applied sciences

MDPI

Article

Control Synthesis as Machine Learning Control by Symbolic Regression Methods

Elizaveta Shmalko *,† **and Askhat Diveev** †

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333 Moscow, Russia; aidiveev@mail.ru

* Correspondence: e.shmalko@gmail.com; Tel.: +7-964-636-6669

† These authors contributed equally to this work.

Abstract: The problem of control synthesis is considered as machine learning control. The paper proposes a mathematical formulation of machine learning control, discusses approaches of supervised and unsupervised learning by symbolic regression methods. The principle of small variation of the basic solution is presented to set up the neighbourhood of the search and to increase search efficiency of symbolic regression methods. Different symbolic regression methods such as genetic programming, network operator, Cartesian and binary genetic programming are presented in details. It is shown on the computational example the possibilities of symbolic regression methods as unsupervised machine learning control technique to the solution of MLC problem of control synthesis for obtaining the stabilization system for a mobile robot.

Keywords: machine learning control; control synthesis; symbolic regression; genetic programming; network operator; mobile robot

Citation: Shmalko, E.; Diveev, A. Control Synthesis as Machine Learning Control by Symbolic Regression Methods. *Appl. Sci.* **2021**, *11*, 5468. https://doi.org/10.3390/app11125468

Academic Editor: Alessandro Gasparetto

Received: 8 May 2021
Accepted: 10 June 2021
Published: 12 June 2021

1. Introduction

From the moment of its appearance, when the first regulators for machines with steam engines appeared, and in more than a century and a half of its development, the theory of automatic control has come a long way of transformation from a scattered set of control methods for mechanical, hydrodynamic and other systems to fundamental science. Almost all scientific research in control was carried out within the framework of studying the possibility of creating various innovative technical solutions from machine tools at the inception stage to automatic flying and space vehicles or nuclear power plants. The 20th century is famous for the creation of automatic control systems for industrial complexes and production processes using computers. However, the 21st century prepares new challenges associated with the emergence of universal objects, such as autonomous robots and robotic systems, capable of autonomously performing completely different tasks in different conditions and environments. Modern control systems must be able to quickly change, refine, learn. This circumstance requires both the universalization and automation of the very process of developing control systems that are not tied to the physics of the control object, operating with laws and patterns that are valid for objects of any complexity and nature. Machine learning control (MLC) meets these new challenges.

In control theory, two main tasks for machine learning are distinguished—the identification of the control object model and the synthesis of the control system. In both cases, the task involves finding an unknown function. The function can be set up to parameters, then machine-learning techniques are used only to adjust the parameters [1]. In general case, both the structure of the function and its parameters should be found.

Typically, machine learning implies the use of neural network technologies [2–6]. At first glance, it seems that a variety of neural network structures can satisfy any control problems. However, in fact, the structure of the neural network is determined by the developer and it is a given structure, in which only parameters are configured, while the

structure itself remains unchanged. It is difficult to even guess whether this structure is optimal for a given task. In addition, for complex tasks, a neural network has a complex structure, and for a development engineer who is used to describing objects and systems with some functions that have physical meaning and geometric representation, working with a neural network seems to be a kind of black box. Inspite of the popularity of the neural networks, tutorials on machine learning [7,8] indicate that neural networks are only part of machine learning and there are other learning technologies.

Today, more and more examples appear on the application of symbolic regression methods to MLC problems. Symbolic regression methods look for a control function in the form of code. The variety of symbolic regression methods is defined by different ways of coding functions. The search for the optimal control function is carried out on the code space using evolutionary algorithms. The genetic programming (GP) [9] was the first symbolic regression method capable to search for mathematical expressions. Now there is a variety of such methods [10–16].

The present paper is focused on the application of symbolic regression to the solution of the control synthesis. The successful solutions of applied control problems have so far been demonstrated mainly by the method of genetic programming [17–19]. In this paper, we want to show that there are many different symbolic regression methods that can cope with solving MLC problems. For this purpose, the mathematical formulation of the ML is introduced in Section 2 either for supervised machine learning or unsupervised one (or commonly known reinforcement learning). Then the problem of control synthesis is addressed as MLC.

All methods of symbolic regression can search for functions, so they can automate the process of synthesis of control systems, but very little are used in this direction, in view of a number of difficulties. One such setback is a complexity of the search space. The search is organized on the non-numerical space of codes of functions where only some symbolic metric can be set such as the Levenshtein, Hamming or Jaro distance. However, the estimation of the solutions during the search is performed in the space of functions with absolutely another metric. It turns out that the search process is carried out on the space of function codes, where there is no single metric. To overcome the mentioned difficulty of the search space complexity, the paper discusses how the convergence of symbolic regression methods can be significantly accelerated for solution of control synthesis problem through the application of the principle of small variations of the basic solution (Section 3).

Methods of symbolic regression are discussed in Section 4. The example section provides a detailed description of four most effective symbolic regression methods and their application to control synthesis for a mobile robot as unsupervised machine learning control.

2. Problem Statement of Control Synthesis as MLC

In general, machine learning aims to establish some functional dependency that determines the relationship between input and output data. MLC is aimed to design a control law for some object to achieve given goals optimally in terms of some formulated functional. The control law in general case is a multi-dimensional vector-function that should be defined.

Today machine learning, especially in the field of control, is mostly used for the optimal tuning of parameters, like the parameters of a neural network or some given regulator. As a result of learning, an estimate of the unknown parameter vector is received.

With the new possibilities that symbolic regression methods open up in machine learning, we are now able to define more broadly a machine learning problem that is to find an unknown function, including both the optimal structure and its parameters.

For this, let us introduce the following definitions.

Definition 1. *A set of computational procedures, that transforms a vector* $\mathbf{x}$ *from an input space* X *to a vector* $\mathbf{y}$ *from an output space* Y*, and there isn't any mathematical expression* $\mathbf{y} = \mathbf{f}(\mathbf{x})$ *for them, is called an unknown function.*

Denote the unknown function between input vector $\mathbf{x}$ and output vector $\mathbf{y}$ as

$$\mathbf{y} = \alpha(\mathbf{x}). \tag{1}$$

The unknown function (1) can be presented as some device, or collection of experimental results. It is called a black box, because the exact description of it is not known.

