.

1. Introduction

There is a growing interest about the interconnection between evolutionary techniques and controlling chaotic systems. The first steps were made in [1–3], where the control law was based on the Pyragas method, which is also called extended time delay auto synchronization (ETDAS) [4]. These papers were concerned with tuning several parameters inside the control technique for chaotic systems. This research showed the possibility for generating the whole control law (not only optimizing several parameters) for the purpose of stabilizing a chaotic system. The control law synthesis is inspired by Pyragas's time delay auto synchronization (TDAS) and the ETDAS [5, 6].

Analytic programming (AP) is used in this research. AP is a superstructure of evolutionary algorithms and is used for synthesizing an analytic solution according to the required behavior. A control law from the proposed system can be viewed as a symbolic structure that can be synthesized according to the requirements for stabilizing the chaotic system. The advantage is that it is not necessary to have some "preliminary" control law and to only estimate its parameters. This system will generate the whole structure of the law, even including suitable parameter values.

https://doi.org/10.25088/ComplexSystems.20.3.205

This research is an extension of previous work [7] focused on stabilizing the simple p-1 orbit (stable state).

2. Analytic Programming

Basic principles of AP were developed in 2001 [8]. Until then, only genetic programming (GP) and grammatical evolution (GE) existed. GP uses genetic algorithms (GAs), while AP can be used with any evolutionary algorithm, independent of the individual representation.

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 (also used in GP), which are usually constants or independent variables. This set of variables is usually mixed together and consists of functions having different numbers of arguments. Because of the variability of the set content, it is termed the general functional set (GFS). The structure of the 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; $GFS_{3\,arg}$ is a subset containing functions with only three arguments; $GFS_{0\,arg}$ represents only terminals, and so on. The subset structure present in the GFS is vitally important for AP. It is used to avoid synthesizing pathological programs, for example, programs containing functions without arguments. The content of the GFS depends only on the user. Various functions and terminals can be mixed together.

The second part of the AP core is a sequence of mathematical operations that 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) and the second is for security procedures that disallow synthesizing pathological programs. The method of DSH, when used, allows handling arbitrary objects including nonnumeric objects such as linguistic terms (hot, cold, dark, …), logic terms (True, False), or other user-defined functions. In AP, DSH is used to map an individual into the GFS and together with security procedures creates the mapping that transforms an arbitrary individual into a program.

AP needs some evolutionary algorithm that consists of a population of individuals for its execution. Individuals in the population consist of integer parameters, that is, an individual is an integer index pointing into the GFS. Three versions of AP exist: AP_{basic}, without constant estimation; AP_{nf}, where estimation is made using a nonlinear fitting package in the *Mathematica* environment; and AP_{meta}, where constant estimation is made by means of another evolutionary algorithm (meta implies meta-evolution).

https://doi.org/10.25088/ComplexSystems.20.3.205

3. Problem Design

3.1 Selected Chaotic System

The two-dimensional Hénon map was chosen as the chaotic system:

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

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.

3.2 Control Methods

This work is focused on explaining the application of AP for synthesizing a whole control law, instead of demanding the tuning of TDAS or EDTAS method control laws to stabilize desired unstable periodic orbits (UPOs). In this research, the desired UPOs are p-1 (stable state) and p-2 (higher periodic orbits resulting in an oscillation between two values).

Our inspiration for preparing sets of basic functions and operators for the synthesis of control laws for the p-1 orbit (a fixed point) was the simpler TDAS control method in its discrete form, given as

$$F_n = K(x_{n-m} - x_n). \tag{2}$$

For the purpose of stabilizing higher periodic orbits, the ETDAS method was obviously an inspiration for preparing the sets of basic AP functions and operators. The original control method, ETDAS in the discrete form suitable for the two-dimensional Hénon map, has this form:

$$\begin{aligned} x_{n+1} &= a - x_n^2 + b\, y_n + F_n, \\ F_n &= K[(1-R)\, S_{n-m} - x_n], \\ S_n &= x_n + R\, S_{n-m}, \end{aligned} \tag{3}$$

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 the m-periodic orbit to be stabilized. Due to the recursive attributes of delay equation S utilizing previous system states in the discrete ETDAS of equation (3), the AP dataset had to be expanded to cover a longer system output history (x_n to x_{n-9}), which imitates the inspiring control method for the successful control law synthesis securing the stabilization of higher periodic orbits.

