CONTROLLER DESIGN BASED ON CARTESIAN GENETIC PROGRAMMING IN MATLAB

Branislav Kadlic, Ivan Sekaj

ICII, Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava 812 19 Bratislava, Ilkovičova 3

Abstract

The aim of this contribution is to describe an evolutionary-based design of a controller of non-linear dynamic systems which is constructed using interconnection and parametrisation of simple building blocks. The library of building blocks contains gain, elementary dynamic units such as the integrator and derivative, elementary mathematic operations of input signals such as the summation, subtraction and multiplication. The evolutionary algorithm searches for such a solution, which minimise the selected performance index.

1 Introduction

Evolutionary algorithms (EA) are powerful means for design and optimisation and they are able to solve various problems in many technological, but also non-technological application domains. EA can be used with advantage also in control engineering applications for continuous-time process control, robotic system design, etc.

If the task is to search/optimise parameters of an a-priori known, respectively fixed defined structure of an object, the Genetic algorithms (or Evolution strategies, Differential evolution, PSO, etc.) can be used [1,2,3,6]. On the other hand, if the structure of the designed object is unknown, other more general evolutionary approaches have to be used as Genetic programming (GP) [4]. GP is able to solve complex tasks also in process control area and to produce powerful results [7,8]. But the drawback of GP is the high (extremely high) computation effort/time needed to obtain a solution. Design of simple single input / single output (SISO) controllers can take hours of computation. In this contribution an alternative approach is presented which is based on Cartesian programming (CP) [5]. The basic idea of CP is to consider some limitations/simplifications in the task definition in comparison with GP, which allows obtain acceptable performance under lower computation requirements. In our project CP was used for designing of controllers of continuous-time processes. An evolution of a controller of a non-linear system is demonstrated, which is based on the search and optimisation of an interconnected scheme of elementary building blocks under minimisation of a selected performance index.

2 The CP design principle

The goal is to design a controller of a non-linear dynamic system, which is constructed by means of optimisation of the interconnection and parameterisation of simple building blocks. The library of building blocks contains gain, elementary dynamic units such as integrator and derivative, elementary mathematic operations of input signals as summation, subtraction and multiplication (Fig.1). If using GP, an equal library of building elements can be considered, but for individual representation various data structures are used. The most frequent representation is the tree or table [4,8]. Such representations allow generate unlimited structures of individuals. The only limitation here is the number of building blocks or the size of the tree or size of the table, respectively. Opposite to this, in the CP also additional limitations are considered, which define the possible structures of individuals. The building blocks are often organised in a fixed grid with a-priori defined size and the task is to find their types, parameters and interconnections among them.

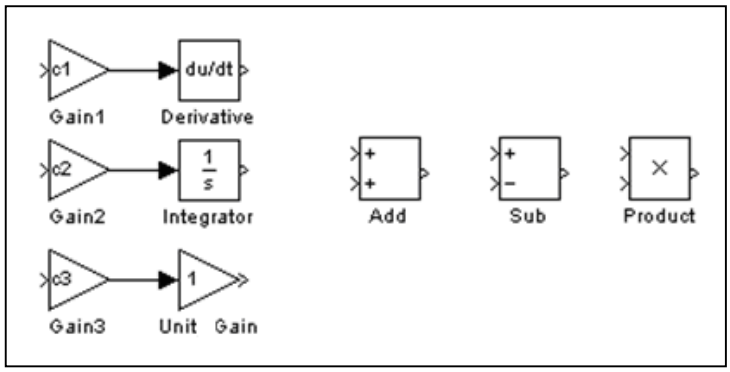

Fig.1 Set of used building blocks

2.1 Individual representation

The controller architecture is based on *N* interconnected modules, where each module contains a set of 4 elementary building blocks of various types (see example in Fig.6). Each of the *N* modules contains a multiplexer, a mathematic unit, a variable gain and the main function unit. The main function unit is either an integrator, a derivative or unit gain and its type is generated by the evolutionary algorithm. The constant of the variable gain is a next parameter which is generated by the EA. The elementary mathematic operations (summation, subtraction, multiplication) operate with the input signals to each module. The type of the mathematical operation is generated by the EA. The input blocs to each module are connected using multiplexers. The interconnections between the controller inputs, between the *N* modules and the controller outputs are generated by the EA and they number is limited to *M*. *N* and *M* are a-priori defined values by the designer and they depend on the complexity of the controlled system.

