UDC 681.5.015.24 (045)

V. Apostolyuk, Ph.D. [National Aviation University, Ukraine]

EVOLUTIONARY OPTIMAL CONTROL

Evolutionary approach to the numerical solving of multi-dimensional procedural optimal control problems with respect to arbitrary representation of plant model and constraints is considered in this paper. New genetic algorithm with variable control programs length is proposed and its characteristics are investigated while solving the test problem.

Introduction. Simulated natural evolution being applied to the multi-parameter and multi-objective global optimisation problems have already resulted in a well established group of so-called evolutionary algorithms [1, 2]. At the same time, constantly increasing computational power of the microprocessors significantly widens areas of application for such algorithms. In recent years, genetic algorithms have been successfully applied to several control problems, such as non-linear model structural identification [3], system identification [4], and controller parameters optimization [5]. Apart from that, application of conventional genetic algorithms to predictive control with fixed prediction horizon (PH) has been in active development during the last decade [6, 7]. Conventional genetic algorithm being applied to the model prediction based direct control utilizes encoding of optimisation parameters into a fixed length binary representation, which naturally fixes the PH distance and results in fundamentally sub-optimal control functions. In addition to that, using fixed PH it is impossible to detect in advance the case of non-existing solution. In order to overcome the mentioned above problems, new variable length genetic algorithm (VLGA) is proposed and studied in this paper, which could be applied to the direct procedural control, thus delivering fundamentally optimal solutions. Such an application requires both proper control functions encoding and specially developed fitness functions formulations, which will successfully solve multi-objective optimisation of the procedural direct control.

Goals and motivations. Taking into considerations model and constraints complexity level, possible applications of the procedural optimal control approach can be represented in a form of a "bubble diagram" shown in the figure 1. The darker area in the diagram reflects higher efficiency of the approach in solving corresponding problems. One should also note that the classic control approaches as well as fuzzy logic don't deliver fundamentally optimal control in terms of a given criterion. The main goal of the research is to develop and study algorithms able to solve the optimal control problems with respect to the arbitrary complexity level of both process model and constraints representation, thus overcoming limitations of the conventional control methodologies.
At the same time, considering the exponential growth of the computational power of modern processors some novel and pure algorithmic approaches are reckoned to be feasible and highly efficient in solving the optimal control problems even under conditions of real-world applications. Such algorithms must apparently have certain features that would make them advantageous in comparison with the existing methods in system optimal control:

- The only requirement to the mathematical model is that it must be able to adequately predict the system state with respect to the given control functions. No special constraints in terms of its linearity or mathematical simplicity are required whatsoever.
- Control algorithms must be able to efficiently readjust to system parameters variations.
- More control functions are considered as admissible that could lead to more optimal results.
- Straightforward treatment of optimization criteria and control constraints that could be given in an arbitrary form.

The latter would also allow the control algorithm to explicitly solve the path-planning problem with respect to the plant dynamics and arbitrary obstacle configuration. There are two major groups of algorithms that could be used either separately or combined to solve the optimal control problems:

- States traversing techniques

- Evolutionary programming.

First group includes Dijkstra's algorithm [8] and A-star with all of its modifications [9-10]. The second one refers mainly to the genetic algorithms both with fixed and variable "chromosome" length [11]. The latter variable length genetic algorithm has several specific features that make it advantageous in solving optimal direct control problems.

Optimal control problem. Formulation of general optimal control problem is well known and here we shall give emphasis only to the aspects that are essential to the subject. Let the state of the system at time t be a vector $\vec{x}(t)=\{x_1(t),...,x_n(t)\}$ in an n-dimensional Euclidean space which we shall call the state space X. The steering device or control we model as an *m*-dimensional vector function of time $\vec{u}(t)=\{u_1(t),...,u_m(t)\}$. The components of $\vec{u}(t)$ are allowed to be piecewise continuous and the values they can take are bounded so that at any time t, $\vec{u}(t)$ lies in some bounded region U of the control space. Without loss of generality we impose the restriction $|u_i|\le 1$, $i=1,...,m$. Such controls are deemed admissible in terms of the considered algorithms. We shall study systems whose behaviour can be modelled by the most general state transfer function F that calculates state of the system given its current state and control vectors:

$$\vec{x}(t+\Delta t)=F(\vec{x}(t),\vec{u}(t)). \tag{1}$$

Function F here is assumed to be defined for all $\vec{x}\in X$ and all admissible $\vec{u}$.
This approach to system representation significantly expands number of systems that could be successfully controlled compared to the conventional systems representation via systems of either ordinary or non-linear differential equations. We now wish to control the system from the initial state $\vec{x}(t_0)$ to the given final state $\vec{x}(t_1)$ while minimizing some cost function. This function could be also given in a form of a conventional cost functional

$$J(\vec{x}_0\to\vec{x}_1)=\int_{t_0}^{t_1} f_0(\vec{x}(t),\vec{u}(t))dt. \tag{2}$$