Let a set of input vectors be determined

$$\tilde{\mathrm{X}} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\} \subseteq \mathrm{X}. \tag{2}$$

For every input vector, an output vector is determined by the unknown function (1)

$$\tilde{\mathrm{Y}} = \{\mathbf{y}^1 = \alpha(\mathbf{x}^1), \ldots, \mathbf{y}^N = \alpha(\mathbf{x}^N)\} \subseteq \mathrm{Y}. \tag{3}$$

Definition 2. *A pair of sets of compatible dimensions*

$$(\tilde{\mathrm{X}}, \tilde{\mathrm{Y}}) \tag{4}$$

is called a training set if $\tilde{\mathrm{X}} \subseteq \mathrm{X}$, $\tilde{\mathrm{Y}} \subseteq \mathrm{Y}$ *and it is assumed that there is a one-to-one mapping* $\mathrm{X} \to \mathrm{Y}$.

Therefore, from the search of the unknown control function perspective, machine learning can be divided into two main classes: supervised learning, when there is a training sample, and unsupervised, when there is no training sample but some estimate function is given. So machine learning problems can be formulated as follows.

Definition 3. *Unsupervised machine learning consists in finding a function*

$$\mathbf{y} = \beta(\mathbf{x}, \mathbf{q}), \tag{5}$$

and parameters $\mathbf{q}$, $\mathbf{q} = [q_1 \ldots q_p]^T$*, such that for some given estimate* $\gamma : \mathrm{Y} \to \mathbb{R}^1$, $\forall \mathbf{x} \in \mathrm{X}$ *it is true*

$$\|\gamma(\mathbf{y}) - \gamma(\beta(\mathbf{x}, \mathbf{q}))\| \leq \delta, \tag{6}$$

where δ *is a small positive value.*

In machine learning control problems, such an evaluation criterion is a functional.

Definition 4. *Supervised machine learning consists in building a training set (4) and finding a function (5) such that if the total error for the training set is less than the given value* ε

$$\sum_{i=1}^{N} \|\mathbf{y}^i - \beta(\mathbf{x}^i, \mathbf{q})\| \leq \varepsilon, \tag{7}$$

then for $\forall \mathbf{x}^*$ *not included in the training set* $\mathbf{x}^* \notin \tilde{\mathrm{X}}$ *the following inequation is fulfilled*

$$\|\mathbf{y}^* - \beta(\mathbf{x}^*, \mathbf{q})\| \leq \delta, \tag{8}$$

where $\mathbf{y}^* = \alpha(\mathbf{x}^*)$.

The function $\beta(\mathbf{x}, \mathbf{q})$ in (5) includes a vector of parameters $\mathbf{q}$. Often in the control tasks the structure of this function is defined beforehand on the basis of experience or intuitively, and it is necessary to find only values of some parameters, for example, coefficients of a PID

controller [20,21] or parameters of some feedback NN-based controller [22–24]. Symbolic regression methods allow to search for both the very structure of the control function and its parameters.

The problem of finding a control function (or a control law) that depends on the state of an object in control theory is called a control synthesis problem. For example, the design task of a feedback controller is a control synthesis problem since such controllers produce control signal based on the object state. The search of such a control function should be considered as machine learning control problem.

Consider the formal statement of control synthesis problem.

A mathematical model of control object is given

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{9}$$

where $\mathbf{x}$ is a vector of the state space, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u}$ is a vector of control, $\mathbf{u} \in \mathrm{U} \subseteq \mathbb{R}^m$, U is a compact set, $m \leq n$. We limit our analysis to all those control problems where U is compact, because this encompasses real examples of control problems where the control is bounded, not infinite, and the boundary values also belong to the admissible control.

An area of initial conditions is given

$$\mathrm{X}_0 \subseteq \mathbb{R}^n. \tag{10}$$

A terminal conditions is given

$$\mathbf{x}(t_f) = \mathbf{x}^f \in \mathbb{R}^n, \tag{11}$$

where t_f is a time of reaching the terminal condition from the initial area. This time isn't given, but is limited, $t_f = \leq t^+$, t^+ is a given limited time of reaching the terminal condition.

It is necessary to find one control function in the form

$$\mathbf{u} = \mathbf{h}(\mathbf{x}) \in \mathrm{U}, \tag{12}$$

where $\mathbf{h}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, that makes the object described by the dynamic model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x})), \tag{13}$$

achieve the terminal goal $\mathbf{x}^f \in \mathbb{R}^n$ from any initial condition $\mathbf{x}^{0,i} \in \mathrm{X}_0 \subseteq \mathbb{R}^n$ with the optimal value of the given quality criterion.

$$J = \int_{\mathrm{X}_0} \cdots \int \int_{t_0}^{t_f} f_0(\mathbf{x}(t, \mathbf{x}^0), \mathbf{u}(t)) dx_1^0 \ldots dx_n^0 dt \to \min_{\mathbf{u} \in \mathrm{U}}. \tag{14}$$

The described MLC problem of control synthesis can be solved as either supervised or unsupervised machine learning control.

The supervised approach is machine learning with application of a training set. A training set for control synthesis problem can be obtained by constructing optimal trajectories from multiple points of the given initial set. For this it is necessary to solve the optimal control problem for each particular initial condition from $\mathbf{x}^{0,i} \in \mathrm{X}_0$ and to receive sets of optimal controls

$$\mathrm{U}_0 = \{\mathbf{v}(t, \mathbf{x}^{0,1}), \ldots, \mathbf{v}(t, \mathbf{x}^{0,K})\}, \tag{15}$$

and optimal trajectories

$$\tilde{\mathrm{X}} = \{\tilde{\mathbf{x}}(t, \mathbf{x}^{0,1}), \ldots, \tilde{\mathbf{x}}(t, \mathbf{x}^{0,K})\}. \tag{16}$$

Therefore, to solve the control synthesis problem and to find the control function in the form (12) it is enough to approximate by any symbolic regression method the training set (16) on a criterion

$$J_{synt} = \sum_{i=1}^{K} \sum_{j=0}^{M_i} \|\mathbf{x}(t_j, \mathbf{x}^{0,i}) - \tilde{\mathbf{x}}(t_j, \mathbf{x}^{0,i})\| \to \min_{\mathbf{h}(\mathbf{x}) \in \mathrm{U}}, \tag{17}$$

where $t_0 = 0$, $\mathbf{x}(t, \mathbf{x}^{0,i})$ is a partial solution of the Equation (13) with the initial conditions $\mathbf{x}^{0,i}$, $\tilde{\mathbf{x}}(t, \mathbf{x}^{0,i})$ is a trajectory from (16), $i \in \{1, \ldots, K\}$.

