mathematics

MDPI

Article

Machine Learning Feedback Control Approach Based on Symbolic Regression for Robotic Systems

Askhat Diveev * and Elizaveta Shmalko

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

Abstract: A control system of an autonomous robot produces a control signal based on feedback. This type of control implies the control of an object according to its state that is mathematically the control synthesis problem. Today there are no universal analytical methods for solving the general synthesis problem, and it is solved by certain particular approaches depending on the type of control object. In this paper, we propose a universal numerical approach to solving the problem of optimal control with feedback using machine learning methods based on symbolic regression. The approach is universal and can be applied to various objects. However, the use of machine learning methods imposes two aspects. First, when using them, it is necessary to reduce the requirements for optimality. In machine learning, optimization algorithms are used, but strictly optimal solutions are not sought. Secondly, in machine learning, analytical proofs of the received properties of solutions are not required. In machine methods, a set of tests is carried out and it is shown that this is sufficient to achieve the required properties. Thus, in this article, we initially introduce the fundamentals of machine learning control, introduce the basic concepts, properties and machine criteria for application of this technique. Then, with regard to the introduced notations, the feedback optimal control problem is considered and reformulated in order to add to the problem statement that such a property adjusts both the requirements of stability and optimality. Next, a description of the proposed approach is presented, theoretical formulations are given, and its efficiency is demonstrated on the computational examples in mobile robot control tasks.

Keywords: control synthesis; optimal control; stabilization; symbolic regression; machine learning; evolutionary algorithm; mobile robot

MSC: 49M25; 68T05

Citation: Diveev, A.; Shmalko, E. Machine Learning Feedback Control Approach Based on Symbolic Regression for Robotic Systems. *Mathematics* **2022**, *10*, 4100. https://doi.org/10.3390/math10214100

Academic Editor: Liliya Demidova

Received: 9 October 2022
Accepted: 31 October 2022
Published: 3 November 2022

1. Introduction

Aiming at the automation of processes, we intend to automate the very process of control systems development in order to make it fast and generic. This sounds especially relevant in the context of ever-increasing robotization and the emergence of a variety of robots as control objects. To reach this goal of all-round automation, it is necessary to generalize the needed tasks, that is, to formulate them in general mathematical statements, and then develop universal methods for solving them. However, the problem here is that despite the extensive theoretical background of control theory, today, there is a wide range of applied problems that do not have exact analytical solutions. At the same time, there is an objective need for solving them.

In fact, in robotics, most modern control systems for robots are programmed by hand, and engineers do not even set the general problems because there are no general ways to solve them. The developer, based on his experience, sets the structure of the control system, determines the control channels, types of regulators, and then adjusts the parameters of the given system so that they meet certain requirements [1]. However, every problem can

and should be considered an optimal one, defining not only the parameters, but also the structure of the control system optimally and, again, automatically.

If a robot has to perform rather simple actions, for example, moving from one point to another and going around some obstacles, then the program of its control system contains supposedly several hundreds of lines. In more complex control tasks, the programs that must control robots can include several tens or hundreds of thousands of lines. These programs will grow as the tasks or the robots structure become more complex. One can assume that a control system for a robot that repeats the actions of a fly must contain some millions of lines. It follows from that stated above that the manual creation of the robot control system is an unpromising direction. It is necessary to automate this process.

Any problem for robots, as well as any other control objects, can be formulated as a mathematical optimization problem, such as a problem of providing stability, an optimal control problem for finding optimal path in current real conditions, a problem of stabilization of movement along of the optimal path, a problem of avoiding collisions with static and dynamic obstacles, the problem of interaction with other control objects, the problem of precise achievement of some given terminal conditions and so on. The most general problem in robotics is feedback control synthesis. It assumes that a control system that makes the object reach its goal is designed as a function of the object state optimally according to given criteria. Even if the optimal control problem is solved and the optimal path is found, we must further ensure the movement of the object along the obtained trajectory to compensate for possible ever-existing uncertainties.

The general synthesis problem was formulated back in the early 1960s by Bellman [2,3], where the continuous-time nonlinear optimal control problem was solved through the Hamilton–Jacobi–Bellman equation, which is a nonlinear partial differential equation. Even in simple cases, the HJB equation may not have global analytic solutions. Various numerical methods based on dynamic programming have been proposed in the literature [4–7], including the modern adaptive dynamic programming technique [8–10] and reinforcement learning [11–13]. However, the main drawback of dynamic programming methods today is still the computational complexity required to describe the value function, which grows exponentially with the dimension of its domain.

A different way to construct a feedback optimal control is firstly to solve an optimal control problem by direct methods of nonlinear programming or by the indirect approach of the Pontryagin maximum principle and then to synthesize a feedback stabilization system in order to supply movement along the received optimal trajectory. For example, in [14], points are placed on the trajectory, and the object is stabilized at these points. This is the most popular practical approach to feedback optimal control system design.

However, concerning the optimality criterion, this approach is not correct since it turns out that the optimal path is considered for one control object, and the introduced stabilization system changes the object so that the calculated path may not be optimal for the modified object model. In addition, when approaching a given point on the path, the system slows down, so it is necessary to carry out additional estimates in each specific task, according to the optimal moments of points switching.

In this work, we propose an inverse approach to feedback optimal control system synthesis. The general idea is the following. We firstly stabilize an object according to some point in the state space by solving the stabilization system synthesis problem. Note that this problem is computationally easier than the general synthesis problem. The stabilization task can be solved by a plain variety of methods depending on the complexity of the object model, particularly analytical methods of backstepping [15,16] or the analytical design of aggregated controllers [17], or synthesis based on the application of the Lyapunov function [18,19], as well as any classical methods for liner systems, such as modal control [20], differently tuned PID controllers [21], and fuzzy [22] and neural network [23] controllers. In the overwhelming majority of cases, the control synthesis problem is solved analytically or technically by taking into account the specific properties

of the mathematical model. Today, modern numerical machine learning methods can be applied to find a solution for generic dynamic objects [24].

This new paradigm of machine learning control [25,26] allows to find some good near optimal solutions in a limited amount of time. However, due to the novelty of these methods, it becomes necessary to substantiate the results obtained by machine learning. In this paper, we introduce definitions of some machine properties of the system. We introduce the definition of machine learning control from our point of view, give machine proof of the existence of a specific property in some mathematical model, refine the definition of the feasibility property of the mathematical model of the control object and present the extended statement of the optimal control problem.

The addition of the stabilization system into the object model gives it a new property: at each moment of time, the object has a point of equilibrium. Thus, in the synthesized optimal control approach, the uncertainty in the right parts is compensated by the stability of the system relative to a point in the state space. Near the equilibrium point, all solutions converge. Now, we can solve the problem of optimal control through the optimal position of the equilibrium point. The found synthesized optimal control can be realized in the real object directly without additional feedback stabilization loops.

The paper is structured in the following order. After the introduction, the theoretical base of machine learning control is presented in Section 2, introducing the main definitions and machine criteria for justification of the results received by machine numerical methods. Next, in Section 3, we formulate the mathematical statement of the problem of feedback machine learning control, extending the optimal control problem statement with additional requirements. Then in Section 4, the paper proposes a synthesized approach to the solution of the stated feedback optimal control problem. Algorithms for its solution are considered in Sections 5 and 6, and computational examples of solving control problems for mobile robots are presented in Section 7. In the experimental part, the computational examples of synthesized control application for solving feedback control problems in the class of feasible controls for mobile robots are presented.

2. Theoretical Base of Machine Learning Control

Summarizing various definitions of machine learning [27–29], we can conclude that machine learning is an inexact numerical solution of some mathematical optimization problem, that is, the solution obtained by machine learning differs from the exact one by some known value but satisfies the researcher, and it can be improved with continuing learning. In all cases, different optimization algorithms are used for machine learning, but for these algorithms, it is enough to find a near optimal solution.

Let us introduce some definitions.

Definition 1. *The machine learning problem is a search of an unknown function.*

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

where $\mathbf{y}$ is a vector of function values, $\mathbf{y} \in \mathbb{R}^r$, $\mathbf{x}$ is a vector of arguments, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{q}$ is a vector of constant parameters, $\mathbf{q} \in Q \subseteq \mathbb{R}^p$,

$$\alpha(\mathbf{x}, \mathbf{q}) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^r. \tag{2}$$

This function during training approximates some data set, which is called the training sample:

$$J = \sum_{i=0}^{N} \|\hat{\mathbf{y}}^i - \alpha(\mathbf{x}^i, \mathbf{q})\|, \tag{3}$$

where $\hat{Y} = \{\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^N\}$ is a training sample.

With unsupervised learning, this function is used for the minimization of some functional

$$J = \int_0^{t_f} f_0(\mathbf{x}(t), \alpha(\mathbf{x}(t), \mathbf{q}))dt, \tag{4}$$

where t_f is the goal achievement time.

Definition 2. *Machine learning is finding a solution of the optimization problem in the given* Δ *neighborhood of the optimal solution.*

The peculiarity of machine learning is that learning does not require the exact achievement of minimum criterion (3) or (4):

$$J_1 \leq \min J + \Delta^*, \tag{5}$$

where Δ^* is a given positive value determining a functional value achievable during learning. For criterion (3), a minimum value is equal to zero. For criterion (4), the minimal value can be unknown. Then, the limit minimum value can be used instead:

$$J_1 = J^- + \Delta^*, \tag{6}$$

where $J^- \leq \min J$.

If, as a result of learning, the found function (1) must acquire some properties, then the proof of the presence of these properties is confirmed by simulation.

$$J_2 = \sum_{i=0}^{K} \vartheta(\phi(\alpha(\mathbf{x}^i, \mathbf{q}))), \tag{7}$$

where $\vartheta(z)$ is the Heaviside function

$$\vartheta(z) = \begin{cases} 1, \text{if } z > 0 \\ 0, \text{otherwise} \end{cases}, \tag{8}$$

$\phi(\mathbf{x}, \mathbf{q})$ is a condition that determines whether a function property exists

$$\phi(\mathbf{x}, \mathbf{q}) \leq 0, \tag{9}$$

where K is a number of consecutive experiments performed with a positive result (9), set to prove the presence of a property.

Definition 3. *Machine learning control is a search of control function.*

Machine learning searches for a function that, for some sets of arguments, returns the required values. Note that there can be many such functions, and all they can have various structures and parameter values.

According to the introduced Definitions 1–3, an optimization problem of the control function search must be formulated for machine learning control. A solution of this problem is not optimal, as the found function gives a value of the quality criterion close to optimal one. On the one hand, this might reduce the importance of the solution found, but on the other hand, it allows for solving very complex problems.

Let us be given a mathematical model of a control object. This model can be derived from physical laws or identified by some machine learning technique [30,31]. Generally, this model is described by a system of ordinary differential equations with a free control vector in the right-hand side:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{10}$$

where $\mathbf{x}$ is a state vector, $\mathbf{u}$ is a control vector,

$$\begin{array}{rcl} \mathbf{x} & = & [x_1 \dots x_n]^T, \\ \mathbf{u} & = & [u_1 \dots u_m]^T,\ m \leq n, \\ \mathbf{f}(\mathbf{x}, \mathbf{u}) & = & [f_1(\mathbf{x}, \mathbf{u}) \dots f_n(\mathbf{x}, \mathbf{u})]^T. \end{array} \tag{11}$$

The problem of control, including machine learning control, is to find a control function instead of the control vector

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

to make the differential equation system

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

acquire some new properties. For example, these can be such properties as stability, the optimality of solutions, and others.

In machine learning, the control function of these new properties of the control object has to be checked by a computer as well.

When the control function is derived analytically, then the system is guaranteed to have the desired property. In the case of machine learning control, events occur when the system does not have the desired property. Let us call them bad events. For example, the robot reaches the terminal position from almost all initial conditions, but does not reach it from some other initial condition. Although such events are rare with good training, they can occur, and the probability of its occurrence is not known. We also need to introduce some estimate when we can consider that the probability of the bad event is small, and we can consider learning to be successful, i.e., assume that the system has obtained the desired property.