Each individual (its genotype) which is a member of the population of the EA 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 (constants of the variable gain)

ic – vector of *M* input connection points (order number of previous connected module)

oc – vector of *M* appropriate output connection points (order number of next connected module)

mc – vector of *M* mathematic operations, one for each connection

(1-summation, 2-subtraction, 3-multiplication)

bt contains the list of main function units used, *bg* contains gains of each module, *ic* and *oc* define the interconnections and *mc* the mathematic operation with each input connection. The length of each individual (string) is *N*+*N*+*M*+*M*+*M* items.

2.2 Evolutionary algorithm

We have used an evolutionary algorithm which is similar to the conventional Genetic algorithm. The only difference is the individual representation, which is characteristic for CP. The algorithm operates over a population of *100* individuals. The algorithm can be described by following steps:

1. population initialisation (by random values),
2. selection of unchanged individuals (20% by random, but including the best one), selection of parents (40% by tournament selection) which are crossed over, selection of other parents (35% by tournament selection) which are mutated, 5% of new individuals are generated by random in each generation,
3. crossover and mutation of parents,
4. completion of the new population,
5. test of terminating conditions, end or jump to point 2.

The fitness function, which is to be minimised, represents the integral performance index: *Integral of absolute control error* in form

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

and it is evaluated for each individual (phenotype) after the closed-loop simulation in Simulink. The block scheme of the controller evolution is in Fig.2.

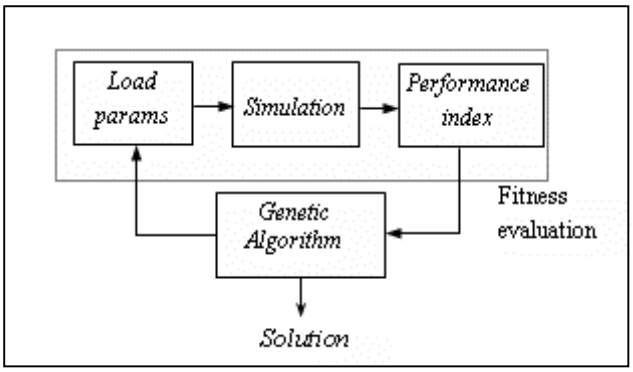

Fig.2 Block scheme of the controller design

3 Case study

Consider the system to be controlled is a SISO system, which is described by the differential equation

$$y'' + y' + y + 2y^3 - u = 0$$

The system has a non-linear dynamics. Near the working point *y=0* the time-response is non-periodical and with growing values of *y* it starts to oscillate. In Fig.3 the responses of a PID controller on changing reference values is depicted. The PID controller parameters have been designed also using the Genetic algorithm. In Fig.4 the result of the CP-based controller design is shown, the evolution of the fitness function is in Fig.5. The obtained controller for *N=10* and *M=25* is in Fig.6. We show the form obtained from the CP procedure. But the controller can be manually transformed in a more simple and transparent form, because some blocks remain unused and some operations can be cumulated in a single block.

The PID controller is not able to ensure satisfied performance, because of its linear behaviour and insufficient robustness, which is required for control of the non-linear process in a wide operating range. On the other hand, the obtained CP-based controller is able to reach the reference value in the entire operating range of the considered process output.