The unsupervised machine learning for control synthesis is a direct search of the control function (12) on the basis of the quality criterion minimization. The difficulty here is a huge and complex search area on a non-numerical space of codes of functions where there is no single metric. As in the space of words: there is an alphabet and words can be close, based on the assessment of symbols, but have completely different meanings, based on the semantic assessment. The proximity of the names does not correspond to the proximity of the meanings. The same is the case with the search on the space of function codes. Function evaluation works with mappings. The search is carried out on codes. Thus, the metric between the names of the functions does not correspond to the distances between the values of the functions. Probably, this can explain the fact that symbolic regression methods, despite the wide range of their capabilities, have not yet emerged as a powerful tool in solving the problem of machine learning control. Most likely, the introduction of additional mechanisms is required to facilitate and accelerate the search for optimal solutions. One such mechanism can be a principle of small variations of the basic solution.

3. Small Variations of the Basic Solution

Searching for an optimal solution in the space of codes is complicated by the fact that this space does not have a metric measure. For such search spaces it is impossible to use evolutionary algorithms with arithmetic operations. Evolutionary algorithms are the search engine in all symbolic regression methods. Most of known evolutionary algorithms include arithmetic operations to transform possible solutions and produce evolution. Therefore, genetic algorithm is a main searching algorithm on the space of codes, that doesn't use arithmetic operations in its steps.

Studies of this problem have led to the formulation of the principle of small variations of the basic solution. According to this principle, the search for the mathematical expression can be started in the neighbourhood of one given possible solution. This solution is coded by a symbolic regression method and it is called a basic solution. The essence of the principle is that for a code of function, many possible small variations are determined. A small variation is such a minor change in the code that leads to the appearance of a new possible solution. According to the principle, others possible solutions are coded as sets of small variations of this basic solution. To obtain any other possible solution it is necessary to apply a vector of small variations to the basic solution. Genetic operations are applied to the vector of variations. After some generations the basic solution is changed on the best current solution. Such an approach is very convenient for search of control systems, because there are many specialists in control that can create good control system intuitively or on the basis of their experience. This control system can be considered as a basic solution.

The code of small variation is an integer vector with three or four components depending on the method of symbolic regression. These components include information about the location of the element to be changed and its new value.

For example, to record a small variation in Cartesian genetic programming it is enough to use an integer vector with three components

$$\mathbf{w} = [w_1 \; w_2 \; w_3]^T, \tag{18}$$

where w_1 is a number of column in the code, w_2 is a number of line in the column w_1, w_3 is a new value of element.

The vector of small variation in the network operator method consists of four elements

$$\mathbf{w} = [w_1\ w_2\ w_3\ w_4]^T, \tag{19}$$

where w_1 is a type of variation, w_2 is the line number, w_3 is the column number of the network operator matrix, and w_4 is a new value of an element.

Possible solutions are coded as sets of small variation vectors

$$\mathrm{W}^i = \{\mathbf{w}^{i,1}, \ldots, \mathbf{w}^{i,d}\} \tag{20}$$

where d is a length or depth of variations.

Genetic algorithm performs evolution on the sets of small variation vectors.

For example, to perform the operation of crossover two possible solutions are selected

$$\begin{array}{rcl} \mathrm{W}^\alpha & = & \{\mathbf{w}^{\alpha,1}, \ldots, \mathbf{w}^{\alpha,d}\} \\ \mathrm{W}^\beta & = & \{\mathbf{w}^{\beta,1}, \ldots, \mathbf{w}^{\beta,d}\} \end{array} \tag{21}$$

A point of crossover is determined randomly $k_c \in \{1, \ldots, d\}$.

New possible solutions are received by exchanging tails of the selected possible solutions starting from the crossover point

$$\begin{array}{l} \mathrm{W}^\gamma = \{\mathbf{w}^{\alpha,1}, \ldots, \mathbf{w}^{\alpha,k_c}, \mathbf{w}^{\beta,k_c+1}, \ldots, \mathbf{w}^{\beta,d}\} \\ \mathrm{W}^\delta = \{\mathbf{w}^{\beta,1}, \ldots, \mathbf{w}^{\beta,k_c}, \mathbf{w}^{\alpha,k_c+1}, \ldots, \mathbf{w}^{\alpha,d}\} \end{array} \tag{22}$$

To receive the symbolic regression code of possible solution G_i and evaluate the performance of this solution, the set of small variations is applied to the basic solution

$$\mathrm{G}_i = \mathbf{w}^{i,d} \circ \ldots \circ \mathbf{w}^{i,1} \circ \mathrm{G}_0, \tag{23}$$

where G_0 is a code of the basic solution.

Such an approach might seem like a double coding of the possible solutions but it provides two benefits. First, the use of the basic solution provides a guideline for searching in a complex space of functions and significantly speeds up the search. Secondly, the application of the operations of the genetic algorithm not to the codes of possible solutions directly, but to the vectors of variations, allows to always get the correct codes of possible solutions without the need to introduce additional checks.

So the principle of small variations can be applied to any known symbolic regression method to overcome challenges of solving control synthesis problem.

4. Symbolic Regression Methods

All methods of symbolic regression encode the mathematical expression and search for optimal solution on the space of codes by the genetic algorithm. To encode a mathematical expression it is needed to create the base sets of elementary functions. These sets will be alphabets for codding, and their elements are letters for creating words or codes.

The base sets of functions can be combined by the number of arguments. The following base sets are possible:

- a set of arguments or a set of functions without arguments

$$\mathrm{F}_0 = \{f_{0,1}, \ldots, f_{0,n+p+v}\} = \{x_1, \ldots, x_n, q_1, \ldots, q_p, e_1, \ldots, e_v\}, \tag{24}$$

  where x_1,...,x_n are variables, q_1,...,q_p are parameters, e_1,...,e_v are unit elements for functions of two arguments;

- a set of functions with one argument

$$\mathrm{F}_1 = \{f_{1,1}(z) = z, f_{1,2}(z), \ldots, f_{1,w}(z)\}, \tag{25}$$

where $f_{1,1}(z)$ is an identity function that is often needed for codding;

- a set of functions with two arguments

$$\mathrm{F}_2 = \{f_{2,1}(z_1, z_2), \ldots, f_{2,v}(z_1, z_2)\}. \tag{26}$$

All functions with two arguments have to possess the following properties:

- be commutative

$$f_{2,i}(z_1, z_2) = f_{2,i}(z_2, z_1),\ i = 1, \ldots, v \tag{27}$$

- be associative

$$f_{2,i}(z_1, f(z_2, z_3)) = f_{2,i}(f_{2,i}(z_1, z_2), z_3), \tag{28}$$

- have a unit element

$$f_{2,i}(z, e_i) = f_{2,i}(e_i, z) = z, \tag{29}$$

where e_i is a unit element for the function, $i = 1, \ldots, v$.