The appearance of bad events is due to the presence of various uncertainties and disturbances in the system. According to Lyapunov [32], the existing uncertainties can be considered uncertainties in the initial conditions.

Let us formulate a machine criterion of obtaining some property by a differential equation system. To define the property of the whole system (13), it is enough to set a quantity K of partial solutions that obtain this property.

Definition 4. *If D experiments are carried out and in every i experiment, K_i partial solutions of the differential equation perform the required property from any $M_i \geq K_i$ randomly selected initial conditions from the initial domain, and*

$$\lim_{D \to \infty} \sum_{i=1}^{D} \frac{K_i}{M_i} \to 1, \tag{14}$$

the existence of this property for the differential equation in this domain is proven by machine.

In other words, as the number of experiments increases, the probability of such a bad event, when the system does not have the desired property, tends to zero. From a mathematical point of view, this means that all private solutions for the domain of initial conditions have this property, except solutions for a subset of a zero measure.

Now we can redefine some properties of differential equations into appropriate machine properties.

Let the computer check the new properties in terminal time interval, $(0; t^+)$.

Let, in the state space of differential equation system (13), a manifold of the dimension $n - s$ be defined by

$$\phi_i(\mathbf{x}) = 0,\ i = 1, \dots, s. \tag{15}$$

Definition 5. *In some domain* $X \in \mathbb{R}^n$*, the following properties are performed: for given quantity* K *of initial conditions* $\mathbf{x}^{0,i} \in X$, $i = 1, \ldots, K$ *for the partial solution* $\mathbf{x}(t, \mathbf{x}^{0,i})$ *of differential Equation (13)* $\exists t'$, $0 < t' \leq t^+$*. Then,*

$$\|\phi(\mathbf{x}(t', \mathbf{x}^{0,i}))\| \leq \Delta,\ i = 1, \ldots, K, \tag{16}$$

where $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ldots \phi_s(\mathbf{x})]^T$*, and* $\forall t' < t \leq t^+$

$$\|\phi(\mathbf{x}(t, \mathbf{x}^{0,i}))\| < \Delta,\ i = 1, \ldots, K. \tag{17}$$

Then, differential equation system (13) is machine stable on a bound time interval $(0; t^+)$ *relative to the manifold (15).*

If a dimension of the manifold equals to 0, then a machine stable equilibrium point is obtained. Coordinates of this point in the state space are determined from solving the algebraic equation system,

$$\phi_i(\mathbf{x}) = 0,\ i = 1, \ldots, n. \tag{18}$$

The definition of machine stability uses a manifold (15) that can be expressed from the partial solution. Let $\mathbf{x}(t, \mathbf{x}^0)$ be a partial solution of differential Equation (13):

$$\mathbf{x}(t, \mathbf{x}^0) = [\tilde{x}_1(t) \ldots \tilde{x}_n(t)]^T. \tag{19}$$

Let us solve one component (19) relative to t. Let it be the last component:

$$t = \omega(\tilde{x}_n). \tag{20}$$

After inserting Equation (20) in solution (19), a one-dimensional manifold is received:

$$x_i(\omega(\tilde{x}_n), \mathbf{x}^0) - \tilde{x}_i(\omega(\tilde{x}_n)) = 0,\ i = 1, \ldots, n-1. \tag{21}$$

Machine stability relative to the manifold (21) is the machine stability of solution (19) of differential Equation (13).

Now consider the equilibrium points of some generic differential equation:

$$\dot{\mathbf{x}} = \mathbf{w}(\mathbf{x}), \tag{22}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$.

Analytically, the equilibrium points are defined as solutions of the system of algebraic equations:

$$\mathbf{w}(\mathbf{x}) = 0. \tag{23}$$

Machine-determined equilibrium points $\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^K$ are the points that satisfy the following condition:

$$\|\mathbf{w}(\tilde{\mathbf{x}}^i)\| \leq \varepsilon_1 \tag{24}$$

and $\forall \tilde{\mathbf{x}}^i, \tilde{\mathbf{x}}^j, i \neq j$,

$$\|\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\| > \varepsilon_1, i, j \in \{1, \ldots, K\}, \tag{25}$$

where ε_1 is a given small positive number.

Definition 6. *An equilibrium point* $\tilde{\mathbf{x}} \in \mathbb{R}^n$ *of differential Equation (22) is stable if there is a domain* $X_0 \subseteq \mathbb{R}^n$, $\tilde{\mathbf{x}} \subset X_0$ *such that it contains a sphere* S

$$\sum_{i=1}^{n} (x_i - \tilde{x}_i)^2 = r^2, \tag{26}$$

where $r > \varepsilon_1$ *is located completely in this domain* X_0 $S \subset X_0$, *and* $\forall \mathbf{x}^0 \in S$, *a partial solution* $\mathbf{x}(t, \mathbf{x}^0)$ *of differential Equation (22), will reach the point* $\mathbf{x} \in S$ *for limited time*

$$\|\mathbf{x}(t_f, \mathbf{x}^0) - \tilde{\mathbf{x}}\| \leq \varepsilon_1, \tag{27}$$

where $t_f < t^+ < \infty$.

The sphere is introduced here in order to guarantee that the equilibrium point is inside the region and exclude it from falling on the boundary.

The introduced machine interpretations of the known properties of objects eliminate the need to analytically prove the existence of these properties for an object since this is often very laborious or completely impossible. This allows further solving complex technical problems by machine methods and checking the achievement of the required properties by machine.

3. Machine Learning Feedback Control

Recall our goal. We want to automate the design of an automatic control system. For this purpose, it is necessary to formulate for the computer the control problem and make the computer solve it automatically and design a control system for a control object without human.

To do this, let us formulate the problem in a general mathematical setting of optimal control.

The mathematical model of the control object is given in the form of differential equation system (10).

The initial condition is given:

$$\mathbf{x}(0) = \mathbf{x}^0 \in \mathbb{R}^n. \tag{28}$$

Given the terminal position as a goal,

$$\mathbf{x}^f = [x_1^f \dots x_n^f]^T. \tag{29}$$

The quality criterion is given in the form of an integral functional:

$$J_1 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{u}) dt \to \min. \tag{30}$$

It is necessary to find a control function in the form

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, t). \tag{31}$$

where $\mathbf{g}(\mathbf{x}, t) = [g_1(\mathbf{x}, 1) \dots g_m(\mathbf{x}, t)]^T$, which makes object (10) achieve given goal (29) with the optimal value of quality criterion (30). A found control function (31) has to satisfy the boundaries:

$$u_i^- \leq g_i(\mathbf{x}, t) \leq u_i^+, \quad i = 1, \dots, m. \tag{32}$$

We are looking for control as a function of the state of the object, which corresponds to the principle of feedback control. It is generally accepted that this type of control is implemented in real systems since it allows leveling the inaccuracies of the model.

Definition 7. *For a mathematical model to correspond to a dynamic real object, it is necessary and sufficient that the mathematical estimation error of the real object state does not increase over time.*

That is, the introduction of the feedback control to the differential equation system gives the system some property that allows the object to achieve the goal with the optimal quality value, that is, to be feasible. The question is, what is this property?

It is clear that not all control systems are feasible. For example, optimal but open-loop control systems do not have the feasibility property. Conversely, Lyapunov-stable systems are feasible. However, there are examples when the solution is not Lyapunov stable but at the same time it is feasible [32]. For example, when moving through points, the movement itself to a point is Lyapunov stable, but movement along a trajectory consisting of points is not Lyapunov stable, but this control is now most often implemented. Thus, it becomes necessary to formulate a property that makes it possible to determine the feasibility of the system.

In fact, by introducing a feedback system, we change the differential equations of the system so that a certain area appears around some particular solution of the system (the optimal trajectory) such that other trajectories that fall into this area will not leave it.

This trajectory is a partial solution of differential equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}, t)) \tag{33}$$

for the found optimal control.

Definition 8. *The partial solution* $\mathbf{x}(t, \mathbf{x}^0)$ *of differential Equation (22) has a compressibility property if, for any other partial solution* $\mathbf{x}(t, \mathbf{x}^*)$*, the following conditions are performed.*

If

$$\|\mathbf{x}(t', \mathbf{x}^0) - \mathbf{x}(t', \mathbf{x}^*)\| \leq \sigma, \tag{34}$$

where $t' > 0$, $\sigma > 0$, *then* $\exists \alpha > 0$ *such, that for any* $\varepsilon^+ > 0$

$$\|\mathbf{x}(t' + \alpha, \mathbf{x}^0) - \mathbf{x}(t' + \alpha, \mathbf{x}^*)\| \leq \varepsilon^+. \tag{35}$$

Hypothesis 1. *To realize the found optimal control function (31) in the real control object, the optimal trajectory must compressibility properties (34) and (35).*

Obviously, if a control function provides performing properties (34) and (35), then this control function according to Definition 8 can be realized in the real object directly. According to Definition 8, an unstable differential equation cannot be realized. Highly unstable systems exist, but they cannot be described by unstable differential equations because these differential equations cannot estimate the state of unstable objects in time. Any small error in the initial conditions for the unstable differential equation of a mathematical model will be increasing over time. To estimate the state of an unstable object, it is necessary to use a stable differential equation.

Thus, to solve the stated feedback optimal control problem, it is necessary to construct such a control function (31) that makes the object (10) achieve given goal (29) with the optimal value of quality criterion (30) and obtain required properties (34) and (35).

4. Synthesized Optimal Control Approach for the Solution of the Stated Problem

In this section, we propose our synthesized optimal control approach [33] that completely satisfies requirements (34) and (35) in the construction of optimal control (31).

The idea of the approach consists in providing the object with the existence of some equilibrium point in the state space and then constructing such a control function that controls the position of the equilibrium point in order to make the object reach the goal with the optimal value of the quality criterion.

Initially, the control synthesis problem is solved to provide the existence of the equilibrium point. As a result, the control function in the following form is found:

$$\mathbf{u} = \mathbf{h}(\mathbf{x}^* - \mathbf{x}), \tag{36}$$

where $\mathbf{x}^*$ in each fixed moment of time is some point in the state space that affects the position of the equilibrium point of the differential equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x})), \tag{37}$$

$$\mathbf{h}(\mathbf{x}^* - \mathbf{x}) = [h_1(\mathbf{x}^* - \mathbf{x}) \dots h_m(\mathbf{x}^* - \mathbf{x})]^T. \tag{38}$$

The control function (38) must satisfy restrictions for any position of the point $\mathbf{x}^*$

$$u_i^- \le h_i(\mathbf{x}^* - \mathbf{x}) \le u_i^+, \ \ i = 1, \dots, m. \tag{39}$$

For any value $\mathbf{x}^*$, the differential equation system (37) has an equilibrium point $\tilde{\mathbf{x}}(\mathbf{x}^*)$:

$$\mathbf{f}(\tilde{\mathbf{x}}(\mathbf{x}^*), \mathbf{h}(\mathbf{x}^* - \tilde{\mathbf{x}}(\mathbf{x}^*))) = 0. \tag{40}$$

A matrix of Jacobi

$$\mathbf{A}(\mathbf{x}^*) = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{x}^* - \mathbf{x})}{\partial \mathbf{x}}, \tag{41}$$

computed in the equilibrium point $\tilde{\mathbf{x}}(\mathbf{x}^*)$ has all eigenvalues in the left part of the complex plane.

$$\det(\mathbf{A}(\mathbf{x}^*) - \lambda \mathbf{E}) = \prod_{i=1}^{n} (\lambda - \lambda_j) = 0, \tag{42}$$

where

$$\lambda_j = \alpha_j + i\beta_j, \tag{43}$$

$\alpha_j < 0, j = 1, \dots, n, i = \sqrt{-1}$.

In many cases, the equilibrium point $\tilde{\mathbf{x}}$ coincides with the point $\mathbf{x}^*$, but in some cases, it is impossible. For example, if the differential equation system includes an equation $\dot{x}_k = x_l$, then the component x_k of the equilibrium point will have only value 0 for any values of components x_k^*.