3.3 Cost Function

The cost function used is, in general, based on searching for the desired stabilized periodic orbit and thereafter calculating the difference between the desired and actual periodic orbit on the short time inter-

https://doi.org/10.25088/ComplexSystems.20.3.205

val τ_S (20 iterations for the p-1 orbit and 40 iterations for the p-2 orbit) from the point, where the first minimum value of difference between the desired and actual system output is found. Such a cost function design should secure the successful stabilization of either the p-1 orbit (stable state) or a higher periodic orbit that is phase shifted. The $\mathrm{CF}_{\mathrm{Basic}}$ function has the following form:

$$\mathrm{CF}_{\mathrm{Basic}} = pen_1 + \sum_{t=\tau_1}^{\tau_2} |T S_t - A S_t|, \tag{4}$$

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

4. Evolutionary Algorithms

This research uses two evolutionary algorithms: self-organizing migrating algorithm (SOMA) [9] and differential evolution (DE) [11]. SOMA is a stochastic optimization algorithm that is modeled on the social behavior of cooperating individuals. DE is a population-based optimization method that works on real-number-coded individuals. Both algorithms were chosen because they have the proven ability to converge toward the global optimum.

4.1 Self-Organizing Migrating Algorithm

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 a social group, for example, a herd of animals looking for food. This algorithm can be classified as a social environment algorithm. Particle swarm optimization (PSO) can also be classified in the same group of algorithms, 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 a migration loop.

The rules are as follows. In every migration loop the best individual is chosen, that is, the individual with the minimum cost value, and is called the Leader. An active individual from the population moves in the direction toward the Leader in the search space. At the end of the crossover, the position of the individual with the 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 ap-

https://doi.org/10.25088/ComplexSystems.20.3.205

pears in the new population. Otherwise, the old one remains. The main principle is depicted in Figures 1 and 2. For the source codes in *Mathematica*, MATLAB, and C++ together with detailed descriptions, please refer to [10].

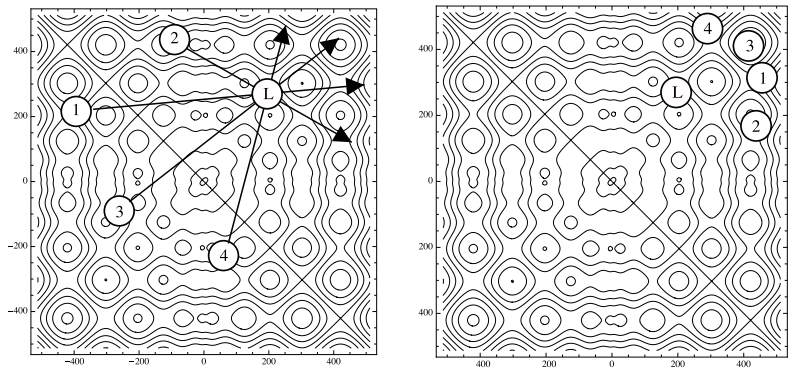

Figure 1. Principle of SOMA, movement in the direction toward the Leader.

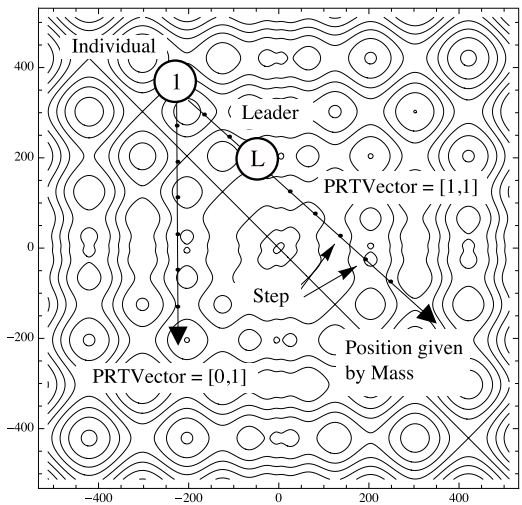

Figure 2. Basic principle of crossover in SOMA. PathLength is replaced here by Mass.

https://doi.org/10.25088/ComplexSystems.20.3.205

4.2 Differential Evolution

DE is quite robust, fast, and effective, with the ability for global optimization. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. For a description of the strategy DERand1Bin that is used here and all other DE strategies, please refer to [11, 12].