These base sets (24)–(26) are generally enough to describe any mathematical expression of the control function based on the Kolmogorov–Arnold representation (or superposition) theorem, which states that any complex function can be represented as a combination of one-dimensional functions [25,26].

Let us consider an example of codding of a mathematical expression by different symbolic regression methods.

Let the mathematical expression be given

$$y = (q_1 x_1^2 + q_2 x_2^2) \exp(-q_2 x_1) \sin(q_1 x_2 + x_3). \tag{30}$$

To present this mathematical expression the following base sets are enough:

- the set of arguments of the mathematical expression (30)

$$\mathrm{F}_0 = \{f_{0,1} = x_1, f_{0,2} = x_2, f_{0,3} = x_3, f_{0,4} = q_1, f_{0,5} = q_2, f_{0,6} = 0, f_{0,7} = 1\}; \tag{31}$$

- the set of functions with one argument

$$\mathrm{F}_1 = \{f_{1,1}(z) = z, f_{1,2}(z) = z^2, f_{1,3}(z) = -z, f_{1,4}(z) = \exp(z), f_{1,5}(z) = \sin(z)\}; \tag{32}$$

- the set of functions with two arguments

$$\mathrm{F}_2 = \{f_{2,1}(z_1, z_2) = z_1 + z_2, f_{2,2}(z_1, z_2) = z_1 z_2\}. \tag{33}$$

4.1. The Genetic Programming

Genetic programming (GP) is the first invented and the most popular method of symbolic regression [9].

GP codes the mathematical expression in the form of computational tree. Figure 1 shows the computational tree for the example mathematical expression (30).

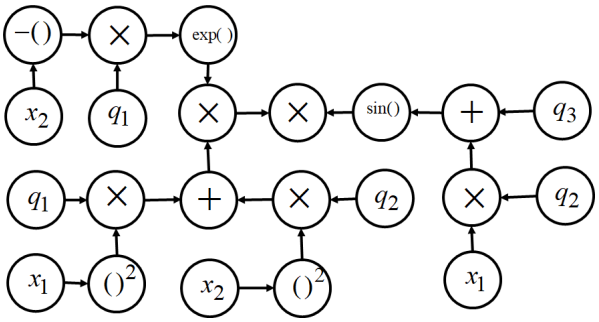

Figure 1. The computational tree for the function (30).

In the computer memory the mathematical expression is written in the form of an ordered multi-set of integer vectors with two components, the first component is the number of arguments of function, the second component is a function number.

The code of GP for the example is

$$C_{GP} = \left(\begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 4 \end{bmatrix} \right). \quad (34)$$

Despite the popularity of the method, it has a computational weakness. The code of GP has different length for different mathematical expressions. In this regard, other methods considered below seem to us more attractive from a computational point of view.

4.2. The Network Operator

The network operator method (NOP) codes a mathematical expression in the form of oriented graph [15]. Source-nodes correspond to arguments of the mathematical expression, other nodes correspond to functions of two arguments, and arcs correspond to functions with one argument.

The graph of NOP for the mathematical expression (30) is presented in Figure 2.

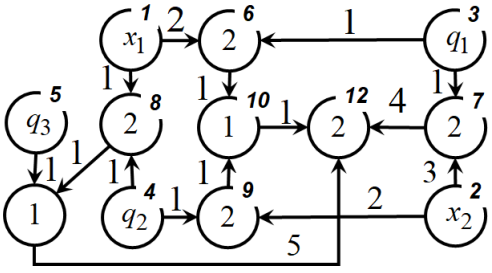

Figure 2. The NOP graph of the function (30).

In the graph (Figure 2), numbers near arcs are numbers of functions with one argument. Numbers inside nodes (except source nodes) are numbers of functions with two arguments. Numbers in upper parts of the nodes are the node numbers.

In the computer memory the NOP-code of the mathematical expression is presented in the form of an integer matrix. The NOP matrix of the mathematical expression (30) is

$$C_{NOP} = \begin{bmatrix} 0&0&0&0&0&2&0&1&0&0&0&0 \\ 0&0&0&0&0&0&3&0&2&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&2&0&0&0&1&0&0 \\ 0&0&0&0&0&0&2&0&0&0&0&4 \\ 0&0&0&0&0&0&0&2&0&0&1&0 \\ 0&0&0&0&0&0&0&0&2&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&5 \\ 0&0&0&0&0&0&0&0&0&0&0&2 \end{bmatrix} \tag{35}$$

In the matrix of NOP, each line corresponds to a node. Nonzero numbers in the main diagonal are numbers of functions with two arguments. Other numbers are numbers of functions with one argument.

4.3. Cartesian Genetic Programming

Cartesian genetic programming (CGP) encodes consecutive calls of elementary functions [13]. To encode a call, the sets of elementary functions is united

$$\mathrm{F} = \mathrm{F}_1 \cup \mathrm{F}_2. \tag{36}$$

For the example the following set of elementary functions is obtained

$$\begin{gathered} \mathrm{F} = \{f_1(z) = z, f_2(z) = z^2, f_3(z) = -z, \\ f_5(z) = f_4(z) = \exp(z), \sin(z), f_6(z_1, z_2) = z_1 + z_2, \\ f_7(z_1, z_2) = z_1 z_2\}. \end{gathered} \tag{37}$$

CGP encodes every call in the form of an integer vector. The first component of this code is a number of the element from the set (36). Other components are numbers of elements from the set of arguments. As soon as the elementary function is calculated, the result is included into the set of arguments (24) of the mathematical expression, therefore, after every calculation, the number of elements in the set of arguments is increased. A number of components of integer vector is equal to the largest number of arguments and plus one for function number.