Note that when this control synthesis problem is solved by some machine learning method, conditions (41) and (42) cannot be checked for each mathematical expression $\mathbf{h}(\mathbf{x}^* - \mathbf{x})$ of the control function because these are very time-consuming procedures. In machine learning control, to prove the stability in an equilibrium point, Definition 6 is used.

To synthesize control function (36), it is necessary to determine domain $\mathrm{X} \in \mathbb{R}^n$ and then to determine equilibrium point $\tilde{\mathbf{x}}$. If the equilibrium point is equal to point $\mathbf{x}^*$, then the control function is searched in the form of (36), where $\mathbf{x}^* = \tilde{\mathbf{x}}$.

Computationally, to provide a stable property of equilibrium point $\tilde{\mathbf{x}}$, the synthesis problem (10)–(12) is solved with the terminal point $\mathbf{x}^f = \tilde{\mathbf{x}}$, the initial domain $\mathrm{X}_0 \subset \mathrm{X}$, and the quality criterion

$$J = \max\{t_{f,1}, \dots, t_{f,K}\} + a_1 \sum_{i=1}^{K} \Delta_{f,i} \to \min, \tag{44}$$

where a_1 is the weight coefficient,

$$\Delta_{f,i} = \left\| \mathbf{x}^f - \mathbf{x}(t_{f,i}, \mathbf{x}^{0,i}) \right\|, \tag{45}$$

where $t_{f,i}$ is the time of achievement of the terminal position (29) from the initial condition $\mathbf{x}^{0,i}$ of the set of initial conditions $\mathrm{X}_0 = \{\mathbf{x}^{0,1}, \dots, \mathbf{x}^{0,K}\}, i \in \{1, \dots, K\}$,

$$t_{f,i} = \begin{cases} t, & \text{if } t < t^+ \text{ and } \Delta_{f,i} \le \varepsilon \\ t^+ & \text{otherwise} \end{cases}, \tag{46}$$

where t^+ and ε are given positive values, $\mathbf{x}(t, \mathbf{x}^{0,i})$ is a partial solution of the system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^f - \mathbf{x})), \tag{47}$$

for initial conditions $\mathbf{x}(t_0) = \mathbf{x}^{0,i}$, $i \in \{1, \ldots, K\}$,

$$\left\| \mathbf{x}^f - \mathbf{x} \right\| = \sqrt{\sum_{i=1}^{n} (x_i^f - x_i)^2}. \tag{48}$$

In the second stage, the following optimal control problem is solved. The mathematical model of the control object is given in the form of (37), and the initial conditions are given as (28). It is necessary to find control as a function of time:

$$\mathbf{x}^* = \mathbf{v}^*(t), \tag{49}$$

in order to minimize the functional

$$J_2 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{x}^* - \mathbf{x}) dt \to \min_{\mathbf{x}^* \in \mathrm{X}}. \tag{50}$$

The obtained control

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, t) = \mathbf{h}(\mathbf{v}^*(t) - \mathbf{x}) \tag{51}$$

allows performing conditions (34) and (35); therefore, it can be realized in the real object.

Further in the paper, we discuss the machine learning methods appropriate to solve the described problems and show the examples of applying the proposed approach to the solution of two different robotic tasks.

5. Symbolic Regression for Machine Learning

According to the introduced Definitions 1, 2 and 3, the task of searching for the needed control function (36) in the first step is to be considered a machine learning task.

A search of an unknown function consists in searching for the structure and parameters of this function. Usually structures of the functions are set by a researcher on the base of data analysis, experience, or intuition. Today different universal structures become popular such as various mathematical series and artificial neural networks. If a structure of the needed function is set, then machine learning searches for the optimal values of parameters according to some criterion [34].

An ML technique such as symbolic regression allows to look for the optimal structure of the needed function and parameters as well [35].

Symbolic regression methods have made huge strides over the past decade and recently, the importance of interpretable machine learning has been recognized by the wider scientific community. However, to a greater extent, symbolic regression methods are used for so-called supervised machine learning, when there are some data that need to be approximated [36–39].

The considered problem of machine learning for control does not have a training set, and the search for a control function must be based on minimizing the quality criterion. This approach, in conventional terminology, refers to unsupervised learning. In this direction, there are much fewer examples due to the complexity of the search. In [40–42], the control functions are searched as linear combinations of basic functions, and mainly smooth functions are used as basic functions. We perform the control function search [43,44] in the form of function nesting, which allows to obtain more complex mathematical expressions, and also use a wider set of basic functions, including discontinuous functions.

All symbolic regression methods code the searched mathematical expression in the form of special code and search for the optimal solutionon the space of codes by a special genetic algorithm. For this purpose, a special crossover operation is developed. The application of a special crossover operation for two codes of parents allows to receive two new codes of child chromosomes. Different crossover operations are used for different code forms.

A complex crossover operation in symbolic regression methods, in our opinion, makes it difficult to find a solution. Creating new possible solutions as a result of a complex crossover operation is similar to generating new possible solutions. Therefore, the search process does not use the properties of evolution and is more like a random search. In order for the search algorithms of symbolic regression methods to have metaheuristic evolutionary properties, it is necessary that new possible solutions obtained by transforming existing possible solutions have the property of inheritance.

Definition 9. *The evolutionary algorithm has an inheritance property, if among the new possible solutions obtained, as a result of the evolutionary transformations of existing possible solutions, named parents, at least a given part of the new possible solutions have functional values, which differ from the functional values of the parents by not more than a given value.*

A universal approach to provide the inheritance property to any symbolic regression algorithm is using the principle of small variations of the basic solution [45]. The application of this principle makes it possible to find solutions that are close to optimal in a reasonable time.

In [24], this principle was applied to Cartesian genetic programming, and it improved the search process of the optimal solution. In the present paper, in the experimental part, the network operator method [46] is used, which was developed exactly for the solution of the control synthesis problem and was the first method where the principle of small variations was applied.

6. Hybrid Algorithm for Optimal Control Problem

The second step of the proposed approach (49) is essentially a pure optimization problem. Today, most generic optimization algorithms are based on population search [47], and so we also use them as a main technique, but according to the task, any other optimization algorithm can also be appropriate.

For the most complex optimal control problems with complex phase constraints, we propose to use a hybrid algorithm that combines GA [48], GWO [49] and PSO [50]. As we experimentally noticed, such a combination of the evolutionary algorithms allows to avoid the local minimum in complex tasks. A pseudocode of the algorithm can be found in Appendix A.

7. Computational Experiment

To demonstrate the proposed synthesized approach for the machine learning feedback control problem solution, let us consider two different optimal control tasks with mobile robots in complex environments with phase constraints.

7.1. Two Mobile Robots with Bottlenecks Phase Constraints

The first task we considered was to make two robots switch places with each other while accurately passing through the given areas, as if through bottlenecks.

The mathematical model of the control object has the following form:

$$\begin{array}{rcl} \dot{x}_1 & = & 0.5(u_1+u_2)\cos(x_3), \\ \dot{x}_2 & = & 0.5(u_1+u_2)\sin(x_3), \\ \dot{x}_3 & = & 0.5(u_1-u_2), \\ \dot{x}_4 & = & 0.5(u_3+u_4)\cos(x_6), \\ \dot{x}_5 & = & 0.5(u_3+u_4)\sin(x_6), \\ \dot{x}_6 & = & 0.5(u_3-u_4), \end{array} \tag{52}$$

where x_1, x_2, and x_3 are coordinates of the state vector of the first mobile robot, x_4, x_5, and x_6 are coordinates of the state vector of the second mobile robot, u_1 and u_2 are components of the control vector for the first robot, and u_3 and u_4 are components of the control vector for the second robot.

The values of control are limited:

$$-10 = u_i^- \le u_i \le u_i^+ = 10, \quad i = 1,\ 2,\ 3,\ 4. \tag{53}$$

The initial and terminal conditions are given:

$$\mathbf{x}(0) = \mathbf{x}^0 = [0\ 0\ 0\ 10\ 10\ 0]^T, \tag{54}$$

$$\mathbf{x}(t_f) = \mathbf{x}^f = [10\ 10\ 0\ 0\ 0\ 0]^T. \tag{55}$$

The quality functional is given:

$$J_3 = t_f + p_1\|\mathbf{x}^f - \mathbf{x}(t_f)\| + p_2\int_0^{t_f}\vartheta(\chi(\mathbf{x}))dt + p_3\sum_{i=1}^{k_g}\sum_{j=1}^{2}\vartheta(\Delta_{i,j}-\varepsilon_i) \to \min, \tag{56}$$

where

$$t_f = \begin{cases} t, \text{if } t < t^+, \ \texttt{and} \ \|\mathbf{x}^f - \mathbf{x}(t_f)\| < \varepsilon_0 \\ t^+, \text{otherwise} \end{cases}, \tag{57}$$

$$\chi(\mathbf{x}) = r_0 - \sqrt{(x_1-x_4)^2+(x_2-x_5)^2}, \tag{58}$$

$\vartheta(\alpha)$ is a Heaviside step function

$$\vartheta(\alpha) = \begin{cases} 1, \text{if } \alpha \ge 0 \\ 0, \text{otherwise} \end{cases}, \tag{59}$$

$$\Delta_{i,j} = \min_t \sqrt{(x_{1+(j-1)3} - y_1^i)^2 + (x_{2+(j-1)3} - y_2^i)^2}, \tag{60}$$

$r_0 = 2$, $k_g = 4$, $\varepsilon_i = 0.1$, $i = 1,2,3,4$, $y_1^1 = 4$, $y_2^1 = 2$, $y_1^2 = 6$, $y_2^2 = 4$, $y_1^3 = 4$, $y_2^3 = 6$, $y_1^4 = 6$, $y_2^4 = 8$, $p_1 = 4$, $p_2 = 3$, $p_3 = 4$, $\varepsilon_0 = 0.01$, $t^+ = 4.8$.

In the first stage, according to the proposed approach, the control synthesis problem is solved in order to provide the existence of the equilibrium point.

The stabilization system was received by the network operator method. As far as the received expressions of the control function, both the encoded and decoded forms are too long, so we place them into the Appendix B.

In the second stage, it is necessary to solve the optimal control problem and to find the control function in the form of piece-wise constant control function

$$\mathbf{x}^* = \mathbf{x}^{*,i},\ (i-1)\Delta t \le t < i\Delta t, \tag{61}$$

where $i = 1, \ldots, K$, Δt is a time interval, $\Delta t = 0.6$, and K is the number of time intervals

$$K = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor = \left\lfloor \frac{4.8}{0.6} \right\rfloor = 8. \tag{62}$$

To solve the optimal control problem and find $\mathbf{x}^{*,i}$, $i = 1, \ldots, 8$, the described hybrid algorithm was used. The following optimal solution was found:

$$\begin{array}{rcl}
\mathbf{x}^{*,1} & = & [5.4629\ 0.8503\ -0.0834\ 5.5730\ 7.4134\ -0.2191]^T, \\
\mathbf{x}^{*,2} & = & [8.0879\ 4.6283\ -0.1367\ 2.7272\ 4.3233\ -04311]^T, \\
\mathbf{x}^{*,3} & = & [6.6929\ 4.2258\ -1.4392\ 1.1911\ 7.7361\ 0.2100]^T, \\
\mathbf{x}^{*,4} & = & [1.8651\ 7.0765\ -0.0173\ 6.6029\ 2.6080\ 0.1310]^T, \\
\mathbf{x}^{*,5} & = & [3.6284\ 4.0688\ 0.3204\ 0.7814\ 9.4491\ -0.1612]^T, \\
\mathbf{x}^{*,6} & = & [8.4951\ 8.4002\ 0.3134\ 1.0557\ 0.7920\ 0.7306]^T, \\
\mathbf{x}^{*,7} & = & [7.7752\ 9.9316\ -0.0237\ 1.6134\ 1.9251\ 0.0495]^T, \\
\mathbf{x}^{*,8} & = & [10.0465\ 9.8035\ 0.1303\ -1.0000\ 0.0051\ 0.2369]^T.
\end{array} \tag{63}$$

Optimal trajectories on horizontal plane for robots are presented in Figure 1.