We are, of course, assuming that there are admissible controls that transfer the system from $\vec{x}(t_0)$ to $\vec{x}(t_1)$ and we are looking amongst this subset of admissible controls for a control that minimizes cost function J. Although the cost function (2) is given in a conventional form of a functional, it is not required by the nature of control algorithms under consideration. In fact, any cost function perfectly suits the most of the algorithms.

System transition in state-space. Let us assume that every component of the control vector $\vec{u}(t)=\{u_1(t),...,u_m(t)\}$ can take a value only from the given set of constant values:

$$\vec{u}^0=\{u_1^0,...,u_p^0\}. \tag{3}$$

In this case control vector $\vec{u}(t)$ has $s=m^p$ different possible constant values $\vec{u}(t)\in\vec{u}_1^0,...,\vec{u}_s^0$. Application of the piecewise constant control $\vec{u}_i=\vec{u}(t_i)$ to the system at the current state $\vec{x}_i=\vec{x}(t_i)$ can result therefore in s new states at the every subsequent instant of time $t_{i+1}=t_i+\Delta t$. In terms of the optimal control problem formulated earlier, we are searching for the finite sequence of *v* control inputs $\vec{u}_i(t)$ that transfers the system from its initial state $\vec{x}_0$ to the goal state $\vec{x}_g$. Each of the control $\vec{u}_i$ is assumed to be acting on the system and remained constant within the time interval Δt. Such a limitation causes the found solution to be a sub-optimal rather than optimal, to which it tends when $\Delta t\to 0$.

Control function encoding. There are many forms of solution representation that could be successfully used control algorithms. The simplest one is an array, where each component is an integer index of a particular component of the control values vector $\vec{u}^0=\{u_1^0,...,u_p^0\}$:

$$B=\{b_1,\dots,b_v\},\ b_i\in[1,\dots,p], \tag{4}$$

where p is the number of possible values for the control, v is the number of subsequent constant control inputs to the system. Here and after this controls array is referred to as *control program*, since each component of the array (4) can be considered as an instruction from the given instruction set (3). For the multidimensional control (m>1), instructions for every component of the control vector $\vec{u}(t)=\{u_1(t),...,u_m(t)\}$ are taken from the program (4) consecutively. If the number v is not a multiplier of m, some predefined default instruction is used. In case of the control program (4), the system is subjected to the piecewise constant control input completely defined by application of a control program

$$u_i(k\cdot\Delta t)=u^0_{b_k},\ \ k\in[1,\dots,v], \tag{5}$$

during the period of time $v\cdot\Delta t$. One should note that the piecewise constant control (5) is the simplest but not the only possible interpretation of a control program instructions. Both higher order function approximation and Boolean controls can be successfully parameterised using the integer values from the array (4) as well.

Algorithms operation. In a sense, the algorithmic optimal control is the form of a model predictive control, since it uses the model of a process or a plant to predict the effect of the suggested control function. Generalised schematic representation of the algorithms operation as a part of the control system is shown in the figure 2.