To encode the example function (30) three components are enough. The CGP code of the mathematical expression from the example has the following form.

$$\begin{gathered} C_{CGP} = \left(\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 7 \\ 10 \\ 4 \end{bmatrix}, \right. \\ \begin{bmatrix} 6 \\ 11 \\ 9 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 7 \\ 13 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 14 \\ 1 \end{bmatrix}, \begin{bmatrix} 7 \\ 2 \\ 3 \end{bmatrix}, \\ \left. \begin{bmatrix} 6 \\ 16 \\ 5 \end{bmatrix}, \begin{bmatrix} 5 \\ 17 \\ 2 \end{bmatrix}, \begin{bmatrix} 7 \\ 18 \\ 12 \end{bmatrix}, \begin{bmatrix} 7 \\ 18 \\ 12 \end{bmatrix} \right). \end{gathered} \tag{38}$$

4.4. Binary Complete Genetic Programming

The binary complete genetic programming (BCGP), unlike the genetic programming, has codes of the same length for different mathematical expressions [10]. The graph of BCGP for the example (30) is presented in Figure 3. In the BCGP graph, nodes correspond to functions with two arguments. Leaves correspond to arguments of the mathematical expression. Arcs correspond to functions with one argument. Each level has a certain number of elements equal to a power of two. In the code of BCGP extra functions are entered for obtaining complete binary tree.

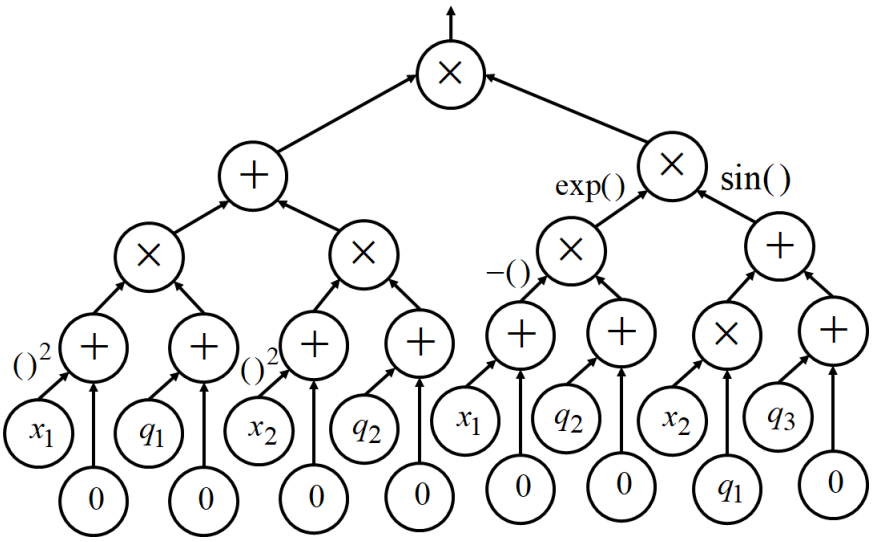

Figure 3. The BCGP graph of the function (30).

In the computer memory, the BCGP code is presented in the form of an integer array with determined number of elements. The number of elements depends on the number of levels in the complete binary tree.

The BCGP code for the example is

$$\begin{aligned} C_{BCGP} \;=\; & \{1, \\ & 2, \\ & 1\ 1 \\ & 1\ 2 \\ & 1\ 1\ 4\ 5 \\ & 2\ 2\ 2\ 2\ 1 \\ & 1\ 1\ 1\ 1\ 3\ 1\ 1\ 1 \\ & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 2\ 1 \\ & 2\ 1\ 1\ 1\ 2\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ & 1\ 6\ 3\ 6\ 2\ 6\ 4\ 6\ 1\ 6\ 4\ 6\ 1\ 3\ 5\ 6\} \end{aligned} \tag{39}$$

The last level contains numbers of elements from the set of arguments (31), where 6 is the number of unit element for the sum function $f_{2,1}(z_1, z_2) = z_1 + z_2$.

5. Computational Experiment

Now let us consider the solution of the control synthesis problem as MLC with the considered symbolic regression methods. Consider the problem of stabilization system synthesis for a mobile robot Khepera II [27]. The mathematical model of the control object is

$$\begin{aligned} \dot{x}_1 &= 0.5(u_1 + u_2)\cos(x_3), \\ \dot{x}_2 &= 0.5(u_1 + u_2)\sin(x_3), \\ \dot{x}_3 &= 0.5(u_1 - u_2), \end{aligned} \tag{40}$$

where $\mathbf{x} = [x_1\ x_2\ x_3]^T$ is a vector of state, $\mathbf{u} = [u_1\ u_2]^T$ is a control vector.

The control has restrictions

$$-10 \le u_i \le 10,\ i = 1,2. \tag{41}$$

The given set of initial conditions included 26 elements:

$$\begin{gathered}\bar{X}_0 = \{[-2.5\ \ -2.5\ \ -5\pi/12]^T, [-2.5\ \ -2.5\ \ 0]^T,\\ [-2.5\ \ -2.5\ \ 5\pi/12]^T, [-2.5\ \ 0\ \ -5\pi/12]^T,\\ [-2.5\ \ 0\ \ 0]^T, [-2.5\ \ 0\ \ 5\pi/12]^T, [-2.5\ \ 2.5\ \ -5\pi/12]^T,\\ [-2.5\ \ 2.5\ \ 0]^T, [-2.5\ \ 2.5\ \ 5\pi/12]^T, [0\ \ -2.5\ \ -5\pi/12]^T,\\ [0\ \ -2.5\ \ 0]^T, [0\ \ -2.5\ \ 5\pi/12]^T, [0\ \ 0\ \ -5\pi/12]^T,\\ [0\ \ 0\ \ 5\pi/12]^T, [0\ \ 2.5\ \ -5\pi/12]^T, [0\ \ 2.5\ \ 0]^T,\\ [0\ \ 2.5\ \ 5\pi/12]^T, [2.5\ \ -2.5\ \ -5\pi/12]^T,\\ [2.5\ \ -2.5\ \ 0]^T, [2.5\ \ -2.5\ \ 5\pi/12]^T,\\ [2.5\ \ 0\ \ -5\pi/12]^T, [2.5\ \ 0\ \ 0]^T,\\ 2.5\ \ 0\ \ 5\pi/12]^T, [2.5\ \ 2.5\ \ -5\pi/12]^T,\\ [2.5\ \ 2.5\ \ 0]^T, [2.5\ \ 2.5\ \ 5\pi/12]^T\}.\end{gathered} \tag{42}$$

The terminal conditions were set as one point

$$\mathbf{x}^* = [x_1^*\ x_2^*\ x_3^*]^T = [0\ 0\ 0]^T. \tag{43}$$

It is necessary to find a control function in the form

$$u_i = h_i(x_1^* - x_1, x_2^* - x_2, x_3^* - x_3, r_1, r_2, r_3), \tag{44}$$

where r_1, r_2, r_3 are constant parameters, $i = 1,2$, such that a robot from all 26 initial conditions (42) got to terminal condition (43) with minimal total time and high accuracy.