5. Simulation Results

This research uses the AP_{meta} version. A meta-evolutionary approach means using one main evolutionary algorithm for the AP process and a second algorithm for coefficient estimation. The SOMA algorithm is used for the main AP process and DE is used in the second evolutionary process. The EA parameter settings for both processes are based on numerous experiments with chaotic systems and simulations with AP_{meta} (see Tables 1 and 2).

Parameter	Value
PathLength	3
Step	0.11
PRT	0.1
PopSize	50
Migrations	4
Maximum cost function evaluations	5345

Table 1. SOMA settings for the main AP process.

Parameter	Value
PopSize	40
F	0.8
CR	0.8
Generations	150
Maximum cost function evaluations	6000

Table 2. DE settings for the secondary evolution.

https://doi.org/10.25088/ComplexSystems.20.3.205

Here is the basic set of AP elementary functions:

$\mathrm{GFS}_{2\,\mathrm{arg}} = +, -, /, *, \wedge$

$\mathrm{GFS}_{0\,\mathrm{arg}} = \mathrm{data}_{n-1}$ to data_n, K (for p-1 orbit),

$\mathrm{GFS}_{0\,\mathrm{arg}} = \mathrm{data}_{n-9}$ to data_n, K (for p-2 orbit).

The simulation results in Table 3 represent the best examples of synthesized control laws for the p-1 orbit stabilization and Table 4 shows those for p-2 orbit stabilization. The selected simulation results cover the AP output representing the synthesized control law with simplification after estimating constants by means of the second algorithm DE, corresponding cost function value, and average error between actual and required system output.

Figure	Synthesized Control Law	Cost Function Value	Average Output Error
3(a)	$F_n \dfrac{x_n x_{n-1}}{x_{n-1}(2x_n - x_{n-1}) - \frac{x_n + 0.035}{x_{n-1} - x_n}}$	1.67×10^{-15}	8.33×10^{-17}
3(b)	$F_n = -0.850\,(x_{n-1} - x_n)$	1.67×10^{-15}	8.33×10^{-17}
3(c)	$F_n = \dfrac{1.111\,(x_{n-1} - x_n)\,(0.0112\,x_n - x_{n-1}\,x_n)}{x_n}$	1.55×10^{-15}	7.77×10^{-17}
3(d)	$F_n = \dfrac{(-x_{n-1} + x_n - 0.497)\,(x_n - x_{n-1})}{x_{n-1}^2}$	1.67×10^{-15}	8.33×10^{-17}

Table 3. Simulation results for p-1 orbit.

Figure	Synthesized Control Law	Cost Function Value	Average Output Error
4(a)	$F_n = 0.524\,(x_{n-1} + x_n - 0.700)$	1.40×10^{-5}	3.50×10^{-7}
4(b)	$F_n = 0.204\,x_{n-8}(x_{n-4} - x_{n-3} + 1.880) \times x_{n-3}(x_{n-2} - x_n)$	1.70×10^{-5}	4.26×10^{-7}
4(c)	$F_n = 0.523\,(x_{n-1} + x_n - 0.7)$	2.67×10^{-5}	6.67×10^{-7}
4(d)	$F_n = 0.154\,x_{n-7}\,x_{n-3}(x_{n-2} - x_n)$	3.04×10^{-5}	7.60×10^{-7}

Table 4. Simulation results for p-2 orbit.

5.1 Stabilization of p-1 Orbit

The simulations depicted in Figure 3 lend weight to the argument that AP is able to synthesize new control laws securing very quick and precise stabilization. The average cost function value was

https://doi.org/10.25088/ComplexSystems.20.3.205

1.71085×10^{-15}, the minimal value was 1.33227×10^{-15}, and the maximal value was 4.66294×10^{-15}. The average error between the actual and required system output per iteration was calculated as the cost function value over 20 iterations.