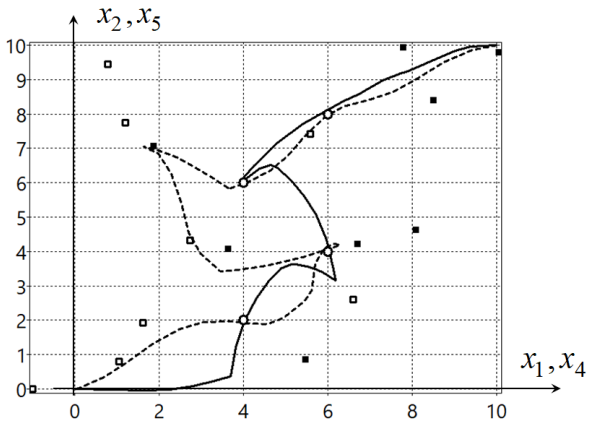

Figure 1. Optimal trajectories of robots for synthesized control

In the Figure 1 the solid black line is a trajectory of the first robot, the dash line is a trajectory of the second robot, the small circles are the bottlenecks, the small black squares are the optimal control points (63) for the first robot, and the small white squares are the optimal control points (63) for the second robot. The optimal value of the functional (56) is 4.8347.

For comparison, the optimal control problem (52)–(60) was solved by the direct approach of optimal control. For this purpose, the time axis was divided on $\tilde{K}$ intervals. The control function is found in the form of the piece-wise linear function

$$u_j = \begin{cases} u_j^+ = 10, \text{ if } \hat{u}_i > u_i^+ \\ u_j^- = -10, \text{ if } \hat{u}_i < u_i^- \\ \hat{u}_j, \text{ otherwise} \end{cases}, j = 1, 2, 3, 4, \tag{64}$$

where

$$\hat{u}_j = (q_{i+(j-1)\tilde{K}+1} - q_{i+(j-1)\tilde{K}}) \frac{t - (i-1)\tilde{\Delta t}}{\tilde{\Delta t}} + q_{i+(j-1)\tilde{K}} \tag{65}$$

where $j = 1, 2, 3, 4$, $j = 1, \ldots, \tilde{K}$, $\tilde{\Delta}t$ is a time interval, $\tilde{\Delta}t = 0.2$,

$$\tilde{K} = \left\lfloor \frac{t^+}{\tilde{\Delta}t} \right\rfloor = \left\lfloor \frac{4.8}{0.2} \right\rfloor = 24. \tag{66}$$

In total, it is necessary to find 96 parameters, $\mathbf{q} = [q_1 \ldots q_{96}]^T$. The problem was very difficult for many evolutionary algorithms. The most successful in solving this problem was the described hybrid evolutionary algorithm.

In the result, the following solution was obtained:

$$\begin{aligned} \mathbf{q} = [&3.6515\ -5.6155\ -5.8103\ -4.1722\ -3.1398\ -5.4711\ 0.0936 \\ &6.6150\ 2.6432\ -8.8133\ -2.4498\ -18.5059\ -9.5896\ 0.1931 \\ &-5.5797\ -14.2516\ 9.5304\ 0.2181\ 0.6002\ -11.9435\ -12.2196 \\ &-0.0127\ -19.4712\ 8.8589\ 5.5695\ 8.4877\ 5.6459\ 0.7054\ -3.6582 \\ &-8.0966\ -0.6840\ -8.6774\ 7.7892\ -5.6366\ -5.3715\ -4.8317 \\ &-16.6047\ -19.8104\ -14.9474\ 7.6756\ 4.7000\ 9.7919\ -14.6483 \\ &-3.5860\ -3.1178\ -9.7188\ -16.2048\ -15.9328\ -1.3150\ 1.9570 \\ &-10.2673\ -0.5094\ -6.4163\ -4.9303\ -3.7649\ -6.3955 \\ &-5.8384\ -15.8273\ -9.2860\ -0.1217\ 9.0490\ -3.0543\ 0.8906 \\ &7.6340\ 10.8459\ 10.2492\ 3.4207\ -10.6311\ -4.9477\ -3.4041 \\ &-13.6140\ -15.2029\ 4.8782\ -10.4763\ -10.5894\ 6.4966\ -4.2872 \\ &-12.7573\ -8.2174\ -0.8267\ -14.1822\ -1.6810\ -15.3973\ 12.1957 \\ &15.4694\ 10.3573\ -12.7840\ 7.9684\ -6.3937\ 17.4171\ -6.6234\ 1.3378 \\ &-8.2870\ -0.2343\ -18.0791\ -5.3433]^T. \end{aligned} \tag{67}$$

The functional value is 4.8132. The optimal trajectories on the horizontal plane are presented in Figure 2.

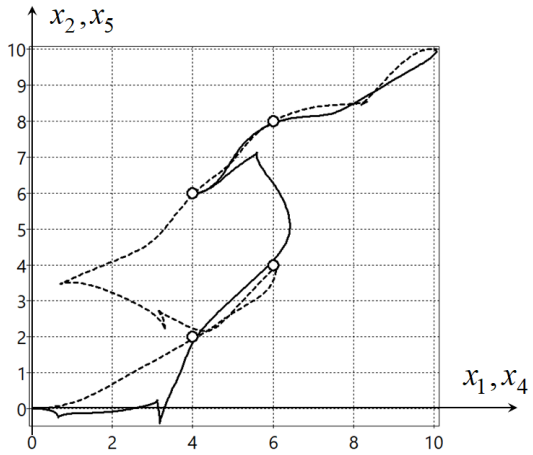

Figure 2. Optimal trajectories of robots for direct control.

To analyze the received results, these optimal control functions were tested for the mathematical model with perturbations:

$$\begin{aligned} \dot{x}_1 &= 0.5(u_1 + u_2)\cos(x_3) + \beta\xi(t), \\ \dot{x}_2 &= 0.5(u_1 + u_2)\sin(x_3) + \beta\xi(t), \\ \dot{x}_3 &= 0.5(u_1 - u_2) + \beta\xi(t), \\ \dot{x}_4 &= 0.5(u_3 + u_4)\cos(x_6) + \beta\xi(t), \\ \dot{x}_5 &= 0.5(u_3 + u_4)\sin(x_6) + \beta\xi(t), \\ \dot{x}_6 &= 0.5(u_3 - u_4) + \beta\xi(t), \end{aligned} \tag{68}$$

where $\xi(t)$ is a function that returns a random number from diapason $(-1;\ 1)$ at every call, and β is a constant value.

For the system (68), disturbances were also introduced into the initial conditions

$$x_(0) = x_i^0 + \beta_0 \xi, \; i = 1, \ldots, 6, \tag{69}$$

where β_0 is a constant value. The results of the tests are presented in Table 1. For every disturbance, 10 tests were performed. In Table 1, J_3 is an average functional value for synthesized control, $\sigma(J_3)$ is a standard deviation for values J_3, J_4 is an average functional value for direct control, and $\sigma(J_4)$ is a standard deviation for values J_4.

Table 1. Functional values for perturbed control object model.

β_0	β	J_3	$\sigma(J_3)$	J_4	$\sigma(J_4)$
0	0	4.8347	0	4.8132	0
0	0.01	4.9999	0.1989	4.9267	0.1462
0	0.02	5.0868	0.288	5.1706	0.2559
0	0.05	5.3896	0.2087	6.1610	1.0704
0.01	0	5.2286	0.2053	6.5891	0.9445
0.02	0	5.4569	0.266	7.2853	1.6878
0.05	0	5.7369	0.6871	13.4286	3.5257
0.01	0.01	5.2365	0.2861	6.4381	0.9502
0.1	0	6.2945	0.7365	19.8192	8.0565

As the test results show, synthesized control is much less susceptible to perturbations of the mathematical model and the initial conditions than direct control. Direct optimal control is the most sensitive to disturbances of initial conditions. Even the smallest disturbances of the initial conditions make direct control unacceptable. The results show that the synthesized control obtains the compression property, and it is feasible in real systems.

7.2. Synthesized Control for Omni-Mecanum-Wheeled Robot

A mecanum robot has special wheels that allow it to move under a direct angle to its direction of axis without any turns [51]. In Figure 3, an example of a mecanum robot is shown.

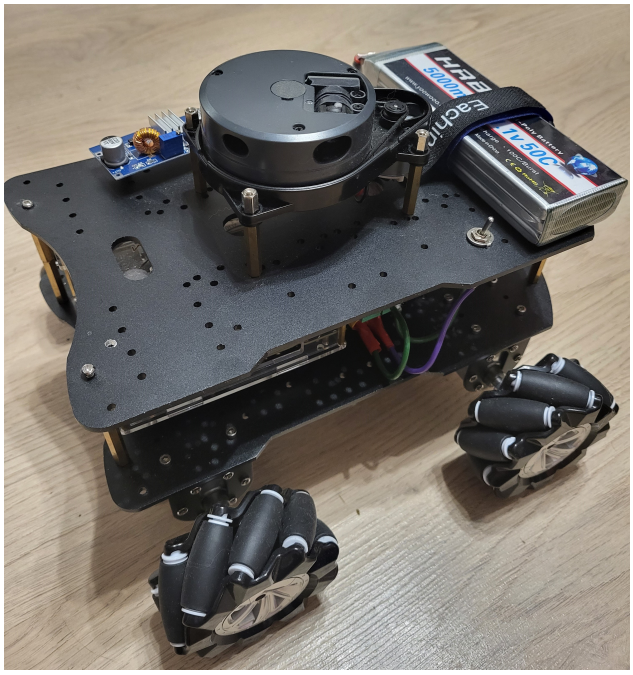

Figure 3. Omni-mecanum-wheeled robot.

Consider the optimal control problem, where two identical mecanum robots have to swap their places in some area with obstacles and without collisions for a minimal amount of time.

The mathematical model of the control object is the following:

$$\begin{array}{rcl}
\dot{x}_1 & = & 0.25((u_1+u_4)(\cos(x_3)+\sin(x_3))+(u_2+u_3)(\cos(x_3)-\sin(x_3))), \\
\dot{x}_2 & = & 0.25((u_1+u_4)(\sin(x_3)-\cos(x_3))+(u_2+u_3)(\cos(x_3)+\sin(x_3))), \\
\dot{x}_3 & = & 0.25(-u_1+u_2-u_3+u_4)/(L_0+H_0), \\
\dot{x}_4 & = & 0.25((u_5+u_8)(\cos(x_6)+\sin(x_6))+(u_6+u_7)(\cos(x_6)-\sin(x_6))), \\
\dot{x}_5 & = & 0.25((u_5+u_8)(\sin(x_6)-\cos(x_6))+(u_6+u_7)(\cos(x_6)+\sin(x_6))), \\
\dot{x}_6 & = & 0.25(-u_5+u_6-u_7+u_8)/(L_0+H_0),
\end{array} \tag{70}$$

where x_1, x_2, and x_3 are the state vector coordinates of the first mecanum robot, x_4, x_5, and x_6 are the state vector coordinates of the second mecanum robot, u_1, u_2, u_3, and u_4 are components of the control vector of the first mecanum robot, u_5, u_6, u_7, and u_8 are components of the control vector of the first mecanum robot, and L_0 and H_0 are geometric parameters of the robots, $L_0 = 2$, $H_0 = 1$.

The control is restricted:

$$-10 = u_i^- \le u_i \le u_i^+ = 10,\ i = 1,\dots,8, \tag{71}$$

where u_i^- and u_i^+ are given lower and upper limits for values of control, respectively, $i = 1,\dots,8$.

The initial state is given:

$$\mathbf{x}(0) = \mathbf{x}^0 = [0\ 0\ 0\ 10\ 10\ 0]^T. \tag{72}$$

The terminal state is given:

$$\mathbf{x}(t_f) = \mathbf{x}^f = [10\ 10\ 0\ 0\ 0\ 0]^T, \tag{73}$$

where t_f is the time of achievement of the terminal state. It is determined by Equation (57) with $\varepsilon_0 = 0.05$, $t^+ = 1.9$.