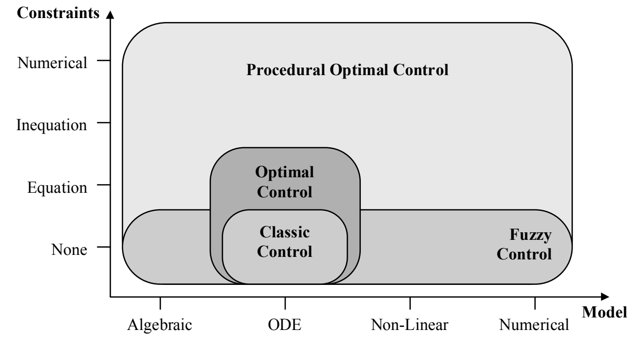


Fig. 1. Procedural control fields of application

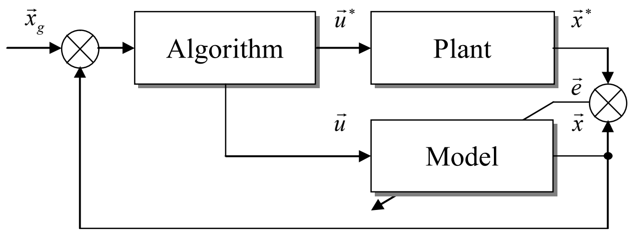


Fig. 2. Algorithms operation scheme

Here $\vec{x}_g$ is the goal state of the plant, $\vec{x}$ is the state, which is predicted by the model, $\vec{x}^*$ is the actually measured stated of the plant, $\vec{u}$ and $\vec{u}^*$ are the suggested and the best found controls respectively. System representation error $\vec{e}$ is used to adjust the model accordingly. The algorithm heavily uses the model to predict the plant state while searching for the acceptable control that would transfer it to the desired goal state $\vec{x}_g$. As soon as an acceptable control function is found, it is used to control the plant, while adjusting the model parameters comparing the predicted and its actual state. For some algorithms the searching process can continue and would most probably result in improved control function, which could be then used instead of the one found earlier.

There are three clearly distinctive modes for the algorithms operation as follows:

- Search for the control function (off-line operation, no control applied to the system).
- Application the control function to the system (on-line operation, system is under control, model is verified, control function is improving).
- Readjustments of the control function if the operation results are not acceptable.

Control function readjustment could be required due to the different reasons, including system parameters variations detected in the second mode, when the model verification is performed. For some control algorithms, as well as for others in some extreme cases, readjustment means that the search process for the acceptable control function must be started again with no acceptable control function available. As a result, the algorithm goes back into the first mode instead of the third one, which is supposedly more efficient in terms of solution time and required computational power. One should also note that it is not necessary to use the same algorithm in all modes of operation. Proper combination of different algorithms will deliver better results since it allows using the most appropriate algorithm for the task.

Evolutionary control algorithms. Evolutionary algorithms are widely used to solve multi-parametric and multi-objective optimisation problems. In conventional evolutionary algorithms, which also could be referred to as genetic algorithms, the search process is stopped as soon as the acceptable solution is found. However, in order to implement the ongoing optimal control for a dynamic system, the searching process should not be interrupted while the best acceptable solution is taken to be applied to the system. Such a modification is represented by a flowchart in the figure 3. In a sense, the evolutionary control algorithm above has no explicit ending. However, it is terminated as soon as the system has been successfully transferred to the goal state. Reaching the goal is checked all the time no matter whether the better control has been found or not (it is reflected in the figure 3 by the dotted line). Generating of controls requires additional adjustment of all of the controls in the current set with respect to the updated time of the system, which is referred to as time trimming. Although the "evolutionary" nature of the algorithms is not obvious from the flowcharts above, it becomes apparent when all of the stages are given in greater details. Let us now study in greater details major blocks that form the evolutionary algorithm with respect to its control application. During the initialization stage, predefined number N of control programs in the form (4) is generated randomly. This initial set can be either evenly distributed across the search space ("uneducated search") or having higher density in vicinity of suggested solution ("educated search"). The latter could be efficiently used especially in the control readjustment mode. During the control programs evaluation stage, every program from the current set is decoded into a control function using the instruction set (3). This function is then evaluated in terms of its capability to solve the control problem (transfer the system to the goal state) with respect to the optimisation criteria and constraints. After evaluation, the acceptable control function could be selected and then used to control the system. Moreover, based on the results of the controls evaluation, every control program in the current set of programs is assigned a "fitness" value, which defines the probability for this program to be later used to produce the next generation of programs. With or without the acceptable control program found, the evolution of the control programs is continued, and new set of control programs is generated. The process of the new controls generating is shown as a flowchart in the figure 4. In order to generate new set of programs, two encoded control programs from the current set are selected randomly using the fitness defined probabilities (the higher fitness value, the higher probability for the program to be selected). After that, application of so called genetic operations results in a new control program, which is placed to the new set. This solution generation process is repeated until N new control programs are generated, and old set can be finally discarded.

