Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Design of Continuous-time Controllers using Cartesian Genetic Programming

Branislav Kadlic. Ivan Sekaj. Daniel Pernecký

Institute of Control and Industrial Informatics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology, Bratislava, Slovak Republic
(e-mail: ivan.sekaj@stuba.sk)

Abstract: An evolutionary computation - based design/optimisation approach using the Cartesian Genetic Programming is proposed for non-linear continuous-time process control. It is a simplification of a more general Genetic Programming – based design, which is powerful, but more computationally demanding. The approach is able to produce effective and non-intuitive controllers in the form of a network of interconnected elementary building blocks, which minimize the defined performance index. Each building block performs mathematic operations between its inputs, next it contains gain and an elementary dynamic part as integrator, derivative or unity gain. The proposed design method is demonstrated on water turbine control design, and the results are compared with the genetic algorithm-based PID controller design.

Keywords: cartesian genetic programming, continuous-time control, controller structure design, control performance optimization, non-linear systems.

1. INTRODUCTION

In continuous-time process control, we have to design controllers for different types of systems with complex structures and specific dynamics. For that reason it is necessary to use various control algorithms. Modern control theories are able to solve complex tasks, however sometimes the applied approaches do not give satisfactory results. This appears often in cases when the system to be controlled has a nontrivial structure, complex dynamic behaviour, it contains nonlinearities, more inputs/outputs and it is affected by noise or disturbances. The design process may be complicated and many times analytical methods may not be able to yield satisfactory results. In contrast to these, evolutionary-based search techniques are often able to find good solutions, to generate new control laws and also non-intuitive solutions.

Many authors have applied various evolutionary computation methods for control design applications. It is possible to classify the known approaches which are using evolutionary algorithms (EA) into two groups. In the first group the EA's are used as a powerful optimisation/search tool in analytically formulated control design methods. The parameters of controllers (or any dynamic system) are designed mathematically, based on the stability theory or other analytically formulated conditions e.g. Kawabe et al., (1996), Krohling & Rey (2001), Sekaj & Veselý (2005). The second group of methods applies simulation-based closed-loop evaluation of the model as in Khatib (1999), Mitsukura et al (1999), Sweriduk et al. (1999), Sekaj (1999), Herrero et al. (2002), Lewin (2005), Yang (2005), Sekaj (2011). Survey of evolutionary-based control system designs can be found in Lewin (2005) or Sekaj (2011).

If the task is to search/optimize parameters of an a-priori known, respectively fixed defined structure of an object, Genetic algorithms (or Evolution strategies, Differential evolution, PSO, etc.) can be used e.g. Eiben (2003). On the other hand, if the structure of the designed object is unknown, an extension of these approaches is possible using Genetic programming (GP) e.g. Koza (1992), Banzhaf et al. (1999), Koza et al. (2000). GP is also able to solve complex tasks in process control and to produce powerful and non-intuitive results e.g. Sekaj (2007). A possible way, which is here described uses the control algorithm representation based on an interconnected network consisting of the following elementary continuous-time dynamic function blocks: integrator, derivative unit, amplifier (multiplication by a constant), summation and multiplication unit (Fig. 1) where A, B and D are real constants.

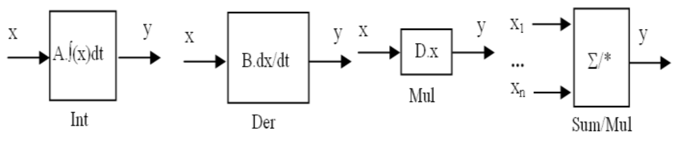

Fig. 1. Elementary building blocks of a GP-based controller

The objective is to find the optimal control network consisting of such types of elementary function blocks and their interconnection, which minimizes a selected performance criterion (as described in part 2.3).

In another approach, which is also presented in Sekaj (2007) a discrete-time recurrent control algorithm has been designed in the form of a function of selected time-delayed input variables. The following set of variables has been used:

$\theta = \{e(k),\ e(k-1),\ \ldots,\ e(k-m),\ y(k),\ y(k-1),\ \ldots,\ y(k-n),\ u(k-1),\ u(k-2),\ \ldots,\ u(k-p), r(k), c\}$

where k is the control step, e is control error, y is the controlled value, u is control value, r is the reference value, c are real constants. The aim is to find the optimal form of the controller function

$$F(\theta)=?$$

such that the cost function (as in part 2.3) is minimized. The function F contains operators +, -, * and arguments, which are arbitrary items of the vector θ.