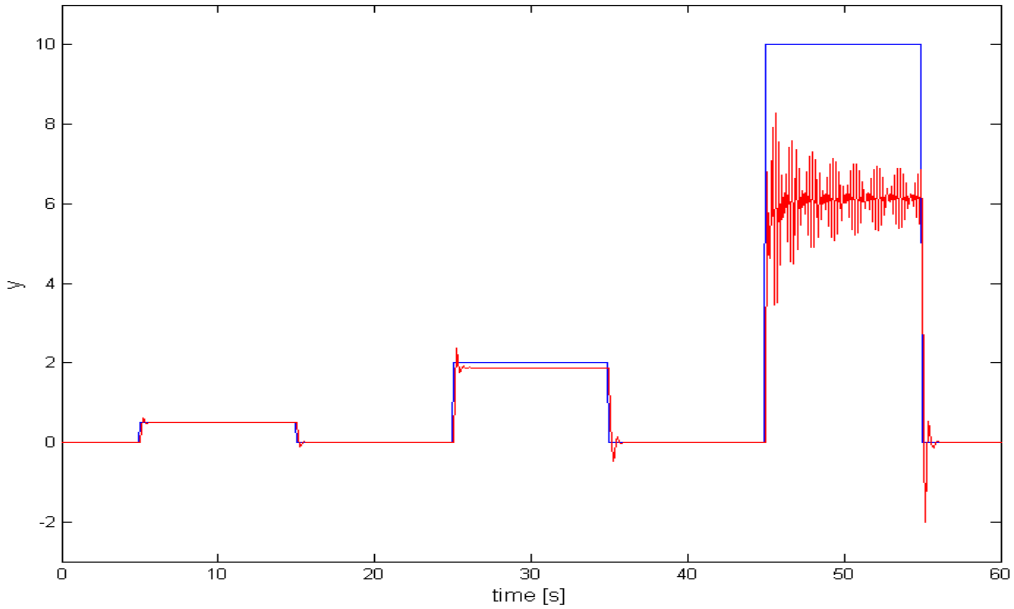

Fig.3 Response of the PID controller designed by GA

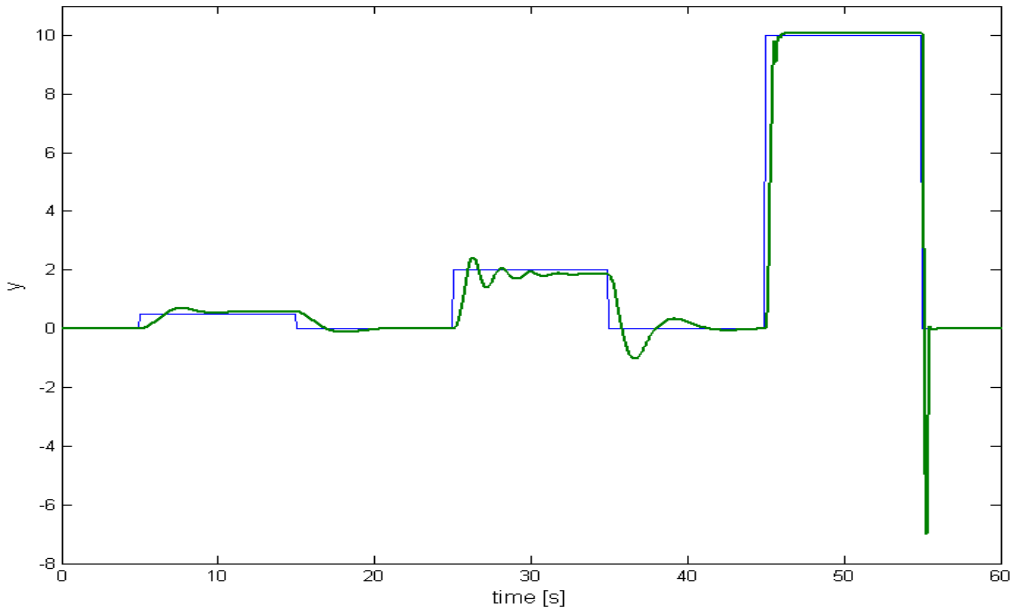

Fig.4 Response of the CP-designed controller

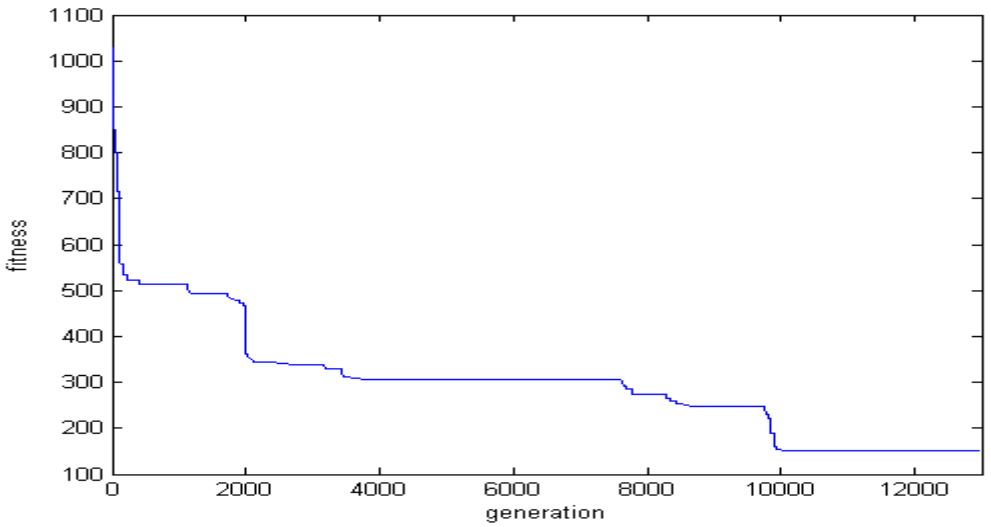

Fig.5 Fitness function evolution

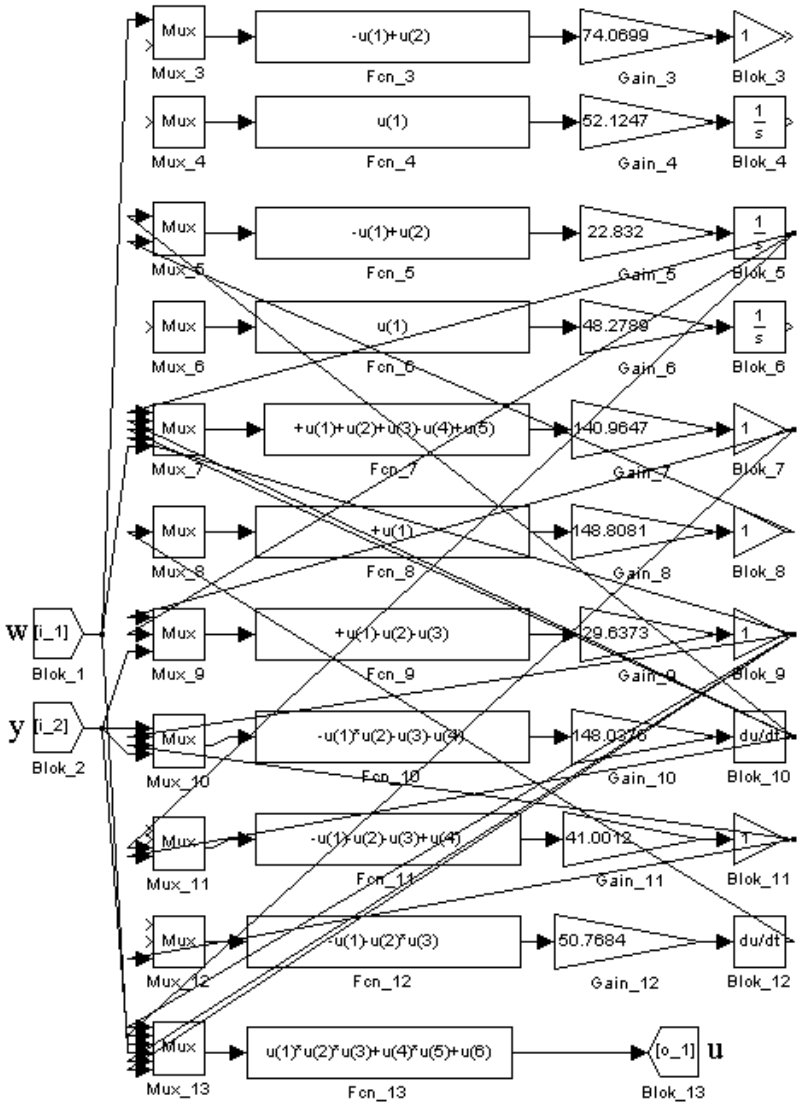

Fig.6 Block scheme of the CP-designed controller in Simulink (*N=10, M=25*)
y-controlled system output, *w*-control error, *u*-control value

4 Conclusion

In our contribution a Cartesian programming-based procedure for a non-linear continuous-time controller is presented. The obtained results have been compared to a PID controller, which parameters have been optimised using Genetic Algorithm. The experimental results show, that the CP-based controller is able to produce promising results for control of complex dynamic systems. It is evident, that the in our experiment obtained structure is not the best possible solution (global optimum). The finding of the optimal controller structure is practically impossible due to the huge search space. But as demonstrated above, the proposed approach can produce acceptable results. The next improvement of the results is possible thanks more generation used in the CP-evolution or/and extension of the population size.