The quality criterion includes phase constraints and the accuracy of the terminal state achievement.

$$J = p_1\|\mathbf{x}^f - \mathbf{x}(t_f)\| + \sum_{i=1}^{K} w_i \int_0^{t_f} \vartheta(\phi_i(\mathbf{x}))dt + p_2 \int_0^{t_f} \vartheta(\chi(\mathbf{x}))dt + t_f \to \min_{\mathbf{u}}, \tag{74}$$

where p_1 and p_1 are the penalty coefficient, where $p_1 = 3$ and $p_2 = 3$, w_i is a weight coefficient, $i = 1,\dots,K$, $K = 8$, $\vartheta(\alpha)$ is a Heaviside step function (59), and $\chi(\mathbf{x})$ is determined by Equation (58):

$$\phi_i(\mathbf{x}) = r_i - \sqrt{(x_1) - x_{1,i})^2 + (x_2 - x_{2,i})^2},\ i = 1,2,3,4, \tag{75}$$

$$\phi_i(\mathbf{x}) = r_{i-4} - \sqrt{(x_4) - x_{1,i-4})^2 + (x_5 - x_{2,i-4})^2},\ i = 5,6,7,8, \tag{76}$$

$r_1 = 2$, $r_2 = 2.5$, $r_3 = 2.5$, $r_4 = 2$, $x_{1,1} = 2$, $x_{2,1} = 2$, $x_{1,2} = 8$, $x_{2,2} = 2$, $x_{1,3} = 2$, $x_{2,3} = 8$, $x_{1,4} = 8$, $x_{2,4} = 8$, $r_0 = 1$.

According to the synthesized method, initially, the feedback control synthesis problem is solved, such that the closed-loop control system is stable relative to some equilibrium point in the state space. For this purpose, again, the network operator method is used.

Since the robots are the same, we make the synthesis of the stabilization system for one robot, for example, the first robot.

In the result, the network operator method found the following control function:

$$u_i = \begin{cases} u_i^+, \text{ if } \tilde{u}_i \geq u_i^+ \\ u_i^-, \text{ if } \tilde{u}_i \leq u_i^- \\ \tilde{u}_i \text{otherwise} \end{cases}, \tag{77}$$

where

$$\tilde{u}_1 = D + \arctan(q_1\Delta_1), \tag{78}$$

$$\begin{aligned} \tilde{u}_2 &= \rho_{19}(\tilde{u}_1) + \rho_4(C) + \rho_{17}(B) + \operatorname{sgn}(q_2\Delta_2 + \\ &\quad q_1\Delta_1 + q_3\sqrt[3]{\Delta_3}) + \rho_{18}(q_2\Delta_2) + q_1\Delta_1 - (q_1\Delta_1)^3, \end{aligned} \tag{79}$$

$$\tilde{u}_3 = \tilde{u}_2 + C^3 + A + \arctan(q_1) - (A + \arctan(q_1))^3 + \Delta_2^{-1}, \tag{80}$$

$$\begin{aligned} \tilde{u}_4 &= \sin(\tilde{u}_3) + \rho_{16}(\tilde{u}_2) + (C + \sqrt[3]{\Delta_1})^3 + B + \\ &\quad \vartheta(q_2\Delta_2 + q_1\Delta_1) + q_3\sqrt[3]{\Delta_3}, \end{aligned} \tag{81}$$

$$A = \operatorname{sgn}(q_2\Delta_2 + q_1\Delta_1 + q_3\sqrt[3]{\Delta_3}) + \vartheta(q_2\Delta_2 + q_1\Delta_1),$$

$$B = A + \arctan(q_1) + \operatorname{sgn}(q_2\Delta_2 + q_1\Delta_1) + \arctan(q_1\Delta_1),$$

$$C = \cos(B) + \rho_4(q_2\Delta_2 + q_1\Delta_1) + \rho_{16}(\Delta_1) + \rho_{19}(q_1\Delta_1 + q_2\Delta_2) + \exp(q_1\Delta_1),$$

$$D = C + \sqrt[3]{\Delta_1} + C^3 + \rho_B - A - \arctan(q_1) + \rho_{16}(q_1\Delta_1),$$

$\Delta_1 = x_1^* - x_1$, $\Delta_2 = x_2^* - x_2$, $\Delta_3 = x_3^* - x_3$, $q_1 = 11.89282$, $q_2 = 10.15381$, $q_3 = 15.25903$,

$$\rho_4(\alpha) = \operatorname{sgn}(\alpha)\sqrt{|\alpha|},\ \rho_{16}(\alpha) = \begin{cases} \alpha, \text{ if } |\alpha| < 1 \\ \operatorname{sgn}(\alpha), \text{ otherwise} \end{cases},$$

$$\rho_{17}(\alpha) = \operatorname{sgn}(\alpha)\ln(|\alpha| + 1),\ \rho_{18}(\alpha) = \operatorname{sgn}(\alpha)(\exp(|\alpha|) - 1),$$

$$\rho_{19}(\alpha) = \operatorname{sgn}(\alpha)\exp(-|\alpha|).$$

For the second robot in the control function (77), it is necessary to replace x_i, $i = 1, 2, 3$ with x_i, $i = 4, 5, 6$, and x_i^*, $i = 1, 2, 3$, with x_i^* $i = 4, 5, 6$, respectively.

Plots of the trajectories of the robot movement from four initial states are presented in Figure 4.

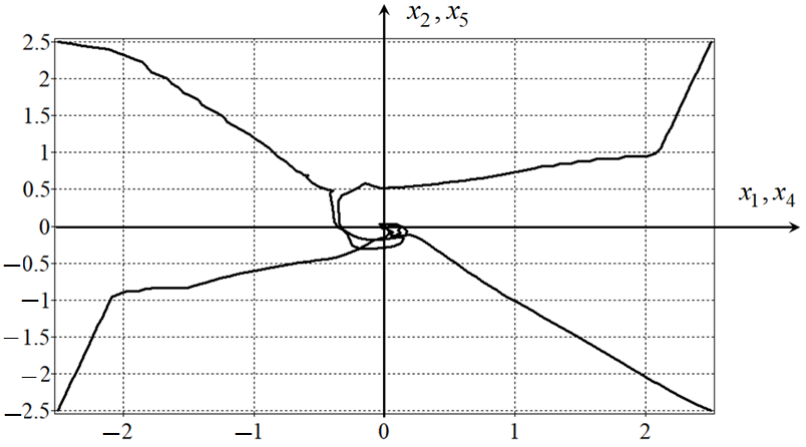

Figure 4. Trajectories on the horizontal plane for four initial states.

On the second stage, the optimal control problem (70)–(74) is solved for the closed loop control system with the control function (77). A control on the second stage is vector $\mathbf{x}^*$, determining the position of the stable equilibrium point in the state space. For solving the optimal control problem, the time axis is divided on the intervals, and in each interval, a control function is approximated by a piece-wise constant function. The value of the interval is equal to $\Delta t = 0.19$.

$$x_i^* = x_i(t_j) = q_{i+(j-1)6}, \tag{82}$$

where $i = 1, \ldots, 6$, $t_j = (j-1)\Delta t$ $j = 1, \ldots, D$, D is the number of intervals,

$$D = \frac{t^+}{\Delta t} = \frac{1.9}{0.19} = 10. \tag{83}$$

In total, it was necessary to find an optimal vector with $10 \cdot 6 = 60$ parameters:

$$\mathbf{q} = [q_1 \ldots q_{60}]^T.$$

For solving the problem, the hybrid evolutionary algorithm was applied. The values of the parameters were restricted:

$$\begin{array}{r} -2.5 = q_1^+ \le q_{1+3(k-1)} \le q_1^- = 2.5 \\ -2.5 = q_2^+ \le q_{2+3(k-1)} \le q_2^- = 2.5 \\ -5\pi/12 = q_2^+ \le q_{3+3(k-1)} \le q_2^- = 5\pi/12 \end{array}, \tag{84}$$

where $k = 1, \ldots, 20$.

The following solution was obtained:

$$\begin{array}{c} \mathbf{q} = [-2.4862\ 2.0000\ 0.6231\ -2.1975\ -1.3799\ 0.7058\ -2.5000\ 1.9421 \\ 1.2094\ -0.6933\ -2.0088\ -0.3378\ 0.1956\ 1.7643\ 0.6378\ 1.4052 \\ -2.4611\ -0.3230\ 1.7245\ 2.0000\ 0.8872\ 1.6769\ -1.2832\ -0.4372 \\ 0.5624\ 1.9987\ -1.1601\ 1.9452\ -1.8859\ 0.7357\ 1.6876\ 1.5024 \\ 0.0997\ 1.1642\ -1.3678\ 0.5321\ 1.6945\ 1.9946\ 0.5580\ 0.7886 \\ -1.6027\ 1.3090\ 1.1795\ 1.5385\ -1.0798\ 0.6781\ -1.9975\ 0.0204 \\ -0.8939\ 1.0578\ -0.4106\ -2.1003\ -1.2544\ -1.3090\ -2.5000\ 1.0746 \\ -1.2216\ -2.0840\ -1.3344\ -0.8913]^T. \end{array} \tag{85}$$

The optimal value of the functional (74) is $J = 1.923$.

Optimal trajectories on the horizontal plane of two robots are presented in Figure 5. The solid line is the trajectory of the first robot, the dash line is the trajectory of the second robot, the red circles are the phase constraints, the black small squares are projections of control $\mathbf{x}^*$ on the horizontal plane for the first robot, and the white small squares are projections of control $\mathbf{x}^*$ on the horizontal plane for the second robot.

Figure 5 shows very clearly, as in the previous example with bottlenecks, that the equilibrium points are located not on the trajectory of the robot, as is done with conventional stabilization, but outside the trajectory. By placing points on the trajectory, we can lose quality because when approaching the equilibrium point, the robot must slow down. Such an optimal arrangement of points ensures the optimal value of the functional.

In this example, we would like to note the following. It occurs that two components of the control vector are enough to control the mecanum robot. The other two components have limit values and do not change during the control process. This can be seen in the additional control plots presented in Appendix C. However, indeed, we noted that in the mathematical model of the mecanum robot (70), the control was redundant, $m > n$. So, the computer itself found a solution for how to proceed in this case, and in the solution found, the two components of control, in fact, do not participate in the search for the

optimal solution. This is one of the more successful demonstrations of intelligent machine automation of the process of creating control systems.

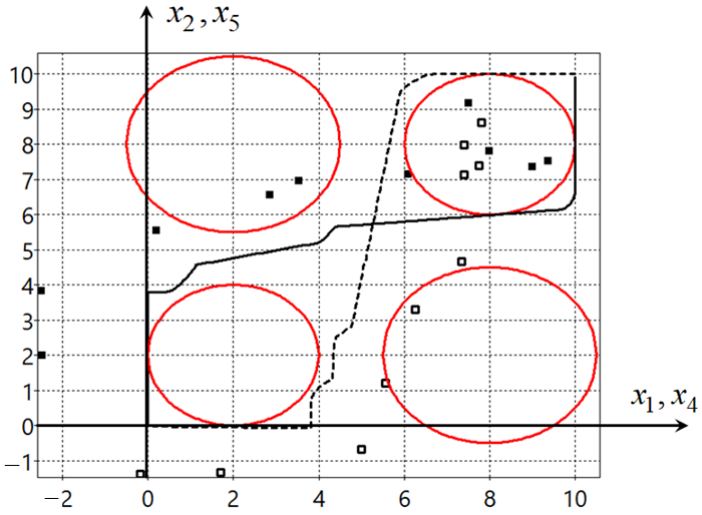

Figure 5. Optimal trajectories on the horizontal plane.

8. Discussion

In this work, we are laying the theoretical foundations of machine learning control. The main feature is that the machine proof of various properties is implemented experimentally basing on examples. In particular, this is what happens in neural networks, when experiments are carried out on a test sample to check whether the system achieves certain properties. We formulate several machine properties in the control system design. In the future, the proposed formal descriptions can be developed, new properties of systems can be considered, and quantitative probabilistic estimates can be given based on positive test results.