The main drawback of the mentioned GP-based approaches is the high (extremely high) computation effort/time needed to obtain a solution. Design of simple SISO controllers can take days of computation time.

For that reason in this contribution, an alternative approach is presented which is based on Cartesian Genetic Programming (CGP) e.g. Miller (2011). The basic idea of CGP is to introduce some limitations/simplifications in the task definition in comparison to GP, which allows obtaining acceptable performance under much lower computation requirements. In our approach the building blocks in CGP are located in an orthogonal grid.

The proposed approach is demonstrated in the evolution of a controller of a hydro-turbine.

2. CONTROLLER DESIGN BASED ON CARTESIAN GENETIC PROGRAMMING

2.1 Problem formulation

The goal of the proposed CGP is the design of a controller of a non-linear dynamic system which is constructed using the searching/optimization of interconnections and parametrisation of simple building blocks. Each building block consists of a serial connection of 3 elementary units (Fig.2). The inputs of the block are processed using arithmetic operators (op): summation, subtraction, multiplication or division, which define the mathematical relations between all input signals of the particular block. Next, the signal is multiplied by a gain and finally it is processed by an elementary dynamic operator: integrator, derivative or unit gain. More such building blocks are organized in a column, where the output of each block can be connected to an arbitrary other input of another building block (Fig. 10). Input signals of the controller (i.e. control error, controlled value, etc.) can be connected to inputs of each building block. The output of the last building block represents the output of the entire interconnected controller network. Such a network can be considered as an orthogonal grid, where each node of the grid contains an elementary operator. The interconnection of the nodes is not arbitrary (as in case of GP), but it is limited to the above mentioned rules. Thank to such simplifications the computation time of CGP in comparison to GP is shorter. The goal of the controller design is to find such (sub)optimal network of building blocks, which maximize the control performance (minimizes cost function) in a defined control loop.

2.2 Representation of the individual in the CGP

An individual in CGP is a potential solution, which represents the complete information of a controller including its internal structure and its parameters. Each individual is a member of the population. The population contains a set of individuals and represents the main data structure of the CGP.

Each individual contains N interconnected building blocks. Each building block consists of elementary units (the mathematical operations and constants), which are generated by the evolutionary algorithm (EA). Also the interconnections between the controller inputs, the building blocks and the controller outputs are generated and evolved by the EA. Their number is limited to M. N and M are a-priori defined values and they depend on the complexity of the controlled system. They should be estimated by the designer. Note, that if N or M is larger than needed, the algorithm generates less than N blocks or M interconnections respectively or they remain unused. If N or M is smaller than needed, the performance of the controller will be insufficient.

Each individual is represented by a string in the following form: *[bt, bg, ic, oc, mc]*.

bt - vector of N block types (1-unit gain, 2-integrator, 3-derivative)
bg – vector of N gain values of each module (real values)
ic – vector of M input connection points (order number of previous connected modules)
oc – vector of M appropriate output connection points (order number of next connected modules)
mc – vector of M mathematic operations, one for each connection (1-summation, 2-subtraction, 3-multiplication)

The length of each individual is $L=N+N+M+M+M$ items. It is a string of L real numbers.

Remark: If using GP for interconnection design of the elementary building blocks (operators) no such strict limitations are considered. This allows generating practically unlimited structures. The only limitation in GP is the number of building blocks or the size of the interconnected network respectively.

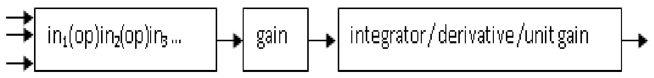

Fig. 2. Controller building block in the proposed CGP

2.3 Cost function

The control design objective is to provide required dynamic behaviour of the controlled process, usually represented in terms of the well-known performance measures: maximum overshoot, settling time, decay rate, steady state error or various integral performance indices.

Without loss of generality let us consider a simple feedback loop (Fig. 3) where *y* is the controlled value, *u* is the control value, *ref* is the reference and *e* is the control error (*e=r-y*). Let an appropriate simulation model of the controlled object be available. The closed-loop performance will be assessed using the simple integral performance index "integral of absolute control error" defined as

$$I_{AE} = \int_0^T |e(t)|dt \qquad (1)$$

where *T* is the simulation time. The discrete form of this performance index is

$$I_{AE} = \sum_{k=1}^{S} T_{s,k} |e_k| \qquad (2)$$

where $T_{s,k}$ is the simulation step size, *k* is the simulation step and *S* is the number of simulation steps.