Genetic operations. There are four genetic operations that can be applied to the selected functions: crossover, mutation, inversion, and length modification. The latter operation defines the difference between the conventional genetic algorithms and VLGA. During the crossover, for each of the two selected control programs B_1 and B_2 a random index $n \in (1, \min(v1, v2))$ is generated that splits the corresponding program into two sub-programs:

$$B_1 = \{b_{11}, \dots, b_{n-1}, b_n, \dots, b_{v1}\},$$
$$B_2 = \{b_{12}, \dots, b_{n-1}, b_n, \dots, b_{v2}\}.$$

After that, two new control programs are produced by exchanging the obtained earlier sub-programs:

$$\begin{aligned} B_1^* &= \{b_{11}, \dots, b_{n1-1}\} \cup \{b_{n2}, \dots, b_{v2}\}, \\ B_2^* &= \{b_{12}, \dots, b_{n2-1}\} \cup \{b_{n1}, \dots, b_{v1}\}. \end{aligned} \tag{6}$$

Finally, only one of the resulting control programs (6) is randomly selected for the further processing. Application of the mutation operation means increasing or decreasing randomly selected instruction within the selected control program:

$$B = \{b_1, \dots, b_n, \dots, b_v\} \to B^* = \{b_1, \dots, b_n \pm 1, \dots, b_v\}. \tag{7}$$

Needless to say, that the boundary condition $b_n \in [1,\ldots,p]$ must be checked and corresponding adjustments must be applied if necessary. Next genetic operation, which is inversion, is applied with relatively small occurrence probability, and is used to provide necessary global optimum search capability for the algorithm.

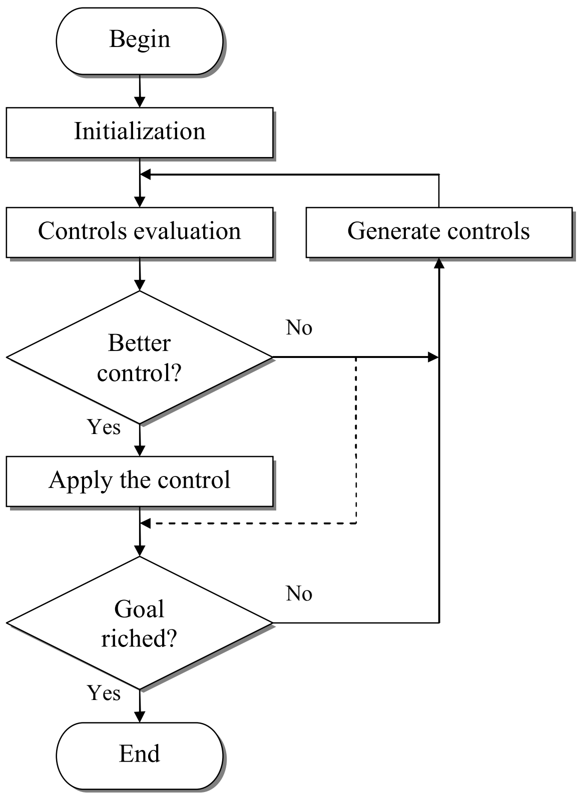


Fig. 3. Evolutionary control algorithm

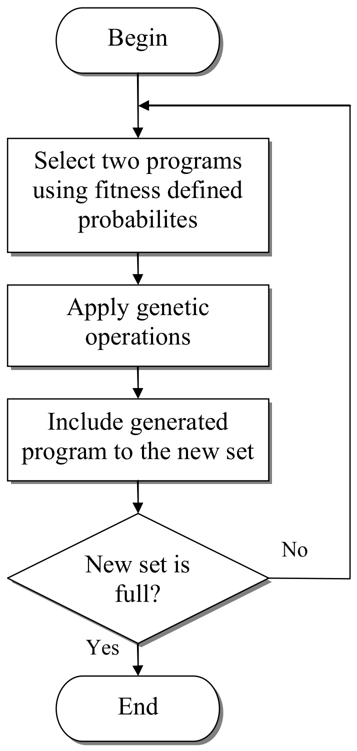

Begin
Select two programs using fitness defined probabilites
Apply genetic operations
Include generated program to the new set
New set is full?
No
Yes
End

Fig. 4. New controls generating process

First random index is generated for the selected control program, thus dividing it into two sub-programs. After that, new control program is produced by exchanging the order of the sub-programs:

$$B = \{b_1,\ldots,b_{n-1},b_n,\ldots,b_v\} \rightarrow B^* = \{b_n,\ldots,b_v,b_1,\ldots,b_{n-1}\}\,. \tag{8}$$

One should note that contrary to the crossover operation (6), mutation (7) and inversion (8) do not modify length of the subjected control program. Additional variation of the control program length is done by means of the modification operation

$$B = \{b_1,\ldots,b_v\} \rightarrow B^* = \{b_1,\ldots,b_{v*}\},\ v^* = v \pm n\,. \tag{9}$$

Here n is the number of instructions either added to or subtracted from the control program. In case of length increasing modification, added instructions are randomly selected from the instruction set. Each of the described above genetic operation is applied using its own application probability.