For solution of this problem, the network operator method, the Cartesian genetic programming, and the complete binary genetic programming were used with the principle of small variation of the basic solution.

Proportional controllers for each variable were used as a basic solution in all algorithms. The operations of addition and multiplication were used as binary operations, and a set of 28 smooth elementary functions was used as unary operations. The description of these functions can be found in the supplementary material to [28].

The network operator found the following control law

$$u_i = \begin{cases} 10, \text{ if } \tilde{u}_i > 10 \\ -10, \text{ if } \tilde{u}_i < -10 \\ \tilde{u}_i, \text{ otherwise} \end{cases},\ i = 1,2, \tag{45}$$

where

$$\tilde{u}_1 = \rho_{16}(A) + \operatorname{sgn}(x_3^* - x_3) + \rho_{19}(r_3(x_3^* - x_3)) + \sqrt[3]{D} + (\rho_{23}(A) + D)^{-1}, \tag{46}$$

$$\begin{gathered}\tilde{u}_2 = \rho_{23}(\sin(x_1^* - x_1) + \rho_{19}(r_2(x_2^* - x_2)) +\\ \tanh(r_1(x_1^* - x_1)) + r_2(x_2^* - x_2) + r_3(x_3^* - x_3)) +\\ \tilde{u}_1 + \operatorname{sgn}(x_1^* - x_1) + \rho_{16}(A) + \arctan(B) + \sin(E),\\ A = x_1^* - x_1 + \tanh(\tanh(r_1(x_1^* - x_1)) +\end{gathered} \tag{47}$$

$$r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) + r_3(x_3^* - x_3)) +$$
$$\sin(x_1^* - x_1) + \rho_{19}(r_2(x_2^* - x{+}2)) +$$
$$\tanh(r_1(x_1^* - x_1)) + r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) + r_3(x_3^* - x_3),$$
$$B = \rho_{19}(r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1)) +$$
$$\tanh(r_1(x_1^* - x_1)) +$$
$$r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) + r_3(x_3^* - x_3)) +$$
$$\arctan(r_1(x_1^* - x_1)) + \rho_4(\sin(x_1^* - x_1) +$$
$$\mathrm{sgn}(A) + \sqrt[3]{\sqrt[3]{x_1^* - x_1} + A} + \rho_4(r_1(x_1^* - x_1)) +$$
$$\tanh(z_7) + r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) +$$
$$r_3(x_3^* - x_3) + \mathrm{sgn}(z_{12}) + \sqrt[3]{x_1^* - x_1} + A,$$
$$C = \rho_9(x_3^* - x_3) + \arctan(r_1) + \mathrm{sgn}(r_1(x_1^* - x_1)) +$$
$$+\rho_{23}(\sin(x_1^* - x_1) + B +$$
$$\rho_{19}(r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1)) +$$
$$\tanh(r_1(x_1^* - x_1)) +$$
$$r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) + r_3(x_3^* - x_3)),$$
$$D = \sin(r_3(x_3^* - x_3)) + \sin(x_1^* - x_1) +$$
$$\rho_{19}(r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) +$$
$$\tanh(r_1(x_1^* - x_1)) + r_2(x_2^* - x_2)\mathrm{sgn}(x_1^* - x_1) +$$
$$r_3(x_3^* - x_3) + \left(\sqrt[3]{x_1^* - x_1} + A\right)^3 + \tanh(C),$$
$$E = \rho_{16}(A) + \mathrm{sgn}(x_3^* - x_3) + \rho_{19}(r_3(x_3^* - x_3)) +$$
$$\sqrt[3]{D} + (\rho_{23}(A) + D)^{-1},$$
$$\rho_{16}(z) = \begin{cases} z, \text{ if } |z| < 1 \\ \mathrm{sgn}(z), \text{ otherwise} \end{cases},$$
$$\rho_{19}(z) = \mathrm{sgn}(z)\exp(-|z|),$$
$$\rho_{23}(z) = z - z^3, \;\; \rho_4(z) = \mathrm{sgn}(z)\sqrt{|z|},$$
$$\rho_9(z) = \begin{cases} 1, \text{ if } z > 0 \\ 0, \text{ otherwise} \end{cases},$$

$r_1 = 14.72876$, $r_2 = 2.02710$, $r_3 = 4.02222$.

The Cartesian genetic programming found the control law as (45), where

$$\tilde{u}_1 = G + H + \rho(G), \tag{48}$$

$$\tilde{u}_2 = H - G - \rho(G), \tag{49}$$

$$G = r_1(x_3^* - x_3) + \sigma_(x_1^* - x_1)(x_2^* - x_2)),$$
$$H = 2(x_1^* - x_1) + \mathrm{sgn}(x_1^* - x_1)r_2,$$
$$\rho(\alpha) = \begin{cases} \mathrm{sgn}(\alpha)B^+, \text{ if } |\alpha| > -\log(\delta^-) \\ \mathrm{sgn}(\alpha)(\exp(|\alpha|) - 1) \end{cases},$$

$$\sigma(\alpha) = \mathrm{sgn}(\alpha)\sqrt{|\alpha|},$$

$r_1 = 3.1094$, $r_2 = 3.6289$, $B^+ = 10^8$, $\delta^- = 10^{-8}$.

The complete binary genetic programming found the following solution (45), where

$$\begin{aligned} \tilde{u}_1 \quad = \quad & -\exp(2r_2)+ \\ & \left((x_3^* - x_3)\mathrm{sgn}(x_2^* - x_2)\sqrt{|x_2^* - x_2|}\right)^3 + \\ & (r_1 + x_3^* - x_3)(x_1^* - x_1)\cos(x_3^* - x_3)+ \\ & (r_1+1)\mathrm{sgn}(x_2^* - x_2)/(x_3^* - x_3)\times \\ & \sqrt{|(r_1+1)(x_2^* - x_2)/(x_3^* - x_3)|}, \end{aligned} \tag{50}$$

$$\begin{aligned} \tilde{u}_2 \quad = \quad & (r_2+1)(\arctan((x_2^* - x_2)/r_2)- \\ & \sin(x_3^* - x_3)) + (W^3 - W)\times \\ & (r_2^3 - r_2 + 1 - \exp + (r_1)), \end{aligned} \tag{51}$$

$$W = \frac{1 - \exp(x_2^* - x_2)}{1 + \exp(x_2^* - x_2)} + \sin(r_1 + 1),$$

$r_1 = 3.9356$, $r_2 = 2.6748$.