An important result of the work is the expansion of the formulation of the optimal control problem and the introduction of additional requirements for the required control. Ensuring the introduced conditions additionally requires the introduction of feedback. The paper presents the approach of synthesized optimal control, which allows implementing optimal control systems, taking into account the introduced additional requirement. In this case, other approaches can be considered and proposed.

9. Conclusions

A general machine learning approach for the automatic design of feedback control systems for any dynamical nonlinear control objects is considered. The main perspective is the machine-automated development of control systems. According to this trend, the control system is obtained as a result of a machine solution of some formal mathematical control problem and it is to be implemented in the real object directly.

Since the control system is created by machine learning, the paper formulates a machine check of all the properties required from the control system. Mathematical statements of control problems and some theoretical justifications for solution of these problems by machine methods are presented. The paper introduces and discusses such notions as machine learning control, stability, optimality and feasibility of machine-made control systems. In this regard, substantiations are introduced for the machine learning feedback control approach based on symbolic regression and evolutionary algorithms.

Thus, the feedback control design is generalized and automated with a generic approach applicable to any nonlinear models, including machine-learning-identified models. It is shown that with this approach, the computer is able to propose interesting outstanding solutions, which sometimes an engineer cannot even suppose.

There is another feature in the proposed approach that can be developed in a good direction. It is possible to control an object by controlling the position of the equilibrium point both offline under predetermined operating conditions, and online, when the positions of the points can be optimally planned for some short term and then adjusted according to the situation. This is one more direction for further research and application of the presented approach.

Author Contributions: Conceptualization, A.D. and E.S.; methodology, A.D. and E.S.; software, A.D. and E.S.; validation, A.D. and E.S.; formal analysis, A.D.; investigation, E.S.; writing—original draft preparation, A.D. and E.S.; writing—review and editing, E.S.; supervision, A.D. 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.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

HJB	Hamilton–Jacobi–Bellman Equation
PID	Proportional–Integral–Derivative Controller
ML	Machine Learning
SR	Symbolic Regression
GA	Genetic Algorithm
GWO	Grey Wolf Optimizer
PSO	Particle Swarm Optimization

Appendix A. Hybrid Evolutionary Algorithm

During the experiments, we noticed that evolutionary algorithms work faster if they use as much information as possible about the goal function values in the space of search at the evolutionary transformation of each possible solution. The PSO algorithm at the construction of new possible solutions uses information about the current best possible solution, about the best possible solution among random selected informants, and about previous goal function values for each possible solution. The GWO algorithm uses information about some current best possible solutions. Therefore, these algorithms work well.

However, when the goal function has a complex form, or it includes many complex constraints, then these algorithms stop at some points of the local minimum. The GA in these cases begins to work better than other evolutionary algorithms. The GA often can shift the search from the current local minimum.

We propose a hybrid algorithm that includes all three listed above algorithms.

Initially, all arrays are created, and the type of evolutionary transformation is randomly chosen: GA, GWO or PSO. In each generation, the number of transformations of every algorithms is approximately the same. A pseudocode of the algorithm can be found in the Algorithms A1–A4. We proposed that the hybrid algorithm worked better than GA, PSO and GWO individually.

Here is the description of the proposed hybrid algorithm.

In the description of the algorithm, a function $Goal(\mathbf{q})$ returns the goal function value for the vector of parameters $\mathbf{q}$. Procedure $NumbertoGray(a, \mathbf{y})$ converts the real number a to Gray code $\mathbf{y}$. Procedure $GraytoNumber(\mathbf{y}, a)$, on the contrary, converts Gray code to a real number a. Procedure $Sort(I, F, k)$ sorts the first k elements in array F and sets the first indexes in the array of index I.

Algorithm A1 Hybrid algorithm.

Require: $H > 0$ is a number of possible solutions in the initial population, $G > 0$ is the number of generations, $R > 0$ is the number of evolutionary changes in one generation, p is the number of searched parameters, $q_i^+ > q_i^-$, $i = 1, \ldots, p$, are restrictions on the parameters, c is the number bits in the integer part of the parameter, d is the number of bits in the fractional part of the parameter, α, β, γ, σ, k_0 are parameters for the PSO algorithm, and k_w is the number of leaders for the GWO algorithm.

Ensure: $\tilde{\mathbf{q}} = [\tilde{q}_1 \ldots \tilde{q}_p]^T$ is the optimal vector of parameters

$q_j^0 = \tilde{q}_j \; j = 1, \ldots, p,$
$q_j^i \leftarrow (q_j^+ - q_j^-)\xi + q_j^-, j = 1, \ldots, p, i = 1, \ldots, H-1$
$v_j^i \leftarrow 0, j = 1, \ldots, p, i = 0, \ldots, H-1$
$F_j \leftarrow Goal(\mathbf{q}^j), j = 0, \ldots, H-1$
$I_j = j, \tilde{F}_j = F_j, j = 0, \ldots, H-1$
$t \leftarrow 0$
while $t < G$ **do**
  $j_- \leftarrow 0, F_- \leftarrow F_0, j \leftarrow 1$
  while $j < H$ **do**
    if $F_j < F_{j_-}$ **then**
      $j_- \leftarrow j,$
      $F_{j_-} \leftarrow F_j$
    end if
    $j \leftarrow j + 1$
  end while
  $Sort(I, \tilde{F}, k_w)$
  $s \leftarrow 0$
  while $s < R$ **do**
    $k_a \leftarrow \xi(3)$
    if $k_a = 0$ **then**
      GWO transformations
    end if
    if $k_a = 1$ **then**
      GA transformation
    end if
    if $k_a = 2$ **then**
      PSO transformation
    end if
    $s \leftarrow s + 1$
  end while
  $t \leftarrow t + 1$
end while
$j_- \leftarrow 0, F_- \leftarrow F_0, j \leftarrow 1$
while $j < H$ **do**
  if $F_j < F_{j_-}$ **then**
    $j_- \leftarrow j,$
    $F_{j_-} \leftarrow F_j$
  end if
  $j \leftarrow j + 1$
end while
$\tilde{\mathbf{q}} \leftarrow \mathbf{q}^{j_-}$

Algorithm A2 GWO transformation.

$L \leftarrow 2 - t(2/G)$
$i \leftarrow \xi(H)$
$j \leftarrow 0$
while $j < p$ **do**
  $\alpha_x \leftarrow 0$
  $k \leftarrow 0$
  while $k < k_w$ **do**
    $g_A \leftarrow 2L\xi - L$
    $g_C \leftarrow 2\xi$
    $\alpha_D \leftarrow |g_C q_j^{I_k} - q_j^i|$
    $\alpha_x \leftarrow \alpha_x + q_j^{I_k} - g_A \alpha_D$
    $k \leftarrow k + 1$
  end while
  $\hat{q}_j \leftarrow \alpha_x / k_w$
  if $\hat{q}_j > q_j^+$ **then**
    $\hat{q}_j \leftarrow q_j^+$
  end if
  if $\hat{q}_j < q_j^-$ **then**
    $\hat{q}_j \leftarrow q_j^-$
  end if
  $j \leftarrow j + 1$
end while
$\hat{F} \leftarrow Goal(\hat{\mathbf{q}})$
if $\hat{F} < F_i$ **then**
  $F_i \leftarrow \hat{F}$
  $\mathbf{q}^i \leftarrow \hat{\mathbf{q}}$
  $Sort(I, F, k_w)$
end if

Algorithm A3 GA transformation.

$k_1 \leftarrow \xi(H), k_2 \leftarrow \xi(H)$
$d \leftarrow \xi$
if $d > F_{j_-}/F_{k_1}$ or $d > F_{j_-}/F_{k_2}$ **then**
  $k_s \leftarrow \xi(p)$
  $NumbertoGray(q_{k_s}^{k_1}, \mathbf{y}^1)$
  $NumbertoGray(q_{k_s}^{k_2}, \mathbf{y}^2)$
  $k_c \leftarrow \xi(c+d)$
  $j \leftarrow 0$
  while $j < k_c$ **do**
    $Sony_j^1 \leftarrow y_j^1, Sony_j^2 \leftarrow y_j^2, j \leftarrow j+1$
  end while
  $j \leftarrow k_c$
  while $j < c+d$ **do**
    $Sony_j^1 \leftarrow y_j^2, Sony_j^2 \leftarrow y_j^1, j \leftarrow j+1$
  end while
  $i \leftarrow 0$
  while $i < k_s$ **do**
    $Son_i^1 \leftarrow q_i^{k_1}, Son_i^2 \leftarrow q_i^{k_2}, i \leftarrow i+1$
  end while
  $i \leftarrow k_s + 1$
  while $i < p$ **do**
    $Son_i^1 \leftarrow q_i^{k_2}, Son_i^2 \leftarrow q_i^{k_1}, i \leftarrow i+1$
  end while
  $GraytoNumber(Sony^1, Son_{k_s}^1)$
  $GraytoNumber(Sony^2, Son_{k_s}^2)$
  $j \leftarrow 1$
  while $j \leq 2$ **do**
    if $Son_{k_s}^j > q_{k_s}^+$ **then**
      $Son_{k_s}^j \leftarrow q_{k_s}^+$
    else if $Son_{k_s}^j < q_{k_s}^-$ **then**
      $Son_{k_s}^j \leftarrow q_{k_s}^-$
    end if
    $\hat{F} \leftarrow Goal(Son^j)$
    $i_+ \leftarrow 0, i \leftarrow 1$
    while $i < H$ **do**
      if $F_i > F_{i_+}$ **then**
        $i_+ \leftarrow i$
      end if
    end while
    if $\hat{F} < F_{i_+}$ **then**
      $\mathbf{q}^{i_+} \leftarrow \mathbf{Son}^j$
      $F_{i_+} \leftarrow \hat{F}$
    end if
    $j \leftarrow j+1$
  end while
end if

Algorithm A4 PSO transformation.

$j \leftarrow \xi(H)$
$k \leftarrow \xi(H)$
$i \leftarrow 0$
while $i < k_0$ **do**
  $l \leftarrow \xi(H)$
  if $F_l < F_k$ **then**
    $k \leftarrow l$
  end if
end while
$i \leftarrow 0$
while $i < p$ **do**
  $v_i^j \leftarrow \alpha v_i^j + \xi\beta(q_i^k - q_i^j) + \xi\gamma(q_i^{j-} - q_i^j)$
  $\hat{q}_i \leftarrow q_i^j + \sigma v_i^j$
  if $\hat{q}_i > q_i^+$ **then**
    $\hat{q}_i \leftarrow q_i^+$
  else if $\hat{q}_i < q_i^-$ **then**
    $\hat{q}_i \leftarrow q_i^-$
  end if
  $i \leftarrow i + 1$
end while
$\hat{F} \leftarrow Goal(\hat{\mathbf{q}})$
if $\hat{F} < F_j$ **then**
  $F_j \leftarrow \hat{F}$
  $\mathbf{q}^j \leftarrow \hat{\mathbf{q}}$
end if

Appendix B. Stabilization System of the Mobile Robot

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{\Psi}_{1,1} & \mathbf{\Psi}_{1,2} \\ \mathbf{0}_{12\times12} & \mathbf{\Psi}_{2,2} \end{bmatrix} \tag{A1}$$

$$\mathbf{\Psi}_{1,1} = \begin{bmatrix} 0&0&0&0&0&0&1&1&0&0&0&12&1 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&2&0&0&8&0&0&0 \\ 0&0&0&0&0&0&0&2&0&1&19&0&0 \\ 0&0&0&0&0&0&0&0&2&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&1&8&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0 \end{bmatrix} \tag{A2}$$

$$\mathbf{\Psi}_{1,2} = \begin{bmatrix} 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 13 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 19 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 23 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 23 \\ 1 & 10 & 10 & 0 & 0 & 0 & 0 & 23 & 0 & 16 & 0 & 16 \end{bmatrix} \tag{A3}$$

$$\mathbf{\Psi}_{2,2} = \begin{bmatrix} 1 & 1 & 15 & 0 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 \\ 0 & 0 & 0 & 1 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A4}$$

In the matrices, the numbers correspond to the functions with one and two arguments according to [46].