Fitness function. One of the problems related to the application of evolutionary algorithms to the procedural optimal control is a proper choice of the used fitness function. In a sense, finding the optimal control function requires solving a multi-objective optimisation problem. Successful fitness function must adequately represent not only control accuracy, based on which the acceptable control program can be identified, but the optimisation criterion and problem constraints as well. The simplest linear aggregation of the mentioned quantities is given by the following expression:

$$f(B) = w_C \cdot \{w_D \cdot D(\vec{x}_v,\vec{x}_g) + w_J \cdot J(\vec{x}_0 \rightarrow \vec{x}_v)\}\,. \tag{10}$$

Here w_C, w_D, and w_J are the weight coefficients for the constraints violation penalty, Euclidean distance in the state space between the final state $\vec{x}_v$ after control program execution and the goal state $\vec{x}_g$, and cost function correspondingly. In this case, fitness function is inversely proportional to the expression (10). One should note, that as soon as the successful program able to control the system to the desired state is found, which means $D(\vec{x}_v, \vec{x}_g) \to 0$, fitness value solely depends on the cost function (2). The latter provides smooth transition of the algorithm operation from the searching for any acceptable solution to the searching for the optimal one. In general, proper choice of the fitness function providing the best algorithm operation in terms of its accuracy and reliability is certainly subject for further research.

Algorithm testing. The benchmarking testing case is the control of a linear system that is defined by simple ordinary differential equations

$$\frac{d}{dt}\begin{bmatrix} X \\ V \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -0.5 \end{bmatrix} \cdot \begin{bmatrix} X \\ V \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot U, \tag{11}$$

from the state $\vec{x}_0 = \{1,1\}$ to the state $\vec{x}_g = \{0,0\}$ with respect to either minimal time or shortest trajectory in the state-space. Although the system (11) is simple and linear, it will be presented to the algorithm as a purely numerical model. The system (11) will be controlled with and without presence of constraints. Constraints are given by the following system of inequations, defined in the state space:

$$\begin{cases} (X-1.5)^2 + (V-0.5)^2 > 0.625, \\ (X-1)^2 + (V+0.25)^2 > 0.625. \end{cases}, \tag{12}$$

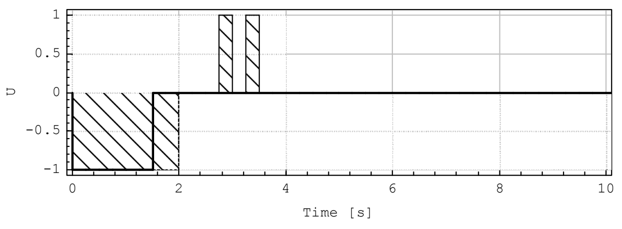
The controls are allowed to have only three admissible values $\vec{u}^0 = \{-1,0,1\}$, and the control time step is $\Delta t = 0.25$ s. Algorithm performance is suggested to be evaluated in terms of *sims*. One "sims" corresponds to a single estimation (prediction) of the system state after time Δt.

The following parameters were used for the algorithm: number of programs in the generation N = 100, initial program length $v_0 = 16$, mutation probability – 0.1, crossover probability – 0.9, modification probability – 0.1, and inversion has been disabled. Simulations results are given in the table 1 and shown in the figures from 5 to 8. In the figures 7-10 the dashed line corresponds to the first found acceptable solution and the solid line corresponds to the best found optimal solution. Transitional states are shown by means of black and white circles for the acceptable and optimal solutions correspondingly.