Acknowledgement

The project has been supported from the grant VEGA No. 1/0690/09 of the Slovak scientific grant agency.

References

[1] A. E. Eiben, J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003
[2] V. Kvasnička, J. Pospíchal, P. Tiňo. *Evolučné algoritmy.* Vydavateľstvo STU, Bratislava, 2000 (in slovak)
[3] I. Zelinka a kol. *Evoluční výpočetní techniky, principy a aplikace*. Ben, Praha 2009
[4] J. R. Koza. *Genetic Programming*. Cambridge, MA, MIT Press, 1992
[5] L. Sekanina a kol. *Evoluční hardvare*. Academia Praha, 2009 (in czech)
[6] I. Sekaj. *Evolučné výpočty a ich využitie v praxi*. IRIS Bratislava, 2005 (in slovak)
[7] I. Sekaj. *Evolutionary Controller design.* In: Wellington Pinheiro dos Santos. Evolutionary Computation, In-Teh, Vukovar, Chroatia, 2009 (www.intechopen.com)
[8] I. Sekaj, J. Perkácz. *Genetic Programming - Based Controller Design*. IEEE Congress on Evolutionary Computation, Singapore, 2007

Branislav Kadlic
branislav.kadlic@gmail.com

Ivan Sekaj
ivan.sekaj@stuba.sk