If it is necessary to reduce the overshoot or to damp oscillations, it is recommended to insert in the integral additional terms, which include absolute values of the first or also the second derivatives of the control error

$$J = \int_0^T \left(|e(t)| + \alpha|e'(t)| + \beta|e''(t)|\right)dt \qquad (3)$$

and to increase weight coefficients α and β to increase the oscillation dumping. The weight constants can be determined according the specific needs of the designer experimentally (minimising of steady state error or overshoot, damping of oscillations, etc.) under consideration of the dynamics of the particular controlled system. Besides (1), (2) and (3) also other possible control performance criterions (e.g. energy minimisation, reference signal tracking, robustnes measures) are in Sekaj (2011).

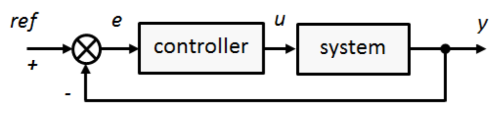

Fig. 3. Block scheme of the closed loop
ref – reference value, *e* – control error,
u – control value, *y* – controlled value

The controller design is actually an optimisation task – a search for such a controller structure and its parameters from the defined parameter space that minimises the performance index. The cost function evaluation consists of two steps. The first step is the closed-loop time-response simulation, the second one is the performance index evaluation (Fig. 4).

Remark about the closed-loop stability: Due to the applied performance index minimization (of type (1), (2), (3), etc.), the closed-loop stability is an implicit attribute of each solution. During the evolution, unstable individuals are eliminated because of their high value of performance index and the solution is directed into a stable parameter region. However, it is possible to include a stability test into each performance evaluation. Unstable individuals can additionally obtain high penalty values and they will be eliminated during the evolution.

2.4 *Evolutionary algorithm*

The evolutionary algorithm used is a conventional genetic algorithm (GA) e.g. Goldberg (1989), Eiben (2007) and others. For our needs the GA is based on following steps:

1. Initialization of population (set of individuals, between 30-50), randomly (random blocks and interconnections are generated),
2. cost function (fitness) calculation of each individual of the population = simulation + performance index evaluation,
3. if termination conditions are met (in our case - predefined number of generations) then end, else continue in step 4,
4. parent selection, more fit individuals have higher probability to be selected, in our case the stochastic universal sampling was used, 70% of individuals of the population were selected,
5. modification of parents by crossover and mutation = children,
6. new population completion (children + selected unchanged individuals),
7. continue in step 2.

The block scheme describing the controller evolution is shown in Fig. 4.

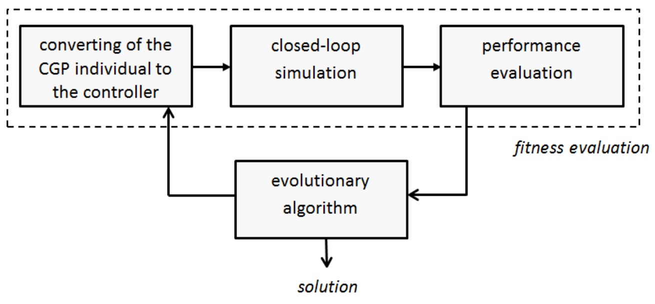

Fig. 4. Block scheme of the GA-based controller design

Note, that each mutation in the CGP design represents one from the set of following operations: random change of a mathematic operation to another operation, random change of a constant to another random value, random change of an interconnection, addition or removing of a building block, addition or removing of an input to the block. Crossover represents a random exchange of attributes of two individuals.

3. CASE STUDY

3.1 The controlled system

The proposed design procedure has been demonstrated in the example of the hydraulic turbine power control. The model and an analysis of possible control strategies of this non-linear system are frequently discussed in literature e.g. Kundur (1994), Malik (1995), Garipov (2007) and others. The turbine model parameters vary significantly in a wide range with unpredictable load variations. Various control algorithms have been proposed for control, which are based on multi model approach, adaptive methods, gain scheduling or robust methods, etc.