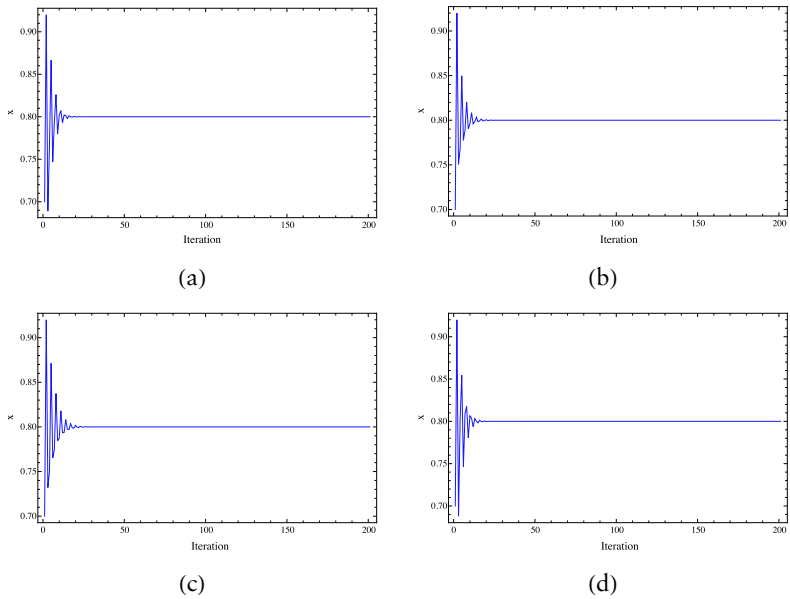

Figure 3. Simulation results showing the best new control laws synthesized for the p-1 orbit.

5.2 Stabilization of p-2 Orbit

The simulation results depicted in Figure 4 show the ability of AP to synthesize new control laws that quickly secure quality stabilization for the p-2 orbit, which is otherwise hard to control. The average cost function value was 0.335833, the minimal value was 1.39845×10^{-5}, and the maximal value was 1.23592. The average error between actual and required system output was calculated as the cost function value over 40 iterations.

https://doi.org/10.25088/ComplexSystems.20.3.205

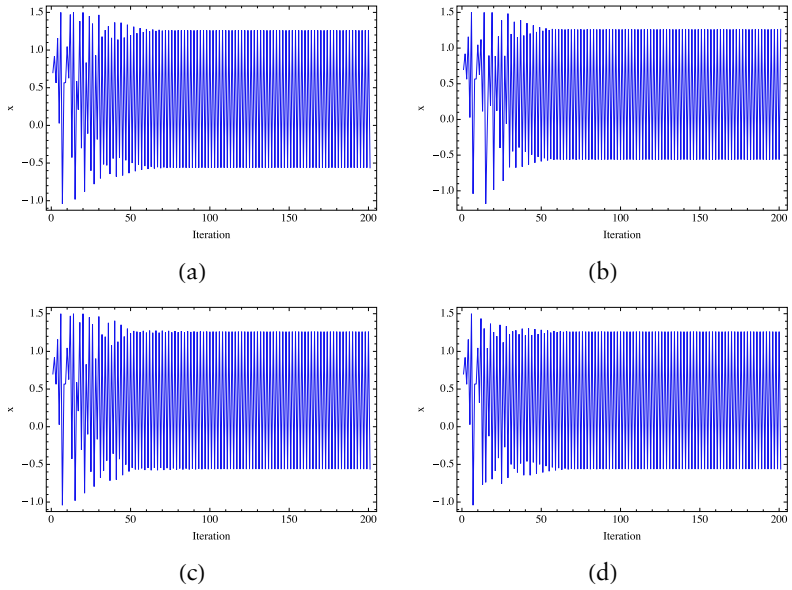

Figure 4. Simulation results with the best new synthesized control laws for p-2 orbit.