The mathematical expression for the found control function described by these matrices has the following form and parameters:

$$u_{i+(j-1)2} = \begin{cases} u^{+}_{i+(j-1)2} \text{ if } \tilde{u}_{i+(j-1)2} > u^{+}_{i+(j-1)2} \\ u^{-}_{i+(j-1)2} \text{ if } \tilde{u}_{i+(j-1)2} < u^{-}_{i+(j-1)2} \\ \tilde{u}_{i+(j-1)2} \text{ otherwise} \end{cases}, \tag{A5}$$

where $i = 1, 2$, $j = 1, 2$,

$$\tilde{u}_{1+(j-1)2} = \operatorname{sgn}(q_3(x^*_{3+(j-1)3} - x_{3+(j-1)3}))\exp(-|q_3(x^*_{3+(j-1)3} - x_{3+(j-1)3})|) + a^{-1} + \sqrt[3]{a} + \operatorname{sgn}(x^*_{3+(j-1)3} - x_{3+(j-1)3}) + \mu(b), \tag{A6}$$

$$\tilde{u}_{2+(j-1)2} = \tilde{u}_{1+(j-1)2} + \sin(\tilde{u}_{1+(j-1)2}) + \arctan(h) + \mu(b) + c - c^3, \tag{A7}$$

$$a = \tanh(d) + \left(b + \sqrt[3]{x^*_{1+(j-1)3} - x_{1+(j-1)3}}\right)^3 + c + \sin(q_3(x^*_{3+(j-1)3} - x_{3+(j-1)3})),$$

$$b = g + \operatorname{sgn}(\operatorname{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3})q_2(x^*_{2+(j-1)3} - x_{2+(j-1)3}))\times \exp(-|\operatorname{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3})q_2(x^*_{2+(j-1)3} - x_{2+(j-1)3})|) + \sin(x^*_{1+(j-1)3} - x_{1+(j-1)3}) + \tanh(g) + x^*_{1+(j-1)3} - x_{1+(j-1)3},$$

$$c = g + \operatorname{sgn}(\operatorname{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3})q_2(x^*_{2+(j-1)3} - x_{2+(j-1)3}))\times \exp(-|\operatorname{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3})q_2(x^*_{2+(j-1)3} - x_{2+(j-1)3})|) +$$

$$\sin(x^*_{1+(j-1)3} - x_{1+(j-1)3}),$$

$$d = h + c - c^3 + \mathrm{sgn}(q_1(x^*_{1+(j-1)3} - x_{1+(j-1)3})) +$$

$$\arctan(q_1) + \vartheta(x^*_{3+(j-1)3} - x_{3+(j-1)3}),$$

$$g = \mathrm{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3}) q_2(x^*_{2+(j-1)3} - x_{2+(j-1)2}) +$$

$$q_3(x^*_{3+(j-1)3} - x_{3+(j-1)3}) + \tanh(q_1(x^*_{1+(j-1)3} - x_{1+(j-1)3})),$$

$$h = \arctan(q_1(x^*_{1+(j-1)3} - x_{1+(j-1)3})) +$$

$$\mathrm{sgn}(w)\sqrt{|w|} + w + v + 2\mathrm{sgn}(w + \tanh(v)) +$$

$$\sqrt[3]{w + \tanh(v) + \sqrt[3]{x^*_{1+(j-1)3} - x_{1+(j-1)3}}} +$$

$$\mathrm{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3})\sqrt{|x^*_{1+(j-1)3} - x_{1+(j-1)3}|} +$$

$$\sqrt[3]{x^*_1 - x_1} + \tanh(v),$$

$$w = \mathrm{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3}) +$$

$$\mathrm{sgn}(q_2(x^*_{2+(j-1)3} - x_{2+(j-1)3}))\mathrm{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3}) \times$$

$$\tanh(x^*_{1+(j-1)3} - x_{1+(j-1)3}),$$

$$v = q_3(x^*_{3+(j-1)3} - x_{3+(j-1)3}) + \mathrm{sgn}(x^*_{1+(j-1)3} - x_{1+(j-1)3}) q_2 \times$$

$$(x^*_{2+(j-1)3} - x_{2+(j-1)3}) + \tanh(x^*_{1+(j-1)3} - x_{1+(j-1)3}),$$

$$\mu(\alpha) = \mathrm{sgn}(\alpha)\min\{1, |\alpha|\},\ \tanh(\alpha) = \frac{1 - \exp(-2\alpha)}{1 + \exp(-2\alpha)},$$

$j = 1, 2$, $q_1 = 14.7288$, $q_2 = 2.0271$, $q_3 = 4.0222$.

Appendix C. Plots for Mecanum Robot

Figures A1–A6 demonstrate the plots of optimal values of the state-space vector components (black lines) and the corresponding values of the vector $\mathbf{x}^*$ (red lines).

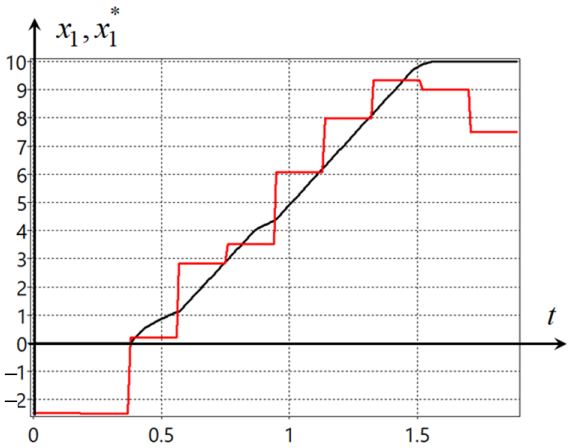

Figure A1. Plots of optimal x_1 and ${x_1}^*$.

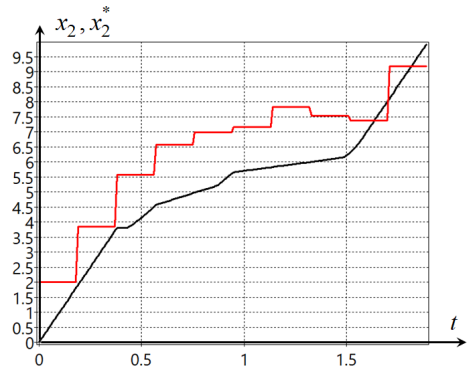

Figure A2. Plots of optimal x_2 and $x_2{}^*$.

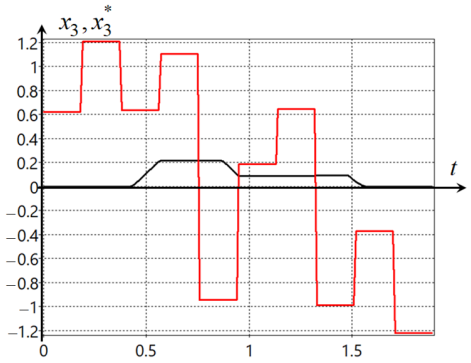

Figure A3. Plots of optimal x_3 and $x_3{}^*$.

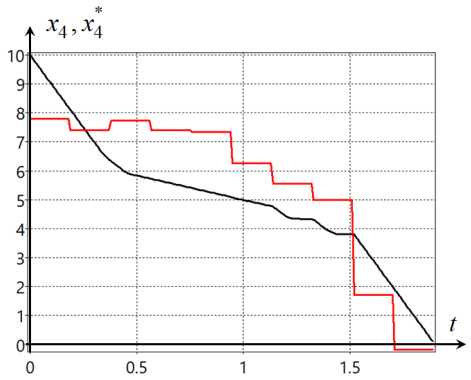

Figure A4. Plots of optimal x_4 and $x_4{}^*$.

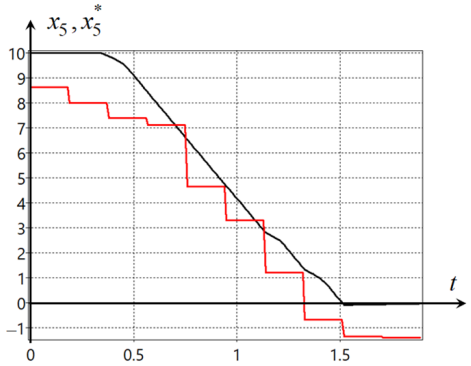

Figure A5. Plots of optimal x_5 and $x_5{}^*$.

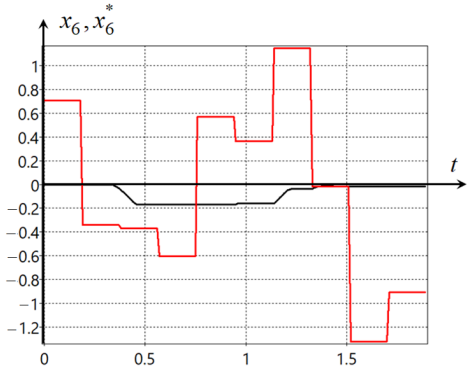

Figure A6. Plots of optimal x_6 and $x_6{}^*$.

In Figures A7–A14, the plots of optimal values of the control vector components are presented.

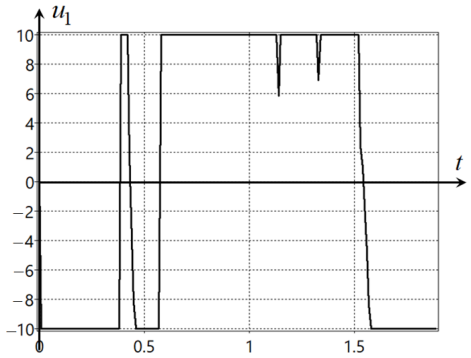

Figure A7. Plot of optimal values of control component u_1.

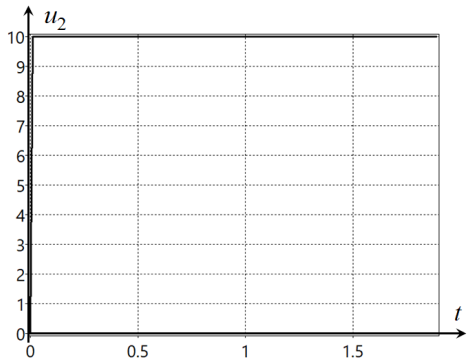

Figure A8. Plot of optimal values of control component u_2.

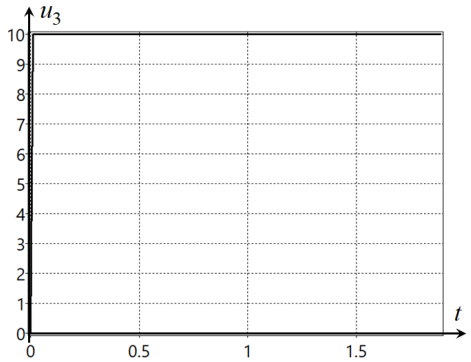

Figure A9. Plot of optimal values of control component u_3.

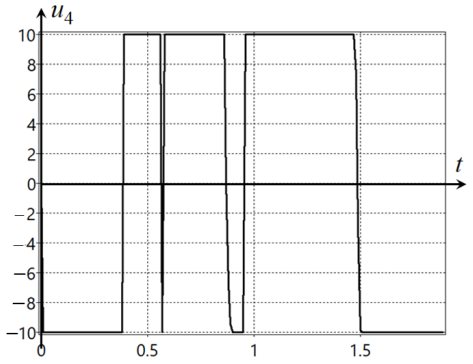

Figure A10. Plot of optimal values of control component u_4.

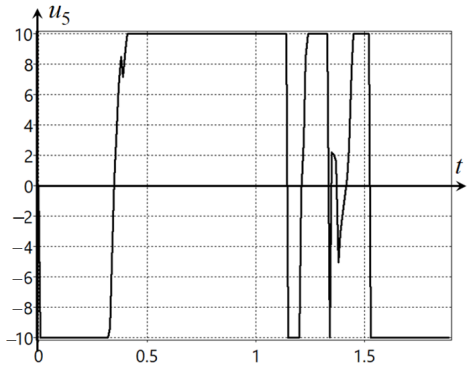

Figure A11. Plot of optimal values of control component u_5.