The model of the turbine and the penstock are determined by three equations relating to the velocity of the water in the penstock, the turbine mechanical power and the acceleration of water column. Water velocity in the penstock is given by

$$U = k_u G\sqrt{H} \tag{4}$$

where U is the water velocity, G is gate opening, H is hydraulic head at gate, k_u is a proportionality constant. The turbine mechanical power is proportional to the product of pressure and flow

$$P_m = k_p HU\,. \tag{5}$$

Acceleration of the water column due to a change in head at turbine is described by Newton's second law of motion and can be expressed in the form

$$\rho LA\frac{dU}{dt} = -A\rho a_g (H - H_0), \tag{6}$$

where H_0 is the initial steady-state value of H, A is the pipe area, L is the length of the penstock, ρ is the mass density, a_g is the acceleration due to gravity. The plant model is shown in Fig. 5. The equations (4)-(6) are in a normalized form, inelastic water column is assumed, e.g Kundur (1994).

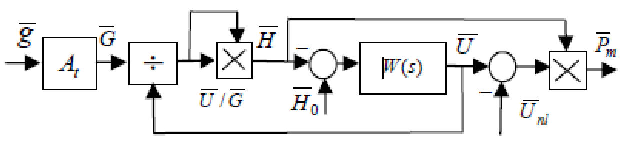

Fig. 5. Block diagram of the Hydraulic turbine

If we assume an inelastic water column the transfer function $W(s)$ will be in form

$$W(s) = \frac{1}{T_w s} \tag{7}$$

where T_w is the water starting time at rated load. It has a fixed value for a given penstock and is given by

$$T_w = \frac{LG_r}{a_g A H_r} \tag{8}$$

where H_r and Q_r are the rated values of the hydraulic head and gate and the turbine flow rate respectively. The following data related to the turbine, penstock and generator of a hydro generating unit are considered: penstock length is 700 m, rated hydraulic head is 180 m, piping area is10.25 m^2, water flow rate at rated load is 75 m^3 / s, gate opening at rated load is 0.96 p.u., gate opening at no load is 0.04 p.u., turbine rating is 150 MW, generator rating is 140 MVA and water starting time at rated load is T_w= 2.9 s.

3.2 CGP design results

In Fig. 6 and 7 the obtained regulation process of the turbine power using the controller obtained by the CGP procedure is compared with a conventional PID controller, which parameters were optimized using a genetic algorithm. The control values of both controllers are shown in Fig. 8. In Fig. 9 the graphs of the cost function minimization according (3) (*α=0.1, β=0*) during the evolution computation are shown. Finally the obtained controller using the CGP approach is depicted in Fig. 10. Six building blocks have been used. The signals turbine speed and the reference value remain unused by the CGP algorithm.

The computation time of these example on a PC (AMD Ahtlon II X4 630, 4-core, 2.8 GHz, memory 4GB) takes approximately 78 hours. The used program environment was Matlab, Release 2010b, Mathworks (2010).

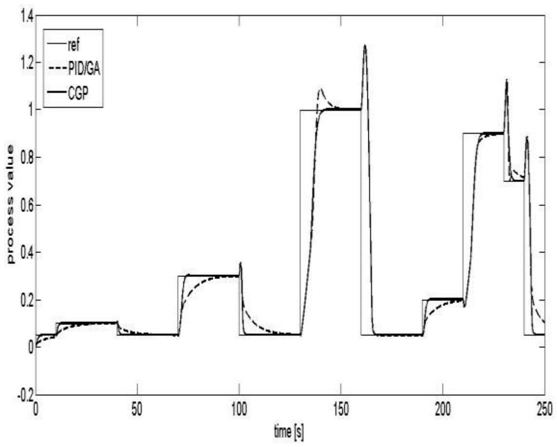

Fig. 6. Comparison of the process values (power) using CGP controller (solid) and PID controller designed using GA (dashed).

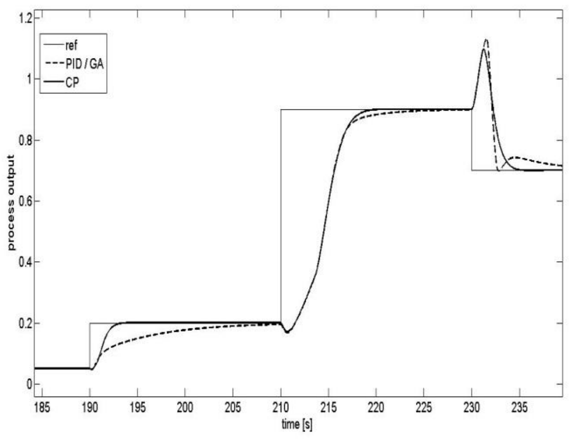

Fig. 7. Comparison of the process values (power) using CGP controller (solid) and PID controller designed using GA (dashed) – detail.