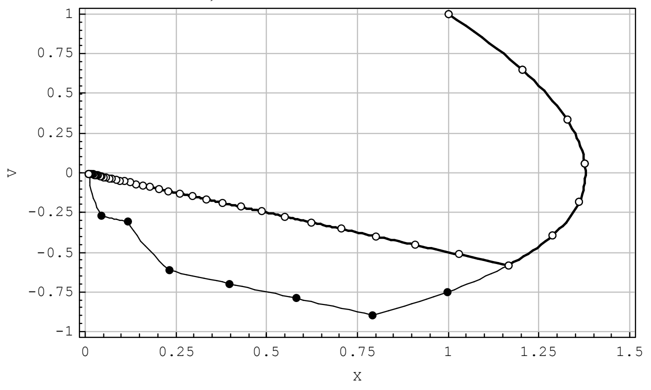
Parallelisation of the algorithm. Analysing the shown above results one should note that finding the optimal solution to the given problem using VLGA may require significant amount of simulations to be done, which results in increased solution finding time. On the other hand, real-time applications of the algorithm require the solution to be found as soon as possible. One of the approaches to this problem is to use parallelisation during the "controls evaluation" stage . In this case, every control program B_i (9) from the set of N programs is evaluated at the separate processing unit PU$_i$, which performs system simulation with respect to the given control function and calculates quality function (10). The best performance is achieved when number of PUs equals to the number of the control programs N. Using this kind of parallelisation, evaluation time of the whole set of programs equals the time of the longest program evaluation. At the same time, by increasing the number N of the control programs and corresponding PUs, the solution could be found within smaller number of iterations, while keeping the evaluation time constant.

Conclusions. From the presented in this paper results, one can see that the variable length genetic algorithm is perfectly capable of solving the procedural optimal control problem even for the numerically defined system model and constraints. At the same time, sub-optimal but acceptable control solution could be also found prior to the optimal one after significantly less number of iterations.

Although the algorithm allows its parallelisation, the following approaches to its improvement in terms of performance and quality of the obtained solution are yet to be investigated: other forms of multi-objective fitness functions, fitness weight coefficients optimisation, adaptive time step, and so on. Algorithm readjustment performances in case of varying system parameters and stochastic disturbances must be properly studied as well.