The trajectories of the robot moving from eight initial conditions (52) to the terminal position (43) are presented in Figures 4–6.

$$\begin{aligned} \bar{X}_0(8) = \{&[-2.5 \ -2.5 \ -5\pi/12]^T, [-2.5 \ -2.5 \ 5\pi/12]^T, \\ &[-2.5 \ 2.5 \ -5\pi/12]^T, [-2.5 \ 2.5 \ 5\pi/12]^T, \\ &[2.5 \ -2.5 \ -5\pi/12]^T, [2.5 \ -2.5 \ 5\pi/12]^T, \\ &[2.5 \ 2.5 \ -5\pi/12]^T, [2.5 \ 2.5 \ 5\pi/12]^T\}. \end{aligned} \tag{52}$$

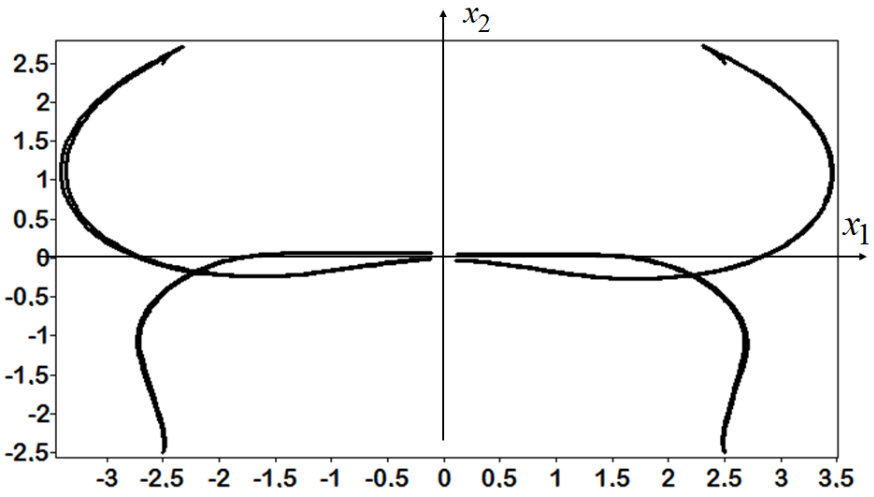

Figure 4. Trajectories of the robot with control law trained by NOP.

The goal of experiments was to show that computational symbolic regression methods allow to obtain a control function that, when substituted into the right-hand sides of a system of differential equations of the control object, makes this object stable. As a result, it was demonstrated that various symbolic regression methods can successfully solve this machine learning problem without laborious construction of a training set, basing only on the criterion for minimizing the quality functional.

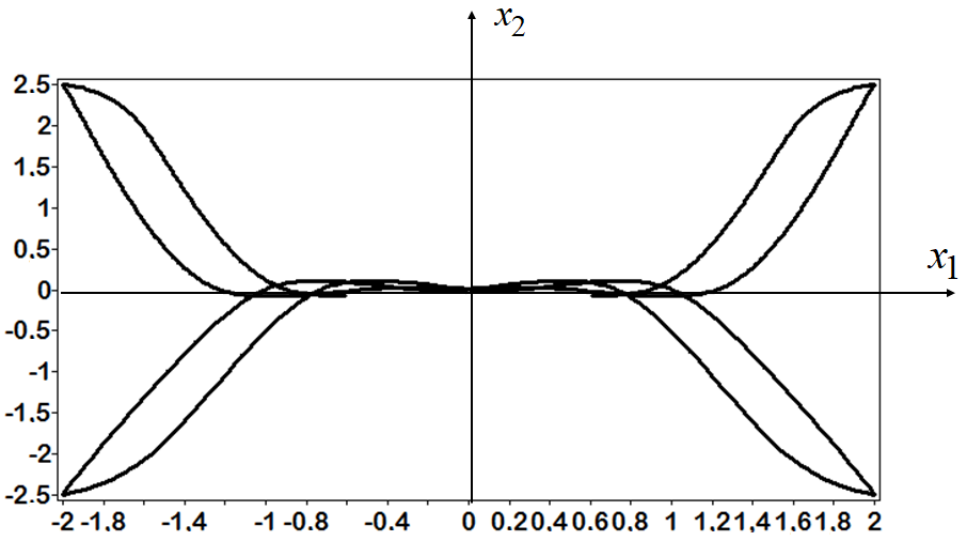

Figure 5. Trajectories of the robot with control law trained CGP.

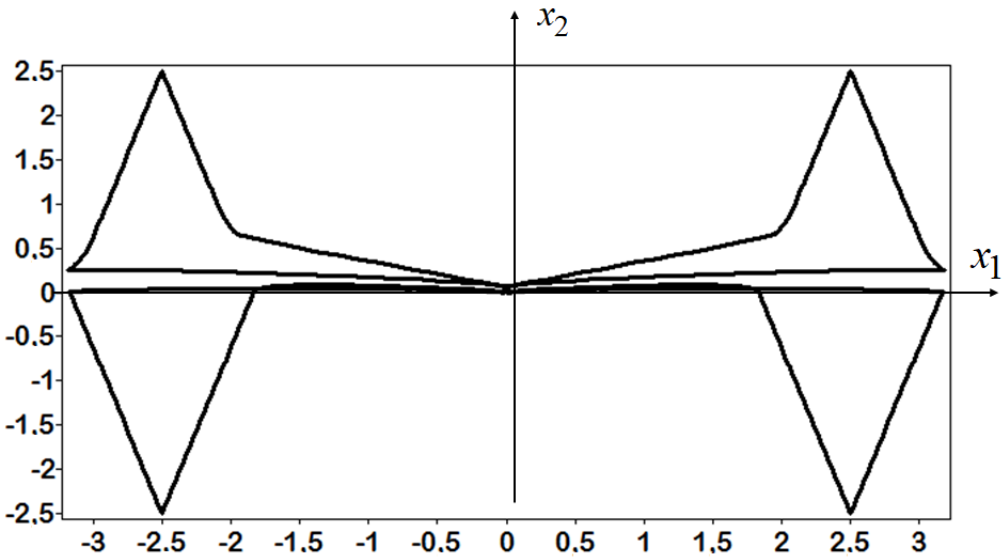

Figure 6. Trajectories of the robot with control law trained by BCGP.