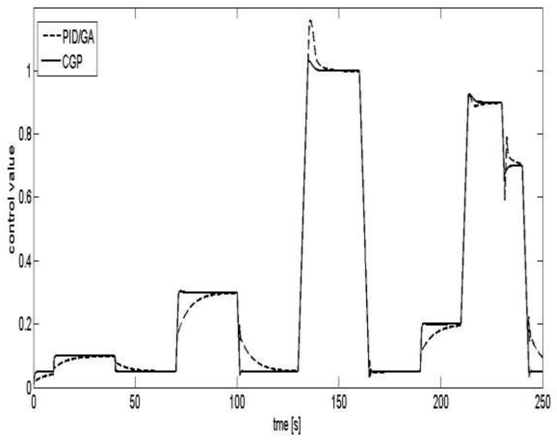

Fig. 8. Comparison of the control values (u) using CGP controller (solid) and PID controller designed using GA (dashed).

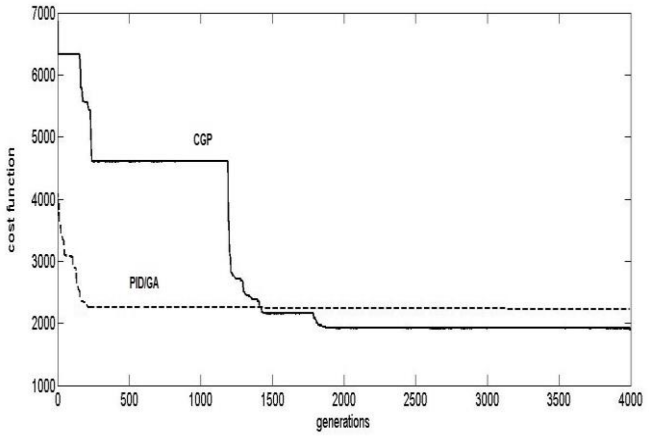

Fig. 9. Comparison of the cost function evolution graphs of the CGP controller (solid) and PID controller using GA (dashed).

4. CONCLUSIONS

Evolutionary computation methods are suitable and very effective tools for the design/optimization of control algorithms and control systems. If a fixed number of parameters of an a-priori defined control structure is to be optimized, the suitable way is to use Genetic Algorithms (or other evolutionary-based numerical optimization approaches e. g. Evolution Strategies, Differential Evolution, PSO, etc.). When the evolution of the internal controller structure is also required, a more general approach such as Genetic Programming can be used. It is able to produce practically unlimited controller structures. But the main drawbacks of GP are the extremely time demanding computation procedure and problems with control of growth of the generated structures. In this paper a new Cartesian Genetic Programming approach has been presented. Here additional limitations related to the controller structure and its size allow reduction of the unacceptable computation effort. The results have proved that a CGP is a powerful tool which is capable of finding very good but also non-intuitive solutions. The only conditions of using this approach are the existence of an appropriate simulation model of the controlled system and sufficient computation capacity. In this paper only a simple feedback closed-loop example has been presented. The proposed CGP-based design approach can be used for design of complex MIMO systems and controller design tasks with any type of non-linearity.

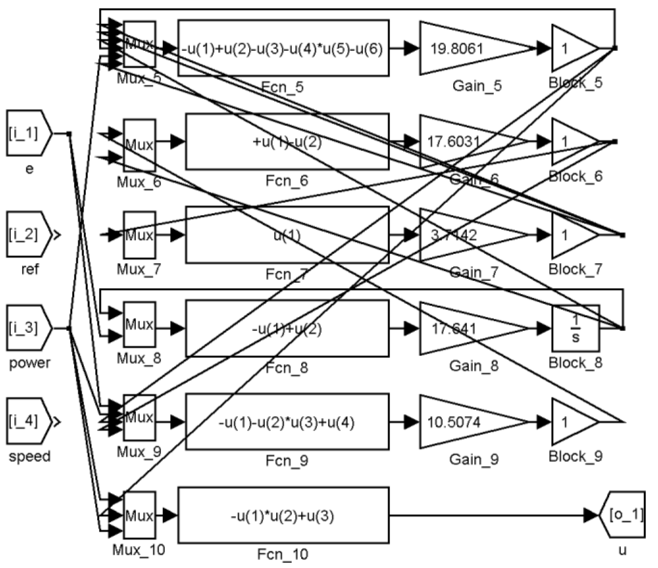

Fig. 10. Structure of the controller designed by the CGP

ACKNOWLEDGEMENT

This project was supported by the Slovak research and grant agency, grant No. VEGA 1/0178/13 and VEGA 1/2256/12.