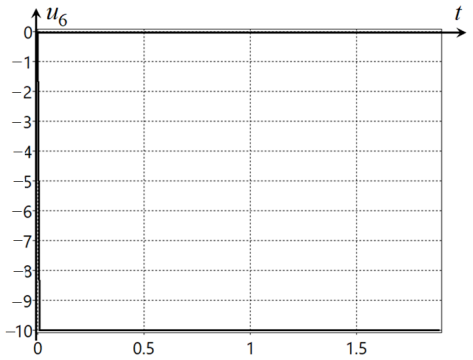

Figure A12. Plot of optimal values of control component u_6.

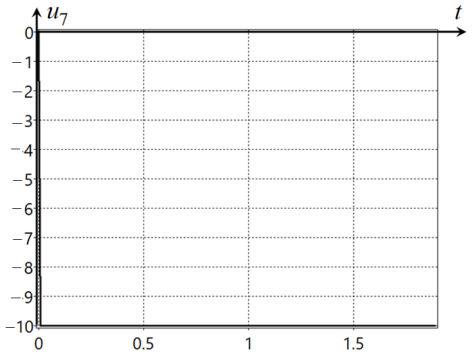

Figure A13. Plot of optimal values of control component u_7.

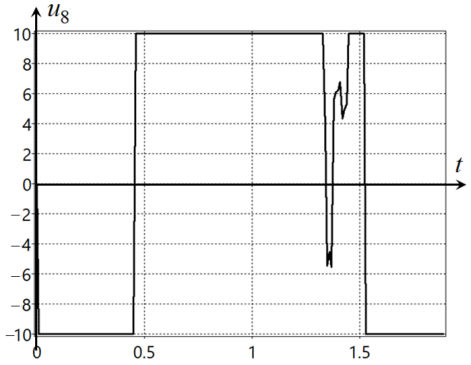

Figure A14. Plot of optimal values of control component u_8.

References

1. Egerstedt, M. Motion Planning and Control of Mobile Robots. Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden, 2000.
2. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37. [CrossRef] [PubMed]
3. Jones, M.; Peet, M.M. A generalization of Bellmans equation with application to path planning, obstacle avoidance and invariant set estimation. *Automatica* **2021**, *127*, 109510. [CrossRef]
4. Aguilar, C.O.; Krener, A.J. Numerical solutions to the Bellman equation of optimal control. *J. Optim. Theory Appl.* **2014**, *160*, 527–552. [CrossRef]
5. Aliyu, M.D.S. An iterative relaxation approach to the solution of the Hamilton-Jacobi-Bellman-Isaacs equation in nonlinear optimal control. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 360–366. [CrossRef]
6. Fraga, S.L.; Pereira, F.L. Hamilton-Jacobi-Bellman Equation and Feedback Synthesis for Impulsive Control. *IEEE Trans. Autom. Control* **2012**, *57*, 244–249. [CrossRef]
7. Liu, D.; Wang, D.; Wang, F.; Li, H.; Yang, X. Neural-Network-Based Online HJB Solution for Optimal Robust Guaranteed Cost Control of Continuous-Time Uncertain Nonlinear Systems. *IEEE Trans. Cybern.* **2014**, *44*, 2834–2847. [CrossRef]
8. Wei, Q.; Liu, D.; Lin, H. Value Iteration Adaptive Dynamic Programming for Optimal Control of Discrete-Time Nonlinear Systems. *IEEE Trans. Cybern.* **2016**, *46*, 840–853. [CrossRef]
9. Liu, D.; Xue, S.; Zhao, B.; Luo, B.; Wei, Q. Adaptive Dynamic Programming for Control: A Survey and Recent Advances. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 142–160. [CrossRef]
10. Lu, J.; Wei, Q.; Wang, F.-Y. Parallel control for optimal tracking via adaptive dynamic programming. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1662–1674. [CrossRef]
11. Lewis, F.L.; Vrabie, D.; Vamvoudakis, K.G. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst.* **2012**, *32*, 76–105.
12. Wen, G.; Chen, C.L.P.; Ge, S.S. Simplified Optimized Backstepping Control for a Class of Nonlinear Strict-Feedback Systems With Unknown Dynamic Functions. *IEEE Trans. Cybern.* **2021**, *51*, 4567–4580. [CrossRef] [PubMed]
13. Kim, J.; Shin, J.; Yang, I. Hamilton-Jacobi Deep Q-Learning for Deterministic Continuous-Time Systems with Lipschitz Continuous Controls. *J. Mach. Learn. Res.* **2021**, *22*, 1–34.
14. Walsh, G.; Tilbury, D.; Sastry, S.; Murray, R.; Laumond, J.P. Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Trans. Autom. Control* **1994**, *39*, 216–222. [CrossRef]
15. Wang, S.; Dai, M.; Wang, Y. Robust Adaptive Backstepping Sliding Mode Control for a Class of Uncertain Nonlinear System. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 3534–3538. [CrossRef]
16. Zhao, X.; Wang, X.; Zhang, S.; Zong, G. Adaptive Neural Backstepping Control Design for A Class of Nonsmooth Nonlinear Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1820–1831. [CrossRef]
17. Tyutikov, V.V.; Panteleev, E.R.; Zhilnikova, Y.F. Analysing Impact of Transfer Function Zeros in Controlled Object on Parametric Sensitivity of Systems Synthesized by Method of Aggregated Controller Analytical Design (ACAD). In Proceedings of the 2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 18–22 May 2020; pp. 1–6. [CrossRef]
18. Clarke, F. Lyapunov Functions and Feedback in Nonlinear Control. In *Optimal Control, Stabilization and Nonsmooth Analysis*; de Queiroz, M.S., Malisoff, M., Wolenski, P., Eds.; LNCIS 301; Springer: Berlin/Heidelberg, Germany, 2004; pp. 267–282.
19. Benzaouia, A.; Hmamed, A.; Mesquine, F.; Benhayoun, M.; Tadeo, F. Stabilization of Continuous-Time Fractional Positive Systems by Using a Lyapunov Function. *IEEE Trans. Autom. Control* **2014**, *59*, 2203–2208. [CrossRef]
20. Simon, J.D.; Mitter, S.K. A theory of modal control. *Inf. Control* **1968**, *13*, 316–353. [CrossRef]
21. Tousi, S.M.A.; Mostafanasab, A.; Teshnehlab, M. Design of Self Tuning PID Controller Based on Competitional PSO. In Proceedings of the 2020 4th Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), Mashhad, Iran, 2–4 September 2020; pp. 22–26. [CrossRef]

22. Cherroun, L.; Nadour, M.; Kouzou, A. Type-1 and Type-2 Fuzzy Logic Controllers for Autonomous Robotic Motion. In Proceedings of the 2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID), Elazig, Turkey, 25–27 September 2019; pp. 1–5. [CrossRef]
23. Ahmed, A.A.; Alshandoli, A.F.S. On replacing a PID controller with Neural Network controller for Segway. In Proceedings of the 2020 International Conference on Electrical Engineering (ICEE), Takamatsu, Japan, 28 June–2 July 2020; pp. 1–4. [CrossRef]
24. Diveev, A.I.; Shmalko, E.Y. Machine-Made Synthesis of Stabilization System by Modified Cartesian Genetic Programming. *IEEE Trans. Cybern.* **2022**, *52*, 6627–6637. [CrossRef]
25. Duriez, T.; Brunton, S.L.; Noack, B.R. *Machine Learning Control—Taming Nonlinear Dynamics and Turbulence*; Springer International Publishing: Cham, Switzerland, 2017.
26. 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, Proceedings of the 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, 11–13 December 2018*; Bramer, M., Petridis, M., Eds.; LNCS; Springer: Cham, Switzerland, 2018.
27. Deisenroth, M.P.; Faisal, A.A.; Ong, C.S. *Mathematics for Machine Learning*; Cambridge University Press: Cambridge, UK, 2020. [CrossRef]
28. Burkov, A. *The Hundred-Page Machine Learning Book*; Andriy Burkov: Quebec City, QC, Canada, 2019; 160p.
29. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2019; 856p.
30. Brunton, S.L.; Proctor, J.L.; Kutz, J.N. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA* **2015**, *113*, 3932–3937. [CrossRef]
31. Shmalko, E.; Rumyantsev, Y.; Baynazarov, R.; Yamshanov, K. Identification of Neural Network Model of Robot to Solve the Optimal Control Problem. *Inform. Autom.* **2021**, *20*, 1254–1278. [CrossRef]
32. Malkin, I.G. *Theory of Motion Stability*; Nauka: Moscow, Russia, 1966.
33. Diveev, A.; Shmalko, E.; Serebrenny, V.; Zentay, P. Fundamentals of Synthesized Optimal Control. *Mathematics* **2021**, *9*, 21. [CrossRef]
34. Sun, S.; Cao, Z.; Zhu, H.; Zhao, J. A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Trans. Cybern.* **2020**, *50*, 3668–3681. [CrossRef] [PubMed]
35. Diveev, A.; Shmalko, E. *Machine Learning Control by Symbolic Regression*; Springer International Publishing: Cham, Swizterland, 2021.
36. Silviu-Marian, U.; Max, T. AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* **2020**, *6*, eaay2631. [CrossRef]
37. Jin, Y.; Fu, W.; Kang, J.; Guo, J.; Guo, J. Bayesian Symbolic Regression. *arXiv* **2019**, arXiv:1910.08892.
38. La Cava, W.; Moore, J.H. Learning feature spaces for regression with genetic programming. *Genet. Program. Evolvable Mach.* **2022**, *21*, 433–467. [CrossRef] [PubMed]
39. Petersen, B.K.; Larma, M.L.; Mundhenk, T.N.; Santiago, C.P.; Kim, S.K.; Kim, J.T. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
40. Derner, E.; Kubalík, J.; Ancona, N.; Babuška, R. Symbolic Regression for Constructing Analytic Models in Reinforcement Learning. *Appl. Soft Comput.* **2020**, *94*, 106432. [CrossRef]
41. Alibekov, E.; Kubalık, J.; Babuska, R. Symbolic Method for Deriving Policy in Reinforcement Learning. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 2789–2795. [CrossRef]
42. Derner, E.; Kubalík, J.; Babuška, R. Reinforcement Learning with Symbolic Input–Output Models. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3004–3009. [CrossRef]
43. Diveev, A.I.; Shmalko, E.Yu. Evolutionary computations for synthesis of control system of group of robots and the optimum choice of trajectories for their movement. In Proceedings of the CEUR Workshop Proceedings: VIII International Conference on Optimization and Applications (OPTIMA-2017), Petrovac, Montenegro, 2–7 October 2017; pp. 158–165.
44. Shmalko, E.; Diveev, A. Control Synthesis as Machine Learning Control by Symbolic Regression Methods. *Appl. Sci.* **2021**, *11*, 5468. [CrossRef]
45. Diveev, A.I. Small Variations of Basic Solution Method for Non-numerical Optimization. *IFAC-PapersOnLine* **2015**, *48*, 28–33. [CrossRef]
46. 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]
47. Diveev, A.I.; Konstantinov, S.V. Study of the Practical Convergence of Evolutionary Algorithms for the Optimal Program Control of a Wheeled Robot. *J. Comput. Syst. Sci. Int.* **2018**, *57*, 561–580. [CrossRef]
48. Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley Professional: Reading, MA, USA, 1989.
49. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]

50. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95–International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948. [CrossRef]
51. Huang, H.-C.; Tao, C.-W.; Chuang, C.-C.; Xu, J.-J. FPGA-Based Mechatronic Design and Real-Time Fuzzy Control with Computational Intelligence Optimization for Omni-Mecanum-Wheeled Autonomous Vehicles. *Electronics* **2019**, *8*, 1328. [CrossRef]