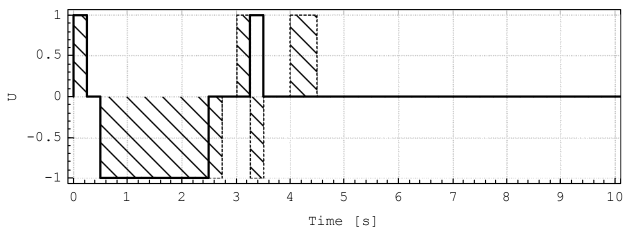


a) Control functions

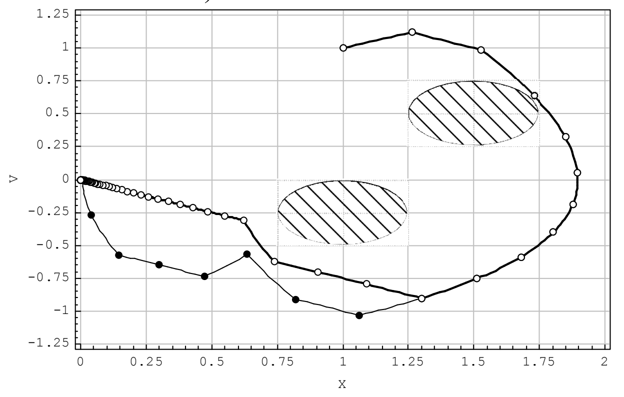


b) State-space trajectories

Fig. 5. Unconstrained minimizing trajectory length

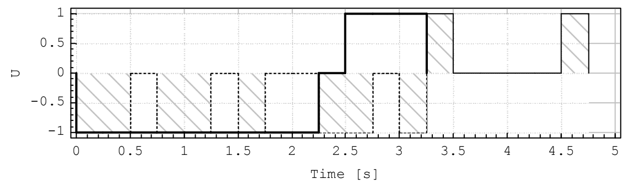


a) Control functions

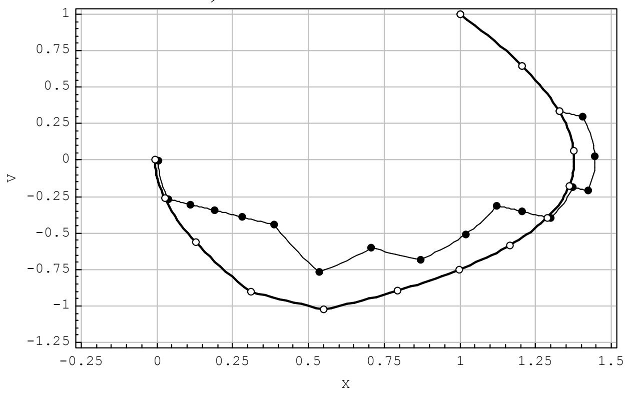


b) State-space trajectories

Fig. 6. Constrained minimizing trajectory length

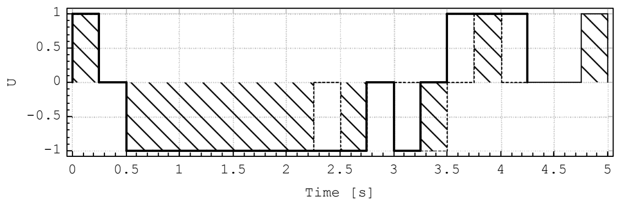


a) Control functions

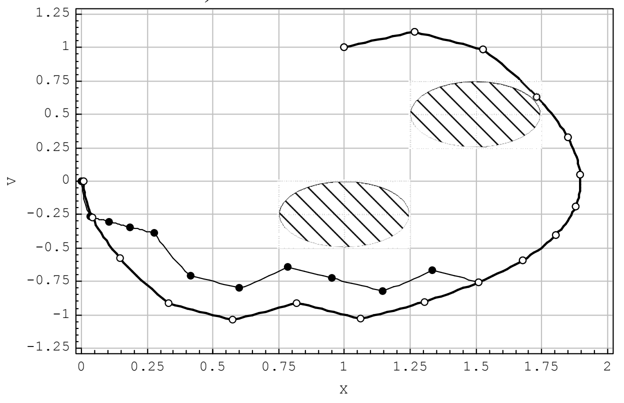


b) State-space trajectories

Fig. 7. Unconstrained control minimizing time



a) Control functions



b) State-space trajectories

Fig. 8. Constrained control minimizing time

Table 1. Algorithm performances

Problem and type of solution	**Iterations**	**Sims**	**Cost**
Minimal trajectory length			
Unconstrained acceptable	14	24 198	3.51
Unconstrained optimal	510	1 202 969	3.01
Constrained acceptable	99	141 693	4.90
Constrained optimal	8741	21 464 704	4.43
Minimal time			
Unconstrained acceptable	14	25 337	4.75
Unconstrained optimal	110	181 347	3.25
Constrained acceptable	223	305 420	5.00
Constrained optimal	3711	5 269 306	4.25

References

[1] *Holland, J.,* Adaptation in Natural and Artificial Systems*,* The MIT Press, 1992. – P. 228.

[2] *Goldberg, D. E.* Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989. – P. 372.

[3] *Gray, G. J., Murray-Smith, D. J., Li, Y., Sharman K. C., and Weinbrunner T.,* Nonlinear model structure identification using genetic programming, Control Engineering Practice. – №6(11). – 1998. – P. 1341 – 1352.

[4] *Kristinsson, K., and Dumont G. A.,* System identification and control using genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22(5), 1992, – P. 1033 – 1046.

[5] *Porter, B., and Jones, A. H.*, Genetic tuning of digital PID controllers. / Electronics Letters. – Vol. 28(9). – 1992. – P. 843 – 844.

[6] Genetic Algorithms for Optimization in Predictive Control / C. Onnen, R. Babuska, U. Kaymak, etc. // Control Engineering Practice, – 1997, Vol. 5(10). – P. 1363 – 1372.

[7] *Haber R., Schmitz U., and Bars R.,* Optimal choice of horizons for predictive control by using genetic al gorithms, Chin. J. Comput. Phys. 28. – 2004. – P. 53 – 58.

[8] *Dijkstra E. W.,* A note on two problems in connexion with graphs, // Numerische Mathematik. – №1 (1959). – P. 269 – 271.

[9] *Nilsson N. J.,* Problem solving methods in artificial intelligence, McGraw Hill. 1971. – P. 255.

[10] *V. Apostolyuk*, Application of State-Space Search Algorithms to Optimal Control, Proceedings of VI Intl. Conference on Gyro Technology, Navigation, and Motion Control. – Vol. 2. – 2007. – P. 104 – 110.

[11] *V. Apostolyuk*, Evolutionary approach to the procedural optimal control, Electronics and Control Systems. – #2(16) . – 2008. – P. 57 – 65.