6. Conclusion

This paper deals with synthesizing laws by means of analytic programming (AP) for the stabilization of Hénon maps, which was selected as an example of a discrete chaotic system. Obtained results reinforce the argument that AP is able to solve this kind of difficult problem and produce new synthesized control laws in a symbolic way to secure the desired behavior of a chaotic system. Precise and fast stabilization lends weight to the argument that AP is a powerful symbolic regression tool able to strictly and precisely follow the rules given by a cost function and synthesize any symbolic formula. In the case of this research, AP is used to synthesize the feedback controller for chaotic systems. The question of energy costs and more precise stabilization will be included in future research together with the development of better cost functions, different AP datasets, and performance of numerous simulations to obtain more results and produce better statistics in order to confirm the robustness of this approach.

Acknowledgments

This work was supported by grant MSM 7088352101 from the Ministry of Education of the Czech Republic and by grants of Grant

https://doi.org/10.25088/ComplexSystems.20.3.205

Agency of Czech Republic GACR 102/09/1680 and by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References

[1] R. Senkerik, I. Zelinka, D. Davendra, and Z. Oplatkova, "Evolutionary Design of Chaos Control in 1D," in *Evolutionary Algorithms and Chaotic Systems* (I. Zelinka, S. Celikovski, H. Richter, and G. Chen, eds.), Berlin: Springer-Verlag Berlin, 2010 pp. 165–190.

[2] R. Senkerik, I. Zelinka, D. Davendra, and Z. Oplatkova, "Utilization of SOMA and Differential Evolution for Robust Stabilization of Chaotic Logistic Equation," *Computers & Mathematics with Applications*, **60**(4), 2010 pp. 1026–1037. doi:10.1016/j.camwa.2010.03.059.

[3] I. Zelinka, R. Senkerik, and E. Navratil, "Investigation on Evolutionary Optimization of Chaos Control," *Chaos, Solitons & Fractals*, **40**(1), 2009 pp. 111–129. doi:10.1016/j.chaos.2007.07.045.

[4] K. Pyragas, "Control of Chaos via Extended Delay Feedback," *Physics Letters A*, **206**(5–6), 1995 pp. 323–330. doi:10.1016/0375-9601(95)00654-L.

[5] K. Pyragas, "Continuous Control of Chaos by Self-Controlling Feedback," *Physics Letters A*, **170**(6), 1992 pp. 421–428. doi:10.1016/0375-9601(92)90745-8.

[6] W. Just, "Principles of Time Delayed Feedback Control," in *Handbook of Chaos Control* (H. G. Schuster, ed.), Wiley-VCH, 1999. doi:10.1002/3527607455.ch2.

[7] R. Senkerik, Z. Oplatkova, I. Zelinka, D. Davendra, and R. Jasek, "Synthesis of Feedback Controller for Chaotic Systems by Means of Evolutionary Techniques," in *Proceedings of the Fourth Global Conference on Power Control and Optimization (PCO10)*, Sarawak, Malaysia (N. N. Barsoum, J. F. Webb, and P. Vasant, eds.), Melville, NY: American Institute of Physics, 2011 pp. 273–279. doi:10.1063/1.3592477.

[8] I. Zelinka, Z. Oplatkova, and L. Nolle, "Analytic Programming—Symbolic Regression by Means of Arbitrary Evolutionary Algorithms," *International Journal of Simulation: Systems, Science and Technology*, **6**(9), 2005 pp. 44–56. http://ducati.doc.ntu.ac.uk/uksim/journal/Vol-6/No.9/Paper5.pdf.

[9] I. Zelinka, "SOMA—Self Organizing Migrating Algorithm," in *New Optimization Techniques in Engineering* (G. Onwubolu and B. V. Babu, eds.), New York: Springer-Verlag, 2004.

[10] I. Zelinka. "SOMA Homepage." (Jan 19, 2012) http://www.fai.utb.cz/people/zelinka/soma.

[11] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, New York: Springer, 1995.

[12] K. Price and R. M. Storn. "Differential Evolution Homepage." (Jan 19, 2012) http://www.icsi.berkeley.edu/~storn/code.html.

https://doi.org/10.25088/ComplexSystems.20.3.205