REFERENCES

Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D. (1999). *Genetic Programming: An introduction.* Morgan Kaufmann, San Francisco

Eiben, A.E., Smith, J.E. (2003). *Introduction to Evolutionary Computing.* Springer

Garipov, E., Puleva,T., Haralanova, E. (2010). Modeling and simulation of hydraulic turbine power control., *Proc. Int. Conf. on Modeling and Simulation (MS'10 Prague),* Chech Republic.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addisson-Wesley

Herrero, J.M., Blasco, X., Martínez, M., Salcedo, J.V. (2002). Optimal PID Tuning with Genetic Algorithms for Non Linear Process Models. In *Proceedings on the 15th World Congress of IFAC,* Barcelona, July 21-26

Kawabe, T., Tagami, T., Katayama, T. (1996). A Genetic Algorithm based Minimax Optimal Design of Robust I-PD Controller. In *UKACC Int. Conference on Control '96,* Conf. Publication No. 427, IEE, pp.436-441

Khatib, W., Silva, V., Chipperfield, A., Fleming, P. (1999). Multidisciplinary Optimisation with Evolutionary Computing for Control Design. In *Proceedings on the 14th World Congress of IFAC.* Beijing, July 5-9

Koza, J.R. (1992). *Genetic Programming.* Cambridge, MA, MIT Press

Koza, J.R., Yu, J., Keane, M.A., Mydlowec, W. (2000). Evolution of a controller with a free variable using genetic programming. In *Proceedings on the European Conference EuroGP 2000.* Edinburgh, Scotland, UK, Lecture Notes in Computer Science, Volume 1802. Berlin, Germany: Springer-Verlag, pp. 91–105, April 2000

Krohling, R.A., Rey, J.P. (2001). Design of Optimal Disturbance Rejection PID Controllers Using Genetic Algorithms. In *IEEE Trans. On Evolutionary Computation,* Vol.5, No.1

Kundur, P. (1994). *Power system stability and control*. McGraw-Hill, Inc.

Lewin, D.R. (2005). Evolutionary Algorithms in Control System Engineering. In *Proceedings on the 16th World Congress of IFAC*, July 3-8, Prague

Malik, O., Zeng, Y.(1995). Design of a robust adaptive controller for a water turbine governing system. *IEEE Transaction on EC*, Vol.10, No2, pp.354-359

MathWorks, (2010). Matlab Relase 2010b. www.mathworks.com

Miller. J.F. (ed.). (2011). *Cartesian Genetic Programming.* Springer

Mitsukura, Y., Yamamoto, T., Kaneda, M., Fujii, K. (1999). Evolutionary Computation in Designing a PID Control System. In *Proceedings on the 14th IFAC World Congress,* Beijing, 5-9 july, P.R.China, pp. 497-502

Sekaj, I. (1999). Genetic Algorithm-based Control System Design and System Identification. In *Proceedings on the Int. Conference Mendel'99*, June 9-12, Brno, Czech Republic, pp.139-144

Sekaj, I., Veselý, V. (2005). Robust output feedback controller design: Genetic Algorithm approach. In *IMA Journal of Mathematical Control and Information,* 22, pp. 257-263

Sekaj,I., Perkácz, J. (2007). Genetic Programming Based Controller Design. In *Proceedings of the IEEE Congress on Evolutionary Computation'07.* Singapore, September 25-28

Sekaj, I. (2011). Control algorithm design based on evolutionary algorithms. In: Chugo, D., Yokota, S.(ed.). *Introduction to Modern Robotics*. iConcept Press, Hong Kong. (http://www.iconceptpress.com /books/introduction-to-modern-robotics)

Sweriduk, G.D., Menon, P.K., Steinberg, M.L. (1999). Design of a pilot-activated recovery system using genetic search methods. In: *Optimal Synthesis*

Yang, Z.Y., Chan, C.W., Xue, M.S., Luo, G.J. (2005). On-line Temperature Control of an Oven Based on Genetic Algorithms. In *Proceedings on the 16th World Congress of IFAC*, Prague, July 3-8