6. Conclusions

The paper considered symbolic regression methods for machine learning control tasks. The theoretical formalization of MLC is proposed. The paper focused on the control synthesis problem as MLC. The scope of application of symbolic regression methods for supervised and unsupervised machine learning is discussed. The principle of small variations is considered to overcome the computational difficulties of symbolic regression methods associated with the complexity of the search space. Computational example of control synthesis for a mobile robot demonstrates the capabilities and prospects of such symbolic regression methods as network operator, Cartesian genetic programming and binary complete genetic programming as unsupervised machine learning control technique. Thus, the presented possibilities of various methods of symbolic regression in the field of machine learning control open up new perspectives in the field of control associated with the departure from manual and analytical search for solutions and the transition to machine search and machine learning control.

Author Contributions: conceptualization, A.D. and E.S.; methodology, A.D.; software, A.D.; validation, E.S.; formal analysis, A.D.; investigation, E.S.; data curation, E.S.; writing—original draft preparation, A.D. and E.S.; writing—review and editing E.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2020-799.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fleming P.J.; Purshouse, R.C. Evolutionary algorithms in control systems engineering: A survey. *Control Eng. Pract.* **2002**, *10*, 1223–1241. [CrossRef]
2. Narendra, K.S.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 4–27. [CrossRef] [PubMed]
3. Bukhtoyarov, V.; Tynchenko, V.; Petrovskiy, E.; Bashmur, K.; Kukartsev, V.; Bukhtoyarova, N. Neural network controller identification for refining process. *J. Phys. Conf. Ser.* **2019**, *1399*, 044095. [CrossRef]
4. Moe, S.; Rustad, A.M.; Hanssen, K.G. Machine Learning in Control Systems: An Overview of the State of the Art. In *Artificial Intelligence XXXV. SGAI 2018, LNCS*; Bramer, M., Petridis, M., Eds.; Springer: Cham, Switzerland, 2018.
5. Barto, A.G. Reinforcement learning control. *Curr. Opin. Neurobiol.* **1994**, *4*, 888–893. [CrossRef]
6. Pham, D.T.; Xing, L. *Neural Networks for Identification, Prediction and Control*; Springer: London, UK, 1995.
7. Chollet, F. *Deep Learning with Python*, 2nd ed.; Manning Publications: Shelter Island, NY, USA, 2020.
8. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
9. Koza, J.R. *Genetic Programming II*; MIT Press: Cambridge, MA, USA, 1994.
10. Diveev, A.; Shmalko, E. Complete binary variational analytic programming for synthesis of control at dynamic constraints. *ITM Web Conf.* **2017**, *10*, 02004. [CrossRef]
11. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
12. Zelinka, I. Analytic programming by Means of Soma Algorithm. In Proceedings of the 8th International Conference on Soft Computing, Mendel, Brno, Czech Republic, 5–7 June 2002; Volume 10, pp. 93–101.
13. Miller, J.F. *Cartesian Genetic Programming*; Springer: Berlin/Heidelberg, Germany, 2011.
14. Luo, C.; Zhang, S.L. Parse-matrix evolution for symbolic regression. *Eng. Appl. AI* **2012**, *25*, 1182–1193. [CrossRef]
15. Diveev, A.I. Numerical method for network operator for synthesis of a control system with uncertain initial values. *J. Comp. Syst. Sci. Int.* **2012**, *51*, 228–243. [CrossRef]
16. Nikolaev, N.; Iba, H. Inductive Genetic Programming. In *Adaptive Learning of Polynomial Networks. Genetic and Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2006; 25–80.
17. Dracopoulos, D.C.; Kent, S. Genetic programming for prediction and control. *Neural Comput. Appl.* **1997**, *6*, 214–228. [CrossRef]
18. Derner, E.; Kubalík, J.; Ancona, N.; Babuška, R. Symbolic Regression for Constructing Analytic Models in Reinforcement Learning. *Appl. Soft Comput.* **2020**, *94*, 1–12. [CrossRef]
19. Duriez, T.; Brunton, S.L.; Noack B.R. *Machine Learning Control–Taming Nonlinear Dynamics and Turbulence*; Springer International Publishing: Cham, Switzerland, 2017.
20. Saad, M.; Jamaluddin, H.; Mat Darus, I. PID controller tuning using evolutionary algorithms. *WSEAS Trans. Syst. Control* **2012**, *7*, 139–149.
21. Urrea-Quintero, J.; Hernández-Riveros, J.; Muñoz-Galeano, N. Optimum PI/PID Controllers Tuning via an Evolutionary Algorithm. In *PID Control for Industrial Processes*; Shamsuzzoha, M., Ed.; IntechOpen: London, UK, 2007; pp. 43–69.
22. Ted Su, H.; Tariq Samad. Chapter 10—Neuro-Control Design: Optimization Aspects. In *Neural Systems for Control*; Omidvar, O., Elliott, D.L., Eds.; Academic Press: San Diego, CA, USA, 1997; pp. 259–288.
23. Jurado, F.; Lopez, S. Chapter 4—Continuous-Time Decentralized Neural Control of a Quadrotor UAV. In *Artificial Neural Networks for Engineering Applications*; Alanis, A.Y., Arana-Daniel, N., López-Franco, C., Eds.; Academic Press: Cambridge, MA, USA, 2019; pp. 39–53.
24. Yichuang Jin; Pipe, T.; Winfield, A. 5—Stable Manipulator Trajectory Control Using Neural Networks. In *Neural Systems for Robotics*; Omidvar, O., van der Smagt, P., Eds.; Academic Press: Boston, MA, USA, 2007; pp. 117–151.
25. Kolmogorov, A. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Proc. USSR Acad. Sci.* **1956**, *108*, 179–182.
26. Arnold, V. Arnold, V. On functions of three variables. *Am. Math. Soc. Transl.* **1963**, *28*, 51–5.
27. Šuster, P.; Jadlovská, A. Tracking Trajectory of the Mobile Robot Khepera II Using Approaches of Artificial Intelligence. *Acta Electrotech. Inform.* **2011**, *11*, 38–43. [CrossRef]
28. Diveev, A., Shmalko, E. Machine-Made Synthesis of Stabilization System by Modified Cartesian Genetic Programming. *IEEE Trans. Cybern. (Early Access)* **2020**, 1–11. [CrossRef]